Fast Fourier Transform Based Force Histogram Computation for 3D Raster Data

by
Jaspinder Kaur

A Thesis
presented to
The University of Guelph

In partial fulfilment of requirements
for the degree of
Master of Science
in
Computer Science

Guelph, Ontario, Canada
© Jaspinder Kaur, September, 2019
ABSTRACT

FAST FOURIER TRANSFORM BASED FORCE HISTOGRAM COMPUTATION FOR 3D RASTER DATA

Jaspinder Kaur
University of Guelph, 2019

Advisor:
Professor Pascal Matsakis

The force histogram is a quantitative representation of relative position between two objects. For 2D data, two algorithms are defined: a well-functioning line-based algorithm, and Fast Fourier Transform (FFT) based algorithm that has a high computational cost. The line-based algorithm has previously been extended to the 3D case, but found to be unstable, and affected by a variety of factors.

This thesis presents an extension of the FFT-based algorithm to the 3D case along with an analysis that demonstrates that, with the exception of a few special cases, the computational time of the 3D FFT-based algorithm is less than the line-based version. In addition, the results included here shown that the FFT-based algorithm is independent of the number of directions, the types of forces, and the shapes of the objects (convex, concave, disjoint or overlapping).
Acknowledgments

First, I would like express my gratitude and deep regards to my advisor, Dr. Pascal Matsakis, for his useful comments, his advice and engagement through out my Masters.

Next, I would like to thank my Advisory Committee Member, Dr. Andrew Hamilton, for his valuable comments and suggestions.

I would also like to take this opportunity to thank my Parents, family members and friends (Vicky, Charry, Cheeko and Raja) for their endless support through out my degree. This would not have been possible without their help.
# Table of Contents

Abstract ii

Acknowledgements iii

1 Introduction 1
   1.1 Background 1
   1.2 Thesis Statement 2
   1.3 Motivation and Significance 3
   1.4 Notation 4
   1.5 Thesis Structure 5

2 Literature Review 6
   2.1 Relative Position Descriptors 6
   2.2 Examples of Relative Position Descriptors 7
      2.2.1 Angle Histogram 7
      2.2.2 Φ Descriptor 8
      2.2.3 Force Histogram 9
   2.3 Traditional Definition of the Force Histogram 11
      2.3.1 Line Based Algorithm for 2D Raster Data 12
      2.3.2 Line Based Algorithm for 3D Raster Data 15
   2.4 Alternative Definition of the Force Histogram 17
   2.5 Geometric Properties of the Force Histogram 20
   2.6 Summary 21

3 Force Histogram Computation: New Algorithm for 3D Raster Data 22
   3.1 Introduction 22
   3.2 Implementation of $\Psi^{AB}$ 23
   3.3 Implementation of $\hat{F}^{AB}_i$ 25
   3.4 Summary 28
4 Experiments

4.1 Experimental Settings ............................................. 29
4.2 Effect of the Type of Forces ...................................... 31
  4.2.1 Test Data .................................................. 31
  4.2.2 Results and Discussion ..................................... 32
4.3 Effect of the Number of Directions .............................. 33
  4.3.1 Test Data .................................................. 33
  4.3.2 Results and Discussion ..................................... 34
4.4 Effect of Size and Relative Position .............................. 35
  4.4.1 Test Data .................................................. 35
  4.4.2 Results and Discussion ..................................... 36
4.5 Effect of Shape .................................................... 37
  4.5.1 Test Data .................................................. 37
  4.5.2 Results and Discussion ..................................... 38
4.6 Comparing Histogram values ....................................... 39
4.7 Conclusion ......................................................... 41

5 Conclusions and future work ..................................... 43
  5.1 Summary ....................................................... 43
  5.2 Conclusions and Future Work ................................... 45

Bibliography .......................................................... 46
Chapter 1

Introduction

1.1 Background

Humans often rely on the relative position of the objects around them in order to understand and express information about space. In our daily lives, relative position of objects is often conveyed through linguistic expressions like, “the store is north of the apartment”, “the school is nearby my home”, “the folder is inside the case”. Spatial prepositions like “north”, “inside” and “nearby” represent spatial relationships. Distance relationships describe how far apart the objects are in space, like “far away”, “nearby”, “at”. Directional relationships (also called cardinal [1] or projective [2] relationships) are mostly identified by words like “west”, “to the right of”. Topological relationships specify concepts of connectivity, adjacency and enclosure [3–5].

Comprehending the spatial relationships in a scene plays a vital role in several areas such as pattern recognition, human-robot communication, scene description in natural language, image understanding and so forth. Models of spatial relationships (distance, directional and topological) among spatial entities are either quantitative or qualitative. In qualitative models [6], a relationship either holds or does not hold.
On the other hand, in quantitative models, a relationship may hold to some degree [7]. Over the years, qualitative models have been used in several fields of computer vision. However, many practical image processing and computer vision applications need quantitative models. These models are called relative position descriptors [8].

A relative position descriptor (RPD) is a visual descriptor like colour, shape and texture descriptors, which quantifies the relative position of an object with respect to another object. It forms the foundation from which the models of spatial relationships can be obtained. A relative position descriptor fills the gap between the low-level spatial features (e.g., pixels in 2D raster image) and the high-level features (e.g., spatial relationships). Over the years, several RPD’s have been proposed in the literature. The force histogram may be one of the most well known relative position descriptor, and many algorithms for force histogram computation are described in the literature. Algorithms exist to compute the force histogram in case of 2D raster objects, 2D vector objects, 3D raster objects and 3D vector objects, although the algorithms for the 3D objects are computationally costly.

1.2 Thesis Statement

In the present thesis, our research focuses on the quantitative representation of relative position among objects in an image. Force histogram seems to be a simple yet powerful tool available to quantify the relative position among objects. Two practical
algorithms have been introduced so far to compute the force histogram among objects in an image: the line-based approach that usually works well with 2D data, however, in the case of 3D data, its computational performance is unstable and usually affected by several factors; the Fast Fourier Transfer (FFT) based approach is computationally costly in the case of 2D data, and it has not been implemented for 3D data.

The thesis presents an extension of FFT-based algorithm for force histogram computation to 3D raster data. Contrary to the existing 3D line-based algorithm, its computational performance does not depend on the number of directions considered, or whether the objects are concave or convex, disjoint or overlapping. Except in few particular cases, the new 3D FFT- based algorithm outperforms the existing 3D line-based algorithm.

