Translations on the Triangular Grid

Khaled Hussein Abuhmaidan

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
May 2019
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Applied Mathematics and Computer Science.

Prof. Dr. Benedek Nagy
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev _____

2. Prof. Dr. Rza Bashirov _____

3. Prof. Dr. Gergely Kovács _____

4. Prof. Dr. Benedek Nagy _____

5. Prof. Dr. Cem Tezer _____

ABSTRACT

The concept of the grid is broadly used in digital geometry and other fields of computer science; it consists of discrete points with integer coordinates. Coordinate systems are essential for making grids easy to use. Up to now, for the triangular grid, only discrete coordinate systems have been investigated. These have limited capabilities for some image-processing applications, including transformations like rotations or interpolation. In this thesis, we introduce the continuous triangular coordinate system as an extension of the discrete triangular and hexagonal coordinate systems. The new system addresses each point of the plane with a coordinate triplet. Conversion between the Cartesian coordinate system and the new system is described. The sum of three coordinate values lies in the closed interval [-1, 1], which gives many other vital properties of this coordinate system. Moreover, addition of two vectors in the new triangular coordinate system is presented and illustrated.

Accordingly, in discrete and digital geometry, rotations with the composition of translations have been measured and examined carefully on the square and the hexagonal grids. The translation has never been considered individually because it obviously leads to the isometric translation on these grids. However, the triangular grid is not a point lattice, thus, it is worth to consider the translation itself. Therefore in this thesis, translations on the triangular grid are investigated and the vectors of bijective and non-bijective translations are specified.

Keywords: Barycentric coordinate system, coordinate system, hexagonal grid, triangular grid, trihexagonal grid, non-traditional grids, transformations, image processing, computer graphics, discretized translations, digital geometry

ÖZ

Grid kavramı koordinatları tamsayı olan ayrık noktalardan oluşur ve genellikle dijital geometri ve bilgisayar bilimlerinin diğer alanlarında kullanılır. Koordinat sistemleri gridlerin kolayca kullanımı için gereklidir. Şu ana kadar üçgensel grid için sadece ayrık koordinat sistemleri incelenmiştir. Bu sistemler rotasyon, interpolasyon gibi dönüşümleri içeren bazı görüntü işleme uygulamaları için sınırlı kapasiteye sahiptir. Bu tezde sürekli üçgensel koordinat sistemlerini ayrık üçgensel ve hegzagonal koordinat sistemlerinin bir genişlemesi olarak tanıtıldı. Yeni sistem düzlemdeki koordinat üçlüsü ile birlikte her noktayı adres eder. Kartezyen koordinat sistemi ve yeni sistem arasındaki dönüşüm tanımlandı. Üç koordinat değeri toplamının [-1, 1] aralığında olması bu koordinat sistemine birçok önemli özellik katar. Ayrıca iki vektörün toplanması işlemi yeni üçgensel koordinat sistemde gösterilmiş ve örneklendirilmiştir**.**

Buna göre, ayrık ve dijital geometride bileşke dönüşümleri ile birlikte rotasyonlar ölçülmüş ve kare ve hegzagonal gridlerde dikkatli bir şekilde incelenmiştir. Öteleme tek başına hiçbir zaman düşünülmemiştir çünkü bu gridlerde aşikar bir şekilde izometriği verir. Buna rağmen, üçgensel grid bir kafes noktası değildir, dolayısıyla öteleme kendi başına değerlendirilmeye değerdir. Bu sebepten, bu tezde öteleme üçgensel gridlerde incelenmiş ve bijektif ve bijektif olmayan dönüşümlerin vektörleri belirlenmiştir.

Anahtar kelimerler: Barisentrik koordinat sistemi, koordinat sistemi, hegzagonal grid, üçgensel grid, trihegzagonal grid, geleneksel olmayan gridler, dönüşümler, görüntü işleme, bilgisayar grafikleri, ayrıklandırılmış döçnüşüm, dijital geometri

DEDICATION

This thesis is dedicated: to the sake of Allah (Almighty) my creator; to my master, great teacher and messenger, Mohammed (Peace be upon him), who taught us the purpose of life; to my great and beloved parents, who never stop giving of themselves in countless ways; to my darling wife, who leads me through the valley of darkness with the light of hope and support; to my treasured brothers and sisters who stand by me whenever things look bleak; to all my family, the symbol of love and giving; to my friends who always encourage and support me; and finally to all the people in my life who have touched my heart, I dedicate this thesis.

ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Benedek Nagy for the continuous support during my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me all through the time of research and writing of this thesis. I could not have imagined having a better supervisor and mentor for my Ph.D. study. My thanks also go to the monitoring jury members, Prof. Rza and Prof. Rashad for their valuable suggestions during the past two years.

Lastly, I would like to acknowledge all the staff members at the Department of Mathematics at Eastern Mediterranean University for their friendly and lovely feeling.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Ω	Continuous coordinate system
2D	Two-dimensional space
3D	Three-dimensional space
$\mathbb{Z}^2$	Two-dimensional squere grid at discrete space
$\mathbb{Z}^3$	Three-dimensional cubic grid at discrete space
BCS	Barycentric coordinate system
CCS	Cartesian coordinate system
RGB	Red, Green, and Blue color value

Chapter 1

INTRODUCTION

Digital geometry or the geometry of the computer screen is the study of geometric properties of digital images (the images displayed on the TV screen or on a computer screen). Digital geometry can be seen as a subcategory of discrete geometry (which has a long history). The motivation for the creation of digital geometry is due to the significant consideration given to the discrete geometry in order to satisfy image processing and computer vision necessities. Therefore, this subject has tremendous new research areas nowadays.

Basically, digital geometry deals with integer points in Euclidean space (the elements are points with integer coordinates) where it considered to be its digitized model. The possible operations and presentations of the digital images are connected directly with the underlying grid, which comprise of discrete points addressed with integer coordinates. There are three regular tessellations of the plane which define the square, hexagonal and triangular grids (named after the form of the pixels used as tiles) [1] (see Figure 1.1).

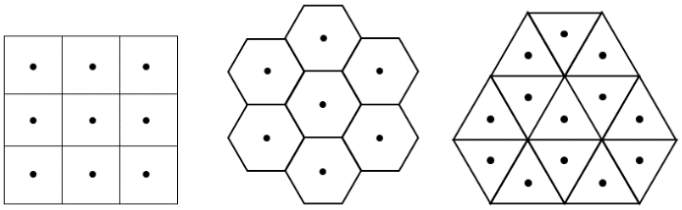


Figure 1.1: The square, the hexagonal and the triangular grids and their grid points

In this thesis, the concentration will be given to the triangular grid. Where a new coordinate system with real numbers will be granted to this grid together with a procedure to add two vectors using this new coordinate system. Furthermore, since the translation operation is strictly connected to the addition operation of two vectors on a coordinate system, a comprehensive study to characterize translation operation on this grid will be introduced as well.

Consequently, this thesis is comprised of four chapters. The introduction is given in this chapter. The new coordinate system is presented in the second chapter, while in the third chapter, characterizing the translation operation on the triangular grid is presented. Finally, this thesis ends with conclusion in chapter four.

Accordingly, this chapter has been divided into three sections. While the first section is a brief historical review of digital geometry, the second is about the coordinate systems of traditional grids. Finally, the transformations on traditional grids are introduced in the third section.

1.1 A Brief History of Digital Geometry

Rosenfeld and Pfaltz started the classical digital geometry in reference [2], where they provide an algorithm that computes various functions on a digital picture based on the distance of a given subset of it. Also, they provide and describe an application that distinguishes the clusters and regularities of the picture and divides its regions into pieces.

In addition to that, on the square grid, $\mathbb{Z}^2$, they defined the two possible neighborhood. And later, in references [3] and [4] the square grid $\mathbb{Z}^2$ and its 3D extension, $\mathbb{Z}^3$ become popular and well known. In reference [5], Voss provided a well-developed theory for

$\mathbb{Z}^n$. However, after 13 years to Voss work, numerous widespread works in discrete and digital geometry are available. A very good book to detail the concepts, algorithms, and practices of this discipline is in reference [6].

Since it is of rapid increase of capabilities power for graphical devices and computers, and based on reference [7] the theoretical study of non-square based structures became worth to consider. The hexagonal, and its dual, the triangular grid have some nice properties, such that they have more symmetries that decrease the computation time and storage spaces, also they have more neighbors to each pixel in comparison to the square grid.

Many authors have studied the using of the hexagonal grid to represent graphics and digital images starting of Golay, in reference [8]. Where in many aspects it shows to be superior to the square grid system as in references [8, 9, 10, 11]. Furthermore, it showed and proved to be optimal in some applications as in references [12, 13, 14].

Accordingly, digital geometry of the triangular grid has been given considerable attention as well. It is proven to have various advantages in applications e.g., by the flexibility of the used neighborhood (it has the largest number and types of neighbors). Basically, digital geometry of the triangular grid is based on various digital distance functions, including neighborhood sequences as in references [15, 16, 17,18] and weighted distances as in references [19, 20, 21, 22]. The triangular grid is recently applied in various image processing algorithms, including discrete tomography as in references [23, 24], mathematical morphology in reference [25], cellular topology in reference [26], and in thinning in references [26, 27].

1.2 Coordinate Systems of Traditional Grids

Digital images are consist of a limited set of pixels. Thus, both the representations of the images and the possible operations on them are strictly connected to the underlying grid. The concept of the grid is essential and heavily used in digital geometry and in digital image processing.

As mentioned earlier, there are three regular grids of the plane the square, hexagonal and triangular grids (see Figure 1.1). An adequate and elegant coordinate system for these kinds of grids are required for their use in both theory and applications, e.g., in image processing or engineering applications. Most of the applications use the square grid because it's orthogonal coordinate system, the well-known Cartesian coordinate system (CCS), which fits very well to it, addressing each square pixel of the grid by a pair of independent integers. The dual of the square grid (that is, the grid formed by the nodes, which are the crossing points of the gridlines) is again a square grid; therefore, essentially, the same CCS is used as well. However, working with real images, we may need to perform operations that do not map the grid to itself, e.g., zooming or rotations. The Cartesian coordinate system (CCS) allows real numbers to be used in such cases, moreover, digitization operation can easily be defined by rounding operation. In fact, the digitization process is when the corresponding pixels are computed after some operations which may result some points (of the plane) that are not gridpoints.

Accordingly, the hexagonal grid, tiling the plane by same size regular hexagons (hexagonal pixels), has been used for decades in image processing applications, as in reference [12], in cartography, as in references [28, 29], in biological simulations, in

reference [30] and in other fields, since the digital geometry of the hexagonal grid provides better results than the square grid in various cases. Also, it is used in various table and computer games based on its compactness. It is the simplest 2D grid, since the only usual neighborhood using the nearest neighbors is simpler and less confusing than the two types of neighbors in the square grid, reference [31], the neighborhood of a pixel contains six other hexagons (See Figure 1.2a).

A three-coordinate-valued system of zero-sum triplets is used to describe the hexagonal grid capturing nicely the triangular symmetry of the grid in reference [9]. In Figure 1.2a, the first coordinate value is ascending to right-upwardly, the second values are ascending into right-downward direction, and the third one is ascending into left-upward direction, references [9, 32]. We should mention that this system could be seen as the extension of the oblique coordinate system using two independent integer values, reference [10] by concerning the third value to obtain 0-sum for every triplet. The digital distance based on the neighborhood relation is computed in reference [10]. Since the vectors describing the grid are not orthogonal, some geometric descriptions based on Cartesian coordinates are not so obvious, however, to simplify the expressions the constrained three-dimensional coordinate system is recommended. We should also mention that 0-sum triplets allowing real numbers were used in reference [32] to describe rotations (that may not map the hexagonal grid to itself). In this way also a useful digitization operator is found. Her's system was mentioned and used e.g., in references [33] and [34] for various imaging related disciplines.

In contrast to the square grid, the dual of the hexagonal grid is not the hexagonal, but the triangular grid (See Figures 1.2a,b). However, the triangular grid has a similar symmetry to the hexagonal grid; therefore, in reference [15] a coordinate system with

0-sum and 1-sum triplets are used to describe this grid (the three values are not independent, since this is also a 2D grid; see Figures 1.2c,d). We note here that Her's 0-sum triplets could match only up to the half of the grid points in the triangular grid (Figures 1.2c,d), and therefore his coordinate system addressing the whole plain cannot be used for the triangular grid. More about the coordinate system for the triangular grid is given in chapter 2.

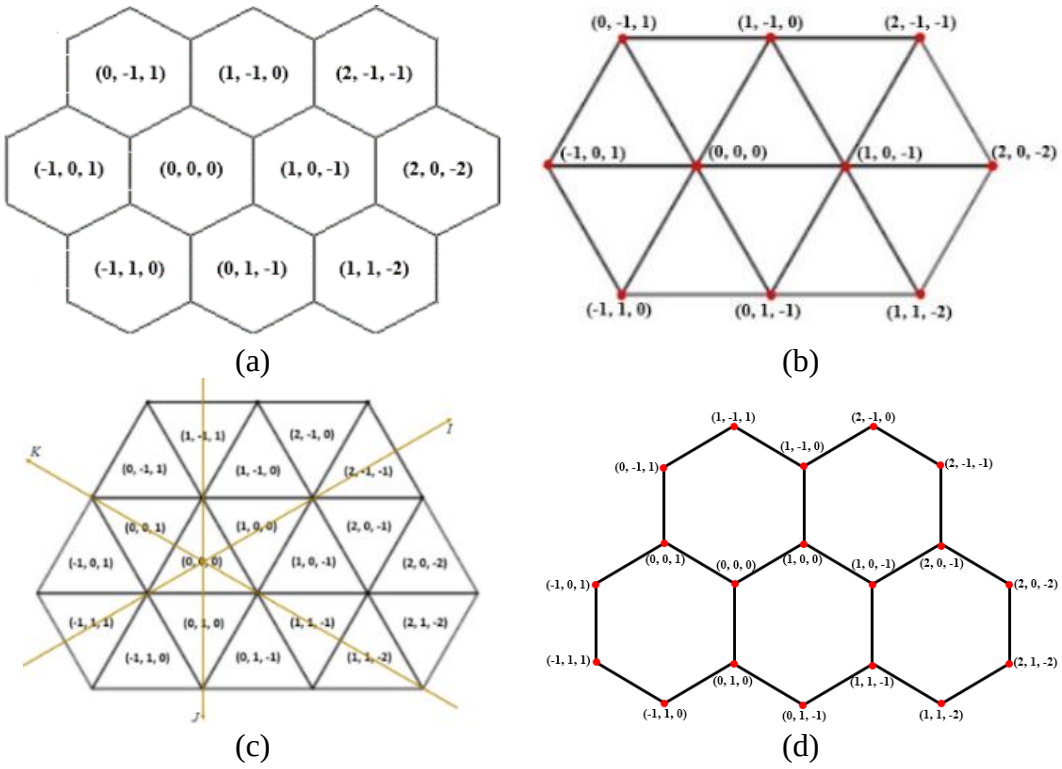


Figure 1.2: (a) The coordinate system for the hexagonal grid and its dual (b). (c) The coordinate system for the triangular grid and its dual (d).

1.3 Transformations on Traditional Grids

Digital geometry is a relatively new field of mathematics that produces and works with several functions over discrete (digital) grids such as discrete rotation and translation. The digitized transformations are informally "close" to their original, Euclidian version, but usually do not satisfy the same properties, such as bijectivity, transitivity. These properties are, in general, hard to keep in the discrete spaces, reference [35],

when a discretized form of an arbitrary Euclidean transformation is considered. Of course, in special cases, e.g., rotating the square grid by a right angle, the discrete transformation is essentially the same as the Euclidean one, but this is not true for the general case. Isometric transformations are essential in the Euclidean space. Equally, we assume to have a similar role for them in discrete geometry theory, too. Also, for example, the definition of a line or of the circle, have to be close to the discrete isometrics since they should reproduce the properties of usual isometric transformations. But up to now, discrete and continuous transformation yield very different theories, reference [36], and thus, it is still a hot topic to consider the former ones.

There are numerous widespread works that are available in discrete and digital geometry, as in reference [6]. A survey on the properties of the major rotation algorithms, and also improvements on Pythagorean rotations can be found in references [37, 38].

The need for isometric rotation is mentioned in many articles. For instance, about the square grid, Kaufman in a lecture for SigGraph'99 [39] said, "*Manipulation and transformation of the discrete volume are difficult to achieve without degrading the image quality or losing some information. Rotation of rasters by angles other than* 90 *degrees is especially problematic since a sequence of consecutive rotations will distort the image*".

Among the three regular two-dimensional grids. The square grid is usually considered in digital geometry, since it is the most known and most frequently used one. Even though, it has some unpleasant properties, e.g., it has a topological (connectivity)

paradox, i.e., the two diagonals of the chessboard cross each other without a shared pixel (without a common gridpoint). In the hexagonal grid, there is only one usual neighborhood, and there is no connectivity paradox, thus it gives a nice and simple option in various applications. Therefore, we can say that, the non-traditional grid: hexagonal and the triangular grids are valid alternatives of the traditional grid both in computer graphics and in digital image processing, moreover, in some cases, they could lead to better results than similar approach on the square grid, reference [21].

Various transformations including the isometric ones are known from geometry. They are also widely used in computers. However, in the discrete (or digital) world most of these transformations are not trivial.

Rotations with the composition of translations based on the square and hexagonal grids have been considered and examined intensively in reference [40]. Translations defined on $\mathbb{Z}^2$, the square grid, are simple and essential for various transformations in several applications related to 2D in image processing such as image matching. The most basic technique to design translation on $\mathbb{Z}^2$ is to combine Euclidian translation defined on $\mathbb{R}^2$ with a digitization operator that maps the outcomes back into $\mathbb{Z}^2$. Normally, the original image is stored as discretized (digital) version of the image. Translations may apply to process the image; which may end in real numbers. As an example, a 2D digital image is represented as a set of $\mathbb{Z}^2$ discrete points. When the Euclidian translation is applied to this digital image, then a set of points that belong to $\mathbb{R}^2$ instead of $\mathbb{Z}^2$ would be obtained (especially when the translation vector is not an integer vector). Consequently, the outcome of the translation must be digitized to obtain an output similar to the input.

Considering the isomorphic transformations on these digital grids, translation has never been considered individually because it obviously leads to an isometric translation on point lattice. Translations of images are such a basic and frequently used operation, which usually do not have any attention alone. One can simply move her or his finger on the screen of her or his mobile/tablet/laptop and various images (icons, signs, or even the whole screen) are translated.

The square and hexagonal grids are point lattices, because when a grid-vector is taken from any gridpoint it will always end up at a certain gridpoint (Figure 1.3). Therefore, any translation by a grid-vector is always a bijection from the grid into itself (for an example, see Figure 3.1 in chapter 3). Also, translations by real vectors can be digitized easily to grid translations. The digitization process plays important roles also on the traditional grid when other operation, e.g., discrete rotation is considered which may not be bijective [35, 40,41].

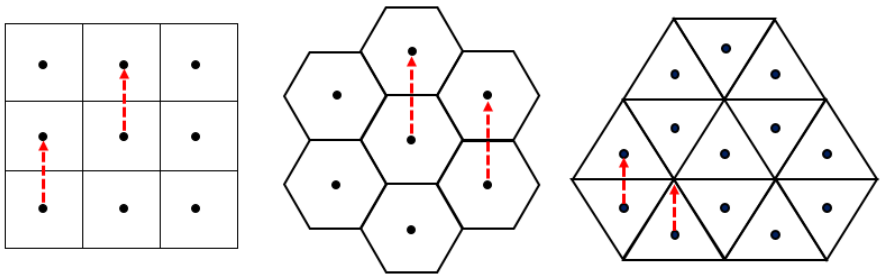


Figure 1.3: Any grid-vector of specific length and direction will lead to a certain gridpoint in the square and the hexagonal grids but not in the triangular grid.

Changing the grid to the triangular grid, the isometric transformations mapping the grid into itself are described in references [42, 43], but transformations that could map some gridpoints out of the grid were not considered. Throughout the translation, some problems arise such as points with no preimage or points with two preimages, which

leads to non-bijectivity translation cases. This is because the triangular grid is not a point lattice, there are grid-vectors that do not translate the grid into itself (see Figure 1.3, on the right, comparing this to the other two grids which are point lattices.)

