# Rotations in 2D and 3D discrete spaces

Yohan Thibault

**UNIVERSITÉ PARIS-EST**
École Doctorale MSTIC

# Rotations in 2D and 3D discrete spaces

by

THIBAULT Yohan

Supervised by

COUPRIE Michel

KENMOCHI Yukiko

A thesis submitted to obtain the
Doctor in computer science

in the University
Paris-Est
Laboratoire d'informatique Gaspard-Monge

Jury

COUPRIE Michel Professeur

REVEILLÈS Jean-Pierre Professeur Émérite

ANDRÈS Éric Professeur

KENMOCHI Yukiko Chargée de recherche

SUGIMOTO Akihiro Professeur

FUCHS Laurent Maître de conférences

Septembre 22nd 2010

Thesis prepared in

Université PARIS-EST

Équipe A3SI du Laboratoire d'Informatique de l'Institut Gaspard Monge

ESIEE-PARIS 2, Bd Blaise Pascal, Cité Descartes, BP99 Noisy le Grand Cedex

*"Petite pensé. En anglais, il existe le verbe "to rotate", en français son équivalent serait le verbe rotationner, sa définition serait* effectuer ou faire effectuer une rotation. *Cependant le verbe "rotationner" n'existe pas dans la langue française. Et pourtant, il tourne."*

*"The discrete geometry is to classical geometry what the language is to thought, i.e. an imperfect means to represent the reality. It took centuries for the language to evolve in a way almost capable to faithfully describe our though.*

*Today the discrete geometry tries to do the same thing with the continuous geometry. The continuous geometry is a mathematical model that cannot be correctly or exactly reproduced in the real world and in computer science. A simple example is the famous number $\pi$. The theoretical mathematic model supposes an exact value of this number, however, the representation of a circle on the ground with a rope or on a sheet of paper by a compass can only give an approximated value of $\pi$, whatever the diameter of the circle, the size of the rope or the precision of the compass. In computer science, for any approximation of $\pi$ used during computations, results will always be an approximation.*

*Today, one of the biggest challenge in computer science is to find new methods so that computers can represent reality as faithfully as possible. Regarding geometry, we strongly believe that these methods belong to the discrete geometry."*

UNIVERSITÉ PARIS-EST

# *Abstract*

Paris-Est
Laboratoire d'informatique Gaspard-Monge

Doctor of Philosophy

by THIBAULT Yohan

Cette thèse présente une étude sur les rotations dans les espaces discrets en 2 et 3 dimensions. Un espace discret est, par opposition à un espace continu, un espace borné avec un nombre fini de points. En informatique, les espaces continus n'existent pas ; en effet, même l'utilisation de nombres flottants ne permet qu'une approximation grossière du continu. Les données utilisées dans le cadre de l'informatique sont le plus souvent entières. Par exemple, une image numérique n'est composée que de points à coordonnées entières et, pour la couleur, à valeurs entières. De plus l'utilisation des nombres flottants pour approcher le continu pose des problèmes de précision. Pour ces raisons, nous avons choisi durant cette thèse de nous concentrer sur les espaces discrets et de n'utiliser que des entiers durant les calculs.

Dans le domaine de la vision par ordinateur, la rotation est une transformation requise pour de nombreuses applications. Dans la plupart des applications, la rotation utilisée est la rotation euclidienne discrétisée. Les résultats donnés par cette rotation dans les espaces discrets ne sont pas bons car il y a une importante perte d'informations, la qualité visuelle de l'image est dégradée et une partie des propriétés mathématiques de la rotation continue est perdue. Par conséquent avec le développement de l'informatique, le besoin de développer de nouvelles méthodes de rotations adaptées aux espaces discrets s'est fait sentir.

Dans cette thèse, nous nous sommes donc concentrés sur le développement des rotations dans les espaces discrets et sur leur compréhension. Nous nous sommes principalement intéressés aux angles charnières qui représentent la discontinuité de la rotation dans les espaces discrets. En effet, dans ces espaces, effectuer deux rotations d'une image avec deux angles très proches peut donner le même résultat. Cette particularité est capturée par les angles charnières. L'utilisation de ces angles particuliers permet de décrire une rotation qui donne les mêmes résultats que la rotation continue discrétisée sans avoir recours aux calculs flottants. Ces angles permettent aussi de décrire une rotation incrémentale qui décrit toutes les rotations possibles d'une image numérique donnée (chose impossible dans le continu car il y a une infinité de rotations possibles). L'utilisation des angles charnières peut-être étendue dans les espaces discrets en trois dimensions. Pour ce faire, on définit les multi-grilles qui sont des plans de rotations qui contiennent trois ensembles de droites parallèles. Ces droites représentent les discontinuités de la rotation en 3D. Elles servent donc à définir les angles charnières dans les plans de rotations. Les multi-grilles permettent d'obtenir les mêmes résultats en 3D que ceux obtenus en 2D.

Cette thèse s'articule autour de cinq chapitres en plus de l'introduction aux problématiques des espaces discrets et des rotations dans ces espaces présentée dans le chapitre 1. Le chapitre 2 présente un état de l'art des rotations discrètes, d'une présentation des

propriétés de ces rotations et des problèmes liés à l'absence de certaines de ces propriétés. S'ensuit une étude sur les rotations dans des espaces discrets hexagonaux et triangles. Le chapitre 3 introduit les angles charnières qui permettent de définir une rotation discrète dans le plan ainsi qu'une rotation incrémentale qui calcule l'ensemble des rotations possible pour une image donnée. À la fin du chapitre, nous proposons une nouvelle méthode pour estimer la rotation faite entre deux images numériques utilisant les angles charnières. Les chapitres 4 et 5 étendent les résultats présentés dans le chapitre 3 aux rotations en 3D. Le chapitre 6 présente une étude des $n$-tuples pythagoriciens. Ils représentent les triangles rectangles à longueur entière dans toutes les dimensions. Nous montrons qu'ils sont denses sur la sphère unitaire et proposons une méthode approcher n'importe quel vecteur par un $n$-tuples pythagoricien. Cette étude est nécessaire pour obtenir les résultats des chapitres 4 et 5.

**Mots clés**

`Rotation`
`Rotation discrète`
`Géométrie discrètes`
`Angle charnière`
`Multi-grille`

UNIVERSITÉ PARIS-EST

# *Abstract*

Paris-Est
Laboratoire d'informatique Gaspard-Monge

Doctor of Philosophy

by THIBAULT Yohan

This thesis presents a study on rotations in 2- and 3-dimensional discrete spaces. By opposition to the continuous space, a discrete space is a space with bounds and a finite number of points. In computer science, continuous spaces do not exist; indeed, the floating numbers used to simulate the continuous space only give a crude approximation of the continuous space. Data used in the computer science field are mostly composed of integers. For example, a digital image contains only discrete points with integer coordinates and integer values that encode the color of the point. Moreover, using floating numbers to approach real numbers implies precision problems. For these reasons, we decided, during this thesis, to work in discrete spaces and to use only integers for all computations. In the computer vision field, the rotation is a transformation required for many applications. Most of applications use the Euclidean rotation and then apply the rounding function on the obtained results. The results obtained by using this rotation in discrete spaces are not satisfactory since much information is lost, the visual quality is affected and most of the properties of the continuous rotation are lost. Accordingly, with the expansion of computer science, it is necessary to develop new rotation methods adapted to discrete spaces.

In this thesis, we have chosen to study the rotations in discrete spaces and to improve their understanding. We mainly studied hinge angles that represent the discontinuity of the rotation in the discrete space. Indeed, in discrete space, it is possible to perform two rotations of the same digital image with two angles that are slightly different while obtaining the same result. This particularity of the rotation in discrete space is formalized by hinge angles. Using hinge angles allows us to describe a discrete rotation that gives the same results than the discretized Euclidean rotation, although using only integers during computations. These particular angles also allow the description of incremental rotation that performs all possible rotations of a given digital image. Such an incremental rotation is impossible in continuous space since there is an infinity of continuous rotations. Using hinge angles can also be extended to the rotations in 3-dimensional discrete spaces. The extension requires multi-grids, which consist in rotation planes containing three sets of parallel lines. These parallel lines represent the discontinuities of the rotation in 3D discrete space. Thus they are useful to describe hinge angles in rotation planes. Multi-grids allow us to obtain the same results in 3D discrete rotations as the results obtained in 2D discrete rotations.

This thesis contains five chapters given after an introduction in Chapter 1 to the problematic of discrete spaces and rotations in discrete spaces. Chapter 2 gives a state of the art about discrete rotations in the plane, shows the properties of Euclidean rotations that are lost in discrete spaces and explains which problems arise with the lost of properties. The end of this chapter is a study of rotations in the discrete spaces tiled by triangles or hexagons. Chapter 3 introduces the hinge angles that allow the design

of a discrete rotation in the plane and an incremental discrete rotation that computes all possible rotations for a given digital image. Then we propose a method to estimate, for a pair of digital images, the rotation that transforms the first image into the second one. The chapters 4 and 5 are the extension of the results presented in chapter 3 in the 3D discrete space. Chapter 6 presents a study on Pythagorean $n$-tuples. We show that they are dense in any dimensions and we propose a method to find for any $n$-vector a Pythagorean $n$-tuple that approximates this vector. This study is required to obtain the result of Chapters 4 and 5.

**Keywords**

`Rotation`
`Discrete rotation`
`Digital geometry`
`Hinge angle`
`Multi-grid`

# *Acknowledgements*

Pour commencer, j'aimerais remercier Yukiko Kenmochi, ma directrice de thèse dans les faits si ce n'est sur le papier, pour m'avoir encadré durant ma thèse. Pendant plus de trois ans elle a toujours su faire preuve d'une patiente sans limite aussi bien pour corriger mes articles que pour me remettre sur la bonne voie quand mes recherches commençaient à s'égarer. Elle a aussi toujours su trouver les conseils judicieux lorsque lors des moments de doutes. Bien que je fusse son premier doctorant, son encadrement fut d'une qualité remarquable. J'en veux pour preuve qu'elle a réussi à me donner une certaine rigueur scientifique que vingt ans d'éducation dont six ans de faculté n'ont jamais pu m'inculquer.

Je voudrais aussi remercier Akihiro Sugimoto, mon co-directeur de thèse, là encore dans les faits si ce n'est sur le papier, pour m'avoir co-encadré durant ma thèse, pour toute l'aide qu'il m'a apporté pour mes recherches ainsi que pour son incroyable capacité à raturer mes articles. Un grand merci aussi pour m'avoir accueilli dans son laboratoire à Tokyo durant près d'un an et ainsi permis de découvrir le Japon.

Merci à Michel Couprie, mon directeur de thèse, le vrai cette fois, pour l'ensemble de ses conseils. Pour m'avoir donné la chance de réaliser une thèse dans le domaine de l'imagerie malgré mon background plus orienté sur la logique et la complexité (merci aussi à Yukiko pour cette chance).

Merci aux professeurs Jean Pierre Reveillès et Éric Andrès pour m'avoir fait l'honneur d'accepter sans hésitation le rôle de rapporteur. Pour leurs encouragements lors de nos quelques rencontres. Merci aussi à Laurent Fusch pour avoir accepter le rôle d'examinateur.

Merci à toutes les personnes avec lesquelles j'ai travaillé durant ma thèse pour leurs idées, leurs commentaires et plus simplement pour le plaisir que j'ai eu à travailler avec elles.

Merci à Anne pour avoir proposé son aide à la relecture du manuscrit, pour la correction d'un grand nombre de fautes ainsi que pour ses goûts très sur en matière de musique.

Merci à l'ensemble du laboratoire A3SI pour l'excellente ambiance générale et les très bonnes conditions de travails qui ont régnées tout au long de ma thèse. Les repas aux discussions très animées allants de la politique aux sujets scientifiques les plus complexes en passant par les sujets les plus improbables comme le jeu " eternity " ou encore " la fratrie de la petite Sophie ". Un merci particulier aux personnes qui ont partagé avec moi le bureau 5305. À Émilie, pour nous avoir supportée pendant plusieurs années sans jamais se plaindre de nous. À John, pour m'avoir permis de réaliser ma thèse en quatre ans au lieu de trois, je sais d'ailleurs que ce dernier remerciement sera réciproque. Et plus récemment à Mohamed, ce pauvre stagiaire qui a atterri dans notre bureau il y a peu, pour son sens de l'autodérision très développé.

Merci à tous mes amis qui me considèrent encore comme un étudiant.

Et enfin, merci à ma famille pour m'avoir toujours encouragé dans la suite de mes études même quand j'avais des doutes ...

# Contents

**Abstract** iii

**Abstract** vi

**Acknowledgements** ix

**List of Figures** xiv

**1 Introduction** 1
- 1.1 Introduction to the rotation problem in $\mathbb{Z}^2$ . . . 1
- 1.2 A quick history of geometry . . . 2
- 1.3 Digital geometry . . . 3
- 1.4 From digital geometry to discrete rotations . . . 4
- 1.5 Structure of this manuscript . . . 4

**2 Rotations in 2D discrete spaces** 6
- 2.1 Introduction . . . 6
- 2.2 Definitions and Properties . . . 7
- 2.3 Survey on discrete rotations in the plane . . . 12
  - 2.3.1 Rotation by discrete circle . . . 12
  - 2.3.2 Rotation by Pythagorean lines . . . 13
  - 2.3.3 Shear rotation and quasi shear rotation . . . 14
  - 2.3.4 Rotation by hinge angles . . . 14
- 2.4 Rotation in other grids . . . 15
  - 2.4.1 Connectivity on the discrete grids . . . 16
  - 2.4.2 Methodology . . . 17
  - 2.4.3 Experimental result . . . 19
- 2.5 Conclusion . . . 22

**3 2D rotations in a discrete space using hinge angles** 25
- 3.1 Introduction . . . 25
- 3.2 Hinge angles . . . 26
  - 3.2.1 Definition of hinge angles . . . 27
  - 3.2.2 Properties related to Pythagorean angle . . . 28

3.3 Rotation by hinge angles . . . 29
3.3.1 Computing the lower bound rotation angle for grid point . . . 30
3.3.2 Computing the lower bound rotation angle for a set of grid points . 31
3.3.3 Digital image rotation by a lower bound rotation angle . . . 33
3.4 Obtaining admissible rotation angles from two digital images . . . 34
3.4.1 Setting Rotation Centers . . . 34
3.4.2 Computing lower and upper bounds of rotation angles from two corresponding point pairs . . . 35
3.4.3 Incremental computing lower and upper bounds of rotation angles 38
3.5 Experimental Evaluation using Synthetic Data . . . 40
3.6 Discussion on practical application to digital images . . . 41
3.6.1 Synthetic data . . . 41
3.6.2 Real data . . . 43
3.7 Conclusion . . . 45

**4 3D Rotations in discrete space using hinge angles 47**
4.1 Introduction . . . 47
4.2 Hinge angles . . . 48
4.3 Multi-grids . . . 50
4.3.1 Definition of multi-grids . . . 50
4.3.2 Multi-grid line equations . . . 52
4.3.3 Rational multi-grids . . . 54
4.3.4 Properties of rational multi-grids . . . 54
4.4 Hinge angles characterized by a multi-grid . . . 56
4.4.1 Hinge angles and rational multi-grids . . . 57
4.4.2 Quintuplet integer representation of hinge angles . . . 58
4.5 3D discrete rotations around a Pythagorean axis . . . 59
4.5.1 Comparing hinge angles with integer computations . . . 60
4.5.2 Upper bound of the number of 3D hinge angles . . . 62
4.5.3 3D discrete rotations induced by hinge angles . . . 63
4.5.4 3D incremental discrete rotation . . . 64
4.6 Conclusion . . . 65

**5 Estimation of rotation between a pair of 3D digital images 66**
5.1 Introduction . . . 66
5.2 Finding a 3D discrete rotation from a pair of digital images . . . 67
5.2.1 Approximation of the rotation axis by a 3D Pythagorean vector . . . 67
5.2.1.1 First step . . . 67
5.2.1.2 Second step . . . 68
5.2.2 Approximation of the rotation angle using rational multi-grids . . . 70
5.3 Conclusion . . . 71

**6 Study on Pythagorean n-tuples 72**
6.1 Introduction . . . 72
6.2 Pythagorean triples . . . 73
6.3 Pythagorean quadruples . . . 75
6.4 Approximation of a given 3D vector by a Pythagorean 3D vector . . . 78

6.4.1 Generation of Pythagorean quadruples . . . . . 79
6.4.2 Approximation methods . . . . . 82
6.4.3 Experiments . . . . . 84
6.5 Pythagorean $n$-tuples . . . . . 88
6.6 Approximation of an $n$D vector by an $n$D Pythagorean vector . . . . . 91
6.7 Conclusion . . . . . 92

**7 Conclusion** **93**
7.1 Discrete rotation in 2D . . . . . 93
7.2 Discrete rotation in 3D . . . . . 94
7.3 Discrete rotation in $n$D . . . . . 95

**8 Appendix** **97**

**Bibliography** **101**

# List of Figures


1.1 Intersection between two circles and between their discretization. The discrete circles have four pixels for the intersection while the continuous circles have only one point of intersection. That shows one of the main differences between Euclidean geometry and digital geometry. . . . . 3

2.1 The four main problems of discrete rotation algorithms: (a) double point, (b) blank point, (c) errors due to composition and (d) inverse rotation. . . 8
2.2 The DER rotation of a white image by the four angles (a)$\alpha = 15°$, (b)$\alpha = 30°$, (c)$\alpha = \sin^{-1}\frac{3}{5} \approx 36.87°$ and (d)$\alpha = 37°$ . . . . . 9
2.3 Percent of point lost during the rotation of a $200 \times 200$ picture for all angles between $[0, \frac{\pi}{2}]$. . . . . 9
2.4 During the composition of DER, the number of points lost due to double point is cumulative. For an angle $\alpha = 40°$ we lost approximately 15% of points for (a), 28% for (b), 39% for (c) and 50% for (d). . . . . 11
2.5 If a discrete rotation is DER-like, then the red way that discretize then rotate will gives the same result than the blue way that rotate then discretize. 12
2.6 The rotation by circles by an angle of 45°. . . . . 13
2.7 The twelve biggest angles that can possibly be obtained by Pythagorean lines . . . . . 14
2.8 The quasi-shear rotation and its two intermediate results. . . . . 15
2.9 A pixel and (a) its four 4-connected pixels, (b) its eight 8-connected pixels in a square grid. . . . . 16
2.10 A pixel and its six 6-connected pixels in a hexagonal grid . . . . . 17
2.11 A voxel and (a) its six 6-connected voxels, (b) its eighteen 18-connected voxels and (c) its twenty-six 26-connected voxels in a cubic grid. . . . . 18
2.12 Conversion from a square grid into a triangle grid . . . . . 18
2.13 Conversion from a square grid into a hexagonal grid . . . . . 19
2.14 The percentage of points remaining after a rotation in a triangular grid. . 19
2.15 Comparison of the average conservation of neighborhood during a rotation angle between the square grid and (a) 4-connectivity on a triangle grid, (b) 8-connectivity on a triangle grid,(c) 4-connectivity on a hexagonal grid and (d) 8-connectivity on a hexagonal grid. . . . . 20
2.16 The percentage of points remaining after rotation in a hexagonal grid. . . 21
2.17 Result of four rotations of the image (a) by an angle of 15° (b) using a square grid, (c) using a hexagonal grid and (d) using a triangle grid. . . . 22
2.18 Result of four rotations of the image (a) by an angle of 30° (b) using a square grid, (c) using a hexagonal grid and (d) using a triangle grid. . . . 23
2.19 Result of four rotations of the image (a) by an angle of 45° (b) using a square grid, (c) using a hexagonal grid and (d) using a triangle grid. . . . 24

3.1 A hinge angle $\alpha(P,Q,K)$ (left) and four symmetrical hinge angles (right). 28
3.2 The corners of $\mathscr{H}(\vec{q})$, namely, four corners of a pixel around $\vec{q}$. . . . . 35
3.3 Illustration of cases $\Sigma\mathscr{F}(C_i(\vec{q}_2)) = 0,1,2$ or 3. . . . . 37
3.4 Running of the incremental algorithm. . . . . 39
3.5 Randomly generated points and their corresponding rotated points. . . . . 40
3.6 Result of our algorithm applied on the sets of points in Figure 3.5 . . . . . 40
3.7 Result of our algorithm applied on a set of floating points. . . . . 42
3.8 Examples of errors for computing lower and upper bounds of rotation angles introduced by rotation of floating points. . . . . 43
3.9 Result of our algorithm applied on real data. . . . . 44
3.10 Problem of corresponding points with real data. . . . . 45

4.1 All hinge angles in the first quadrant for the grid point $\vec{p} = (2,1)^\top$, such that $\alpha_1 \approx -13.68°$, $\alpha_2 \approx 15.59°$, $\alpha_3 \approx 21.31°$, $\alpha_4 \approx 50.52°$. The hinge angle $\alpha_5$ is obtained by symmetry such that $\alpha_5 = \frac{\pi}{2} - \alpha_1 \approx 76.32°$ . . . . 49
4.2 The 3D half-grid cut by a plane (a), and its multi-grid (b). . . . . 51
4.3 Parallel lines of a set $\mathcal{L}_i^{\mathcal{A},\vec{p}}$ and geometric interpretation of their parameters 51
4.4 The five different shapes of convexels, which are constructed as the intersections between a rotation plane and voxels [1]. . . . . 52
4.5 Two multi-grids generated by a rotation axis but with two different points $\vec{p}_1, \vec{p}_2$ one of which is obtained by a translation of $\pm(\vec{p}_1 - \vec{p}_2)$ from the other. 54
4.6 A quadruple of grid points $\{\vec{q}_1 = \vec{q}, \vec{q}_2 = (q_x + A_x, q_y + A_y, q_z)^\top, \vec{q}_3 = (q_x + A_x, q_y, q_z - A_z)^\top, \vec{q}_4 = (q_x + 2A_x, q_y + A_y, q_z - A_z)^\top\}$ forms two types of triangles, $\alpha$ and $\beta$ tiled in a multigrid. . . . . 56
4.7 the two impossible cases of intersection between a circle centered on a grid point and a rational multi-grid. . . . . 57
4.8 A hinge angle $\alpha$ for a point $\vec{p}$ in a rational multi-grid. . . . . 59

5.1 Example of the approximation of a 3D vector using our method. (a) and (b) represent the projections of $\vec{r}$ into $\vec{r}_2$ and $\vec{r}_3$ in 2D planes respectively. Each 2D convex cone in (a) and (b) is the admissible approximation for $\vec{r}_2$ and $\vec{r}_3$ regarding to $\epsilon$. (c) represents the square pyramid constructed from the two 2D convex cones that contains the sets of admissible approximation for $\vec{r}$. . . . . 70
5.2 In a rational multi-grid $\mathcal{M}^{\mathcal{A},\vec{p}}$ a convexels can have four intersections with the locus of rotation of $\vec{p}$. . . . . 71

6.1 Two representations of a Pythagorean triple $(i_1, i_2, \lambda)$: Pythagorean angle $\theta$ and Pythagorean vector $\vec{v}$ in the plane . . . . . 73
6.2 Construction of all Pythagorean triples of the form $(i_1, i_2, i_1 + 1)$ . . . . . 74
6.3 A convex cone in the plane defined by the pair of vector $\vec{v}_1, \vec{v}_2$ . . . . . 75
6.4 Geometrical representations of a Pythagorean quadruple $(i_1, i_2, i_3, \lambda)$ by the Pythagorean vector $(i_1, i_2, i_3)^\top$ and the pair of angles $(\theta, \psi)$. . . . . 76
6.5 A 3D convex cone generated by three vectors $\vec{v}_1, \vec{v}_2, \vec{v}_3$ in the space . . . . 77
6.6 (a) The 3D convex cone $\mathcal{C}_3$ defined by $\vec{v}_1, \vec{v}_2, \vec{v}_3$ and their projected vectors $\vec{v}'_1, \vec{v}'_2, \vec{v}'_3$ on the OXY plane, and (b) the 2D convex cone $\mathcal{C}_2$ defined by $\{\vec{v}'_1, \vec{v}'_2\}$. . . . . 77
6.7 (a) The plane $\mathcal{P}$ defined by $OZ$ and $\vec{u}$, and (b) the vector $\vec{w}$ that belongs to $\mathcal{C}'_2$. . . . . 78

6.8 The three first level of the Pythagorean tree. . . . . 80
6.9 The Pythagorean quadruple $< 23, 24, 24, 41 >$ can only be reached using Matrix 6.4. . . . . 82
6.10 In most cases, $\mathcal{M}_4$ gives the father of a Pythagorean quadruple. . . . . 82
6.11 There exists an infinity of Pythagorean quadruples that have themselves as grandfather. . . . . 82
6.12 (a) Distance between 1000 vectors and their approximated Pythagorean vector for a Pythagorean tree of depth 4 (a), of depth 5 (b), of depth 6 (c), of depth 7 (d) and of depth 8 (e) . . . . . 85
6.13 Distance between 1000 vectors and their approximated Pythagorean vector for a partial Pythagorean tree of depth 10 (a), of depth 20 (b), of depth 50 (c), of depth 100 (d), of depth 200 (e) and of depth 300 (f) . . . 87
6.14 Distance between 1000 vectors and their approximated Pythagorean vector for a random Pythagorean tree of depth 10 (a), of depth 20 (b), of depth 50 (c), of depth 100 (d), of depth 200 (e) and of depth 300 (f) . . . 89

*À beaucoup trop de personnes pour qu'elles soient citées sur cette simple page*

# Chapter 1

# Introduction

## 1.1 Introduction to the rotation problem in $\mathbb{Z}^2$

Rotation is a well known function learned at junior high school. For most of the people, rotation has been a well defined function from $\mathbb{R}^2$ into $\mathbb{R}^2$ for centuries.

Recently, with the spread of computer sciences, more and more people are becoming interested in the rotation as a function from $\mathbb{Z}^2$ into $\mathbb{Z}^2$, commonly denoted by discrete rotation. Indeed, in computer science, the data acquired from digital cameras or other methods are often digital data, i.e. containing only integers, and it is known that using only integers during computations reduce the computation time and avoid approximations errors.

Nonetheless, performing rotations in $\mathbb{Z}^2$ is not obvious. Many problems arise mainly due to the fact that $\pi$, which seems to be inseparable of the rotation function, is a transcendental number.

For the last twenty years, some works have been done to understand the particularity of rotations in the discrete space[2–5], to define and design discrete rotation in 2D [6, 7] or discrete rotation in 3D [8–10]. However, the discrete rotation problem is far to be solved. The comprehension on discrete rotations is far to be the same as comprehension on Euclidean rotation. The apparently obvious properties of the Euclidean rotation such as bijectivity, transitivity are hard to keep in discrete spaces. Today, at our best knowledge, there is no discrete rotation that have all the properties of the Euclidean rotation.

During my thesis, we try to improve the understanding of the discrete rotation. The results obtained are more theoretical than practical. Our main results showed that it is possible to perform rotations in 2D and in 3D using only integers computations, then

obtaining exact results under some assumptions. We also showed that every rotations performed with floating number can be performed using the methods proposed in this manuscript. However, even if these results are exact, the associated algorithms have a hight time complexity.

The discrete rotation is a function coming from a new mathematical field denoted by digital geometry. In the next sections of this introduction, we make a brief history of the geometry and its evolutions that lead us to the digital geometry and then to the discrete rotations. The following sections of this chapter are not required for the understanding of this thesis and can be skipped, except the last one that gives a quick overview of the content of this manuscript.

## 1.2 A quick history of geometry

Geometry, literally "Earth-Measuring" is a part of mathematics concerned with questions of sizes, shapes, relative positions of figures, and spaces properties. Geometry is one of the oldest sciences. Initially a body of practical knowledge concerning lengths, areas, and volumes, in the 3rd century before Christ geometry was put into an axiomatic form by Euclid [11]. Euclidean geometry set a standard for many centuries to follow.

The introduction of coordinates by Descartes, during the 17th century, and the concurrent development of algebra, marked a new stage for geometry, since geometric figures, such as plane curves, could now be represented analytically, i.e., with functions and equations. This played a key role in the emergence of calculus in the 17th century. The subject of geometry was further enriched by the study of intrinsic structure of geometric objects that originated with Euler and Gauss and led to the creation of topology and differential geometry.

Since the discovery of non-Euclidean geometry in the 19th-century, the concept of space has undergone a radical transformation. Contemporary geometry considers manifolds, spaces that are considerably more abstract than the familiar Euclidean space. These spaces may be endowed with additional structures, allowing speaking about lengths.

For fifty years, geometry has been required in a lot of applications in computer science such as computer vision, computer graphics, medical image analysis, etc. However, in computer science, the classical geometry is not suitable. First, in computer science, it is only possible to represent exactly integers and rational numbers in memory, whereas Euclidean geometry mainly uses real numbers. Secondly, data used in many application are represented using only integer numbers. Therefore, using the floating numbers, which are the numbers that approximate real numbers in computer science, inherent to the

classical geometry is not desirable. Consequently, a new field of geometry, the digital geometry, have been introduced a few years ago in order to design a geometry using integers and rational numbers only.

## 1.3 Digital geometry

In recent years, with the spread of the usage of images generated or processed by computers, a new mathematical theory called digital geometry has emerged as a subfield of the discrete geometry. This theory aims at studying objects called discrete objects consisting of countable sets of discrete points, whereas Euclidean continuous objects generally consist of non-denumerable sets of points. Besides their names, discrete objects have little in common with Euclidean geometric objects. Indeed, most elementary results of Euclidean geometry do not hold in discrete spaces: fundamental notions like continuity does not hold (what does "continuous" mean in a discrete space ?), even the basic definition of objects become complex (many definitions of a discrete circle that are not equivalent exists). Digital geometry tries to address these problems by devising new results specific to discrete spaces and objects, and transposing the familiar notions of Euclidean geometry to discrete geometry.

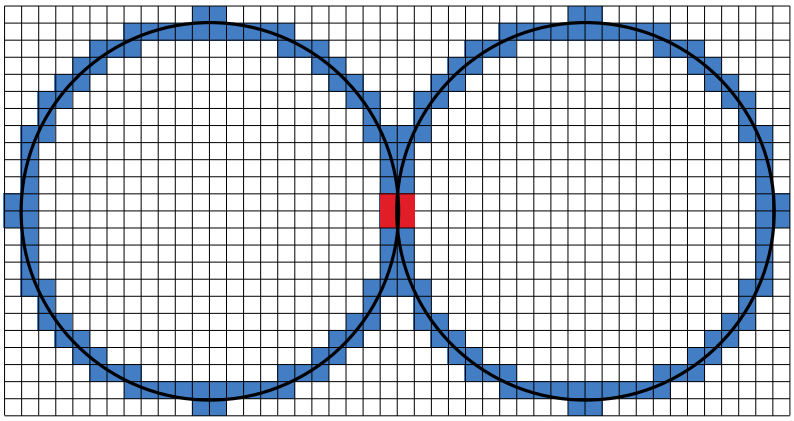


FIGURE 1.1: Intersection between two circles and between their discretization. The discrete circles have four pixels for the intersection while the continuous circles have only one point of intersection. That shows one of the main differences between Euclidean geometry and digital geometry.

Figure 1.1 gives a simple example of the difference between the Euclidean geometry and the digital geometry. In Euclidean geometry, two circles have zero, one, two or an infinity of intersections (if they are merged). However, in digital geometry, two discrete circles can have any number of common pixels (four common pixels on Figure 1.1) but not an infinity since the number of pixels in a discrete circle is necessarily finite. Another example can be two circles that have no intersection in the Euclidean space, but their discretized versions have one or more common pixels.

## 1.4 From digital geometry to discrete rotations

The rotation is a geometrical transformation that have probably been introduced by the Greeks one or two centuries B.C. for studying astrology. The rotation is inseparable from the Archimedes constant, better-known since the 17th or 18th century as $\pi$. Archimedes was the first to rigorously study $\pi$ by realizing that he can obtain a lower and a upper bound by inscribing circles in regular polygons and calculating the outer and inner polygons respective perimeter [12].

Between Archimedes, about five centuries B.C. and the end of the 19 th century, many mathematicians tried to give a good approximation of $\pi$ or its exact value. The accuracy of the estimation of $\pi$ quickly increases with the development of mathematics. With the introduction by Liouville of transcendental number in 1844[13] and the demonstration by Lindemann in 1882 that $\pi$ is one of these numbers [14], we know that $\pi$ cannot be exactly represented or computed.

The impact of the belonging of $\pi$ to the transcendental numbers on digital geometry is very important. Indeed, we know that each computation from $\mathbb{Z}^n$ to $\mathbb{Z}^n$ using $\pi$ will give an approximated result. More generally, $\pi$ is not the only problem of the rotations in discrete space. Particular angles exist that allow rotations to be performed without using $\pi$ denoted by Pythagorean angles[15]. Whereas, performing rotations using these angles will imply different problems that will be explained in Chapter 2. Some methods have been developed in digital geometry in order to solve these problems and to perform rotations in discrete space that keep some of the properties of the Euclidean rotations. These rotations will be introduced in Chapter 2. However, the method that have been proposed cannot solve all the problems of rotations in discrete space. Each one of them have some properties but none have all the properties of Euclidean rotation. Therefore, during my thesis, we tried to improve the understanding of rotations in discrete space.

In the next section, we give a quick overview of the content of this manuscript.

## 1.5 Structure of this manuscript

This manuscript is centered on six chapters. Chapter 1 is finished, so it is useless to introduce it.

Chapter 2 presents a survey on 2D discrete rotations. The rotations in the 2D discrete space have been studied during the last 30 years. However many problems are remaining in the application of rotations in the discrete space. Thus, in Chapter 2, we also introduce and explain these problems. We also present a short study on the rotation in different

regular grid and show that using other regular grid can give better result than orthogonal grid.

In the Euclidean space, for a given point and center of rotations, there is an infinity of different rotations of the point around the given center. This is not true in a discrete space as the number of different rotations is finite and depends on the distance between the point and the center. Chapter 3 presents a discrete rotation using this property of rotations in the discrete space and then gives a method that estimates the angle of rotation from a given pair of digital images that represent the same object before and after a rotation.

Chapter 4 presents a short survey on rotations in the 3D continuous space since, at our best knowledge, there is not discrete 3D rotation. Rotations in the 3D space can be done in many ways such as composing three 2D rotations or by rotating around an arbitrary axis. In this chapter, we will present the extension of the rotation introduced in Chapter 3 to the 3D discrete space using a rotation around an arbitrary axis.

Chapter 5 extends the second part of Chapter 3. It presents a discrete method to find from a pair of 3D digital images the angle of rotation and the axis of rotation that transforms the first digital image into the second digital image where the correspondence of points between these two digital images are supposed to be known.

Chapter 6 presents a study on Pythagorean $n$-tuples that are the extension to dimension $n$ of the well-known Pythagorean triples. They can be represented in any dimension as a vector. In this chapter, we will show that they are dense in every dimension, i.e. any vector in any dimension can be approximated by an infinity of Pythagorean $n$-tuples. Since all computations on Pythagorean $n$-tuples can be done using only integers, they are helpful for designing discrete rotation in any dimension.

We will conclude by some future works in Chapter 7. The first one will be to design a discrete rotation valid in any dimension. As for angles, there is a finite number of rotation axis in the discrete space. However, there is no clear relation established between these two points. Another future work will be to clarify this relation.

In appendix, we add a glossary that recalls the main concepts and definitions used and introduced in this thesis but in a less formal way.

# Chapter 2

# Rotations in 2D discrete spaces

## 2.1 Introduction

In this chapter, we present a survey on the discrete rotations and the problems arising while we apply rotations in a discrete space. For many applications, the rotation in the discrete space is a required transformation for image computation in computer science such as image matching, construction of mosaic images[16] but also in physics such as celestial mechanics[7]. Typically, the input image is a discretized version of the original analog signal and, therefore, stored as a byte image. A rotation is applied to process the image; this process may result in intermediate floating point numbers. However, the final image displayed or analyzed is similar to the input image Ů stored as a byte image.

As an example, a 2D digital image is represented as a set of $N \times N$ discrete points where each discrete point $\vec{p} = (i, j), \in \{0, \ldots, n\}^2$ contains the color value. If we apply the classic rotation matrix $\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$ on this set of points, we obtain a set of points that belongs to $\mathbb{R}^2$ instead of $\mathbb{Z}^2$. Therefore, the result of the rotation must be discretized in order to obtain an output comparable to the input. During the discretization, some problems occur such as points without pre-image or points with two pre-images.

Usually, in application of discrete rotations, we consider that the discrete plane is tiled by squares, usually denoted by pixel. However, there exist other regular forms that tile the plane such as triangles and hexagons. Performing rotation in the plane using these two forms can give some interesting results.

In this section, we will present the classic definitions and properties expected for the discrete rotation algorithms. The expected properties are the same than the Euclidean rotation, however, we will also explain why it is difficult to keep these properties in the

discrete space. Then we will present a survey on the main existing rotation algorithms. Finally, we propose a study on rotation in triangle and hexagonal grid.

## 2.2 Definitions and Properties

It is generally admitted that a discrete rotation is a rotation algorithm that transforms a set of discrete points into another set of discrete points. More formally:

**Definition 2.1.** An algorithm is denoted by a discrete rotation algorithm if it performs a rotation of a set of points from $\mathcal{Z}^2$ into $\mathcal{Z}^2$.

This definition allows discrete rotation algorithms to use floating numbers[1] during computations. However, using floating numbers during computations involves errors of approximation in the results. A simple example is the sum: $\sin^2\alpha+\cos^2\alpha$. In mathematics this sum is always equal to 1 although with the use of floating numbers, we can obtain the result $1 \pm \epsilon, \epsilon > 0$. Therefore, we believed that Definition 2.1 is not reliable enough. Then we propose a more restrictive definition for the discrete rotation:

**Definition 2.2.** An algorithm is denoted by a discrete rotation algorithm if it performs a rotation of a set of points from $\mathcal{Z}^2$ into $\mathcal{Z}^2$ and uses only integers during computations.

A rotation algorithm using only rational numbers during computations is also a discrete rotation algorithm since, as explained in introduction, rational numbers and integers are equivalent. Hereafter, the notation *discrete rotation* will always refers to Definition 2.2.

Definition 2.2 does not ensure any property on discrete rotation algorithms. In the next section, we will present some discrete rotation algorithms that do not have the same properties than the Euclidean rotation. Hereafter, we will introduce the Euclidean rotation adapted to discrete space: discretized Euclidean rotation, abbreviate DER and explain the most common problems arising while applying DER and associate to each of these problems a property that a discrete rotation algorithm should have. Note that DER is not a discrete rotation regarding to Definition 2.2 since it uses sine and cosine functions.

The first problem denoted by *double points* is illustrated in Figure 2.1(a). During the DER, is it possible that two points, which are 4-connected, have their images in the same pixel. Then during the discretization of the rotation, these two points are discretized in the same pixel. To give an idea of this problem, we must notice that the Euclidean

[1] Floating numbers are the representation in computer science of the real numbers. They have a limited precision that implies an approximation of the real numbers.