1.3 Motivation and Significance

Humans tend to understand and convey information about the space through relative position of objects in the space. The task seems natural for humans, however, it is cumbersome for machines. This requires thorough empirical investigation of variety and uncertainty embedded in the environment. Automated quantitative representation of spatial relationships is often considered as a task of making computerized models for spatial cognition. Our work in this thesis presents a step in that direction. Several applications in computer vision and image processing depends on the
effective capture of spatial relationship information. Content based image retrieval (CBIR) [9], robot navigation [10, 11], Geographic Information System (GIS) [12] are few typical applications among many others that require quantitative representation of spatial relationship.

For example, an autonomous robot should know its location with respect to its surrounding in order to reach its target accurately and safely. This is usually achieved by generating the quantitative representation of relative position between robot and the objects in its surrounding.

Force histogram is one such mathematical tool that encapsulates the spatial relationship information along with the structural information (size, shape etc.,) of objects. It quantifies the relative position of objects in an image using a set of real numbers; where each number denotes the magnitude of force exerted by an object on another object in a particular direction.

1.4 Notation

A 2D raster object is a set of pixels, whereas 3D raster object is a set of voxels. Symbols $\mathfrak{R}$, $\mathfrak{R}_+$ and $\mathfrak{R}_*$ denotes the set of real numbers, positive real number and non-negative real numbers. Similarly $\mathbb{Z}$, $\mathbb{Z}_+$ and $\mathbb{Z}_*$ denotes the set of integers, non-negative integers and positive integers. $\mathcal{P}$ refers to the Euclidean plane. Consider a point $p \in \mathcal{P}$ and direction $\theta$, $\Lambda_\theta(p)$ is the straight line run through the $p$ in the direction $\theta$. The magnitude of the $\vec{p}$ is denoted by $|p|$, and a direction is given by $\angle p$. 
Affine transformations are denoted by $\text{aff}$. Translation $\text{tra}$, dilation $\text{dil}$, reflection $\text{ref}$ and rotation $\text{rot}$ are few example of affine transformations.

### 1.5 Thesis Structure

Chapter 2 reviews the various relative position descriptor’s found in the literature, and focuses on the force histogram, and its various applications and geometric properties. A detailed review of the algorithms existed for the computation of force histogram in case of 2D and 3D raster data has also been presented.

Chapter 3 focuses on implementing FFT-based algorithm for 3D raster data.

Chapter 4 describes the test data and the experiments conducted to examine and compare the computational performance of FFT-based algorithm with the line-based algorithm for force histogram computation.

Chapter 5 presents the summary and conclusion of our work.
Chapter 2

Literature Review

2.1 Relative Position Descriptors

In our daily lives, people often use relative position of objects to understand, and to communicate about the space. Relative position descriptor (RPD) is an image descriptor that describes the quantitative representation of relative position between the objects. In other words, RPD is a vector of numbers that provide information about the relative position of objects in the space. While describing the relative position of object B with respect to object A, the object B is called the argument, and the object A is called the referent. An ideal relative position descriptor should have low computation time, and the ability to handle both vector and raster data. In addition, it should also encapsulates all the possible spatial relationship information between the objects e.g directional relationships (north of, to the east of.....) [1], topological relationships (adjacent, in...) [3–5] and distance relationships (far, near...) [13].

RPD’s can be used either separately or in combination with other image descriptors for several image processing and computer vision applications. RPD’s are mostly commonly used in scene matching [14, 15], shape matching [16], human-robot interaction [17], geospatial information retrieval and indexing [18], land cover classifica-
tion [19] and map to image conflation [20]. In the past, considerable attention has been given to the modelling of relative position descriptors. Force histogram [21,22], Phi-descriptor [8] are the well known RPD’s.

This chapter gives a review of existing relative position descriptors in the Section 2.2. Section 2.3 focuses on the force histogram - the most commonly used RPD with maximum theoretical and practical applications. Geometric properties of force histogram are discussed in Section 2.5

2.2 Examples of Relative Position Descriptors

2.2.1 Angle Histogram

The angle histogram [23,24] is one of the oldest relative position descriptor found in the literature. For a given object pair (A,B), the angle histogram describes the position of A relative to B. Consider $a \in A$, $b \in B$. Mathematically, for $\theta \in \mathcal{R}$, the angle histogram $H^{AB}(\theta)$ count the pixel pairs $(a, b) \in (A \times B)$; such that $\angle a,b \in \theta$, where $\angle a,b$ is the angle between the oriented line that passes through the centre of pixels a and b and the x-axis.

Although the angle histogram is simple to compute, it has many weaknesses: (1) it is sensitive to the size, orientation and shape of objects, (2) the computation of angle histogram is slower than other RPD’s, and (3) it can’t handle vector objects.
2.2.2 Φ Descriptor

The Φ descriptor [8] is a recently introduced relative position descriptor that has a competitive edge over other RPD’s due to its ability to encapsulate a large amount of spatial relationship information between the objects. Additionally, it can handle both raster and vector objects despite their topology (connected, disconnected, with or without holes). The Φ descriptor for the objects A and B is denoted as $\Phi^{AB}$.

Consider objects A and B, points p and q, and a direction $\theta$ in the interval $[0,2\pi]$ as in Figure 2.1. The segment $[p,q]$ is called interaction segment, if $[p,q] \in A \cup B$ and to a line $M$ that intersect both objects A and B along the direction $\theta$. The union of all interaction segments between the objects A and B along the direction $\theta$ is called region of interaction (Figure. 2.1a). The area of region of interaction is denoted by $\Phi$.
$Z^{AB}_0$, and its average width is denoted by $W^{AB}_0$.

For any direction $\theta$, the region of interaction can be subdivided into number of sub-regions, where each sub-region express some fundamental spatial relationship between the objects (Figure 2.1b). The sub-regions can be further classified into 10 categories. We attach two functions to each category: the first record the areas of the corresponding sub-regions, and the second record the average widths of the sub-regions. Therefore, we have 10 area functions, $Z^{AB}_1$, $Z^{AB}_2$, $Z^{AB}_3$, $Z^{AB}_4$ .......$Z^{AB}_{10}$, and 10 width functions, $W^{AB}_1$, $W^{AB}_2$, $W^{AB}_3$, $W^{AB}_4$ .......$W^{AB}_{10}$, in addition to the functions $Z^{AB}_0$ and $W^{AB}_0$.

The $\Phi^{AB}$ between the objects A and B is defined as:

$$\Phi^{AB} = (\text{area}(A), \text{area}(B), Z^{AB}_0, Z^{AB}_1, Z^{AB}_2 ......Z^{AB}_{10}, W^{AB}_0, W^{AB}_1, W^{AB}_2 ......W^{AB}_{10})$$ (1)

The function $\Phi^{AB}$ represents the relative position of object A with respect to B, and holds a large amount of quantitative and qualitative spatial relationship information [25].