Consequently, as we show here, it is interesting to consider translations and analyze how the resulted image may change. Therefore, as a part of this thesis, the translation on the triangular grid will be investigated (see chapter 3), where the results of the translations itself are really interesting.

Chapter 2

A COORDINATE SYSTEM FOR THE TRIANGULAR GRID

2.1 Introduction

In general, coordinate systems are essential because they constitute the main tool in making a simple, easily usable and effectively programmable system with integer numbers (coordinates). For the triangular grid, only discrete coordinate systems have been investigated and there was no such extension of this coordinate system that is able to address the entire triangular plane. The isometric transformations on the triangular grid are described in [43]. Therefore, for various applications, including, e.g., arbitrary angled rotations, an extension of the coordinate system is needed.

In this chapter, we introduce a continuous coordinate system for the triangular grid (Ω), where every point of the 2D plane has its unique coordinate triplet. Basically, we use three coordinate values to describe triangular grid as in reference [15], but also to address the points of the plane "between" and "around" the nodes of the dual grid. Our new coordinate system is shown to be an extension of the hexagonal and also of the triangular coordinate systems; moreover, our system builds upon the coordinate system for the so-called, tri-hexagonal grid [44], also called three-plane triangular grid in reference [45]. For further applications, we also provide a mapping between our continuous coordinate system for the triangular grid (Ω) and the Cartesian coordinate

system (CCS) of the 2D plane. Adding two vectors of the new coordinate system is provided as well.

2.2 Preliminaries

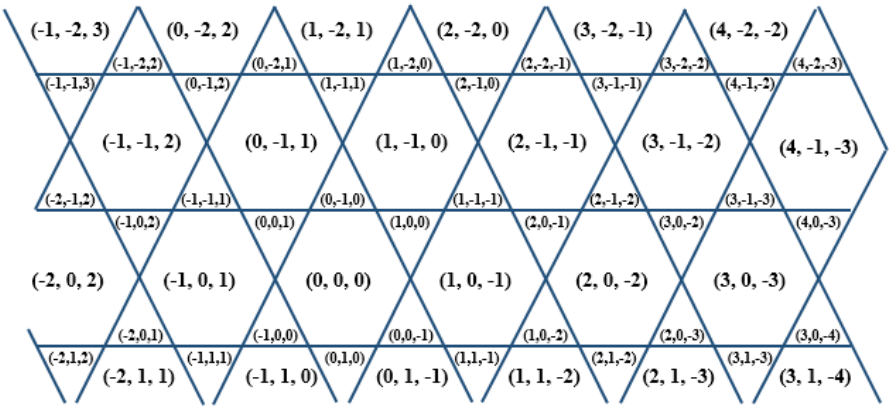
In this thesis, as usual, $\mathbb{Z}^3$ denotes the cubic grid, whose points are addressed by integer triplets according to the three coordinates *x*, *y*, *z*.

In order to create a continuous coordinate system for the triangular grid (Ω) that enables us to uniquely address any point of triangular grid, we combine discrete triangular coordinate systems from reference [46] (see also Figure 1.2 for some examples) with the barycentric coordinate system (BCS), discovered by Möbius (see, e.g., references [1, 47]).

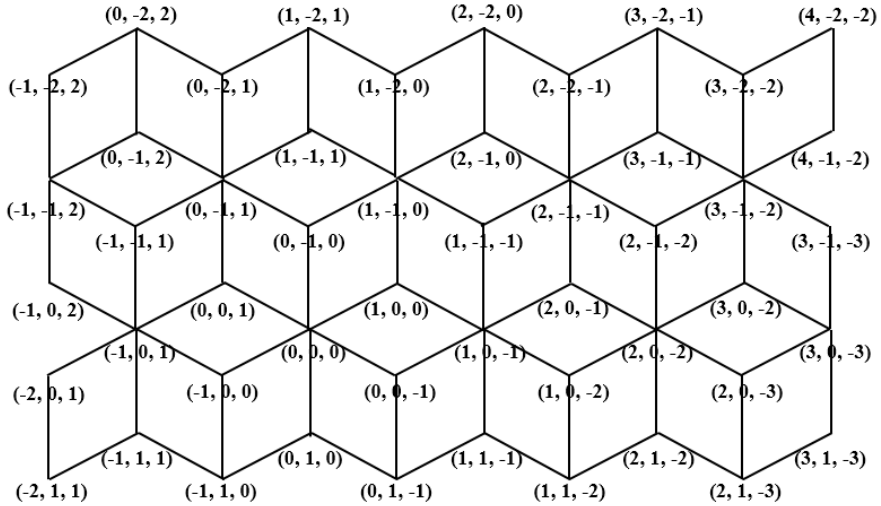
In Figure 2.1, a coordinate system for the tri-hexagonal grid (the three-plane triangular grid of regions [45]) and its dual is given. This grid resembles a mix of the triangular and hexagonal grids since it is a combination of the one- and two-plane grids [9, 46]. The new coordinate system will be an extension of the discrete triangular and hexagonal coordinate systems.

The triangular grid has similar symmetry to the hexagonal grid; therefore, in reference [15] a coordinate system with sum 0 and 1 triplets are used to describe this grid. Observe that exactly the same coordinate triplets are used on the left and on the right-hand side in Figure 1.2c and Figure 1.2d, respectively, which showing the duality of the triangular and hexagonal grids.

Next, a brief description is given for the discrete triangular coordinate system and barycentric coordinate system.



(a)



(b)

Figure 2.1: Representation of the trihexagonal coordinate system (a) and its dual (b). The same coordinate system is used to address the pixels (a) and the nodes of the dual grid (b).

2.2.1 Discrete Triangular Coordinate System

The discrete Hexagonal Coordinate System uses 0-sum triplets (Figure 1.2a,b). The discrete Triangular Coordinate System [46] is a symmetric coordinate system that addresses each pixel by an integer triplet. The three coordinate axes have angles of 120° as in the hexagonal grid. The sum of the triplets is equal to 0 or 1 referring to the two types of orientations of triangles (△,▽). The triangles with 0-sum are the “even” triangles (△), and the triangles with 1-sum triplets are the “odd” triangles (▽) (see Figure 1.2).

For finding an appropriate extension to this system that is able to address all points of the 2D plane, we start by addressing the midpoints of triangles with integer triplets of +1 and –1 sum; therefore, we call them "Positive △" and "Negative ▽" triangles, respectively. See Figure 2.2, where the coordinate system of Figure 2.1b is used to address midpoints of triangles of the triangular grid. Observe that each triplet assigned to a midpoint (see the blue triplets in Figure 2.2) builds up from the coordinate values shared by two of the corners of the given pixel (see the three red triplets around each blue triplet). There is already an important difference between our proposed and Her's 0-sum coordinate system: he used 0-sum triplets to address these midpoints as well (actually, 3 fractional values for each midpoint) which was a very good and efficient choice to extend the coordinate system of the hexagonal grid. However, it does not meet our requirements, therefore we have fixed these coordinate values in another way. We should also mention here, that Her's system inside the regular triangles can be seen as an application of the barycentric coordinate system (see next subsection) based on the values assigned to the corners of the triangle.

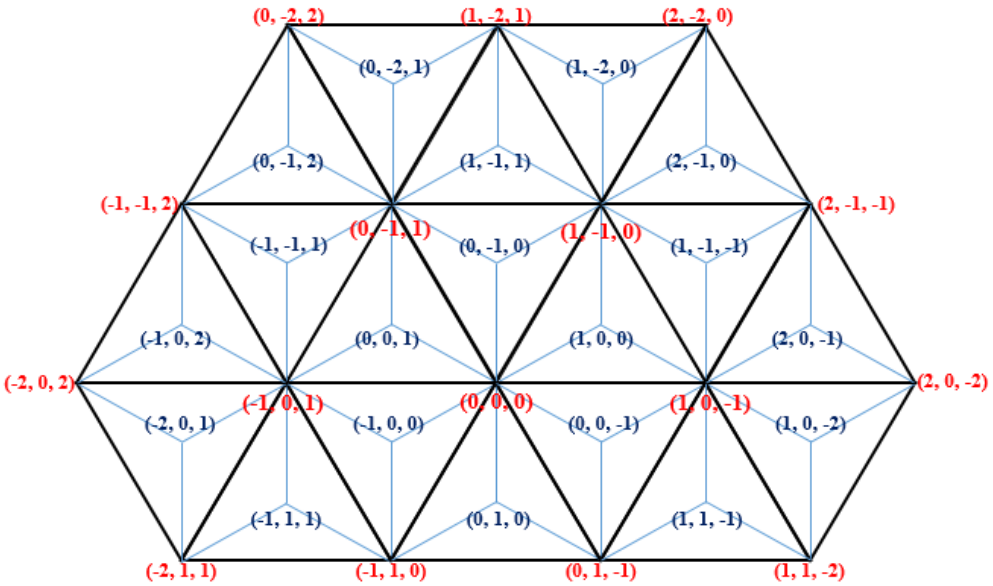


Figure 2.2: The coordinate system for the trihexagonal grid is used for the triangular grid (and also for its dual, at the same time).

2.2.2 The Barycentric Coordinate System (BCS)

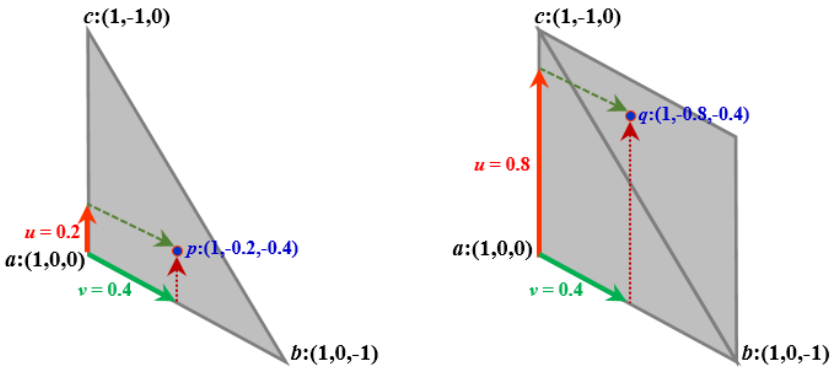
One of the main motivations of the barycentric technique is to use a coordinate system only for a finite (bordered) segment of the plane which is also more "balanced" inside this region than the values of the Cartesian frame. In this way, they do not depend on the choice of a fixed point called "origin" in the Cartesian system. BCS is also used in computer graphics, e.g., to interpolate RGB colors inside a triangle.

The barycentric technique uses coordinate triplets to address any points inside (and on the border) of a given triangle. One may think about it as follows: we put three 1-sum weights, (w, v, u), to the corners let us say, a, b, c of the triangle and the mass center p inside the triangle is assigned to that triplet of weights. It is also known that if the area of the triangle abc is one unit, then the areas of bcp, acp, and abp are exactly w, v and u, respectively. The coordinates of p can be computed from the coordinates of the corners of the triangle by weighted average, i.e., $p = wa + vb + uc$ (where p, a, b and c are the vectors representing the positions of these points). This formula can easily be transformed to the following formula using the fact that $w = 1 - v - u$:

$$p = a + v\,(b - a) + u\,(c - a). \qquad (2.1)$$

Actually, since the three barycentric coordinate values (w, v and u) of point p are not independent (sum of 1), we may use only two of them, v and u here, to address the point p similarly as in an oblique coordinate system. One may understand equation (2.1) as stating that the starting point is a, and we can go to the direction of b and the direction of c by some distance, indicated by v and u, respectively. However, in our approach, the starting point a is the midpoint of the regular triangle and, according to the values of v and u, the coordinate triplet for point p is calculated (see Figure 2.3 and Example 2.1).

In classical barycentric technique, the point is considered to be inside a triangle, as long as the sum of (u and v) is between 0 and 1: $0 < (u + v) < 1$. If the sum is equal to 1 then the point will be on the edge (cb), while it will be out of the triangle if the sum is less than 0 or greater than 1 (see, e.g., Figure 2.3a). Now, we relax the condition of barycentric technique and allow the sum of u and v to be any real number between 0 and 2, besides the conditions $0 \leq u \leq 1$ and $0 \leq v \leq 1$ hold (see, e.g., Figure 2.3b). In this way we may also address some points outside of the triangle. In Figure 2.3 and Example 2.1, a composition of the barycentric technique and the discrete coordinate system (assigned to the corners of an isosceles triangle in the regular triangle) is given to address other points of the plane.



(a) (b)

Figure 2.3: A composition of the barycentric technique and discrete coordinate system to address points p and q in the triangular plane by coordinate triplets.

Example 2.1 Consider the triangle defined by corners (1, 0, 0), (1, 0, –1) and (1, –1,0), which represent the three vertices a, b and c, respectively. Let $u = 0.2$ and $v = 0.4$, where $0 < (u + v) < 1$. Then, based on equation (2.1), the coordinate triplet of point p is (1,–0.2,–0.4) (see Figure 2.3a and Table 2.1). If $u = 0.8$ and $v = 0.4$ ($0 < (u + v) < 2$), then the coordinate triplet of point q is (1, – 0.8, – 0.4) (See Figure 2.3b).

Table 2.1: Computing the coordinate triplet of point *p* using equation (2.1).

Coordinate Values of the Triplet	$p = a + v(b - a) + u(c - a)$	***p***
1st Coordinate Value	p = 1 + 0.4 (1 – 1) + 0.2 (1 – 1)	1.00
2nd Coordinate Value	p = 0 + 0.4 (0 – 0) + 0.2 (–1 – 0)	– 0.2
3rd Coordinate Value	p = 0 + 0.4 (–1 – 0) + 0.2 (0 – 0)	– 0.4

The point *q* can be calculated in the same manner of point *p* in Table 2.1.

2.3 Continuous Coordinate System for the Triangular Plane

In order to create a continuous coordinate system for triangular plane that works efficiently, we combine the discrete coordinate system for the triangle grid with BCS. In the discrete triangular coordinate system, integer coordinate triplets with various sums were used. In BCS coordinate triplets with fractional values address the points inside a triangle. We develop a new system which uses triplets on the entire plane. We start by dividing each equilateral triangle of the triangular grid into three isosceles obtuse-angled triangles, which will possess areas A, B, and C, as shown in Figure 2.4.

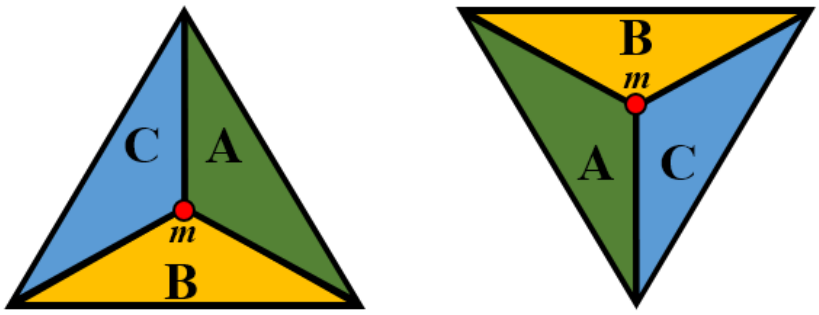
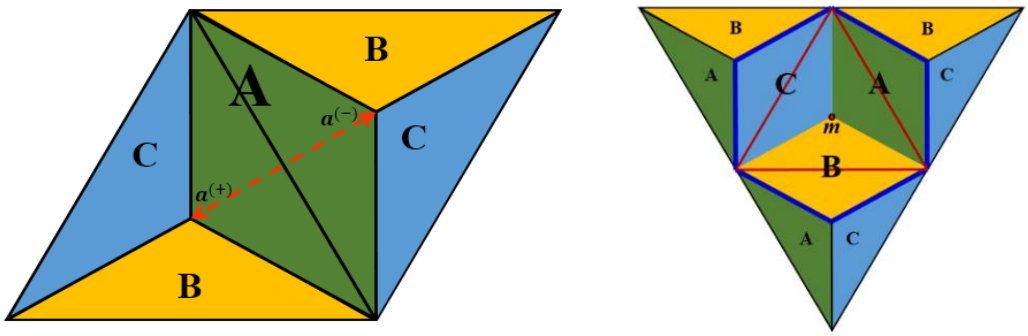


Figure 2.4: Dividing each triangle to three areas A, B and C. The letters assigned to the isosceles triangles are based on the orientation of sides.

In this case, the midpoint *m* between areas will be the start point (the red point), which is represented by symbol *a* in equation (2.1). This point will be used to calculate the coordinates of the points in the three areas A, B, and C.

As we have already mentioned above, we use coordinate triplets with sum +1 and –1 for these midpoints, depending on the orientation of the original triangle: the sum of 1 represents the midpoint of the positive triangles △ and the sum of –1 represents the midpoint of negative triangles ▽. Therefore, using the barycentric, equation (2.1), based on these midpoints, we obtain a unique triplet for each point in each area of the plane as we will describe below.

Based on the barycentric, equation (2.1), we know that the values u and v are limited by $0 \leq (u + v) \leq 1$ (inside or on the border of the given triangle), which gives the ability to address the points inside areas A, B and C of each type of triangle (△,▽) separately. However, let us consider the case in which the sum of u and v satisfies $0 \leq (u + v) \leq 2$, such that the conditions $0 \leq u \leq 1$ and $0 \leq v \leq 1$ hold. Then, consequently, each midpoint can be used to address not only the points in the area located in this original triangle but also the points in the neighboring area denoted with the same letters. To illustrate this, the green area in Figure 2.5a can be completely addressed by using either midpoint $a^{(+)}$ or $a^{(-)}$ as the starting point in equation (2.1).



(a) (b)

Figure 2.5: (a) By using either $a^{(+)}$ or $a^{(-)}$, the whole green area A could be addressed. (b) The hexagon surrounded by the thick dark blue line shows the entire area that can be addressed by using a positive midpoint m.

Proposition 2.1 *To address the points inside a rhombus* A, B *or* C *the coordinate triplet of a point does not depend on the choice of whether the midpoint of the positive or the negative triangle is used for the addressing.*

Proof. *Assume point* (p) *in area A of negative triangle. Let* $a^{(+)} = (a_1, a_2, a_3)$, $a^{(-)} = (a_1, (a_2 - 1), (a_3 - 1))$, $p = (p_1, p_2, p_3)$, $b = (b_1, b_2, b_3)$ *and* $c = (c_1, c_2, c_3)$ *where* $a^{(+)}$ *is the midpoint of positive triangle and* $a^{(-)}$ *is the midpoint of negative one.* (*See Figure* 2.6).

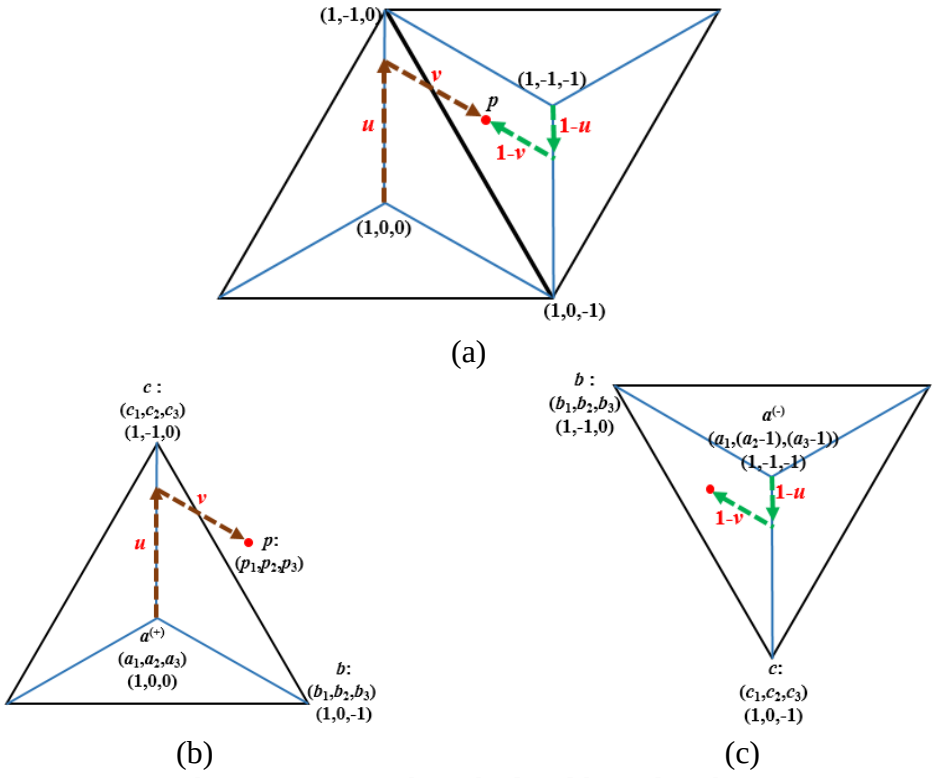


Figure 2.6: Proving how point p can be calculated by either the positive or negative midpoint ($a^{(+)}$ or $a^{(-)}$). (a) Shows the position of point p with respect to both a positive and a negative triangle, while (b) and (c) represent the calculation of the coordinates of point p based on the positive and negative triangles, respectively.

