Isometry Registration among Deformable Objects, A Quantum Optimization with Genetic Operator

by

Hamid Hadavi

Thesis submitted to the Faculty of Graduate and Postdoctoral Studies in partial fulfilment of the requirements for the MCS degree in Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering, Faculty of Engineering, University of Ottawa

Abstract

Non-rigid shapes are generally known as objects whose three dimensional geometry may deform by internal and/or external forces. Deformable shapes are all around us, ranging from protein molecules, to natural objects such as the trees in the forest or the fruits in our gardens, and even human bodies. Two deformable shapes may be related by isometry, which means their intrinsic geometries are preserved, even though their extrinsic geometries are dissimilar. An important problem in the analysis of the deformable shapes is to identify the three-dimensional correspondence between two isometric shapes, given that the two shapes may be deviated from isometry by intrinsic distortions. A major challenge is that non-rigid shapes have large degrees of freedom on how to deform. Nevertheless, irrespective of how they are deformed, they may be aligned such that the geodesic distance between two arbitrary points on two shapes are nearly equal. Such alignment may be expressed by a permutation matrix (a matrix with binary entries) that corresponds to every paired geodesic distance in between the two shapes. The alignment involves searching the space over all possible mappings (that is all the permutations) to locate the one that minimizes the amount of deviation from isometry. A brute-force search to locate the correspondence is not computationally feasible. This thesis introduces a novel approach created to locate such correspondences, in spite of the large solution space that encompasses all possible mappings and the presence of intrinsic distortion.

In order to find correspondences between two shapes, the first step is to create a suitable descriptor to accurately describe the deformable shapes. To this end, we developed deformation-invariant metric descriptors. A descriptor constitutes pair-wise geodesic distances among arbitrary number of discrete points that represent the topology of the non-rigid shape. Our descriptor provides isometric-invariant representation of the shape irrespective of its circumstantial deformation. Two isometric-invariant descriptors, representing two candidate deformable shapes, are the input parameters to our

optimization algorithm. We then proceed to locate the permutation matrix that aligns the two descriptors, that minimizes the deviation from isometry.

Once we have developed such a descriptor, we turn our attention to finding correspondences between non deformable shapes. In this study, we investigate the use of both classical and quantum particle swarm optimization (PSO) algorithms for this task. To explore the merits of variants of PSO, integer optimization involving test functions with large dimensions were performed, and the results and the analysis suggest that quantum PSO is more effective optimization method than its classical PSO counterpart. Further, a scheme is proposed to structure the solution space, composed of permutation matrices, in lexicographic ordering. The search in the solution space is accordingly simplified to integer optimization to find the integer rank of the targeted permutation matrix. Empirical results suggest that this scheme improves the scalability of quantum PSO across large solution spaces. Yet, quantum PSO's global search capability requires assistance in order to more effectively manoeuvre through the local extrema prevalent in the large solution spaces. A mutation based genetic algorithm (GA) is employed to augment the search diversity of quantum PSO when/if the swarm stagnates among the local extrema. The mutation based GA instantly disengages the optimization engine from the local extrema in order to reorient the optimization energy to the trajectories that steer to the global extrema, or the targeted permutation matrix.

Our resultant optimization algorithm combines quantum Particle Swarm Optimization (PSO) and mutation based Genetic Algorithm (GA). Empirical results show that the optimization method presented is scalable and efficient on standard hardware across different solution space sizes. The performance of the optimization method, in simulations and on various near-isometric shapes, is discussed. In all cases investigated, the method could successfully identify the correspondence among the non-rigid deformable shapes that were related by isometry.

Acknowledgement

I would like to express my gratitude to Dr Eric Paquet and Dr Herna Viktor for supervising this thesis. I never forget their kind receptions in numerous visits I made to their offices at the National Research Council and the University of Ottawa, the stimulating conversations, and their support throughout this thesis project.

I am also grateful to Dr Tony White and Dr Eric Dubois for the comments and corrections with respect to this thesis.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

In the domain of three dimensional shape analysis, similarity refers to the quantitative measurement of "distance" between two shapes. If the distance is small, then we may conclude that the two shapes are similar [1]. The significance of similarity arises in applications such as shape registration, shape recognition, shape indexing, object retrieval, shape deformation and morphing analysis, symmetry analysis, and so on. The similarity, or correspondence, analysis assumes much more complexity within the domain of non-rigid deformable objects. In this latter case, the very same object may be transposed from one pose to another. In other word, the non-rigid three-dimensional object may assume many forms as a result of its deformations. For example, consider a football that has been half or fully inflated. In both cases, the very same object is deformed in different ways. An isometry analysis algorithm identifies and recognizes the same football irrespective of its circumstantial deformations.

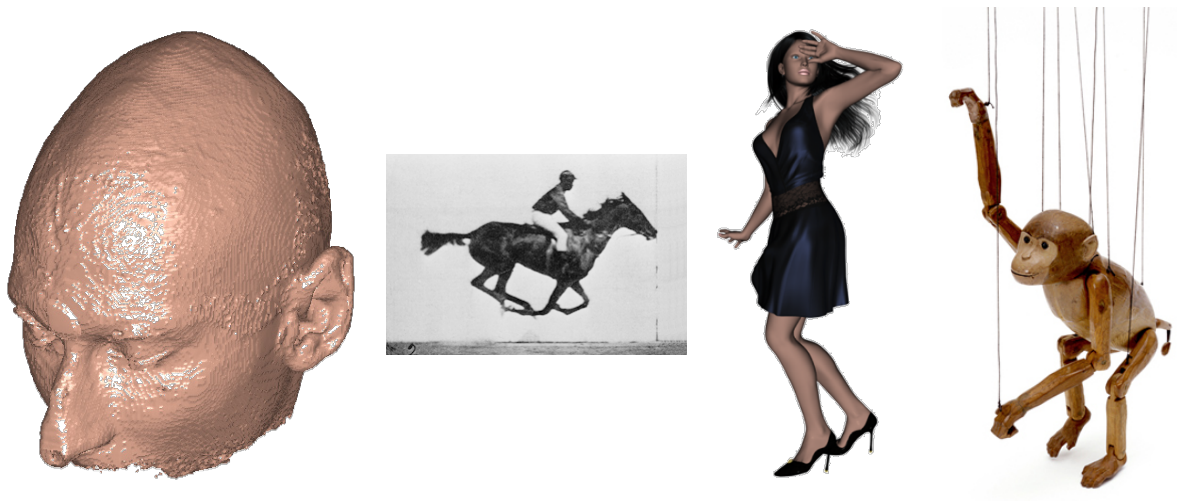


Figure 1-1 Medical Volumetric Data, Two Dimensional Shape, Computer Graphic Model, and Articulated Object [2]

Consider the items depicted in Figure 1-1. Depending on the circumstances, each item may be situated and posed in numerous different formations. Yet, irrespective of the pose and articulation, the intrinsic geometry of the objects remains invariant across various positions and/or articulation. A computational intelligence engine using the intrinsic

geometry of each encountered object may identify and may recognize the object irrespective of the pose or spatial articulation.

Moving to image and face analysis, human face may assume many expressions reflecting different emotions. Yet, in spite of the different facial expressions, they are all recognized as the different articulations of the emotions belonging to the very same face. In this case, different formations are isometric purporting the existence of invariant attributes irrespective of different articulation of the deformable object (human face).

Figure 1-2 Human face and varied emotional expressions [3]

Let us explore isometry/similarity within the field of computer games. An avatar centaur may assume different formations, two of which illustrated in Figure 1-3. Yet, irrespective of the formation, an adversary avatar should recognize its opponent centaur. In other word, isometry detection enables the game logic to recognize the centaur in whatever formation it may be, given the centaur's intrinsic geometry signature.

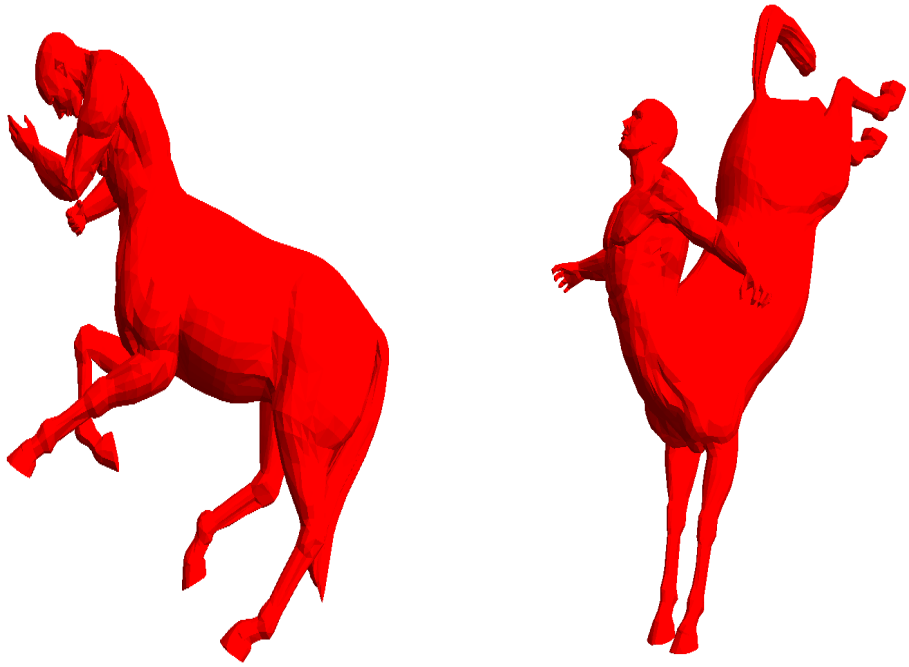


Figure 1-3 Two different poses of Centaur

In life sciences isometry detection is of crucial significance in pharmaceutical applications, pathology, and pathobiology. Many cases of isometry detection are explored and researched in bioinformatics and/or computational biology, for example, within the context of proteins and protein docking. Protein molecules constitute the primary vehicle that mediates the essential structures and functions in cellular life [4]. A distinct form of protein interaction with its environment constitutes the interaction with another protein molecule. Two protein molecules may interact and bind forming a distinctive structure known as quaternary structure [5]. The formation of quaternary structure, which is the apex in the hierarchy of protein structures, is the basis for many industrial and pharmaceutical applications. For example, in drug design a protein based drug may bind with a villain protein molecule in order to neutralize the latter [6].