### 2.2.3 Force Histogram

In [21,22], Matsakis and Wendling introduced a new relative position descriptor called force histogram. The force histogram is a quantitative representation of the relative position between two objects. The force histogram assumes that the objects A and
B have uniform density and negligible thickness, and the particles of B are exerting elementary forces on those of A. The sum of elementary forces along each direction \( \theta \) is computed, and then recorded. The force histogram is, therefore, a function from \( \mathbb{R} \) (set of directions) to \( \mathbb{R}_+ \) (Figure 2.2b).

![Figure 2.2: Force histogram:](image)

(a) Consider two objects A and B, and the forces in direction \( \theta \). (b) The histogram \( F_{t}^{AB} \) between the objects A and B

Mathematically, the force histogram is a numerical function \( F_{t}^{AB}(\theta) \) defined on \([0,2\pi]\), where \( t \) is any real number. The function \( F_{t}^{AB}(\theta) \) quantifies the position of an object A relative to another object B. Here, the object A is called argument and the object B is called referent. For a given direction \( \theta \), the value \( F_{t}^{AB}(\theta) \) is the sum of the magnitude of all elementary forces exerted by the points of B on those of A,
which in turn tends to move A in the direction \( \theta \) (Figure 2.2a). The magnitude of each force is \( 1/|d|^t \), where \( d \) is the distance between the points, and \( t \) is any real number. The function \( F_t^{AB}(\theta) \) has two special cases: \( t = 2 \), and \( t =0 \). When \( t=2 \), \( F_2^{AB}(\theta) \) is called the gravitational force histogram, and the forces between the points are inversely proportional to the the square of distance between them \( (1/d^2) \). When \( t =0 \), \( F_0^{AB}(\theta) \) is called constant force histogram, and the forces do not depend on the distance.

### 2.3 Traditional Definition of the Force Histogram

Consider A and B are the two objects defined in \( \mathcal{P} \). Here, each object contains finite number of points, and quantifiable nonzero area. Let \( a \in A \) and \( b \in B \). The distance between the points \( a \) and \( b \) is given by \( |ab| \), and the direction of the line that runs from \( a \) to \( b \) is given by \( \angle ab \). The points \( a \) and \( b \) can be seen as the particles that attract each other, and \( b \) exerts an elementary force on \( a \) whose direction is \( \angle ab \) and the magnitude is \( 1/|ab|^t \).

In [21], Matsakis introduced the formal definition of force histogram as follows:

\[
F_t^{AB}(\theta) = \int_{-\infty}^{+\infty} F_t(A,B,r,\theta)dr,
\] (2)
where

$$F_t(A, B, r, \theta)dr = \int_{a \in \Lambda_\theta(r)} \int_{b \in \Lambda_\theta(r)} A(a)B(b)\varphi_t(ab, \vec{\theta})da\,db$$ \hspace{1cm} (3)

and

$$\varphi_t(d) = 1/d^t \text{ if } t \in \mathbb{R}_+ \text{ and } \varphi_t(d) = 0 \text{ otherwise}$$

Two algorithms have been implemented in the past for the 2D and 3D data using the traditional definition of force histogram. The algorithm for the the 2D data — line-based algorithm for 2D raster data, is discussed in Section 2.3.1, and for the 3D data — line-based algorithm for 3D raster data, is discussed in Section 2.3.2.

### 2.3.1 Line Based Algorithm for 2D Raster Data

Matsakis indicated in [21, 22] that the force histogram between objects in an image can be computed by partitioning the image into number of raster lines, and then batch-processing pixels on each line.

Mathematically, for a given direction $\theta$, the value of force histogram $F_t^{AB}(\theta)$ can be approximated using the Riemann sum:
\[ F_t^{AB}(\theta) = \int_{-\infty}^{+\infty} F_t(A, B, r, \theta) \approx (\cos \theta) \sum_{-\infty}^{+\infty} F_t(A, B, d \cos \theta, \theta) \]  \hspace{1cm} (4)

Figure 2.3: **Line Based Algorithm:** \( \Lambda_\theta(d \cos \theta) \) is the straight line that run through the pixels of object A and B in the direction \( \theta \). Pixels of A ("A", "AB") make segment \( I_i \), and pixels of B ("B", "AB") make segment \( J_j \).

For the sake of lucidity, let us suppose that A and B are the objects and \( \theta \in [0, \pi/4] \). The Bresenham’s algorithm [26] is used to rasterize the line \( \Lambda_\theta(d \cos \theta) \) into set of pixels (Figure 2.3). The pixels of object A are marked with “A”, object B are marked with “B”, and both A and B are marked with “AB”. The pixels of object A make segment \( I_i \), and object B make segment \( J_j \).
In other words, we have $B \cap \Lambda_{\theta}(dcos\theta) \approx \bigcup J_i$ and $A \cap \Lambda_{\theta}(dcos\theta) \approx \bigcup I_i$.

$$F_t(A, B, dcos\theta, \theta) \approx \int_{a \in \bigcup I_i} \int_{b \in \bigcup J_j} \varphi_t(\vec{ab}, \vec{\theta})dadb = \sum_i \sum_j \int_{a \in I_i} \int_{b \in J_j} \varphi_t(\vec{ab}, \vec{\theta})dadb \quad (5)$$

The value $F_t(A, B, dcos\theta, \theta)$ is a scalar resultant of the elementary forces exerted by the points of $B \cap \Lambda_{\theta}(dcos\theta)$ on those of $A \cap \Lambda_{\theta}(dcos\theta)$ in the direction $\theta$, where $\Lambda_{\theta}(dcos\theta)$ is the oriented line passing through the objects $A$ and $B$. Therefore, according to Equation (5), for any oriented line $\Lambda_{\theta}(dcos\theta)$, computing force histogram is equivalent to calculating the forces between the segments $I_i$ and $J_j$. Each segment comprises a set of adjacent pixels that are batch processed. From the Equation (5), value of force histogram yields a set of algebraic equations while exploiting the power of integral calculus, and are hard coded.

In an ideal case, the shapes of objects $A$ and $B$ are simple (e.g., convex). For any direction $\theta$, the line $\Lambda_{\theta}(dcos\theta)$ intersects each object exactly once, thus number of segments $I_i$ and $J_j$ included in a line is at most 2. The computation of force histogram is equivalent to computing forces between two segments, thus lowering the computation times. The algorithm runs in $O(KN)$ time for convex objects, where the $K$ is the the number of directions in which forces are considered and $N$ is the number of pixels in the object. In the worst case, objects $A$ and $B$ have complex shapes (e.g., fractal like structure or objects with multiple disconnected components). For any direction $\theta$, the
line $\Lambda_{\theta}(d\cos\theta)$ intersects each objects multiple times, thus the number of segments $I_i$ and $J_j$ may be very high. This will leads to computing forces between multiple segments, thus increasing the computation time. The algorithm runs in $O(KN\sqrt{N})$ time.