As we mentioned earlier that each triplet assigned to a midpoint builds up from the coordinate values shared by two of the corners of the given pixel. (*See Sub-section* 2.2.1).

Thus, we have some equalities:

In the positive triangle, we have: $a_1 = b_1 = c_1, b_2 = a_2, c_3 = a_3$ *and* $c_2 = a_2 - 1$,

In the negative triangle, we have: $a_1 = b_1 = c_1, b_2 = a_2, c_3 = a_3$ *and* $b_3 = a_3 - 1$.

Now, to compute the 1^{st} *coordinate value,* p_1*, from the positive triangle we have:*

$p_{1(+)} = a_1 + v\,(b_1 - a_1) + u\,(c_1 - a_1)$, *since* $a_1 = b_1 = c_1$ *then* $p_{1(+)} = a_1$.

From the negative side we have:

$p_{1(-)} = a_1 + (1 - v)\,(b_1 - a_1) + (1 - u)\,(c_1 - a_1)$, *then also* $p_{1(-)} = a_1$.

For the 2^{nd} *coordinate value,* p_2*, from positive triangle we have:*

$p_{2(+)} = a_2 + v\,(b_2 - a_2) + u\,(c_2 - a_2)$, *since* $b_2 = a_2$, *then* $p_{2(+)} = a_2 + u\,(c_2 - a_2)$,

using $c_2 = a_2 - 1$ *we get* $p_{2(+)} = a_2 - u$.

From the negative side we have:

$p_{2(-)} = (a_2 - 1) + (1 - v)\,(b_2 - (a_2 - 1)) + (1 - u)\,(c_2 - (a_2 - 1))$,

and since $b_2 = (a_2 - 1)$, *then it is* $p_{2(-)} = (a_2 - 1) + (1 - u)\,(c_2 - (a_2 - 1))$,

since $c_2 = a_2$ *then we have* $p_{2(-)} = a_2 - u$.

Finally for the 3^{rd} *coordinate value,* p_3*, from positive triangle we have:*

$p_{3(+)} = a_3 + v\,(b_3 - a_3) + u\,(c_3 - a_3)$. *Since* $c_3 = a_3$, $p_{3(+)} = a_3 + v\,(b_3 - a_3)$,

further, $b_3 = a_3 - 1$ *which yields to* $p_{3(+)} = a_3 - v$.

From the negative side we have:

$p_{3(-)} = (a_3 - 1) + (1 - v)\,(b_3 - (a_3 - 1)) + (1 - u)\,(c_3 - (a_3 - 1))$,

since $c_3 = a_3 - 1$ *and* $b_3 = a_3$, *it can be written as:*

$p_{3(-)} = (a_3 - 1) + (1 - v)\,(b_3 - (a_3 - 1)) = a_3 - v$.

Having the point inside other regions the proof goes in a similar manner. ■

As we have already mentioned, a popular way to understand BCS for a point (inside a triangle) goes by the ratio of the areas defined by the triangles determined by the point and two of the triangle corners. In fact, our system uses a similar technique to address the points inside a triangle as it is stated in the next corollary based on the previous proposition.

Corollary 2.1 *Let p be any point inside or on the border of an obtuse-angled triangle determined by a midpoint* $a = (a_1, a_2, a_3)$, *and corners* $b = (b_1, b_2, b_3)$, $c = (c_1, c_2, c_3)$. *Let the barycentric coordinates of p be* (w, v, u), *with* $w + v + u = 1$, *i.e., by assigning these weights to a, b and c, respectively, the weighted midpoint is at p. Then the coordinates of* $p = (p_1, p_2, p_3)$ *are exactly* $p_i = w\, a_i + v\, b_i + u\, c_i$ *for* $i = 1, 2, 3$.

Notice that the three points *a*, *b* and *c* above must have a fixed coordinate value (depending on the type of the triangle). The weighted average of this coordinate value will be the same for any point inside or on the border of this obtuse-angled triangle.

As we have seen a given triplet of corners, including a midpoint can be used to address points not only at the inside and on the border of the triangle determined by them. The type of these regions plays importance in this issue. In Figure 2.5b, the thick, dark blue line shows the entire area that the positive midpoint can address. The key issue is to use triplets of the discrete coordinate system and to use only 2 barycentric fractional values inside, by using the directions of the sides of the appropriate rhombus in which the point is located. The sides of a rhombus are actually parallel to two of the coordinate axes. Hereafter, for simplicity, we will use only the positive midpoints for further calculations, while ignoring the negative ones. The triangular plane can be seen as in Figure 2.8a.

Actually, in Figure 2.2, the cubes with light-blue edges are clearly noticeable (also Figure 2.8a can be seen as a 3D image of cubes). Therefore, it is very easy to look at these figures not only as 2D figures but as 3D figures: a surface that built up by same size square tiles, each oriented to be parallel to two of the Cartesian coordinate axes. Our squares marked by A, B and C are parallel to axes *y* and *z*; to *x* and *z*; and to *x* and *y*, respectively. Thus, to obtain the 2D view, the "triangular plane," where the discrete part of the system is exposed by an embedding in $\mathbb{Z}^3$, the light-blue cubes give the lower boundary, say B, (e.g., the rhombus region of the points (0,0,1), (0,0,0), (–1,0,0), and (–1,0,1)), of a naive digital plane. Therefore our proposed coordinate system can also be seen as the orthogonal projection of regions, like B, on the plane defined by the black lines (the plane of the triangular grid lines in Figure 2.2), where points have 0-sum.

In the next two subsections, we will illustrate the conversion between the Ω to/from the CCS. Namely, we can convert the coordinate triplet of a certain point in our new coordinate system to its corresponding Cartesian coordinates and vice versa.

2.3.1 Converting Triplets to Cartesian Coordinates

Assume that we use (*i*, *j*, *k*) as a coordinate triplet of a point, where *I, J,* and *K* are the axes of triangular plane (see, e.g., Figure 1.2c), and also, suppose (*x*, *y*) is used to indicate the same point in the Cartesian plane where *X* and *Y* are the axes.

For the conversion, we fix the side-length of the triangle of the triangular grid to $\sqrt{3}$. Consequently, its height is 1.5 (see the dashed blue lines in Figure 2.7). Then, the following matrix equation (2.2) computes the corresponding coordinate values *x* and *y* for the given triplet (*i*, *j*, *k*):

$$\begin{pmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} i \\ j \\ k \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} \sqrt{3}(i-k) \\ i-2j+k \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.2}$$

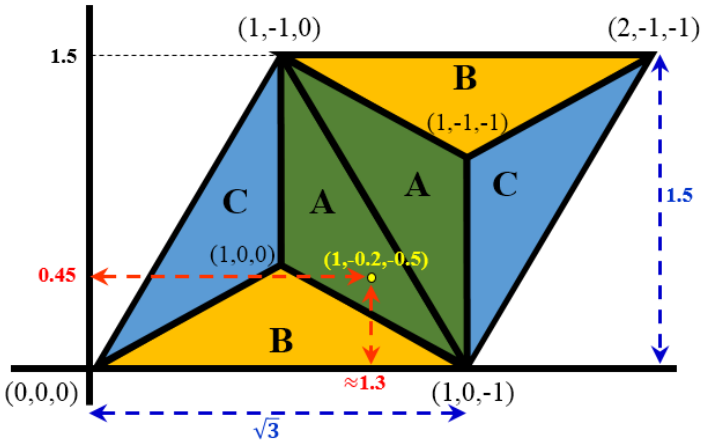


Figure 2.7: The dashed red lines indicate the Cartesian coordinates of the point.

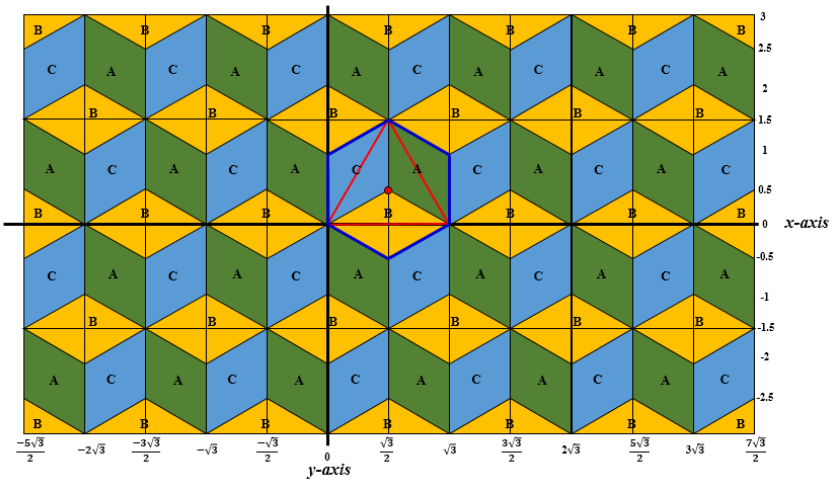
Example 2.2 Let (1, –0.2, –0.5) be a point in the triangular plane. Then, based on equation (2.2):

$$\begin{pmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -0.2 \\ -0.5 \end{pmatrix} = \begin{pmatrix} \approx 1.3 \\ 0.45 \end{pmatrix}$$

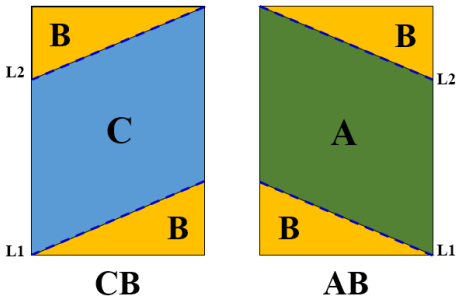
Thus, $(x, y) \approx (1.3, 0.45)$ as shown in Figure 2.7.

2.3.2 Converting Cartesian Coordinates to Equivalent Triplets

One of the simplest ways to do such conversion is to determine the midpoint and the two corner points which defines the triangle in which the given point locates (inside or on the border). Then, by computing the barycentric coordinates (weights) of the point with respect to these triangle corners, by Corollary 2.1, the continuous coordinate triplet is computed. In this subsection, we present a slightly different method with little more details to convert Cartesian coordinates to continuous triangular coordinates.



(a)



(b)

Figure 2.8: (a) Re-structuring the triangular plane to fit the Cartesian plane. (b) The two distinguished rectangles of the plane.

As we already mentioned earlier, the midpoints of positive triangles are used to address all points in the triangular plane. Therefore, every positive midpoint will address areas A, B and C in neighbor (negative) triangles, as already seen in Figure 2.5b. Consequently, the triangular plane will be re-structured as can be seen in Figure 2.8a. However, two kinds of rectangles can be clearly distinguished in this plane, called CB and AB (see Figure 2.8b).

We may start by specifying the area (i.e., rhombus) A, B or C that a Cartesian point (x, y) belongs to. Then we can use appropriate formulae that are assigned to each type of area, a process we will describe later in this section (see Table 2.2). Hence, the following three steps are used to specify the area:

Step 1: Which quarter of the Cartesian plane is involved? Note that the 1st and 3rd quarters have the same structure, while the 2nd and 4th quarters have another one.

Step 2: Which rectangle is involved (AB or CB)?

Step 3: Which area is involved (A, B or C)?

We show how these steps can be computed by pseudo codes. The first step is the easiest one since we can inquire whether the values of (x, y) are greater or equal to zero or not; then, this task is completed and the involved quarter is specified. (See Code 1)

Code 1:

IF (((x ≥ 0) AND (y ≥ 0)) OR ((x < 0) AND (y < 0)))

THEN "1st or 3rd quarter"

ELSE "2nd or 4th quarter"

To complete the second step, note that the basic measurements of every rectangle are known, with a height and width equal to 1.5 and $(\sqrt{3}/2)$, respectively. Then, the CB rectangle is involved whenever the integer part of $2x/\sqrt{3}$ and y/1.5, are both even or both odd, otherwise the AB rectangle is involved. See Code 2 to clarify this step.

Code 2:

IF ((int $(2x/\sqrt{3})$ mod 2 = 0) AND (int (y/1.5) mod 2) = 0) OR

(int $(2x/\sqrt{3})$ mod 2 = 1) AND (int (y/1.5) mod 2) = 1))

THEN "CB Rectangle is involved"

ELSE "AB Rectangle is involved"

where *int* takes the integer part of the decimal number and *mod* is the modulus (or remainder, after division by two).

Now, the involved rectangle is specified and the third step follows. The two types of rectangles AB and CB are symmetric, thus, we briefly discuss only one of them, namely, type AB rectangles

In rectangle AB, a point (*x, y*) will belong to either part A or part B. Consider Line 1 (L1) and Line 2 (L2) in Figure 2.8b: they divide the rectangle into parts A and B. If the point is between L1 and L2 or on one of them, then part A is involved, otherwise it is in part B. Equations (2.3) and (2.4) of a straight line are used for Line 1 and Line 2, respectively:

$$m \cdot x + r_1 - y = 0, \tag{2.3}$$

$$m \cdot x + r_2 - y = 0, \tag{2.4}$$

where, m is the slope of L1 and L2, which is $(-\sqrt{3}/3)$; r_1 and r_2 are *y-axis* intercept with L1 and L2 respectively, thus $r_2 = r_1 + 1$.

This step is started by substituting the point (*x, y*) in equations (2.3) and (2.4). Then, it is determined that part A is involved if the left side of equation (2.3) is not greater than 0 and the left side of equation (2.4) is not less than 0; otherwise, part B is involved. Code 3 is used to clarify this step.

Code 3:

IF ($r_1 - x\sqrt{3}/3 - y) \leq 0$ AND ($r_2 - x\sqrt{3}/3 - y) \geq 0$

THEN "Area A is involved"

ELSE "Area B is involved"

The *y-axis* intercept with Line 1, r_1, in area AB, is computed by computing the bottom-right corner of rectangle AB (see the red point in Figure 2.9). Hence, equation (2.5) is used to find the value of r_1. In equation (2.5), we used the modulus (mod) as a function naturally extended also to real numbers, i.e., it gives the remainder after division by a real number (and that is between 0 and the divisor). Moreover, by adding 1 to r_1 we get the value of *y-axis* intercept with Line 2, r_2 (see Figure 2.9).

$$r_1 = y - \left(y \bmod \frac{3}{2}\right) + \frac{\sqrt{3}}{3} \cdot \left(x + \frac{\sqrt{3}}{2} - \left(x \bmod \frac{\sqrt{3}}{2}\right)\right) \tag{2.5}$$

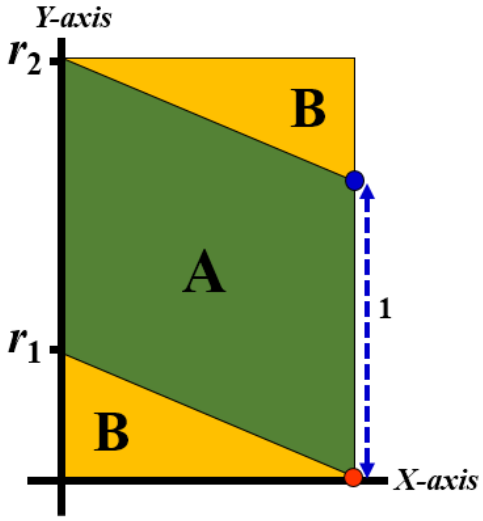


Figure 2.9: The red point is used to compute the value of r_1, which is the *y*-axis intercept with line 1.

Similar strategies are used when the CB rectangle is involved, taking care that the slopes of Line 1 and Line 2 is $(\sqrt{3}/3)$.

Finally, when the involved area A, B or C is specified, particular formulae are used that are specified in Table 2.2. See Example 2.3 for further explanation.

Table 2.2: The coordinate triplets formulae, based on area type, where $\langle ... \rangle$ is a rounding operation*.

	Area A	**Area B**	**Area C**
i	$\langle \frac{x}{\sqrt{3}} \rangle + \langle \frac{y}{3} \rangle$	$\frac{x\sqrt{3}}{3} + y + j$	$\frac{2x}{\sqrt{3}} + k$
j	$\frac{i+k}{2} - y$	$\langle \frac{-2y}{3} \rangle$	$\frac{i+k}{2} - y$
k	$i - \frac{2x}{\sqrt{3}}$	$i - \frac{2x}{\sqrt{3}}$	$\langle \frac{y}{3} \rangle - \langle \frac{x}{\sqrt{3}} \rangle$

*rounding operation returns the nearest integer to the real number, such that numbers exactly the same distance from two integers are rounded to the larger absolute valued one, e.g. $\langle 1.5 \rangle = 2$, $\langle -1.5 \rangle = -2$ and $\langle -0.4 \rangle = 0$.

Example 2.3 Consider a point of CCS: (x, y) = (1.299, 0.45). In order to convert this Cartesian coordinate pair to its corresponding triangular triplet, the three steps above will be followed:

Step 1: Based on Code 1: if x and y are positive, then the 1st or 3rd quarter is involved.

Step 2: Based on Code 2: if (int $\left(\frac{2x}{\sqrt{3}}\right)$ = 1) is odd, and (int (y / 1.5) = 0) is even, then rectangle AB is involved.

Step3: Substitute (x, y) in equations (2.3) and (2.4), and based on Code 3, area A is matched.

Thus, formulae of area A (see Table 2.2) should be applied in this order, so:

1) $i = \langle 0.75 \rangle + \langle 0.15 \rangle = 1 + 0 = 1$

2) $k = 1 - \frac{2x}{\sqrt{3}} \approx -0.500$.

3) $j \approx -y + 0.5 \cdot 0.5 = -0.2$

The corresponding triplet is (1,–0.2,–0.5) which is approximately the same as in Example 2.2, as it should be.

To show the conversion of points on other areas (e.g. Area B or Area C), the following examples are given.

Example 2.4 (Point in Area B)

a) Converting from Continuous Coordinate System to CCS.

Let (i, j, k) = (0.683, 0, –0.183) be a point in the triangular plane. To convert to CCS, equation (2.2) is used:

$$\begin{pmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 0.683 \\ 0 \\ -0.183 \end{pmatrix} \approx \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix}$$

b) Converting from CCS to Continuous Coordinate System.

Let (x, y) = (0.75, 0.25). The corresponding Continuous Coordinate System triplet can be calculated based on the three steps above as the following:

Step 1: It belongs to the 1st quarter.

Step 2: It belongs to rectangle CB.

Step 3: Area B is matched.

Thus, formulae of area B from Table 2.2 are applied in the following order:

1) $j = \langle \frac{-2y}{3} \rangle = 0$

2) $i = \frac{x\sqrt{3}}{3} + y + j \approx 0.433 + 0.25 + 0 = 0.683$

3) $k = i - \frac{2x}{\sqrt{3}} \approx 0.683 - 0.866 = -0.183$

The corresponding triplet is (i, j, k) = (0.683, 0, –0.183) which is exactly the original value.

Example 2.5 (Point in Area C)

a) Converting from Continuous Coordinate System to CCS.

Let (i, j, k) = (0.346, – 0.626, 0) be a point in the triangular plane. To convert to CCS, we use equation (2.2):

$$\begin{pmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 0.346 \\ -0.626 \\ 0 \end{pmatrix} \approx \begin{pmatrix} 0.299 \\ 0.799 \end{pmatrix}$$

b) Converting from CCS to Continuous Coordinate System.

Let (x, y) = $(0.299, 0.799)$. The corresponding Continuous Coordinate System triplet can be calculated based on the three steps as the following:

Step 1: It belongs to the 1st quarter.

Step 2: It belongs to rectangle CB.

Step 3: Area C is matched.

Thus, formulae of area C from Table 2.2 are applied in the following order:

1) $k = \langle \frac{y}{3} \rangle - \langle \frac{x}{\sqrt{3}} \rangle = 0 - 0 = 0$

2) $i = \frac{2x}{\sqrt{3}} + k \approx 0.346 + 0 = 0.346$

3) $j = \frac{i+k}{2} - y \approx 0.173 - 0.799 = -0.626$

The corresponding triplet is (i, j, k) = (0.346, –0.626, 0) which is exactly the original triplet.