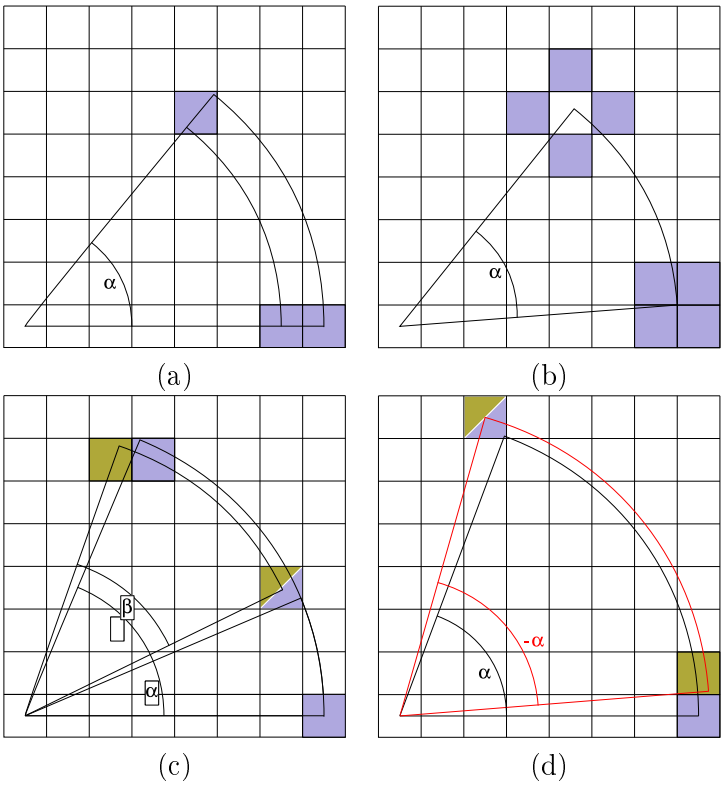



FIGURE 2.1: The four main problems of discrete rotation algorithms: (a) double point, (b) blank point, (c) errors due to composition and (d) inverse rotation.

distance between two 4-connected pixels is 1 when the diagonal of a pixel is $\sqrt{2}$. Then after a rotation and before the discretization, the centers of two 4-connected pixels can belong to the same pixel. However, using the same idea, we can conclude that triple points cannot exist.

The second problem denoted by *blank points* is illustrated in Figure 2.1(b). There is sets of four points that form a $2 \times 2$ square, which is 4-connected, that after a rotation by a given angle become 8-connected. We call the point within the 8-connected square a blank point. This problem has the same origin than the problem of double points. After the application of DER on an image and if we do not crop the rotated image, the number of double points is the same as the number of blank points, however, this is not true on an arbitrary set of points.

Few examples of these two problems are given in Figures 2.2 (a),(b),(c) and (d) representing four rotations of a white image of size $450 \times 450$. The black point on the resulting image (except the four black triangles on each corner ) are points that have no pre-image. Figure 2.3 illustrates the percentage of points lost during the rotation of a $200 \times 200$ picture for all angles between $[0, \frac{\pi}{2}]$[2].

[2]A study on hinge angle in Chapter 3 shows that in discrete space for a finite size, there is a finite number of possible rotations

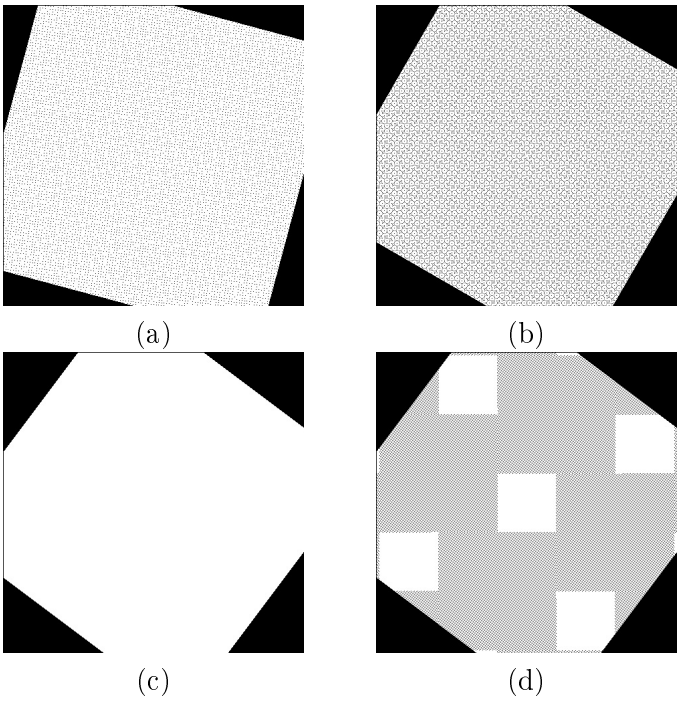



FIGURE 2.2: The DER rotation of a white image by the four angles (a)$\alpha = 15°$, (b)$\alpha = 30°$, (c)$\alpha = \sin^{-1}\frac{3}{5} \approx 36.87°$ and (d)$\alpha = 37°$

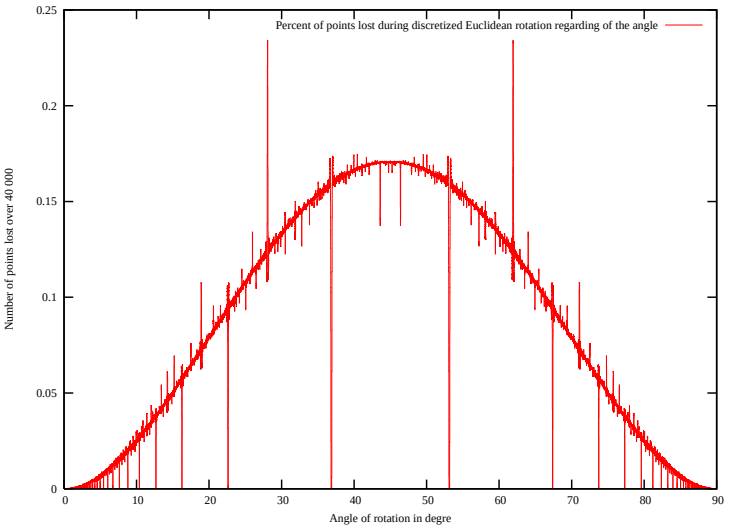



FIGURE 2.3: Percent of point lost during the rotation of a $200 \times 200$ picture for all angles between $[0, \frac{\pi}{2}]$.

Note that the black points resulting of the rotations by DER draw regular patterns on the different images on Figures 2.2 (a),(b),(c) and (d). A short study of these patterns have been done in [4]. Figures 2.2 (c) shows that some particular angles do not create any blank points. These particular angles are used to create a discrete rotation presented Section 2.3.2.

These two problems show that a DER is not necessarily a bijective function since there

are points resulting of a discrete rotation that have two pre-images and the inverse rotation does not give the original image. To capture these two problems, we introduce the following definition:

**Definition 2.3.** A function $\mathcal{F} : \mathbb{E}^2 \to \mathbb{E}^2$ is bijective if for all elements $e \in \mathbb{E}^2$, there is exactly one element $e' \in \mathbb{E}^2$ such that $\mathcal{F}^{-1}(e') = e$.

Another problem is arising during composition of discrete rotations. Let $\vec{p}_1$ be a point and $\alpha, \theta$ be two angles. The point $\vec{p}_2$ is the result of the application of DER on $\vec{p}_1$ by $\alpha$ and $\vec{p}_3$ is the result of the application of DER on $\vec{p}_2$ by $\theta$. We cannot ensure that the result of the application of DER on $\vec{p}_1$ by $\alpha + \theta$ is $\vec{p}_3$ as illustrated on Figure 2.1(c). Another problem similar to the problem of composition of discrete rotations is the inverse rotation as illustrated in Figure 2.1(d). Let $\vec{p}_2$ be the result of the rotation by DER of $\vec{p}_1$ by an angle $\alpha$. We cannot ensure that the result of the rotation by DER of $\vec{p}_2$ by an angle $-\alpha$ is $\vec{p}_1$.

These two problems show that rotations perform in discrete space are not necessarily transitive. Note that the inverse rotation is just a particular case of the transitivity problem were $\theta = -\alpha$. To capture this problem, we introduce the following definition:

**Definition 2.4.** A discrete rotation is transitive if for all angles $\alpha$ and $\gamma$ and for all points $\vec{p}$ the rotations of $\vec{p}$ by $\alpha$ then by $\gamma$ give the same result than the rotation of $\vec{p}$ by $\alpha + \gamma$.

Note that, the problems of double points and blank points are cumulative with the problem of compositions. An example is given in Figure 2.4. These figures represent the four rotations of a white image of size $450 \times 450$ by an angle of $\alpha = 40°$. Figure 2.4(a) is the result of the first rotation of the white image by $\alpha$. Figure 2.4(b) is the result of the rotation of Figure 2.4(a) by the same angle $\alpha$. We use the same method to obtain Figure 2.4(c) and Figure 2.4(d). After the last rotation, it remains approximately 50% of the white points of the original image.

The last definition is not resulting from a problem of discrete rotation. In the field of computer vision, data are often acquired from real objects and during the acquisition process, they are discretized. Then the rotation done on an object between two discretizations is the continuous rotation, so we can assume that the result of this rotation will be captured by DER. However, during the discretization of the object after the rotation, we do not have any blank point. Therefore, an important property of a discrete rotation is to obtain the same results as DER, at least for non double points and non blank points. Then we propose the following definition

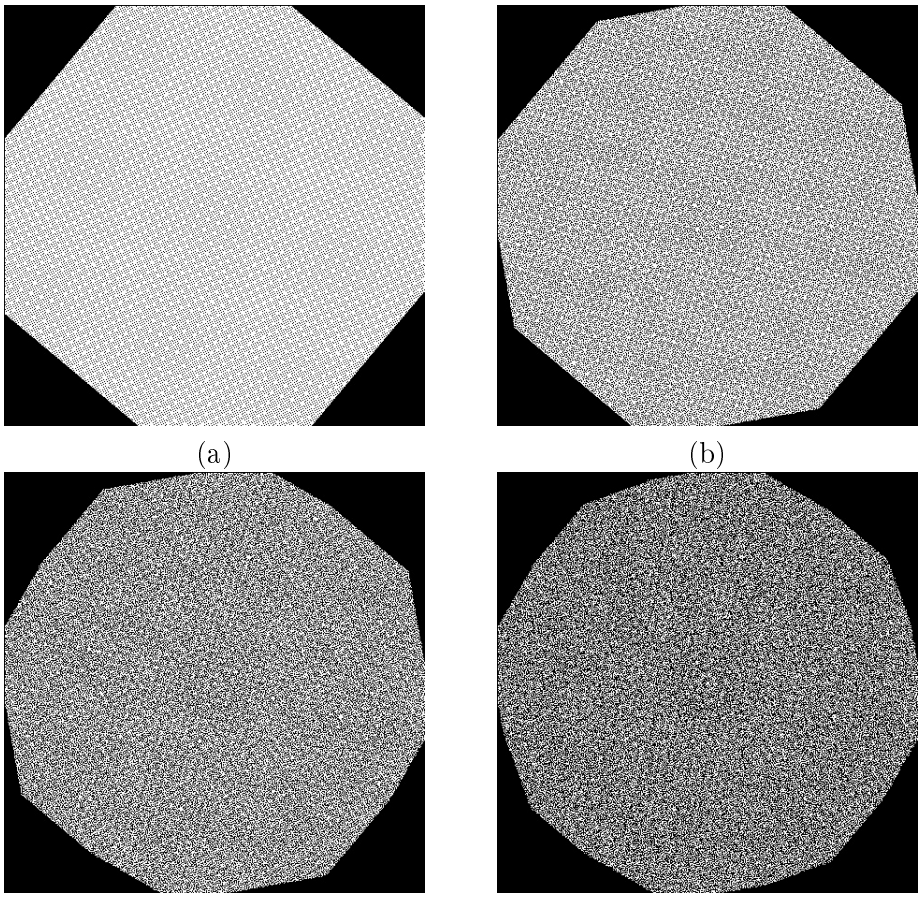



FIGURE 2.4: During the composition of DER, the number of points lost due to double point is cumulative. For an angle $\alpha = 40°$ we lost approximately 15% of points for (a), 28% for (b), 39% for (c) and 50% for (d).

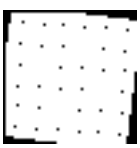
**Definition 2.5.** A discrete rotation is said to be DER-like if the set of points resulting of the discrete rotation by an angle $\alpha$ includes the set of points resulting of DER by $\alpha$

Figure 2.5 illustrates the DER-like problem. We say that a rotation is DER-like if the object obtained using the red way is the same than the object obtained using the blue way.

At our best knowledge there is no discrete rotation that is bijective, transitive and DER-like for any angle. We strongly believe that is it not possible to design such a rotation. Therefore, we propose the following conjecture:

**Conjecture.** 1. If the function $\mathcal{R} : \mathbb{Z} \to \mathbb{Z}$, is a discrete rotation, this function is not bijective and transitive and DER-like for any angle.

In the next section, we present some discrete rotation algorithms. We will use the three definitions presented above to characterize these discrete rotations.



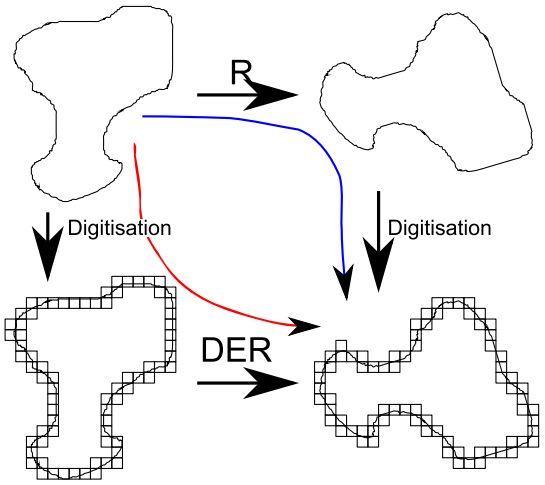



FIGURE 2.5: If a discrete rotation is DER-like, then the red way that discretize then rotate will gives the same result than the blue way that rotate then discretize.

## 2.3 Survey on discrete rotations in the plane

In this section, we present some of the existing discrete rotation according to Definition 2.2. The first rotation is the rotation by discrete circle, then we present the rotation by Pythagorean line already addressed in previous section. We then present the shear rotation that consist in the decomposition of the rotation into three sheer transforms. Finally, we present the rotation by hinge angles which is the rotation that we mainly studied in this thesis.

### 2.3.1 Rotation by discrete circle

We first present the rotation by discrete circles[2]. A discrete circle of radius $r$ is the set of $n$ pixels $\vec{p_1} = (p_{1x}, p_{1y}), \vec{p_2} = (p_{2x}, p_{2y}) \ldots, \vec{p_n} = (p_{nx}, p_{ny})$ such that $(r - \frac{1}{2})^2 \leq p_{ix}^2 + p_{iy}^2 < (r + \frac{1}{2})^2, i = 1 \ldots n$[3]. Using this definition of discrete circles, we know that we will obtain a tiling of the plane by discrete circles. In other words, we can ensure that each pixel belongs to a discrete circle and to only one. The main idea of the rotation by discrete circles is to move the pixel in each discrete circle of a number depending on the rotation angle. This number is obtained using the following equation :$\lfloor \frac{\alpha \times n}{360} \rfloor$ where $\alpha$ is the given rotation angle and $n$ the number of pixels belonging to the discrete circle. An example of the rotation by discrete circles is given in Figure 2.6. In his thesis, Andres shows that the rotation by discrete circles is transitive and bijective. We can easily be convinced of these two properties as we just displaced pixel in the circle during

[3]There are different definitions of a discrete such as Bresenham circle[17], but this rotation requires a definition of discrete circles that tile the plane.

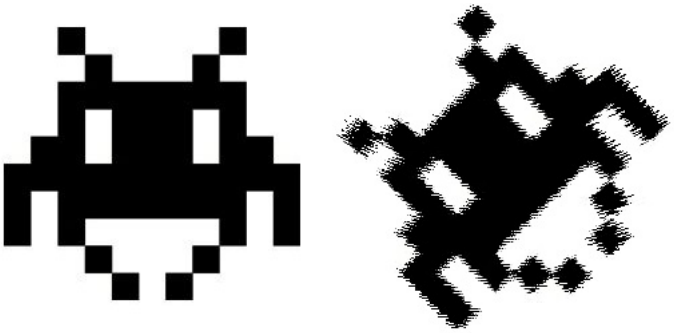


FIGURE 2.6: The rotation by circles by an angle of 45°.

the rotation of a discrete circle. Therefore, there is no blank or double points. In other words, this discrete rotation does not lose any pixel.

The application of DER and the rotation by discrete circles on the same set of points will give different results. This remark seems obvious since the rotation by discrete circles is bijective. However, Figure 2.6, which gives an example of a rotation by discrete sphere on an angle of 45°, shows some shears in the resulting digital image. These shears do not arise during a rotation by DER. Thus, we can conclude that according to Definition 2.5, the rotation by discrete circles is not DER-like.

### 2.3.2 Rotation by Pythagorean lines

Another discrete rotation is the rotation by Pythagorean lines. A line of slope $\frac{i_1}{i_2}$ that verifies $i_1^2 + i_2^2 = \lambda^2$ is a Pythagorean line if $i_1, i_2, \lambda \in \mathbb{Z}$. In [2], the author introduce a rotation algorithm using the particular set of Pythagorean lines where $\lambda = i_2 + 1$. In [18], authors give the proof that the rotations induced by these set of Pythagorean lines is bijective, transitive and DER-like. However, the rotation by Pythagorean lines has restricted angles. The biggest angle is given by the Pythagorean triple $(3, 4, 5)$, which is approximately 36.87°. All other angles given by a Pythagorean triple are smaller than this one, they are given by $(5, 12, 13)$ that is approximately 22.62°, $(7, 24, 25)$ that is approximately 16.26°, $(9, 40, 41)$ that is approximately 12.68° and so on. There is about 50 different angles greater than 1°. The twelve biggest angles are illustrated in Figure 2.7. Note that the results obtained by DER illustrated in Figure 2.3(c) are the same than the results obtained by a rotation by the Pythagorean line associated to $(3, 4, 5)$. It is possible to compose rotations by Pythagorean lines in order to obtain any angle, but in case of composition, the final result will not be consistent since the connectivity between pixels will be lost during the composition[4]. In the final result, while using composition, some shear effects will appear, which do not arise during a rotation by DER.

[4]Some results on connectivity are presented below in this section.

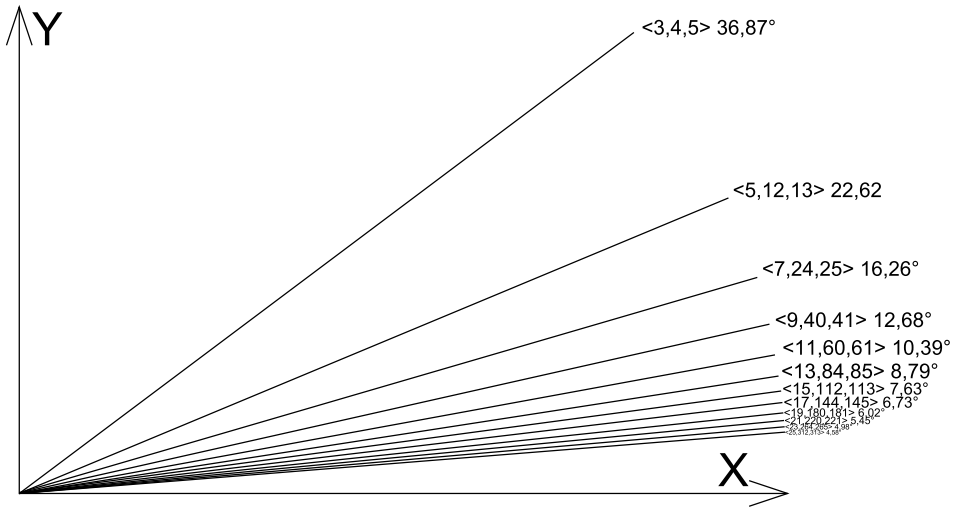



FIGURE 2.7: The twelve biggest angles that can possibly be obtained by Pythagorean lines

### 2.3.3 Shear rotation and quasi shear rotation

It is also possible to design a discrete rotation using three translations. Indeed, the rotation matrix $\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$ can be rewritten as the product of the three matrices : $\begin{pmatrix} 1 & -\tan\frac{\alpha}{2} \\ 0 & 1 \end{pmatrix}$, $\begin{pmatrix} 1 & 0 \\ \sin\alpha & 1 \end{pmatrix}$, $\begin{pmatrix} 1 & -\tan\frac{\alpha}{2} \\ 0 & 1 \end{pmatrix}$ that are translation matrix. This method called shear rotation is well studied in the continuous plane [19]. In [6], the author gives a method, called quasi-shear rotation, to discretize these three matrices and then obtain a discrete rotation, working in the discrete space, using three horizontal or vertical shear. Obviously, the three translations are bijective, then we can conclude that the quasi-shear-rotation is bijective. However, the translations are not transitive, indeed the composition of rotations of angles $\theta$ and $-\theta$ does not necessarily gives the original image and the composition of the two rotations of angles $\theta_1$ and $\theta_2$ does not necessarily gives the same result as the rotation of angle $\theta_1+\theta_2$. Moreover, this rotation is not DER-like since some shears will appear during the rotation. We can conclude that QSR is bijective, transitive and non DER-like. An example of the quasi-shear rotation where the three translations are decomposed is given in Figure 2.8.

### 2.3.4 Rotation by hinge angles

More recently, a new discrete rotation has been designed by Nouvel in [4] and improved by us Chapter 3. This rotation is based on hinge angles and will be more developed in the next sections. Roughly speaking, an angle is a hinge angle for a discrete point if the result of this point's rotation by this angle belongs to the half-grid. In other words, if the

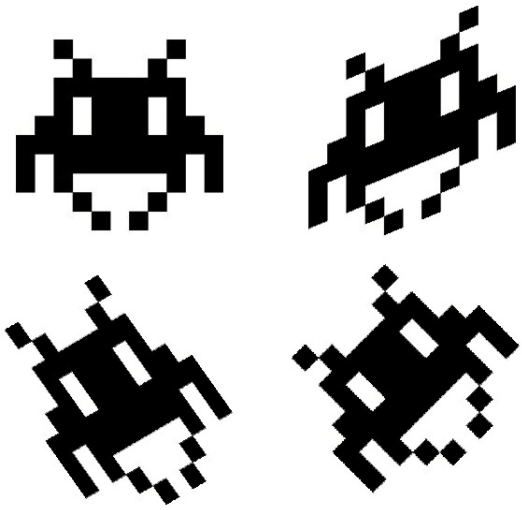


FIGURE 2.8: The quasi-shear rotation and its two intermediate results.

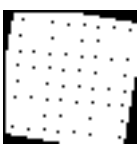
angle of rotation is slightly greater than a hinge angle, the result of the rotation of the associated point will be in a pixel $p$, if the angle of rotation is slightly lower than a hinge angle the result of the rotation of the same point will be in a pixel that is 4-connected with $p$. As far as we know, the discrete rotation by hinge angles is the only discrete rotation DER-like for any angle, however, rotation by hinge angles is neither bijective nor transitive.

In our purpose, we assume that discrete data are the result of the acquisition by a digital camera of continuous objects. This supposition enforces the property that the discrete rotation between two different sets of discrete data gives the same result as DER. For this reason, during my thesis, we have chosen to improve the rotation by hinge angles. Therefore, in the next chapter, we will define hinge angles and give their properties.

## 2.4 Rotation in other grids

All the rotations presented in Section 2.3 are based on the square grid. However, other regular grids exist, in this section, we present a short study on rotations in the two other kinds of regular grids that are triangles and hexagonal grids. In discrete geometry, the square grid is the most commonly used, however, this grid is not the only regular grid that tiles the plane. There are two other regular geometrical forms that tile the plane: the triangle and the hexagonal.

In the field of discrete geometry, the square grid is almost the only grid used to perform transformations. During my thesis we exclusively used the square grid. However, using other kind of grids can present interesting properties such as conservation of the neighborhood, that helps to keep a visual consistency of the digital image, or loosing



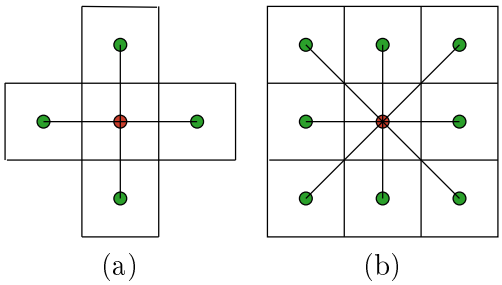



FIGURE 2.9: A pixel and (a) its four 4-connected pixels, (b) its eight 8-connected pixels in a square grid.

less points during rotations, that allows keeping more information on the transformed digital image. Thus in this subsection, we present some experiments on rotations done using these two other kinds of grids. Here we only consider the two other regular grids: the triangles and hexagons, that tile the plane with an area of 1 and with regular edges. First, we introduce some notions required to understand the studies done in this chapter, then, we present the methodology used to perform our test and finally, we present our results on the rotation on triangle grid then on hexagonal grid.

### 2.4.1 Connectivity on the discrete grids

In this section, we present the notion of connectivity in discrete grids.

The notion of neighborhood in the continuous and in the discrete space is totally different. In the continuous space a neighborhood is the set of all the points whose distance from a given point is shorter than a given value. This set contains an infinity of points if the given value is not null. However, this is not the case in the discrete space as the neighborhood of a point only contains a finite number of points if the given value is finite. Moreover, for any discrete point, we can always find at least one point so that there is no other points between them. In discrete geometry, instead of talking of neighborhood between two points, we use the connection between pixels/voxels. Usually, we say that two pixels/voxels are $m$-connected. The variable $m$ depends on the dimension of the discrete space and the grid that we are considering. Considering the 2D discrete space tile by squares, the variable $m$ can take the values of 4 or 8 as illustrated in Figures 2.9(a) and (b). Two pixels are 4-connected if they share an edge and they are 8-connected if they share an edge or a vertex. If we consider the hexagonal grid, two pixels are 6-connected or not connected as illustrated on Figure 2.10.

Regarding the 3D case, we only consider the orthogonal grid. In this case, the variable $m$ can take the values of 6, 18 or 26 as illustrated on Figures 2.11(a) (b) and (c). Two voxels are 6-connected if they share a face, they are 18-connected if they share a face or

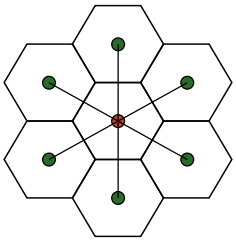


FIGURE 2.10: A pixel and its six 6-connected pixels in a hexagonal grid

an edge and they are 26-connected if they share a face, an edge or a vertex. Note that in 2D, there exist three shape of regular polygons that tile the plane. However, in 3D, the only regular polygon that tile the space is the cube.

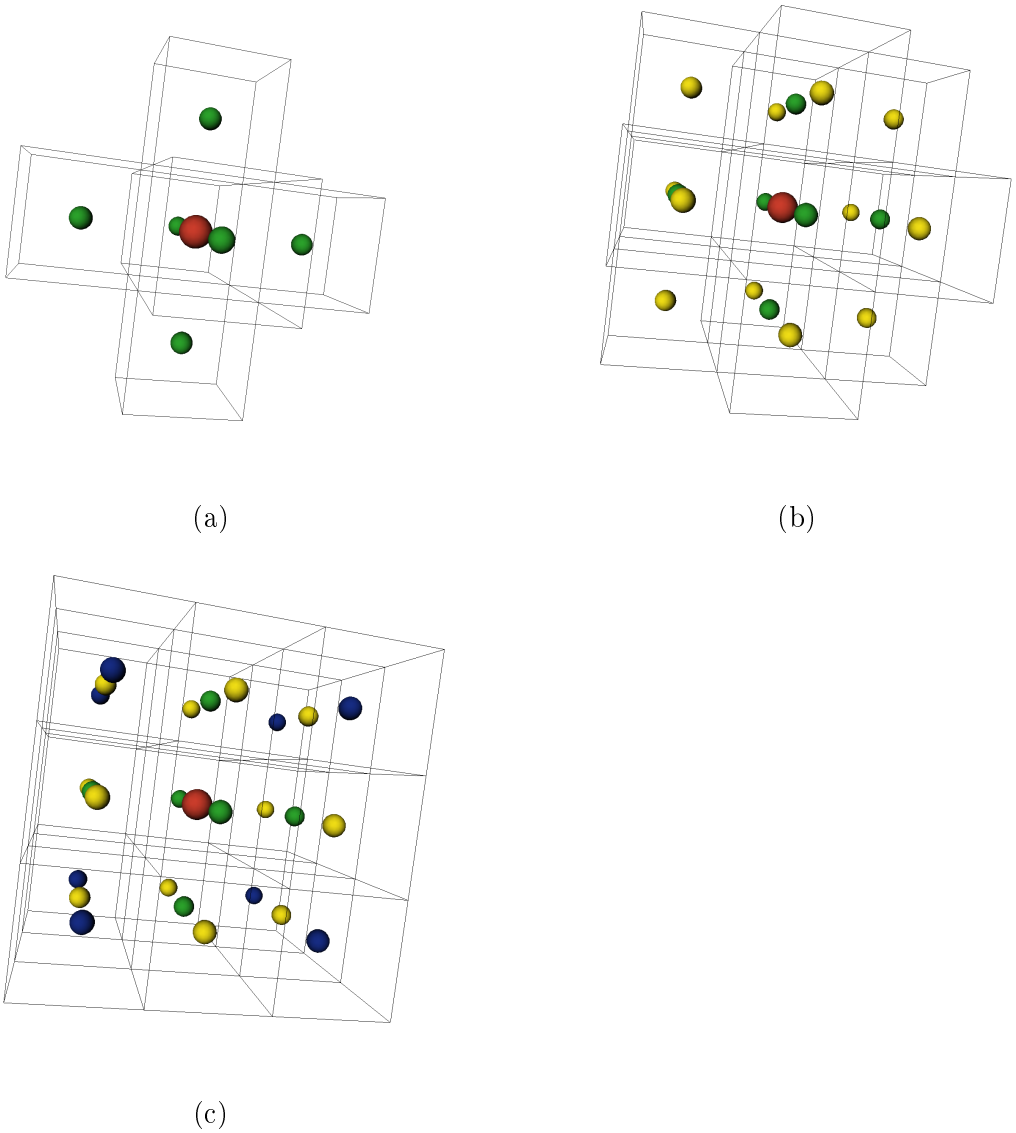
The notions of connectivity explained in this section is required for the study presented hereafter.

### 2.4.2 Methodology

In this section, we analyze two aspects of the discrete rotations: the number of points lost during rotations and the conservation of the neighborhood after rotations.

The first step to study the number of points lost during a rotation on a non square grid is to establish a bijection between the square grid and the other regular grid. The aim is to transform a digital image into a set of points belonging to the regular grid or to transform a set of points of the regular grid into a digital image. Figures 2.12 and 2.13 illustrate the bijections between the square, triangle and hexagonal grid. By using the bijection between the two grids, we convert a digital image of size $1000 \times 1000$ containing white points only into a set of $10^6$ floating points centered on a polygon of the regular grid. Then we apply the standard matrix rotation $\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$ on these set of points and discretize them by using a function, which depends on the regular grid used, that returns a point centered on a regular grid's polygon. We then apply the bijection function that transforms the regular grid into a squared grid, count the number of remaining white points and compare that number to the original white points number.

To study the conservation of the neighborhood after a rotation, we first generate a set of points containing all the points of a square grid. Then, we generate a table that contains for each points four or eight corresponding neighbors. The next step converts the set of points into an-other regular grid, applies the rotation transformation on it and converts the result into a square grid. Finally, from the obtained results, we generate a second table similar to the first one and compare for each point how many neighbors from the first table are remaining in the second table. Here we compare the 4-connectivity and



(a) (b)

(c)

FIGURE 2.11: A voxel and (a) its six 6-connected voxels, (b) its eighteen 18-connected voxels and (c) its twenty-six 26-connected voxels in a cubic grid.

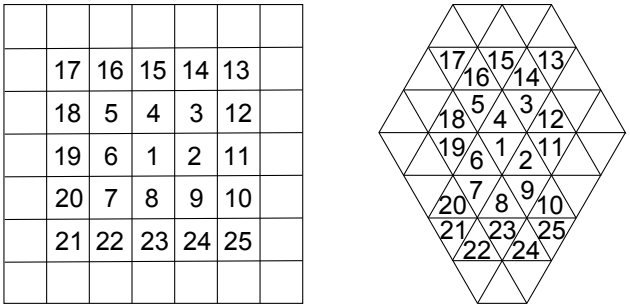



FIGURE 2.12: Conversion from a square grid into a triangle grid

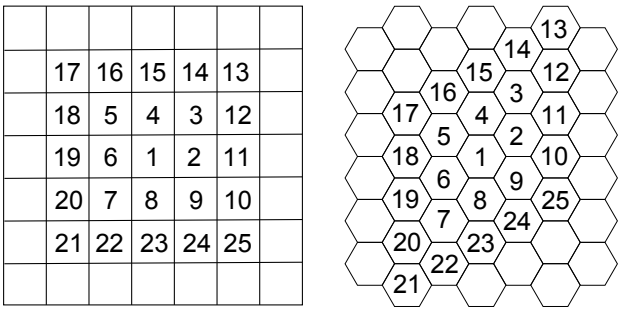



FIGURE 2.13: Conversion from a square grid into a hexagonal grid

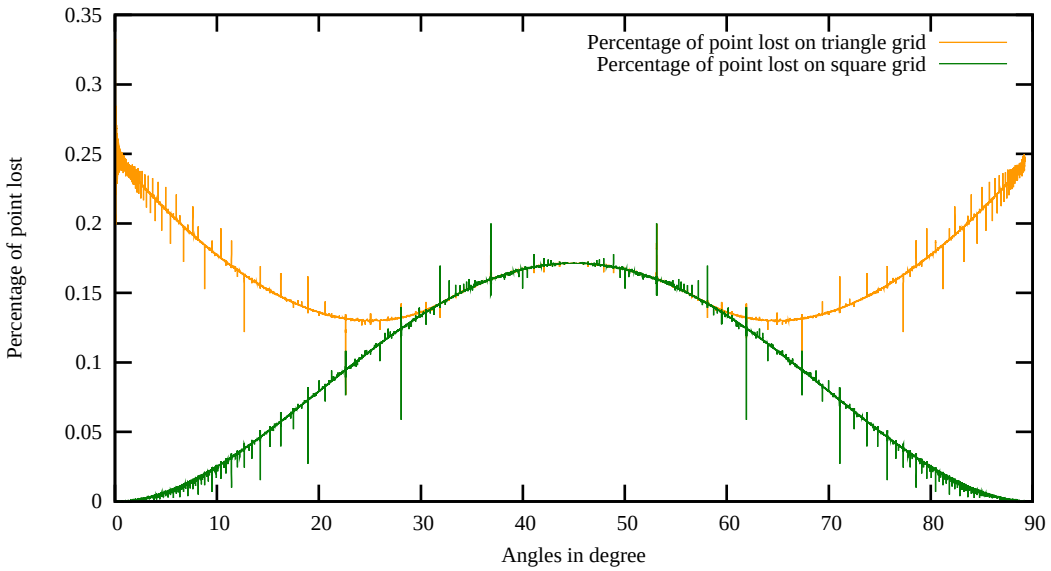



FIGURE 2.14: The percentage of points remaining after a rotation in a triangular grid.

the 8-connectivity after a rotation in every kind of regular grids because the comparison of remaining connectivity between a 4/8-connected grid and a 6 connected grid is not relevant.

### 2.4.3 Experimental result

#### Results on triangle grid

Figure 2.14 presents the results obtained on a rotation using the triangle grid. We compare them with the results obtained in Figure 2.3. First, the rotation using triangle grid always lose at least 13% of points after a rotation. This result is globally worse than the results obtained on a square grid. However, for angles between 32° and 58°, the results are almost the same than the results obtained on a square grid.

Figures 2.15 (a) and (b) show the average conservation of neighborhood depending on the rotation angle on a triangle grid and compare them with the result obtained on

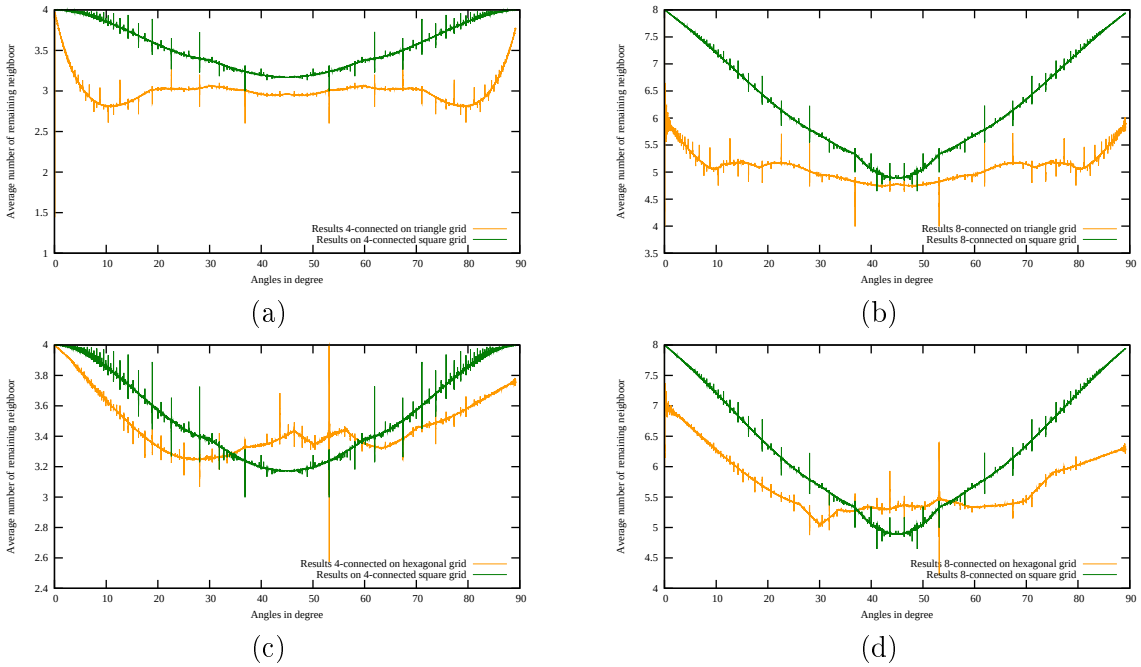



FIGURE 2.15: Comparison of the average conservation of neighborhood during a rotation angle between the square grid and (a) 4-connectivity on a triangle grid, (b) 8-connectivity on a triangle grid,(c) 4-connectivity on a hexagonal grid and (d) 8-connectivity on a hexagonal grid.

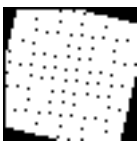
square grid. In the case of 4-connectivity, we can see that rotations on triangle grids keep approximately 3 neighbors during rotation while rotations on square grid keep, in the worst case, approximately 3.2 neighbors. The same results are observed in the case of 8-connectivity with respectively 4.9 and 5 neighbors.

We can conclude that rotation on triangle grid has no interest regarding the number of points lost or in the connectivity's preservation.

### Results on Hexagonal grid

Figure 2.16 presents the results obtained on rotations using the hexagonal grid. We compare them with the result obtained in Figure 2.3. First, Figure 2.16 seems to be the contrary of Figure 2.14, this impression is probably due to the fact that the triangle grid is the dual of the hexagonal grid.

We can see that the rotations on hexagonal grid rarely lose more than $9,5\%$ of points. This result is really better than the results obtained on square grid since on the square grid we lost more that 15% of points between $33°$ and $57°$. However, for angles lower than $20°$ or greater than $70°$, rotations on square grid give slightly better results regarding the number of points lost.