### 2.3.2 Line Based Algorithm for 3D Raster Data

In [27], the line-based algorithm for force histogram computation is extended for 3D data. The basic principal remains unchanged. For a direction $\vec{\theta}$ (here defined by a unit vector), the function $F_i^{AB}(\vec{\theta})$ is the scalar resultant of elementary forces exerted by the voxels of B on those of A. The $\vec{\theta}$ is taken from a set $D^K$ of evenly distributed reference directions. The $K$ reference directions in the set $D^K$ should follow three primary constraints: (1) they should be evenly distributed in 3D space, (2) they should include six primitive directions i.e., $(\pm 1,0,0)$ (left/right), $(0,\pm 1,0)$ (above/below), and $(0,0,\pm 1)$ (front/behind), and (3) if a direction $\theta$ belongs to the set, then the opposite direction $-\theta$ should also be a part of the direction set. The above constraints indicate $m = 6 + 8l$, where 6 denotes six primitive directions, 8 denotes the regions demarcated by the XY, YZ, ZX plane, and $l$ imply the number of reference directions in each plane.

However, contrary to 2D case, mostly $K$ reference directions are not evenly placed in 3D space. Bourke’s [28] random-start-hill-climbing heuristic have been exploited
to evenly place $K$ reference directions in the 3D space. According to Bourke in [28], initially the points are randomly placed on the 3D sphere. These points tends to repel each other, and an iterative process have been followed until a stable configuration is assembled. Additionally, a slight modification of Bourke’s heuristic is made to ensure that second and third constraints are met.

However, 3D data is significantly larger than the 2D data, so suitable optimization methods need to be followed. Therefore, in order to ease the computational burden, number of irrelevant directions are identified and dropped. Forces for remaining directions are computed almost similar to 2D case. A simple yet powerful test is conducted in order to drop irrelevant directions. According to the test in [27], a direction $\vec{\theta}$ is considered relevant, if $F^{AB}_t(\vec{\theta}) \neq 0$ i.e., for any oriented line $\Lambda$, $A \cap \Lambda \neq 0$ and $B \cap \Lambda \neq 0$. Otherwise, a direction $\vec{\theta}$ is irrelevant, and should not be considered to compute force histogram $F^{AB}_t(\theta)$.

Consider a direction $\vec{\theta} = (x,y,z)$, $\vec{\theta}_{\text{max}}$ plane is described as follows:

if $\text{max}(x,y,z) = x$, then $\vec{\theta}_{\text{max}}$ is YZ plane.

if $\text{max}(x,y,z) = y$, then $\vec{\theta}_{\text{max}}$ is XZ plane.

if $\text{max}(x,y,z) = z$, then $\vec{\theta}_{\text{max}}$ is XY plane.
2.4 Alternative Definition of the Force Histogram

The line-based algorithm, based on the image partitioning and batch processing of pixels (Section 2.3), has two significant drawbacks: (1) the computational time is di-
rectly proportional to the number of directions \((K)\) in which the forces are calculated, and (2) it depends on the shape of the object (concave or convex) and their relative position (overlapping or disjoint). To solve the issues encountered by the line-based algorithm, Matsakis and Jing Bo Ni [29] introduced an alternative definition for force histogram computation. The force histogram is derived from the mapping defined over the discrete space. The mapping is defined as — spatial correlation, which offers the raw information on the relative position of the objects, and can be computed fast using the Fast Fourier transform (FFT).

Consider a function \(\Psi_{\text{AB}} : \vec{Q} \rightarrow \mathbb{R}^*\) between the objects A and B given by:

\[
\Psi_{\text{AB}}(\vec{v}) = \int_{q \in Q} A(p) B(q + \vec{v}) \, dp
\]  

Equation (6) is a mathematical correlation that gives the raw information about the position of an object A relative to another object B. Below, Equation (6) is restated using the Cartesian coordinates:

\[
\Psi_{\text{AB}}(s, r) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} A(x, y) B(x + s, y + r) \, dx \, dy
\]  

Then, the force histogram \(\hat{F}_{\text{AB}}(\theta)\) can be computed using the mapping defined
in Equation (6). For two objects A and B, the function \( \hat{F}_{t}^{AB}(\theta) \) is defined by:

\[
\hat{F}_{t}^{AB}(\theta) = \int_{0}^{\infty} \frac{\Psi^{AB}(d\theta)}{d^{t}} dd
\]  

(8)

\( \hat{F}_{t}^{AB}(\theta) \) can be rewritten using the Cartesian coordinates as:

\[
\hat{F}_{t}^{AB}(\theta) = \int_{0}^{\infty} \frac{\Psi^{AB}(d\cos\theta, d\sin\theta)}{d^{t}} dd
\]  

(9)

We have \( F_{t}^{AB}(\theta) = \hat{F}_{t}^{AB}(\theta) \), where \( F_{t}^{AB}(\theta) \) is the force histogram computed using the line-based algorithm, defined through Equations (4)-(5) and \( \hat{F}_{t}^{AB}(\theta) \) is the force histogram computed using the FFT-based algorithm, defined through Equations (8)-(9). The proof is provided in the Appendix.

The computation time for the new FFT-based algorithm is independent of the type of forces (t) and number of directions (K) in which the forces are calculated. All objects regardless of their shape (convex, concave, complex or simple) and relative position (disjoint or overlapping) are treated in the same manner.
2.5 Geometric Properties of the Force Histogram

Various computer vision applications require robust descriptors that are not sensitive to the camera’s position in relation to the photographed scene. An ideal relative position descriptor should not get influenced by the variation in the size, rotation and translation of the scene. However, mostly the RPD for the object pair (A,B) changes, if some geometric transformation is being applied to the object pair. The force histogram can be normalized to make it a affine invariant relative position descriptor [30]. Therefore, if we know the Force histogram \( \hat{F}_{t}^{AB} \) and affine transformation \( \text{aff} \) for the object pair (A,B), we can easily retrieve \( \hat{F}_{t}^{\text{aff}(A)\text{aff}(B)} \). The behaviour of \( \hat{F}_{t}^{AB} \) towards invertible affine transformations such as translation (\( \text{tra} \)), rotation (\( \text{rot} \)), dilation (\( \text{dil} \)), reflection (\( \text{ref} \)) has been discussed in [30]. Proofs of the properties have been given in Appendix A. These properties hold true for both traditional definition of force histogram and the alternative definition of force histogram, since \( \hat{F}_{t}^{AB} = F_{t}^{AB} \).