Example 2.6 (A mid-point)

a) Converting from Continuous Coordinate System to CCS.

Let (i, j, k) = (1, 0, 0) be a point in the triangular plane. To convert to CCS, equation (2.2) is used:

$$\begin{pmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix}$$

b) Converting from CCS to Continuous Coordinate System.

Let $(x, y) = (\sqrt{3}/2, 0.5)$. The corresponding Continuous Coordinate System triplet can be calculated based on the three steps as the following:

Step 1: It belongs to the 1st quarter.

Step 2: based on Code 2, it belongs to rectangle CB (but, since it is a mid-point, then either rectangle AB or CB may use).

Step 3: Area B is matched.

Thus, formulae of area B from Table 2.2 are applied in the following order:

1) $j = \langle \frac{-2y}{3} \rangle = 0$

2) $i = \frac{x\sqrt{3}}{3} + y + j = 0.5 + 0.5 + 0 = 1$

3) $k = i - \frac{2x}{\sqrt{3}} = 1 - 1 = 0$

The corresponding triplet is $(i, j, k) = (1, 0, 0)$ which is exactly the original value.

2.3.3 Properties of the Continuous Triangular Coordinate System

In this section, we will focus on the most important properties of Ω.

2.3.3.1 On the Triplets of a General Point

In Figure 2.5a, consider the red straight line between $a^{(+)}$ and $a^{(-)}$, in the green area. Then, the sum of the coordinate values of the points on this line would change continuously from 1 until –1. Depending on the sum, we can classify the points as follows.

If a point with triplet's sum is:

- equal to 1, then it is a positive midpoint (i.e. $a^{(+)}$);
- equal to –1, then it is a negative midpoint (i.e. $a^{(-)}$);
- equal to 0, then the point is on the triangle's edge;
- positive, then the point belongs to positive triangle;
- negative, then the point belongs to negative triangle.

Theorem 2.1 *The sum of the coordinate triplet of any point in the plane is in the range of the closed interval* [–1, 1].

Proof. *Consider an area A of a positive triangle with the corners* $a = (a_1, a_2, a_3)$, $b = (b_1, b_2, b_3)$ *and* $c = (c_1, c_2, c_3)$, *where b and c are vertices* (*corners of an equilateral triangle) of the grid, while a is the midpoint of a positive triangle and* $p = (p_1, p_2, p_3)$ *is a randomly chosen point belonging to this area (inside or on the border of A*). *Now based on the barycentric equation* (2.1) *we have*:

$$\sum_{i=1}^{3} p_i = \sum_{i=1}^{3} a_i + v \cdot \left(\sum_{i=1}^{3}(c_i - a_i)\right) + u \cdot \left(\sum_{i=1}^{3}(b_i - a_i)\right)$$

It is clear that, $\sum_{i=1}^{3} a_i = 1$, *whereas*:

$$\begin{aligned}\sum_{i=1}^{3}(c_i - a_i) &= (c_1 - a_1) + (c_2 - a_2) + (c_3 - a_3)\\ &= (c_1 + c_2 + c_3) - (a_1 + a_2 + a_3)\\ &= 0 - 1 = -1\end{aligned}$$

Similarly, $\sum_{i=1}^{3}(b_i - a_i) = -1$.

Then, by substitution, we have:

$$\sum_{i=1}^{3} p_i = 1 - u - v = 1 - (u + v). \qquad (2.6)$$

Since $0 \leq u + v \leq 2$*, the maximal and minimal value of the sum of any coordinate triplet is equal to* 1 (*when* $u + v = 0$) *and to* –1 (*when* $u + v = 2$)*, respectively.* ■

Theorem 2.2 *The sum of the coordinates of a triplet is non-negative in the positive triangle and non-positive in the negative triangle.*

Proof*. Consider a point p that belongs to a positive triangle. Clearly, the coordinates of p are based on u and v such that* $0 \leq u + v \leq 1$*. Now, by substituting* $u + v$ *in equation* (2.6) (*at the proof of Theorem* 2.1)*, the summation will always be nonnegative* (*moreover, it is positive inside the triangle*).

Similarly, let p belong to a negative triangle, then $1 < u + v \leq 2$*, thus, by substitution into equation* (2.6)*, the sum will be always a non-positive value.* ■

Every point in the triangular plane has at least one integer value in its triplet, moreover, the place of the integer value indicates to its area (A, B or C).

Theorem 2.3 *The* 1^{st} *coordinate value of every point in area* A *is the same as the* 1^{st} *coordinate value of the midpoint. The* 2^{nd} *coordinate value of every point in area* B *equals the* 2^{nd} *coordinate value of the midpoint. Similarly, the* 3^{rd} *coordinate value of any point in area* C *equals the* 3^{rd} *coordinate value of midpoint.* (Figure 2.10).

Proof*. Consider an area A of a positive triangle with the corners* $a = (a_1, a_2, a_3)$*,* $b = (b_1, b_2, b_3)$ *and* $c = (c_1, c_2, c_3)$*, where b and c are vertices* (*corners of an equilateral triangle*) *of the grid, while a is the midpoint and* $p = (p_1, p_2, p_3)$ *is a randomly chosen point belonging to this area* (*i.e. inside or on the border of the triangle abc*)*. Since it*

is area A, we have $a_1 = b_1 = c_1$*, now substituting this into equation* (2.1)*,* $p_1 = a_1$ *follows for any point p in this area. A similar proof can be considered for areas B and C.* ■

If a triplet contains two integer values, then the point is located on the line bordering the areas. For example, a triplet of the form (1, 0, *k*) addresses a point on the line (side of the obtuse-angled triangle) between area A and B ($0 \leq k \leq -1$). However, if three integers are a triplet, then this triplet addresses either a midpoint or a vertex (corner) of a triangle.

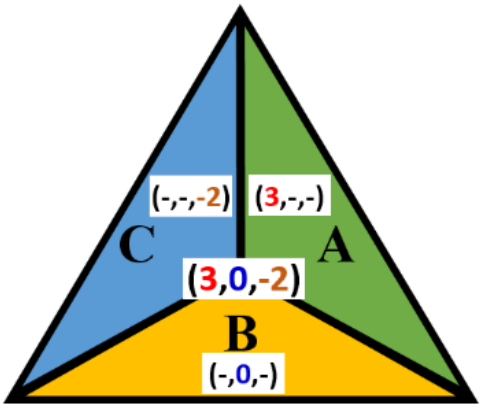


Figure 2.10: The corresponding constant coordinate value for each area.

2.3.3.2 Relation to Discrete Coordinate Systems

In [45], the hexagonal grid is called a one-plane triangular grid since it is a sub-plane of $\mathbb{Z}^3$ and because of its symmetry. The triangular grid (nodes of the hexagonal grid) is called a two-plane triangular grid. Combining one- and two-plane grids produces the so-called three-plane triangular grid, the trihexagonal grid (Figure 2.1). In this subsection, their coordinate systems are compared to the new coordinate system.

Theorem 2.4 *The triplets containing only integers such that their sum equals to* 0 *represent exactly the hexagonal grid* (*one-plane triangular grid*).

Proof. *See Figure* 1.2*b for the points of this grid. One may check that exactly those points are addressed with* 0*-sum integer triplets for which the Cartesian coordinate pair is described:*

$H = \{(x, y) \mid x = \frac{\sqrt{3}}{2}m,\ y = 1.5n$, *where m, n are integers such that* $m + n$ *is even*$\}$. ■

Theorem 2.5 *The triplets containing only integers such that their sum is either* 0 *or* 1 *represent exactly the triangular grid* (*two-plane triangular grid*).

Proof. *See Figure* 1.2*d for the points of this grid. The locations of the points with* 0*-sum coordinate triplets are already known by Theorem* 2.4*. Now, we give the locations of the points addressed with* 1*-sum* (*integer*) *triplets:*

$T = \{(x, y) \mid x = \frac{\sqrt{3}}{2}m,\ y = 1.5n - 1$, *where m, n are integers such that* $m + n$ *is even*$\}$.

One can easily see that the union of these two sets (*H and T*) *of points gives back exactly the vertices of the hexagons of the figure, i.e., the coordinate system of the dual triangular grid.* ■

Theorem 2.6 *The triplets containing only integers such that their sum is either* 0 *or* ±1 *represent exactly the trihexagonal grid* (*that is the three-plane triangular grid*).

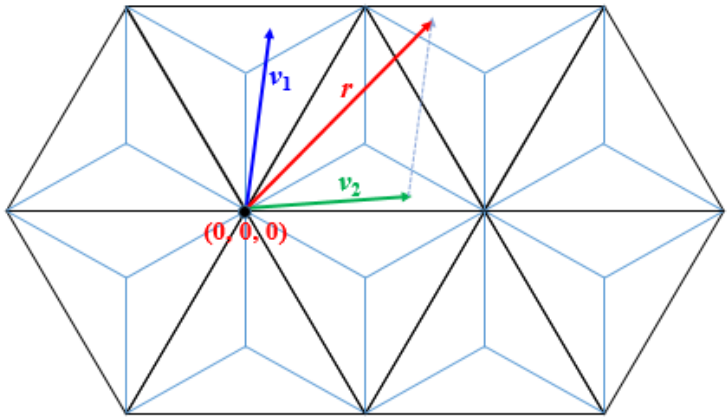
Proof. *See Figure* 2.2 *for the points of this grid. According to Theorem* 2.5*, the locations of the zero-sum and one-sum integer coordinate triplets are already shown. We need to show the locations of the points addressed with* (*integer*) *triplets that have* –1*-sum. They are:*

$M = \{(x, y) \mid x = \frac{\sqrt{3}}{2}m,\ y = 1.5n + 1$, *where m, n are integers such that* $m + n$ *is even*$\}$.

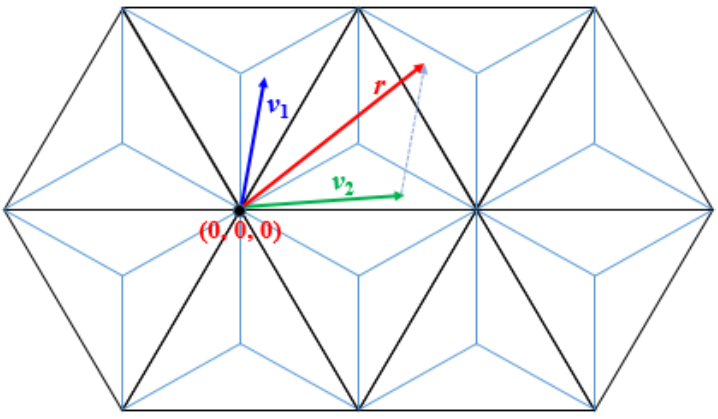
Actually, the points of this grid, $T \cup H \cup M$, are exactly those which were the base of the method of creating the coordinate system. ■

2.3.4 A Procedure for Adding Two Vectors in Ω

In contrast to the simplicity of adding two vectors in the CCS, such addition in Ω is somewhat harder, i.e., adding two vectors directly may give an improper vector to Ω (see Figures 2.11a,b). Therefore some modifications on the direct-sum (s) of the vectors are needed to get the result-vector (r), which is compatible with Ω.



(a)



(b)

Figure 2.11: (a) Consider vectors $v_1 = (0.387, –1, 0.213)$ and $v_2 = (0.677, 0, –0.477)$; both are type *B*. In this case, the direct-sum of vectors will be $s = (1.064, –1, –0.264)$, which is type *B* as well and hence is a result-vector for Ω. (b) Consider vectors $v_1 = (0.173, –0.813, 0)$ and $v_2 = (0.677, 0, –0.477)$ of types C and *B*, respectively. In this case, the direct-sum of vectors will be $s = (0.851, –0.813, –0.477)$, which is not compatible with Ω.

Consider vectors $v_1 = (i_1, j_1, k_1)$ and $v_2 = (i_2, j_2, k_2)$ of Ω with their coordinate triplets. Let the direct-sum of the vectors be $s = (i, j, k) = (i_1 + i_2, j_1 + j_2, k_1 + k_2)$. This vector may not represent any point of Ω. To describe our method and formula mathematically, we shall introduce some more notions and notations.

The coordinate values of each vector are real numbers, having some integer and fractional parts. We use the following description for them.

Each coordinate value x consists of:

1) $\langle x \rangle$: The integer part of x with sign, i.e.,

   $\langle x \rangle = sgn(x) \cdot \lfloor |x| \rfloor$, where

   $$sgn(x) = \begin{cases} +1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

   and $\lfloor x \rfloor$ is the floor function, that is applied on the absolute value of x above.

2) $\{x\}$: The fractional part of x with sign, i.e.,

   $\{x\} = x - \langle x \rangle$

3) $\lfloor x \rceil$: The absolute-rounding-up operation that rounds a number up in absolute value, such that:

   $\lfloor x \rceil = \langle x \rangle + sgn(\{x\})$.

Example 2.7

- $\langle 1.3 \rangle = 1$ and $\langle -1.3 \rangle = -1$
- $\{1.3\} = 0.3$ and $\{-1.3\} = -0.3$
- $\lfloor 1.3 \rceil = 2$ and $\lfloor -0.4 \rceil = -1$

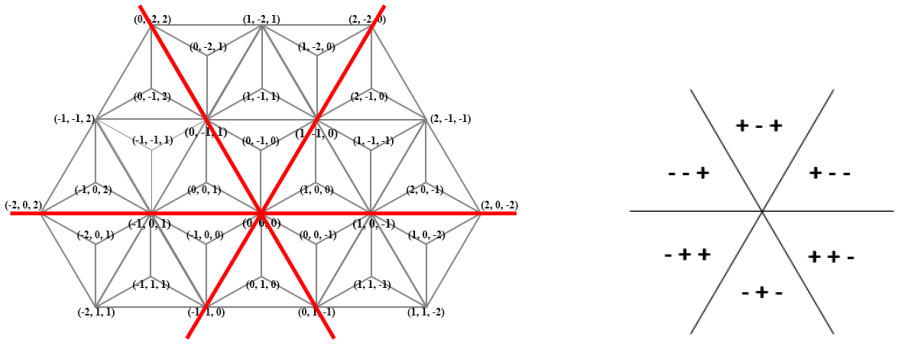
In order to produce a valid addition of two vectors in Ω, some calculations must be done. However, as a basic part of these calculations, the following two notions, the *Rsum* and the region, must be defined first:

1) $Rsum = \lfloor i \rceil + \lfloor j \rceil + \lfloor k \rceil$, where *i*, *j*, and *k* are the coordinate values of the direct-sum (*s*).
2) The region: basically, the triangular grid is partitioned into six regions based on the signs of its triplet, as it is shown in Figures 2.12a,b. Accordingly, to determine the region of result-vector we need to check the sign of each coordinate value of the direct-sum vector, such that the coordinate value is presented by a positive (+) sign if it is greater than zero and by a negative (–) sign if it is less than zero. Also, for the cases when the coordinate value is exactly zero, we examine the sum of the direct-sum vector triplet, such that, the sign is negative (–) when the sum is positive, otherwise it is positive (+).

   In short, for each coordinate value, *x*, of the direct-sum vector, the sign is described as the following:

$$x = \begin{cases} +, \text{if } x > 0, \text{or } x = 0 \text{ and } sum \leq 0 \\ -, \text{if } x < 0, \text{or } x = 0 \text{ and } sum > 0 \end{cases}$$

   where *sum* is the sum of the three coordinate values of direct-sum vector.



(a) (b)

Figure 2.12: (a) The six regions of the triangular plane. (b) The signs of the coordinate triplet for each region of the triangular plane.

For the sake of a simple illustration of all calculations, the below procedure is given. Calculations in this procedure are made upon the type of the two added vectors (the area they belong to, A, B, or C) and some features of the direct-sum $s = (i, j, k)$.

A Procedure to find the result-vector r

Input: $v_1 = (i_1, j_1, k_1)$ and $v_2 = (i_2, j_2, k_2)$, two vectors of Ω.
Output: $r = (i, j, k) = v_1 + v_2$, the result-vector in Ω.

Step1) Let:

a) direct-sum $s = (i, j, k) = (i_1 + i_2,\ j_1 + j_2, k_1 + k_2)$

b) $Rsum = \lfloor i \rceil + \lfloor j \rceil + \lfloor k \rceil$

c) Region: is the signs of the triplet of direct-sum.

Step 2) Find the type of the result-vector (r):

- If (v_1 type = v_2 type)
  Then, the type of r is determined based on the *Rsum* value and the comparison of $\{\bar{\imath}\}$, $\{\bar{\jmath}\}$, and $\{\bar{k}\}$ values. (*Explained later*)
- If (v_1 type ≠ v_2 type)
  Then, the r type is determined based on the *Rsum* value and the *minimum* (*maximum*) of ($\{\bar{\imath}\}$, $\{\bar{\jmath}\}$, and $\{\bar{k}\}$ values. (*Explained later*)

Step 3) Calculate the result-vector (r) by applying the following two steps:

a) Switch (r type):

case A: $r = (i - (\{i_1\}+\{i_2\})\ ,\ j - (\{i_1\}+\{i_2\})\ ,\ k - (\{i_1\}+\{i_2\}))$
case B: $r = (i - (\{j_1\}+\{j_2\})\ ,\ j - (\{j_1\}+\{j_2\})\ ,\ k - (\{j_1\}+\{j_2\}))$
case C: $r = (i - (\{k_1\}+\{k_2\}), j - (\{k_1\}+\{k_2\})\ ,\ k - (\{k_1\}+\{k_2\}))$

b) If $((i + j + k) > 2)$ Then, $r = (i - 1, j - 1, k - 1)$
Else If $((i + j + k) < -2)$ Then, $r = (i + 1, j + 1, k + 1)$.

This procedure contains three steps. The first two steps are used to gain only the type of the result-vector r (A, B or C), whereas the last step, the third one, is for calculating the value of the result-vector (r). The result-vector (r) will be the final correct vector, while the direct-sum (s) will be the vector that is generated by direct addition, which is in many cases, incompatible with Ω. In the next subsections, we will explain the above procedure in detail.

2.3.4.1 Step 1: Finding the Direct-sum, Rsum, and the Region

The first step of this procedure is to find the direct-sum (*s*), the *Rsum* value, and the region, as mentioned above.

For simplicity, only the region of (+, –, –) is considered in the next description, while for all other regions similar calculation methods are applied.

Now, regarding the *Rsum* value, note here that whenever adding two vectors of same type, at the region (+, –, –), the value of *Rsum* is in the set {–1, –2, 0}. More precisely, within the three coordinate values of each vector, one is an integer value; thus, actually, the fractional parts of the other two coordinate values are responsible for producing one of the three values of the set above.

For a better explanation, consider Table 2.3, where samples (*a*), (*b*), and (*c*) have additions of vectors type A. In sample (*a*), the addition of the 2^{nd} coordinate value is the only one that carries 1, therefore, *Rsum* = –1 is produced, while for sample (*b*), the addition of the 2^{nd} and 3^{rd} coordinate values carries 1, thus *Rsum* = –2, whereas the addition of sample (*c*) does not lead to carrying 1, hence *Rsum* = 0. Moreover, samples (*a*), (*b*), and (*c*) generate the three possible values (–1, –2, and 0) of *Rsum* for the addition of two vectors of type A at this region, (+, –, –). Carry in the direct addition may occur on the coordinates which have a nonzero fractional part in both vector 1 and vector 2.

In contrast, the addition of two vectors of different types at this region, (+, –, –), will produce only two possible values of *Rsum* as in the set {–1, 0}, where at most one carry could be occurred in these cases. (See Table 2.3, samples (d) and (e))

Table 2.3: Different samples to demonstrate the first step of the procedure.