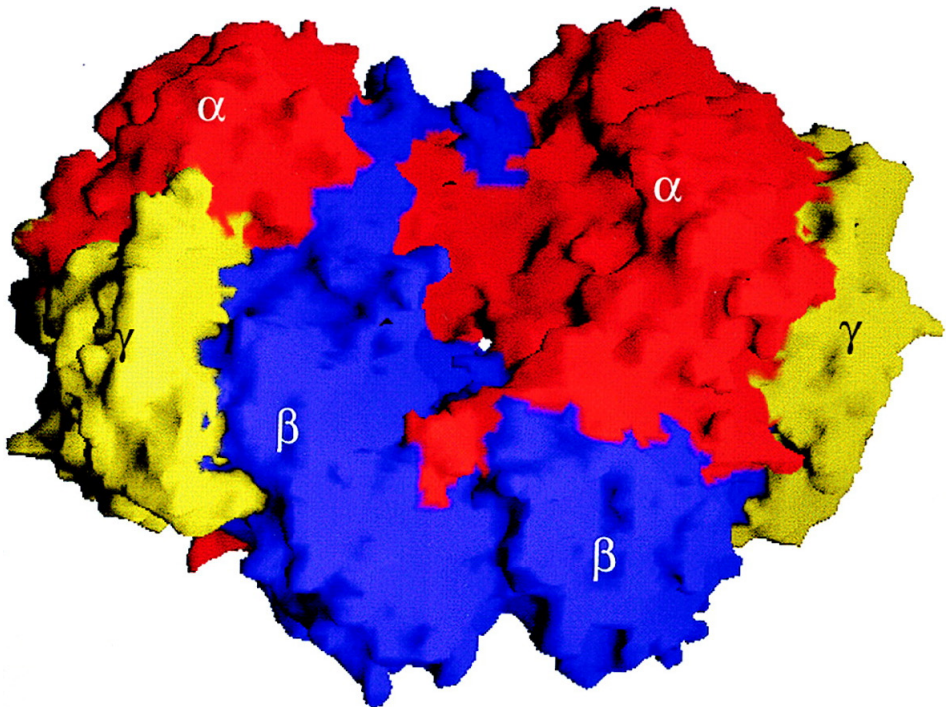


Figure 1-4 Example of protein docking [7]

The successful binding (or docking), however, is dependent upon the degree of geometrical compatibility and/or similarity between the binding sites (or regions) on the two molecules envelopes [4]. Hence, correspondence/isometric detection among the deformable protein structures may be the basis to formulate new pharmaceutical drugs.

1.1. Motivation

An effective algorithm that is able to measure and to register the similarity/isometry of the two deformable topologies becomes of acute utility. The quantitative measurement of the similarity must remain invariant to the deformations. In other words, two topologies may have similar or identical intrinsic geometries, and yet deformed in different ways resulting in shapes that are apparently dissimilar. This proves to be very challenging as three dimensional deformable objects have the prerogative to articulate themselves spatially with extremely large degrees of freedom.

An analogy in graph theory may better communicate the extent of the challenge. Assume a graph with n vertices. The vertices may be expressed by a set $V = \{v_1, v_2, ..., v_n\}$. The size of the space that encompasses all possible ways that V may be expressed correspond to the set of all permutations of the set V. Hence, there are $n!$ ways to articulate V. Consequently, if a deformable object is identified by a signature constituting of n coordinates, then there are $n!$ possible ways to rearticulate the object. As n enlarges, $n!$ grows to astoundingly large values.

n	$n!$
10	3,628,800
15	1,307,674,368,000
20	2,432,902,008,176,640,000
100	9.33262154439441526816992388562667E+157
200	7.88657867364790503552363213932 19E+374
300	3.06057512216440636035370461297 27E+614

Table 1.1 n and $n!$

The choice of n has direct impact on the resolution and accuracy of the representations. In many applications, such as Medical Volumetric Imaging, the resolution and the accuracy of the representation are crucial (Figure 1-5). In such cases, the topologies under considerations may more accurately be visualized with three dimensional shapes constituting with large number of vertices.

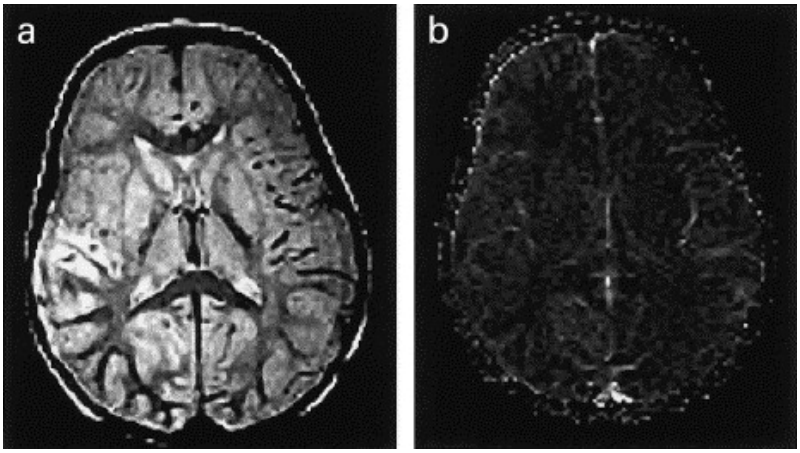


Figure 1-5 Medical Imaging Resolution [2]

Another domain where the choice of n is of significance is in computer graphics. For example, graphics may be rendered based on the hardware capabilities of the machine. In a machine with a powerful GPU, the program may render the graphics with high resolution. Higher resolution amounts to more tessellations, which means a graph with larger number of vertices (Figure 1-6).

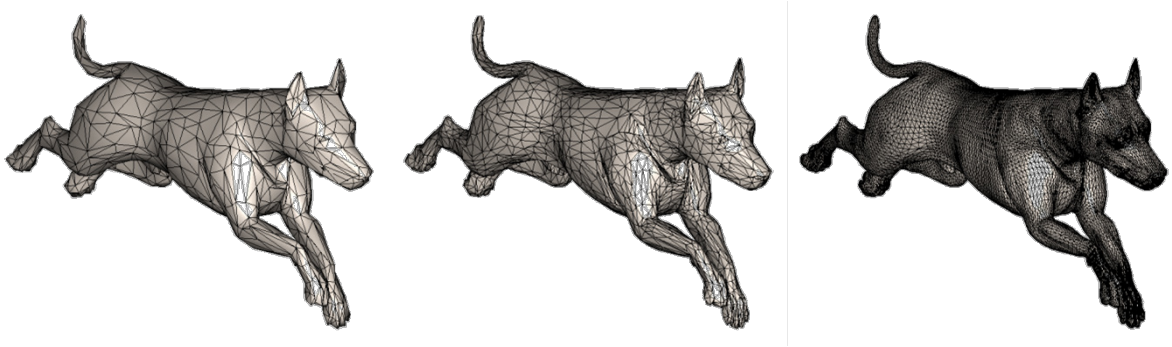


Figure 1-6 Multi-resolution graphics [2]

A deterministic approach to register similarity between two candidate topologies constitutes an exhaustive search through the entire $n!$ solution space to locate and to identify a unique permutation that satisfies the similarity criteria. Further, the approach must also be scalable for different sizes of n, from small n to large n. For large values of n the complexity of such brute force search space overwhelms the computational resources of most digital systems except for massive parallel computational farms. Hence, a deterministic solution involving a brute force strategy may not be a viable proposition.

Yet, the most significant challenge is the fact that the solution space can never be homogeneous. The three dimensional deformable shape has the prerogative to articulate itself arbitrarily with large degrees of freedom. A region on the deformable shape may be concave up surrounded by neighborhood regions that are concave down. A region may be expressed with higher resolution (a greater number of vertices) than its immediate neighbors. Thus, any method to register and/or to detect similarity among deformable objects may not rely on some preconceived assumptions about the homogeneous distribution of the vertices. The algorithmic approach to register similarity should tailor its strategy to the nuances of its solution space. As the algorithm learns about the solution

space, it must also adapt its strategy to navigate its exploration for the optimal solution given the peculiarities of the solution space.

Finally, the algorithm to register similarity among deformable objects should be computationally efficient. In this respect, efficiency signifies not only processing time to register similarity but also complexity of the control parameter by which the algorithm operates. Too many control parameters may hinder algorithm's portability across various platforms, or from one size n to another. The complexity of fine-tuning control parameters should not overshadow the ultimate objective to register similarity. Ideally, the algorithm should self-adjust the values associated with its control parameters without any user intervention.

Optimization is a technique most appropriate for problems with a non-convex, non-polynomial solution space [8]. It is a methodology premised upon two fundamental ingredients, namely adaptation and purpose. It is a process that advocates modification and change in order to navigate to its goal and purpose. The goal may be in the form of maximization/minimization of some numerical function. In other words, optimization is a teleological processes oblivious to how and yet mindful of what [9]. The significant caveat here is the fact that optimization process fulfils its specified goal with some degrees of approximation. The process, nevertheless, is expected to fall within the acceptable error boundaries (or distortion).

Particle swarm optimization (PSO) is a stochastic optimization algorithm based on the emergence of collective intelligence behavior among individual members of a population. The algorithm employs a population perspective wherein the population or the swarm moves stochastically in the solution space [10]. As the swarm navigates the solution space, it accordingly learns about the solution space, and it modifies and reorients its global perspective in pursuance of its objective. The individual members of the swarm are in constant communication sharing their individual perspectives so that the swarm may form a global perspective towards the fulfilment of the optimization's objective. The

swarm, therefore, exhibits an adaptive social network with the sole objective of convergence to attain a goal. The goal, nonetheless, is always formulated in the form of maximization/minimization of some function.

In swarm optimization, the navigation of the particles in the solution space is modeled by the laws of physics. There are two variants of physics-based swarm intelligence. In the first variants, the swarm and its members are governed by the Newtonian dynamics. In the second, quantum mechanics laws are the governing enterprise. In classical PSO, however, the collective behavior of the swarm is choreographed by fine-tuning of the control parameters. Proper selection of the control parameters renders a highly effective collective intelligence to navigate the solution space in order to realize the optimization objectives. Nevertheless, in very complex domain models, the proper fine tuning of the control parameters becomes a mini optimization problem on its own. On the other hand, in quantum PSO, certain well established assumptions about the particles' potential energy permits the incorporation of bounded regions, known as *potential wells*, in the solution model, which eliminates the proliferation of control parameters and their associated complexities in classical PSO. In fact, quantum PSO has only a single control variable.