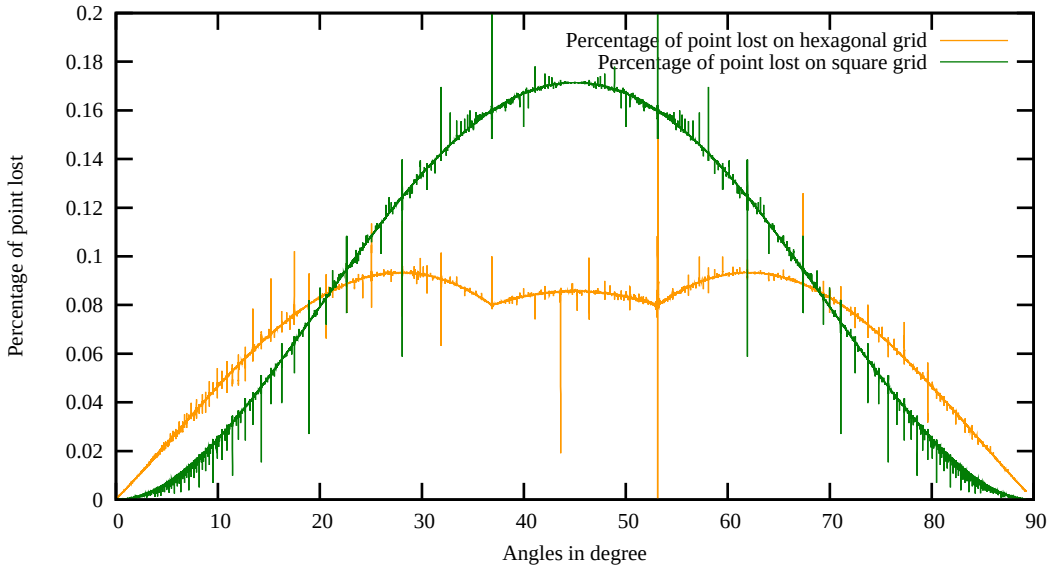



FIGURE 2.16: The percentage of points remaining after rotation in a hexagonal grid.

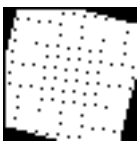
Figures 2.15 (c) and (d) show the average conservation of neighborhood depending on the rotation angle on a hexagonal grid and compare them with the result obtained on square grid. The conservation of neighborhood on the hexagonal grid is not symmetrical around the angle 45°. We do not have any convincing explanation for this lack of symmetry. Around $53^circ$ we can see that a low average neighborhood is immediately and suddenly followed by a high average neighborhood. Since this event appears on Figures 2.15 (c) and (d), we believe that it may be caused by a rounding problem during computation around a particular angle.

We can see that the results obtained on neighborhood using hexagonal grid are globally worse than the results on square grid. This is probably due to the fact that the hexagonal grid counts only six neighbors instead of four or height. However, for the angle between 33° and 58° for the 4-connectivity and between 37° and 54° for the 8-connectivity, rotations on hexagonal grid give slightly better results.

Finally, regarding the number of points lost during rotations, we see that rotations on hexagonal grid globally give better results, than the two other regular grids. We also see, that for 4 and 8-connectivity, results around the angle of 45° are better. Thus, we can conclude that rotations on hexagonal grid globally give better results than rotations on square grid.

### Visual result

In this paragraph, we present the result of rotations for three kind of regular grids. They illustrate the results given above that are theoretical results that cannot give a good idea to the problems of double points and connectivity encountered. These pictures are



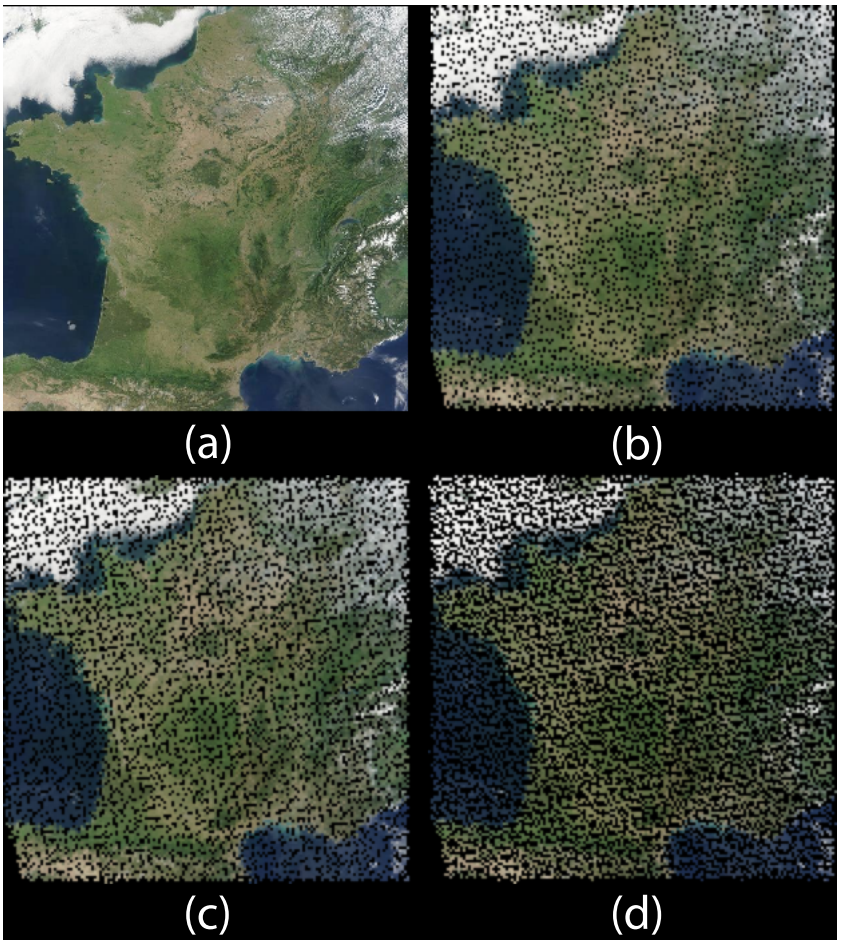


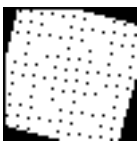

FIGURE 2.17: Result of four rotations of the image (a) by an angle of 15° (b) using a square grid, (c) using a hexagonal grid and (d) using a triangle grid.

the results of the composition of four rotations of a given angle for each regular grid. Figure 2.17 shows that for an angle of 15°, the rotation on square grid keep a little bit more points than the rotation on hexagonal grid. The curve and line bone between French lands and the sea seem to be better preserved by the rotation on square grid. Figures 2.18 and 2.19 shows that rotations on hexagonal grid keep more points than rotations on other grids. However, in both figures, we see that the lines and the curves are better preserved by rotation on square grid.

We can conclude that rotations on hexagonal grids are useful to keep more points during rotations, however rotations on square grids are better for visual consistency.

## 2.5 Conclusion

In this Chapter, we have introduced the problematic of rotations in the discrete plane. We have shown that existing discrete rotations cannot have all the properties of the



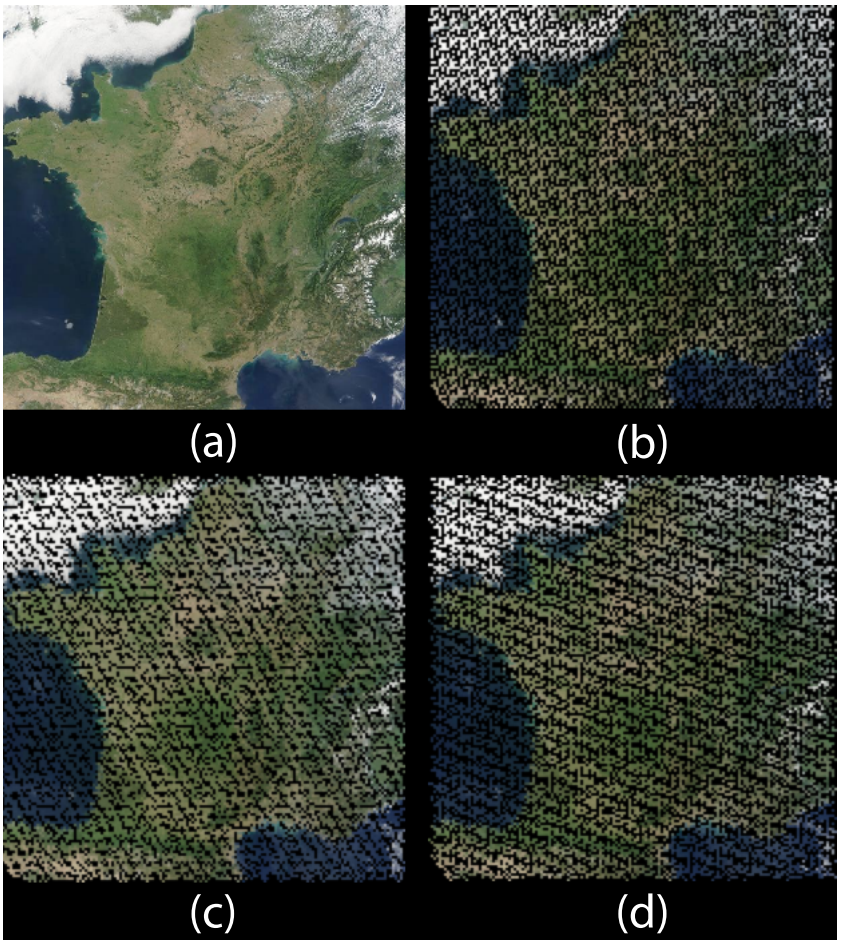
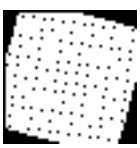



FIGURE 2.18: Result of four rotations of the image (a) by an angle of 30° (b) using a square grid, (c) using a hexagonal grid and (d) using a triangle grid.

continuous rotation, and according to Conjecture 1 we strongly believe that a discrete rotation equivalent to Euclidean rotation cannot be designed. As explained in introduction, we want to develop tools for computer vision. Then we want to design a rotation that is DER-like and discrete according to Definition 2.2. In Chapter 3, we define the hinge angles and introduce a discrete rotation based on hinge angle and a method to estimate for a pair of digital images the rotation that transforms the first image into the second one.



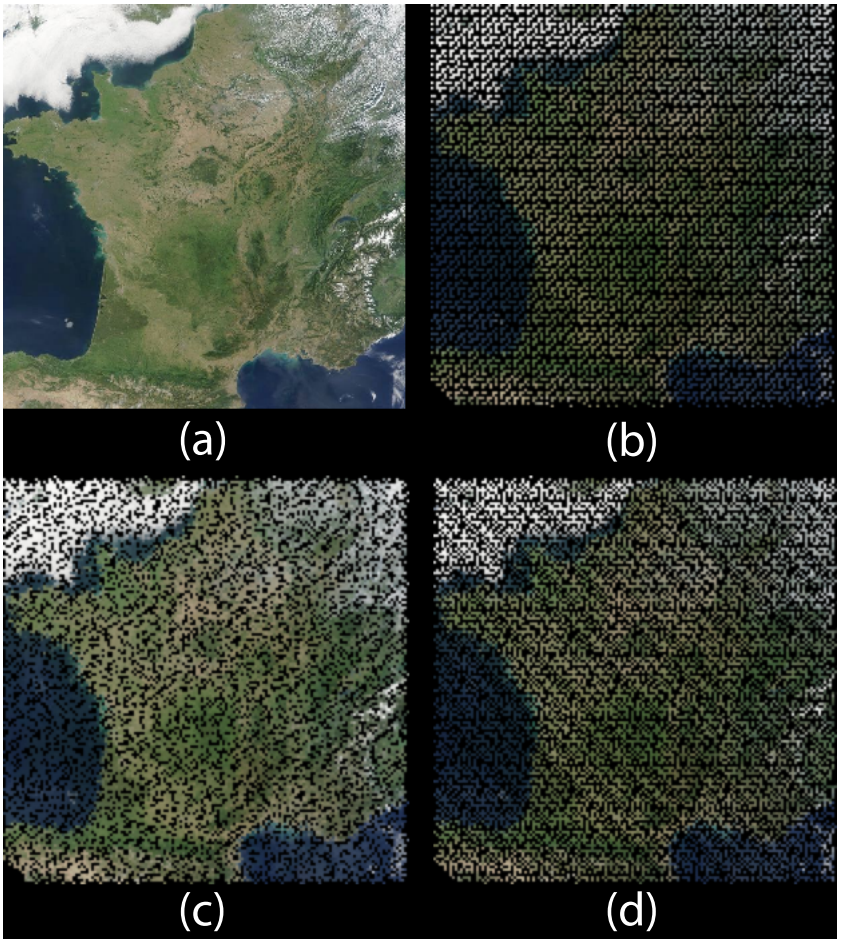
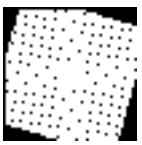



FIGURE 2.19: Result of four rotations of the image (a) by an angle of 45° (b) using a square grid, (c) using a hexagonal grid and (d) using a triangle grid.



# Chapter 3

# 2D rotations in a discrete space using hinge angles

## 3.1 Introduction

Rotations in the discrete plane are required in many applications for image computation such as image matching, construction of mosaic images [16]. At the moment, the most popular method to estimate the rotation angle is to approximate the rotation matrix by minimizing errors [16].

In the continuous plane, the Euclidean rotation is well-defined and possesses the property of bijectivity. This implies that for two angles $\gamma_1, \gamma_2$ and a set of points $A$, if the Euclidean rotation with angle $\gamma_1$ applied to $A$ gives the same result as the Euclidean rotation with angle $\gamma_2$ applied to $A$, then we have $\gamma_1 = \gamma_2$.

In the discrete plane, however, two points resulting of two different Euclidean rotations (with two different angles) of the same grid point can be discretized in the same pixel, i.e. discretized as the same grid point. For this reason, two different angles can give the same rotation result for a set $A$ of grid points[1]. In other words, we can define a set of *admissible rotation angles* $S$ so that any angle in $S$ gives the same rotation result for $A$. Note that $S$ depends on $A$. The two most interesting angles in $S$ are the lower and the upper bounds. Indeed, with only these two angles we can deduce the other angles in $S$. To identify the exact bounds, we should not involve any computation error. Thus, we work with discrete geometry tools, which guarantee computation avoidance with real numbers. Moreover, because we assume that our data are discretized from continuous

[1] Accordingly, DER is not bijective.

images of an object, we enforce the property that the discrete rotation[2] between two different sets of grid points gives the same result as DER.

Some works on discrete rotations already exist. The most widely used discrete rotation at the beginning is the CORDIC algorithm [20]. The CORDIC algorithm uses a sequence of fractions for an approximation of $\pi$. It multiplies/adds fractions in this sequence to approximate values of sine and cosine. Thus multiple approximations lead to little differences between results of DER and those of CORDIC. Note that CORDIC algorithm is not a discrete rotation according to Definition 2.2. As presented in Chapter 2, Andres described in [2, 6] some discrete rotations such as the rotation by discrete circles, the rotation by Pythagorean lines or the quasi-shear rotation. Computation executed during these rotations are exact. However, we have shown in Chapter 2 that these rotation algorithms do not give the same results as DER.

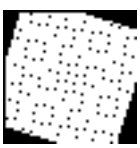
On the other hand, Nouvel and Rémila proposed in [21] another discrete rotation based on hinge angles, which gives the same results as DER. It is known that a sequence of hinge angles is a set of particular angles determined by a digital image in the sense that any angle between two consecutive hinge angles gives the identical rotated digital image. This means that hinge angles correspond to the discontinuity of DER. Nouvel and Rémila showed that each hinge angle is represented by an integer triplet, so that any discrete rotation of a digital image is realized only with integer calculation. Because their algorithm gives the same rotation results as DER, we see that hinge angles represented by integer triplets give sufficient information for executing any digital image rotation.

In this chapter, we introduce the hinge angles and give their properties. Then we present a method to perform a discrete rotation using hinge angles. Finally, we propose a discrete method to find the lower and upper bounds of admissible rotation angles. Our method uses hinge angles because this algorithm is DER-like and because they allow exact computations. The input data of our method is two sets of grid points where point correspondences across the two sets are known. The output is two hinge angles that give the lower and the upper bounds of the admissible rotation angles for the two sets.

## 3.2 Hinge angles

We consider grid points in $\mathbb{Z}^2$ as the centers of pixels and rotate them in such a way that the rotation center has integer coordinates. Hinge angles are particular angles that make some points in $\mathbb{Z}^2$ rotated to points at the frontier between adjacent pixels. In this

[2] A discrete rotation is a rotation designed for the discrete space. It transforms a set of grid points into another set of grid points.



section, we give the definition of hinge angles and their properties related to Pythagorean angles.

### 3.2.1 Definition of hinge angles

Let $\vec{p} = (x, y)$ be a point in $\mathbb{R}^2$. We say that $\vec{p}$ has a semi-integer coordinate if $x + \frac{1}{2} \in \mathbb{Z}$ or $y + \frac{1}{2} \in \mathbb{Z}$. The set of points each of which has a semi-integer coordinate is called the half-grid and is denoted by $\mathscr{H}$. Thus, $\mathscr{H}$ represents the set of points on the frontiers of all pixels whose centroids are points in $\mathbb{Z}^2$.

**Definition 3.1.** An angle $\alpha$ is called a hinge angle if at least one point in $\mathbb{Z}^2$ exists such that its image by the Euclidean rotation with $\alpha$ belongs to $\mathscr{H}$.

Because $\mathscr{H}$ can be seen as the discontinuity of the rounding function, hinge angles can be regarded as the discontinuity of the discretized Euclidean rotation. In other words, hinge angles determine a transit of a grid point from a pixel to its adjacent pixel during the rotation.

The following theorem is important because it shows that we can represent every hinge angle with three integers.

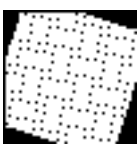
**Theorem 3.2** ([21]). *An angle $\alpha$ is a hinge angle for a grid point $(P, Q) \in \mathbb{Z}^2$ if and only if there exists $K \in \mathbb{Z}$ such that*

$$2Q\cos\alpha + 2P\sin\alpha = 2K + 1. \tag{3.1}$$

Geometrically, a hinge angle $\alpha$ is formed by two rays going through $(P, Q)$ and a half-grid point $(K + \frac{1}{2}, \lambda)$ where the two rays share the origin as their endpoints as shown in Figure 3.1 (left). This theorem indicates that all calculations related to hinge angles can be done only with integers. Hereafter, $\alpha$ indicates a hinge angle.

We denote by $\alpha(P, Q, K)$ the hinge angle generated by an integer triplet $(P, Q, K)$. Setting $\lambda = \sqrt{P^2 + Q^2 - (K + \frac{1}{2})^2}$, we easily derive the following equations from (3.1) and Figure 3.1 (left),

$$\cos\alpha = \frac{P\lambda + Q(K + \frac{1}{2})}{P^2 + Q^2}, \qquad \sin\alpha = \frac{P(K + \frac{1}{2}) - Q\lambda}{P^2 + Q^2}. \tag{3.2}$$



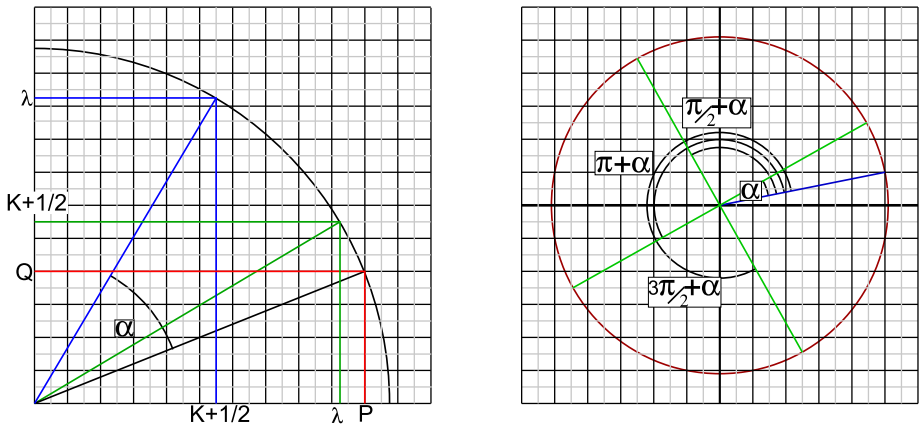



FIGURE 3.1: A hinge angle $\alpha(P, Q, K)$ (left) and four symmetrical hinge angles (right).

Note that we have a case where a half-grid point is $(\lambda, K + \frac{1}{2})$ instead of $(K + \frac{1}{2}, \lambda)$. In such a case, the above equations become

$$\cos\alpha = \frac{Q\lambda + P(K + \frac{1}{2})}{P^2 + Q^2}, \qquad \sin\alpha = \frac{P\lambda - Q(K + \frac{1}{2})}{P^2 + Q^2}. \tag{3.3}$$

The symmetries on hinge angles are important because it allows us to restrict rotations in the first quadrant of the circle such that $\alpha \in [0, \frac{\pi}{2}]$.

**Corollary 3.3.** *Each triplet* $(P, Q, K)$ *corresponds to four symmetrical hinge angles* $\alpha + \frac{\pi k}{2}$ *where* $k = 0, 1, 2, 3$.

Figure 3.1 (right) gives an example of Corollary 3.3. To distinguish the case of $(K+\frac{1}{2}, \lambda)$ from that of $(\lambda, K + \frac{1}{2})$, we change the sign of $K$; we use $\alpha(P, Q, K)$ for the case of $(K+\frac{1}{2}, \lambda)$, and $\alpha(P, Q, -K)$ for the case of $(\lambda, K+\frac{1}{2})$. Because the symmetries allow us to restrict $\alpha$ to the range $[0, \frac{\pi}{2}]$, as mentioned above, we may assume that $K$ is positive.

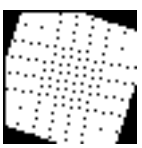
### 3.2.2 Properties related to Pythagorean angle

Because hinge angles are strongly related to Pythagorean angles, certain properties of Pythagorean angles are required to prove some properties of hinge angles. Thus, we first give the definition of Pythagorean angles and their properties.

**Definition 3.4.** An angle $\theta$ is called Pythagorean if and only if both its cosine and sine belong to the set of rational numbers $\mathbb{Q}$.

We can deduce from Definition 3.4 that a Pythagorean angle $\theta$ is represented by an integer triplet $(a, b, c)$ such that

$$\cos\theta = \frac{a}{c}, \qquad \sin\theta = \frac{b}{c}. \tag{3.4}$$



In the following, $\theta$ indicates a Pythagorean angle. The lemma below for Pythagorean angles is well known.

**Lemma 3.5.** *Let $(a, b, c)$ be an integer triplet generating a Pythagorean angle where $|c| = \max\{|\ a\ |, |\ b\ |, |\ c\ |\}$. If $\gcd(a, b, c) = 1$, then $c$ is odd.*

If $\gcd(a, b, c) = i$, then $\gcd(\frac{a}{i}, \frac{b}{i}, \frac{c}{i}) = 1$ and the triplet of integers $(\frac{a}{i}, \frac{b}{i}, \frac{c}{i})$ generates the same Pythagorean angle as $(a, b, c)$.

**Theorem 3.6.** *Let $E_h$ be the set of hinge angles and $E_p$ be the set of Pythagorean angles. Then we have $E_h \bigcap E_p = \emptyset$.*

**Proof.** 1. Assume that there is an angle $\alpha$ such that $\alpha \in E_h$ and $\alpha \in E_p$. Since $\alpha \in E_p$, we can find an integer triplet $(a, b, c)$ generating $\alpha$ where $\gcd(a, b, c) = 1$. By substitution of (3.4) in (3.1), we obtain

$$2\frac{Qa + Pb}{c} = 2K + 1, \tag{3.5}$$

from which we derive $2\frac{Qa+Pb}{c} \in \mathbb{Z}$. Because we know that $c$ is odd according to Lemma 3.5, we obtain $\frac{Qa+Pb}{c} \in \mathbb{Z}$. This means that $\frac{2(Qa+Pb)}{c}$ is even, while $2K + 1$ is always odd, which leads to a contradiction.

This theorem shows that it is not possible to rotate a point $(i, j) \in \mathbb{Z}^2$ to a point $(x, y)$ such as $x = i + \frac{1}{2}$, $y = j + \frac{1}{2}$, if the angle of the rotation is a hinge angle.

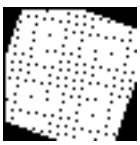
The next theorem shows an interesting relation between hinge angles and Pythagorean angles.

**Theorem 3.7** ([4]). *Let $\theta$ be a Pythagorean angle and $\alpha$ be a hinge angle. The angle $\alpha' = \alpha + \theta$ is a hinge angle.*

## 3.3 Rotation by hinge angles

As explained in Chapter 2, as our work is focused on the discrete space, the rotation of a grid point by two different angles can give the same result. Namely, two different angles give the same result after the rotation of a grid point followed by discretization. Generally, there is a range of angles in which the same result is obtained. We thus define *admissible rotation angles*, abbreviated hereafter by ARA, to represent this range of angles.

In this section, we propose a method for computing the lower bound $\alpha_{\text{inf}}$ of ARA for a given digital image from a given angle. Note that with minor modifications, this method can also find the upper bound $\alpha_{\text{sup}}$ of ARA. Thus applying any rotation to the given



**Algorithm 1** Function for the lower bound rotation angle for a point.

**Require:** A point $\vec{p}(P,Q)$, a Pythagorean angle $\theta$
**Ensure:** A hinge angle $\alpha(P,Q,K)$
  $K_{\max} \leftarrow \lfloor\sqrt{P^2+Q^2}-1\rfloor$;
  $K_{\min} \leftarrow 0$;
  $K \leftarrow \lfloor\frac{K_{max}+K_{min}}{2}\rfloor$;
  **while** $K_{\max} - K_{\min} \neq 1$ **do**
    **if** $\alpha(P,Q,K) > \theta$ **then**
      $K_{\max} = K$;
    **else**
      $K_{\min} = K$;
    **end if**
    $K = \lfloor\frac{K_{\max}+K_{\min}}{2}\rfloor$;
  **end while**
  **return** $\alpha(P,Q,K)$;

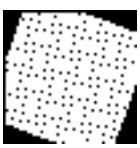
digital image with an angle between $\alpha_{\text{inf}}$ and $\alpha_{\text{sup}}$ gives the same result. We note that both $\alpha_{\text{inf}}$ and $\alpha_{\text{sup}}$ are hinge angles.

Our input is a Euclidean angle. However, we can replace it by a Pythagorean angle as there is a method with linear time complexity $\mathrm{O}(m)$ to approximate a given Euclidean angle with a Pythagorean angle with a precision of $\frac{1}{10^m}$ [15], where $m$ is a fixed integer that represents the quality of approximation. Below, we assume that a Pythagorean angle is given as in [21].

Nouvel and Rémila presented a method for computing all possible hinge angles for a grid point or a pixel in a digital image [21]. Their method can be used for finding the hinge angle that is the lower bound of the admissible rotation angles. Its time complexity is $\mathrm{O}(n\log(n))$ where $n$ is the number of all hinge angles for a given grid point. Note that $n$ depends on the coordinates of the grid point.

### 3.3.1 Computing the lower bound rotation angle for grid point

For each grid point $\vec{p} = (P,Q) \in \mathbb{Z}^2$, there are less than $n = \lfloor\sqrt{P^2+Q^2}+\frac{1}{2}\rfloor$ different hinge angles in each quadrant [21]. We can compare in magnitude any pair of hinge angles. This means that we have a totally ordered set $\{\alpha(P,Q,K_1), \alpha(P,Q,K_2), ..., \alpha(P,Q,K_n)\}$ of hinge angles in the ascending order where $K_i \in \mathbb{Z}$. Given a Pythagorean angle $\theta$, in order to find the lower bound rotation angle $\alpha(P,Q,K_i)$ such that $\alpha(P,Q,K_i) < \theta < \alpha(P,Q,K_{i+1})$, we use a binary search. The binary search allows us to find $\alpha(P,Q,K_i)$ in $\mathrm{O}(\log(n))$, providing that we can compare a hinge angle with a Pythagorean angle in a constant time. The algorithm is described in Function 1. Note that thanks to Theorem 3.9 a Pythagorean angle for the input can be replaced by a hinge angle.



The following theorem shows that the comparison between a hinge angle and a Pythagorean angle is executed in a constant time.

**Theorem 3.8.** *Let $\alpha$ be a hinge angle and $\theta$ be a Pythagorean angle. We can check whether $\alpha > \theta$ in a constant time with integer calculation.*

**Proof.** 2. Let $\alpha(P, Q, K)$ be a hinge angle in $[0, \frac{\pi}{2}]$ and $\theta(a, b, c)$ be a Pythagorean angle in $[0, \frac{\pi}{2}]$. From (3.2) and (3.4), we obtain

$$\cos\alpha - \cos\theta = \frac{Q(K+\frac{1}{2}) + P\lambda}{P^2+Q^2} - \frac{a}{c}.$$

If $\theta$ is greater than $\alpha$, $\cos\alpha - \cos\theta > 0$. Thus

$$cQ(2K+1) - 2a(P^2+Q^2) > -2cP\lambda. \tag{3.6}$$

Since we know that $c, P, \lambda$ are positive, the right-hand side of (3.6) is always negative. Thus, if the left-hand side of (3.6) is not negative, then $\theta > \alpha$. Otherwise, we take the square of (3.6), so that we only have to check whether the following inequality holds:

$$\left[cQ(2K+1) - 2a(P^2+Q^2)\right]^2 < 4c^2P^2\lambda^2. \tag{3.7}$$

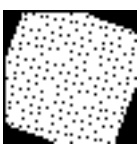
Note that because $\lambda = \sqrt{P^2+Q^2-(K+\frac{1}{2})^2}$, we see that $4\lambda^2$ in the right-hand side of (3.7) contains only integer values. Therefore, we can verify (3.7) with integer calculation. If it is true, $\theta > \alpha$; otherwise, $\alpha > \theta$. Note that because of Theorem 3.6, it is impossible to obtain $\theta = \alpha$.

We claim that this comparison of a hinge angle and a Pythagorean angle is executed in a constant time because even in the worst case, we only have to check two equations (3.6) and (3.7).

We mention the importance of the rotation with angle $\frac{\pi}{2}$ and its multiplications. In fact, if the angle of a rotation is equal to $\frac{\pi}{2}, \pi, \frac{3\pi}{2}$, we just have to flip $x$ and/or $y$-coordinates by changing their signs. It gives the justification that we can restrict the input angle $\theta$ to $0 < \theta < \frac{\pi}{2}$.

### 3.3.2 Computing the lower bound rotation angle for a set of grid points

In this subsection, we present an algorithm for computing the lower bound rotation angle from a given Pythagorean angle $\theta$ for a digital image consisting of $m$ grid points $A$. The output is a triplet of integers that represents the lower bound rotation angle for $A$. We note that the lower bound rotation is a hinge angle.



The algorithm computes all hinge angles for all points in $A$, and sorts them to keep the largest one. More precisely, we first compute the lower bound rotation angle for the first point of $A$, and store it as a reference. Then, we compute the lower bound rotation angle for the second point in $A$ and compare it with the reference to keep the larger one. After repeating this procedure for all points in $A$, our algorithm returns the lower bound rotation angle $\alpha$ such that $\alpha < \theta$. The time complexity of this algorithm is $\mathrm{O}(m\log(n))$ because we call $m$ times the binary search (Function 1) whose time complexity is $\mathrm{O}(\log(n))$. Function 2 illustrates our algorithm. As shown in Theorem 3.9, the comparison between two hinge angles is realized in a constant time, so that our algorithm does not change the global complexity.

**Theorem 3.9.** *Let $\alpha_1, \alpha_2$ be two hinge angles. We can check whether $\alpha_1 > \alpha_2$ in a constant time and with integer calculation.*

**Proof.** 3. Let $\alpha_1(p,q,k)$ and $\alpha_2(r,s,l)$ be two hinge angles in $[0, \frac{\pi}{2}]$. From (3.3) we obtain

$$\cos\alpha_1 - \cos\alpha_2 = \frac{p(k+\frac{1}{2}) + q\lambda_1}{p^2+q^2} - \frac{r(l+\frac{1}{2}) + s\lambda_2}{r^2+s^2}. \tag{3.8}$$

If $\alpha_2$ is greater than $\alpha_1$, $\cos\alpha_1 - \cos\alpha_2 > 0$. Thus

$$(r^2+s^2)p(2k+1) - (p^2+q^2)r(2l+1) > 2(p^2+q^2)s\lambda_2 - 2(r^2+s^2)q\lambda_1. \tag{3.9}$$

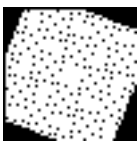
If the left-hand side of (3.9) is negative and the right-hand side of (3.9) is positive, then $\alpha_1 > \alpha_2$. If the left-hand side of (3.9) is positive and the right-hand side of (3.9) is negative, then $\alpha_2 > \alpha_1$. We can easily check the signs of the left-hand side and the right-hand side of (3.9) with integer computation. Note that $p, q, k, r, s, l$ are all positive, and that $(2(p^2+q^2)s\lambda_2)^2$ and $(2(r^2+s^2)q\lambda_1)^2$ contain only integer values.

If the signs of the left-hand side and the right-hand side of (3.9) are the same, we first compute the square of each side and then compare the values to identify which is the greater. For simplicity, we assume that the signs of both sides of (3.9) are positive, and let $A = (r^2+s^2)p(2k+1)$, $B = (p^2+q^2)r(2l+1)$, $C = (r^2+s^2)q$ and $D = (p^2+q^2)s$. Now (3.9) is rewritten by

$$A - B > 2D\lambda_2 - 2C\lambda_1. \tag{3.10}$$

Then we take the square of equation (3.10) to obtain

$$(A-B)^2 - 4(C^2\lambda_1^2 + D^2\lambda_2^2) > -8CD\lambda_1\lambda_2. \tag{3.11}$$



**Algorithm 2** Function for finding the lower bound rotation angle for a digital image.

**Require:** A set of points $A$, a Pythagorean angle $\theta$
**Ensure:** A hinge angle $\alpha$
  $\alpha$ = Function 1 (first point $\vec{p}_1$ of $A$, $\theta$);
  **for all** $\vec{p} \in A \backslash \{\vec{p}_1\}$ **do**
    $\alpha_{\text{temps}}$ = Function 1 ($\vec{p}$, $\theta$);
    **if** $\alpha < \alpha_{\text{temps}}$ **then**
      $\alpha = \alpha_{\text{temps}}$;
    **end if**
  **end for**
  **return** $\alpha$;

If the sign of the left-hand side of (3.11) is positive, we can deduce that $\alpha_2 > \alpha_1$. Otherwise, taking the square of each side gives us

$$\left[(A-B)^2 - 4(C^2\lambda_1^2 + D^2\lambda_2^2)\right]^2 < 64C^2D^2\lambda_1^2\lambda_2^2. \tag{3.12}$$

We note that we can easily verify whether (3.12) is satisfied with integer computation alone. If (3.12) is true, $\alpha_1 < \alpha_2$; otherwise $\alpha_2 < \alpha_1$. The same logic can be applied to the case where the signs of both sides of (3.9) are negative.

This comparison of a pair of hinge angles is executed in a constant time because in the worst case, we only have to check three equations (3.9), (3.11) and (3.12).

### 3.3.3 Digital image rotation by a lower bound rotation angle

This section uses the results obtained in Section 3.3 to present an algorithm for rotating a set of points with a given lower bound rotation angle.

It is already proved in [21] that we can obtain the same result as the DER with respect to the original rotation angle. Note that our input is a lower bound rotation angle and the input of the algorithm presented in Function 1 is a Pythagorean angle. In spite of this difference, we can apply the same algorithm thanks to Theorem 3.9, since we are

**Algorithm 3** Discrete rotation algorithm by a lower bound rotation angle.

**Require:** A set of points $A$, a lower bound rotation angle $\alpha$ *(hinge angle)*
**Ensure:** A set of points $A'$
  **for all** $\vec{p} \in A$ **do**
    $\alpha_1$ = Function 1 ($\vec{p}, \alpha$);
    Move $\vec{p}$ to $(K, \lfloor \lambda + \frac{1}{2} \rfloor)$ or $(\lfloor \lambda + \frac{1}{2} \rfloor, K)$, depending on the sign of $K$ and store the new point in $A'$;
  **end for**
  **return** $A'$;

looking for $K$, which gives the arriving pixel $(K, \lfloor \lambda + \frac{1}{2} \rfloor)$, for each pixel $(P, Q)$. The algorithm is presented in Function 3. It supposes that the center of rotation is the origin.

For each point, our algorithm calls the binary search (Function 1) to find the corresponding hinge angle, which designates its new position. If we consider $n$ as the biggest coordinate of all points in $A$, we can assume that there are less than $4n^2$ points in $A$. Thus we can conclude that the complexity of our algorithm is $\mathrm{O}(n^2 \log(n))$. The first advantage of our method is that it does not require any floating number calculation. The second advantage is that the exact rotation of the set of points is obtained with only an integer triplet. We need neither matrices nor angles to realize the rotation.

## 3.4 Obtaining admissible rotation angles from two digital images

Let us assume that a set of grid points in the first image and its corresponding set in the second image are given: $A = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_l\}$ and $B = \{\vec{q}_1, \vec{q}_2, ..., \vec{q}_l\}$ are given where $\vec{p}_i$ corresponds to $\vec{q}_i$. Given $A$ and $B$, we obtain a hinge angle pair $\{\alpha_{\inf}, \alpha_{\sup}\}$. This pair of hinge angles is the lower and the upper bounds of the ARA. Therefore each $\gamma$ such that $\alpha_{\inf} \leq \gamma < \alpha_{\sup}$ is consistent with the point correspondences between $A$ and $B$. Hereafter, we assume that $A$ is the original point set and $B$ is the rotated point set by angle $\gamma$. In this section, we show how to obtain the ARA from $A$ and $B$.

### 3.4.1 Setting Rotation Centers

For any rotation, we need to set a rotation center. Without a loss of generality, we may choose any grid point in a digital image for the rotation center. Indeed, if we apply two rotations on the same image with the same angle but two distinct centers of rotation, the difference between the two result is captured by a translation. Assuming that rotation centers for $A$ and $B$ are $\vec{p}_1$ and $\vec{q}_1$[3] respectively, we define two translation functions $\mathscr{T}_A$ and $\mathscr{T}_B$ such that

$$\begin{aligned} \mathscr{T}_A(\vec{p}_i) &= \vec{p}_i - \vec{p}_1, \\ \mathscr{T}_B(\vec{q}_i) &= \vec{q}_i - \vec{q}_1, \end{aligned}$$

for all $\vec{p}_i \in A, \vec{q}_i \in B$. We can regard the origin as the rotation centers after these translations. Hereafter, we assume that these translations have already been applied in order to obtain $A = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_l\}$ and $B = \{\vec{q}_1, \vec{q}_2, ..., \vec{q}_l\}$.

[3] $\vec{p}_1$ and $\vec{q}_1$ can have different coordinates since they are a pair of point in correspondence.

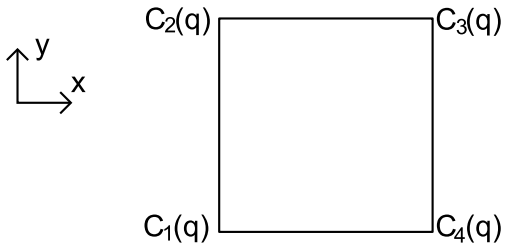



FIGURE 3.2: The corners of $\mathscr{H}(\vec{q})$, namely, four corners of a pixel around $\vec{q}$.