	Sample (a)	***Sample*** (b)	***Sample*** (c)	***Sample*** (d)	***Sample*** (e)
***Vector*1**	(1.0,–**0.6**,–0.1)	(1.0,–**0.68**,–**0.36**)	(1.0,–0.6, –0.1)	(1.0,–0.9,–**0.4**)	(1.0,–0.9,–**0.4**)
***Vector*2**	(1.0,–**0.9**,–0.7)	(1.0,–**0.97**,–**0.73**)	(1.0,–0.2,–0.7)	(1.9,–1.0,–**0.9**)	(1.9,–1.0,–**0.5**)
direct-sum	(2.0,–1.5,–0.8)	(2.0,–1.65,–1.09)	(2.0,–0.8,–0.8)	(2.9,–1.9,–1.3)	(2.9,–1.9,–0.9)
$\lfloor i \rceil, \lfloor j \rceil, \lfloor k \rceil$	2, –2, –1	2, –2, –2	2, –1, –1	3, –2, –2	3, –2, –1
Rsum	**–1**	**–2**	**0**	**–1**	**0**
Region	(+, –, –)	(+, –, –)	(+, –, –)	(+, –, –)	(+, –, –)

2.3.4.2 Step 2: Finding the Type of the Result-Vector

As mentioned above, adding two vectors of the same type would produce any of the three possible values of *Rsum*. Only one of these values would lead directly to the type of the result-vector, while for the other two values one needs more steps to find it. However, the values of *Rsum* that lead directly to the type of the result-vector, in the region (+,–,–), are as the following:

- If (adding vectors of types (A + A)) and (*Rsum* = –1)

  Then the result-vector type is A.

- If (adding vectors of types (B + B)) and (*Rsum* = 0)

  Then the result-vector type is B.

- If (adding vectors of types (C + C)) and (*Rsum* = 0)

  Then the result-vector type is C.

In order to demonstrate the three cases above, let us consider the first one, while the other two points would have a similar demonstration.

Assume that $a = (i, j, k)$ is the midpoint of a positive triangle. Let P be any point (vector) in Area A, then based on the barycentric equation (2.1):

$$P = a + v \cdot (b - a) + u \cdot (c - a)$$

The values of $(b - a)$ and $(c - a)$ are fixed for all points in the given area A, hence we have:

$$P = \begin{pmatrix} i \\ j \\ k \end{pmatrix} + v \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} + u \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} i \\ j - v \\ k - u \end{pmatrix}$$

Now, $Rsum = \lfloor i \rceil + \lfloor j - v \rceil + \lfloor k - u \rceil$

Since the region of (+, –, –) is considered here, then:

i: is a positive integer. So, $\lfloor i \rceil$ equals to i itself.

$(j - v)$: j is a negative integer. So, $\lfloor j - v \rceil$ equals to $(j - 1)$

$(k - u)$: k is a negative integer. So, $\lfloor k - u \rceil$ equals to $(k - 1)$

Then $Rsum = i + (j - 1) + (k - 1)$, but since (i, j, k) is a positive midpoint then $i + j + k = 1$, hence, $Rsum = -1$. Therefore, two vectors of type A would produce a new vector of type A if and only if the *Rsum* value is equal to –1.

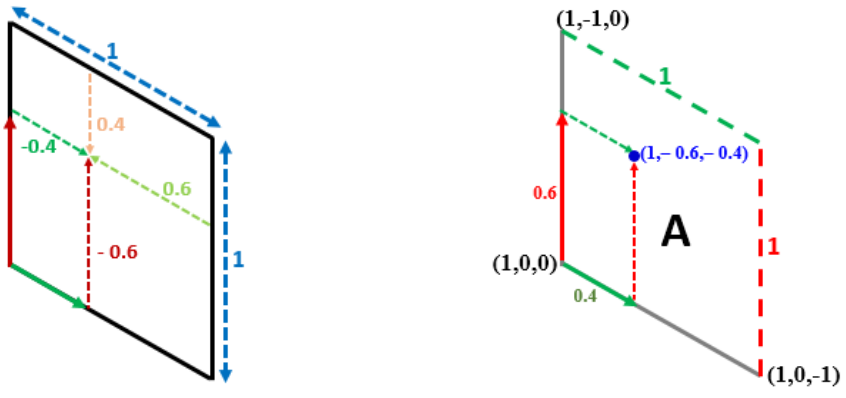
Basically, the result-vector type would be one of the three possible types (A, B or C), since one of them has been obtained for a particular *Rsum* value, the remaining two possibilities will be determined next.

Before determining the other possibilities, it is worth mentioning here that, since every direct-sum has one value among its three coordinates with a different sign, based on

the region signs, it is inconvenient to use logical comparison operations (<, >, min or max) among the values $\{i\}$, $\{j\}$, and $\{k\}$. Thus, one of the tricks used in this procedure is to unify these signs, just for the comparison purpose, such that all of them must be converted to one of the two signs, either all positive or all negative. Here, since we are describing the region (+, –, –), we will convert those with zero or positive factional value signs into negative ones, such that, for only zero or positive fractional value we subtract 1. (See Example 2.8). It is also possible to convert the negative signs to positive signs as well.

Example 2.8 If a positive fractional part is equal to 0.35, it will be converted to – 0.65, such that 0.35 – 1 = – 0.65. Also, zero as a fractional part will be converted to –1, such that 0 – 1 = –1.

This technique comes from the idea displayed in Figure 2.5, where each point inside a rhombus A, B, or C can be addressed based on the positive or on the negative midpoint. See Figure 2.13, for more clarification as well.



(a) (b)

Figure 2.13: (a) Measurements of the sides of area A and how the conversions from negative to positive fractions and vice versa would happen. (b) A coordinate triplet of a point in area A and how its fractional parts indicate its position within area A.

Therefore, whenever a comparison operation is applied, we give these notations $\{\bar{i}\}$, $\{\bar{j}\}$, and $\{\bar{k}\}$ to the fractional part of each coordinate value $\{i\}$, $\{j\}$ and $\{k\}$, respectively, to make sure that the conversion must be utilized to unify the signs if needed.

Now, *Rsum* values are in the set {–2, –1, 0} and *Rsum* = –1 leads directly to result-vector type A. If *Rsum* = 0 and $\{\bar{j}\} < \{\bar{k}\}$, then the result-vector is type B, otherwise it is type C. If *Rsum* = –2 and $\{\bar{j}\} > \{\bar{k}\}$, then the result-vector is type B, otherwise it is type C. Note here that only $\{j\}$ and $\{k\}$ are considered but not $\{i\}$ because the two added vectors are type A.

If vectors of different types were added, apart from the *Rsum* value, the maximum (or minimum) value among some values related to $\{i\}$, $\{j\}$, and $\{k\}$ will also be evaluated and thus the result-vector type would be specified. In order to specify the result-vector type in all other cases see Table 2.4.

Note here, that when applying comparison operation on equal values then selecting any type would be correct because the point is on a border or on a vertex.

Table 2.4: All conditions and rules for specifying the type of result-vector

Regions	***Vectors of type* (A + A)**
(–, –, +) (–, +, –) (+, –, +) (+, +, –)	IF (*Rsum* = 0) THEN result-vector type is A
	IF (*Rsum* = 1) & ($\{\bar{j}\} \leq \{\bar{k}\}$) THEN result-vector type is B ELSE C
	IF (*Rsum* = –1) & ($\{\bar{j}\} \geq \{\bar{k}\}$) THEN result-vector type is B ELSE C
(–, +, +)	IF (*Rsum* = 1) THEN *result-vector* type is A
	IF (*Rsum* = 0) & ($\{\bar{j}\} \geq \{\bar{k}\}$) THEN result-vector type is B ELSE C
	IF (*Rsum* = 2) & ($\{\bar{j}\} \leq \{\bar{k}\}$) THEN result-vector type is B ELSE C
(+, –, –)	IF (*Rsum* = –1) THEN result-vector type is A
	IF (*Rsum* = 0) & ($\{\bar{j}\} \leq \{\bar{k}\}$) THEN result-vector type is B ELSE C
	IF (*Rsum* = –2) & ($\{\bar{j}\} \geq \{\bar{k}\}$) THEN result-vector type is B ELSE C

Regions	***Vectors of type* (B + B)**
(–, +, +) (–, –, +) (+, –, –) (+, +, –)	IF (*Rsum* = 0) THEN result-vector type is B
	IF (*Rsum* = 1) & ($\{\bar{\imath}\} \leq \{\bar{k}\}$) THEN result-vector type is A ELSE C
	IF (*Rsum* = –1) & ($\{\bar{\imath}\} \geq \{\bar{k}\}$) THEN result-vector type is A ELSE C
(+, –, +)	IF (*Rsum* = 1) THEN result-vector type is B
	IF (*Rsum* = 0) & ($\{\bar{\imath}\} \geq \{\bar{k}\}$) THEN result-vector type is A ELSE C
	IF (*Rsum* = 2) & ($\{\bar{\imath}\} \leq \{\bar{k}\}$) THEN result-vector type is A ELSE C
(–, +, –)	IF (*Rsum* = –1) THEN result-vector type is B
	IF (*Rsum* = 0) & ($\{\bar{\imath}\} \leq \{\bar{k}\}$) THEN result-vector type is A ELSE C
	IF (*Rsum* = 2) & ($\{\bar{\imath}\} \geq \{\bar{k}\}$) THEN result-vector type is A ELSE C
Regions	***Vectors of type* (C + C)**
(–, +, +) (+, –, +) (+, –, –) (–, +, –)	IF (*Rsum* = 0) THEN result-vector is of type C
	IF (*Rsum* = 1) & ($\{\bar{\imath}\} \leq \{\bar{\jmath}\}$) THEN result-vector is of type A ELSE B
	IF (*Rsum* = –1) & ($\{\bar{\imath}\} \geq \{\bar{\jmath}\}$) THEN result-vector is of type A ELSE B
(–, –, +)	IF (*Rsum* = –1) THEN result-vector is of type C
	IF (*Rsum* = 0) & ($\{\bar{\imath}\} \leq \{\bar{\jmath}\}$) THEN result-vector is of type A ELSE B
	IF (*Rsum* = –2) & ($\{\bar{\imath}\} \geq \{\bar{\jmath}\}$) THEN result-vector is of type A ELSE B
(+, +, –)	IF (*Rsum* = 1) THEN result-vector is of type C
	IF (*Rsum* = 0) & ($\{\bar{\imath}\} \geq \{\bar{\jmath}\}$) THEN result-vector is of type A ELSE B
	IF (*Rsum* = 2) & ($\{\bar{\imath}\} \leq \{\bar{\jmath}\}$) THEN *result-vector* is of type A ELSE B
Regions	***Vectors of type* (A + B) or (A + C) or (B + C)**
(–, –, +) (–, +, –) (+, –, –)	IF (*Rsum* = 0) THEN Min($\{\bar{\imath}\},\{\bar{\jmath}\},\{\bar{k}\}$) is the result-vector type* IF (*Rsum* = –1) THEN Max($\{\bar{\imath}\},\{\bar{\jmath}\},\{\bar{k}\}$) is the result-vector type*
(+, –, +) (+, +, –) (–, +, +)	IF (*Rsum* = 0) THEN Max($\{\bar{\imath}\},\{\bar{\jmath}\},\{\bar{k}\}$) is the result-vector type* IF (*Rsum* = 1) THEN Min($\{\bar{\imath}\},\{\bar{\jmath}\},\{\bar{k}\}$)is the result-vector type*

** if $\{\bar{\imath}\}$, $\{\bar{\jmath}\}$ or $\{\bar{k}\}$ is the minimum (maximum, resp.) then the result-vector type is A, B or C respectively. If any two or three values of $\{\bar{\imath}\}$, $\{\bar{\jmath}\}$ and $\{\bar{k}\}$, are equal then the result-vector would be on a border or on a vertex which means selecting any type of them would be correct.*

2.3.4.3 Step 3: Finding Coordinate Triplet of the Result-Vector

Once the type of the result-vector has been determined, we proceed to step 3. Where part (*a*) has three possibilities based on the type of the result-vector, as the following.

- If the result-vector type is A then its coordinate triplet is:

  $r = (i - (\{i_1\}+\{i_2\})\ ,\ j - (\{i_1\}+\{i_2\})\ ,\ k - (\{i_1\}+\{i_2\}))$

  where $\{i_1\}$ and $\{i_2\}$ are the fractional parts of the first coordinate value of the first and second vectors respectively.

- If the result-vector type is B then its coordinate triplet is:

  $r = (i - (\{j_1\}+\{j_2\})\ ,\ j - (\{j_1\}+\{j_2\})\ ,\ k - (\{j_1\}+\{j_2\}))$

  where $\{j_1\}$ and $\{j_2\}$ are the fractional parts of the second coordinate value of the first and second vectors respectively.

- If the result-vector type is C then its coordinate triplet is:

  $r = (i - (\{k_1\}+\{k_2\}),\ j - (\{k_1\}+\{k_2\})\ ,\ k - (\{k_1\}+\{k_2\}))$

  where $\{k_1\}$ and $\{k_2\}$ are the fractional parts of the third coordinate value of the first and second vectors respectively.

Eventually, part (*b*) of this step, is about subtracting or adding 1 from/to each coordinate value if their sum is greater than 2 or less than –2 respectively.

For detailed example, see Table 2.5, which includes three samples to show different cases.

Table 2.5: The full procedure to compute the result-vector with different samples.

	***Sample* (*a*)**	***Sample* (*b*)**	***Sample* (*c*)**
$v_1 = (i_1, j_1, k_1)$	(1.577, **–2.0**, 0.423)	(1.155, –1.423, **0.0)**	(**1.0**, –0.735, –0.270)
$v_2 = (i_2, j_2, k_2)$	(1.005, **0.0**, –1.305)	(0.808, **0.0,** –0.808)	(**1.0**, –0.966, –0.732)
Step 1			
direct-sum	(2.582, –2.0, –0.882)	(1.963, –1.423, –0.808)	(2.0, –1.701, –1.002)
Rsum	3 + (–2) + (–1) = **0**	2 +(–2) +(–1) = **–1**	2 +(–2) +(–2) = **–2**
Region	(+, –, –)	(+, –, –)	(+, –, –)
Step 2			
Rule 1 and 2	If (*Rsum* = 0) Then B	If (*Rsum* = –1) Then Max($\{\bar{\imath}\},\{\bar{\jmath}\},\{\bar{k}\}$)	If (*Rsum* = –2) & ($\{\bar{\jmath}\} \geq \{\bar{k}\}$) Then B else C
Apply Rule 1 and 2	–	Max ((0.963–1), –0.423, –0.808) = –0.037 = $\{\bar{\imath}\}$	(–0.701) $\not\succ$ (–0.002)
Result-vector Type	**B**	**A**	**C**
Step 3			
a) *s* =	($i - (\{j_1\} + \{j_2\}$, $j - (\{j_1\} + \{j_2\})$, $k - (\{j_1\} + \{j_2\})$)	($i - (\{i_1\} + \{i_2\})$, $j - (\{i_1\} + \{i_2\})$, $k - (\{i_1\} + \{i_2\})$)	($i - (\{k_1\} + \{k_2\})$, $j - (\{k_1\} + \{k_2\})$, $k - (\{k_1\} + \{k_2\})$)
Apply a) *s* =	(2.582,**–2.0**,–0.882)	(**1.0**, –2.386, –1.771)	(3.002, –0.699, **0.0**)
b) Sum =	Sum = –0.3, then no need for addition or subtraction of 1	Sum = –3.157 < -2, then add 1 to each coordinate value	Sum = 2.303 > 2, then Subtract 1 from each coordinate value
Result-vector	(2.582, **–2.0**, –0.882)	(**2.0**, –1.386, –0.771)	(2.002, –1.699, **–1.0**)

In order to demonstrate the correctness of the procedure above, we may convert the result-vector to/from the CCS by using the way introduced in sections 2.3.1 and 2.3.2 of this chapter. Such that, two vectors of Ω would be converted to the CCS, then applying the Cartesian addition to the produced vectors of Cartesian system. Finally we convert back the CCS vector into Ω vector.

Therefore, consider *sample* (*a*) of Table 2.5 where we have:

v_1 = (1.577, –2.0, 0.423)

v_2 = (1.005, 0.0, –1.305)

By converting them into the CCS, using equation (2.2), we have:

$$\text{For } v_1 = \begin{pmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 1.577 \\ -2.0 \\ 0.423 \end{pmatrix} = \begin{pmatrix} 0.999 \\ 3.000 \end{pmatrix}$$

Then $v_1 = c_1 = (x_1, y_1)$ = (0.999, 3.000) with Cartesian coordinates, and

$$\text{for } v_2 = \begin{pmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 1.005 \\ 0.0 \\ -1.305 \end{pmatrix} = \begin{pmatrix} 2.001 \\ -0.150 \end{pmatrix}$$

Then $v_2 = c_2 = (x_2, y_2)$ = (2.001, –0.150) with Cartesian coordinates.

Now, applying the Cartesian addition to c_1 and c_2, we have:

$c = (x, y) = c_1 + c_2$ = (3.000, 2.850) of Cartesian coordinates

Finally, convert $c = (x, y)$ into Ω triplet, then we have:

Step 1: *c* belongs to the 1st quarter.

Step 2: based on Code 2, it belongs to rectangle CB.

Step 3: Area B is matched.

Thus, formulae of area B from Table 2.2 are applied in the following order:

1) $j = \langle \frac{-2y}{3} \rangle$ = –2.0

2) $i = \frac{x\sqrt{3}}{3} + y + j = \sqrt{3}$ + 2.85 – 2.0 = 2.582

3) $k = i - \frac{2x}{\sqrt{3}}$ = 2.582 – 3.464 = –0.882

Then the corresponding triplet of (3.000, 2.850) is r = (2.582, –2.000, –0.882). Which is exactly the same answer in Table 2.5 as it should be.

Chapter 3

PROPERTIES OF TRANSLATIONS ON THE TRIANGULAR GRID

The discretized translation in the triangular grid is an extension of the classical Euclidean translation with a rounding operation to the nearest grid midpoint. The translation has never been discussed individually in the cases of the square and the hexagonal grids because they are always bijective (see Figures 1.3 and 3.1). In contrast to this, it is not the case when the triangular grid is considered, it is not always bijective (see Figures 1.3 and 3.3). Even though only specific translations have been introduced, as somewhat integer translations, with the property that the gridpoints are mapped to gridpoints as in reference [43]. Therefore, it is interesting to consider translations on the triangular grid in detail and analyze how the resulted image may change.

Accordingly, this chapter is about the translations on the triangular grid. It begins by recalling some basic facts about discrete translations on the square and hexagonal grid, defining the discrete translations on the triangular grid and specifying some notations, next, a technical detail for the description of translations on the triangular grid is given. Finally, this chapter will be concluded by characterizing the translation vectors by necessary and sufficient conditions for strongly bijective, semi-bijective and non-bijective translations.

3.1 Preliminaries

3.1.1 Discrete Translations

Discrete translations are the discretized forms of the Euclidean translations. They are usually defined as a composition of the Euclidean translation τ by a given vector, applied on the points of the discrete grid *G* and a rounding operator to make it sure that the result/the image of the discrete translation belongs to the grid. Digitized translations are, then, formally, usually defined as functions $D_\tau = D \circ \tau(G)$ where both the domain and the target are the given (digital or discrete) grid *G*.

In order to analyze discretized translations, two basic concepts must be defined/recalled first, the surjective and injective translations. We generally use the set *E* which could be the Euclidean plane/space or a discrete space (e.g., the square grid).

Definition 3.1 *A translation $f: E \to E$ is surjective if $\forall i \in E$ in the target, there is at least one element $i' \in E$ in the domain, such that $f(i') = i$.*

Definition 3.2 *A translation $f: E \to E$ is injective, if $\forall a, b \in E$ in the domain, whenever $f(a) = f(b)$ then a = b. Formally:*

$$\forall a, b \in E, \qquad f(a) = f(b) \Longrightarrow a = b.$$

Based on them, the bijective translations are defined as follows.

Definition 3.3 *A translation is bijective if it is both injective and surjective translation*.

It is well known and one can easily check that every translation is bijective on the square and the hexagonal grid (see Figure 3.1, for examples).