In registering similarity for non-rigid deformable objects, we already argued that, given the factorial size of the solution space, a deterministic brute force search is not viable. Therefore, optimization appears to be the more plausible route to pursue. In swarm optimization, however, the stochastic navigation and the adaptability of the swarm to a heterogeneous solution space are of notable significance. The inherent parallel processing realized by the members of the swarm when collectively searching the solution space renders this optimization method very attractive. The swarm's social network collectively searches, samples, and surveys the solution space in parallel. The constant communication among the social network, and the continually evolved individual and collective perspectives enable the swarm to re-evaluate and readapt its search strategy in the heterogeneous space to locate an optimal solution.

Another class of evolutionary computing known as Genetic Algorithms may also render a parallel processing approach towards the optimization goals. The perforce driver in this class of computing is the emulation of evolutionary theories in biological systems. Genetic algorithms have been employed for search strategies to solve multi-dimensional, multi-objectives, and multi-constraints problems that are also frequently slow, complex, and extremely difficult to articulate and to formulate using conventional and analytical techniques [11, 12]. Like swarm intelligence, a Genetic Algorithm is population based. However, a fitness function orchestrates the procreation of future generations whose individual offspring are more adept in fulfilling the optimization objectives in each successive generation.

1.2. Prior Work

Elad and Kimmel in [13] used an isometric-invariant comparison of deformable objects based on Euclidean embedding. Their algorithm maps the intrinsic geometry of the deformable shapes into a common Euclidean space of lower dimensionality in which the two shapes are compared. As a result of the embedding, distortion may occur which means that the accuracy of the representation may be compromised.

Dubrovina and Kimmel in [14] developed a framework that used surface descriptors based on the Eigen functions of the Laplace-Beltrami operator. In this framework, they introduce a matching cost function which is based on the permutation matrix. The correspondence in between two shapes is found by minimizing the matching cost function with quadratic integer programming.

Gromov in [15] introduced the concept of Gromov-Hausdorff distance that measures the intrinsic similarity, in metric spaces, between two deformable objects, represented by two sets of point clouds. The computation of the distance is a combinatorial problem of factorial complexity involving the evaluation of all resultant permutations. Memoli and Sapiro in [16] provided a theoretical probabilistic framework for the estimation of the Gromov-Hausdorff distance. Bronstein further developed this approach in [17] using

Generalized Multidimensional Scaling in order to calculate the Gromov-Hausdorff distance.

1.3. Contribution

In this thesis, we present an algorithm between Quantum Particle Swarm Optimization and Genetic Algorithm for similarity registration among non-rigid deformable three-dimensional objects. As we will show, the symbiosis between the Quantum Particle Swarm Optimization and Genetic Algorithm realizes a novel, uncomplicated, efficient, and highly scalable algorithm by which similarity among various three dimensional shapes, represented by up to 1000 vertices, may be successfully registered and established.

We present analysis of the performances of the variants of quantum PSO vs. classical PSO. The analysis is based on conducting simulations utilizing PSO variants for the optimization of the test functions. The results of the simulations provide evidence to the superiority of quantum PSO in complex problems with large dimensionality.

We propose a novel representation of the solution space based on lexicographic ordering. Such a representation associates a unique integer value to each permutation matrix and allows to design a simplified and more efficient optimization algorithm for the cost function. The newly developed quantum PSO with lexicographic ordering improves scalability and performance when registering shapes related by an isometry.

1.4. Thesis Outline

The rest of this document is outlined as follow. In Chapter 2 we provide mathematical definitions and formalisms for geometry of non-rigid deformable shapes. The terms and definitions outlined in Chapter 2 will be utilized throughout the subsequent chapters. In particular, we will formalize the definitions for distance, similarity, correspondence, isometry, path, and surface signatures. Chapter 3 is dedicated to develop the mathematical models by which the isometry registration may be computed. We illustrate that isometry registration is computationally intensive. We show that a deterministic

registration of isometry requires a brute search in the solution space that is not computationally viable. Chapter 4 covers a thorough investigation of Optimization, Swarm Intelligence, Classical Particle Swarm Optimization, and Quantum Particle Swarm Optimization. We provide extensive experiment results and analysis to support the claim that Quantum Particle Swarm Optimization might be the more effective optimization approach in high dimensionality problems. In Chapter 5 we apply Quantum Particle Swarm Optimization (Quantum PSO) to the problem of isometry registration in three dimensional non-rigid deformable shapes. We provide extensive experiment results and analysis to show Quantum PSO effectiveness. The quantum PSO algorithm eliminates the need for any user intervention, as the control parameters are self-adjusted by the algorithm over the course of its operations. We analyse some of the challenges encountered, and argue for the advantages of symbiosis between the Quantum PSO and Genetic Algorithm. In Chapter 6, we develop a hybrid algorithm comprised of Quantum PSO and Genetic algorithm to register isometry among the deformable objects. We provide simulation experiments and their results to illustrate that the hybrid algorithm is not only effective to register isometries but also is highly scalable from small to very large non-rigid deformable three-dimensional objects. The algorithm is efficient as it delivers superior results on a regular CPU with no special configuration. The algorithm is straightforward to use, and adaptable to the characteristics of the solution spaces. In Chapter 7, we apply the hybrid algorithm to the three-dimensional shapes in order to register isometry on various sites located on the corresponding shapes. The results of our experiments and the analysis are also presented. In Chapter 8, we conclude the thesis with a summary of the contributions made, and a survey of future research and the applications that may be pursued.

Chapter 2 Geometry of Non-Rigid Deformable Objects

Non-rigid shapes are generally known as spatial objects whose three dimensional geometry may deform under internal and external influences [1]. Intrinsic influences are generally those pertinent to the physical and/or chemical properties of the deformable objects. For example, RNA molecules may fold and adopt complex three dimensional shapes as a result of the base nucleotide interactions [4]. Clouds, however, are deformed by forces of the extrinsic winds. Yet, the world around us abounds with non-rigid deformable objects.

Figure 2-1 The world around us abounds with deformable objects [2]

In all items in Figure 2-1, the geometry of the deformable objects change by forces internal and/or external. The analytical investigation of deformable and non-rigid shapes has fundamental applications in wide range of fields including computer graphics, image analysis, and/or pattern recognition. Examples may include face recognition [18, 19], matching of articulated objects [20, 21], and/or image segmentation [22]. A crucial dilemma among all the purported applications is the identification of the criterion by which the similarity and/or the distance among the articulated forms of the non-rigid objects is measured. The dilemma is pronounced by the fact that the measure of similarity

among the shapes must remain oblivious or invariant to the deformations. In other words, the criterion to measure similarity must find the two objects similar no matter what forms of objects are articulated. Further, the criterion must have metric property, discretizeable, computationally efficient, and scalable [1].

The objective of this chapter is to introduce mathematical concepts that will be utilized in the next chapters. In particular, mathematical definitions and tools pertinent to the geometry and similarity of non-rigid objects will be reviewed and investigated.

2.1. Similarity of Non-Rigid Shapes

The similarity of deformable objects refers to a quantitative measurement that quantifies the "distance" between the objects under consideration [1]. If the distance is small, then we may conclude that the objects are similar. On the other hand, if the distance exceeds a certain threshold, then the objects are non-similar. The similarity detection becomes challenging among non-rigid objects as the objects continually transform assuming divergent geometries. The face of an individual may assume different geometrical variations reflecting diverse emotional expressions; yet, in all cases the diverse facial expressions are all of the same face and of the same individual. An accurate criterion for similarity would identify all facial expressions as variations in the emotional expressions of the same individual. This could be only accomplished should the measure of similarity be based upon the criteria that are invariant to deformation. A measurement of the distance of the similarity between deformable objects must detect and compare the deformable invariant attributes. The deformable invariants are those attributes pertinent to the objects that do not undergo any variations while the objects deform and adopt new geometries [23]. Figure 2-2 depicts many articulations of the same object. The object may be articulated in infinite possible ways; yet, the criterion to detect similarity must be able to recognize the articulations as different geometrical expressions of the same object (or similar objects).

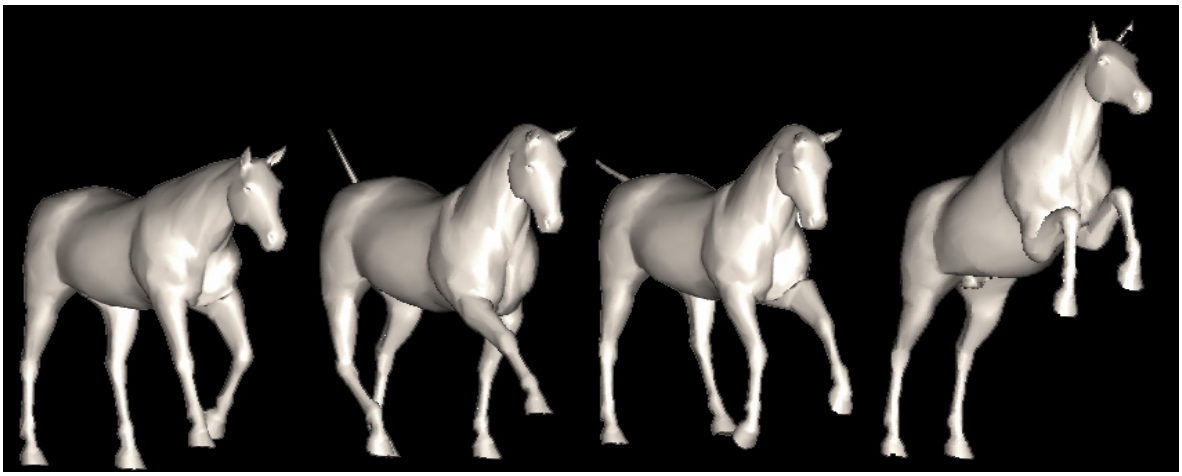


Figure 2-2 Articulations of the same deformable object [2]

2.2. Correspondence Problem

Another class of problem in deformable shape analysis is the *correspondence problem* [1]. When a jacket is put on and the hands are placed in the sleeves, an intensive computation is performed to correspond the hands-one set of the deformable objects, with the sleeves-the second set of deformable objects. In this case a measure of similarity is performed in order to detect the similarity between the geometry of the hands and the geometry of the sleeves. Again, in this case, to address the correspondence problem the implicit computation has relied on the measure of similarities between the deformable invariant attributes of the hands of the individual and the sleeves of the jacket [1].