### 3.4.2 Computing lower and upper bounds of rotation angles from two corresponding point pairs

In this subsection, we consider a special case of two corresponding point pairs.

We let $A = \{\vec{p}_1, \vec{p}_2\}$ and $B = \{\vec{q}_1, \vec{q}_2\}$ where $\vec{p}_i = (P_i, Q_i)$ and $\vec{q}_i = (R_i, S_i)$. We then define a circle $\mathscr{C}(\vec{p}_2)$ going through $\vec{p}_2$ whose center is $\vec{p}_1$. Thus the radius of $\mathscr{C}(\vec{p}_2)$ is $r = d(\vec{p}_1, \vec{p}_2)$ where $d(\vec{p}_1, \vec{p}_2)$ is the Euclidean distance between $\vec{p}_1$ and $\vec{p}_2$.

Let us define the half-grid $\mathscr{H}(\vec{q}_2)$ around $\vec{q}_2$:

$$\begin{aligned}\mathscr{H}(\vec{q}_2) = \{(x,y) \in \mathscr{H} \quad : \quad & S_2 - \frac{1}{2} \leq y \leq S_2 + \frac{1}{2} \text{ if } x = R_2 \pm \tfrac{1}{2}, \\ & R_2 - \frac{1}{2} \leq x \leq R_2 + \frac{1}{2} \text{ if } y = S_2 \pm \tfrac{1}{2}\}.\end{aligned}$$

We set $\vec{p}_1$ and $\vec{q}_1$ to be the rotation centers. Then, we need to detect intersections between $\mathscr{C}(\vec{p}_2)$ and $\mathscr{H}(\vec{q}_2)$ in order to find a hinge angle pair. We thus investigate which corners of $\mathscr{H}(\vec{q}_2)$ are inside of $\mathscr{C}(\vec{p}_2)$.

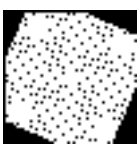
Setting four corners of $\mathscr{H}(\vec{q}_2)$ to be $C_1(\vec{q}_2) = (R_2 - \frac{1}{2}, S_2 - \frac{1}{2}), C_2(\vec{q}_2) = (R_2 - \frac{1}{2}, S_2 + \frac{1}{2}), C_3(\vec{q}_2) = (R_2 + \frac{1}{2}, S_2 + \frac{1}{2}), C_4(\vec{q}_2) = (R_2 + \frac{1}{2}, S_2 - \frac{1}{2})$ as shown in Figure 3.2, we define a binary function $\mathscr{F}$:

$$\mathscr{F}(C_i(\vec{q}_2)) = \begin{cases} 1 & \text{if } C_i(\vec{q}_2) \text{ is inside of } \mathscr{C}(\vec{p}_2), \\ 0 & \text{otherwise.} \end{cases}$$

In order to obtain $\mathscr{F}(C_i(\vec{q}_2))$ with integer calculation, we compare each of $\|(2(R_2 \pm \frac{1}{2}), 2(S_2 \pm \frac{1}{2}))\|^2$ with $(2r)^2$. Note that we may assume that $\mathscr{C}(\vec{p}_2)$ and $\mathscr{H}(\vec{q}_2)$ always intersect with each other. This is because no intersection between $\mathscr{C}(\vec{p}_2)$ and $\mathscr{H}(\vec{q}_2)$ indicates that $\vec{p}_2$ and $\vec{q}_2$ are not corresponding.

The following lemmas are needed to prove Theorem 3.12 below.

**Lemma 3.10.** *For a circle $\mathscr{C}(\vec{p}_2)$ centered on $\vec{p}_1$, any $C_i(\vec{q}_2) = (R_2 \pm \frac{1}{2}, S_2 \pm \frac{1}{2})$ cannot be on $\mathscr{C}(\vec{p}_2)$ for $i \in \{1,2,3,4\}$ where $R_2, S_2 \in \mathbb{Z}$.*



**Proof.** 4. Let $r$ be the radius of $\mathscr{C}(\vec{p}_2)$. Because $\mathscr{C}(\vec{p}_2)$ goes through $\vec{p}_2$, $r^2 \in \mathbb{Z}$. Let us assume that $C_i(\vec{q}_2)$ is on $\mathscr{C}(\vec{p}_2)$. Thus the Euclidean distance between the origin and $C_i(\vec{q}_2)$ is equal to $r$. This indicates that $r^2 = (R_2 \pm \frac{1}{2})^2 + (S_2 \pm \frac{1}{2})^2$. However, this contradicts the above fact that $r^2 \in \mathbb{Z}$.

**Lemma 3.11.** *Let $\mathcal{D}$ be a line that belongs to $\mathscr{H}$. If $\mathscr{C}(\vec{p}_2)$ is a circle centered on $\vec{p}_1$. Then, the number of distinct intersections of $\mathscr{C}(\vec{p}_2)$ and $\mathcal{D}$ is two or zero.*

**Proof.** 5. Let $\vec{p}_2 = (P_2, Q_2) \in \mathbb{Z}^2$ and the equation representing $\mathcal{D}$ be $x = i + \frac{1}{2}$ where $i \in \mathbb{Z}$. Letting $(x_1, y_1)$ be the coordinates of the intersecting point of $\mathcal{D}$ and $\mathscr{C}(\vec{p}_2)$. Then we have

$$\begin{cases} x_1^2 + y_1^2 &= P_2^2 + Q_2^2, \\ x_1 &= i + \frac{1}{2}. \end{cases}$$

From these equations, we obtain $y_1^2 = P_2^2 + Q_2^2 - (i + \frac{1}{2})^2$. Because $P_2^2 + Q_2^2 - (i + \frac{1}{2})^2$ does not belong to $\mathbb{Z}$, $y_1$ cannot be equal to 0. Therefore there are two distinct solutions for $y_1$ if $P_2^2 + Q_2^2 - (i + \frac{1}{2})^2 > 0$; no solution otherwise. Similar discussion can be done for the case of $y_1 = j + \frac{1}{2}$ where $j \in \mathbb{Z}$.

**Theorem 3.12.** *If two points $\vec{p}_2$ and $\vec{q}_2$ are corresponding, the circle $\mathscr{C}(\vec{p}_2)$ and the half-grid $\mathscr{H}(\vec{q}_2)$ always have two distinct intersections.*

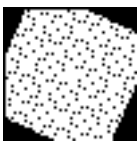
**Proof.** 6. In general, if a circle intersects with a square and the center of the circle is not inside the square, we have $1, 2$ or $4$ intersections. Having just one intersection means that the circle goes through one corner of the square or the circle is tangential to a half-grid. Lemma 3.10 shows that the circle cannot go through one corner. Lemma 3.11 shows that the circle cannot be tangential to any half-grid.

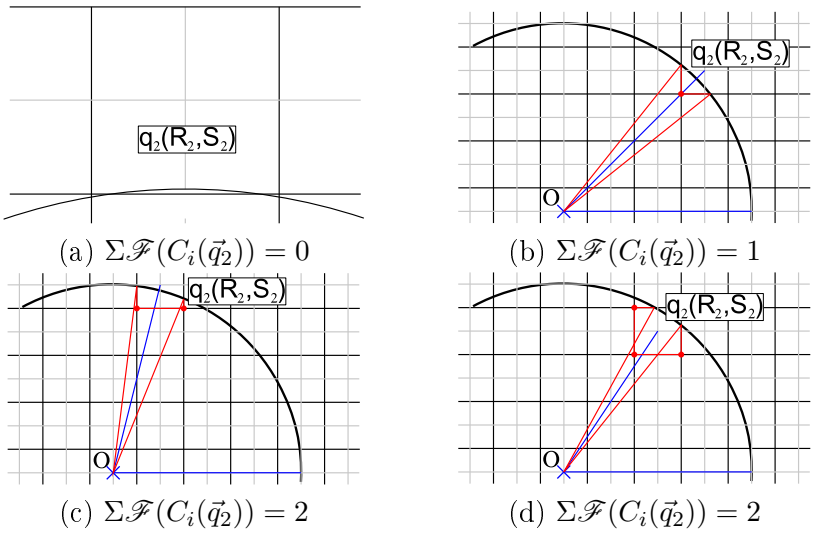
Let $\mathcal{C}$ be a circle centered on the origin and going through the point $\vec{p} = (p_x, p_y)^\top \in \mathbb{Z}^2$. We assume that there is a pixel with four intersections with $\mathcal{C}$. If such a pixel exists, one of its coordinates must be equal to zero and the intersections are symmetric regarding to the $x$-axis or the $y$-axis (according to the null coordinate). We assume that $\vec{p}' = (0, p'_y)^\top \in \mathbb{Z}^2$ is the center of the pixel. Then the four points of intersection $\vec{p}_1, \vec{p}_2, \vec{p}_3, \vec{p}_4$ have the coordinates $\vec{p}_1 = (\delta_1, p'_y + \epsilon_1)^\top, \vec{p}_2 = (\delta_2, p'_y + \frac{1}{2})^\top, \vec{p}_3 = (-\delta_2, p'_y + \frac{1}{2})^\top, \vec{p}_1 = (-\delta_1, p'_y + \epsilon_1)^\top$ where $\delta_1, \delta_2, \epsilon_1 \in [0, \frac{1}{2}]$. Because these four points belong to $\mathcal{C}$, we have

$$p_x^2 + p_y^2 = \delta_2^2 + (p'_y + \frac{1}{2})^2. \tag{3.13}$$

By definition, $p_x^2, p_y^2, p_y'^2$ are integers, thus $\delta^2 + \frac{1}{4}$ must be an integer that is impossible since $\delta_2 \in [0, \frac{1}{2}]$. We can deduce that $\vec{p}_2$ and $\vec{p}_3$ cannot exist.

Therefore we have only 2 intersections.






(a) $\Sigma\mathscr{F}(C_i(\vec{q}_2)) = 0$ (b) $\Sigma\mathscr{F}(C_i(\vec{q}_2)) = 1$

(c) $\Sigma\mathscr{F}(C_i(\vec{q}_2)) = 2$ (d) $\Sigma\mathscr{F}(C_i(\vec{q}_2)) = 2$

FIGURE 3.3: Illustration of cases $\Sigma\mathscr{F}(C_i(\vec{q}_2)) = 0,1,2$ or 3.

From Theorem 3.12, we always have two distinct intersections between $\mathscr{C}(\vec{p}_2)$ and $\mathscr{H}(\vec{q}_2)$, and we see that there are four cases corresponding to different possibilities to have 0,1,2 or 3 corners inside of $\mathscr{C}(\vec{p}_2)$, as illustrated in Figure 3.3.

*Remark* 3.13. The case represented in (a) in Figure 3.3 never happen unless two conditions are satisfied.

1. $\vec{q}_2$ is on the $x$-axis or on the $y$-axis.
2. The radius $r$ of $\mathscr{C}(\vec{p}_2)$ is sufficiently close to a half-integer.

Supposing that $\vec{q}_2$ is neither on the $x$-axis nor the $y$-axis, we see that neither $R_2$ nor $S_2$ is zero. We assume that they are positive. In the first quadrant, the $y$-coordinate (respectively the $x$-coordinate) of points in $\mathscr{C}(\vec{p}_2)$ is strictly decreasing with respect to $x$ (respectively $y$). Thus it cannot intersect twice with a line parallel to the $x$-axis (respectively the $y$-axis).

Supposing that $\vec{q}_2$ is on the $y$-axis, we see that the distance between the origin and $C_1(\vec{q}_2)$ (respectively $C_3(\vec{q}_2)$) is greater (respectively smaller) than $r$. Thus we have $r < \sqrt{(R_2 - \frac{1}{2})^2 + (\frac{1}{2})^2}$ (respectively $r > \sqrt{(R_2 + \frac{1}{2})^2 + (\frac{1}{2})^2}$ ). Letting $\epsilon = r - (R_2 - \frac{1}{2})$ (respectively $\epsilon = r - (R_2 + \frac{1}{2})$) where $\epsilon < \frac{1}{2}$, we obtain $\epsilon(8r - 4\epsilon) < 1$ (in both cases). We experimentally observe that as far as $r \leq 10^5$, $\epsilon(8r - 4\epsilon) < 1$ never holds. On the other hand, if $r$ is sufficiently large enough to have $8r - 4\epsilon \approx 8r$ then we have $\epsilon < \frac{1}{8r}$, which means that $r$ is sufficiently close to a half-integer. We can thus conclude that (2) in Remark 3.13 is satisfied only if $r$ is sufficiently large. Note that a similar discussion can be done when supposing that $\vec{q}_2$ is on the $x$-axis.

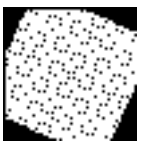


TABLE 3.1: Corners of $\mathscr{H}(\vec{q}_2)$ inside of $\mathscr{C}(\vec{p}_2)$ and ARA.

| Value of $I_{\vec{q}_2}$ | $\alpha_{\inf}$ | $\alpha_{\sup}$ |
|---|---|---|
| $I_{\vec{q}_2} = 1$ or $I_{\vec{q}_2} = 14$ | $\alpha(P_1, Q_1, R_2 - 1)$ | $\alpha(P_1, Q_1, 1 - S_2)$ |
| $I_{\vec{q}_2} = 2$ or $I_{\vec{q}_2} = 13$ | $\alpha(P_1, Q_1, R_2 - 1)$ | $\alpha(P_1, Q_1, -S_2)$ |
| $I_{\vec{q}_2} = 3$ or $I_{\vec{q}_2} = 12$ | $\alpha(P_1, Q_1, 1 - S_2)$ | $\alpha(P_1, Q_1, -S_2)$ |
| $I_{\vec{q}_2} = 4$ or $I_{\vec{q}_2} = 11$ | $\alpha(P_1, Q_1, R_2)$ | $\alpha(P_1, Q_1, -S_2)$ |
| $I_{\vec{q}_2} = 6$ or $I_{\vec{q}_2} = 9$ | $\alpha(P_1, Q_1, R_2 - 1)$ | $\alpha(P_1, Q_1, R_2)$ |
| $I_{\vec{q}_2} = 7$ or $I_{\vec{q}_2} = 8$ | $\alpha(P_1, Q_1, R_2)$ | $\alpha(P_1, Q_1, 1 - S_2)$ |
| $I_{\vec{q}_2} = 0$ and $R_2 = 0$ | $\alpha(P_1, Q_1, 1 - S_2)$ | $\alpha(P_1, Q_1, 1 - S_2)$ |
| $I_{\vec{q}_2} = 0$ and $S_2 = 0$ | $\alpha(P_1, Q_1, R_2 - 1)$ | $\alpha(P_1, Q_1, R_2 - 1)$ |

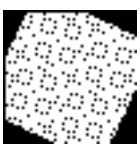
The main function of our algorithm for finding the lower and the upper bounds of admissible rotation angles, consists of three steps. The first step sets the rotation center at $\vec{p}_1$ and $\vec{q}_1$, as described in Section 3.4.1. The second step computes the corners that are in $\mathscr{C}(\vec{q}_2)$ and then compute the index $I_{\vec{q}_2} = \sum_i 2^i \times \mathscr{F}(C_i(\vec{q}_2))$. Therefore we can easily identify which corners are in $\mathscr{C}(\vec{p}_2)$ from $I_{\vec{q}_2}$. The third step calls a function that returns hinge angles corresponding to $I_{\vec{q}_2}$. There are fourteen possible values for $I_{\vec{q}_2}$ from 0 till 15 except for 5 and 10. Note that geometrically $I_{\vec{q}_2}$ can be neither 5 nor 10. The value 15 of $I_{\vec{q}_2}$ implies an error such that all corners are inside of $\mathscr{C}(\vec{q}_2)$. Since $I_{\vec{q}_2}$ whose value is 0 corresponds to the case $\Sigma\mathscr{F}(C_i(\vec{q}_2)) = 0$, we should verify whether $\mathscr{H}(\vec{q}_2)$ really intersects with $\mathscr{C}(\vec{p}_2)$. Note that for the other values for $I_{\vec{q}_2}$, we can make a pair $(d, e)$ such that $d + e = 15$. The two indices of each pair design the same pair of lower and upper bounds of ARA. Table 1 gives the corresponding lower and upper bound rotation angles for each value of $I_{\vec{q}_2}$. Each step of this algorithm has the constant time complexity. Thus the global complexity of this algorithm is also O(1).

### 3.4.3 Incremental computing lower and upper bounds of rotation angles

In general, the corresponding point sets contain more than two points. Therefore, in this section, we extend our algorithm in Section 3.4.2 to two sets of corresponding point pairs, $A$ and $B$, each of them having $l$ points where $l > 2$.

To simplify the notation, we denote by $ARA(\vec{p}_i, \vec{q}_i) = (\alpha_{i\,\inf}, \alpha_{i\,\sup})$ the pair of angles that gives the lower and the upper bounds of admissible rotation angles for the pair of points $(\vec{p}_i, \vec{q}_i)$. Note that $\alpha_{i\,\inf}, \alpha_{i\,\sup}$ are hinge angles. $ARA(A_n, B_n)$ denotes the two most restrictive angles for all points $i$ such as $i \leq n$. We recursively define it by

$$ARA(A_n, B_n) = ARA(A_{n-1}, B_{n-1}) \bigcap ARA(\vec{p}_n, \vec{q}_n).$$



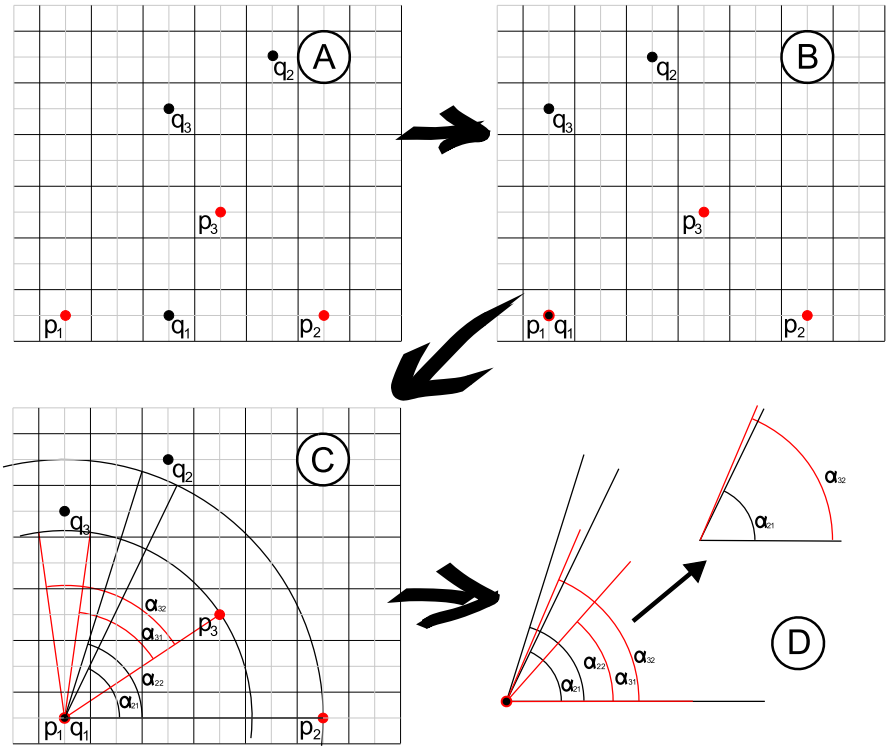
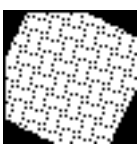



FIGURE 3.4: Running of the incremental algorithm.

A new algorithm handles all points incrementally. This algorithm is divided into two parts. The first part is to initialize the algorithm by computing $ARA(\vec{p}_2, \vec{q}_2)$. The second part computes $ARA(A_i, B_i)$ for $i = l$. Note that $ARA(\vec{p}_1, \vec{q}_1)$ cannot be computed because $\vec{p}_1$ and $\vec{q}_1$ are the centers of the rotation.

The time complexity of this algorithm is $O(l)$. As explained in Section 3.4.2, the function giving a pair of lower and upper bound rotation angles from a pair of points is realized in a constant time $O(1)$. Moreover, as explained in Section 3, we can compare two hinge angles in a constant time $O(1)$. Therefore, the computation of this algorithm for $l$ points takes the time complexity of $l \times (O(1) + O(1)) = O(l)$ as a whole.

Figure 3.4 gives an example of the incremental algorithm for two sets of three points. Given input data of the algorithm as shown in Figure 3.4 (A), we first obtain the result of the translation described in Section 3.4.1 as illustrated in (B). We then compare, for each pair of points $(\vec{p}_i, \vec{q}_i)$ with $i \geq 2$, the distance of $\vec{p}_i$ from the origin with that of each corner from $\mathscr{H}(\vec{q}_i)$ to deduce the corresponding hinge angle as explained in Section 3.4.2. Finally, we obtain (D) that shows the intersection of all $ARA(\vec{p}_i, \vec{q}_i)$ obtained in (C).



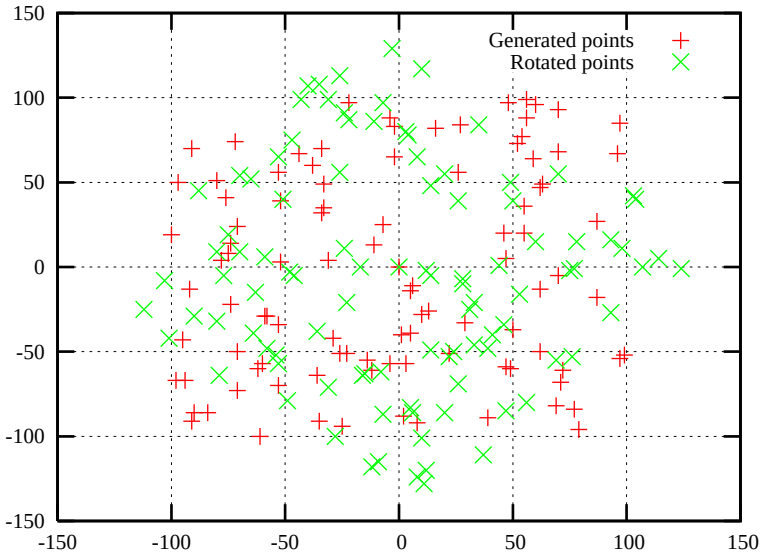



FIGURE 3.5: Randomly generated points and their corresponding rotated points.

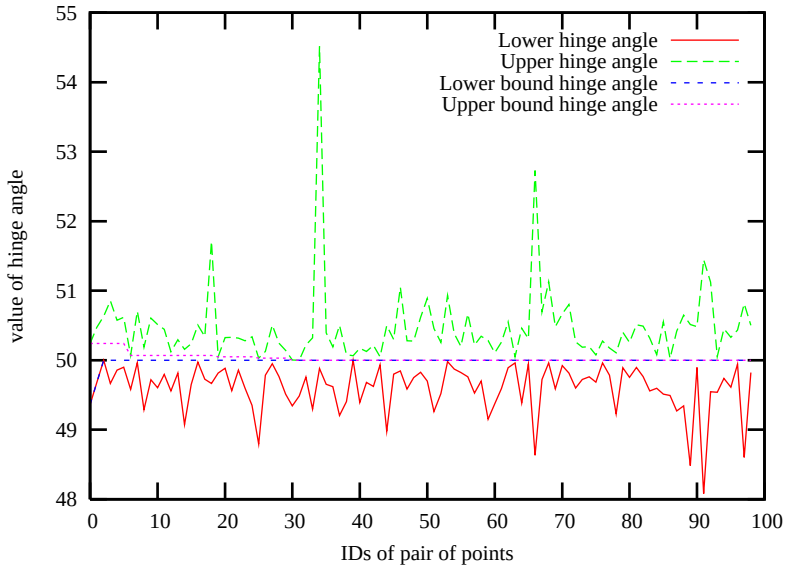
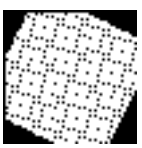



FIGURE 3.6: Result of our algorithm applied on the sets of points in Figure 3.5 .

## 3.5 Experimental Evaluation using Synthetic Data

We evaluated our algorithm using synthetic data and verified the quality of obtained admissible rotation angles. To test our algorithm, we need two sets of points. We randomly generated the set $F$ of 100 floating points in a $200 \times 200$ square. The first set $I$ is obtained by discretizing $F$. The second set $I'$ is obtained by applying the DER with angle $\theta = 50°$ to the first set $I$ (Figure 3.5). We assumed that the point correspondences across the two sets $I$ and $I'$ are known.

Figure 3.6 shows the hinge angles obtained by our algorithm. The green and the red



curves give the lower and the upper bound rotation angles for each pair of point in correspondence. We can see that lower bound rotation angles are always lower than $\theta$ and that upper bound rotation angles are greater than $\theta$. The blue and purple lines give the lower and the upper bound rotation angles for points with the lower IDs than the point of interest. As we can see, after only ten pairs of points, the range of admissible rotation angles becomes less than $0.1°$ while the range is reduced to $0.02°$ after twenty pairs of points. Then the precision increases slowly. This shows that the precisions of admissible rotation angles acquired after twenty pairs of points are not significant in this particular case.

Figure 3.6 also indicates that most pairs of points give the admissible rotation angle range smaller than $1°$ (for example, pair #39). Pair #34 and pair #66, however, give a range greater than $4°$. In fact, the maximum range for a pair of points is directly related to the distance between the center of rotation and the pair of points. We denote by $d$ the distance between the rotation center and the pair of points. Then, the maximum difference between the lower and the upper bound rotation angles is $\sin^{-1}\frac{\sqrt{2}}{d}$. For the pair #34, $d = \sqrt{173}$; thus the maximum range is approximately $6.1°$. For the pair #39, $d = \sqrt{16505}$ and thus the maximum range is approximately $0.63°$. This is consistent with our experimental result.

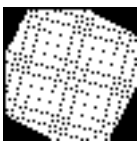
## 3.6 Discussion on practical application to digital images

In the previous section, we start with grid points in the discrete plane. In other words, all coordinates of the points are integers. But in reality, points in the Euclidean space are not represented by integers but real numbers. Thus we start here with the set of floating points to see how our algorithm works. Then we test our algorithm with real data acquired by a digital camera.

### 3.6.1 Synthetic data

To test our algorithm using synthetic data more similar to real data, we applied the Euclidean rotation with angle $\theta = 50°$ directly to the set $F$ that is used in Section 3.5. Then we discretized the rotated image to obtain the second set $I''$. To the two sets $I$ and $I''$, we applied our algorithm. We note that the set $I$ comes from Section 3.5.

Figure 3.7 shows the hinge angles obtained. Because we started with the set of floating points, we observe some errors in bounding admissible rotation angles. In fact, we see that some pairs of points, the pairs #64 and #73 for instance, do not contain $\theta$.



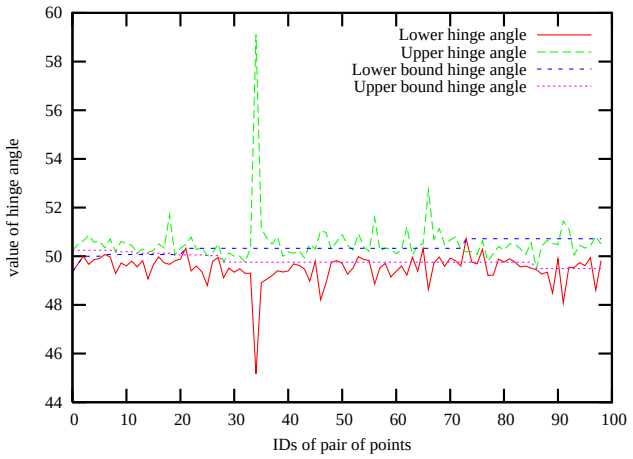


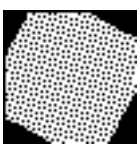

FIGURE 3.7: Result of our algorithm applied on a set of floating points.

Figure 3.8 (left) explains how errors arise in rotating floating points. Let assume that the point $\vec{q}_1$ is obtained after the Euclidean rotation of the floating point $\vec{p}_1$ with angle $\gamma$. Let $\vec{p}_2$ and $\vec{q}_2$ denote the discretization of $\vec{p}_1$ and $\vec{q}_1$. If we apply our algorithm to the pair of points $\{\vec{p}_2, \vec{q}_2\}$, we obtain $ARA(\vec{p}_2, \vec{q}_2) = (\alpha_{\text{inf}}, \alpha_{\text{sup}})$. In this case, $\gamma$ does not belong to the interval $[\alpha_{\text{inf}}, \alpha_{\text{sup}}]$ (Figure 3.8 (right)). We see that this is caused by discretization of the floating points. We can give a bound for this error. In fact, this bound directly depends on the distance between the rotation center and the pair of points. If we denote by $d$ this distance, the maximum error is equal to $\sin^{-1} \frac{\sqrt{2}}{2d}$. Note that the pair of points $\{\vec{p}_3, \vec{q}_3\}$ and the angle $\rho$ bring the same problem.

To avoid this problem, we take the example of modifying our algorithm as follows. We keep all the pairs of bound rotation angles determined by all pairs of points. After sorting all lower and upper bound rotation angles into two lists $\mathscr{L}_{\text{inf}}, \mathscr{L}_{\text{sup}}$, we remove all hinge angles $\alpha_{\text{inf}}$ from $\mathscr{L}_{\text{inf}}$ such that for $\exists \alpha_{\text{sup}} \in \mathscr{L}_{\text{sup}}$, $\alpha_{\text{inf}} > \alpha_{\text{sup}}$. We also remove all hinge angles $\alpha_{\text{sup}}$ from $\mathscr{L}_{\text{sup}}$ such that for $\exists \alpha_{\text{inf}} \in \mathscr{L}_{\text{inf}}$, $\alpha_{\text{sup}} < \alpha_{\text{inf}}$.

With this procedure, we can guarantee that the lower bound hinge angle is smaller than the rotation angle applied to the points and that the upper bound hinge angle is larger than the rotation angle. The complexity $\mathrm{O}(n \log n)$ is required in sorting the remaining hinge angle pairs, which does not increase the computational cost of the algorithm as a whole.

Another way to avoid the problem is to compute the smallest distance $d$ of all the points from the rotation center and then compute a Pythagorean angle $\theta$ satisfying $\theta > \sin^{-1} \frac{\sqrt{2}}{2d}$. We then add (subtract) $\theta$ to the upper (lower) bound rotation angle obtained by our algorithm in Section 3.4. As a result we obtain the upper and the lower bound rotation angles that define the interval accurately including the true rotation angle. Thanks to



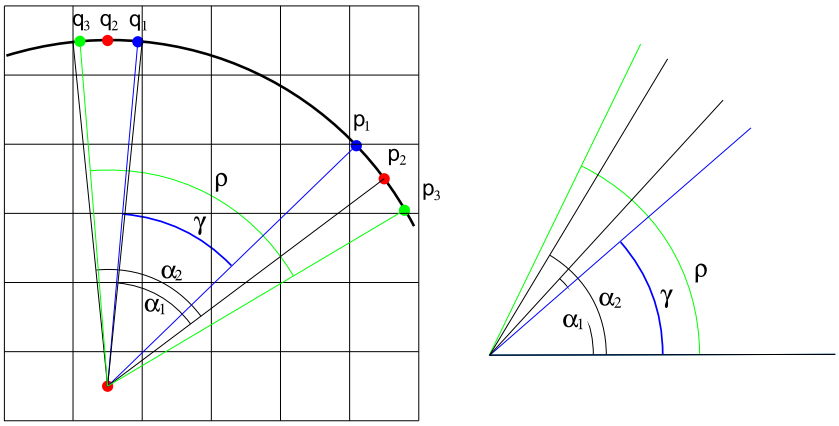



FIGURE 3.8: Examples of errors for computing lower and upper bounds of rotation angles introduced by rotation of floating points.

Theorem 3.7, the addition of a Pythagorean angle does not change the nature of the upper (lower) bound rotation angle.

Both methods return a valid ARA. It is preferable to use the second method when there are a few pairs of points because it does not remove any hinge angles. The first method is preferable when data contains many pairs of points in correspondence because the obtained ARA is more restrictive than the ARA obtained by the second method.

### 3.6.2 Real data

We applied our algorithm to see its practical usefulness. We used a turntable that is rotated with respect to the vertical axis with respect to a digital image plane. The precision of rotations in control is $10^{-3}$ degrees.

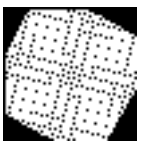
We put a toy block on the turntable and then took its image using a standard digital camera where the camera was fixed so that its optical axis is parallel with the rotation axis (Figure 3.9(a)). Next we rotated the turntable with the angle of 44.99° and then took another image of the toy block by the fixed camera (Figure 3.9(b)).

We manually selected five points in the first image and their corresponding points in the second image. Then we applied our algorithm to the five pairs of corresponding points.

Our algorithm might return the empty result (Figure 3.9(c)). This is because no intersection between $\mathscr{C}(\vec{p}_i)$ and $\mathscr{H}(\vec{q}_i)$ was found for all $i = 1, 2, \ldots, 5$.

In the case of real data, we cannot always guarantee to detect correct corresponding pairs of points even manually. This indicates that a problem different from the discretization problem (see Section 3.6.1) arises[4]. Namely the maximum difference of distances from the

[4]We assume here that we are to find ARA from given correspondences.



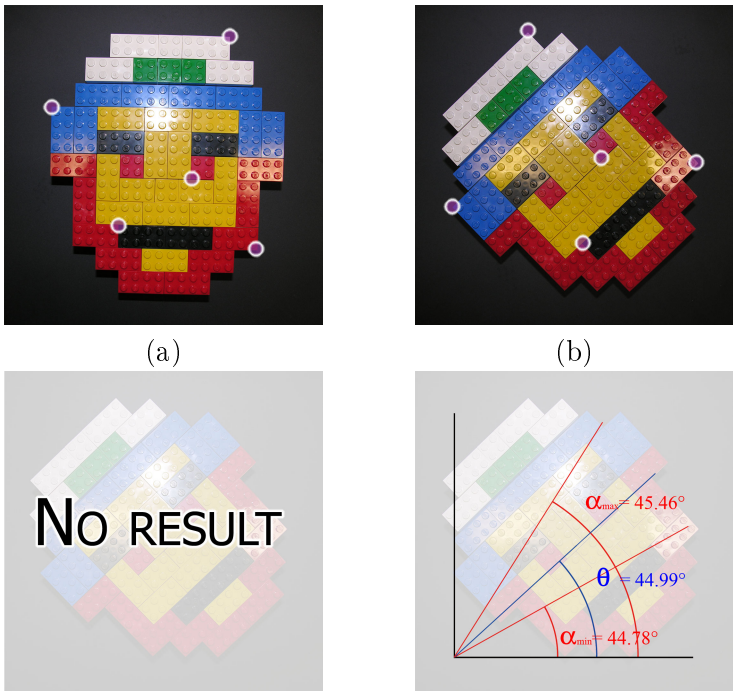


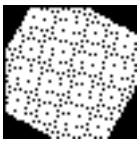

FIGURE 3.9: Result of our algorithm applied on real data.

rotation center between two points in correspondence can become greater than $\sqrt{2}$. This is illustrated in Figure 3.10 where points $\vec{p}$ and $\vec{q}$ are in correspondence and $\epsilon_d = |d_{\vec{p}} - d_{\vec{q}}|$ is greater than $\sqrt{2}$ where $d_{\vec{p}}, d_{\vec{q}}$ are respectively the distances from $\vec{p}, \vec{q}$ to the origin. We see that no intersection exists between the circle $\mathscr{C}(\vec{p})$ going through $\vec{p}$ and $\mathscr{H}(\vec{q})$.

To avoid this problem on corresponding points, we can change the resolution of the images. Namely, we degrade the image resolution until we find an intersection between $\mathscr{C}(\vec{p})$ and $\mathscr{H}(\vec{q})$[5]. Because of the change in image resolution, we cannot accept the obtained hinge angles as they are. Instead, we add (subtract) a Pythagorean angle $\theta$ to (from) the obtained upper (lower) hinge angle. Let assume that $d_{\vec{p}} < d_{\vec{q}}$ and that $\mathscr{H}(\vec{q})$ is defined for the image whose resolution is degraded with $2^{-n}$ from the original one. Then we have to choose $\theta$ satisfying $\theta > \frac{n\sqrt{2}}{2d_p}$. As mentioned in Section 3.6.1, thanks to Theorem 3.7, the addition of a Pythagorean angle does not change the nature of the upper (lower) bound rotation angle.

Figure 3.9 (d) shows the results obtained by the modification above to the same data. We can see that the real rotation angle (44.99°) is included in the ARA obtained by our algorithm with this modification. Accordingly, we can conclude that our modification is effective.

[5]There are other criteria for finding a reasonable image resolution. For example, Brimkov presents criteria of faithfully digitization for continuous objects in [22].



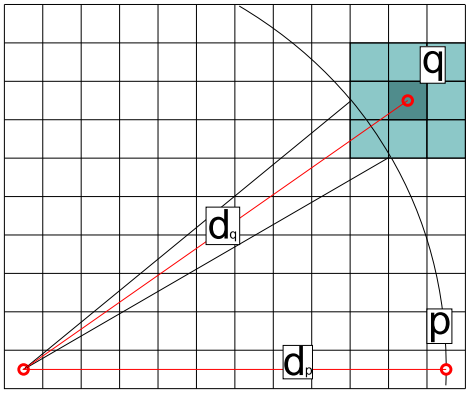



FIGURE 3.10: Problem of corresponding points with real data.

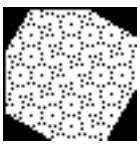
## 3.7 Conclusion

In this chapter, we have presented a new method to use rotation in discrete space. Because rotations are mathematically defined for the continuous space, it is necessary to develop new operational tools for such discrete rotations.

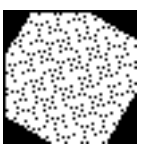
Our problem was from two digital images to find all possible rotation angles to rotate the first image into the second. Namely, to find the admissible rotation angles between two digital images. Since we need to have only exact computation without any approximation, we decide to adopt hinge angles, which allow us rotations using only integer computation. By using their properties, we have proposed a method for incrementally computing the lower/upper bound rotation angles from a Pythagorean angle for a set of discrete points. Based on this method, we have also proposed a linear algorithm for finding from two digital images the lower and the upper bound rotation angles representing the admissible rotation angles between them.

In addition, we have shown with experiments that our algorithm is efficient with synthetic data and gives consistent results. Then we also have done some experiments to real data. Results given by our algorithm were not consistent because of the loss of correct point correspondences in our input. We then explained why these problems appear and gave necessary modifications to obtain consistent results. These modifications did not increase the time complexity while keeping the integer computation.

The range size of ARA can be considered as the unreliability of the point correspondence. In other words, larger the range of ARA between two points, less is the reliability of their point correspondence. Therefore, it may be interesting if we can propose a new method for evaluating "good" point correspondence from a given pair of digital images by using our results.



In this paper we always assume that rotations centers are on integer points. But with real data it is not always the case. One of our future work will be to adapt the presented method to such rotations. In this purpose, studies on digital discs whose centers are not only on integer points [3, 23, 24] would help.