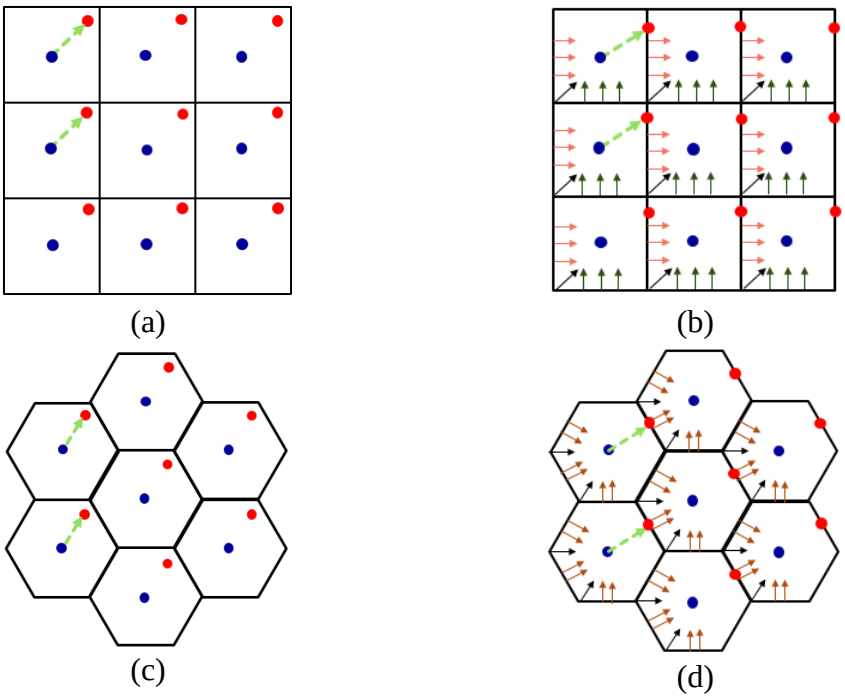


Figure 3.1: A translation by the vector represented by the broken arrow in the square (a) and in the hexagonal (c) grid. The centers (blue points) represent the original gridpoints, while the red ones are the translated ones. In (b) and (d), is also to show how to deal with points on the edges and on the corners of square and hexagonal grids, respectively.

The translation by a vector $t(t_x, t_y) \in \mathbb{R}^2$, in the two-dimensional Euclidean space, i.e., in the plane is the function $f : \mathbb{R}^2 \to \mathbb{R}^2$ such that $f(x, y) = (x + t_x, y + t_y)$. On the square grid, discrete translations are defined analogously, but changing the domain and the target of the function f to $\mathbb{Z}^2$. Moreover, usually not only integer translation vectors are allowed, but any vector $t(t_x, t_y) \in \mathbb{R}^2$, and, then a rounding operator is used for both coordinates to assign the closest gridpoint to the resulted point. Analogously, $(t_x, t_y) = (x_0, y_0) + (x_1, y_1)$ where (x_0, y_0) is the integer vector to the closest gridpoint to t, and (x_1, y_1) is the fractional vector within the grid-square where $-0.5 \leq x_1, y_1 < 0.5$. Then the rounding operator will be defined by the help of the floor function:

$$\left(\left\lfloor x_1 + \frac{1}{2} \right\rfloor, \left\lfloor y_1 + \frac{1}{2} \right\rfloor\right).$$

This can also be seen as, based on the square tessellation of the grid, to assign the midpoint of that pixel to the translated point for which pixel the point belongs. For those translated points that are on the borders of a square or a hexagonal tile in the grid, the rounding operation is systematic and obvious (see Figure 3.1b,d).

3.1.2 Digitized Translations on the Triangular Grid

In order to explain the scientific part of the triangular grid, some basic measures of triangles of the grid will be reserved as in chapter 2, while some other notations would be changed for simplicity. For example, in this chapter we will call "even △" and "odd ▽" triangles referring to the orientations of the triangles, instead of "positive △" and "negative ▽" triangles as in chapter 2. The side-length and the height of each triangle are set to $\sqrt{3}$ and 1.5, respectively, as in chapter 2. While the origin of the triangular grid will be set to the midpoint of the "even △" triangle. (See Figure 3.2)

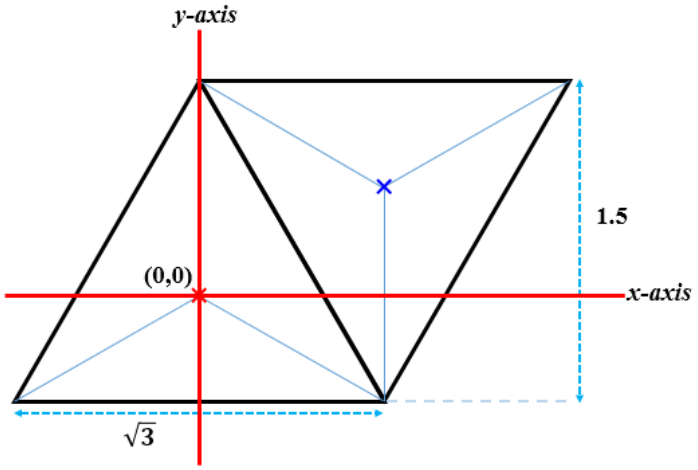


Figure 3.2: Side-length and height of triangle pixels at the triangular grid. The midpoints of the pixels are also marked.

The two marked points in Figure 3.2 are representing the midpoints of the triangles: the red point is the midpoint of the even triangle and the blue point is the midpoint of the odd one. Note that the closest neighbors of an even triangle are odd ones, and vice versa. Further, we simply use the term "neighbors" for closest neighbors. As a

consequence, based on the triangular measurements, the distance of the closest midpoint to the even midpoint would be equal to 1.

The sets of the odd (O) and the even (E) triangles (their midpoints) of the triangular grid can be described, in the Cartesian coordinate system by the two sets defined in equations (3.1) and (3.2), respectively.

$$O = \left\{ (a,b) \;\middle|\; a = \frac{m\sqrt{3}}{2}, b = \frac{3n-1}{2}, m,n \in \mathbb{Z}, m+n \text{ is odd} \right\} \tag{3.1}$$

$$E = \left\{ (a,b) \;\middle|\; a = \frac{m\sqrt{3}}{2}, b = \frac{3n-2}{2}, m,n \in \mathbb{Z}, m+n \text{ is even} \right\} \tag{3.2}$$

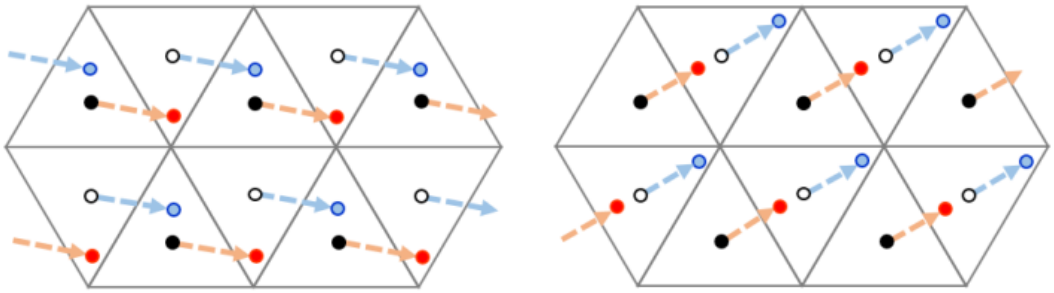
The translations on the triangular grid could be bijective or non-bijective, therefore, these concepts will be investigated and characterized in details in the next sections. Now we present digitized translations on the triangular grid in a formal way.

Let G denote the set of coordinate pairs of the digital plane corresponding to the midpoints of the triangle pixels. Let τ be a translation on G by an arbitrary 2D vector. Generally, $\tau(G) \nsubseteq G$, Therefore to define digitized translation that maps $\tau(G)$ to G, the result of $\tau(G)$ is combined with the digitization operation D. $D: \tau(G) \to G$. Thus, digitized translation is defined as $D_\tau = D \circ \tau(G)$, as usual.

Definition 3.4 *Let D_τ be a digitized translation and let $i \in G$ be the midpoint of a triangular pixel. Then the set of preimages of i with respect to D_τ is defined as $P(i) = \{x \in G | D_\tau(x) = i\}$.*

In the triangular grid G, $|P(i)| \in \{0,1,2\}$. Points q and p can be preimages of the same point i only if the distance between q and p is 1. The non-injective (not one-to-one) and non-surjective (not onto) digitized translations occur when $P(i) = 2$ or 0 for some point i, respectively.

Accordingly, a non-bijective translation in the triangular grid would happen when two distinct points have the same image or no images at all correspond to a pixel (see Figure 3.3b). As one can observe on the figure, we have both bijective and non-bijective translations on the triangular grid depending on the value of the translation vector.



(a) (b)

Figure 3.3: A bijective and a non-bijective translation (a) and (b), respectively. The translation vector is shown. In the case of non-bijective translation, two distinct points have the same image and there are pixels that do not correspond to any original pixel.

It is worth to say that a translation, in the triangular grid, that leads to an injective case will concurrently lead to surjective one and vice versa. Therefore, having or missing one of those properties will automatically lead to having or missing the other. Thus, non-bijective translations are not injective and not surjective on the triangular grid.

In order to make our study easier and to give more comprehensive descriptions of the translations on the triangular grid, we need to examine some properties of the translation vectors.

3.1.2.1 "Integer" and "Fractional" Vectors

For any translation vector $t(t_x, t_y) \in \mathbb{R}^2$ we draw it and consider it in a way that we fix its starting point to the grid origin, which is an even midpoint. Then, moreover, we write t as a sum of two vectors, the so-called integer, and fractional vectors making some analogy to the traditional grid case. The integer vector will start at the grid origin and ends at the closest even midpoint (technical details about this are discussed in the next subsection) to the endpoint of the original translation vector t. On the other hand, the fractional vector will start from the endpoint of the integer vector and ends at the endpoint of the original translation vector t.

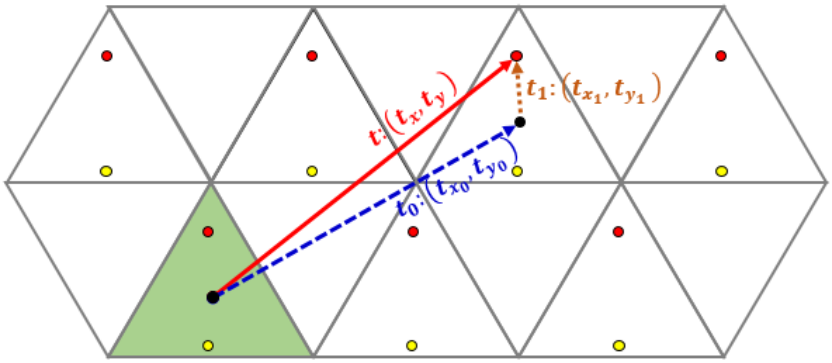
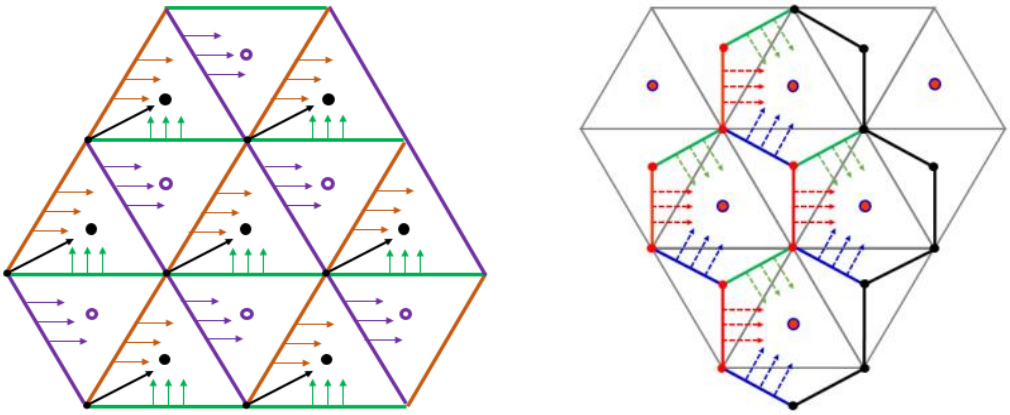


Figure 3.4: A translation vector t is considered as the sum of two vectors, t_0 is the "integer" vector, and t_1 is the "fractional" vector.

For illustration, consider Figure 3.4, where t is the original translation vector, and $t_0 = (t_{x_0}, t_{y_0})$, shown with the longer broken arrow, is the integer translation vector which has its endpoint at the nearest even midpoint to the endpoint of t. While t_1, the other broken arrow in the figure, represents the fractional vector that starts at the endpoint of t_0 and ends at the endpoint of t. We can write the translation vector t

as: $(t_x, t_y) = (t_{x_0}, t_{y_0}) + (t_{x_1}, t_{y_1})$, where (t_{x_0}, t_{y_0}) is the integer part of the translation vector and (t_{x_1}, t_{y_1}) is the fractional part of the translation vector. The integer vector maps the grid into itself. Consequently, the fractional part of the translation vector, t_1 would give the same type (i.e., bijective or not) of translation as the original translation vector t would give.

Therefore, from now on, instead of any translation vectors, we will analyze mainly its fractional part t_1 (see Figure 3.4).



(a) (b)

Figure 3.5: Rounding points of the plane to pixel midpoints (a) and to even midpoints (b).

3.1.2.2 Rounding the Border Points

In the previous section, it was written that we use the closest midpoint of an even pixel, however, geometrically, it may not be a uniquely defined point. Therefore, in this subsection, we will describe and discuss the mapping (rounding) the points of the plane to their closest midpoints. Especially, those points are in our interests that are on the borders of the triangle pixels. Obviously, the points which are not on the border could be rounded simply to its nearest midpoint based on measuring the shortest distance. Opposite to this, for the points on the border, we have to make some decision to force

a unique solution. We round them based on the technique shown also in Figure 3.5, and detailed below.

Let the borders of triangles take three different colors as in Figure 3.5a, *purple* (direction '\' with slope $-\sqrt{3}$), *brown* (direction '/' with slope $\sqrt{3}$) and the *green* (horizontal) border lines. The next points are listing the cases where a point on the border could be.

1) The corners (*C*) of the triangles, the crossing points of these colored lines have coordinates given in equation (3.3):

$$(C) = \left\{ (x, y) \;\middle|\; x = \frac{m\sqrt{3}}{2},\ y = \frac{3n}{2},\ m, n \in \mathbb{Z},\ m + n \text{ is even} \right\} \tag{3.3}$$

Further, a point having a coordinate pair (*x*, *y*) is lying on one of these borders in the following cases.

2) Let (*a*, *b*) be the closest even midpoint (the midpoint of the even triangle) for (*x*, *y*). The point (*x*, *y*) lies on a '/' direction, *brown* line (in Figure 3.5a) if:

$$y = \sqrt{3} \cdot (x - a) + (b + 1) \tag{3.4}$$

3) The point (*x*, *y*) lies on a '\' direction, *purple* line if:

$$y = \sqrt{3} \cdot (a - x) + (b + 1) \tag{3.5}$$

4) The point (*x*, *y*) lies on a '—' direction, i.e., horizontal, *green* border line if:

$$b - \frac{1}{2} = \text{y}. \tag{3.6}$$

Now, let us show the decision which pixel (midpoint) is assigned to the points.

1) Every corner point is mapped to the nearest even midpoint which has the maximal *x* coordinate value among the pixels sharing this corner point.

For points which are not corner points, we have the following strategy:

2) Every not corner point on the '/' direction (*brown*) border lines is mapped to the nearest even midpoint.

3) Every not corner point on the '\' direction (*purple*) border lines is mapped to the nearest odd midpoint.

4) Every point on the horizontal (*green*) border lines that is not a corner is mapped to the nearest even midpoint.

For the sake of completeness, we also give the assignment for all other points:

5) Finally, every point (*x*, *y*) which is not on borders should be mapped to its nearest midpoint based on their distances.

Figure 3.5a summarizes the assigned triangles for the points on the border shared by more than one triangle pixel.

Finally, in a similar manner, we also assign the closest even midpoint to any point of the plane in a unique way. For points equidistant from more even midpoints, Figure 3.5b shows the assigned even midpoint.

3.2 Characterizing Bijective and Non-Bijective Translation Vectors

As we have fixed the beginning of the translation vectors to the midpoint of an even triangle (let us say, to the origin), we can characterize the vectors by the position of their endpoints. Remember, that using an integer translation vector, the grid is mapped exactly to itself [42, 43], thus we are interested to analyze only the possible fractional parts of the translation vectors. In our description, we use various regions meaning that if the endpoint of the fractional vector lies in the given region, then the translation with the given vector has some specific property. In the next subsections, we give our main results by characterizing bijective and non-bijective translations. Moreover, we show two different types of bijective translations.

In the following subsections, we detail the properties of the translations based on the fractional part of their vector. We are willing to show that in Figure 3.6 the shaded regions represent the translations with the following property: the translation is a non-bijective translation if and only if the fractional part of the translation vector ends in the regions N_i (green) where i =1..6; whereas the translation is bijective if and only if the fractional part of the translation vector ends in any of B_e (the yellow) or B_{o_i} (the blue) regions. There is no other case. Therefore, translation by a specific vector (see, e.g., the red arrows in Figure 3.3) would lead to bijective transformation (e.g., Figure 3.3a) or non-bijective transformation (e.g., Figure 3.3b) based on the position of its endpoint.

3.2.1 Vectors of Bijective Translations

First, the bijective translations are considered. Let us analyze translation vectors with the fractional part ending in the regions B. There are two groups of the B regions: B_e and B_o based on their locations in the even triangle $\triangle$, or in an odd triangle $\triangledown$,

respectively. These bijective translation regions are described as follows. In an even triangle, consider a regular hexagon B_e with a side length of $\sqrt{3}/3$ and its center point is the even midpoint. While in the odd triangles, consider the six obtuse-angled triangles B_{o_i} (i = 1..6) as shown in Figure 3.6. Instead of proving the bijectivity of these cases with common proof here, we show that the two types of regions, although in both cases the translations are bijective, have very different behavior. We define and differentiate strongly and semi-bijective translations:

Definition 3.5 *Let a be the midpoint of a triangle and a_i be the midpoints of its neighbors where i = 1..3. Let a and a_i be mapped to $D_\tau(a)$ and $D_\tau(a_i)$, respectively (where $D_\tau(a)$ and $D_\tau(a_i)$ are the digitized translated midpoints of a and its neighbors a_i). If the neighbor relations between a and a_i are kept preserved after the translation, i.e., if $D_\tau(a)$ and $D_\tau(a_i)$ are also neighbors for each i = 1..3, then the translation is strongly bijective.*

Definition 3.6 *Let a be the midpoint of a triangle and a_i be the midpoints of its neighbors where i = 1..3. Let a and a_i be mapped to $D_\tau(a)$ and $D_\tau(a_i)$, respectively. If the translation is bijective, but the neighborhood is not preserved, (i.e., there is a neighbor a_i of a such that $D_\tau(a)$ and $D_\tau(a_i)$ are not neighbors), then the translation is called semi-bijective.*

More precise descriptions and characterizations of both categories will be given in the next two sub-sections.

3.2.1.1 Characterizing Strongly Bijective Translations

Proposition 3.1 *A translation is strongly bijective if and only if it preserves the parity of the points, i.e., the image of an even point is even and the image of an odd point is odd.*

It is clear that if an even point is mapped to even point then all even points are mapped to even points, and a similar statement is fulfilled for the odd points. Now let us prove Proposition 3.1.