2.3. Intrinsic Geometry

Similarity and Correspondence are two fundamental ingredients in deformable shape analysis. In both cases, the algorithm to measure similarity or correspondence relies on intrinsic geometries of the two objects-the objects may be articulated differently; nevertheless, the objects do not stretch or shrink, and accordingly the object's intrinsic geometry remains invariant. To put it differently, the shortest distance curve between any two points on the surface remain the same irrespective of how the object is articulated [1].

2.4. Metric Geometry

To formalize the definition of intrinsic geometry and the inherent concept of distance we need to distinguish Euclidean and Geodesic distances between any two points in $\mathbb{R}^n$.

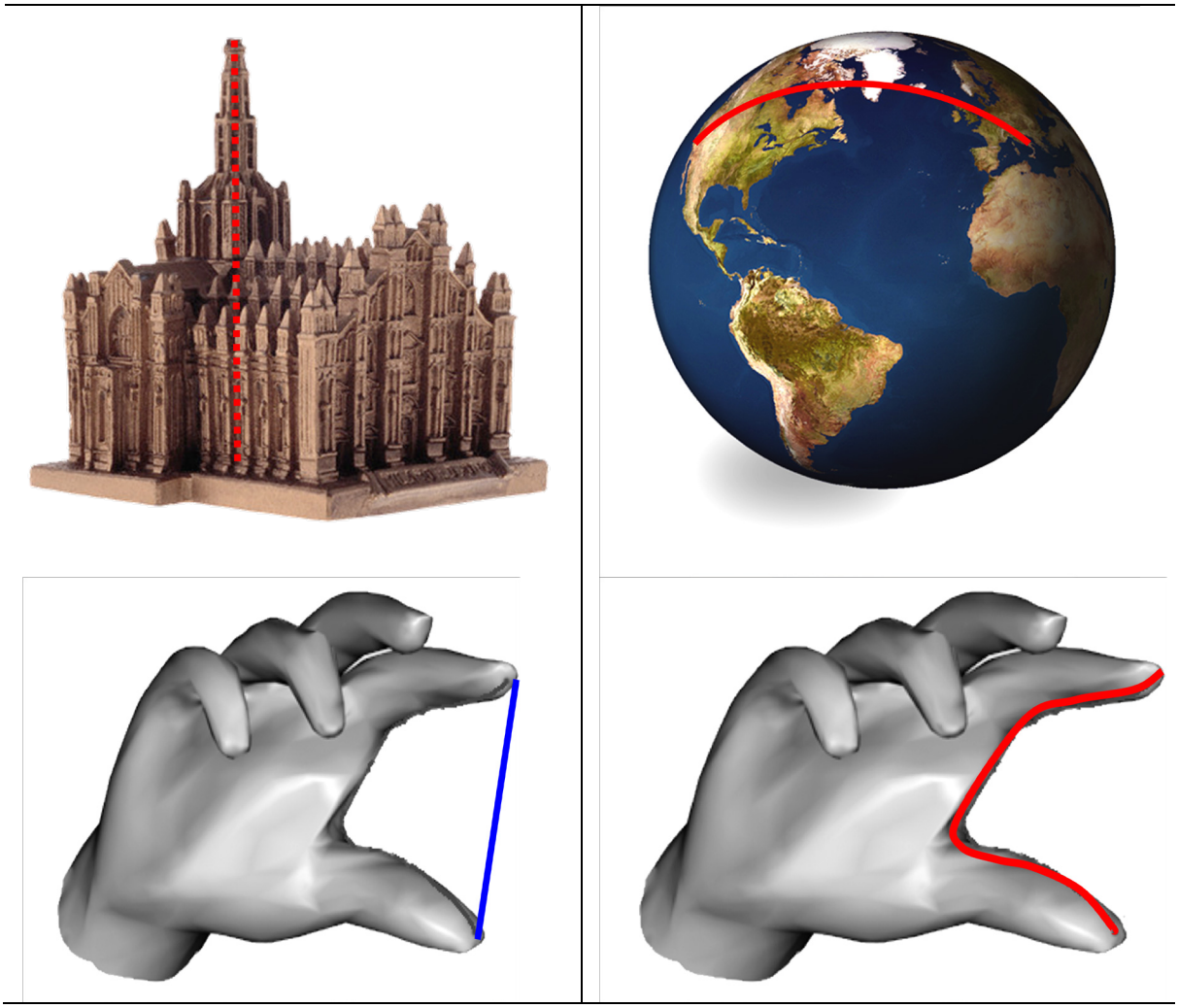


Figure 2-3 Euclidean (left) and Geodesic (right) distances in $\mathbb{R}^3$ [2]

Understanding the Euclidean distance is quite trivial given the intuitive Cartesian framework that the mind immediately perceives. In this framework, the distance is the magnitude of the straight line between any two points in $\mathbb{R}^n$. The geodesic distance, however, implies the magnitude of a curve between two points located on $\mathbb{R}^n$ [19]; both are particular instances of metric spaces known respectively as Euclidean and Riemannian

metrics [1]. In this research, we focus inclusively on the deformable objects whose surfaces are represented by mathematical formulation of a curve in $\mathbb{R}^3$. Hence, we need to formalize the concept of metric space in three dimensional space.

Formally, a set X is called metric space [1] if and only if the following holds true

$$
\begin{aligned}
&d : X \times X \to \mathbb{R} \\
&Non-negativity : \forall x_1, x_2 \in X \mid d(x_1, x_2) \geq 0 \\
&Indiscernability : d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2 \\
&Symmetry : d(x_1, x_2) = d(x_2, x_1) \\
&Triangle-inequality : d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)
\end{aligned}
\tag{2-1}
$$

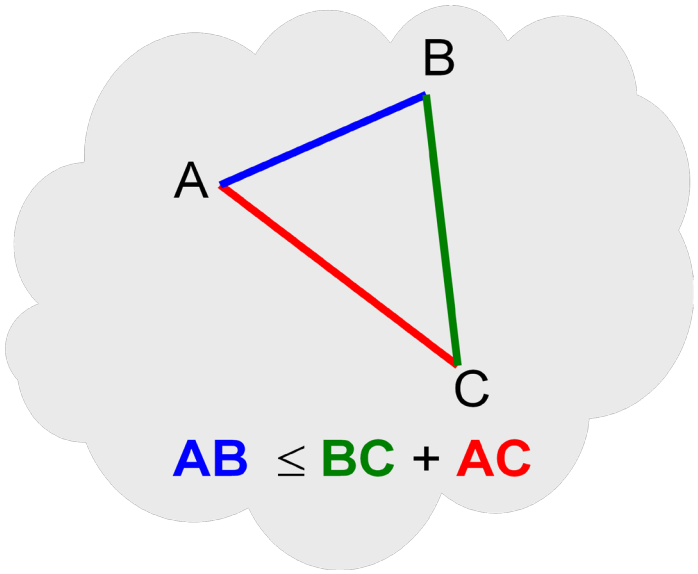


Figure 2-4 Triangle-inequality

In this definition, elements of X are called points and $d(x_n, x_m)$ represents the distance which is the magnitude of the path from x_n to x_m.

The notion of distance between any two points, however, must account for all the paths that may connect the two points. Consider two locations A and J on two different islands (Figure 2-5). A driver travelling between the two must take into consideration all the paths over all the bridges that connect the two locations. Hence, the distance may not necessarily be a straight line but a path that covers distance between the two. Further,

the two locations A and J should not be disjoint, and they may not belong to two disjoint sets (or locations).

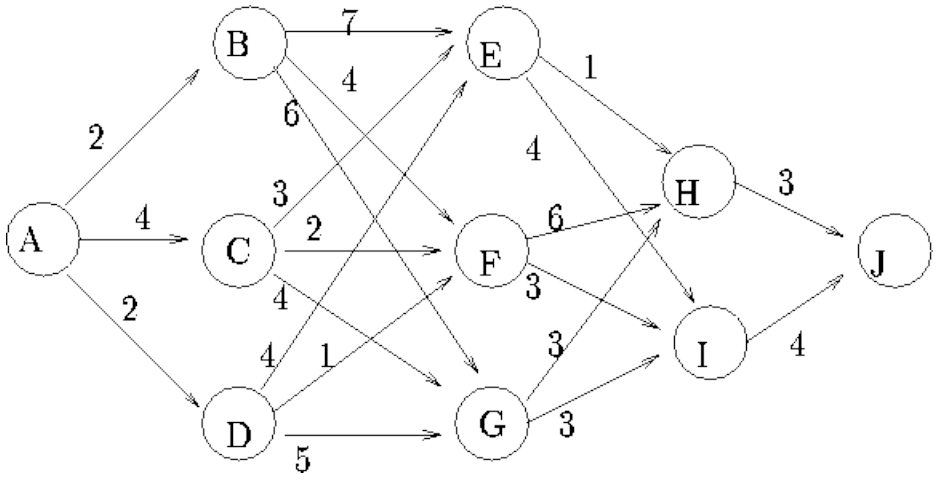


Figure 2-5 Many paths between points A and J [24]

In any case, the point to make is that the distances between two points on a deformable object may have to be represented by one or many paths. Accordingly, we require a function $L(\Gamma)$ that assigns a positive value to every path between the two points. We may formalize the notion of distance [1] by Equation (2-2) .

$$d_L(x_n, x_m) = \inf\{L(\Gamma) \to \Gamma : [a,b] \to X,\ \Gamma(a) = x_n,\ \Gamma(b) = x_m\} \tag{2-2}$$

The geodesic distance between any two point is the infimum[1] of lengths of all admissible paths that connect the two points.

2.5. Isometry

We utilize the metric spaces to formalize the concepts of similarity and correspondence among deformable objects. In this framework, we are often interested to measure the correspondence and similarity between the points on two metric [1] spaces (X, d_X) and (Y, d_Y).

[1] Suppose a subset S such that $S \subset T$ and T is an ordered set. The infimum is the greatest element in T that is less then or equal to all elements in S [43].