For any \( \theta \in [0, 2\pi] \), we have:

Property 1:

\[
F_{t}^{\text{tra}[A]\text{tra}[B]}(\theta) = F_{t}^{AB}(\theta),
\]

Property 2:

\[
F_{t}^{\text{ref}[A]\text{ref}[B]}(\theta) = F_{t}^{AB}(\theta'),
\]

where \( \theta' \in [0, 2\pi] \), and is congruent to \( 2\alpha - \theta \) modulo \( 2\pi \)
Property 3:

$$F_{t}^{\text{rot}[A]\text{rot}[B]}(\theta) = F_{t}^{AB}(\theta'),$$

(12)

where $\theta' \in [0, 2\pi]$, and is congruent to $2\alpha - \rho$ modulo $2\pi$

Property 4:

$$F_{t}^{\text{dil}[A]\text{dil}[B]}(\theta) = \lambda^{3-t} F_{t}^{AB}(\theta),$$

(13)

Property 5:

$$F_{t}^{\text{aff}[A]\text{aff}[B]}(\theta) = |\text{det}(\text{lin})||\text{lin}^{-1}.\vec{\theta}|^{r-1} F_{t}^{AB}(\theta'),$$

(15)

where $\theta' = \angle(\text{lin}^{-1}.\vec{\theta})$ and $\text{aff} = \text{tra.lin}$ is invertible linear transformation.

2.6 Summary

Relative position descriptor quantifies the relative position of objects that can be easily handled by the machines. Since RPD’s can effectively capture the quantitative picture of spatial relations, they are extensively used in various application of computer vision and image processing. Force Histogram is one such RPD with most practical applications. Over the years, many algorithms have been implemented for the computation of force histogram between objects. However, the research in the relative position descriptors is still under development. The work in our thesis is a step towards that direction.
Chapter 3

Force Histogram Computation: New Algorithm for 3D Raster Data

3.1 Introduction

The Line-based algorithm derived from the traditional definition of force histogram have been implemented for both 2D and D raster data (Section 2.3). Computation times depend on type of forces \( t \), the number of directions \( K \) in which the forces are computed, the shape (e.g., convex or concave) and the relative position (e.g., disjoint or overlapping). These algorithms works well only when the objects’ shapes are simple (e.g., convex) and the number of directions in which the force histogram is computed is small. An alternative definition of the force histogram has been introduced (Section 2.4), and the FFT-based algorithm has been derived from the definition in case of 2D raster data. Computation times are independent of the type of forces, number of directions, shape and relative position of the objects.

The Primary contribution of our work is to extend FFT-based algorithm to 3D raster data. A force histogram is computed using a mapping \( \Psi^{AB} \) defined over the 3D space, where \( \Psi^{AB} \) is the correlation between the 3D objects. The mapping \( \Psi^{AB} \) is computed
using the Fast Fourier Transform (FFT).

Section 3.2 focusses on the implementation of the spatial correlation ($\Psi^{AB}$), then computation of the force histogram using $\Psi^{AB}$ is discussed in Section 3.3. Section 3.4 presents a summary of the Chapter.

3.2 Implementation of $\Psi^{AB}$

Consider A and B are the 3D raster objects in a domain $G_{h\times k\times l}$ (Figure 3.1). The $\Psi^{AB}$ provide information about the relative position of object A with respect to another object B. The spatial correlation $\Psi^{AB}$ between raster objects A and B is defined on a mapping from $\mathbb{Z}^3$ to $\mathbb{R^*}$. For any $d \in \mathbb{Z}$, $e \in \mathbb{Z}$ and $f \in \mathbb{Z}$, $\Psi^{AB}$ is defined as following:

$$\Psi^{AB}(d, e, f) = \sum_{h=0}^{h-1} \sum_{k=0}^{k-1} \sum_{l=0}^{l-1} A(x, y, z) B(x + d, y + e, z + f) \quad (9)$$

Let $G_{h\times k\times l} = -h+1 \ldots \ldots h-1 \times k+1 \ldots \ldots k-1 \times l+1 \ldots \ldots l-1$. $G_{h\times k\times l}$ is the effective domain of $\Psi^{AB}$. In Equation (9), we suppose that $B(x + d, y + e, z + f) = 0$, if the voxel $(x+d, y+e, z+f)$ lies outside of $G_{h\times k\times l}$; such that $(x+d, y+e, z+f) \notin (0 \ldots \ldots h-1 \times 0 \ldots \ldots k-1 \times 0 \ldots \ldots l-1)$. $\Psi^{AB}$ counts the number of voxel pairs $(p, q)$, where $p \in A$ and $q \in B$; such that $\vec{pq} = (d, e, f)$. It is not difficult to see that only voxels $(d, e, f)$ that are in effective domain $G_{h\times k\times l}$, will only satisfy $\Psi^{AB} \neq 0$. Since there are $6N$ elements
Consider the reflection of object A about the origin, such that \( A'(-x, -y, -z) = A(x, y, z) \). By replacing \( A(x, y, z) \) with \( A'(x', y', z') \) in Equation 9, where \( x' = -x, y' = -y \) and \( z' = -z \). We get,

\[
\Psi^{AB}(d, e, f) = \sum_{-h+1}^{0} \sum_{-k+1}^{0} \sum_{-l+1}^{0} A(x', y', z') B(d - x, e - y, f - z) \tag{10}
\]
Equation (10) refers to the 3-D discrete convolution. Using matrix notation, we get:

\[
[\Psi^{AB}] = [A'] \otimes [B'],
\]

(11)

where \( \otimes \) is the convolution operator.

According to Convolution theorem, we all know that Fourier transform has the power to convert the convolution into an ordinary multiplication, and vice versa. Using Convolution theorem, equation (11) can be transformed into:

\[
[\Psi^{AB}] = W_{2h-1}^{-1} \cdot W_{2k-1}^{-1} \cdot \left( (W_{2h-1} \cdot W_{2k-1} \cdot [A'] \cdot W_{2l-1}) \times (W_{2h-1} \cdot W_{2k-1} \cdot [B'] \cdot W_{2l-1}) \right) \cdot W_{2l-1}^{-1}
\]

(12)

where \( W_i \) and \( W_i^{-1} \) are the Fourier and inverse Fourier transform matrices of order \( i \) and \( \times \) is the matrix product. As per Equation (12), computation of \([\Psi^{AB}]\) requires three 3-D discrete Fourier transforms. Using Convolution theorem, \( \Psi^{AB} \) can be computed in \( O(N \log N) \) time, as this is the complexity of the Fast Fourier transform (FFT).