# Chapter 4

# 3D Rotations in discrete space using hinge angles

## 4.1 Introduction

Rotations in the 3D space are required in many applications for computer imagery, such as image processing [10], computer vision [9, 25] and computer graphics [8]. A rotation in the 3D Euclidean space can be in general represented in two different typical ways. One way is to represent a rotation as a combination of three particular rotations around the three axes of the coordinate system [5]. The other way is to represent a rotation by a rotation axis together with an angle around the axis [9, 25]. Even if the representations of a rotation are different, computed rotation results are the same as far as the space is continuous. However, this is not the case for the discrete space. Namely, depending on the rotation representation, the computed rotation result can change in the discrete space [2]. As this is the case of 2D rotations, computing a 3D rotation once in the discrete space brings displacement from in the continuous space; computing 3D rotations many times causes difficulty in analyzing inherited displacements during the computation. Accordingly, representing a 3D rotation by a rotation axis together with an angle around the axis is more preferable in the 3D discrete space. Besides, it is known that such axis-angle representation is useful for 3D rotation estimation from given image sets, which is one of the important problems encountered in computer vision [9, 25].

This chapter presents a study of the rotation in the 3D discrete space. Since we admit only integer computations, we assume that our rotation center is a grid point such as the origin, and that a rotation axis has integer coordinates. In the 2D case, hinge angles are known to be corresponding to the discontinuity caused by discretization of the rotation in the continuous plane as explained in Chapter 3 and in [4, 26]. Intuitively, hinge

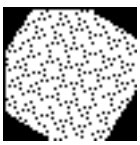
angles determine a transfer of a grid point from a pixel to its adjacent pixel during the rotation. In other words, two rotations with nearby angles transform the same grid point to two adjacent pixels because discrete rotations around a given center are locally continuous with regard to the angle. Hinge angles are their discontinuity angles. Computing hinge angles using integers alone allows us to compute 2D discrete rotations without any approximation errors, which designs the 2D discrete rotation. Extending these to the 3D case, we design a 3D discrete rotation. In the 3D case, however, depending on the rotational axis, we have a variety of transitions of a grid point across voxels. How to capture these transitions systematically is a big issue.

In this chapter, we first define hinge angles for 3D rotations so that they determine a transit of a grid point from a voxel to its adjacent voxel. To compute the hinge angles for 3D rotations, we introduce a notion, called "multi-grids", that is given by the intersection between a plane normal to the rotation axis and the half-grids that mark the boundary between adjacent voxels. The rational multi-grids, which are a subset of multi-grids, allow computations of hinge angles using only rational numbers. Using rational multi-grids, we show that, as in 2D, it is possible to only use integer during computation and to have an integer representation of hinge angles. Then, we give a method to sort all the possible hinge angles in concern to design a 3D discrete rotation. We also propose a method to obtain from a pair of 3D digital images in correspondence a set of 3D rotations, each of them transforming the first digital image into the second one. Note that in a discrete space, there are sets of rotations that give the same rotated image of a given digital image. In this paper, we fix a rotation axis and look for all the possible rotation angles from a given pair of 3D digital images. The set of all possible rotation angles is called admissible rotation angles and its upper and lower bounds are represented by hinge angles. This method is the extension of the method proposed in Chapter 3 for the 2D cases into 3D cases.

Differently from 2D discrete rotations [2, 4, 6], few attempts on 3D discrete rotations have been reported [8, 10]. In particular, to our best knowledge, this is the first work on 3D discrete rotations using integer computations without digitization errors.

## 4.2 Hinge angles

Hinge angles for 2D rotations are defined to represent the discontinuities of rotations in the discrete plane [4]. Hinge angles determine a transit of a grid point from a pixel to its adjacent pixel during the rotation. To characterize those hinge angles, the 2D half-grid plays an important role.



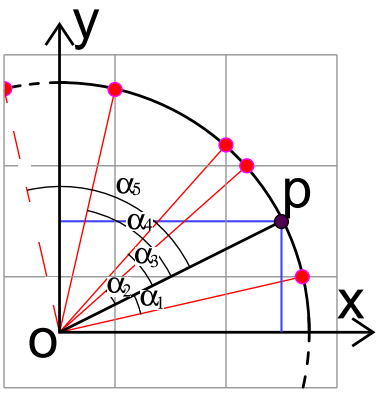



FIGURE 4.1: All hinge angles in the first quadrant for the grid point $\vec{p} = (2,1)^\top$, such that $\alpha_1 \approx -13.68°$, $\alpha_2 \approx 15.59°$, $\alpha_3 \approx 21.31°$, $\alpha_4 \approx 50.52°$. The hinge angle $\alpha_5$ is obtained by symmetry such that $\alpha_5 = \frac{\pi}{2} - \alpha_1 \approx 76.32°$

**Definition 4.1.** The 2D half-grid is the set of lines in the plane, each of which is represented by one of $x = i + \frac{1}{2}$ and $y = i + \frac{1}{2}$ where $i \in \mathbb{Z}$.

In other words, the 2D half-grid represents the border between two adjacent pixels. From the definition of the 2D half-grid, we define the hinge angles in the plane.

**Definition 4.2.** An angle $\alpha$ is called a *hinge angle* if at least one point in $\mathbb{Z}^2$ exists such that its image by the Euclidean rotation with $\alpha$ around the origin is on the 2D half-grid.

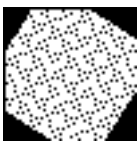
Figure 4.1 illustrates all the hinge angles in the first quadrant for the grid point $(2,1)^\top$. Note that hinge angles in other quadrants are obtained by symmetry with respect to the $x$-axis and/or $y$-axis from those in Figure 4.1.

To extend the definition of hinge angles into the 3D case, we first define the half-grid in the 3D space. Similarly to the 2D half-grid, the 3D half-grid defines the limit between two adjacent voxels in the 3D discrete space.

**Definition 4.3.** The 3D half-grid is the set of planes in the 3D space, each of which is represented by one of $x = i + \frac{1}{2}, y = i + \frac{1}{2}$ and $z = i + \frac{1}{2}$ where $i \in \mathbb{Z}$.

Introducing the definition of the 3D half-grid allows the definition of hinge angles in 3D as a natural extension of hinge angles in 2D. As mentioned in the introduction, we only consider here 3D rotations whose rotation axes have directional vectors with integer coordinates and go through the origin. Hereafter, we call such an axis an integer-axis.

**Definition 4.4.** An angle $\alpha$ is called a *hinge angle* if at least one point in $\mathbb{Z}^3$ exists such that its image by the Euclidean rotation with $\alpha$ around an integer-axis is on the half-grid.



Similarly to the 2D case, for a grid point $\vec{p}$ in 3D, an angle $\alpha$ is a hinge angle if and only if the discretized point of the rotation result of $\vec{p}$ with angle $\alpha+\epsilon$ becomes different from that with angle $\alpha-\epsilon$ for any $\epsilon > 0$.

Differently from the case of 2D rotations, we need not only a rotation angle but also a rotation axis in order to specify a 3D rotation. This requires investigation of the intersection between voxels and a plane determined by a given rotation axis because of a variety of transitions of a grid point across voxels existing on the plane. To capture this variety, we introduce the multi-grid in the next section, which allows us to study hinge angles in the 3D rotation plane.

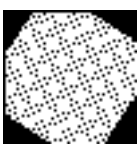
## 4.3 Multi-grids

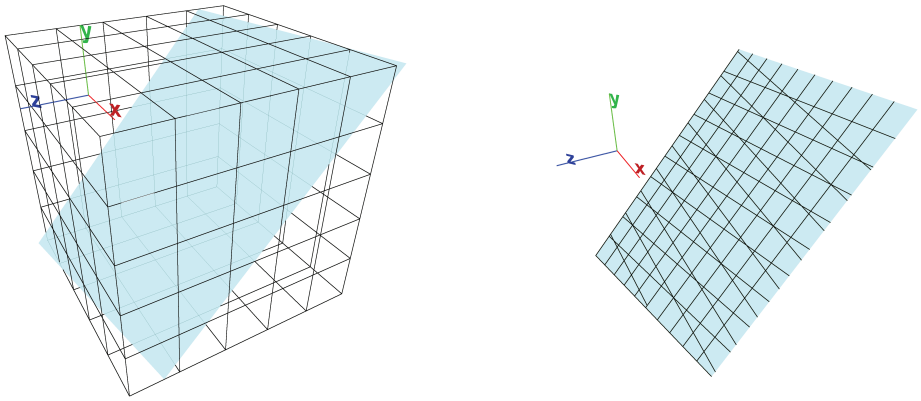
In this section, we introduce a notion that is required to extend hinge angles from 2D to 3D and thus required to perform discrete rotations in the 3D discrete space. As explained in Section 4.2, hinge angles are strongly related to the half-grid both in 2D and 3D. However, in the 3D cases, rotations of a point are always in its rotation plane normal to the rotation axis and goes through the point. In fact, the 3D half-grid is not well adapted to describe rotations for a grid point. We thus consider the intersection between a rotation plane and the 3D half-grid, which is a planar grid consisting of three sets of parallel lines, called a multi-grid instead of the half-grid.

In this section, we give a formal definition of multi-grids. Then we show how to obtain the line equations of a multi-grid from a grid point and a rotation axis. Then, we restrict multi-grids to the rational multi-grids that form a set of multi-grids useful to perform discrete rotations in the 3D discrete space. Finally we present some useful properties of rational multi-grids.

### 4.3.1 Definition of multi-grids

When a rotation plane in 3D is given, the intersection between the plane and the half-grid in the 3D space is obtained as illustrated in Figure 4.2(a). Figure 4.2(b) shows that the intersection consists of three different sets of parallel lines, except for cases where the normal of the rotation plane is parallel to one of the axes, defining the coordinate system of the 3D space. As such exceptional cases provide only two different sets of parallel lines, which are identical with those of the 2D half-grid, we do not take into account those cases here. In other words, in such cases, 3D discrete rotations become identical with 2D discrete rotations. We call the three different sets of parallel lines a multi-grid,






(a) (b)

FIGURE 4.2: The 3D half-grid cut by a plane (a), and its multi-grid (b).

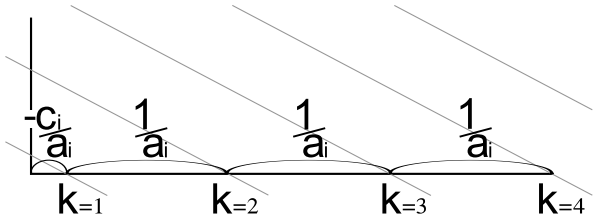



FIGURE 4.3: Parallel lines of a set $\mathcal{L}_i^{\mathcal{A},\vec{p}}$ and geometric interpretation of their parameters

which is used for characterizing hinge angles for 3D discrete rotations instead of the 2D half-grid for 2D discrete rotations.

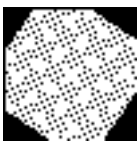
In a multi-grid, the interval between parallel lines having the same directional vector is regular. Normalizing the interval allows us to represent each set of parallel lines having the same normal vector $(a_i, b_i)^\top$ as

$$\mathcal{L}_i = \{(x,y)^\top \in \mathbb{R}^2 | a_i x + b_i y + c_i + k = 0, a_i^2 + b_i^2 \neq 0, k \in \mathbb{Z}, a_i, b_i, c_i \in \mathbb{R}\} \tag{4.1}$$

where $i = 1, 2, 3$. The integer parameter $k$ denotes the index number of each parallel line. Figure 4.3 gives a geometrical explanation of the parameters of $\mathcal{L}_i$. For example, if a point $(x,y)^\top$ is on one of the parallel lines of $\mathcal{L}_i$, $(x - \frac{k}{a_i}, y)^\top$ is on the $k$-th next line, providing that $a_i \neq 0$. Now we can give a formal definition of a multi-grid.

**Definition 4.5.** Let $\mathcal{L}_i$ for $i = 1, 2, 3$ be each set of parallel lines induced from a given rotation plane and the 3D half-grid. Then the multi-grid $\mathcal{M}$ is the union of $\mathcal{L}_i$: $\mathcal{M} = \cup_{i=1}^3 \mathcal{L}_i$.

Hereafter, we denote by $\mathcal{L}_i^{\mathcal{A},\vec{p}}$ the set of parallel lines defined by a rotation plane with a normal vector $\mathcal{A} = (a_x, a_y, a_z)^\top$ going through point $\vec{p} = (p_x, p_y, p_z)^\top$. By using the same idea, we denote by $\mathcal{M}^{\mathcal{A},\vec{p}}$ the multi-grid defined as the union of $\mathcal{M}^{\mathcal{A},\vec{p}} = \cup_{i=1}^3 \mathcal{L}_i^{\mathcal{A},\vec{p}}$



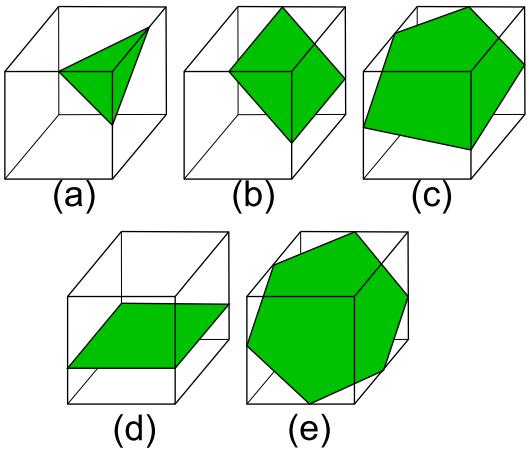



FIGURE 4.4: The five different shapes of convexels, which are constructed as the intersections between a rotation plane and voxels [1].

The multi-grid in the rotation plane forms various closed convex polygons surrounded by lines, that we call convexels. Depending on the rotation plane, we have a variety of shapes of convexels in this paper. The convexel in a multi-grid is the counterpart of the squared pixel defined in the 2D half-grid. The shapes of convexels are investigated in [1] under the context of the intersection of a voxel and a plane, and the number of vertices of a convexel can be 3, 4, 5 or 6 as illustrated in Figure 4.4.

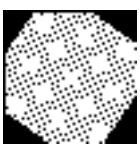
*Remark* 4.6. On any $\mathcal{M}^{\mathcal{A},\vec{p}}$, the convexel containing $\vec{p}$ always has a symmetric shape, and $\vec{p}$ is at the center of the convexel. Moreover, the convexel that contains $\vec{p}$ is necessarily of the type (b),(d) or (e) according to Figure 4.4.

We remark that when the normal of the rotation plane is parallel with one of the axes defining the coordinate system of the 3D space, the notion of convexels coincides with the notion of pixels.

### 4.3.2 Multi-grid line equations

To simplify the derivation of line equations for each $\mathcal{L}_i^{\mathcal{A},\vec{p}}$, we introduce a new base $B_i$ where lines in $\mathcal{L}_i^{\mathcal{A},\vec{p}}$ are parallel with the $x$-axis. In the following, we derive the equations for $\mathcal{L}_1^{\mathcal{A},\vec{p}}$. Note that the same discussion can be applied to $\mathcal{L}_2^{\mathcal{A},\vec{p}}$ and $\mathcal{L}_3^{\mathcal{A},\vec{p}}$.

Let $\mathcal{A} = (a_x, a_y, a_z)^\top$ and $\vec{p} = (p_x, p_y, p_z)^\top$ in the standard orthonormal base $B$ and the plane $\mathcal{P}$ with normal vector $\mathcal{A}$ that goes through $\vec{p}$. Assuming lines in $\mathcal{L}_1^{\mathcal{A},\vec{p}}$ come from the intersection between $\mathcal{P}$ and the planes of the 3D half-grid that are parallel to $yz$-plane, denoted by $x = k$ where $k \in \mathbb{Z}$, we obtain the directional vector $\vec{v}_1$ of lines in $\mathcal{L}_1^{\mathcal{A},\vec{p}}$ as $\vec{v}_1 = \mathcal{A} \wedge \vec{e}_1$ where $\vec{e}_1 = (1,0,0)^\top$. We set $\vec{v}_2 = \frac{\vec{v}_1 \wedge \mathcal{A}}{\|\mathcal{A}\|}$, which is orthogonal to $\vec{v}_1$. Note that both $\vec{v}_1$ and $\vec{v}_2$ are orthogonal with respect to $\mathcal{A}$.



We introduce a new base $B_1$ in such a way that $\vec{v}_1$ and $\vec{v}_2$ respectively become $\vec{u}_1 = (1,0)^\top$ and $\vec{u}_2 = (0,1)^\top$ in $\mathcal{P}$. The transformation from $B$ to $B_1$ is realized by

$$P_{BB_1} = \begin{pmatrix} 0 & \frac{a_z}{a_y^2+a_z^2} & \frac{-a_y}{a_y^2+a_z^2} \\ -\psi & \frac{\psi a_x a_y}{a_y^2+a_z^2} & \frac{\psi a_x a_z}{a_y^2+a_z^2} \end{pmatrix}, \tag{4.2}$$

where $\psi = \frac{1}{\sqrt{a_x^2+a_y^2+a_z^2}}$. We remark that $\mathcal{A}$ is transformed to $(0,0)^\top$ by $P_{BB_1}$; the rotation center in $\mathcal{P}$ thus becomes the origin of $B_1$.

We remark that if $\mathcal{A}$ is collinear with one of the axes defining the coordinate system of the 3D space, $P_{BB_i}$ degenerates: the rank of $M_{BBi}$ becomes 1. In such cases, 3D rotations become identical with 2D rotations. However, in Section 4.3.1, we explained that we do not consider the cases where the vector $\mathcal{A}$ is collinear with an axis defining the coordinate system in concern, so that lines in $\mathcal{L}_i^{\mathcal{A},\vec{p}}$ are not orthogonal to those in $\mathcal{L}_j^{\mathcal{A},\vec{p}}$ where $i \neq j$. Thus, we do not take these particular cases in consideration.

Applying $P_{BB_1}$ to the plane $x = k$ for $k \in \mathbb{Z}$ induces a line in $\mathcal{M}_1^{\mathcal{A},\vec{p}}$ whose equation is

$$\psi(a_y^2 + a_z^2)y + k - \mathcal{A}.\vec{p}\psi^2 a_x = 0. \tag{4.3}$$

Changing the roles between $\mathcal{L}_1^{\mathcal{A},\vec{p}}$ and $\mathcal{L}_2^{\mathcal{A},\vec{p}}$ (resp, $\mathcal{L}_3^{\mathcal{A},\vec{p}}$.) and between $\vec{e}_1$ and $\vec{e}_2 = (0,1,0)^\top$ (resp, $\vec{e}_3 = (0,0,1)^\top$.), we obtain the transformation matrices $P_{BB_i}$ for $i = 2,3$ such that

$$P_{BB_2} = \begin{pmatrix} \frac{-a_z}{a_y^2+a_z^2} & 0 & \frac{a_x}{a_y^2+a_z^2} \\ \frac{\psi a_x a_y}{a_y^2+a_z^2} & -\psi & \frac{\psi a_y a_z}{a_y^2+a_z^2} \end{pmatrix}, \tag{4.4}$$

$$P_{BB_3} = \begin{pmatrix} \frac{a_y}{a_y^2+a_z^2} & \frac{-a_x}{a_y^2+a_z^2} & 0 \\ \frac{\psi a_x a_z}{a_y^2+a_z^2} & \frac{\psi a_y a_z}{a_y^2+a_z^2} & -\psi \end{pmatrix}, \tag{4.5}$$

and the line equations for $\mathcal{L}_2^{\mathcal{A},\vec{p}}$ and $\mathcal{L}_3^{\mathcal{A},\vec{p}}$ such that:

$$\psi(a_x^2 + a_z^2)y + k - \mathcal{A}.\vec{p}\psi^2 a_y = 0, \tag{4.6}$$

$$\psi(a_x^2 + a_y^2)y + k - \mathcal{A}.\vec{p}\psi^2 a_z = 0, \tag{4.7}$$

where $k \in \mathbb{Z}$.

We note that (4.3), (4.6) and (4.7) correspond to the equation of (4.1) for $i = 1,2,3$ respectively. All $a_i$ of (4.1) for $i = 1,2,3$ are null since each $B_i$ is set such that the lines of $\mathcal{L}_i^{\mathcal{A},\vec{p}}$ are parallel to the $x$-axis. We also remark that every $b_i$ of (4.1) depends only on $\mathcal{A}$, but not on $\vec{p}$. Indeed, $c_i$ is the only parameter depending on $\mathcal{A}$ and $\vec{p}$. This implies that all the multi-grids generated from the same normal vector $\mathcal{A}$ with different point

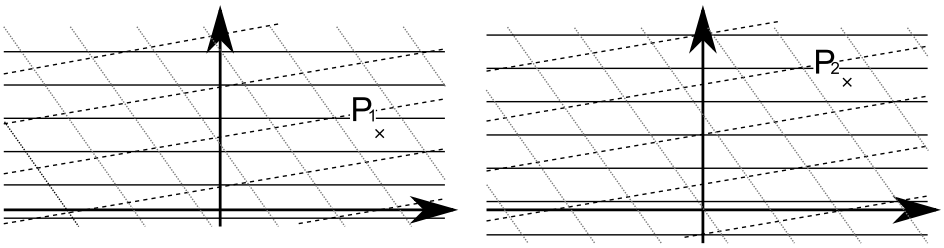



FIGURE 4.5: Two multi-grids generated by a rotation axis but with two different points $\vec{p}_1, \vec{p}_2$ one of which is obtained by a translation of $\pm(\vec{p}_1 - \vec{p}_2)$ from the other.

$\vec{p}$ are similar; more precisely, they are obtained simply by translations as illustrated in Figure 4.5.

### 4.3.3 Rational multi-grids

In Section 4.3.2, we obtained (4.3), (4.6) and (4.7) for the lines in the multi-grid $\mathcal{M}^{\mathcal{A},\vec{p}}$. In general, parameters of these equations belong to $\mathbb{R}$. In order to use only integers during computation, we need these parameters to belong to $\mathbb{Q}$.

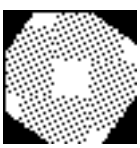
If all elements of $P_{BB_i}$ belong to $\mathbb{Q}$, then all the parameters of (4.3),(4.6) and (4.7) become rational as well. In order to obtain rational parameters in $P_{BB_i}$ we set $\mathcal{A}$ to be a Pythagorean vector; a vector $\vec{v} = (i_1, i_2 \ldots, i_n)^\top, i_1, i_2, \ldots, i_n \in \mathbb{Z}$, is a Pythagorean vector if $\|\vec{v}\| = \lambda$ where $\lambda \in \mathbb{Z}$. Note that a rotation axis whose directional vector is such a Pythagorean vector is called a Pythagorean axis. This assumption ensures that $\psi$ becomes a rational value. We call a multi-grid defined from a Pythagorean axis a rational multi-grid. Note also, that every axis can be approximated by a Pythagorean axis. A study on Pythagorean axis is presented in Chapter 6.

The rational multi-grids presented in this section are a subset of the multi-grid. In addition of the possibility to use only integers during the computations, rational multi-grids also offer some useful properties that are valid only for rational multi-grids.

### 4.3.4 Properties of rational multi-grids

In the case of multi-grids that are not rational, there is an infinity of different convexels. However, for rational multi-grids, the number of convexels is finite and depend only on the coordinates of $\mathcal{A}$ that is the normal vector of rotation plane. Here, we consider any grid point $\vec{p}$ for $\mathcal{M}^{\mathcal{A},\vec{p}}$. A prime Pythagorean axis is a Pythagorean axis $\mathcal{A} = (a_x, a_y, a_z)^\top$ such that $\gcd(a_x, a_y, a_z) = 1$. We define the arithmetical rest of a voxel $\vec{v}$ as follow:

$$\mathcal{R}(\vec{v}) = a_x v_x + a_y v_y + a_z v_z + b. \tag{4.8}$$



**Theorem 4.7.** *Let $\mathcal{M}^{\mathcal{A},\vec{p}}$ be a rational multi-grid associated to the prime Pythagorean vector $\mathcal{A} = (a_x, a_y, a_z)^\top$. The number of different convexels in $\mathcal{M}^{\mathcal{A},\vec{p}}$ for any $\vec{p}$ is either $|a_x| + |a_y| + |a_z|$ or $|a_x| + |a_y| + |a_z| + 1$.*

**Proof.** 7. Let $\mathcal{P}$ be a plane of equation $a_x x + a_y y + a_z z + b = 0$. We know [1] that the voxel around the grid point $\vec{v} = (v_x, v_y, v_z)^\top$ is intersected by $\mathcal{P}$ if and only if:

$$-\frac{|a_x| + |a_y| + |a_z|}{2} \leq \mathcal{R}(\vec{v}) \leq \frac{|a_x| + |a_y| + |a_z|}{2}. \tag{4.9}$$

Since all values in (4.7) are integers , we can deduce that for each $\vec{v}$, $\mathcal{R}(\vec{v})$ is an integer. Thus, if $|a_x| + |a_y| + |a_z|$ is even, we can conclude that there are $|a_x| + |a_y| + |a_z| + 1$ different values for the arithmetical rest $\mathcal{R}(\vec{v})$.

If $|a_x| + |a_y| + |a_z|$ is odd then, (4.9) becomes

$$-\frac{|a_x| + |a_y| + |a_z|}{2} < \mathcal{R}(\vec{v}) < \frac{|a_x| + |a_y| + |a_z|}{2} \tag{4.10}$$

and the number of different values admissible for $\mathcal{R}(\vec{v})$ is $a_x + a_y + a_z$.

Note that, if $\mathcal{A}$ is a Pythagorean vector but not prime, the number of different convexels will be either $\frac{|a_x|+|a_y|+|a_z|}{\gcd(a_x,a_y,a_z)}$ or $\frac{|a_x|+|a_y|+|a_z|}{\gcd(a_x,a_y,a_z)} + 1$.

Such a repetition of convexels in $\mathcal{M}^{\mathcal{A},\vec{p}}$ is shown in the next theorem, which is easily derived from [27] given in the context of discrete planar surfaces.

**Theorem 4.8.** *Let $\mathcal{M}^{\mathcal{A},\vec{p}}$ be a rational multi-grid associated to the prime Pythagorean vector $\mathcal{A} = (a_x, a_y, a_z)^\top$ and a grid point $\vec{p}$. There are three integers such that $A_x = \frac{L}{a_x}, A_y = \frac{L}{a_y}, A_z = \frac{L}{a_z}$ where $L = lcm(a_x, a_y, a_z)$ such that for any grid point $\vec{q} = (q_x, q_y, q_z)^\top$, we have a quadruple of grid points $\{\vec{q}_1 = \vec{q}, \vec{q}_2 = (q_x + A_x, q_y - A_y, q_z)^\top, \vec{q}_3 = (q_x - A_x, q_y, q_z + A_z)^\top, \vec{q}_4 = (q_x, q_y + A_y, q_z - A_z)^\top\}$ satisfying $\mathcal{R}(\vec{q}_1) = \mathcal{R}(\vec{q}_2) = \mathcal{R}(\vec{q}_3) = \mathcal{R}(\vec{q}_4)$.*

This indicates that the triple of integers $A_x, A_y, A_z$ describes the frequency of the repetition of convexels in a rational multi-grid $\mathcal{M}^{\mathcal{A},\vec{p}}$. Let $\mathcal{P}$ be the support plane of $\mathcal{M}^{\mathcal{A},\vec{p}}$. For each grid point $\vec{v} = (v_x, v_y, v_z)$ intersected by $\mathcal{P}$, the grid point $\vec{v}'$ obtained by translation of $v$ by a linear combination of two vectors $(A_x, A_y, 0)^\top$ and $(A_x, 0, A_z)^\top$ has the same formed convexel as that of $\vec{v}$ in $\mathcal{P}$.

Another consequence of Theorem 4.8, illustrated in Figure 4.6, is the existence of triangles that tile the multi-grid. Indeed, let $\vec{q}_1 = (q_x, q_y, q_z)$ be a grid point and $\vec{q}_2 = (q_x + A_x, q_y + A_y, q_z), \vec{q}_3 = (q_x + A_x, q_y, q_z - A_z)$ and $\vec{q}_4 = (q_x + 2A_x, q_y + A_y, q_z - A_z)$ be three points resulting of a translation of $\vec{q}$. The two triangles of vertex $\{\vec{q}_1, \vec{q}_2, \vec{q}_3\}$ and $\{\vec{q}_2, \vec{q}_3, \vec{q}_4\}$

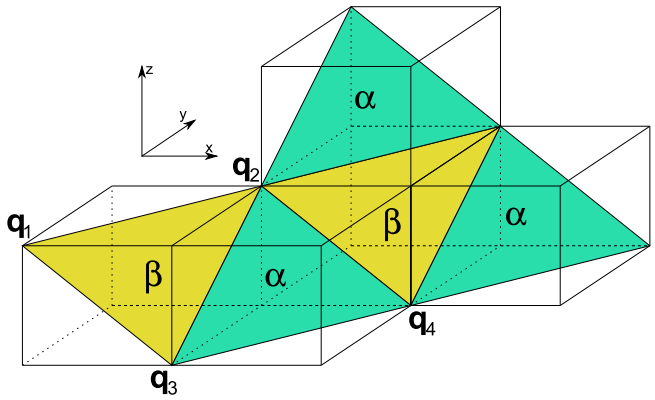



FIGURE 4.6: A quadruple of grid points $\{\vec{q}_1 = \vec{q}, \vec{q}_2 = (q_x + A_x, q_y + A_y, q_z)^\top, \vec{q}_3 = (q_x + A_x, q_y, q_z - A_z)^\top, \vec{q}_4 = (q_x + 2A_x, q_y + A_y, q_z - A_z)^\top\}$ forms two types of triangles, $\alpha$ and $\beta$ tiled in a multigrid.

contain the same set of convexels and are mirror images. Note that triangles formed by four triangles, three $\alpha$ and one $\beta$ or one $\alpha$ and three $\beta$, also tile the plane and contain all convexels. If $A_x, A_y, A_z$ are chosen to be prime, they describe the smallest triangle that tiles $\mathcal{M}^{\mathcal{A},\vec{p}}$.

## 4.4 Hinge angles characterized by a multi-grid

The goal of this section is to show the relation between multi-grids and hinge angles, and then to show how to obtain the unique representation of an integer quintuplet for a 3D hinge angle, namely an injective map from hinge angles to quintuplets. In 2D, any hinge angle can be uniquely represented by a triple of integers as shown in Section 3.2.1. In order to ensure that the representation is unique, some properties on the multi-grid are required. For 3D hinge angles, similar properties on multi-grid are required. Therefore, we first show some properties on the 3D half-grid, then we represent 3D hinge angles using five integers and then explain how to decode them to obtain their hinge angles.

In Section 4.2, we define the 3D hinge angles in the framework of the 3D half-grid. As manipulation of 3D hinge angles is more convenient in the framework of multi-grids and rational multi-grids, we propose an alternative definition of 3D hinge angles in the following

**Proposition 4.9.** *Let $\vec{p}$ be a grid point and $\vec{p}'$ be the result of the rotation of $\vec{p}$ by an angle $\alpha$ around an integer-axis. If $\alpha$ is a hinge angle, then $\vec{p}'$ is on the multi-grid.*

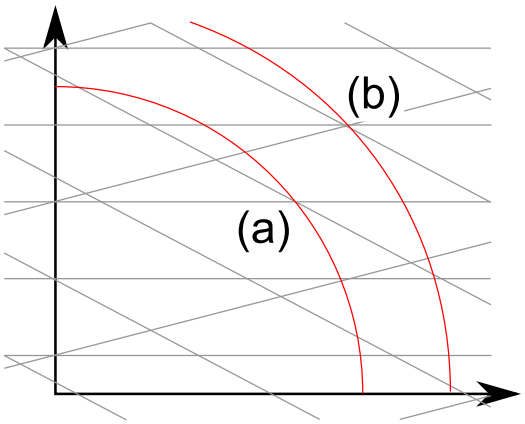



FIGURE 4.7: the two impossible cases of intersection between a circle centered on a grid point and a rational multi-grid.

### 4.4.1 Hinge angles and rational multi-grids

In 2D, there is a property on hinge angles ensuring that the locus of rotation of a grid point cannot contain the intersection of two lines belonging to the half-grid [4]. In this section, we show that a similar property for 3D hinge angles in the rotation plane holds if the rotation axis is an integer axis. In multi-grids, the locus of rotation of a grid point in a rotation plane cannot contain the intersection of two or three lines of the multi-grid as illustrated in Figure 4.7(a) or (b) presented as Lemma 4.10 in the following.

**Lemma 4.10.** *Let $\mathcal{M}^{\mathcal{A},\vec{p}}$ be a multi-grid where $\mathcal{A} \in \mathbb{Z}^3$ and $\vec{p} \in \mathbb{Z}^3$. Then, the locus of the rotation of $\vec{p}$ on the rotation plan $\mathcal{P}$ does not go through any vertex of the convexels on $\mathcal{M}^{\mathcal{A},\vec{p}}$.*

**Proof.** 8. The equation of the rotation plane $\mathcal{P}$ of $\mathcal{M}^{\mathcal{A},\vec{p}}$ is $a_x x + a_y y + a_z z - \mathcal{A}.\vec{p} = 0$. Let $\vec{p'} = (p'_x, p'_y, p'_z)^\top$ be a point which belongs to the locus of the rotation of $\vec{p}$ in $\mathcal{P}$. Let us assume that $\vec{p'}$ is also a vertex of a convexel of $\mathcal{M}^{\mathcal{A},\vec{p}}$, so that it belongs to two planes of the 3D half-grid. Thus we can set, without loss of generality, that $p'_x = k_x + \frac{1}{2}$ and $p'_y = k_y + \frac{1}{2}$ where $k_x, k_y \in \mathbb{Z}$, and then $\vec{p'} = (k_x + \frac{1}{2}, k_y + \frac{1}{2}, p'_z)^\top$.

The locus of the rotation of $\vec{p}$ is the intersection between $\mathcal{P}$ and the sphere $\mathcal{S}$: $x^2 + y^2 + z^2 - (p_x^2 + p_y^2 + p_z^2) = 0$. Thus, by the assumption that $\vec{p'}$ belongs to $\mathcal{P}$ and $\mathcal{S}$, we have

$$a_x\big(k_x + \frac{1}{2}\big) + a_y\big(k_y + \frac{1}{2}\big) + a_z p'_z - \mathcal{A}.\vec{p} = 0, \tag{4.11}$$

$$\big(k_x + \frac{1}{2}\big)^2 + \big(k_y + \frac{1}{2}\big)^2 + p'^2_z - (p_x^2 + p_y^2 + p_z^2) = 0. \tag{4.12}$$

From (4.11) we see that $p'_z$ must be a rational number, so that there is a pair of integers $\lambda_1, \lambda_2$ such that $p'_z = \frac{\lambda_1}{\lambda_2}$ and $\gcd(\lambda_1, \lambda_2) = 1$. From (4.12) we see that $p'^2_z + \frac{1}{2}$ must be

an integer. Thus we have

$$p_z'^2 = \frac{\lambda_1^2}{\lambda_2^2} = k + \frac{1}{2}, \tag{4.13}$$

where $k, \lambda_1, \lambda_2 \in \mathbb{Z}$. Since $\gcd(\lambda_1, \lambda_2) = 1$ and $2\lambda_1^2 = (2k+1)\lambda_2^2$, $\lambda_2$ must be even. Setting $\lambda_2 = 2\lambda_2'$ where $\lambda_2' \in \mathbb{Z}$, we then obtain $2\lambda_1^2 = 4(2k+1)\lambda_2'^2$ and deduce that $\lambda_1$ must be also even, which contradicts the assumption that $\gcd(\lambda_1, \lambda_2) = 1$. Therefore we can conclude that there is not such a point $\vec{p}'$.

This lemma shows that two adjacent convexels, whose transition of $\vec{p}$ during its rotation is done between them, always share a convexel edge. In other words, the rotation locus of $\vec{p}$ passes through a sequence of voxels such that any successive voxels are connected by their common face. Thus, Lemma 4.10 shows that cases (a) and (b) illustrated in Figure 4.7 cannot happen in a rational multi-grid. Note that the above proof only needs the assumption that $\mathcal{A}$ has integer coordinates.

Note that in 2D, there is also another property on hinge angles ensuring that the locus of rotation of a grid point cannot intersect twice on the same line of the half-grid without intersecting another line of the half-grid between the two intersections. In 3D, this property does not hold with rational multi-grids. More details will be discussed in Section 5.2.2.

### 4.4.2 Quintuplet integer representation of hinge angles

A hinge angle in a given rotation plane is represented by a quintuplet of integers $(p_x, p_y, p_z, i, k)$ given by $\alpha(p_x, p_y, p_z, i, k)$. The first three integers $p_x, p_y, p_z$ represent, in the 3D basis $B$, the coordinates of $\vec{p}$. The fourth integer $i$ indicates the index number for the set $\mathcal{L}_i^{\mathcal{A},\vec{p}}, i = 1, 2, 3$, where the hinge angle $\alpha$ is defined. The last integer $k$ represents the index number of the line in $\mathcal{L}_i^{\mathcal{A},\vec{p}}$. Therefore, a quintuplet of integers keeps the coordinates of $\vec{p}$ and the information required for obtaining the coordinates of the arriving point $\vec{q}$ after the rotation of $\vec{p}$ by $\alpha$.

From these five integers, we can obtain, in the basis $B_i$ whose projection matrix $P_{BB_i}$ is given by (4.2), (4.4) and (4.5), the coordinates $(P_x, P_y)^\top$ and $(Q_x, Q_y)^\top$ corresponding to $\vec{p}$ and $\vec{q}$, as represented in Figure 4.8. The coordinates $P_x$ and $P_y$, of $\vec{p}$ in $B_i$, are obtained by applying $P_{BB_i}$ to the coordinates of $\vec{p}$ in $B$. Note that $P_x$ and $P_y$ are rationals. The coordinate $Q_y$ is obtained from one of (4.3), (4.6) and (4.7) depending on the values of $i$ and $k$. All the values in (4.3), (4.6) and (4.7) are rationals, and thus $Q_y$ is also rational. Since $\vec{q}$ belongs to the locus of the rotation of $\vec{p}$, we have $Q_x^2 + Q_y^2 = P_x^2 + P_y^2$. We then notice that $Q_x$ is not a rational number. However, since $P_x, P_y$ and $Q_y$ are rational, all computations involving $Q_x$ can be done using only integer computations.

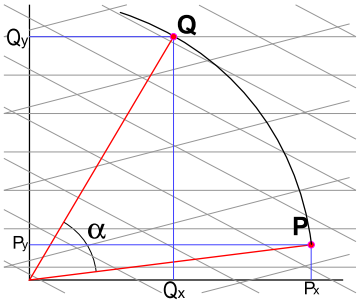



FIGURE 4.8: A hinge angle $\alpha$ for a point $\vec{p}$ in a rational multi-grid.

In general, $Q_x$ can take two values which define two points. To discriminate two different hinge angles corresponding to these two points by the integer quintuplets, we add the positive sign to the fourth integer for the greater $Q_x$ and the negative sign for another value of $Q_x$. If $Q_x = 0$, then the two points merge into one. This particular case where the locus of the rotation of $\vec{p}$ is tangent with the $k$-th line of $\mathcal{M}_i^{\mathcal{A},\vec{p}}$ define no hinge angle since there is no transition of convexel. Hereafter, the representation of hinge angles by integer quintuplets will be denoted by $\alpha(p_x, p_y, p_z, \pm i, k)$

Note that according to Lemma 4.10 we know that any hinge angle cannot rotate a grid point to the intersection of more than one lines of a multi-grid. Therefore, we have the following theorem:

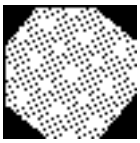
**Theorem 4.11.** *Let $\alpha_p(p_x, p_y, p_z, i_p, k_p)$ and $\alpha_q(q_x, q_y, q_z, i_q, k_q)$ be two hinge angles with their integer quintuplet representations. $\alpha_p = \alpha_q$ if and only if $p_x = q_x$, $p_y = q_y$, $p_z = q_z$, $i_p = i_q$ and $k_p = k_q$.*

Theorem 4.11 is rephrased as: two different integer quintuplets cannot represent the same hinge angle.