A function $f:X\to Y$ is said to be continuous [1] if for every two points (x_n,x_m) $f:X\to Y$ can map to their image $(f(x_n),f(x_m))$ in Y. Given the continuity, one may also conclude that

$$\forall A\subset Y, f^{-1}(A)=\{x\in X: f(x)\in Y\} \tag{2-3}$$

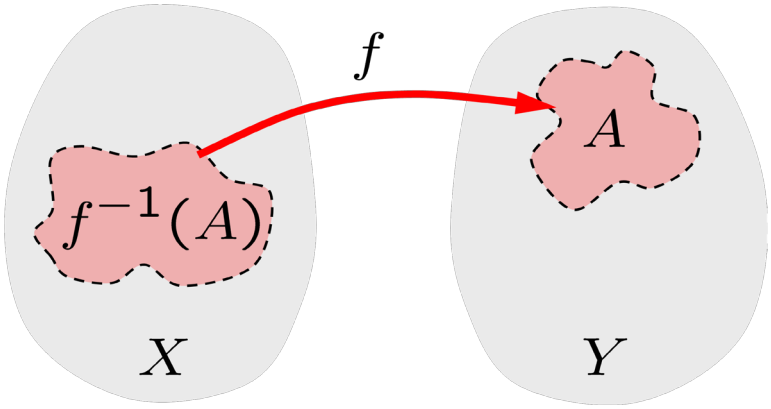


Figure 2-6 Continuous function f

A property stronger than continuity is described by *Lipschitz continuity* in Equation (2-4)

$$\begin{aligned}&\exists C\geq 0\ \&\ x_n,x_m\in X:\\ &d_Y(f(x_n),f(x_m))\leq C.d_X(x_n,x_m)\end{aligned} \tag{2-4}$$

In (2-4) when $C\leq 1$ then $f:X\to Y$ is non-expanding, and $C<1$ then $f:X\to Y$ is contracting. Further if $C=1$ then $f:X\to Y$ is distance preserving.

A stronger form of property is satisfied by $f:X\to Y$ in *bi-Lipschitz continuity* as [1] described in (2-5)

$$\exists C\geq 1: C^{-1}.d_X(x_n,x_m)\leq d_Y(f(x_n),f(x_m))\leq C.d(x_n,x_m) \tag{2-5}$$

Two metric spaces that are related by (2-5) when $C=1$ are said to be Isometric. Perfect Isometries rarely exist in the realms of deformable objects. Predominately, we may only encounter near Isometries, Figure 2-7, with a distortion equal to ε. The distortion ε is defined by Equation [1] (2-6)

$$\sup_{x_m, x_n \in X} |d_X(x_n, x_m) - d_Y(f(x_n), f(x_m))| < \varepsilon \tag{2-6}$$

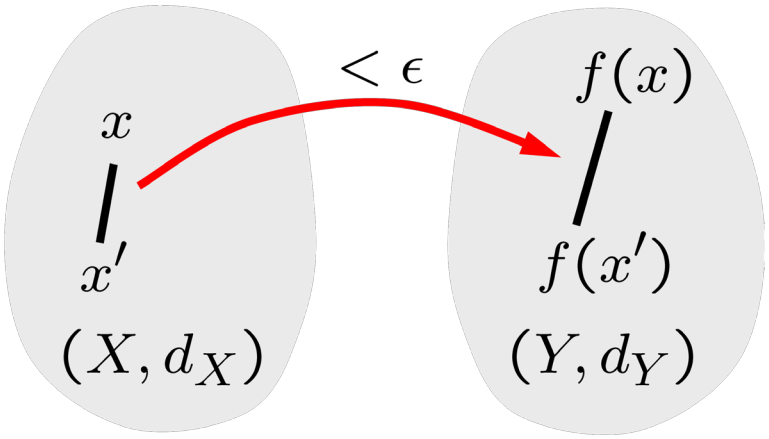


Figure 2-7 Distortion in Isometries

Isometric spaces are indistinguishable from the perspective of metric spaces. This property becomes immensely attractive as the criteria to establish similarity among deformable objects. In other words, in the metric spaces, all articulations in Figure 2-2 are isometric and indistinguishable from one another.

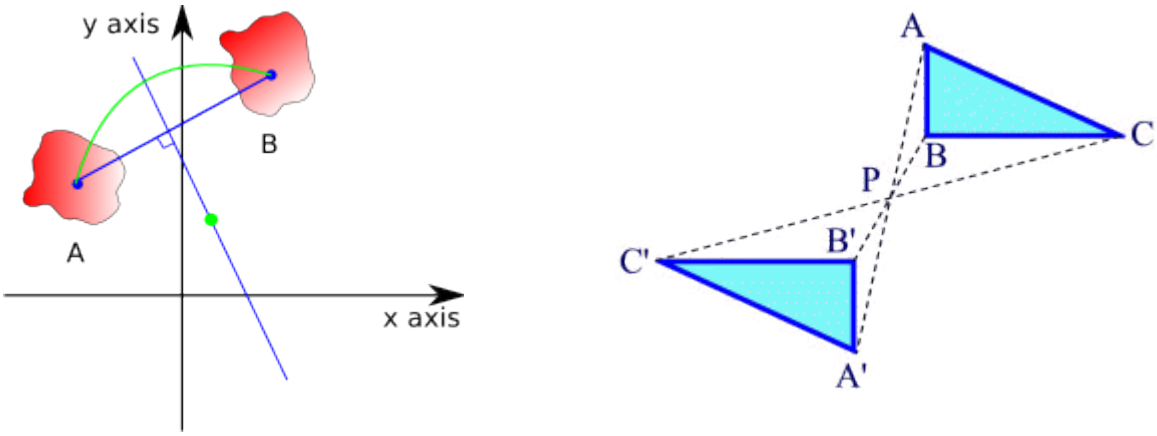


Figure 2-8 Isometries in 2D: rotations about an axis (left), reflection about a point (right)

Figure 2-8 illustrates two simple cases of isometries among the two dimensional objects. In the first case, a plane is rotated around an axis to generate a second plane that is isometric to the first. In the second case, a triangle is reflected about a point to produce an isometry. In both cases, it is trivial to verify the continuity (Equation (2-3)) and isometry (Equation (2-6)) in either shapes.

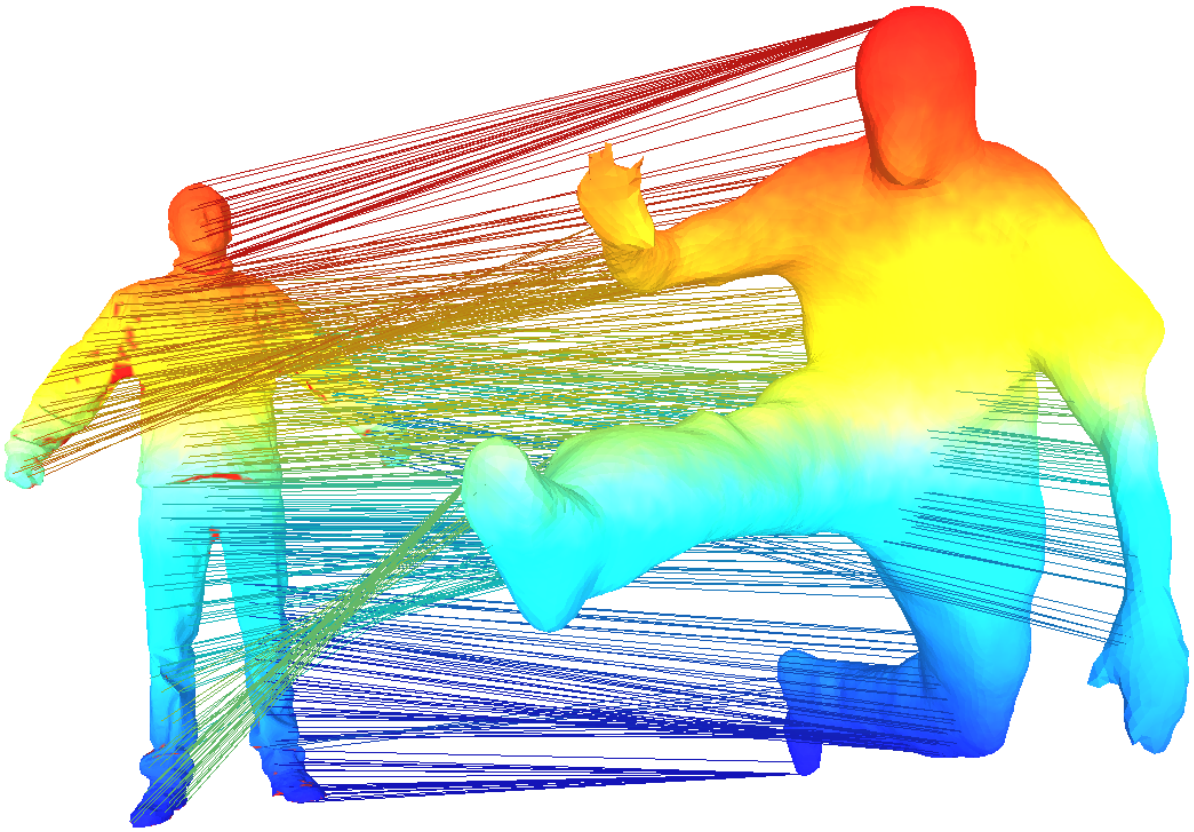


Figure 2-9 Isometry in three dimensional deformable object [2]

Yet, in more complex three dimensional isometric deformable objects (Figure 2-9), distance preservation (Equation (2-2)), continuity (Equation (2-3)), and bi-Lipschitz continuity (Equations (2-5),(2-6)) should also remain visible and verifiable.

So far, isometry was used to identify similarity among different articulations of the same object. For example, in Figure 2-9 the three dimensional posture of the same individual is expressed in two different forms. As shown, the two expressions are isometric, and in metric spaces the two are indistinguishable. Nevertheless, isometry may be used to register similarities among different parts or regions of very different three dimensional deformable shapes.