Proof. *If a translation maps even points to even points and odd points to odd points, it is clear that two neighbor points (from which one must be odd and the other must be even) cannot be mapped to the same triangle. Thus the translation is bijective. Moreover, the neighbor relation is obviously also preserved. (See, e.g. Figure* 3.7*; more explanation is given in the next proposition.)*

In the other direction, if an even pixel is translated to an odd pixel or an odd pixel is translated to an even pixel, then there are two cases. (See e.g. Figures 3.3 *and* 3.9*) In the first case, only one type of pixels is mapped to the opposite type of pixels. More precisely: Either even pixels are mapped to odd pixels and also the odd pixels are mapped to odd pixels, or both the even and odd pixels are mapped to even pixels. Clearly, in this case, the translation is not bijective. The second case is when each pixel is mapped to an opposite type pixel. However, in this case, because of the different (i.e. opposite) directions of the neighbors of the different type points, it is easy to find neighbor points such that their images are not neighbors.* (*More details are shown in the next subsections.*) ■

We continue this subsection by defining formally the region B_e. By equation (3.7) we do it as a union of three smaller pairwise disjoint regions. Notice that in some conditions sharp inequality is used while in some other equality is also allowed. (See Figure 3.6)

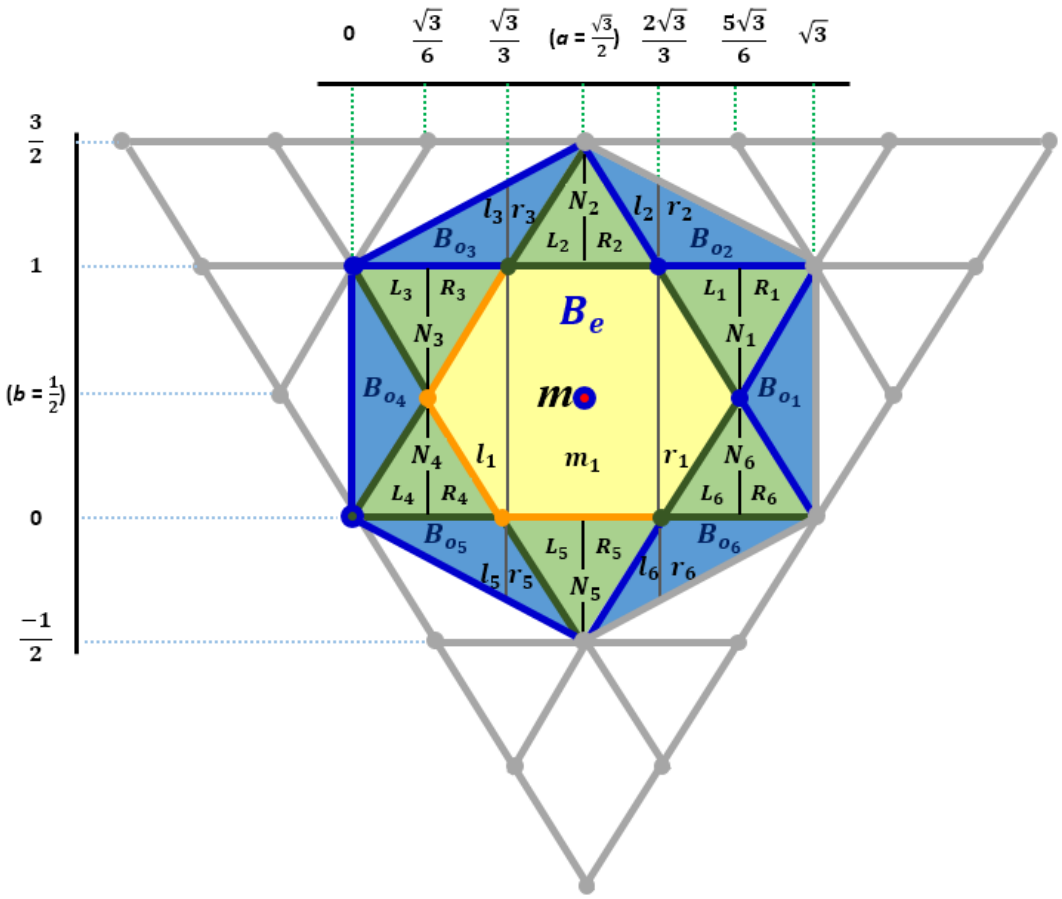


Figure 3.6: An even pixel with its three closest neighbors. The hexagon region B_e (in yellow color) with its orange borders, are referred to the strongly-bijective translation region, whereas the six obtuse-angle triangles B_{o_i}, where i = 1..6 (in blue color with its dark blue borders) are referred to the semi-bijective translation regions. The six equilateral triangles N_i, where i = 1..6 (in green color with its dark green borders) are referred to the non-bijective translation regions (the starting point of the fractional part of the translation vector is at the even midpoint (m)).

$\boldsymbol{B_e} = l_1 \cup m_1 \cup r_1$, where: (3.7)

$$l_1 = \left\{(x,y)\middle| a - \frac{\sqrt{3}}{3} \le x < a - \frac{\sqrt{3}}{6}, \sqrt{3}(a-x) - (1-b) \le y \le \sqrt{3}(x-a) + (1+b)\right\}$$

$$m_1 = \left\{(x,y)\middle| a - \frac{\sqrt{3}}{6} \le x < a + \frac{\sqrt{3}}{6}, b - \frac{1}{2} \le y < b + \frac{1}{2}\right\}$$

$$r_1 = \left\{(x,y)\middle| a + \frac{\sqrt{3}}{6} \le x \le a + \frac{\sqrt{3}}{3}, \sqrt{3}(x-a) - (1-b) < y < \sqrt{3}(a-x) + (1+b)\right\}$$

Theorem 3.1 *A translation with vector t = (t_x, t_y) is a strongly bijective if and only if* (t_{x_1}, t_{y_1}) *ends at the region* B_e*, where* $(t_x, t_y) = (t_{x_0}, t_{y_0}) + (t_{x_1}, t_{y_1})$ *with integer vector* (t_{x_0}, t_{y_0}) *and fractional vector* $t_1 = (t_{x_1}, t_{y_1})$.

Proof. *Based on Proposition* 3.1*, we need to prove that translation with fractional vector belonging to region* B_e *preserve the parity of the pixels. Without loss of generality, consider* B_e *(the yellow) and* B'_e *(the blue) isometric regular hexagons within the area of the even and the odd triangles, respectively, in Figure* 3.7*. Here, m (the red) and n (the yellow) points are the midpoints of the even and odd triangles, respectively.*

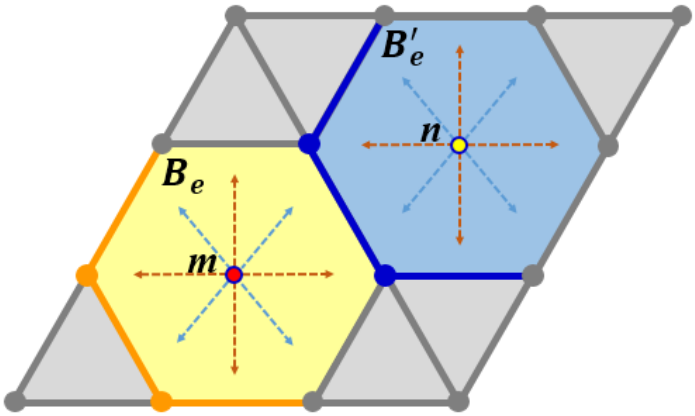


Figure 3.7: Any translation vector that starts at the even midpoint (*m*) and ends within the hexagonal region (B_e) will produce a strongly-bijective translation. The orange and blue colored borders belong to B_e and B'_e regions respectively, while the gray colored borders belong to other regions.

If t is a translation vector with fractional part t_1 in B_e, then the midpoint of an even pixel m (e.g., the red point) is translated into the region B_e around a midpoint of an even pixel as it is shown in Figure 3.7. *By our rounding operator, one can see that the parity of each even pixel is preserved, the points in B_e belong to an even triangle.*

On the other hand, the translation of an odd pixel by a vector with fractional part t_1 in B_e implies that the midpoint of the odd pixel n (e.g., the yellow point) is translated to a point belonging to region B'_e (*Figure* 3.7). *However, by the rounding operator, it is clear that these points are mapped to n* (*the yellow point*), *i.e., to the midpoint of an odd pixel. Therefore, the parity of the original pixel is exactly the same as the parity of its image after the translation. The proof is finished.* ■

3.2.1.2 Characterizing Semi-Bijective Translations

We start this subsection by describing the regions where the fractional part of the translation vector is ending in this case, then we formally state our results.

The regions named B_{o_i} (where i = 1..6) in Figure 3.6 are described by sets defined in equations (3.8-3.13). As we will show these regions represent the semi-bijective translations. Notice that in some conditions sharp equality and inequality are used to deal correctly with the border points.

$$\boldsymbol{B_{o_1}} = \left\{(x,y)\middle| a+\frac{\sqrt{3}}{3} \le x < a+\frac{\sqrt{3}}{2}, \sqrt{3}(a-x)+(1+b) \le y \le \sqrt{3}(x-a)-(1-b)\right\} \tag{3.8}$$

$\boldsymbol{B_{o_2}} = l_2 \cup r_2$, where: (3.9)

$$l_2 = \left\{(x,y)\middle| a < x < a + \frac{\sqrt{3}}{3},\ \sqrt{3}(a-x) + (1+b) \le y < \frac{\sqrt{3}}{3}(a-x) + (1+b)\right\}$$

$$r_2 = \left\{(x,y)\middle| a + \frac{\sqrt{3}}{6} \le x \le a + \frac{\sqrt{3}}{2}, b + \frac{1}{2} \le y < \frac{\sqrt{3}}{3}(a-x) + (1+b)\right\}$$

$\boldsymbol{B_{o_3}} = l_3 \cup r_3$, where: (3.10)

$$l_3 = \left\{(x,y)\middle| a - \frac{\sqrt{3}}{2} \le x < a - \frac{\sqrt{3}}{6}, b + \frac{1}{2} \le y \le \frac{\sqrt{3}}{3}(x-\ a) + (1+b)\right\}$$

$$r_3 = \left\{(x,y)\middle| a - \frac{\sqrt{3}}{6} < x < a, \sqrt{3}(x-a) + (1+b) < y \le \frac{\sqrt{3}}{3}(x-\ a) + (1+b)\right\}$$

$$\boldsymbol{B_{o_4}} = \left\{(x,y)\middle| a - \frac{\sqrt{3}}{2} \le x \le a - \frac{\sqrt{3}}{3}, \sqrt{3}(x-a) + (1+b) < y < \sqrt{3}(x-a) - (1-b)\right\}$$

(3.11)

$\boldsymbol{B_{o_5}} = l_5 \cup r_5$, where: (3.12)

$$l_5 = \left\{(x,y)\middle| a - \frac{\sqrt{3}}{2} < x \le a - \frac{\sqrt{3}}{6}, \frac{\sqrt{3}}{3}\ (a-x) + (b-1) \le y < b - \frac{1}{2}\right\}$$

$$r_5 = \left\{(x,y)\middle| a - \frac{\sqrt{3}}{6} \le x \le a, \frac{\sqrt{3}}{3}\ (a-x) + (b-1) \le y < \sqrt{3}(a-x) - (1-b)\right\}$$

$\boldsymbol{B_{o_6}} = l_6 \cup r_6$, where: (3.13)

$$l_6 = \left\{(x,y)\middle| a \le x < a + \frac{\sqrt{3}}{6}, \frac{\sqrt{3}}{3}\ (x-a) + (b-1) \le y < \sqrt{3}(x-a) - (1-b)\right\}$$

$$r_6 = \left\{(x,y)\middle| a + \frac{\sqrt{3}}{6} < x < a + \frac{\sqrt{3}}{2}, \frac{\sqrt{3}}{3}\ (x-a) + (b-1) + b \le y < b - \frac{1}{2}\right\}$$

Now we state and prove that any translation with fractional vector belonging to a region B_{o_i} switches the parity of the pixels.

Proposition 3.2 *A translation with vector* $t = (t_x, t_y) = t_0 + t_1$ *is mapping every pixel to an opposite type pixel if and only if the fractional vector* $t_1 = (t_{x_1}, t_{y_1})$ *ends in a region* B_{o_i} *where* $i = 1..6$.

Proof. *Let us consider the isometric obtuse angled isosceles triangles* B_{o_1} *and* B'_{o_1} *within the area of the odd and the even triangles, respectively, in Figure* 3.8. *Let m* (*the red point in Figure* 3.8) *denote the midpoint of the corresponding even triangle and n (the yellow point) denote the midpoint of the odd triangle, respectively. If t is a translation vector with fractional part* t_1 *belonging to region* B_{o_1}, *then the midpoint of an even pixel m (e.g., the red point) is translated into the region* B_{o_1} *inside an odd pixel as it is shown in Figure* 3.8. *By our rounding operator, one can see that the parity of each even pixel is changed to odd, the points in* B_{o_1} *belong to an odd triangle.*

On the other hand, the translation of an odd pixel by a vector with fractional part t_1 *belonging to* B_{o_1} *implies that the midpoint of the odd pixel n* (*e.g., the yellow point*) *is translated to a point belonging to region* B'_{o_1} (*Figure* 3.8). *However,* (*by applying the rounding operator on the edges*), *it is clear that these points are mapped to the midpoint of an even pixel* (m'). *Therefore, the parity of the original pixel is opposite to the parity of its image after the translation.*

Having the even midpoint (*m*) *translated to* $B_{o_2}, B_{o_3}, B_{o_4}, B_{o_5}, and\ B_{o_6}$ (*Figure* 3.8) *the proof goes in a similar manner.* ■

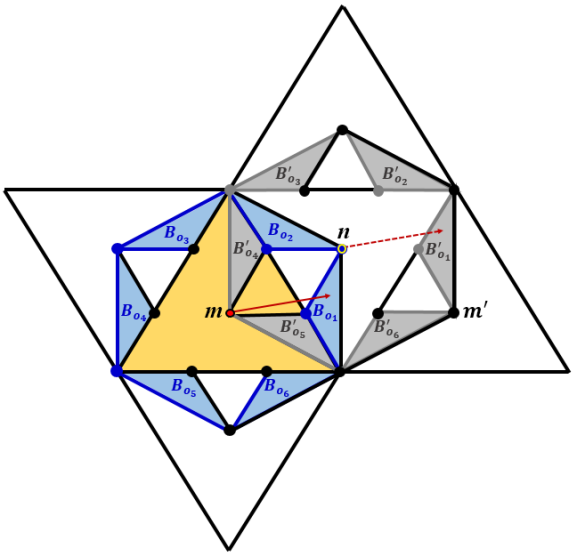


Figure 3.8: Translations with fractional vectors that start at the even midpoint (m) and end at regions B_{o_i} (where i = 1..6, the blue colored regions with their dark blue colored borders) will produce semi-bijective translations.

Theorem 3.2 *A translation with vector* $t = (t_x, t_y) = t_0 + t_1$ *is a semi-bijective mapping if and only if the fractional vector* $t_1 = (t_{x_1}, t_{y_1})$ *ends in a region* B_{o_i} *where* i = 1..6.

Proof. *We have already shown that translations that preserve the parity of the pixels are strongly bijective. It is clear, that if an even point is mapped to an even point, then all even points are mapped to even points, and if an even point is mapped to an odd point then all even points are mapped to odd points. If both even and odd points are mapped to the same type of points, then the translation is not bijective, since no point will be mapped to the other type of points. What is remained to show, that if odd points are translated to even points and even points are translated to odd points, then the neighborhood structure is lost by the translation. Let us consider, first, the regions* B_{o_1} *and* B'_{o_1}. *As one can observe in Figure* 3.9, *the neighbors of a pixel are not mapped to the neighbors of the pixel obtained by the translation.*

The proof goes analogously for the other five regions. ■

Summarizing the cases of bijective translations, the B_e (the yellow) region refers to the strongly bijective transitions, while B_{o_i}, where i = 1..6, (the blue) regions specify the semi-bijective translations (in Figure 3.6). Based on Proposition 3.2 and Theorem 3.2, we can say that a translation is semi-bijective if and only if it is a bijective translation and it maps the elements of O (i.e., the odd points) to E (i.e., even points) and vice versa.

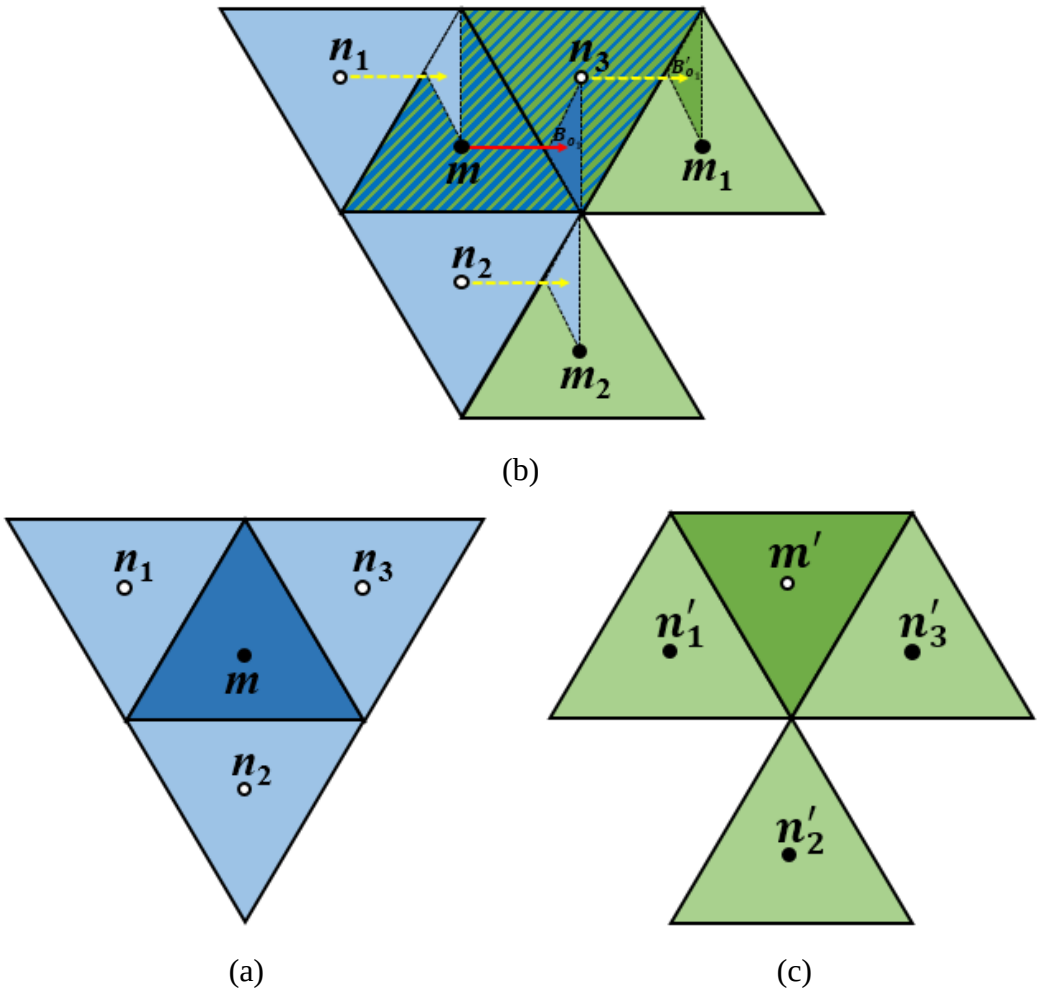


Figure 3.9: (a) An even pixel and its neighbors, the three odd pixels, before translation. (b) A translation by a vector that belongs to the semi-bijective region B_{o_1}. (c) The result of the translation: $m', n'_1, n'_2,$ and n'_3 are the images of $m, n_1, n_2,$ and n_3, respectively. The pixels m' and n'_2 are not neighbors.

3.2.2 Characterizing the Non-Bijective Translation Vectors

Proposition 3.3 *A translation is non-bijective if and only if it maps all the pixels (both E and O) to exactly one of the sets O or E.*

Proof. *Obviously, translations are not bijective if only one of the sets O and E is used as the image of the translation. On the other hand, all translations must satisfy the condition of exactly one of Propositions* 3.1, 3.2 *or* 3.3, *thus the statement of this proposition is proven.* ■

At translations having vectors with fractional parts ending in the regions N_i where i = 1..6 (the green regions in Figure 3.6), points without pre-image (holes) and also points with two pre-images occur. Hence, these are non-bijective translations (see Figure 3.3b, for an example).

Now, in order to describe these non-bijective translations mathematically, the six equilateral triangles N_i where i = 1..6 (the green regions in Figure 3.6), will be categorized into two groups. The odd and the even groups are based on the base odd $\triangledown$ and even $\triangle$ triangles in which these green triangles are located. Remember that the translation vectors in our description start at the midpoint of an even triangle (e.g., the origin), and we deal with its fractional part.

There are three green regions N_i in each odd and even triangle, they are denoted by N_1, N_3, and N_5 for the odd triangle, and N_2, N_4, and N_6 for the even triangle (as Figure 3.6 shows both cases). To simplify their mathematical description, each of them is split into two symmetrical parts L_i and R_i, where i = 1..6, as their left and right parts (Figure 3.6). Equations (3.14 - 3.19) describe them as follows. Let (a, b) denote the coordinate

pair of the midpoint of the even pixel from where the fractional vector starts (denoted by the red circle in Figure7).