We remark that in [4], hinge angles for 2D discrete rotations are represented with a triple of integers that represents the 2D coordinates of the point and the index number of the line that is intersected by the locus of the point's rotation. There is no integer that decides whether the intersected line is parallel to the $x$-axis or the $y$-axis. To differentiate both cases, the last integer of the triple is positive if the line is parallel to $x$-axis or otherwise, it is negative.

## 4.5 3D discrete rotations around a Pythagorean axis

In this section, we develop a 3D discrete rotation based on hinge angles. This rotation is the extension to the 3D space of the 2D discrete rotation given in Section 3.3. In order to obtain the complexity of the algorithm described in this section, we need to enumerate all



the hinge angles existing for an image to rotate. In the case of 2D hinge angles, the upper bound of the number of different hinge angles for an image is $O(m^3)$ where $m$ is the largest side of the image [4, 26]; we will use a similar method for obtaining the upper bound of hinge angles for a 3D image. Our algorithm, to be efficient, needs to compare a pair of hinge angles in constant time. Besides, in order to keep integer computations, we need that the comparison is done using only integers. Note that the hinge angle comparison in constant time is not trivial indeed due to our integer computation constraint. After showing how to compare a pair of hinge angles in constant time with integer computation, we will give the upper bound of the number of hinge angles for a 3D digital image and then present a 3D discrete rotation algorithm for such a given image.

In Section 4.5 we assume that $\mathcal{A}$ is given to be a prime Pythagorean vector.

### 4.5.1 Comparing hinge angles with integer computations

From the integer representation of hinge angles we can obtain the sine and cosine of a hinge angle $\alpha(p_x, p_y, p_z, i, k)$ characterized in the base $B_i$ by the points $\vec{p}$ that have the coordinates $(p_x, p_y, p_z)^\top$ in $B$ and $(P_x, P_y)^\top$ in $B_i$ and $\vec{q}$ that have the coordinates $(Q_x, Q_y)^\top$ in $B_i$. $P_x, P_y, Q_x$ and $Q_y$ can be obtained from $(p_x, p_y, p_z, i, k)$ as explained in Section 4.4.2. The following equations are then derived from Figure 4.8:

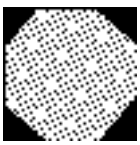
$$\cos\alpha = \frac{P_xQ_x + P_yQ_y}{P_x^2 + P_y^2}, \tag{4.14}$$

$$\sin\alpha = \frac{P_xQ_y - P_yQ_x}{P_x^2 + P_y^2},. \tag{4.15}$$

We remark that if the multi-grid is a rational multi-grid, $P_x, P_y, Q_y$ and $Q_x^2$ are rational. Thus it is possible to compare the sine and the cosine of two hinge angles using only integer computations as follows.

**Proposition 4.12.** *Let $\alpha_1$ and $\alpha_2$ be two hinge angles defined for $\mathcal{A}$. Then it is possible to decide if $\alpha_1 > \alpha_2$ using only integer computations.*

**Proof.** 9. Let $\alpha_1 = \alpha(p_{1x}, p_{1y}, p_{1z}, i_1, k_1)$ and $\alpha_2 = \alpha(p_{2x}, p_{2y}, p_{2z}, i_2, k_2)$. Comparing $\alpha_1$ and $\alpha_2$ is equivalent to comparing their sines and cosines which are given in (4.14) and (4.15). First we compare the signs of both sine and cosine between $\alpha_1$ and $\alpha_2$. If the sines and cosines of both angles have different signs, then we can conclude whether $\alpha_1 > \alpha_2$ without other computation. Otherwise, without loss of generality, we can assume that both $\alpha_1$ and $\alpha_2$ belong to $[0, \frac{\pi}{2}]$, so that $\cos\alpha_i \geq 0$ and $\sin\alpha_i \geq 0$ for both $i = 1, 2$. As the method for comparing two sines is similar to the one for comparing two cosines, we will only show the later one.



If $\alpha_1$ is greater than $\alpha_2$, $\cos\alpha_2 - \cos\alpha_1 > 0$. Thus we have, from (4.14),

$$(P_{1x}^2 + P_{1y}^2)(P_{2x}Q_{2x} + P_{2y}Q_{2y}) > (P_{2x}^2 + P_{2y}^2)(P_{1x}Q_{1x} + P_{1y}Q_{1y}). \tag{4.16}$$

For simplicity, let $A_1 = (P_{1x}^2 + P_{1y}^2)P_{2x}Q_{2x}$, $B_1 = (P_{1x}^2 + P_{1y}^2)P_{2y}Q_{2y}$, $A_2 = (P_{2x}^2 + P_{2y}^2)P_{1x}Q_{1x}$ and $B_2 = (P_{2x}^2 + P_{2y}^2)P_{1y}Q_{1y}$. Note that $A_1^2, B_1, A_2^2, B_2 \in \mathbb{Q}$. Now (4.16) is rewritten as

$$A_1 + B_1 > A_2 + B_2. \tag{4.17}$$

Squaring both sides of (4.17) since they are not negative, and moving rational values to the left-hand side and the rest to the right-hand side, we obtain

$$A_1^2 + B_1^2 - A_2^2 - B_2^2 > 2A_2B_2 - 2A_1B_1. \tag{4.18}$$

If the left-hand side and the right-hand side of (4.18) do not have the same sign, then we can conclude whether $\alpha_1 > \alpha_2$ or $\alpha_2 > \alpha_1$. We can check the sign of both sides of (4.18) with integer computations since the left-hand side contains only rational numbers and the sign of the right side is the same as $A_2^2B_2^2 - A_1^2B_1^2$ which also contains only rational numbers. If signs of both sides are the same, assuming that they are positives, we square both sides of (4.18) to obtain

$$(A_1^2 + B_1^2 - A_2^2 - B_2^2)^2 - 4A_1^2B_1^2 - 4A_2^2B_2^2 > -8A_1B_1A_2B_2. \tag{4.19}$$

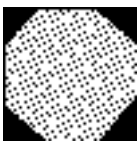
If the sign of the left-hand side of (4.19) is positive, we can deduce that $\alpha_2 > \alpha_1$. Otherwise, taking the square of each side gives us

$$\left[(A_1^2 + B_1^2 - A_2^2 - B_2^2)^2 - 4A_1^2B_1^2 - 4A_2^2B_2^2\right]^2 < 64A_1^2B_1^2A_2^2B_2^2. \tag{4.20}$$

We note that we can easily verify whether (4.20) is satisfied with integer computation alone. If (4.20) is true, $\alpha_1 < \alpha_2$; otherwise $\alpha_2 < \alpha_1$. The same logic can be applied to the case where the signs of the both sides of (4.18) are negative.

Thanks to Proposition 4.10, a hinge angle cannot have two quintuplet integer representations. Thus we can conclude that the comparison of a pair of hinge angles $\alpha_1, \alpha_2$ is always possible with integer computation if they have different quintuplets.

Note that, if the comparison of two hinge angles is done using floating numbers, only one comparison is required. However, if we want to keep integer computations, then in the worst case we have to check (4.18), (4.19) and (4.20). From Proposition 4.12, we now have a guarantee of a constant number for each comparison. Then we conclude that the comparison of a pair of hinge angles is done in constant time.



The following proposition is required for our algorithm of 3D discrete rotation. In the 2D case, is it known that the comparison between a hinge angle and a Pythagorean angle can also be done using only integers during its computation and in constant time. An angle is a Pythagorean angle if its sine and cosine belong to $\mathbb{Q}$ [4]. In the 3D case, a similar proposition will be still valid.

Note that the proof of Proposition 4.13 is similar to the proofs of Proposition 4.12 and on comparison for a hinge angle and a Pythagorean angle in the plane presented in Section 3.3.1.

**Proposition 4.13.** *Let $\theta$ be a Pythagorean angle and $\alpha$ be a hinge angle defined for $\mathcal{A}$. Then it is possible to decide if $\alpha > \theta$ in constant time with integer computations.*

**Proof.** 10. Let $\alpha = \alpha(p_x, p_y, p_z, i, k)$ be an hinge angle and $\theta$ be a Pythagorean angle associated to the Pythagorean triple $(i_1, i_2, \lambda)$ such that $\cos\theta = \frac{i_1}{\lambda}$ and $\sin\theta = \frac{i_2}{\lambda}$ where $i_1, i_2, \lambda \in \mathbb{Z}$.

If $\alpha$ is greater than $\theta$, $\cos\theta - \cos\alpha > 0$ thus we have from (4.14),

$$i_1(P_x^2 + P_y^2) > \lambda(P_xQ_x + P_yQ_y). \tag{4.21}$$

Using similar methods to those used for the proof of Proposition 4.12, we obtain the equation:

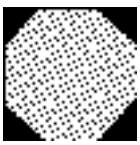
$$(i_1(P_x^2 + P_y^2) - \lambda P_yQ_y)^2 > \lambda^2 P_x^2 Q_x^2. \tag{4.22}$$

We note that we can easily verify whether (4.22) is satisfied with integer computation alone. If (4.22) is true, $\alpha < \theta$; otherwise $\theta < \alpha$.

### 4.5.2 Upper bound of the number of 3D hinge angles

In the 2D case, the upper bound of the number of hinge angles for a given image of size $m \times m$ is known to be $O(m^3)$ [4]. This is obtained by computing the bound for the furthest point from the origin in the image and multiplying this bound by the number of points in the image. To compute the number of hinge angles in the 3D case, we will use a similar method.

In the 3D case, we assume that an image of size $m \times m \times m$ is given. The number of hinge angles for a given point $\vec{p}$ depends on the distance between $\vec{p}$ and the axis of rotation $\mathbb{A}$. Therefore, we define the distance function $d(\vec{p})$ that is the Euclidean distance between $\mathbb{A}$ and $\vec{p}$. Then, the rotation of $\vec{p}$ around $\mathbb{A}$ intersects at most $3\lfloor d(\vec{p})\rfloor$ planes of



**Algorithm 4** Rotation of a 3D image around an integer-axis whose direction is $\mathcal{A}$ by a Pythagorean angle $\theta$.

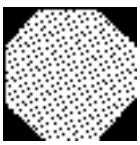
**Require:** An image $\mathcal{I}$, a vector $\mathcal{A}$, a Pythagorean angle $\theta$
**Ensure:** A rotated image $\mathcal{I}'$
**for all** points $\vec{p}$ in $\mathcal{I}$ **do**
  Set $T$ to be an empty list;
  Compute the three generic equations (4.3), (4.6) and (4.7) for each $\mathcal{M}_i^{\mathcal{A},\vec{p}}, i = 1,2,3$;
  **for all** lines in $\mathcal{M}_i^{\mathcal{A},\vec{p}}, i = 1,2,3$ **do**
    Compute all hinge angles corresponding to $\vec{p}$ and the current line,
    and add $\alpha(p_x, p_y, p_z, \pm i, k)$ to the list $T$;
  **end for**
  Sort all the hinge angles corresponding to $\vec{p}$ in $T$;
  Search in $T$ the greatest hinge angle $\alpha$ which is smaller than $\theta$;
  Copy the voxel color from $\vec{p}$ in $\mathcal{I}$ to the rotated point $\vec{q}$ with $\alpha$ in $\mathcal{I}'$
**end for**
**return** $\mathcal{I}'$;

the half-grid and defines at most $6\lfloor d(\vec{p})\rfloor$ different hinge angles. Because $d(\vec{p}) \leq \sqrt{3}m$, the upper bound of the number of hinge angles for any point in the image is $6\sqrt{3}m$. Accordingly, we can conclude that the upper bound of the number of hinge angles for a given image of size $m \times m \times m$ is $6\sqrt{3}m^4$; thus $\mathrm{O}(m^4)$.

### 4.5.3 3D discrete rotations induced by hinge angles

In this section, we explain how to design a discrete rotation of a 3D digital image using hinge angles for a given Pythagorean axis of rotation. This method is the 3D extension of the 2D discrete rotation described in Section 3.3.3. As input of such discrete rotation, we have a digital image $\mathcal{I}$ of size $m \times m \times m$, a vector $\mathcal{A}$, supposed to be a prime Pythagorean vector, and an angle $\theta$ supposed to be Pythagorean. The assumption that the rotation axis is a Pythagorean axis and the angle is a Pythagorean angle does not restrict the field of possible rotations. Indeed, it is proved in [28] that the Pythagorean vectors are dense on the 3D unit sphere and it is proved in [15] that any angle can be approximated with a small difference $\epsilon > 0$ by a Pythagorean angle. The output of our algorithm is a rotated digital image $\mathcal{I}'$.

The rotation algorithm is described in Algorithm 4. For each point $\vec{p} = (p_x, p_y, p_z)^\top$ in the image $\mathcal{I}$, the algorithm computes the corresponding multi-grid $\mathcal{M}^{\mathcal{A},\vec{p}}$ and searches for each line $k$-th in $\mathcal{M}_i^{\mathcal{A},\vec{p}}, i = 1,2,3$, a pair of hinge angles $\alpha(p_x, p_y, p_z, \pm i, k)$. Then we stock and sort all hinge angles corresponding to $\vec{p}$ using Proposition 4.12. The algorithm searches in the sorted list $L$ the greatest hinge angle $\alpha$ which is smaller than $\theta$ using Proposition 4.13. This operation can be done using only integer computations thanks



to our assumption that our input angle $\theta$ is a Pythagorean angle. Finally the new point after the rotation of $\vec{p}$ by $\alpha$ is generated in $\mathcal{I}'$.

The time complexity of this algorithm is $\mathrm{O}(m^4 \log m)$. The computation and the sorting of all hinge angles for each point is done in $\mathrm{O}(m \log m)$ operations because the comparison between two hinge angles is done in constant time according to Proposition 4.12. Searching the largest hinge angle $\alpha$ smaller than $\theta$ is done in $\mathrm{O}(\log m)$ operations because the comparison between a hinge angle and a Pythagorean angle can be performed in constant time according to Proposition 4.13. Therefore, the time complexity of this algorithm is $\mathrm{O}(m^4 \log m)$ because we repeat $m^3$ times the previous operations.

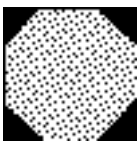
### 4.5.4 3D incremental discrete rotation

In this section, we present an incremental algorithm that performs a 3D discrete rotation. In [4], the author describes a 2D incremental rotation that allows obtaining all possible configurations of 2D discrete rotations for a given image. Similarly, 3D hinge angles on rotation planes also allow us to design a 3D incremental discrete rotation. This algorithm may help us to understand the configurations of 3D discrete rotations even thought we do not see yet its practical uses.

For the incremental rotation algorithm, we consider the input data to be a digital image $\mathcal{I}$ of size $m \times m \times m$ and a given rotation axis.

The incremental algorithm consists in four main steps. The first step generates a table of $m \times m \times m$ pointers that refers to every voxel of $\mathcal{I}$. Then, for each point in $\mathcal{I}$ it computes and sorts all hinge angles associated to this point, and stores them into a list. The third step merges the lists of all points into the final list and then sorts all the hinge angles in the list. The final step browses in ascending order the list containing all the hinge angles and displaces the voxel corresponding to the current hinge angle to its neighborhood.

Firstly, the algorithm initializes a table $\mathcal{T}$ of $m \times m \times m$ pointers, such that each pointer $(i, j, k)$, where $i, j, k \in [0, m]$, refers to the grid point $(i, j, k)$. $\mathcal{T}$ is used to track the voxel location during the incremental rotation. Then for each point $\vec{p} = (p_x, p_y, p_z)^\top$ in $\mathcal{I}$, the algorithm computes and stores in the list $T_{\vec{p}}$ all hinge angles $\alpha(p_x, p_y, p_z, i, k)$ for $i = 1, 2, 3$ and $k = -l, -l+1, \ldots, l-1, l$ where $l = \lfloor \sqrt{p_x^2 + p_y^2 + p_z^2} \rfloor$. Hinge angles are stored using their integer representations. Then each list $T_{\vec{p}}$ is made in ascending order of hinge angles. The second step merges all the lists $T_{\vec{p}}$ into a unique list. In order to avoid the sorting of the final list $T$ this operation should be done using the merge sort algorithm [29]. The last step of the algorithm performs the incremental rotation. Reading every hinge angle $\alpha(p_x, p_y, p_z, i, k)$ in $T$, we displace the grid point referred by $\mathcal{T}(p_x, p_y, p_z)$ to



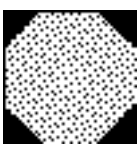
one of its 6-connected voxel according to $i$ and $k$. The pointer in $\mathcal{T}(p_x, p_y, p_z)$ is updated to the new position so that the actual position of $\vec{p}$ will be known using $\mathcal{T}(p_x, p_y, p_z)$. We repeat this operation while hinge angles are remaining in $T$. When all the hinge angles in $T$ have been processed, every point has described its full rotation locus around a given axis of rotation. Then all the possible configurations of rotations of $\mathcal{I}$ around the rotation axis have been reached.

The time complexity for the first step is $\mathrm{O}(m^4 \log m)$ since there is $m^3$ points to consider and for each point, there are at most $\mathrm{O}(m)$ hinge angles. Thus the sorting of each list is done in $\mathrm{O}(m \log m)$. The second step requires the same time complexity than the first step. Note that if the merging does not use the method presented in [29], the time complexity remains the same, but it will practically increase the computation time. The last step browses the list $T$ that contains at most $\mathrm{O}(m^4)$ hinge angles. Each displacement is done in constant time and thus it requires $\mathrm{O}(m^4)$ operations. We conclude that the time complexity for the incremental rotation algorithm is $\mathrm{O}(m^4 \log m)$.

## 4.6 Conclusion

In this chapter, we extended the notion of hinge angles, introduced Chapter 3 to the 3D. Extension of hinge angles from the 2D to 3D space involves many problems because most of the 2D hinge angles properties are not valid for 3D hinge angles. In order to regard hinge angles in the 3D space similarly to the 2D ones, we introduced the notion of a multi-grid that is the intersection of the 3D half-grid and a rotation plane. By redefining the hinge angles on the rotation plane, which are the extension of hinge angles for the rotation in 2D, we showed a subgroup of the multi-grids where all parameters are rational, called rational multi-grids. This subgroup allows us to compare two hinge angles on rotation planes in constant time by using integer computations. It also allows us to design a 3D discrete rotation and a 3D incremental rotation.

The multi-grids introduced in this paper can be extended in any dimension to perform discrete rotations. Roughly speaking, a rotation in $n$D requires $\lfloor \frac{n}{2} \rfloor$ angles and rotation planes. However, rotations in $n$D are not well understood, for example, in Euclidean space we do not know yet if a given $n$D rotation can be uniquely decomposed into $\lfloor \frac{n}{2} \rfloor$ planar rotations [30]. Therefore, the extension of the work presented in this paper to $n$D discrete rotations requires a study of decomposition of $n$D discrete rotation into 2D discrete rotation.



# Chapter 5

# Estimation of rotation between a pair of 3D digital images

## 5.1 Introduction

This chapter is the extension to the 3D space of our algorithm to find the admissible rotation angles for a pair of 2D digital images presented in Chapter 3. We present a discrete method to find from a pair of 3D digital images admissible rotations that transform the first digital image into the second digital image. We assume that the points correspondence between these two 3D digital images are given.

The estimation of a rotation in 3D discrete space is done by estimating a rotation angle and a rotation axis. It is also possible to estimate a 3D discrete rotation by estimating three angles and the rotation center. However, as explained in Chapter 2 and Chapter 4, it is preferable to avoid rotations composition.

Hereafter, we assume that the given data are two or $n$ pairs of points in correspondence. The points $\mathbf{p}_1 = (p_{1x}, p_{1y}, p_{1z})$ and $\mathbf{p}_2 = (p_{2x}, p_{2y}, p_{2z})$ are said to be in correspondence if $\mathbf{p}_1$ in the first image corresponds to $\mathbf{p}_2$ in the second image.

In this chapter, we show how to deduce information on the rotation axis and the rotation angle that transform the first image in the second image from $n$ pairs of points in correspondence. First, we give a method that approximates the rotation axis from $n$ pairs of grid points in correspondence. Secondly, we give a method to obtain the admissible rotation angles knowing a pair of discrete points in correspondence and the rotation axis.

## 5.2 Finding a 3D discrete rotation from a pair of digital images

In this section, we present a method to find a discrete rotation between a pair of 3D digital images that represents the same object from two different points of view. Suppose that we have a set of $n$ grid points $A = (\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n)$ in the first image and its corresponding set $B = (\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n)$ in the second image. Each pair of points $(\vec{a}_i, \vec{b}_i)$ corresponds to the same point of the object. We say that such a pair of grid point sets, $A$ and $B$, are in correspondence.

In 2D, finding a discrete rotation between such a pair of $A$ and $B$ mainly consists in identifying a set of angles that give the same rotated image as explained in Chapter 3. In Euclidean space, such a rotation is unique. This is not the case in discrete space where two slightly different angles $\alpha_1, \alpha_2$ may transform $A$ into $B$. Moreover, for such pair $\alpha_1, \alpha_2$, each angle $\alpha_3$ such that $\alpha_1 \leq \alpha_3 \leq \alpha_2$ also rotate $A$ into $B$. Then, we define the admissible rotation angle, abbreviated by ARA, that is the set of all angles that give the same rotated digital image. It is bounded by the pair of hinge angles $\alpha^{inf}$ and $\alpha^{sup}$. In 3D, it is necessary to find not only a set of admissible rotation angles but also an admissible rotation axis.

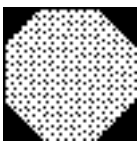
We note that we identify a set of admissible rotation angles but only one admissible rotation axis. Indeed, if we consider two rotation axis, the two multi-grids associated to these axes and a point will provide two distinct sets of hinge angles. Therefore, we set a unique rotation axis.

### 5.2.1 Approximation of the rotation axis by a 3D Pythagorean vector

The determination of an admissible rotation axis from a pair of digital images in correspondence requires two steps. The first step is to obtain a rotation axis that is consistent with all pairs of points in correspondence. The second step is to approximate the axis obtained through the first step by a Pythagorean axis in order to obtain rational multi-grids.

#### 5.2.1.1 First step

Given a pair of Euclidean points in correspondence, it is known that any rotation axis such that the first point is transformed into the second one belongs to the bisection plane of these two points. If we consider two pairs of Euclidean points in correspondence, their rotation axis is the intersection between their two bisection planes. If we add a third pair



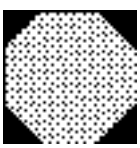
of Euclidean points in correspondence, the intersection between the three bisection plane still give the same rotation axis. However, if we consider three pairs of grid points in correspondence instead of Euclidean points, the intersection between the three bisection plane may not be unique. In other words, we may have three rotation axes because grid points approximate Euclidean points. Therefore, in order to obtain a unique rotation axis, we consider the rotation axis that is the average of the three obtained rotation axis.

Now, we consider voxels instead of Euclidean or grid points. For a given pair of voxels in correspondence, the bisection plane is not unique. Indeed, each pair of Euclidean points that belongs to the pair of voxels in correspondence define a bisection plane. Therefore, there is an infinity of admissible bisection plane for a pair of voxels. Computing the bounds of all the bisection planes for a pair of voxels in correspondence is a big issue. If $n$ pairs of voxels in correspondence are given, and to define an admissible rotation axis for the $n$ pairs of voxels, we have to compute the intersections between all the obtained bounds of admissible bisection planes. In 2D, a study on the intersection of the bound of bisection for $n$ pairs of pixels have been proposed [31] and can be extended to the 3D discrete space. However, this work is not finished yet. Therefore, in order to obtain a rotation axis for $n$ pairs of voxels, we consider the $n$ pairs of grid points, which are the centers of voxels, and compute for each pair the bisection plane. Then we compute $\lfloor \frac{n+1}{2} \rfloor$ axes from the $n$ obtained bisection planes. The rotation axis is the average of the $\lfloor \frac{n+1}{2} \rfloor$ obtained axes. We strongly believe that this method gives a rotation axis that belongs to the intersections between all the bounds of admissible bisection planes. However, we do not have any formal proof yet.

#### 5.2.1.2 Second step

In this section, we show a method to approximate the rotation axis obtained in the previous section by a Pythagorean axis that is required to compute a rational multi-grid. Based on [15], we can derive a naive method to approximate any 3D vector by a 3D Pythagorean vector; its main idea is to construct two 2D Pythagorean vectors to approximate a 3D Pythagorean vector. For this naive method, some simple definitions are required. We call a Pythagorean triple a set of three integers $(i_1, i_2, \lambda)$ such that $i_1^2 + i_2^2 = \lambda^2$. Each Pythagorean triple is associated with the Pythagorean vector $(i_1, i_2)^\top$ and a Pythagorean angle $\theta$ such that $\cos\theta = \frac{i_1}{\lambda}$. Pythagorean quadruples are the sets of four integers $(i_1, i_2, i_3, \lambda)$ such that $i_1^2 + i_2^2 + i_3^2 = \lambda^2$. Each Pythagorean quadruple is associated with the Pythagorean vector $(i_1, i_2, i_3)^\top$.

The method presented by Anglin in [15] allows the approximation of any angle $\alpha$ with a Pythagorean angle $\theta$ such that $|\alpha - \theta| < \epsilon$ for any $\epsilon > 0$. It uses the theorem, given

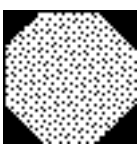


in [32], that for each Pythagorean triple $(i_1, i_2, \lambda)$, there is a pair of integers $(u, v)$ such that $v < u$, $i_1 = 2uv$, $i_2 = u^2 - v^2$ and $\lambda = u^2 + v^2$. Setting $X = \tan(\alpha - \epsilon) + \sec(\alpha - \epsilon)$ and $Y = \tan(\alpha + \epsilon) + \sec(\alpha + \epsilon)$, Anglin proved that for every pair of integers $(u, v)$ such that $X < \frac{u}{v} < Y$, the angle $\theta$ associated with the Pythagorean triple generated by $(u, v)$ approximates $\alpha$ in such that $\alpha - \epsilon < \theta < \alpha + \epsilon$.

Let $\vec{r} = (r_1, r_2, r_3)^\top$ be a 3D real vector to be approximated by a Pythagorean vector such that the angle between the two vectors is smaller than $\epsilon$. Such a pair of vectors is said $\epsilon$-close. From $\vec{r}$, we define two 2D vectors $\vec{r}_2, \vec{r}_3$ such that $\vec{r}_2 = (r_1, r_2)^\top, \vec{r}_3 = (\sqrt{r_1^2 + r_2^2}, r_3)^\top$. Each vector $\vec{r}_i = (x_i, y_i)^\top$ is associated with an angle $\alpha_i$ satisfying $\cos\alpha_i = \frac{x_i}{|\vec{r}_i|}$. Using the algorithm presented in [15] for these two angles $\alpha_i$ with a precision $\epsilon$, we obtain two Pythagorean angles associated with the two Pythagorean triples $(i_2, j_2, \lambda_2), (i_3, j_3, \lambda_3)$. Now, we consider the Pythagorean vectors associated with the two Pythagorean triples. We remark that if $(i_2, j_2, \lambda_2)$ is a Pythagorean triple, then $(ki_2, kj_2, k\lambda_2)$ is also a Pythagorean triple for any $k \in \mathbb{Z}^*$. Note that, these two Pythagorean triples are associated with the same Pythagorean angle. This remark allows us to generate from the two Pythagorean triples, $(i_2, j_2, \lambda_2), (i_3, j_3, \lambda_3)$, a Pythagorean quadruple $(ki_2, kj_2, lj_3, l\lambda_3)$ with two integers $k, l$ such that $k(i_2 + j_2) = li_3$. The 3D vector $\vec{r}'$ associated with this Pythagorean quadruple $(ki_2, kj_2, lj_3, l\lambda_3)$ is our approximation result of $\vec{r}$.

In [15], the author introduced $\epsilon$ that corresponds to the maximum angle between the original vector and the Pythagorean vector that approximates it. With the above method, to construct a 3D Pythagorean vector that approximates a given vector by $\epsilon$, we apply the Anglin's method twice. An example of the method that approximates a 3D vector is given in Figure 5.1. First we decompose the 3D vector $\vec{r}$ to approximate into the two 2D vectors $\vec{r}_2$ and $\vec{r}_3$. $\vec{r}_2$ and $\vec{r}_3$ can be approximated by Anglin's method with a precision of $\epsilon$ by two 2D Pythagorean vectors $\vec{r}'_2$ and $\vec{r}'_3$ that belong to the two 2D convex cones illustrated in Figure 5.1 (a) and (b), respectively. We generate $\vec{r}'$ from $\vec{r}'_2$ and $\vec{r}'_3$, which is the approximation of $\vec{r}$, so that it belongs to the 3D convex cone represented in Figure 5.1 (c) forming a square pyramid. The minimum circular cone including the square pyramid has the solid angle $2\sqrt{2}\epsilon$. Therefore, we can deduce that to reach a precision of $\epsilon'$ while approximating a 3D vector, we need to give to Anglin's method a precision of $\epsilon \leq \frac{\epsilon'}{\sqrt{2}}$.

The algorithm given in this section computes very quickly because, according to Anglin, the computation of a Pythagorean triple is done in constant time. Accordingly, we can give the approximation of any 3D vector in O(1) operations. However, we cannot give any bounds on the size of integers that belongs to the final Pythagorean quadruple.



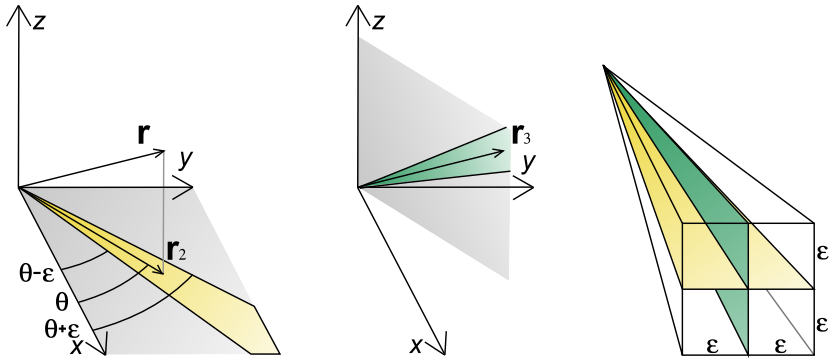



FIGURE 5.1: Example of the approximation of a 3D vector using our method. (a) and (b) represent the projections of $\vec{r}$ into $\vec{r}_2$ and $\vec{r}_3$ in 2D planes respectively. Each 2D convex cone in (a) and (b) is the admissible approximation for $\vec{r}_2$ and $\vec{r}_3$ regarding to $\epsilon$. (c) represents the square pyramid constructed from the two 2D convex cones that contains the sets of admissible approximation for $\vec{r}$.

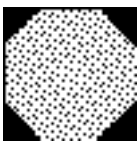
### 5.2.2 Approximation of the rotation angle using rational multi-grids

After obtaining an admissible rotation axis, the next step is to obtain the admissible rotation angles. For the 2D case, we have developed, in Section 3.4, a method to obtain from $n$ pairs of grid points the admissible rotation angles where $n \geq 1$ in time complexity $O(n)$. This method is designed to work in the 2D half-grid. With minor modifications, this method can also be applied on rational multi-grids and keeps the same complexity.

Firstly, we remind the definition of admissible rotation angles abbreviated by $ARA$ and introduced in Section 3.3. The $ARA$ for the two sets of $n$ points $A = (\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n)$ and $B = (\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n)$ in correspondence is the set of angles defined by an upper and a lower bounds such that any angle between them gives the same discrete rotation from $A$ to $B$. We denote by $ARA(\vec{a}_i, \vec{b}_i) = (\alpha_i^{\inf}, \alpha_i^{\sup})$ the pair of hinge angles that gives the lower and the upper bounds of $ARA$ for the pair of points $(\vec{a}_i, \vec{b}_i)$.

For a given pair of points $(\vec{a}_i, \vec{b}_i)$ in correspondence, we first compute the rational multi-grid $\mathcal{M}^{A,\vec{a}_i}$. Then we search for the convexel $c$ containing $\vec{q}$ and compute the two hinge angles between $c$ and the circle centered at the origin of the rational multi-grid and going through $\vec{a}_i$. These two intersections define the $ARA$ for this pair of points. We note that each operation are done in constant time.

Generally, the given input contains $n$ pairs of points instead of one pair of points. We then incrementally compute the upper and lower bounds of the $ARA$ for these $n$ pairs of points. We first compute the $ARA$ corresponding to the two first pair of points $(\vec{a}_1, \vec{b}_1)$ and $(\vec{a}_2, \vec{b}_2)$. We compare the two pairs of hinge angles $(\alpha_1^{inf}, \alpha_1^{sup})$ and $(\alpha_2^{inf}, \alpha_2^{sup})$ obtained and keep the two most restrictive ones so that we have the new pair of hinge



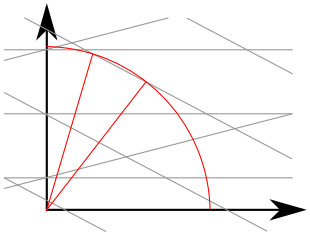


FIGURE 5.2: In a rational multi-grid $\mathcal{M}^{\mathcal{A},\vec{p}}$ a convexels can have four intersections with the locus of rotation of $\vec{p}$.

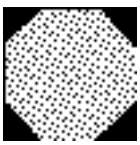
angles $(\max(\alpha_1^{inf}, \alpha_2^{inf}), \min(\alpha_1^{sup}\alpha_2^{sup}))$. We incrementally repeat this operation for the $n-2$ remaining pairs of points and obtain $ARA(A,B) = \cap_i ARA(\vec{a}_i, \vec{b}_i)$ for the $n$ pairs of points. Since the comparison of a pair of hinge angles is done in constant time, the time complexity of the incremental algorithm is $\mathrm{O}(n)$.

In the above algorithm, we assumed that there are always two intersections between a convexel and the locus of the rotation of $\vec{p}$. Practically, it may be possible that the locus of rotation of $\vec{p}$ intersects twice a line of $\mathcal{M}^{\mathcal{A},\vec{p}}$ without intersecting any other line of $\mathcal{M}^{\mathcal{A},\vec{p}}$ as illustrated in Figure 5.2. In other words, four intersections between a convexel and the rotation locus may exist. We perform experiments in order to evaluate the frequency of such a case. We randomly generate ten thousands Pythagorean axes and for each axis we randomly generate ten thousands grid points. For each grid points $\vec{p}$, we compute all intersections between $\mathcal{M}^{\mathcal{A},\vec{p}}$ and the locus of rotation of $\vec{p}$ and search if four of these intersections belong to the same convexel. About 3% of generated points have such particular convexels and the probability that a convexel intersected by the locus of rotation of $\vec{p}$ have four intersections is $\frac{1}{62500}$. Since this particular case does not often occur, we chose not to take it into consideration in our algorithm. Note that in 2D such a case does not happen and the proof can be found in Chapter 3.

## 5.3 Conclusion

In this chapter, we have introduced a method to obtain the rotation angle and rotation axis from a pair of digital images. This work is the extension of the work presented in Section 3.4 of Chapter 3.

In Section 5.2, we introduce the admissible rotation axis that is the set of all possible rotation axis for a pair of digital image. The shape of such a set is not yet studied, however in [31] the authors studied in discrete plane the shape of admissible center of rotation for $n$ pairs of points. Thus, one of the future work will be to extend it in 3D and then in $n$D.



# Chapter 6

# Study on Pythagorean n-tuples

## 6.1 Introduction

In Chapter 4, we introduced a method that performs discrete rotations in 3D using the multi-grid. We explained that to be discrete, this method needs that the rotation axis to be Pythagorean. But in order to approximate all possible 3D rotations it is necessary that Pythagorean vectors, that are the direction vector of Pythagorean axis, are dense in 3D.

The Pythagorean triples are the triple of integers $(i_1, i_2, \lambda)$ such that $i_1^2 + i_2^2 = \lambda^2$. They are well known since Pythagoras, and many studies have been done such as studies on their repartition in the plane [33], how to generate them [34], how to approximate them [15] and the density of their representation [35]. The natural extension of the Pythagorean triples is Pythagorean quadruples that are the quadruple of integers $(i_1, i_2, i_3, \lambda)$ such that $i_1^2 + i_2^2 + i_3^2 = \lambda^2$. Some of the studies on Pythagorean triples have been done for Pythagorean quadruples such as generation [36, 37]. The property of density have been proved arithmetically in [28], here we propose a constructive proof that Pythagorean vectors are dense in 3D.

In this chapter, we first introduce the Pythagorean triples and a sketch of the proof of their density. Then we extend this result to Pythagorean quadruples and to Pythagorean $n$-tuples. Finally we present a naive algorithm to approximate any sets of $(n-1)$ integers by a Pythagorean $n$-tuple.

## 6.2 Pythagorean triples

In this section, we first define the Pythagorean triples and recall some known properties. Then we give the proof of the density of Pythagorean triples, which may originally be proved by Pythagoras [35, 38]. This result will be used in the subsequent sections for the proof of density of Pythagorean quadruples and $n$-tuples.

**Definition 6.1.** A Pythagorean triple is an integer triple $(i_1, i_2, \lambda)$ such that $i_1^2 + i_2^2 = \lambda^2$.

We denote by $\mathcal{P}_2$ the set of Pythagorean triples such that $\mathcal{P}_2 = \{(i_1, i_2, \lambda) \in \mathbb{Z}^3 |\ i_1^2 + i_2^2 = \lambda^2\}$. There are two geometrical representations for each element in $\mathcal{P}_2$ : the angle and the vector.

The first representation of a Pythagorean triple $(i_1, i_2, \lambda)$ can be done by the angle between the line $i_1 x - i_2 y = 0$ and the abscissa axis (Figure 6.1).

**Definition 6.2.** A Pythagorean angle associated with a Pythagorean triple $(i_1, i_2\ , \lambda)$ of $\mathcal{P}_2$ is an angle $\theta$ such that $\sin\theta = \frac{i_2}{\lambda}$ and $\cos\theta = \frac{i_1}{\lambda}$.

*Remark* 6.3. If a triple of integers $(i_1, i_2, \lambda)$ belongs to $\mathcal{P}_2$, the triples $(ki_1, ki_2, k\lambda)$ for all $k \in \mathbb{Z}$ also belong to $\mathcal{P}_2$; they are associated with the same Pythagorean angle.

We denote by $\mathcal{A}_{\mathcal{P}_2}$ the set of Pythagorean angles. Roughly speaking, Pythagorean angles are the representation of Pythagorean triples in $\mathbb{R}$.

Before showing that $\mathcal{A}_{\mathcal{P}_2}$ is dense in $\mathbb{R}$, we give the classic definition of density in $\mathbb{R}$.

**Definition 6.4.** [39] A set $S$ is dense in $\mathbb{R}$, if for any element $e_1$ in $\mathbb{R}$ and for any $\epsilon \in \mathbb{R}$, there is an element $e_2 \in S$ such that $e_1 < e_2 < e_1 + \epsilon$.

Regarding Definition 6.4, we have the following theorem.