### 3.3 Implementation of \( \hat{F}_{t}^{AB} \)

In this section, \( \hat{F}_{t}^{AB} \) is computed using the spatial correlation \( \Psi^{AB} \) (see Section 3.2). Only a finite number of directions \( \theta \) are considered. The reasonable pick for \( \theta \) (Section
2.3.2), will be to choose from a set of evenly distributed reference directions $D^m$. For 3 dimensional objects, Equation (9) takes the following form:

$$
\hat{F}_{t}^{AB}(\theta) = \int_{0}^{\infty} \frac{\Psi^{AB}(d\cos\theta, d\sin\theta, z)}{d^k} dd
$$

(13)

Using spherical coordinates, Equation (13) can be written as:

$$
\hat{F}_{t}^{AB}(\theta) = (\sin\varphi)^{1-t} \int_{0}^{\infty} \frac{\Psi^{AB}(\rho \sin\varphi \cos\theta, \rho \sin\varphi \sin\theta, \rho \cos\varphi)}{\rho^t} d\rho
$$

(14)

where $\rho \sin\varphi = d$ i.e., $dd = d\rho \sin\varphi$ and $z = \rho \cos\varphi$, and $\rho \in \mathbb{R}_+$. Let $s = \rho \cos\varphi$ i.e., $\rho = s/\cos\varphi$ and $d\rho = ds/\cos\varphi$. Equation 14 becomes:

$$
\hat{F}_{t}^{AB}(\theta) = (\cos\varphi)^{t-1} (\sin\varphi)^{1-t} \int_{0}^{\infty} \frac{\Psi^{AB}(s \tan\varphi \cos\theta, s \tan\varphi \sin\theta, s)}{s^t} ds
$$

(15)

$\hat{F}_{t}^{AB}(\theta)$ is then approximated, according to Equation 15, as follows:

$$
\hat{F}_{t}^{AB}(\theta) \approx (\cos\varphi)^{t-1} (\sin\varphi)^{1-t} \sum_{0}^{h-1} \Psi^{AB}(s \tan\varphi \cos\theta, s \tan\varphi \sin\theta, s)f_{i}(s)
$$

(16)

where $f_{i}(s)$ is selected according to the interpretation of $1/s^t$ in context of the Equation (16).
Figure 3.2: **Fast Fourier Based Force Histogram Computation:** A and B are the 3D raster objects in the domain $G_{h \times k \times l}$. $\Delta$ is the line run through the pixel $p$ of the $R(A_\theta \cap B_\theta)$ in the direction $\theta$, here $\theta^{max}$ is the YZ plane.

Consider two points $a$ and $b$ in $P$ such that $\vec{ab}$ is the vector of coordinates $(s \tan\varphi \cos\theta, s \tan\varphi \sin\theta, s)$, where $s \in \mathbb{R}_+$. Let $a'$ and $b'$ is the projection of points $a$ and $b$ on YZ plane. Thus, in Equation (15) $s$ depicts the distance between $a'$ and $b'$, and $1/s^t$ quantify the magnitude of elementary force exerted by $b'$ on $a'$. This force tends to pull $a'$ in direction 0. On the other hand, in Equation (16), which is defined on the discrete image domain $G_{h \times k \times l}$ (Figure 3.2), $a$ and $b$ becomes the voxels such that $\vec{ab}$ is the vector of coordinates $(s \tan\varphi \cos\theta, s \tan\varphi \sin\theta, s)$, where $s \in 0 \ldots h-1$. 

27
These voxels project on the YZ plane as I and J where $I = A \cap \triangle$ and $J = A \cap \triangle$, $\triangle$ is line that passes through the voxels $a$ and $b$. It is therefore easy to say that, $f_t(s)$ sum of magnitude of all elementary forces exerted by the points in J on those in I and tends to pull I in direction 0 (Figure 3.2).

### 3.4 Summary

In conclusion, the FFT-based algorithm has been extended to 3D raster data, and runs in $O(N\log N)$ time. Computation times are independent of the types of forces, number of directions and the type of objects being considered. Thus, it is suitable for computing force histogram for complex and large objects. The processing time stays almost the same for simple convex objects and complicated fractal like structures. The experiments conducted in Section 4 validate our analysis.
Chapter 4

Experiments

4.1 Experimental Settings

In this section, $F_{t}^{AB}$ denotes a force histogram computed using the line-based algorithm and $\hat{F}_{t}^{AB}$ denotes a force histogram computed using the new FFT-based algorithm. Both FFT-based algorithm and line-based algorithm are approximation algorithms. The line-based algorithm is based on the approximation of the Equations (2)–(5), and the FFT based algorithm is based on the approximation of the Equations (8)–(9). So, we can expect $\hat{F}_{t}^{AB} \approx F_{t}^{AB}$, rather than $\hat{F}_{t}^{AB} = F_{t}^{AB}$.

The aim of the experiments is to measure and compare the computational time of $\hat{F}_{t}^{AB}$ and $F_{t}^{AB}$ for 3D raster objects. In addition, the effect of different parameters such as number of directions (K) in which the forces are computed, number of voxels (N), type of forces (t), shape and relative position (disjoint or overlapping) of the objects is studied on the computational time of $\hat{F}_{t}^{AB}$ and $F_{t}^{AB}$. The computation time is measured using the C library function clock().

The experiments are conducted on a variety of 3D raster objects. The test data consist of cubes, objects generated from quadratic Koch island fractals at iterations 2 and 3, overlapping objects and segmented cubes. The size of the objects is kept...
constant \( (N = 512^3) \) for all experiments, except for the scenario where the effect of size is investigated on the computational time of \( \hat{F}^{AB}_t \) and \( F^{AB}_t \). The object pair considered in each experiment is disjoint and identical unless otherwise stated. We have chosen different object pairs to study the impact of each parameter \( (N, K, t, \text{shape and relative position of objects}) \) on computation time of \( \hat{F}^{AB}_t \) and \( F^{AB}_t \).

The cubes (Figure 4.3) are the perfect scenario for the line-based algorithm based on the traditional definition of force histogram. For any direction \( \theta \), the line that passes through both objects A and B intersects each object exactly once, thus number of segments included in a line is at most 2. Therefore, for every direction, the computation of force histogram comes down to computing forces between two segments. The complex shapes (Figures 4.5, 4.7) are the worst scenario for the line-based algorithm. For any direction \( \theta \), the line that passes through both objects A and B intersects each object multiple times, thus the number of segments can be very high. This will lead to computing forces between multiple segments, thus increasing the computation time. Additionally, for overlapping objects, the lines that crosses both objects A and B can be found in every direction \( (F^{AB}_t(\theta) \text{ is never 0}) \). This will further add up to the processing time.