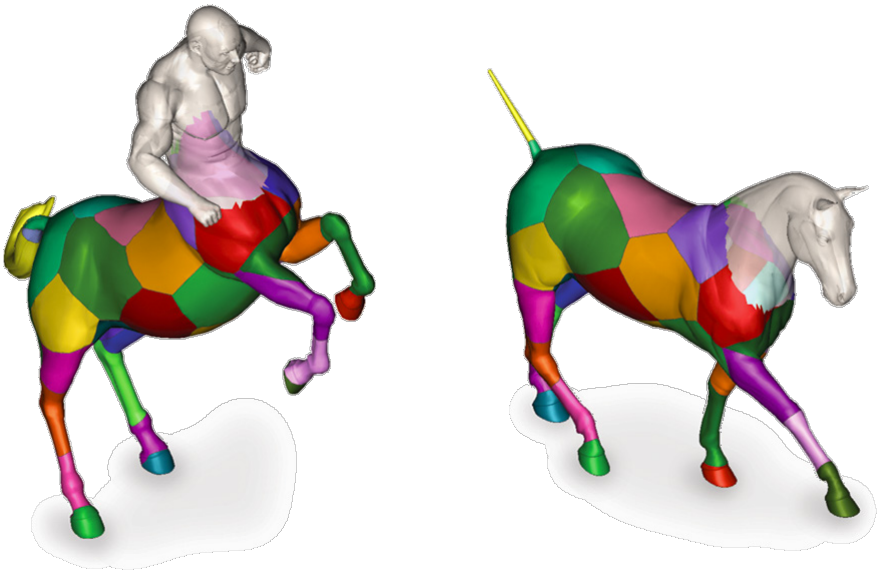


Figure 2-10 Partial Isometry | [2]

Consider, for example, Figure 2-10 or Figure 2-11. In both figures objects of different types share isometric regions. In both cases, objects are not fully isometric; but, the objects are partially isometric. Different neighborhoods or regions in each shape, that are in isometry, are expressed in corresponding colors.

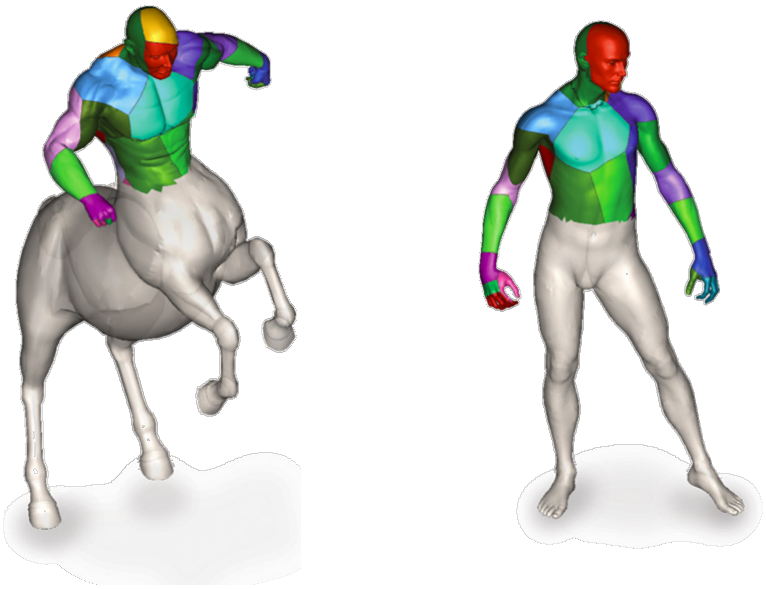


Figure 2-11 Partial Isometry || [2]

In general, the isometric regions may vary in size. For example, in Figure 2-12 there are small regions in the two very different three-dimensional shapes that are in correspondence.

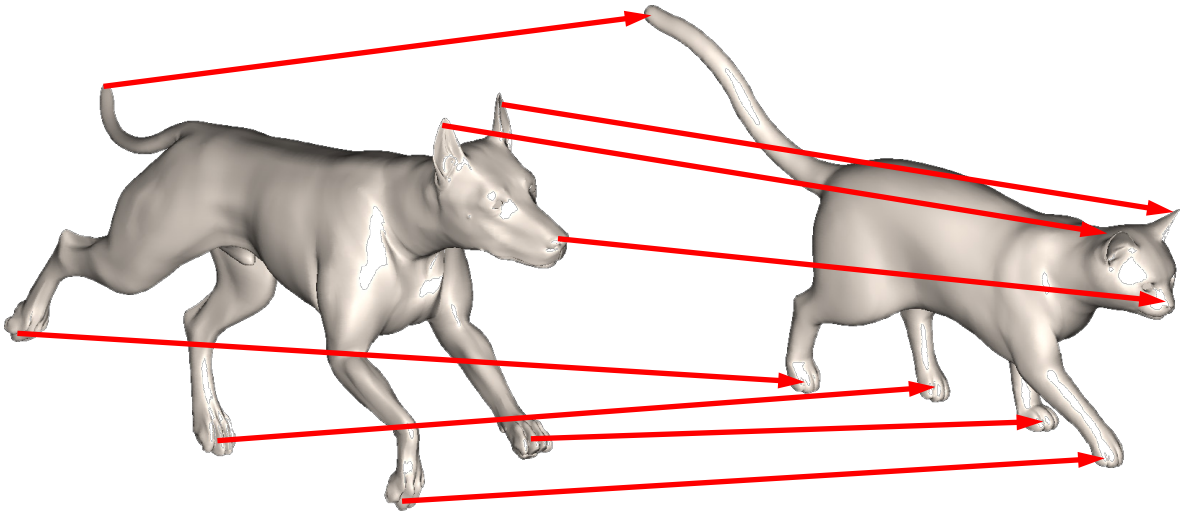


Figure 2-12 Correspondence in specific small regions [2]

2.6.Surface Representation

In shape and similarity analysis, a mechanism is required to represent the surfaces on the deformable objects. Given the objective to locate isometries between the two candidate surfaces, the representation must facilitate the algorithmic computations required in order to match the two surface representations. Point clouds are often used as approximate representation of the surfaces under consideration. The points are identified by their coordinates in three dimensional space. Further, the distances as defined in Equation (2-2) between every two points in the cloud are measured as precursor to search for isometric surfaces that satisfy the *bi-Lipschitz continuity in* Equations (2-5) and/or (2-6). In a sense, point cloud representations are discrete values of a combinatorial topology. Accordingly, the computational analysis of the shapes approximated by point cloud is within the realms of combinatorial mathematics. The point cloud may be formalized [1] by Equation (2-7)

$$X' = \{x_1, x_2, x_3, ..., x_n\} \mid X' \subseteq X \tag{2-7}$$

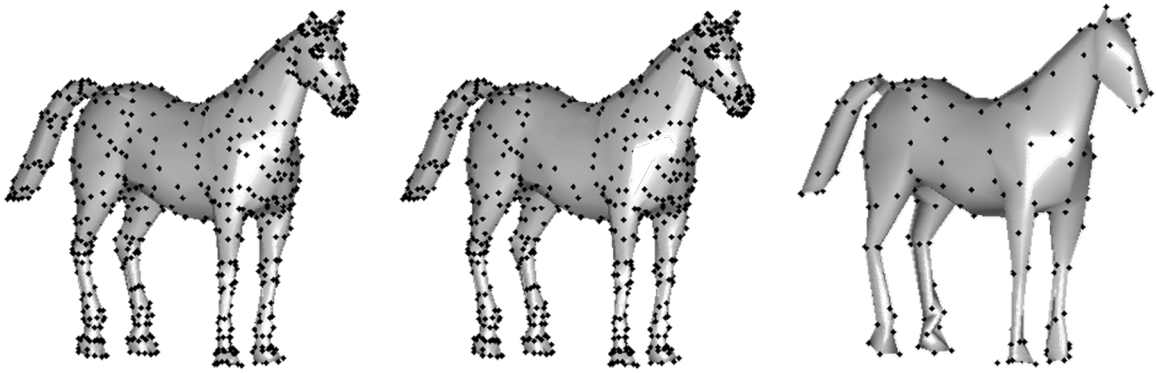


Figure 2-13 The dichotomy of size in point cloud representation. Larger n ensures accurate representation. Smaller n reduces computational complexity [2]

Naturally, the question arises as to how to choose the value for n. Intuitively larger values for n would result in a better representation of the surface (Figure 2-13). In practice, the underlying dichotomy to balance better representation against increased computational and storage requirements determines the choice of the value for $n.$ Further, the choice of n should ensure that no areas on the surface X remains underrepresented. Formally, it could be stated that n should be chosen such that $X' \subseteq X$ is r-covering of X. This may be formalized by Equation [1] (2.8).

$$\bigcup_{x' \in X'} B_r(x') = X \tag{2-8}$$

wherein B_r is a metric ball of radius r.

The set of distances among the points in point cloud may be represented by a matrix of size $n \times n$ whose diagonal constitute of all 0s. Figure 2-14 illustrates a hypothetical distance matrix.

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Figure 2-14 An example of distance matrix

2.7. Algorithms to Construct Distance Matrix

In the preceding sections, it was established that the measurement of distances among the vertices, landmarks, and/or point cloud that constitute the surface representation is fundamental in measuring and/or registering isometries. In outlining the surface representation, an approximation of the surface in discretized geometry is obtained. However, to detect isometry between any two surfaces, a mechanism to discretize the intrinsic distance among every two landmarks in the point cloud must also be established. In this context the intrinsic distance refers to the shortest curve that connects any two landmarks in the point cloud representation. Once the mechanism is developed, then a distance matrix (Figure 2-14) may be constructed, and utilized as input for algorithms to measure similarities and to register isometries. In fact, algorithms developed in the next chapters of this thesis require distance matrices as inputs to register isometries between the two candidate surfaces.

To formalize the definition of shortest path [1], the point cloud representation may be identified by a undirected graph (X, E) which has a length function $L : E \to \mathbb{R}$. The length between any two points may be given by the Euclidian distance $L(x, x') = \| x - x' \|$ for every two adjacent $(x, x') \in E$. A path between two points is the sequence of

$$\Gamma(x,x^{'}) = \{x_k\}_{k=1}^{K} \mid x_1 = x, x_K = x^{'}, (x_k, x_{k+1}) \in E \tag{2-9}$$

Given Equation (2-9), the length of a path between any two points may be formalized by Equation (2-10).

$$L(\Gamma) = \sum_{k=1}^{K-1} L(x_k, x_{k+1}) \tag{2-10}$$

Naturally, there may be many paths among any two points. However, the objective is to identify the shortest path among all the possible paths between any two points. This objective may be formalized by Equation (2-11).

$$d_L(x,x^{'}) = \min_{\Gamma} L(\Gamma(x,x^{'})) \tag{2-11}$$

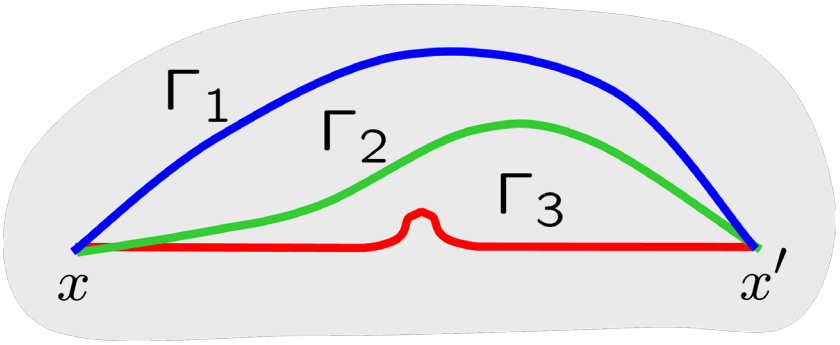


Figure 2-15 The shortest path realizing the length metric is geodesic distance

Dijkstra [25] proposed an algorithm to find the solution in (2-11). It works by calculating a distance for each point $x \in X$. Initially, the value $d(x) = 0$ and infinity for all other points. The algorithm then proceeds to attempt to improve $\forall x \in X$ as outlined in the pseudo-code [1] described in Table 2.1.