**Theorem 6.5.** *The set of Pythagorean angles $\mathcal{A}_{\mathcal{P}_2}$ is dense in $\mathbb{R}$.*

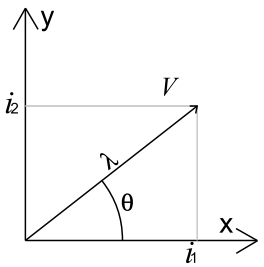



FIGURE 6.1: Two representations of a Pythagorean triple $(i_1, i_2, \lambda)$: Pythagorean angle $\theta$ and Pythagorean vector $\vec{v}$ in the plane

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | ••• |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i^2$ | 0 | 1 | 4 | 9 | 16 | 25 | 36 | 49 | 64 | 81 | 100 | 121 | 144 | 169 | ••• |
| $(i+1)^2-i^2$ | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 | 23 | 25 | | ••• |

{4,3,5} {12,5,13}

FIGURE 6.2: Construction of all Pythagorean triples of the form $(i_1, i_2, i_1 + 1)$

One of simple proofs of Theorem 6.5 uses the two following lemmas.

**Lemma 6.6.** *[40] The sum of any two Pythagorean angles is a Pythagorean angle.*

In other words, $\mathcal{A}_{\mathcal{P}_2}$ is closed by addition. It is easy to admit this lemma, because the sum of two angles consists of additions and multiplications of the sines and cosines of the two given angles. Thus, the sine and cosine of the angle resulting from the addition still belong to rational numbers.

**Lemma 6.7.** *For any angle $\alpha \in [0, 2\pi]$, there is an angle $\theta \in [0, 2\pi]$ of $\mathcal{A}_{\mathcal{P}_2}$ such that $\theta < \alpha$.*

The proof of Lemma 6.7 is similar to a proof of the statement, known since Pythagoras, that the number of Pythagorean triples is infinity. The idea of this proof is illustrated in Figure 6.2; each time we find a square of an integer on the third line, we can form a Pythagorean triple with the two corresponding integers on the first line. Since the third line contains all the odd number squares, we deduce that there is an infinity of Pythagorean triples of the form $(i_1, i_2, i_1+1)$. It is obvious that $i_1 > i_2$ and $\lim_{i_1 \to \infty} \frac{i_2}{i_1} = 0$. Therefore, when $i_1 \to \infty$, the associated angle $\theta \to 0$. This induces Lemma 6.7.

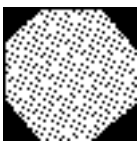
The main idea to prove Theorem 6.5 is to show that for any angle $\alpha \in [0, 2\pi]$ and for any $\epsilon > 0$ there is a Pythagorean angle $\theta$ such that $\alpha < \theta < \alpha + \epsilon$. To obtain such $\theta$, we first find a Pythagorean angle $\theta'$ such that $\theta' < \alpha$ and a Pythagorean angle $\gamma$ such that $\gamma < \epsilon$, thanks to Lemma 6.7. From Lemma 6.6, we know that the angles resulting from the sums $\theta' + \gamma$, $\theta' + 2\gamma, \ldots, \theta' + n\gamma$ are Pythagorean angles as well. Therefore, we conclude that there is an integer $n$ such that $\theta = \theta' + n\gamma$ and $\alpha < \theta < \alpha + \epsilon$.

The second representation of a Pythagorean triple is a vector as illustrated in Figure 6.1.

**Definition 6.8.** A 2D Pythagorean vector is a vector $\vec{v} = (i_1, i_2)^\top$ associated with the element $(i_1, i_2, \lambda)$ of $\mathcal{P}_2$.

Note that the Euclidean norm of a Pythagorean vector is always an integer. We denote by $\mathcal{V}_{\mathcal{P}_2}$ the set of 2D Pythagorean vectors.

To each Pythagorean triple, we have its corresponding Pythagorean angle and Pythagorean vector respectively. Thus, intuitively, we know that $\mathcal{V}_{\mathcal{P}_2}$ is dense because $\mathcal{A}_{\mathcal{P}_2}$ is dense.



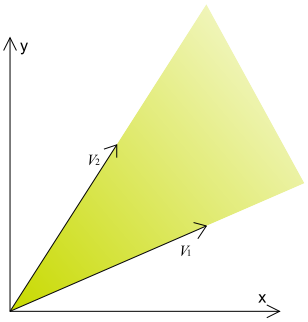



FIGURE 6.3: A convex cone in the plane defined by the pair of vector $\vec{v}_1, \vec{v}_2$

To define the density of a set of such vectors, we need the notion of convex cone, which is defined in [41]. A 2D convex cone in the plane is represented by a pair of linearly independent 2D vectors $\vec{v}_1, \vec{v}_2$ bounding of the cone as illustrated in Figure 6.3. Then a vector $\vec{v}$ belongs to the 2D convex cone if there is a pair of scalars $s_1, s_2 \in \mathbb{R}_+^*$ [1] such that $\vec{v} = s_1\vec{v}_1 + s_2\vec{v}_2$. Similarly an $n$D convex cone in the $nD$ space is represented by a set of $n$ linearly independent $n$D vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ bounding the cone as illustrated for the 3D case in Figure 6.5.

The following definition introduces the notion of density for a set of vectors in any dimensions. This definition is needed to prove that Pythagorean triples and Pythagorean $n$-tuples are dense.

**Definition 6.9.** Let $U$ be a set of $n$D vectors in $\mathbb{R}^n$. We say that $U$ is dense in $\mathbb{R}^n$ if and only if for the $n$D convex cone $\mathcal{C}$ defined by any set of $n$ linearly independent vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \in \vec{R}^n$, there is a vector $\vec{u} \in U$ that belongs to $\mathcal{C}$.

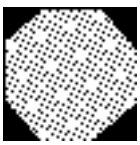
In other words, $U$ is dense if for any set of $n$ vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \in V$, there exists a vector $\vec{u} \in U$ and $s_1, s_2, \ldots, s_n \in \mathbb{R}_+^*$ such that $\vec{u} = s_1\vec{v}_1 + s_2\vec{v}_2 + \cdots + s_n\vec{v}_n$. With Definition 6.9, it is possible to show that $\mathcal{V}_{P2}$ is dense in $\mathbb{R}^2$; however, this proof is omitted since it is similar to the proof of Theorem 6.5.

**Theorem 6.10.** *The set of 2D Pythagorean vectors $\mathcal{V}_{\mathcal{P}_2}$ is dense in $\mathbb{R}^2$.*

## 6.3 Pythagorean quadruples

In Section 6.1, we explained that Pythagorean quadruples can be considered as 3D rational unit vectors. 3D unit vectors appear in many applications in computer vision and computer graphics. Here, we highlight Pythagorean quadruple. For the practical use, those vectors associated with Pythagorean quadruples must be dense on the unit

[1] $\mathbb{R}_+^*$ is equivalent to $]0, +\infty[$.



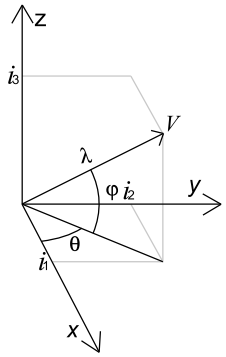



FIGURE 6.4: Geometrical representations of a Pythagorean quadruple $(i_1, i_2, i_3, \lambda)$ by the Pythagorean vector $(i_1, i_2, i_3)^\top$ and the pair of angles $(\theta, \psi)$.

sphere. In this section, we give a constructive proof that they are dense in $\mathbb{R}^3$ using Theorem 6.10.

**Definition 6.11.** A Pythagorean quadruple is an integer quadruple $(i_1, i_2, i_3, \lambda)$ such that $i_1^2 + i_2^2 + i_3^2 = \lambda^2$.

We define the set containing all Pythagorean quadruples such that $\mathcal{P}_3 = \{(i_1, i_2, i_3, \lambda) \in \mathbb{Z}^4 |\ i_1^2 + i_2^2 + i_3^2 = \lambda^2\}$. Similarly to the case of Pythagorean triples, we present two geometrical representations for each element of $\mathcal{P}_3$.

Pythagorean quadruples can be represented by angles, similarly to Pythagorean triples. Each Pythagorean quadruple can be associated with a pair of angles $(\theta, \psi)$ where $\cos\theta = \frac{i_1}{\sqrt{i_1^2+i_2^2}}, \sin\theta = \frac{i_2}{\sqrt{i_1^2+i_2^2}}$ and $\cos\psi = \frac{\sqrt{i_1^2+i_2^2}}{\lambda}, \sin\psi = \frac{i_3}{\lambda}$. Note that, with a Pythagorean triple, we can ensure that the values of the sine and cosine generated by the associated Pythagorean angle belong to $\mathbb{Q}$, and this is not the case with any Pythagorean quadruple because in most cases $\sqrt{i_1^2 + i_2^2}$ does not belong to $\mathbb{Z}$.

We can also represent a Pythagorean quadruple $(i_1, i_2, i_3, \lambda)$ by a vector $\vec{v} = (i_1, i_2, i_3)^\top$ in the 3D space as seen in Figure 6.4.

**Definition 6.12.** A 3D Pythagorean vector is a vector $\vec{v} = (i_1, i_2, i_3)^\top$ associated with the element $(i_1, i_2, i_3, \lambda)$ of $\mathcal{P}_3$.

As well as 2D Pythagorean vectors, each 3D Pythagorean vector has an integer Euclidean norm. We denote by $\mathcal{V}_{\mathcal{P}_3}$ the set of 3D Pythagorean vectors. Generally speaking, such geometrical representation for Pythagorean $n$-tuples is done in $(n-1)$D space.

Similarly to Pythagorean triples with their 2D geometric representation, the question of density for the representation of Pythagorean quadruples arises. In 2D the proof of the density was made by using Pythagorean angles. However, in 3D, as explained above, the representation by angles is not convenient because it uses irrational numbers. Therefore,

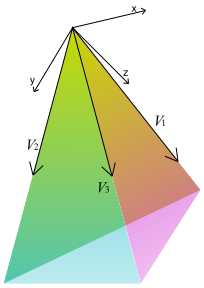


FIGURE 6.5: A 3D convex cone generated by three vectors $\vec{v}_1, \vec{v}_2, \vec{v}_3$ in the space

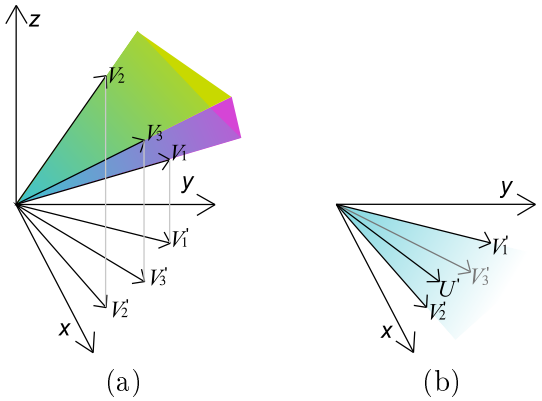



FIGURE 6.6: (a) The 3D convex cone $\mathcal{C}_3$ defined by $\vec{v}_1, \vec{v}_2, \vec{v}_3$ and their projected vectors $\vec{v}'_1, \vec{v}'_2, \vec{v}'_3$ on the OXY plane, and (b) the 2D convex cone $\mathcal{C}_2$ defined by $\{\vec{v}'_1, \vec{v}'_2\}$.

we will prove the density of 3D Pythagorean vectors. The definition of density used here is the one given in Definition 6.9.

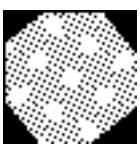
**Theorem 6.13.** *The set of 3D Pythagorean vectors $\mathcal{V}_{\mathcal{P}_3}$ is dense in $\mathbb{R}^3$.*

**Proof.** 11. To prove Theorem 6.13, it is sufficient to show that for every set of three vectors, we can construct a Pythagorean vector belonging to the 3D convex cone defined by the three vectors.

Let $\vec{v}_1, \vec{v}_2, \vec{v}_3$ be three linearly independent 3D vectors. We denote by $\mathcal{C}_3$ the 3D convex cone defined by position vectors $\vec{v}_1, \vec{v}_2, \vec{v}_3$. We also denote by $\vec{v}'_1, \vec{v}'_2, \vec{v}'_3$ the projections of $\vec{v}_1, \vec{v}_2, \vec{v}_3$ in the $OXY$ plane (Figure 6.6(a)).

Because $\vec{v}_1, \vec{v}_2, \vec{v}_3$ are linearly independents, at least one of these three pairs $\{\vec{v}'_1, \vec{v}'_2\}, \{\vec{v}'_1, \vec{v}'_3\}, \{\vec{v}'_2, \vec{v}'_3\}$ is linearly independent in the plane $OXY$. Let us assume that the pair of vectors $\{\vec{v}'_1, \vec{v}'_2\}$ is linearly independent. Thus $\{\vec{v}'_1, \vec{v}'_2\}$ defines in the $OXY$ plane a 2D convex cone, which we denote by $\mathcal{C}_2$ as shown in Figure 6.6(b). Thanks to Theorem 6.10, there is a 2D Pythagorean vector $\vec{u}'$ that belongs to $\mathcal{C}_2$ (Figure 6.6(b)). $\vec{u}'$ is associated with the Pythagorean triple $(i_1, i_2, \lambda_{\vec{u}})$. We set the vector $\vec{u} = (i_1, i_2, 0)^\top$.

Let $\mathscr{P}$ be a plane defined by the two vectors $\vec{u}$ and $(0, 0, 1)^\top$ and the point $(0, 0, 0)^\top$ (Figure 6.7(a)). By construction, $\mathscr{P}$ intersects $\mathcal{C}_3$. This intersection is a 2D convex



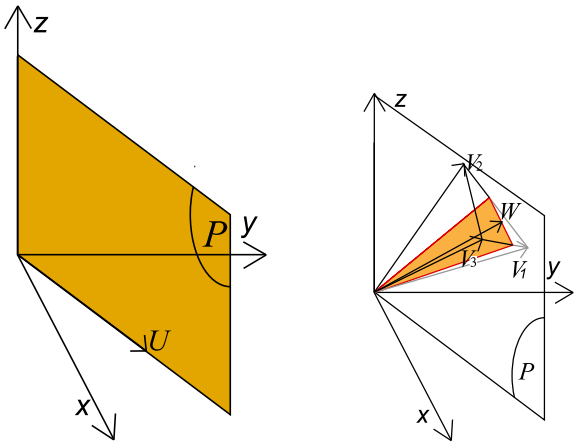



FIGURE 6.7: (a) The plane $\mathcal{P}$ defined by $OZ$ and $\vec{u}$, and (b) the vector $\vec{w}$ that belongs to $\mathcal{C}_2'$.

cone $\mathcal{C}_2'$ that belongs to $\mathscr{P}$ (Figure 6.7(b)). Thanks to Theorem 6.10 again, there is a 2D Pythagorean vector $\vec{w}'$ that belongs to $\mathcal{C}_2'$ (Figure 6.7(b)); $\vec{w}'$ is associated with the Pythagorean triple $(j_1, j_2, \lambda_j)$.

As said in Remark 6.3, if we multiply all integers of a Pythagorean triple by $k \in \mathbb{Z}$, the obtained triple is still a Pythagorean triple. This enables us to design from two Pythagorean triples a Pythagorean quadruple as follows.

Let $a, b \in \mathbb{Z}^*$ such that $a\lambda_{\vec{u}} = bj_1$. Then we obtain the Pythagorean quadruple $(ai_1, ai_2, bj_2, b\lambda_{\vec{w}})$. By construction, it is obvious that the vector $\vec{w} = (ai_1, ai_2, bj_2)^\top$, associated with the Pythagorean quadruple $(ai_1, ai_2, bj_2, b\lambda_{\vec{w}})$, belongs to the 3D convex cone $\mathcal{C}_3$.

## 6.4 Approximation of a given 3D vector by a Pythagorean 3D vector

In this section we explain how to approximate a given vector by a Pythagorean vector. Because we proved in Section 6.3 that 3D Pythagorean vectors are dense, we know that we find a Pythagorean vector as close as possible to the given vector considering the distance $d$ between the two vectors $\vec{u} = (u_x, u_y, u_z)^\top, \vec{v} = (v_x, v_y, v_z)^\top$ as follow $d(\vec{v}, \vec{u}) = \sqrt{(\frac{u_x}{\|\vec{u}\|} - \frac{v_x}{\|\vec{v}\|})^2 + (\frac{u_y}{\|\vec{u}\|} - \frac{v_y}{\|\vec{v}\|})^2 + (\frac{u_z}{\|\vec{u}\|} - \frac{v_z}{\|\vec{v}\|})^2}$. Two vectors are said to be $\vec{v}, \vec{u}$ are $\epsilon$-close if $d(\vec{v}, \vec{u}) < \epsilon$.

In this section, we first explain how to generate all Pythagorean quadruple, then we give two methods to approximate a given vector by an $\epsilon$-close Pythagorean vector. Finally, we show some experiment on approximation using both methods.

### 6.4.1 Generation of Pythagorean quadruples

In this section we explain how to obtain all Pythagorean quadruples using twelve matrices. This method is the extension to the Pythagorean quadruples of the method used for Pythagorean triples and presented in many papers [34, 36, 40]. Here we will first describe the method used to generate Pythagorean triples, then we will extend it to Pythagorean quadruple and show what problems appear during the generation of the tree.

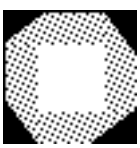
Note that in this section we only consider Pythagorean triples (respectively Pythagorean quadruples) of the form $< i_1, i_2, \lambda >$ (respectively $< i_1, i_2, i_3, \lambda >$) where $0 < i_1 \leq i_2 \leq \lambda$ (respectively $0 \leq i_1 \leq i_2 \leq i_3 \leq \lambda$). This assumption does not restrict the Pythagorean triples (respectively quadruples) generated, since all other Pythagorean triples (quadruples) can be obtained by switching two values or multiplying some values by $-1$.

It is generally admitted that the smallest Pythagorean triple is $< 3, 4, 5 >$. We say that the Pythagorean triple $< i_{11}, i_{12}, \lambda_1 >$ is smaller than the Pythagorean triple $< i_{21}, i_{22}, \lambda_2 >$ if $\lambda_1 < \lambda_2$ or if $\lambda_1 = \lambda_2$ and $i_{12} < i_{22}$. With this order, we can also consider the triple $< 0, 1, 1 >$ as the smallest Pythagorean triple, but this Pythagorean triple is degenerated since it contains a 0.

The method to generate the tree that contains all Pythagorean triples uses the three following matrices:

$$\mathcal{M}_1 = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \end{pmatrix}, \mathcal{M}_2 = \begin{pmatrix} -1 & 2 & 2 \\ -2 & 1 & 2 \\ -2 & 2 & 3 \end{pmatrix}, \mathcal{M}_3 = \begin{pmatrix} 1 & -2 & 2 \\ 2 & -1 & 2 \\ 2 & -2 & 3 \end{pmatrix}. \tag{6.1}$$

Applying these three matrices to the first Pythagorean triple $< 3, 4, 5 >$ will give the second level of the tree, then applying these three matrices to the three children of $< 3, 4, 5 >$ will give nine children that are the third level of the tree. Iterating this operation will give us all possible prime Pythagorean triples, the three first levels of the Pythagorean tree are given in example in Figure 6.8. Note that in [4] a different Pythagorean tree is presented, in his tree the author sorts the Pythagorean triple according to their angles.



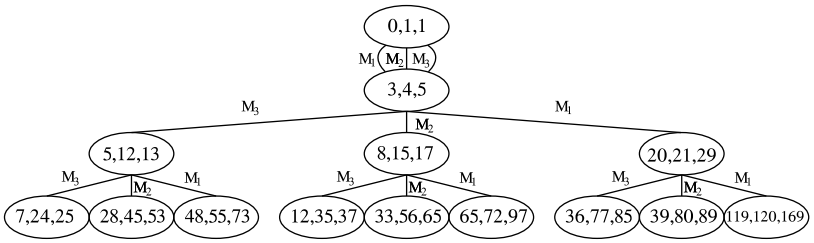



FIGURE 6.8: The three first level of the Pythagorean tree.

An important property of the construction is that there is a proof that each Pythagorean triple in the tree is a prime Pythagorean triple and each Pythagorean triple appears only in the tree.

Note that there is also a fourth matrix:

$$\mathcal{M}_4 = \begin{pmatrix} -1 & -2 & 2 \\ -2 & -1 & 2 \\ -2 & -2 & 3 \end{pmatrix} \tag{6.2}$$

that gives the father of a Pythagorean triple in the tree.

We expect that there is a Pythagorean tree for Pythagorean quadruples that has the same properties than the tree of a Pythagorean triple, however, we will see that this is not the case.

Now we consider Pythagorean quadruples. The naive adaptation of the above method to obtain the Pythagorean quadruple tree is to add a dimension to matrices 6.1 and to apply it on all quadruples $(i_1, i_2, i_3, \lambda)$, $(i_2, i_1, i_3, \lambda)$ and $(i_3, i_2, i_1, \lambda)$ to obtain all sons. As the smallest Pythagorean quadruple, there are two candidates $< 0, 0, 1, 1 >$ and $< 1, 2, 2, 3 >$. The Pythagorean quadruple $< 0, 0, 1, 1 >$ can be considered as degenerated, thus the naive solution is to consider the smallest Pythagorean quadruple as $< 1, 2, 2, 3 >$. However, in the Pythagorean triples case, applying the three matrices 6.1 to $< 0, 1, 1 >$ always gives $< 3, 4, 5 >$, in the Pythagorean quadruple case, application of Matrix 6.3 on $< 0, 0, 1, 1 >$ will give two different Pythagorean quadruples: $< 1, 2, 2, 3 >$ and $< 0, 3, 4, 5 >$. Even if $< 0, 3, 4, 5 >$ is degenerated, some of its sons are not, thus we have to consider it to create the Pythagorean quadruple tree.

The extension of Matrix 6.1 to the 4D gives us the following matrix:

$$\mathcal{M}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 \\ 0 & 2 & 1 & 2 \\ 0 & 2 & 2 & 3 \end{pmatrix}, \mathcal{M}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 2 & 2 \\ 0 & -2 & 1 & 2 \\ 0 & -2 & 2 & 3 \end{pmatrix}, \mathcal{M}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 2 \\ 0 & 2 & -1 & 2 \\ 0 & 2 & -2 & 3 \end{pmatrix}. \tag{6.3}$$

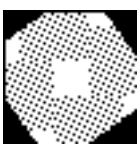
In the Pythagorean triples case, we just have to apply each of these matrices once to obtain all possible sons. However, to obtain all sons of a Pythagorean triple, it is necessary to apply Matrix 6.3 and then to switch $i_1$ and $i_2$ and apply again Matrix 6.3. We repeat this operation switching $i_1$ and $i_3$. The switch operation is needed because Matrix 6.3 does not modify the first value of the Pythagorean quadruple. Therefore the switching operation allows us to obtain nine sons.

When we generate the Pythagorean quadruples tree using the naive method, some problems arise. As explained before, we should consider $< 0, 0, 1, 1 >$ as the smallest Pythagorean quadruple even if its a degenerated one. However, considering this quadruple as the root of the tree will remove the guarantee that each Pythagorean quadruple only appears in the tree. Applying the Matrix 6.3 on this quadruple will only give two different sons, that will lead to repetitions. Pythagorean quadruple of the form $< i_1, i_2, i_2, \lambda >$ like $< 1, 2, 2, 3 >$ and $< 4, 4, 7, 9 >$ are also problematic since they have many repetitions in their sons. The solution may be to remove all repetitions after generating the sons of a Pythagorean quadruple. However, this does not solve the problem since some Pythagorean quadruples can be generated by two distinct Pythagorean quadruples. For example, the two Pythagorean quadruples $< 2, 3, 6, 7 >$ and $< 1, 4, 8, 9 >$ generate the Pythagorean quadruple $< 3, 4, 12, 13 >$ when we apply $\mathcal{M}_3$. One other problem of the naive method is that it does not generate all possible Pythagorean quadruples. There is an infinity of Pythagorean quadruples that cannot be reached using this method. The solution of this problem is to use the fourth matrix

$$\mathcal{M}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & -2 & 2 \\ 0 & -2 & -1 & 2 \\ 0 & -2 & -2 & 3 \end{pmatrix}, \tag{6.4}$$

that is the extension of the matrix that gives the father for a Pythagorean triple. An example of a Pythagorean quadruple that requires $\mathcal{M}_4$ to be reached is illustrated on Figure 6.9. The Pythagorean quadruple $< 23, 24, 24, 41 >$ is reached using $\mathcal{M}_4$ on Pythagorean quadruples $< 10, 10, 23, 27 >$ or $< 11, 12, 24, 29 >$. Using $\mathcal{M}_4$ on $< 23, 24, 24, 41 >$ will give $< 10, 10, 23, 27 >$ or $< 11, 12, 24, 29 >$, then we deduce that the fathers of $< 23, 24, 24, 41 >$ are $< 10, 10, 23, 27 >$ or $< 11, 12, 24, 29 >$ and that we cannot reach it using matrix $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$.

Even if for most Pythagorean quadruples, $\mathcal{M}_4$ gives the father of the Pythagorean quadruple as illustrated on Figure 6.10, there is an infinity of Pythagorean quadruple that are their own grandfather. In other words, there is an infinity of Pythagorean quadruples such that the application of $\mathcal{M}_4$ gives another Pythagorean quadruple and



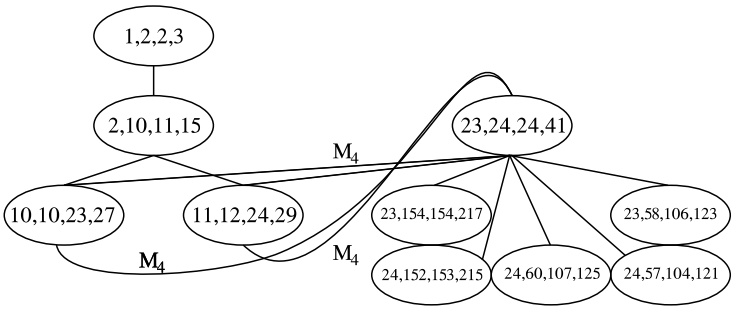



FIGURE 6.9: The Pythagorean quadruple $< 23, 24, 24, 41 >$ can only be reached using Matrix 6.4.

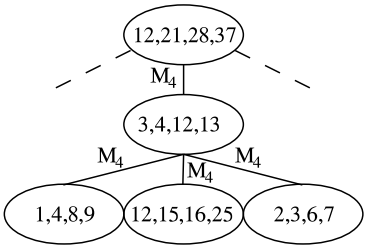



FIGURE 6.10: In most cases, $\mathcal{M}_4$ gives the father of a Pythagorean quadruple.

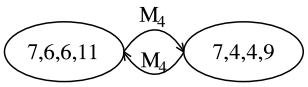



FIGURE 6.11: There exists an infinity of Pythagorean quadruples that have themselves as grandfather.

the application of $\mathcal{M}_4$ on the obtained Pythagorean quadruple will give the original one as illustrated on Figure 6.11.

From all these problems, we can derive a new method of generation of the Pythagorean tree that removes all repetitions. We set the root as $< 0, 0, 1, 1 >$ and use the four matrices $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$. The first level of the tree contains only the two Pythagorean quadruples: $< 0, 3, 4, 5 >$ and $< 1, 2, 2, 3 >$. To obtain the next level, we proceed as follow. For each Pythagorean quadruple of the last level, we compute the twelve sons that correspond to the three applications of the four matrices. For each son that results on the application of $\mathcal{M}_4$ we check if it is the same as its grandfather, if so we remove it. For the other sons, we remove all the repetitions.

### 6.4.2 Approximation methods

In this section, we present two methods to approximate a given vector $\vec{v} = (v_x, v_y, v_z)^\top$ by an $\epsilon$-close Pythagorean vector. In few words, the first method constructs the Pythagorean tree level by level and checks in each level if there is a Pythagorean vector that is $\epsilon$-close to the given vector. The second one selects for each level, the three Pythagorean

quadruples that are the closest one to the given vector and compute only their sons. Roughly speaking, the first method is a force brute algorithm. The second one cuts the Pythagorean tree. We know that the first one will give us a result since Pythagorean vectors are dense and it tries all possible Pythagorean quadruple until it finds an $\epsilon$-close Pythagorean vector. For the second one, we still do not have any proof of its convergence.

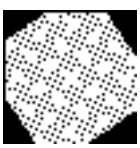
The first approximation method starts from the Pythagorean quadruple $< 0, 0, 1, 1 >$ and its two sons $< 0, 3, 4, 5 >$ and $< 1, 2, 2, 3 >$. For this three Pythagorean quadruples, we compute the distance between $\vec{v}$ and these three Pythagorean vectors. If none of them is $\epsilon$-close, we compute the sons of $< 0, 3, 4, 5 >$ and $< 1, 2, 2, 3 >$ as explained in Section 6.4.1 and for each sons, we compute the distance between them and $\vec{v}$. If none of them is $\epsilon$-close, we iterate the operation until we find a matching vector. Since Pythagorean vectors are dense, we know that this algorithm will end in a finite time, however, we cannot give any upper bound of the number of level to compute, then we cannot give any upper bound of the computation time. The problem of this method is the number of Pythagorean quadruple to compute and to compare with $\vec{v}$. As explained in Section 6.4.1, most of the Pythagorean quadruples have nine sons, thus the number of Pythagorean quadruples for a Pythagorean tree with $n$ levels can be approximated by $\sum_{i=1}^{n} 9^i$ considering that $< 0, 3, 4, 5 >$ and $< 1, 2, 2, 3 >$ are the level 0 of the tree. Thus reaching the 10th level of the tree requires about 3 billions comparisons.

The second approximation method cuts the Pythagorean tree. The first step is the same as the first method. After computing the first level of the Pythagorean tree, we compute the distance between $\vec{v}$ and all computed sons and check if one of the son is $\epsilon$-close to $\vec{v}$. If not, we sort the sons regarding their distance to $\vec{v}$ and search a convex cone formed by three of the sons. Because there are many convex cones, we consider the one that minimize the sum of the distances between the three sons and $\vec{v}$. To decide if the convex cone formed by three vectors contains $\vec{v}$, we compute the three planes formed by two of these three vectors and the origin and check if $\vec{v}$ and the vector that does not belong to the plane belongs to the same half-space. The next step is to compute the sons of the three vectors that form the convex cone and compute the distance between $\vec{v}$ and them. If none of them is $\epsilon$-close to $\sqsubseteq$, we compute a new convex cone and iterate the last operation until we found and $\epsilon$-close Pythagorean vector. For each iteration we compute a maximum of 36 Pythagorean quadruples (12 for each vector of the convex cone), then the number of Pythagorean quadruples computed may be seriously reduced in comparison with the first method. However, unlike the first method, we cannot ensure that this method will finish since we do not have any proof on the repartition of the Pythagorean quadruple depending on the father.

The next section presents some experiments of the two methods presented here.

### 6.4.3 Experiments

In this section, we present some experiment on the two methods presented in Section 6.4.2. In order to evaluate our method, we generate for both methods $n$ vectors to approximate. For the first method, we define a depth $d$ and compute the corresponding Pythagorean tree. Then, for each vector to approximate, we compute the distance between this vector and all Pythagorean vectors in the tree and keep the lowest distance. Then we compute the average distance for all generated vectors. For the second method, we define a maximum depth $d$, and compute incrementally the $d$ convex cones. We keep the closest Pythagorean vector computed.



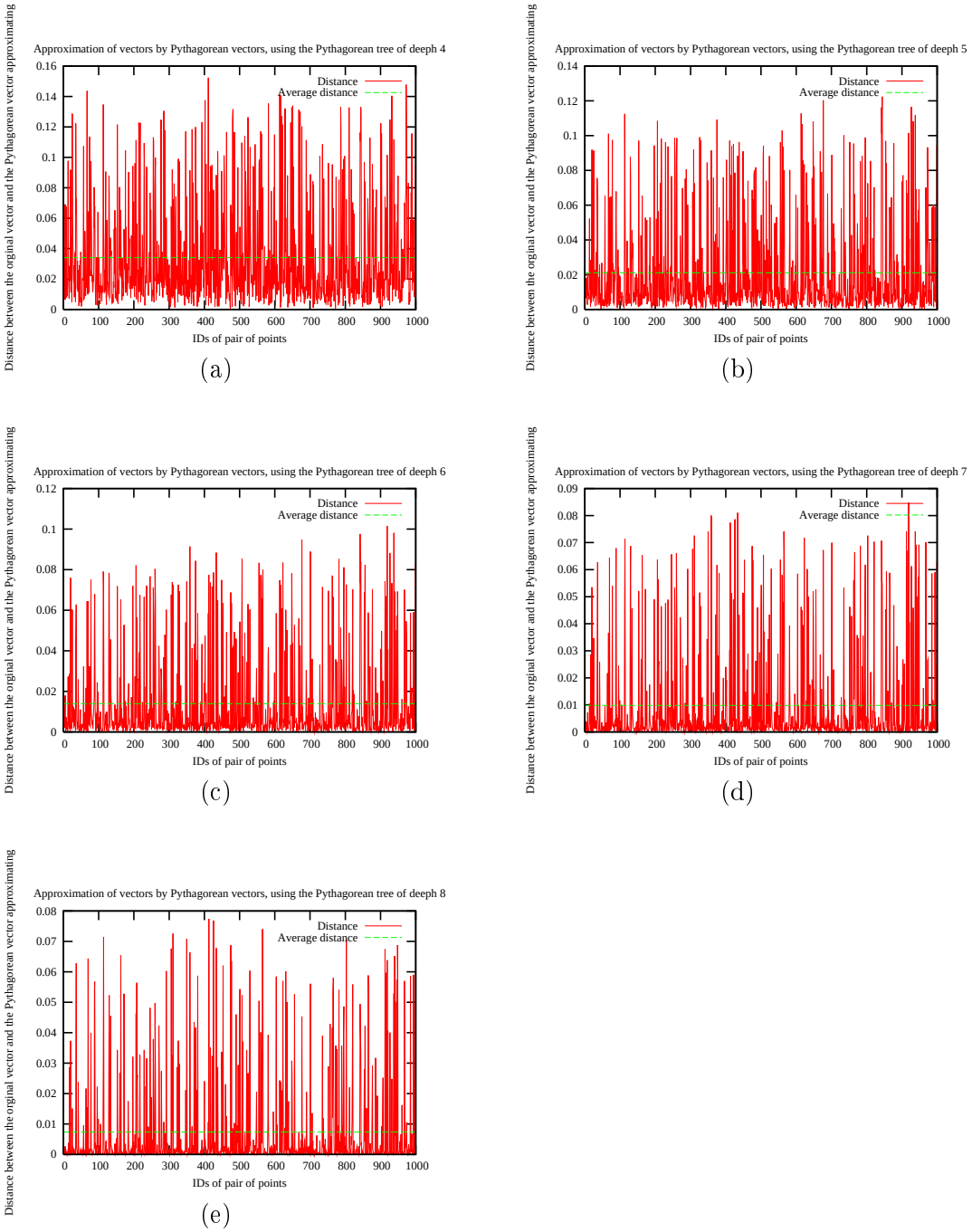
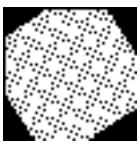


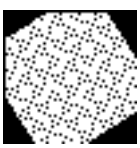

FIGURE 6.12: (a) Distance between 1000 vectors and their approximated Pythagorean vector for a Pythagorean tree of depth 4 (a), of depth 5 (b), of depth 6 (c), of depth 7 (d) and of depth 8 (e)



For the first method, we set $d = 4, 5, 6, 7, 8$ and generated 1000 vectors to approximate. These experiments are illustrated in Figure 6.12(a-e). The computation time depends on the depth $d$ of the Pythagorean three. When $d = 8$, it takes almost one hour to find the 1000 closest Pythagorean vectors or around three seconds by vector to approximate, but when $d = 4$, it takes around half a second to find the 1000 closest Pythagorean vectors or 0.5ms for each Pythagorean vector. The evolution of computation time between two levels of the tree have a factor 9. As we can see, the average distance between the original vector and its approximation decreases between depth 4 and depth 8 from 0.034 to 0.0077. However, the maximum distance is still high in both cases, around 0.15 when $d = 4$ and 0.073 when $d = 8$. Analyzing the results shows that all the vectors with a bad approximation have two small coordinates and one a lot greater. The five vectors with the worst approximation for $d = 8$ are $< 1628, 1706, 30389 >, < 179, 214, 3539 >, < 358, 437, 7611 >, < 547, 668, 11872 >, < 1057, 1106, 21398 >$ for error between 0.07 and 0.075.

For the second method, we set $d = 10, 20, 50, 100, 200, 300$ and use the same 1000 vectors to approximate than the first method. These experiments are illustrated in Figure 6.13(a-f). For all the experiments, computations are fast, approximately 5ms when $d = 10$ to 0.15s when $d = 300$ for each Pythagorean vector. The computation time is linear regarding to the depth of the tree. The average distance between the original vector and its approximation evolves from 0.046 when $d = 10$ to 0.042 when $d = 300$. These average distances are not really improved between $d = 10$ and $d = 300$. The average distance obtained by the second method is always worse than the average distance obtained by the first method for all depth of the tree. However, even if the average distance is bad, many vectors are well approximated. The analysis of the result shows that many vectors that are not correctly approximated by the first method when $d = 4, 5$ obtain a better approximation using the second method for $d = 10$.

The second method cannot be used without the first one since many Pythagorean vectors are not well approximated. However, it is interesting to use the second method in parallel of the first one when $d$ is high since the first method has a high computation time and the second method can give a good result.