The algorithms are implemented in C, and the experiments were conducted on a mac OS equipped with 2GHz Intel Core i5 CPU and 8GB memory.
4.2 Effect of the Type of Forces

4.2.1 Test Data

The object pair given in Figure 4.1 is considered for this experiment. The object pair consists of two identical and disjoint right cylinders generated from quadratic Koch island fractals at iteration 2. The number of voxels \( N = 512^3 \) of the objects, and the number of directions for which the histogram values are computed, are kept constant throughout the experiment.

Figure 4.1: Test Data: The 3D objects used in this experiment are right cylinders whose bases are quadratic Koch island fractals at iteration 2.
4.2.2 Results and Discussion

Figure 4.2 shows the effect of $t$ on the processing times of $F^t_{AB}$ and $\hat{F}^t_{AB}$. There is no impact of $t$ on the processing time of $\hat{F}^t_{AB}$, but the computation of $F^t_{AB}$ is considerably faster when $t=1$ or $t=2$ (Figure 4.2). The prominent reason is that computation of each $F^t_{AB}$ comes down to evaluating a number of numerical expressions which are computationally faster when $t=1$ or $t=2$ (Section 2.3.1).

Figure 4.2: Processing times ($T$) of force histograms with respect to $t$ (type of forces). The objects A and B are from Figure 4.1 where $N = 512^3$ for both objects, and $K=1800$ (number of reference direction for which forces are computed), and the increment for $t$ is 0.5.
4.3 Effect of the Number of Directions

4.3.1 Test Data

The object pairs given in 4.3 are considered for this experiment. The objects are two identical and disjoint cubes. The size of the objects is kept constant ($N = 512^3$). The number of directions (K) is varied from 360 to 2520. The experiment was run for constant force histograms (t=0) and gravitational force histograms (t= 2).

Figure 4.3: Test Data: The objects used in the experiment are disjoint identical 3D cubes.
4.3.2 Results and Discussion

Figure 4.4 shows the effect of the number of directions (K) on the processing time of $F_{t}^{AB}$ and $\hat{F}_{t}^{AB}$. K has no impact on the processing time of $\hat{F}_{t}^{AB}$. On the contrary, the processing time of $F_{t}^{AB}$ grows linearly with K. However, $F_{2}^{AB}$ (gravitational force histogram) performs better than the $F_{0}^{AB}$ (constant force histogram), for the reason explained in section 2.3.1. Nonetheless, $\hat{F}_{0}^{AB}$ performs better than $F_{0}^{AB}$, and $\hat{F}_{2}^{AB}$ performs better than $F_{2}^{AB}$, unless the number of directions in which the forces are computed is small (e.g., K= 360).

The objects considered in Figure 4.3 represent the ideal case for $F_{t}^{AB}$ with respect to processing time (Section 2.3.1). Any line that runs through both objects intersects each object once, therefore, computing a force histogram value comes down to computing the forces between two segments.
Figure 4.4: Processing times (T) of force histograms with respect to K (number of directions for which the forces are computed). The object A and B are from Figure 4.3, N= 512^3. (a) Effect of K on processing time (T) for constant force histogram (t=0). (b) Effect of K on processing time (T) for gravitational force histogram (t=2).

4.4 Effect of Size and Relative Position

4.4.1 Test Data

In this experiment, two object pairs are considered to evaluate the computational performance of $\hat{F}^{AB}_t$ and $F^{AB}_t$, taking into account the relative position of objects. The first pair is composed of right cylinders generated from disjoint quadratic Koch island fractals at iteration 3. The second pair is composed of right cylinders generated from overlapping quadratic Koch island fractals. The number of directions is kept constant: $K = 1800$. 
Figure 4.5: Test Data: The crisp objects used in this experiment are right cylinders whose bases are quadratic Koch island fractals at iteration 3. (a) Disjoint bases, (b) Overlapping bases

4.4.2 Results and Discussion

Figure 4.6, shows that the relative position of the objects has no impact on the processing time of $\hat{F}_{0}^{AB}$. However, computing $F_{0}^{AB}$ for overlapping objects exhibits a significant decrement in computational efficiency.

The object pairs considered in the Figure 4.5 represent the worst case scenario for $F_{0}^{AB}$. Contrary to the objects in Figure 4.3, any line that passes through both objects in Figure 4.5 intersects them in multiple segments due to the fractal nature of the objects, therefore, computing a histogram value comes down to computing the forces between a large number of segments, which leads to longer processing times.
over, in Figure 4.5b the objects overlap, consequently, a line that crosses both the objects can be found in every direction \( \vec{\theta}_i \) (i.e., \( F_{0}^{AB}(\vec{\theta}_i) \) is never 0 ), thus increasing the processing time.

![Processing times of force histograms](image)

Figure 4.6: Processing times of force histograms with respect to \( N \) (number of voxels in each bounding box). The objects are from Figure 4.5 with \( K = 1800 \). Processing time of \( \hat{F}_{0}^{AB} \) stays about the same and does not depend on whether the objects are disjoint or overlapping. Processing time of \( F_{0}^{AB} \) is much longer if the objects overlap.

### 4.5 Effect of Shape

#### 4.5.1 Test Data

In this experiment, the test data is pair of objects with multiple connected components. Each object is derived from a cube, which has been segmented into \( M \) identical
slices. The slices of both objects are aligned and interlocked as shown in 4.7. M is varied from 1 to 10 and N is set to $512^3$. The number of directions is set to $K = 1800$, and the type of forces is set to 0.

![Figure 4.7: Test Data: Segmented and Interlocking objects](file:///Users/jaspinderkaur/Desktop/graphs/segmented.svg)

4.5.2 Results and Discussion

The effect of the number of segments (M) on the processing times of $F_{t}^{AB}$ and $\hat{F}_{t}^{AB}$ is demonstrated in Figure 4.8. The number of segments has no influence on the processing time of $\hat{F}_{t}^{AB}$.