In addition to Dijkstra, there are other methods for calculating shortest path among the vertices of an undirected graph. Among them Fast Marching [26] algorithm and Floyd-Warshall [27] algorithm are noteworthy. For a thorough coverage, interested readers may refer to [1].

Dijkstra Algorithm

$Input: G(X,E),$ Length Function $L: E \to \mathbb{R},$
Source Point $x_0 \in X$
$Output:$ Distance Map $d: X \to \mathbb{R}$
Algorithm:
foreach $x \in X$
  $d(x) \leftarrow \infty$
end
$d(x_0) = 0$
$Q \leftarrow X$
While $Q \neq 0$
  $x \leftarrow \arg\min_{x \in Q} d(x)$
  foreach $x \in N(x') \cap Q$ do
    $d(x') \leftarrow \min\{d(x'), d(x) + L(x,x')\}$
  end
  Q ← Q\{x}
end

Table 2.1 Pseudo-Code For Dijkstra Algorithm [1]

2.8. Summary

In this chapter, we explored mathematical tools and definitions pertinent to the geometry and similarity of non-rigid objects. In particular, we introduced the definitions and formalization for a range of concepts including metric space, intrinsic and extrinsic geometries, surface representation, isometry and correspondence, distance matrix, and the shortest path problem in an undirected graph. These geometrical concepts are essential tools in studying the significance of the three dimensional architecture of deformable objects, and registering surface isometries or correspondence.

In Chapter 3, we utilize the mathematical tools explored in this chapter in order to introduce computational models for measuring similarities and registering isometries/correspondence among non-rigid deformable objects.

Chapter 3 Mining for Isometry in Deformable Objects

In Chapter 2, we introduced mathematical models and terminologies for registering isometries and correspondences among the surfaces of deformable objects. In this chapter, we expand upon the mathematical tools and terminologies from Chapter 2 in order to formalize the computational model by which surface isometry between two deformable objects may be registered. Ultimately, we are aiming for a model upon which optimization algorithms may be developed to formalize isometry registration among deformable objects.

3.1.Topology Description Methods

A computational model for comparing similarities and identifying isometry among deformable objects is premised upon an accurate and effective description of the candidate topologies. The topology description must account for the deformation invariant character of the objects in question. The descriptors can then be operated upon by the computational engines tailored to identify potential isometries or correspondences.

Liu, Fang, and Ramani [28] categorized the methods for topology comparisons into superimposition and descriptor/signature comparisons. In the former, two surface envelopes are computationally superimposed inside a grid box. Individual compartments of the grid are subsequently inspected to score a numerical value for the local similarities between the regional surfaces captured in each compartment. In the latter case a descriptor/signature represents a region on the surface. Descriptors/signatures are index-based and are captured in data sources. As such, they may be retrieved/queried for fast computational analysis in order to detect structural similarities, isometries, and so on. Paquet and Viktor [29] provide a comprehensive survey of different methods to denote

descriptors/signatures on protein molecular surfaces. The most widely used methods in shape analysis is the distance based method [28]. Distance based descriptor/signature is premised upon a complete graph constituting links that represent the weights or distances between every pair of landmarks (or vertices). The distances could be Euclidean or geodesic. The former nonetheless is sensitive to shape deformations whereas the latter is invariant to deformations.

3.2.Geodesic Distance Based Representation

The distance based representation views the topology as a collection of markers that are randomly placed on, or selected from, the object under the study. The markers may be arbitrarily positioned on the candidate surface, or the markers (or vertices) may be extracted from the structural elements within the three-dimensional shape. Once the vertices (or markers) are established then they may be clustered (recall section 2.6 for point cloud) together to compose a representative topology by which one or many segments of the surface are denoted. Finally, the geodesic distances among all pairs of the vertices are calculated using a shortest path algorithm (recall section 2.7). Accordingly, a distance matrix is produced wherein the distances among every paired vertices are correspondingly noted.

We may now formalize the definition of distance matrix in the context of this research. Let's assume a set of vertices (or markers) $V = \bigcup_{i=i}^{p} v_i$ on a given surface. The distance matrix (Figure 2-14) of dimension p is a square matrix D of size $p \times p$ consisting the shortest paths among all pairs $(v_n, v_m) \mid 1 \le n \le p, 1 \le m \le p,\ D[n,m] \in \mathbb{R},$ $D[n,m] = D[m,n],\ if\ (n = m) \Rightarrow D[n,m] = 0.0.$

To summarize, the distance matrix constitutes a surface signature composed of the geodesic distances among all pairs of the constitutive markers. The topology identified by the signature is invariant to deformation, and represents the candidate topology for isometry registration.

Vertices	V_1	V_2	V_3	$V_{...}$	V_{p-1}	V_p
V_1	0.00	$d(V_1,V_2)$	$d(V_1,V_3)$	$d(V_1,V_{...})$	$d(V_1,V_{p-1})$	$d(V_1,V_p)$
V_2	$d(V_2,V_1)$	0.00	$d(V_2,V_3)$	$d(V_2,V_{...})$	$d(V_2,V_{p-1})$	$d(V_2,V_p)$
V_3	$d(V_3,V_1)$	$d(V_3,V_2)$	0.00	$d(V_3,V_{...})$	$d(V_3,V_{p-1})$	$d(V_3,V_p)$
$V_{...}$	$d(V_{...},V_1)$	$d(V_{...},V_2)$	$d(V_{...},V_3)$	0.00	$d(V_{...},V_{p-1})$	$d(V_{...},V_p)$
V_{p-1}	$d(V_{p-1},V_1)$	$d(V_{p-1},V_2)$	$d(V_{p-1},V_3)$	$d(V_{p-1},V_{...})$	0.00	$d(V_{p-1},V_p)$
V_p	$d(V_p,V_1)$	$d(V_p,V_2)$	$d(V_p,V_3)$	$d(V_p,V_{...})$	$d(V_p,V_{p-1})$	0.00

Figure 3-1 Distance matrix of size $p \times p$

3.3. Topologies Representation

Recall in section 2.6 mathematical tools were introduced regarding topology representations in non-rigid deformable objects. As shown, a candidate topology may be represented by Equation (2-7) while satisfying Equations (2-8) and (2-9).

We define two topologies X and Y such that $X = \{x_1, x_2, x_3, ..., x_n\} \mid X \subseteq X^S$ and $Y = \{y_1, y_2, y_3, ..., y_n\} \mid Y \subseteq Y^S$ wherein X^S and Y^S constitute the entire surface envelopes of two different three dimensional shapes [1, 14]. Accordingly, X and Y are candidate topologies representing two distinct spatial regions on their respective structures. Both X and Y are two sets of markers (or vertices) wherein each marker of the sets represent the three dimensional coordinates of the corresponding vertex.

As an example, consider Figure 3-2 which contains two images for a face recognition application. Let us denote the image on the left by X^S and the image on the right by Y^S. Then, $X = \{x_1, x_2, x_3, ..., x_n\}$ constitutes the set of n markers on X^S, and $Y = \{y_1, y_2, y_3, ..., y_n\}$ constitute the set of n markers on Y^S. In section 2.6 general guidelines were given for factors to consider in choosing the value of n. Nevertheless, it is clear that the larger values for n would make more accurate representation of X^S and Y^S.

Figure 3-2 Topology Representations [2]

Given that different considerations may influence the choice for n, we may reiterate that the matching algorithm should be scalable for small, medium, and large values of n.

Having formulated X and Y, we may proceed to define X_d and Y_d wherein the former is the distance matrix for topology X, and the latter is the distance matrix for topology Y. Further, both X_d and Y_d contain inclusively the geodesic distances between every two paired vertices on X and Y respectively. Given that both X_d and Y_d are defined in terms of geodesic distances (recall sections 2.4 and 3.2), then both X_d and Y_d are isometric invariant signatures (recall sections 2.5, 3.2) for X and Y respectively.

3.4 Computational Model for Isometry Registration

Based on the methods developed by Dubrovina and Kimmel [14], we proceed to formulate a model by which an isometry between X_d and Y_d as defined previously is established. We assume that both X_d and Y_d are of the same size; that is, both X and Y

are sets of equal sizes. The two distance matrices, which are invariant under isometry, are related by a permutation matrix. The permutation matrix [14] is defined as

$$\begin{aligned} & x \in X_d,\ \mathrm{y} \in \mathrm{Y}_d \\ & P : X_d \times Y_d \to \{0,1\} : \\ & P(x,y) = \begin{Bmatrix} 1, \text{ x and y are in correspondence} \\ 0, \text{ otherwise} \end{Bmatrix} \end{aligned} \tag{3-1}$$

$$\sum_i P_{ij} = 1, \sum_j P_{ij} = 1, \quad \forall i, j \tag{3-2}$$

The cost function measures the discrepancy in between the two distance matrices once a permutation of the indices has been applied

$$| PX_d P^T - Y_d | \leq \varepsilon \tag{3-3}$$

By applying the permutation matrix to either of the distance matrices, the two distance matrices may become realigned forming near isometries whose distortion is less than or equal to ε. If X_d and Y_d are already aligned, then P is the identity matrix. Note that P in Equation (3-3) refers to the points (the indices) and not the distances. Either X_d or Y_d captures n^2 geodesic distances among n points. Consequently the $n \times n$ permutation matrix P relates the n points in X_d with the n points in Y_d. In Equation (3-3), the left P acts on the column space whereas the right P^T acts on the row space.