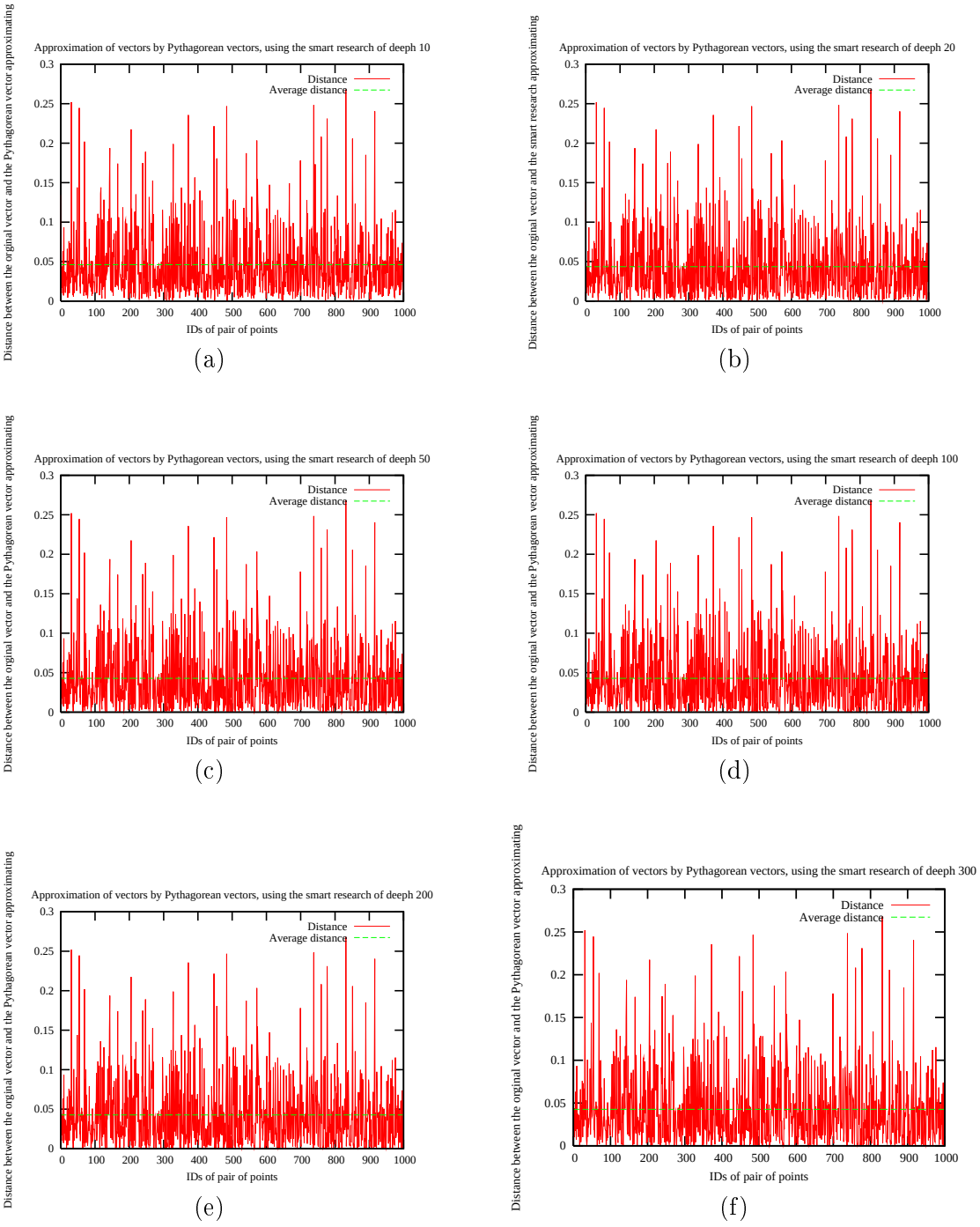
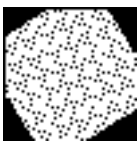



FIGURE 6.13: Distance between 1000 vectors and their approximated Pythagorean vector for a partial Pythagorean tree of depth 10 (a), of depth 20 (b), of depth 50 (c), of depth 100 (d), of depth 200 (e) and of depth 300 (f)



The second method leads to an interesting observation on the repartition on Pythagorean quadruples. Even when we know the father, we cannot predict the direction of the sons. Then the study of the repartition of the Pythagorean quadruples on the unit sphere will be difficult. Therefore, from this observation, we can design a third method. The third method is almost the same than the second one; the main difference is that instead of keeping the three Pythagorean vectors that form a convex cone around the vector to approximate, we select three vectors randomly among the sons computed from the three previous vectors.

For the third method, we set $d = 10, 20, 50, 100, 200, 300$ and use the same 1000 vectors to approximate than the first method. These experiments are illustrated in Figure 6.14(a-f) . The computation times to approximate each vector are almost the same than the computation times obtained with the second method. However, the results are a lot better when $d$ gets high. The obtained result when $d = 300$ is really close to the result obtained by the first method when $d = 6$ since the maximum distance between the vector to approximate and its approximated Pythagorean vector is lower than 0.1 and the average distance is 0.015 for the third method and 0.014 for the first method.

The third method gives a good result; however, this is a theoretical method since the integers which form the Pythagorean quadruples when $d = 300$ are huge ($\approx 10^{40}$) and need a special library to be coded in all languages.

From our experiments, we can conclude that the only practical method is the first one.

## 6.5 Pythagorean $n$-tuples

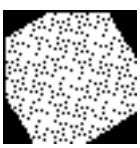
In this section, we first present Pythagorean $n$-tuples and their representation in $n-1$ dimensions. Then, we extend Theorem 6.13 to any dimensions. The following definition is the extension of Definitions 6.1 and 6.11.

**Definition 6.14.** A Pythagorean $n$-tuple is an integer $n$-tuple $(i_1, i_2, \ldots, i_{n-1}, \lambda)$, such that $i_1^2 + i_2^2 + \cdots + i_{n-1}^2 = \lambda^2$.

In order to simplify the reading, hereafter, $n$ denotes the dimension of their representation, instead of the numbers of integers of the $n$-tuples. In other words, we will use notations $n$D Pythagorean vectors and Pythagorean $(n+1)$-tuples.

Similarly to the cases of $n = 2, 3$ in Sections 6.2 and 6.3, we define $\mathcal{P}_n = \{(i_1, i_2, \ldots, i_n, \lambda) \in \mathbb{Z}^{n+1} |\ i_1^2 + i_2^2 + \cdots + i_n^2 = \lambda^2\}$ which is the set of Pythagorean $(n+1)$-tuples.

We represent each element of $\mathcal{P}_n$ by an $n$D Pythagorean vector defined as follows.



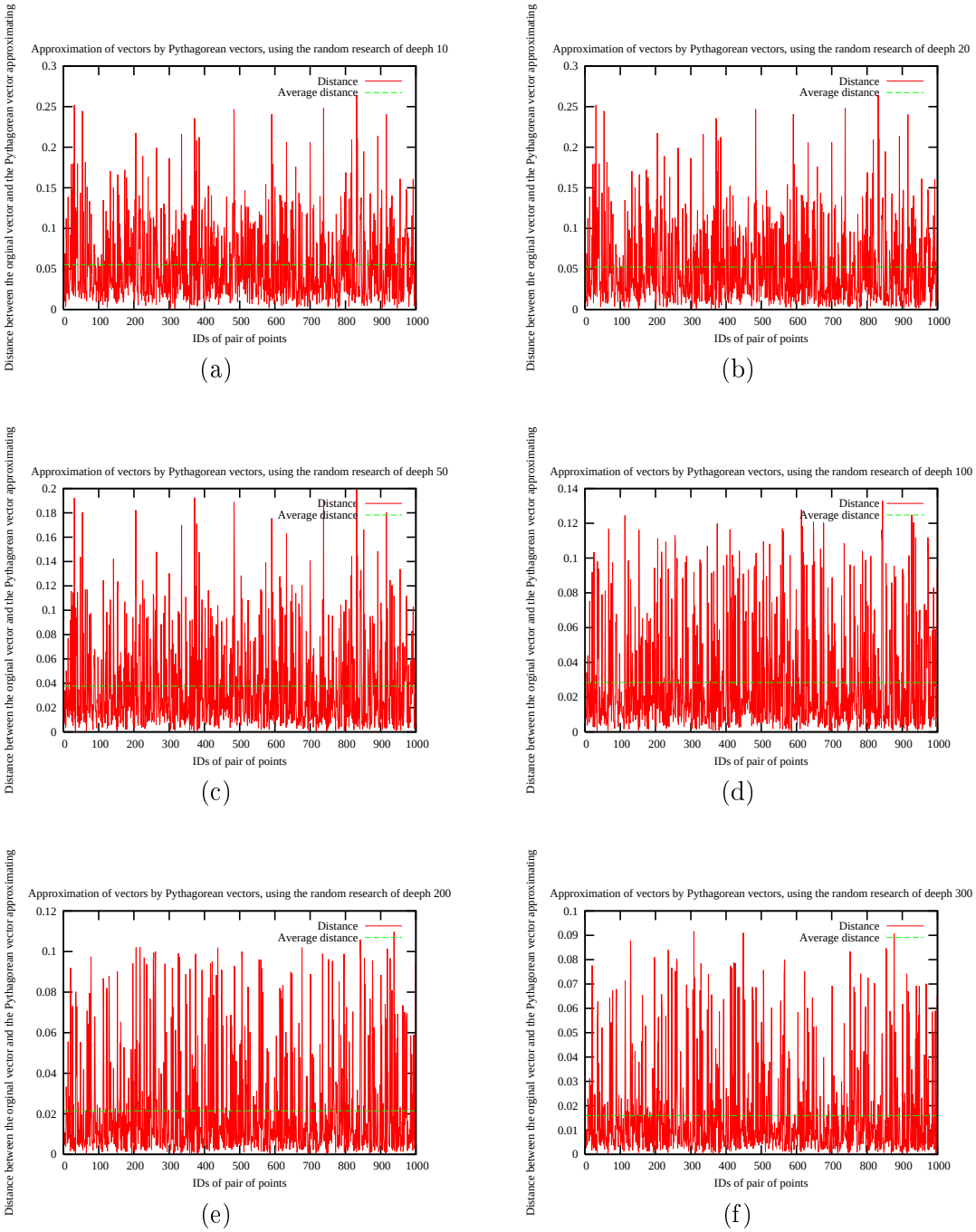
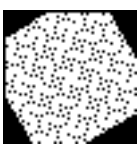



FIGURE 6.14: Distance between 1000 vectors and their approximated Pythagorean vector for a random Pythagorean tree of depth 10 (a), of depth 20 (b), of depth 50 (c), of depth 100 (d), of depth 200 (e) and of depth 300 (f)



**Definition 6.15.** An $n$D Pythagorean vector is a vector $\vec{v} = (i_1, i_2, \ldots, i_n)^\top$ associated with the element $(i_1, i_2, \ldots, i_n, \lambda)$ of $\mathcal{P}_n$.

The Euclidean norm of an $n$D Pythagorean vector is an integer. We denote by $\mathcal{V}_{\mathcal{P}_n}$ the set of $n$D Pythagorean vectors and we are going to prove the density of this set. The following theorem is the generalization of Theorems 6.10 and 6.13.

**Theorem 6.16.** *The set of nD Pythagorean vectors $\mathcal{V}_{\mathcal{P}_n}$ is dense in $\mathbb{R}^n$.*

Since we already proved the case of $n = 2, 3$ in Theorems 6.10 and 6.13, we give the proof for $n \geq 4$ using the induction. The proof for $n \geq 4$ is similar to the proof for $n = 3$. The main idea is to construct an $n$D Pythagorean vector in a given $n$D convex cone.

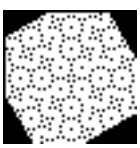
**Proof.** 12. First we assume that the set $\mathcal{V}_{\mathcal{P}_{n-1}}$ is dense in $\mathbb{R}^{n-1}$.

Let $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ be $n$ linearly independent $n$D vectors. We denote by $\mathcal{C}_n$ the $n$D convex cone defined by position vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$.

Let $\vec{v}'_1, \vec{v}'_2, \ldots, \vec{v}'_n$ be the projections of $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ on the hyper-plane $\mathcal{H}$ orthogonal to the $n$-th axis, respectively. Because $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ are linearly independent in $n$D, there is $j \in \{1, \ldots, n\}$ such that $\vec{v}'_1, \vec{v}'_2, \ldots, \vec{v}'_{j-1}, \vec{v}'_{j+1}, \ldots, \vec{v}'_n$ are linearly independent in $\mathcal{H}$. Thus, in $\mathcal{H}$, the set of vectors $\vec{v}'_1, \vec{v}'_2, \ldots, \vec{v}'_{j-1}, \vec{v}'_{j+1}, \ldots, \vec{v}'_n$ defines an $(n-1)$D convex cone, which we denote by $\mathcal{C}_{n-1}$. Because the set $\mathcal{V}_{Pn-1}$ is dense in $(n-1)$D, there is an $(n-1)$D Pythagorean vector $\vec{u}'$ that belongs to $\mathcal{C}_{n-1}$ such that $\vec{u}'$ is associated with the Pythagorean $n$-tuple $(i_1, i_2, \ldots, i_{n-1}, \lambda_{\vec{u}})$. We denote by $\vec{u}$ the $n$D vector $(i_1, i_2, \ldots, i_{n-1}, 0)^\top$.

Let $\mathcal{P}$ be a 2D plane defined by the two directional $n$D vectors: $(0, \ldots, 0, 1)^\top$ and $\vec{u}$ and the point $(0, \ldots, 0)^\top$. By the construction, $\mathcal{P}$ intersects $\mathcal{C}_n$. This intersection is a 2D convex cone, which we denote by $\mathcal{C}_2$ that belongs to $\mathcal{P}$. Thanks to Theorem 6.10, there is a 2D Pythagorean vector associated with a Pythagorean triple $(j_1, j_2, \lambda_{\vec{w}})$ that belongs to $\mathcal{C}_2$.

As said in Remark 6.3, if we multiply all values of a Pythagorean triple by $k \in \mathbb{Z}$, the obtained triple is still a Pythagorean triple. Note that Remark 6.3 is still valid for Pythagorean $k$-tuples even for $k \geq 3$. Therefore, it is possible to design from a Pythagorean triple and a Pythagorean $n$-tuple a Pythagorean $(n+1)$-tuple as follows. Let $a, b \in \mathbb{Z}^*$ such that $a\lambda_{\vec{u}} = bj_1$. Then we obtain the Pythagorean $(n+1)$-tuple $(ai_1, ai_2, \ldots, ai_{n-1}, bj_2, b\lambda_{\vec{w}})$. By construction the vector $\vec{u} = (ai_1, ai_2, \ldots, ai_{n-1}, bj_2)^\top$, associated with the Pythagorean $(n+1)$-tuple $(ai_1, ai_2, \ldots, ai_{n-1}, bj_2, b\lambda_{\vec{w}})$, belongs to the $n$D convex cone $\mathcal{C}_n$.



## 6.6 Approximation of an $n$D vector by an $n$D Pythagorean vector

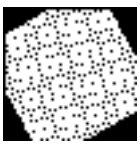
In this section, we explain how to approximate a given $n$D vector by an $n$D Pythagorean vector. Our method is based on the algorithm proposed by Anglin in [15] that allows us to approximate any angle $\alpha$ with a precision $\epsilon$ by a Pythagorean triple and its associated angle $\theta$. It uses the theorem, given in [32], that for each Pythagorean triple $(i_1, i_2, \lambda)$, there is a pair of integers $(u, v)$ such that $v < u$, $i_1 = 2uv$, $i_2 = u^2 - v^2$ and $\lambda = u^2 + v^2$. Setting $X = \tan(\alpha - \epsilon) + \sec(\alpha - \epsilon)$ and $Y = \tan(\alpha + \epsilon) + \sec(\alpha + \epsilon)$, Anglin proved that for every pair of integers $(u, v)$ such that $X < \frac{u}{v} < Y$, the angle $\theta$ associated with the Pythagorean triple generated by $(u, v)$ approximates $\alpha$ such that $\alpha - \epsilon < \theta < \alpha + \epsilon$.

Based on [15], we can derive a naive method to approximate any $n$D vector by an $n$D Pythagorean vector; its main idea, which is similar to the previous proof of density, is to construct $m$ 2D Pythagorean vectors to approximate an $n$D Pythagorean vector.

Let $\vec{v} = (r_1, r_2, \ldots, r_n)^\top$ be an $n$D real vector to be approximated by a Pythagorean vector that is $\epsilon$ close [2]. From $\vec{v}$, we define $(n-1)$ 2D vectors $\vec{v}_2, \vec{v}_3, \ldots, \vec{v}_n$ such that $\vec{v}_2 = (r_1, r_2)^\top, \vec{v}_3 = (\sqrt{r_1^2 + r_2^2}, r_3)^\top, \ldots, \vec{v}_n = (\sqrt{r_1^2 + r_2^2 + \cdots + r_{n-1}^2}, r_n)^\top$. Each vector $\vec{v}_i = (x_i, y_i)^\top$ is associated with an angle $\alpha_i$ satisfying $\cos\alpha_i = \frac{x_i}{|\vec{v}_i|}$. Using the algorithm presented in [15] for each of the $(n-1)$ angles $\alpha_i$ with a precision $\epsilon$, we obtain $(n-1)$ Pythagorean angles associated with the $(n-1)$ Pythagorean triples $(i_2, j_2, \lambda_2), (i_3, j_3, \lambda_3), \ldots, (i_n, j_n, \lambda_n)$. Now, we consider the Pythagorean vectors associated with the $n$ Pythagorean triples. Remark 6.3 allows us to generate from the two first Pythagorean triples $(i_2, j_2, \lambda_2), (i_3, j_3, \lambda_3)$, a Pythagorean quadruple $(k_1 i_2, k_1 j_2, l_1 j_3, l_1 \lambda_3)$ with two integers $k_1, l_1$ such that $k_1(i_2 + j_2) = l_1 i_3$. Similarly, from this Pythagorean quadruple and the third Pythagorean triple $(i_4, j_4, \lambda_4)$, we can generate the Pythagorean quintuple $(k_1 k_2 i_2, k_1 k_2 j_2, l_1 k_2 j_3, l_2 j_4, l_2 \lambda_4)$ with two integers $k_2, l_2$ such that $k_2(k_1 i_2 + k_1 j_2 + l_1 j_3) = l_2 i_4$. We repeat this combination operation between a Pythagorean $(i+2)$-tuple and a Pythagorean triple by adding a pair of integers $k_i$ and $l_i$ until obtaining a Pythagorean $(n+1)$-tuple. The $n$D vector associated with this Pythagorean $(n+1)$-tuple is our approximation result of $\vec{v}$.

In [15], the author introduced $\epsilon$ that corresponds to the maximum angle between the original vector and the Pythagorean vector that approximates it. With the above method, to construct an $n$D Pythagorean vector that approximates a given vector by $\epsilon$, we apply $(n-1)$ times Anglin's method. An example of the method that approximates a 3D vector is given in Figure 5.1. First we decompose the 3D vector $\vec{v}$ to approximate into the two 2D vectors $\vec{v}_2$ and $\vec{v}_3$. $\vec{v}_2$ and $\vec{v}_3$ can be approximated by Anglin's method with

[2] Two vectors are $\epsilon$ close if the angle between them is smaller than $\epsilon$.



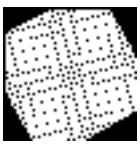
a precision of $\epsilon$ by two 2D Pythagorean vectors $\vec{v}'_2$ and $\vec{v}'_3$ that belong to the two 2D convex cone illustrated in Figure 5.1 (a) and (b) respectively. Thanks to Remark 6.3, we generate $\vec{v}'$ from $\vec{v}'_2$ and $\vec{v}'_3$, which is the approximation of $\vec{v}$, such that it belongs to the 3D convex cone represented in Figure 5.1 (c) forming a square pyramid. The minimum circular cone including the square pyramid has the solid angle $2\sqrt{2}\epsilon$. Therefore, we can deduce that to reach a precision of $\epsilon'$ while approximating a 3D vector, we need to give to Anglin's method a precision of $\epsilon \leq \frac{\epsilon'}{\sqrt{2}}$. In general, to reach a precision of $\epsilon'$ while approximating an $n$D vector, it is necessary to give to Anglin's method a precision of $\epsilon \leq \frac{\epsilon'}{\sqrt{n}}$ for each 2D vector to approximate.

The algorithm given in this section is fast because, according to Anglin, the computation of a Pythagorean triple is done in constant time. Accordingly, we can give the approximation of any $n$D vector in $O(n)$ operations. However, we cannot give any bounds on the size of integers that belong to the final Pythagorean $(n+1)$-tuple.

## 6.7 Conclusion

In this chapter, we introduce the Pythagorean $n$-tuple that are required for hinge angles in 3D. We gave a constructive proof of the density on 3D Pythagorean vector and then on $n$D Pythagorean vectors. This proof is required to show that for any given vector, we can construct a Pythagorean vector that approximates the given vector.

The algorithm presented in Section 6.6 does not give an upper bound for the size of integers composing the Pythagorean $n$-tuple. A future work is to give an upper bound. For reaching this goal, we have two directions. One is to use a method that computes all primitive Pythagorean $n$-tuples using a tree structure [36]. The other is to study the repartition of $n$D Pythagorean vectors in the $n$ dimensional space.



# Chapter 7

# Conclusion

As explained in the introduction, the main goal of this thesis was to improve the understanding of the rotations in the 2D and 3D discrete space and to develop new tools to compute discrete rotations using only integers. To conclude this manuscript, we resume the main result obtained during my thesis and introduce some future works and developments.

## 7.1 Discrete rotation in 2D

During my thesis, we studied the particularity of the 2D rotations in discrete space. In Chapter 2, we present a survey on 2D discrete rotations and the problems that arise during computations. We show that each existing discrete rotation solves some of the problems presented, but not all. We also present a study on rotations in the two other regular grids: triangle and hexagonal grid. We show that the rotations in hexagonal grid give better results than rotation in orthogonal grid regarding points lost and connectivity. However, even if the results obtained are better, the properties of bijectivity and transitivity are lost and the rotation is not discrete regarding our definition. Then one of the future work will be to develop a new discrete rotation based on hexagonal grid that improves the results on points lost and connectivity presented in Chapter 2. A possible method to obtain such results is to mix the results obtained with a rotation on hexagonal and orthogonal grids. Another possible method is to adapt the rotation by circle to the hexagonal grid.

Another property, which is not discussed in this manuscript, is the conservation of topology. To sum up, topology is the mathematical study of the properties that are preserved through deformations, twisting, and stretching of objects. A coffee cup and a donuts are

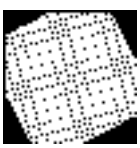
topologically equivalent since they both have one hole but they are not equivalent to a glass that has no hole. Obviously, Euclidean rotation keeps the topology of an object and discrete rotation does not keep it since it is required to keep connectivity and to not create holes while losing points. Another future work will be to design a discrete rotation that ensures that the topology of object is kept during discrete rotations.

In Chapter 3, we studied hinge angles and their applications to discrete rotations. Hinge angles represent the discontinuities of the rotations on discrete space. They allow the design of a discrete rotation having the same results than the discretized Euclidean rotation that is the Euclidean rotation applied on a discrete space. The 2D discrete rotation using hinge angles is neither bijective or transitive. A future work is to try to give these two properties to this rotation. A method based on multi-scale can give such results. A promising idea is to divide the pixel into many little pixels with different weight. The discretization will be done regarding the number of each little pixel in the normal pixel and their respective weight.

## 7.2 Discrete rotation in 3D

As we explained in the first section of Chapter 4, discrete rotations in 3D are not as well studied as discrete rotations in 2D. There was at our best knowledge, no rotation that is discrete under our definition. Then, in Chapter 4, we have designed a 3D discrete rotation based on hinge angles. For this rotation, we have extended the hinge angles in 3D, defined the multi-grids that are the intersection between the rotation plane and the 3D half-grid and present a study on Pythagorean quadruple. This discrete rotation gave the same results than the discretization of a 3D continuous rotation around a given axis. From this discrete rotation, we have extended the research of a rotation done between a pair of digital images in correspondence presented in Chapter 3 to the 3D discrete space. The extension done in Chapter 5 searches for an admissible axis of rotation and fix it in order to obtain admissible rotation angle regarding to this axis. However, another approach is to obtain all admissible rotation axis and to obtain rotation angles regarding to the rotation axis. To obtain such results, we have to compute the bound of all admissible rotation axis for the given $n$ pairs of points in correspondence. In 2D, such bounds have been computed for an admissible center of rotation [31]. The extension of this work to 3D is required to obtain the bound of the admissible rotation axis.

As explain in above section, a future work will be to design a discrete rotation that keep the topology. Such a rotation is also required for 3D space. However, in 3D a new artifact called tunnel, appear in 3D objects. Thus the simple extension of a 2D discrete rotation that keep topology to 3D may not keep topology in 3D.



In 2D we have presented a short study on rotation in different regular grids. In 3D, according to the Russian mathematician *Fyodorov*, the only regular polygon that tile the plane is the cube. However, some other semi-regulars polygons tile the plan such as truncated octahedron or rhombic dodecahedron. The shape of these two polygons is closer to the shape of a sphere than the shape of the cube. Therefore, study the rotation in 3D space tiled by one of these polygons may give interesting result.

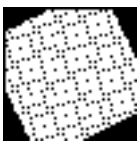
Another future work will be to design a 3D rotation that extends the rotation by discrete circles presented in Chapter 2. The rotation by discrete spheres will be a bijective 3D discrete rotation. A study is required to determine which definition of discrete spheres will give the best results. This definition requires at least to tile the 3D discrete space.

## 7.3 Discrete rotation in $n$D

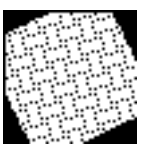
Commonly, the rotation is required in 2D or in 3D. However, for some particular fields, rotations in higher dimension can be required such as color image analysis where 4D rotations are needed or global visibility where 5D or 6D rotation are needed[42] [1]. In 2D, the rotations are computed in a unique plane. In 3D, to each rotation is associated a unique rotation axis and a unique plane, denoted by rotation plane in this thesis, to compute the rotation. It is important to notice that in 2D and 3D, two given rotations with different parameters (angle of rotation and/or axis of rotation) will give different result. When the dimension is higher than three, this is not necessarily true. Indeed, when the dimension $n$ is higher than three, the number of angles required to perform a rotation is $\lfloor \frac{n}{2} \rfloor$. Moreover, to perform a rotation in dimension $n$, we need to compute $\lfloor \frac{n}{2} \rfloor$ rotation planes that are associated to the $\lfloor \frac{n}{2} \rfloor$ angles. Two different sets of rotation planes and angles can define the same rotation. Another remark, which is strongly related to the previous one, is that in $n$ dimensions, a rotation with a unique angle will leave $(n-2)$ stationary dimensions. Then in a planar rotation we have a 0D stationary dimension, in other word a point, in 3D we have a stationary axis, in 4D we have a stationary plane and so on.

In $n$D some continuous method to perform rotation are known such as method using Clifford algebra. At our best knowledge, there exists no method to perform discrete rotation in $n$D. The hinge angles and multi-grids presented in this thesis can be extended to perform rotation in any dimension. However, some problems appear. Firstly, as explain above, a rotation in $n$D must be decomposed in $\lfloor \frac{n}{2} \rfloor$ planar rotation. In 3D, we know that this decomposition in unique, in $n$D the question is open. Another problem is the uniqueness of representation of hinge angle by $n$-tuple of integers. In 2D and 3D,

[1] Reference are required.



we have the proof that these representations are unique, but by carefully reading the proof, we see that for dimension higher than four, the uniqueness disappear. The work presented in [31] that must be extended to find admissible rotation axis in 3D must also be extended in $n$D to find the set of $\lfloor \frac{n}{2} \rfloor$ admissible plane of rotation or at least one set if the decomposition is not unique. We can conclude that the description of discrete rotation methods in $n$D will requires lot of work.



# Chapter 8

# Appendix

This last chapter is a recall of all definitions used in this manuscript. Here the definitions are not formal, just intuitive. It may help the reader to have a better understanding of the notions or find a definition that he forgot.

**4-connected:** Imagine that you are the tower on a chess-board and that you are wounded and can only move of one case. The four cases that you can reach are 4-connected with the one you stand on.

**6-connected:** Imagine one dice with six faces. Now glue six dices to the six faces of the first one. These six dices are 6-connected with the one in the middle.

**8-connected:** Imagine that you are the queen on a chess board and that you are wounded and can only move of one case. The height cases that you can reach are 8-connected with the one you stand on. In other words, kings are wounded queens.

**18-connected:** Imagine a Rubik's cube. Now, we remove the six corner of the cube, the ones with three colored stickers on each. The eighteen remaining little cubes that you can see are the 18-connected cubes of the hidden one (that does not really exists if you dismantle it).

**26-connected:** Imagine a Rubik's cube. The twenty-six little cubes that you can see are the 26-connected cubes of the hidden one (that does not really exists if you dismantle it).

**Admissible rotation angles:** Abbreviated as ARA, the admissible rotation angles are the set of all angles which give the same result by rotation ...

**Convex cone:** Imagine a square based pyramid. If you remove the base and put a powerful lamp, the light will describe an infinite square pyramid which can be seen as a convex cone.

**Convex hull:** Imagine a land containing a finite number of trees. You attach a rope to the most eastern tree, surround all the trees with the rope and attach again the rope to the same tree. The rope is the convex hull of the trees.

**Convexel:** Take a cube, slice it one time. The new surface obtained is a convexel. This surface will contain $3, 4, 5$ or $6$ edges depending on how you sliced the cube.

**Degenerated Pythagorean $n$-tuple:** If a Pythagorean $n$-tuple contains $m$ zero, then the $(n-m)$-tuple which is the Pythagorean $n$-tuple without the zero is a Pythagorean $(n-m)$-tuple.

**Discrete geometry:** It is a subfield of mathematics. Simply put, Discrete geometry is the part of mathematics considering only finite set. Instead of considering an infinity of numbers, we just select some of these numbers such as all the integers smaller than one billion and try to find the geometrical properties of this set.

**Discrete grid:** Can be seen as a computer screen. On a continuous plane, there is always an infinity of points between two points. On a discrete grid, there is a finite number of points between two points. If you consider two neighbor pixels on a computer screen, there is no pixel between them. This difference implies one of the main issue in computer science.

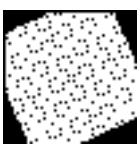
**Discrete point:** A point in any dimension with integer coordinates. On a computer screen, it can be considered as a pixel.

**Floating numbers:** In computer science, they are an approximation of the real numbers. In math, there are different sets of numbers, the two most used sets are natural numbers and real numbers. In computer science, it is possible to represent natural numbers but, since most of the real number have an infinity of digit after the point, they cannot be exactly represented in the memory of a computer. Thus, we use the floating numbers instead. The difference between a real number and its floating approximation necessarily implies errors during computations.

**Grandfather (tree):** In computer science, a tree look like a family-tree. The notion of grandfather is exactly the same.

**Half-grid:** In any dimension, the half-grids are the union of all hyper-plans of equations $x = i + \frac{1}{2}$ where $x$ represents one of the dimensions and $i \in \mathbb{Z}$. The half-grid, is the border between the hypercube of dimension $n$ which tiles the space. On a paper, the half-grid can be compared to the line on a graph paper.

**Multi-grid:** In any dimension, the multi-grids are the intersection between the half-grid and a given plane.



**Pixel:** Pixels can be seen in the discrete space as points in the continuous space. On a chess board, the pixel will be a black or white square, since a piece belongs to only one case, no matter where it is. On a computer screen, pixels are the smallest points on the screen.

**Points in correspondence:** Two points from two different digital images (in any dimension) are said to be in correspondence if they represent the same things in both digital images. For two digital images of the Eiffel tower, the two pixels which represent the top of the tower in both images are said to be in correspondence.

**Prime:** A set $S = (i_1, i_2, \ldots, i_n)$ is prime if $\gcd(i_1, i_2, \ldots, i_n) = 1$. In other words, a set of element is prime, if there is no element that divide each element of the set.

**Pythagorean angles:** An angle is Pythagorean if its sine and its cosine are rational. Each Pythagorean angle is associated to a Pythagorean triple. They are useful to compute rotations using only integers.

**Pythagorean $n$-tuples:** Read first Pythagorean triples. Instead of a right triangle in the plane, we imagine a right triangle in the $(n-1)$D space. We then add $n-3$ integers to the triple to obtain a $n$-tuples of integers $i_1, i_2, \ldots, i_{n-1}, \lambda$ such that $i_1^2 + i_2^2 + \cdots + i_{n-1}^2 = \lambda^2$.

**Pythagorean quadruples:** Read first Pythagorean triples. Instead of a right triangle in the plane, we imagine a right triangle in the 3D space. We then add one integer to the triple to obtain a quadruple of integers $i_1, i_2, i_3, \lambda$ such that $i_1^2 + i_2^2 + i_3^2 = \lambda^2$.

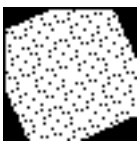
**Pythagorean triples:** Firstly, we have to remember our old mathematic classes with Pythagoras of Samos[35]. Hypotenuse and so on ... So if the three sides of a right triangle (Pythagorean triangle) have integer norms, then the three norms form a Pythagorean triple. In other words, each triple of integers $i_1, i_2, \lambda$ such that $i_1^2 + i_2^2 = \lambda^2$ is a Pythagorean triple.

**Pythagorean vectors:** Consider a Pythagorean triple, the associated Pythagorean vector is the vector with the coordinates $(i_1, i_2)^\top$ and with the norm $\lambda$.

**Rational computation:** Equivalent to integer computation since a rational number can be seen as a pair of integers.

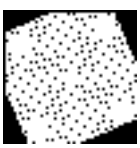
**Tree:** It is a structure mostly used in computer science in order to represents data. Like a real tree, it has a root and many branches denoted by nodes. A node belongs to only one branch but one branch can have many nodes.

**Topology:** It's the study of the spatial properties of objects under continuous deformations. Roughly speaking, if we consider an object done with modeling clay, we can scratch



it, extend it without modifying its topology. However if we cut it, create a hole, etc the topology is modified. A donut and a coffee cup are topologically equivalent since they both have only one hole. According to [43], topology is a useless branch of mathematics and computer science.

**Voxel:** Read first Pixel. Remember the Lego toys. Consider an object made only by using smallest colored cubes. Each cube in the object can be seen as a voxel. They are the smallest and indivisible parts of the object (except if you introduce a three year old child, but we do not consider bugs in this manuscript).



# Bibliography


[1] Eric Andres, Claudio Sibata, Raj Acharya, and Kyu Shin. New methods in oblique slice generation. In *SPIE Medical Imaging 96*, volume 2707 SPIE, pages 580–589, 1996.

[2] Eric Andres. *Cercles discrets et rotations discrètes.* PhD thesis, Université Louis Pasteur, Strasbourg, France, 1992.

[3] Benedek Nagy. An algorithm to find the number of the digitizations of discs with a fixed radius. *Electronic Notes in Discrete Mathematics*, 20:607–622, 2005.

[4] Bertrand Nouvel. *Rotations discrètes et automates cellulaires.* PhD thesis, Ecole Normale Supérieure de Lyon, France, 2006.

[5] Klauss Voss. *Discrete Images, Objects and Functions in* $\mathbb{Z}^n$. Springer-Verlag, 1993.

[6] Eric Andres. The quasi-shear rotation. In Serge. Miguet, Annick. Montanvert, and Stéphane Ubéda, editors, *Discrete Geometry for Computer Imagery*, volume 1176 of *LNCS*, pages 307–314. Springer Verlag, November 1996.

[7] M. Hénon and J.-M. Petit. Good rotations. *J. Comput. Phys.*, 146(1):420–435, 1998.

[8] Baoquan Chen and Arie Kaufman. 3d volume rotation using shear transformations. *Graphical Models*, 62:308–322, 2000.

[9] Jochen Schmidt and Heinrich Niemann. Using quaternions for parametrizing 3-d rotations in unconstrained nonlinear optimization. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 399–406. Aka GmbH, 2001.

[10] Tommaso Toffoli and Jason Quick. Three-dimensional rotations by three shears. *CVGIP: Graphical Model and Image Processing*, 59(2):89–95, 1997.

[11] Bernard Vitrac and Maurice Caveing. *Euclide : Les Eléments. Vol. 1 : Introduction générale et Livres I à IV.* Bibliothèque d'histoire des sciences. Presses Universitaires de France, Paris, 1990.

[12] Archimedes. *Measurement of a circle*. Eurêka edition, 230BC.

[13] Joseph Liouville. *A propos de l'existence des nombres transcendants*. Comptes-rendus de l'Académie des sciences, 1844.

[14] Carl Louis Ferdinand von Lindemann. *Über die Zahl* $\pi$, pages 213–225. Mathematische Annalen 20, 1882.

[15] W. S. Anglin. Using pythagorean triangles to approximate angles. *American Mathematical Monthly*, 95(6):540–541, 1988.

[16] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In *Applications of Computer Vision*, pages 54–64. IEEE Computer Society, 1994.

[17] Jack Bresenham. A linear algorithm for incremental digital display of circular arcs. *Commun. ACM*, 20(2):100–106, 1977.

[18] Marie-Andrée Jacob and Eric Andres. On discrete rotations. In *Discrete Geometry for Computer Imagery*, pages 161–174. Université de Clermont-Ferrand I, September 1995.

[19] A W Paeth. A fast algorithm for general raster rotation. In Andrew Glassner, editor, *Graphics Gems*, pages 179–195. Academic Press Professional, Inc., 1990.

[20] Jack. E. Volder. The cordic trigonometric computing technique. *IRE Transactions on Electronic Computers*, EC-8:330–334, 1959.

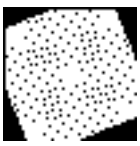
[21] Bertrand Nouvel and Eric Rémila. Incremental and transitive discrete rotations. In *Combinatorial Image Analysis*, volume 4040 of *LNCS*, pages 199–213. Springer-Verlag, 2006.

[22] Valentin E. Brimkov. Scaling of plane figures that assures faithful digitization. In V.E. Brimkov, R.P. Barneva, and H.A. Hauptman, editors, *Combinatorial Image Analysis*, volume 4953 of *LNCS*, pages 87–98. Springer Verlag, 2008.

[23] Martin N. Huxley and Jovisa D. Zunic. The number of $n$-point digital discs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):159–161, 2007.

[24] Son Pham. Digital circles with non-lattice point centers. *The Visual Computer*, 9 (1):1–24, 1992.

[25] Richard Hartley and Fredrik Kahl. Global optimization through rotation space search. *International Journal of Computer Vision*, 82(1):64–79, 2009.



[26] Amihood Amir, Ayelet Butman, Maxime Crochemore, Gad M. Landau, and Malka Schaps. Two-dimensional pattern matching with rotations. *Theor. Comput. Sci.*, 314(1-2):173–187, 2004.

[27] Yukiko Kenmochi and Atsushi Imiya. On combinatorial properties of discrete planar surfaces. In *Free Boundary Problems*, pages 255–272. Gakuto, 2000.

[28] Eric Schmutz. Rational points on the unit sphere. *Central European Journal of Mathematics*, 6(3):482–487, 2008.

[29] M. A. Kronrod. Optimal ordering algorithm without operational field. *Soviet Mathematics*, 10:744–746, 1969.

[30] Aurélie Richard, Laurent Fuchs, and Sylvain Charneau. An algorithm to decompose $n$-dimensional rotations into planar rotations. In *CompIMAGE*, pages 60–71, 2010.

[31] Marc Rodriguez, Adboulaye Sere, Gaelle Largeteau-Skapin, and Éric Andres. Generalized perpendicular bisector and circumcenter. In *Comp'IMAGE*, pages ???–??? ???, 2010.

[32] Ivan Niven, Herbert S. Zuckerman, and Hugh L. Montgomery. *An Introduction to the Theory of Numbers*. Wiley and Sons, 5th edition, 1991.

[33] Manfred Kühleitner. An omega theorem on differences of two squares. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 63(1):105–113, 1992.

[34] M. G. Teigen and D. W. Hadwin. On generating pythagorean triples. *The American Mathematical Monthly*, 78(4):378–379, 1971.

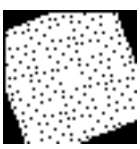
[35] Pythagoras of Samos. How to found a sect based on three numbers, 520BC. Source missing due to destruction of Alexandria's library.

[36] Daniel Cass and Pasquale J. Arpaia. Matrix generation of pythagorean n-tuples. In *American Mathematical Society*, volume 109, pages 1–7. Mathematical Association of America, 1990.

[37] Paul Oliverio. Self-generating pythagorean quadruples and $n$-tuples. *The Fibonacci Quarterly*, 34(2):98–100, 1996.

[38] E. Maor. *The Pythagorean theorem: a 4,000-year history*. Princeton University Press, 2007.

[39] James Glenn and James Robert C. *Mathematic dictionary*, chapter Density. Van Nostrand Reinhold Co, 1976.



[40] Ernest J. Eckert. The group of primitive pythagorean triangles. *Mathematics Magazine*, 57(1):22–26, 1984.

[41] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*, chapter 2.1.5, page 25. Cambridge University Press, 2004.

[42] Thomas Batard, Michel Berthier, and Christophe Saint-Jean. Clifford fourier transform for color image processing. In *Electronic Proceedings of AGACSE08*, pages Saint–Jean.pdf, 2008. 15 pages.

[43] John Chaussard. *Topologie ???!!!* PhD thesis, Univ Paris-Est, To be published ... soon.