However, it has a substantial impact on the performance of $F_{t}^{AB}$. Processing times for M=1 (which is the ideal case for the line-based algorithm) and M=10 differ substantially, for the reason that for M=10, any line that crosses through both objects intersects each object in multiple segments, and more time is needed to compute the forces between all the segments.
4.6 Comparing Histogram values

This experiment demonstrates that $F_{t}^{AB}$ and $\hat{F}_{t}^{AB}$ are almost identical, as expected. The histogram associated with the $F_{t}^{AB}$ is denoted as $h_{1}(\theta)$ and the histogram associated with $\hat{F}_{t}^{AB}$ is denoted as $h_{2}(\theta)$. The unlikeness between $h_{1}(\theta)$ and $h_{2}(\theta)$ is measured using Tversky index ($\mu_T$). The $\mu_T$ value will be high when $h_{1}(\theta)$ and $h_{2}(\theta)$ histograms have more resemblance with each other. The $\mu_T = 1$, when $h_{1}(\theta)$ and $h_{2}(\theta)$ histograms are exact similar. The $\mu_T = 0$, when $h_{1}(\theta)$ and $h_{2}(\theta)$ histograms are orthogonal.

Mathematically, $\mu_T$ is given by the Equation (20) below:
\[
\mu_T(h_1, h_2) = \frac{\sum_{i=0}^{K-1} \min\{h_1(\theta_i), h_2(\theta_i)\}}{\sum_{i=0}^{K-1} \max\{h_1(\theta_i), h_2(\theta_i)\}}
\] (20)

where K is the number of directions in which the forces are computed.

Table 4.1: Tversky Index (\(\mu_T\)), \(N = 512^3\), K=1800

<table>
<thead>
<tr>
<th>Figure</th>
<th>4.1</th>
<th>4.3</th>
<th>4.5a</th>
</tr>
</thead>
<tbody>
<tr>
<td>t= 0</td>
<td>0.91</td>
<td>0.956</td>
<td>0.86</td>
</tr>
<tr>
<td>t= 1</td>
<td>0.90</td>
<td>0.950</td>
<td>0.82</td>
</tr>
<tr>
<td>t = 2</td>
<td>0.87</td>
<td>0.943</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Figure 4.9: (a) $\mu_T$ versus number of voxels (N), the objects are from Figure 4.5a, and $K = 1800$. (b) $\mu_T$ versus number of directions (K). The objects are from Figure 4.5a, and $N = 512^3$

The $\mu_T$ values tends to be close to 1 for the object pairs considered in our experiments (Table 4.1). The effect of K and N has also been studied on the $\mu_T$ (Figure 4.9). There is no significant impact of K on the $\mu_T$ (Figure: 4.9b), while Figure 4.9a shows that the $\mu_T$ grows notably and tends towards 1 when N increases.

## 4.7 Conclusion

The processing time of $\hat{F}_{i}^{AB}$ is independent of the number of directions in which the forces are considered. This is because most of the time is being spent on the computation of the spatial correlation ($\psi^{AB}$) between the objects. However, processing
time of $F_{t}^{AB}$ varies linearly with the number of directions.

Another significant advantage of $\hat{F}_{t}^{AB}$ is that its processing time is independent of the relative position (disjoint or overlapping) of the objects, and shape (simple or complex of the objects. On the other hand, there is a significant increase in the processing time of $F_{t}^{AB}$ when the objects have complex shapes (fractal like structures or objects with multiple disconnected components) because then forces need to be computed in large number of segments.

The processing time of $\hat{F}_{t}^{AB}$ is also independent of the types of forces considered, while the processing time of $F_{t}^{AB}$ is shorter when $t=1$ or $t=2$, and longer otherwise.

Overall, the processing time of $\hat{F}_{t}^{AB}$ is usually much shorter than that of $F_{t}^{AB}$, and the FFT- based algorithm is therefore the appropriate choice over the line-based algorithm unless the number of directions is small, the size of the object is small, or the shape of the object is simple.
Chapter 5

Conclusions and future work

The force histogram is a relative position descriptor (RPD) that encapsulates spatial relationship information (especially directional relationship information) regarding two spatial objects. We have extended the FFT-based algorithm for force histogram computation from 2D raster data to 3D raster data (Chapter 3), and then conducted several experiments to compare its computational performance with the line-based algorithm (Chapter 4).

5.1 Summary

We have provided a comprehensive review of relative position descriptors in Chapter 2. Examples are the angle histogram, the force histogram and the $\Phi$ descriptor. The angle histogram is one of the oldest RPD found in the literature, but suffers from many weaknesses including long processing times, inability to handle vector data, sensitivity to shape, size and orientation of the objects, etc. The $\Phi$ descriptor is the one of most recent and powerful RPD due to its ability to encapsulate a large amount of spatial relationship information between the objects. The main focus of our work was the force histogram. The force histogram has been the centre of attention among researchers for various algorithmic and practical applications.
Two potential algorithms for the computation of the force histogram between the objects were discussed. The line-based algorithm was implemented for both 2D and 3D raster data. The line-based algorithm works effectively for simple objects’ shapes, but its computational performance considerably decreases for more complex shapes (e.g., objects with multiple disconnected components or fractal objects). The computational time is also considerably affected by number of reference directions (K) considered, type of forces (t), and the relative position of objects. The FFT-based algorithm, however, was only implemented for the 2D raster data, but not for the 3D raster data.

The focus of Chapter 3 is to extend the FFT-based algorithm for 3D raster data. The force histogram was computed from the spatial correlation defined over the discrete 3D space.

Chapter 4 presented the test data and the several experiments conducted to compare the computational performance of the line-based algorithm with the FFT-based algorithm for force histogram computation. The test data include simple objects such as cubes, and complex objects such as objects with multiple disconnected components or fractal objects. The different experiments studying the effect of number of directions (K), type of forces (t), shape (convex, concave), size (N), and relative position (disjoint or overlapping) of objects were conducted to compare the processing time for both algorithms. A further experiment was also performed to verify whether the histogram computed
using line-based algorithm \((F^A_B(t)(\vec{\theta}))\) is equivalent to the histogram computed using FFT-based algorithm \((F^A_B(t)(\vec{\theta}))\). The likeliness between the \((F^A_B(t)(\vec{\theta}))\) and \((F^A_B(t)(\vec{\theta}))\) is measured using the Tversky index.

5.2 Conclusions and Future Work

The line-based algorithm for force histogram computation in the case of 3D raster data is computed in \(O(KN\sqrt{N})\) time, where \(K\) is the number of directions in which the forces are calculated and \(N\) is the number of voxels. However, the FFT-based algorithm introduced in this thesis is computed in \(O(N\log N)\) time. Furthermore, processing times are independent of the number of directions \((K)\), type of forces \((t)\), shape and relative position of the objects. The FFT-based algorithm outperforms the line-based algorithm for almost all objects (e.g., convex, concave, disjoint or overlapping), except when the objects’ shapes are simple or when the number of directions in which the forces are computed is small.

In future, the \(\phi\) descriptor algorithm can be extended for 3D raster objects, and then the processing times of force histogram and \(\phi\) descriptor for 3D raster objects could also be compared.
Bibliography