$N_1 = L_1 \cup R_1$, where: (3.14)

$$L_1 = \left\{(x,y) \left| a + \frac{\sqrt{3}}{6} \le x \le a + \frac{\sqrt{3}}{3}, \sqrt{3}(a-x) + (1+b) < y \le b + \frac{1}{2} \right.\right\}$$

$$R_1 = \left\{(x,y) \left| a + \frac{\sqrt{3}}{3} < x \le a + \frac{\sqrt{3}}{2}, \sqrt{3}(x-a) - (1-b) \le y \le b + \frac{1}{2} \right.\right\}$$

$N_2 = L_2 \cup R_2$, where: (3.15)

$$L_2 = \left\{(x,y) \left| a - \frac{\sqrt{3}}{6} < x < a, b + \frac{1}{2} < y < \sqrt{3}(x-a) + (1+b) \right.\right\}$$

$$R_2 = \left\{(x,y) \left| a \le x \le a + \frac{\sqrt{3}}{6}, b + \frac{1}{2} < y < \sqrt{3}(a-x) + (1+b) \right.\right\}$$

$N_3 = L_3 \cup R_3$, where: (3.16)

$$L_3 = \left\{(x,y) \left| a - \frac{\sqrt{3}}{2} \le x \le a - \frac{\sqrt{3}}{3}, \sqrt{3}(x-a) - (1-b) < y \le b + \frac{1}{2} \right.\right\}$$

$$R_3 = \left\{(x,y) \left| a - \frac{\sqrt{3}}{3} < x < a - \frac{\sqrt{3}}{6}, \sqrt{3}(x-a) + (1+b) < y \le b + \frac{1}{2} \right.\right\}$$

$N_4 = L_4 \cup R_4$, where: (3.17)

$$L_4 = \left\{(x,y) \left| a - \frac{\sqrt{3}}{2} < x \le a - \frac{\sqrt{3}}{3}, b - \frac{1}{2} < y < \sqrt{3}(x-a) + (1+b) \right.\right\}$$

$$R_4 = \left\{(x,y) \left| a - \frac{\sqrt{3}}{3} < x \le a - \frac{\sqrt{3}}{6}, b - \frac{1}{2} < y \le \sqrt{3}(x-a) - (1-b) \right.\right\}$$

$N_5 = L_5 \cup R_5$, where: (3.18)

$$L_5 = \left\{(x,y) \,\middle|\, a - \frac{\sqrt{3}}{6} \le x \le a, \sqrt{3}(a-x) - (1-b) < y \le b - \frac{1}{2}\right\}$$

$$R_5 = \left\{(x,y) \,\middle|\, a < x < a + \frac{\sqrt{3}}{6}, \sqrt{3}(x-a) - (1-b) \le y \le b - \frac{1}{2}\right\}$$

$N_6 = L_6 \cup R_6$, where: (3.19)

$$L_6 = \left\{(x,y) \middle| a + \frac{\sqrt{3}}{6} < x \le a + \frac{\sqrt{3}}{3}, b - \frac{1}{2} < y < \sqrt{3}(x-a) - (1-b)\right\}$$

$$R_6 = \left\{(x,y) \middle| a + \frac{\sqrt{3}}{3} < x < a + \frac{\sqrt{3}}{2}, b - \frac{1}{2} < y \le \sqrt{3}(a-x) + (1+b)\right\}$$

Theorem 3.3 *A translation with vector t(x, y) is a non-bijective mapping if and only if* $t(x, y) = t_0(x_0, y_0) + t_1(x_1, y_1)$ *where* t_0 *is the integer vector and* t_1 *is the fractional vector of the translation such that this latter one starts at the endpoint of* t_0 *and ends at a region* N_i *where* $i = 1..6$.

Proof. *Based on Proposition* 3.3*, we need to prove that translation with fractional vector* t_1 *belonging to region* N_i *where* $i = 1..6$*, will map every pixel to the same type of pixels. Without loss of generality, consider the regions* N_1 (*green*) *and* N_1' (*gray*) *in Figure* 3.10*. They are, in fact, isometric regular triangles within the area of an odd triangle. Points m* (*red*) *and n* (*yellow*) *denote the midpoints of the corresponding even and odd triangles, respectively.*

If t is a translation vector with fractional part t_1 *in* N_1*, then the midpoint of an even pixel* (*e.g., the red point m*) *is translated into the region* N_1 *around n, the midpoint of an odd pixel as it is shown in Figure* 3.10*. By our rounding operator, one can see that*

the parity of each even pixel is not preserved since the points in N_1 belong to an odd triangle. On the other hand, the translation of an odd pixel by a vector t with fractional part t_1 in N_1 implies that the midpoint of the odd pixel n (e.g., the yellow point in Figure 3.10) *is translated to a point belonging to region N_1'. However, by the rounding operator, it is clear that these points are mapped again to the midpoint of the odd pixel, i.e. the point n. Therefore, both even and odd points are mapped to odd points in this case. The proof is similar for other regions, in fact, for regions N_1, N_3, and N_5 the translation maps every pixel to odd and for regions N_2, N_4, and N_6 it maps every point to even pixels.* ■

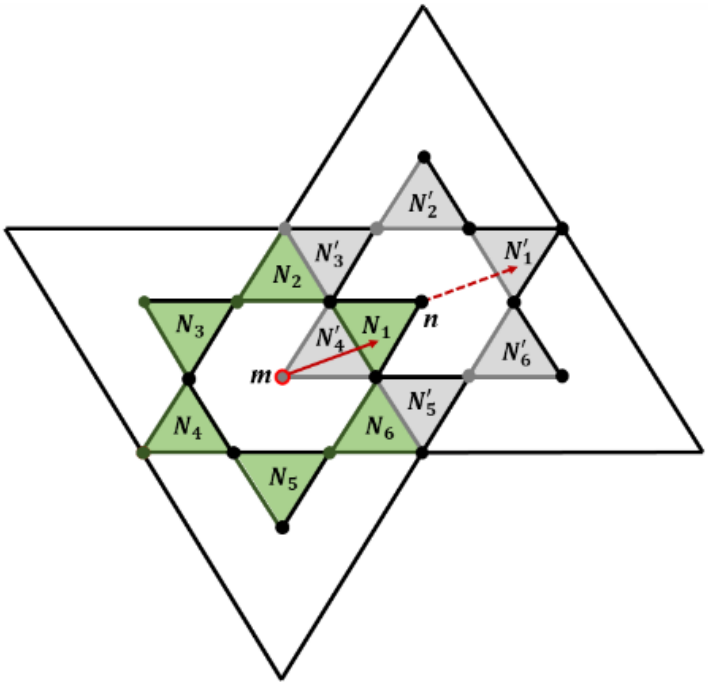


Figure 3.10: A translation to a non-bijective regions N_i. Image of an even and an odd pixel (with the corresponding regions N_i', where i = 1..6) are shown.

Chapter 4

CONCLUSION

In section 2.2, we presented the continuous coordinate system, which is an extension of some previously known discrete coordinate systems, e.g., of the symmetric coordinate frame for the triangular grid. This extension is needed and helpful for various applications, where the grid points are not necessarily mapped to grid points, e.g., arbitrary angled rotations, zooming or interpolation of images. We should also mention translations of images [48] since the triangular grid is not a point lattice. Mathematical morphology operators are also based on local translations [25, 49], thus our coordinate system provides a new tool for that research direction as well. The proposed coordinate system addresses each point of the 2D (triangular) plane (see Figure 4.1). In the subsections of section 2.2, a conversion to/from the Cartesian coordinate system and addition two vectors in the proposed coordinate system is provided, these mappings are inverses of each other. Thus the new coordinate system is ready to use in various applications including those operations that do not necessarily map the grid to itself.

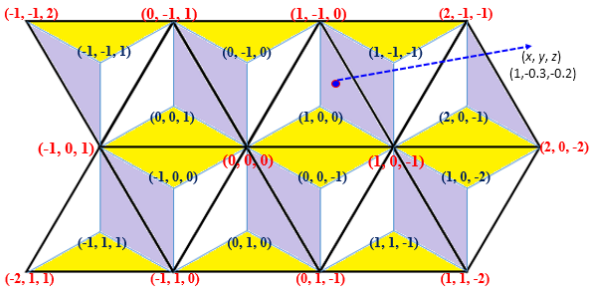


Figure 4.1: An example for addressing a point in the triangular plane by the continuous coordinate system

Accordingly, transformations of digital images are very frequent applications. The simplest transformations are the isometric ones, including translations, rotations, and mirroring. In chapter 3, translations on the triangular grid are investigated and analyzed. Although these geometrical transformations are the simplest ones, their discretized versions are not trivial on the triangular grid. As we have seen, since this grid is not a point lattice, some of the discretized translations are not bijective. Thus, one needs to be careful when applying arbitrary translations to images on the triangular grid. We note here that the technique to determine whether a translation is bijective or not is based on the redigitization of the translated image, and thus, it is somewhat similar to [50] and also to the approach counting the number of digitizations of a disk in [51].

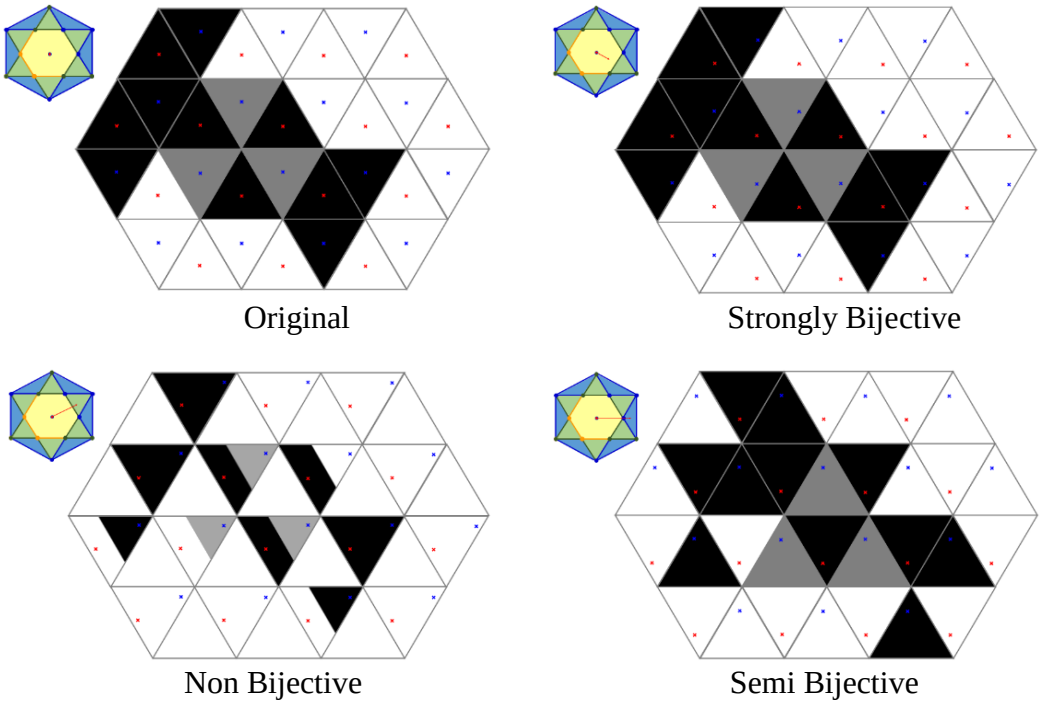


Figure 4.2: The three types of translations on the triangular grid depending on the translation vector.

It is well known that digital rotations are usually not bijective on any grids, however, the translations are bijective on point lattices. In chapter 3, we have discovered various cases of translations on the triangular plane and a full characterization of them is given to three categories (see Figure 4.2).

REFERENCES

[1] Coxeter, H. S. M. (1969). *Introduction to geometry* (Vol. 136). New York: Wiley.

[2] Rosenfeld, A., & Pfaltz, J. L. (1968). Distance functions on digital pictures. *Pattern recognition*, *1*(1), 33-61.

[3] Das, P. P., Chakrabarti, P. P., & Chatterji, B. N. (1987). Generalized distances in digital geometry. *Information Sciences*, *42*(1), 51-67.

[4] Melter, R. A. (1991). A survey of digital metrics. *Contemporary Mathematics*, *119*, 95-106.

[5] Voss, K. (1993). *Discrete images, objects, and functions in* Z^n (Vol. 273). Berlin, Heidelberg, New York: Springer-Verlag.

[6] Klette, R., & Rosenfeld, A. (2004). *Digital geometry: Geometric methods for digital picture analysis*. Elsevier.

[7] Rosenfeld, A. (1997). *Digital geometry: Introduction and bibliography*. CITR, University of Auckland, New Zealand.

[8] Golay, M. J. (1969). Hexagonal parallel pattern transformations. *IEEE Transactions on computers*, *100*(8), 733-740.

[9] Her, I. (1993). Symmetrical coordinate frame on the hexagonal grid for computer graphics and vision. *Journal of Mechanical Design, Transactions Of the ASME, 115*(3), 447-449.

[10] Luczak, E., & Rosenfeld, A. (1976). Distance on a hexagonal grid. *IEEE Transactions on Computers*, (5), 532-533.

[11] Wüthrich, C. A., & Stucki, P. (1991). An algorithmic comparison between square-and hexagonal-based grids. *CVGIP: Graphical Models and Image Processing*, *53*(4), 324-339.

[12] Middleton, L., & Sivaswamy, J. (2006). *Hexagonal image processing: A practical approach*. Springer Science & Business Media.

[13] Yabushita, A., & Ogawa, K. (2002, November). Image reconstruction with a hexagonal grid. In *2002 IEEE Nuclear Science Symposium Conference Record* (Vol. 3, pp. 1500-1503). IEEE.

[14] Asharindavida, F., Hundewale, N., & Aljahdali, S. (2012). Study on hexagonal grid in image processing. *Proc. ICIKM*.

[15] Nagy, B. (2001). Finding shortest path with neighbourhood sequences in triangular grids. In *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat.* (pp. 55-60). IEEE.

[16] Nagy, B. (2002). Metrics based on neighbourhood sequences in triangular grids. *Pure Mathematics and Applications*, *13*(1-2), 259-274.

[17] Nagy, B. N. (2003). Shortest paths in triangular grids with neighbourhood sequences. *Journal of Computing and Information Technology*, *11*(2), 111-122.

[18] Nagy, B. (2007). Distances with neighbourhood sequences in cubic and triangular grids. *Pattern Recognition Letters*, *28*(1), 99-109.

[19] Nagy, B. (2014, May). Weighted distances on a triangular grid. In *International Workshop on Combinatorial Image Analysis* (pp. 37-50). Springer, Cham.

[20] Nagy, B., & Mir-Mohammad-Sadeghi, H. (2016, April). Digital disks by weighted distances in the triangular grid. In *International Conference on Discrete Geometry for Computer Imagery* (pp. 385-397). Springer, Cham.

[21] Mir-Mohammad-Sadeghi, H., & Nagy, B. (2017, June). On the chamfer polygons on the triangular grid. In *International Workshop on Combinatorial Image Analysis* (pp. 53-65). Springer, Cham.

[22] Kovács, G., Nagy, B., & Vizvári, B. (2017, September). An Integer Programming Approach to Characterize Digital Disks on the Triangular Grid. In *International Conference on Discrete Geometry for Computer Imagery* (pp. 94-106). Springer, Cham.

[23] Nagy, B., & Moisi, E. V. (2017). Memetic algorithms for reconstruction of binary images on triangular grids with 3 and 6 projections. *Applied Soft Computing*, *52*, 549-565.

[24] Lukić, T., & Nagy, B. (2014). Deterministic discrete tomography reconstruction by energy minimization method on the triangular grid. *Pattern Recognition Letters*, *49*, 11-16.

[25] Abdalla, M., & Nagy, B. (2016, September). Concepts of binary morphological operations dilation and erosion on the triangular grid. In *International Symposium Computational Modeling of Objects Represented in Images* (pp. 89-104). Springer, Cham.

[26] Nagy, B. (2015). Cellular topology and topological coordinate systems on the hexagonal and on the triangular grids. *Annals of Mathematics and Artificial Intelligence*, *75*(1-2), 117-134.

[27] Kardos, P., & Palágyi, K. (2016, September). Unified characterization of P-simple points in triangular, square, and hexagonal grids. In *International Symposium Computational Modeling of Objects Represented in Images* (pp. 79-88). Springer, Cham.

[28] Carr, D. B., Olsen, A. R., & White, D. (1992). Hexagon mosaic maps for display of univariate and bivariate geographical data. *Cartography and Geographic Information Systems*, *19*(4), 228-236.

[29] Sahr, K. (2011). Hexagonal discrete global grid systems for geospatial computing. *Archiwum Fotogrametrii, Kartografii i Teledetekcji*, *22*, 363-376.

[30] Sakai, K. I. (1957). Studies on competition in plants VII. Effect on competition of a varying number, of competing and non-competing individuals. *Journal of Genetics*, *55*(2), 227-234.

[31] Birch, C. P., Oom, S. P., & Beecham, J. A. (2007). Rectangular and hexagonal grids used for observation, experiment and simulation in ecology. *Ecological modelling*, *206*(3-4), 347-359.

[32] Her, I. (1995). Geometric transformations on the hexagonal grid. *IEEE Transactions on Image Processing*, *4*(9), 1213-1222.

[33] Almansa, A. (2002). *Sampling, interpolation and detection. Applications in satellite imaging* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

[34] Pluta, K., Romon, P., Kenmochi, Y., & Passat, N. (2017, September). Honeycomb geometry: Rigid motions on the hexagonal grid. In *International Conference on Discrete Geometry for Computer Imagery* (pp. 33-45). Springer, Cham.

[35] Pluta, K., Romon, P., Kenmochi, Y., & Passat, N. (2016, April). Bijective rigid motions of the 2D Cartesian grid. In *International Conference on Discrete Geometry for Computer Imagery* (pp. 359-371). Springer, Cham.

[36] Nouvel, B., & Rémila, E. (2005). Configurations induced by discrete rotations: Periodicity and quasi-periodicity properties. *Discrete Applied Mathematics*, *147*(2-3), 325-343.

[37] Andrès, E. (1992). *Discrete circles, and Discrete rotations* (Doctoral dissertation, PhD thesis, Université Louis Pasteur).

[38] Thibault, Y. (2010). *Rotations in 2D and 3D discrete spaces* (Doctoral dissertation, Université Paris-Est).

[39] Kaufman, A. (1994). Voxels as a computational representation of geometry. *The computational representation of geometry. SIGGRAPH*, *94*, 45.

[40] Pluta, K., Romon, P., Kenmochi, Y., & Passat, N. (2017). Bijective digitized rigid motions on subsets of the plane. *Journal of Mathematical Imaging and Vision*, *59*(1), 84-105.

[41] Avkan, A., Nagy, B., & Saadetoğlu, M. (2018, November). Digitized Rotations of Closest Neighborhood on the Triangular Grid. In *International Workshop on Combinatorial Image Analysis* (pp. 53-67). Springer, Cham.

[42] Nagy, B. (2005). Transformations of the triangular grid. In *GRAFGEO: Third Hungarian Conference on Computer Graphics and Geometry, Budapest, Hungary* (pp. 155-162).

[43] Nagy, B. (2009, September). Isometric transformations of the dual of the hexagonal lattice. In *2009 Proceedings of 6th International Symposium on Image and Signal Processing and Analysis* (pp. 432-437). IEEE.

[44] Kovács, G., Nagy, B., & Vizvári, B. (2017, September). Weighted Distances on the Trihexagonal Grid. In *International Conference on Discrete Geometry for Computer Imagery* (pp. 82-93). Springer, Cham.

[45] Nagy, B. (2004, October). A symmetric coordinate frame for hexagonal networks. *Theoretical Computer Science-Information Society*, *4*, 193-196.

[46] Nagy, B. (2004). Generalized triangular grids in digital geometry. *Acta Mathematica Academiae Paedagogicae Nyíregyháziensis*, *20*(1), 63-78.

[47] Skala, V. (2008). Barycentric coordinates computation in homogeneous coordinates. *Computers & Graphics*, *32*(1), 120-127.

[48] Abuhmaidan, K., & Nagy, B. (2018, February). Non-bijective translations on the triangular plane. In *2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI)* (pp. 000183-000188). IEEE.

[49] Abdalla, M., & Nagy, B. (2018). Dilation and erosion on the triangular tessellation: an independent approach. *IEEE Access*, *6*, 23108-23119.

[50] Mazo, L., & Baudrier, É. (2018). Object digitization up to a translation. *Journal of Computer and System Sciences*, *95*, 193-203.

[51] Nagy, B. (2005). An algorithm to find the number of the digitizations of discs with a fixed radius. *Electronic Notes in Discrete Mathematics*, *20*, 607-622.