In Equation (3-3), the remapping has been applied to X_d; yet, the remapping may also be applied to Y_d in order to achieve the very same effect .

$$| P^T Y_d P - X_d | \leq \varepsilon \tag{3-4}$$

A stricter form of isometric alignment ensures that the distortions on both surfaces are individually accounted for. By accounting for the distortions on each surface individually, that is the application of the permutation matrix, we can align the two surfaces not only

by remapping X_d but also by remapping Y_d. Therefore, Equation (3-3) and (3-4) may be combined by accounting the two possible ways to remap the two surfaces.

$$| PX_dP^T - Y_d | + | P^TY_dP - X_d | \leq 2\varepsilon \quad (3\text{-}5)$$

We may call Equation (3-5) the equation to register isometry, or the similarity distance calculator, between two candidate surfaces X_d and Y_d. In other words, if there exists a permutation matrix P that satisfies Equation (3-5), then we may conclude that surfaces represented by X_d and Y_d are isometric.

For any given two topologies X_d and Y_d, the solution for the equation to register isometry (Equation (3-5)) is an exhaustive search effort within the solution space to identify a permutation matrix P which may satisfy the equation to register isometry. The solution space in this case constitutes the set of all permutations of the identity matrix of dimension n which is to be the number of vertices in X_d and Y_d. We can immediately deduce that the size of the solution space is of factorial order of n. As n assumes larger values, $n!$ grows to very large values, making the similarity distance calculator an NP hard problem.

An algorithm involving a linear search among all permutations of the identity matrix to locate P will have $O(n)$ time complexity. In the worst case the $O(n)$ algorithm will have to execute the similarity distance calculator $n!$ times. However, assuming that P may be located with equal likelihood anywhere in the permutation list interval of $[1, n!]$, then on average the $O(n)$ algorithm will have to execute $n!/2$ times. We can come to an immediate realization that an algorithm with $O(n)$ time complexity is not feasible for the equation to register isometry, as n can assume large values. Consider a topology identified by n=10 vertices. Then on average $10!/2 = 1,814,400$ times the similarity distance calculation (i.e. Equation (3-5)) would have to be executed before P is located.

However, consider when $n = 20.$ We come to realization that $20!/2$ is far too large a number. Accordingly, a linear search with $O(n)$ time complexity is only feasible in the most trivial cases where $n \leq 10.$

On the basis of the arguments presented we conclude that we may have to resort to optimization techniques in order to attempt to find a solution for the equation to register isometry (Equation (3-5)) as no algorithm with linear or polynomial computational complexity is feasible.

3.5 Summary

In this chapter, we formulated a computational model by which isometry or correspondence on non-rigid deformable object may be measured and/or registered. The model requires searching the solution space to identify a permutation matrix by which one of the surfaces may be remapped so that the distance between the two surfaces fall under the acceptable threshold level. An exhaustive search of the solution space has $O(n)$ computational complexity. Knowing that the size of the solution space is $n!$, a linear search can only be feasible in the most trivial cases.

In the next chapter, we introduce optimization methods in order to conduct a heuristic search (rather than a brute force exhaustive search) within the solution space whose size is of factorial order. In particular, we discuss Particle Swarm Optimization (PSO) as an evolutionary computing optimization mechanism to navigate the solution space stochastically in order to locate the optimal solution within the acceptable boundary errors. Accordingly, PSO may be utilized to identify the solution for the equation to register isometry (Equation (3-5)).

Chapter 4 Particle Swarm Optimization

In Chapter 3, we developed computational models to register and to establish isometries among deformable objects. We demonstrated that solutions based on brute force search in the corresponding solution spaces would not be feasible and practical. A surface represented by a point cloud of size 100 vertices will have a solution space constituting of 100! elements. Larger topologies will have solution spaces whose sizes grow to extremely large values. Conducting a linear search in this environment amount to unacceptable computational time/complexity.

In this chapter, we examine optimization techniques, with more agreeable computational complexity, in order to more effectively search the solution space. In particular, we discuss Particle Swarm Optimization (PSO) as an evolutionary computing technique to stochastically search the solution space towards the optimal solution. There are many variants of PSO in the literature. Nevertheless, two have been extensively researched in the academia. The first is Classical PSO (or Vectored Based PSO) that operates under the Newtonian dynamics. The second is Quantum PSO that operates under quantum mechanics laws.

Our objective in this chapter is to investigate the merits of Classical PSO and Quantum PSO as the optimization technique for the equation to register isometry (Equation (3-5)) among the deformable objects. As discussed in the previous chapters, the optimization technique must be have the following characteristics:

> **Scalability**: The optimization method should be scalable from small to large sizes of the solution spaces. The size of the solution space is determined by the number of vertices (or landmarks) n that constitutes the topology of the deformable object. As n increments to $n+1$, the size of solution space grows from $n!$ to $n!(n+1).$

Efficiency: The optimization method should be computationally efficient. The method should not rely upon specialized hardware platforms such as parallel CPU farms. The method should be able to provide an optimal solution using a regular CPU machine.

Complexity: The optimization method should be simple and easy to use. In particular, there should not be too many control parameters involved, i.e., the method should shield the user from all internal complexities. Ideally, the method should self-adjust the control parameters as it navigates and learns about the solution space.

Adaptability: Given the heterogeneity of the solution space, the optimization method should adapt its search strategy to the characteristics of the solution space without the need for user's intervention. In particular, the solution space may be infested with distortion and noise. In such a circumstance, the method should be able to self correct its search strategy towards the trajectories that lead to the optimal solution.

We will study Quantum PSO and Classical PSO in detail with respect to SECA guidelines set above, and examine their merits in order to utilize them for the equation to register isometry (Equation (3-5)) among the deformable objects. Finally, we provide a thorough analysis and experimental results of comparing and contrasting Classical PSO against Quantum PSO.

4.1.Optimization

Recall from Chapter 1 that optimization is the process by which a set of instructions outlines the mechanism to navigate to the specified goals, given the accepted tolerance in error [8, 30, 31]. Navigating to the specified goal may be characterized as an exhaustive search within the solution space to locate the near optimal solution within the boundaries of the tolerated error. In navigating to the solutions, certain constraints may be imposed

by the problem itself or by the solution space-thereby reducing the number of acceptable solutions. If an identified solution satisfies all the constraints, then we have a feasible solution. However, there may be many feasible solutions. Among all the feasible solutions, the global optimization attempts to identify the optimal one. Unfortunately, the detection of a optimal one is not always possible. When a global optimal one is an impossibility, then only a local optimal is the possibility. This is usually described as local optimization [8]. An optimization process begins first by a modeling phase. In the modeling phrase a mathematical description of the problem is articulated. The mathematical model must also include all the constraints that are imposed in the problem and the solution space. The product of the modeling phase is a mathematical formalization that is called the *objective function*. The entire effort in the optimization processes is then focused on finding the optimal solution for the objective function formulated in the modeling phase. Interested readers may refer to [32, 30, 33] for comprehensive studies of the subject.

The optimization objective may be formalized by Equations (4-1) and (4-2).

$$\arg\min_{x} f\left(x\right) \left| x \in S \subseteq \mathbb{Z}^n \right. \tag{4-1}$$

$$\arg\min_{x} f\left(x\right) \left| x \in S \subseteq \mathbb{R}^n \right. \tag{4-2}$$

Although Equations (4-1) and (4-2) define a minimization goal, the maximization is a simple transformation of these equations. Ultimately, identifying the solution involves searching within S to look for the candidate(s) that satisfy the goal in (4-1) and (4-2) which is to minimize the value of $f(x)$.

4.2.Particle Swarm Optimization (PSO)

Swarm Intelligence Algorithms [10] are optimization mechanisms wherein the solution space is stochastically searched by a collective and decentralized aggregate of particles. The swarm is constituted of particles that collaborate and navigate through the solution space under two attracting forces, namely the individual particle's perspective and global

swarm's perspective. A particle's individual perspective is the best value that the particle may have achieved. The global perspective identifies the best value that the entire swarm may have attained. These two perspectives (or attraction forces) navigate the swarm towards the ultimate goal. Although decentralized, the swarm exhibits highly choreographed intelligence to steer towards the specified goal under the influences of the two attraction forces. With respect to the optimization objective to minimize $f(x)$ (as formalized by Equations (4-1), (4-2)), the swarm searches the solution space S iteratively. At the end of each iteration, the global and local perspectives may be updated to reflect the smallest value of $f(x)$ achieved individually and/or globally. The search may be continued until the swarm converges to the global minimum in the solution space S, for any given function $f(x)$. As we shortly point out, in order to guarantee the convergence to global minimum certain conditions must be satisfied; otherwise, the particles may diverge, or they may converge to a local minimum (instead of the global one). There are many variants of PSO and an exhaustive coverage of them is provided in [9]. Among those two are of primary interest in this study, namely Classical Particle Swarm Optimization and Quantum Particle Swarm Optimization (quantum PSO). The former operates under classical Newtonian dynamics and has been widely researched and used in computational intelligence [34, 10, 35]. The latter is derived from modern theories in Quantum Mechanics wherein individual particles are modeled after quantum particles whose movement are governed by Schrödinger's equation. We are motivated to using quantum PSO as it is reported [36] to exhibit superior performances under large swarm population sizes. Further, as elaborated in some literature [9], guaranteeing convergence in quantum PSO involves the fine-tuning of only a single control parameter. In contrast, other variants of PSO (for example classical PSO) may require to fine-tune up to five parameters in order to optimize a solution and to guarantee convergence [9]. Finally, we will provide an analysis to show that PSO variants utilizing quantum mechanics may be more stable and accurate, for optimizing complex functions, than classical PSO. Accordingly, for the problems involving large number of dimensions quantum PSO becomes an attractive candidate for optimization.

4.2.1. Classical Particle Swarm Optimization (Classical PSO)

A comprehensive coverage of classical PSO is provided in [9, 10, 34, 35]. A swarm is constituted of M particles searching an N dimensional solution space. Every particle is represented by two N dimensional position and velocity vectors:

$$\begin{aligned} R^i(t) &= \left[R_1^i(t) \ldots R_N^i(t)\right] \\ V^i(t) &= \left[V_1^i(t) \ldots V_N^i(t)\right] \end{aligned} \quad \Big| \quad i \in \left[1 \ldots M\right] \tag{4-3}$$

Here, t represents the time or an iteration of the algorithm (or clock tick in simulation parlance). Further, two memory locations identified by $P_n^{i,L}$ and P_n^G represent the vectors for the local and global perspectives. The individual perspective is updated at the end of each iteration. If $f\left(R^i(l+1)\right) < f\left(P^{i,L}(l)\right)$ then $P^{i,L}(l+1) = R^i$.

The global perspective is also updated: If $f\left(P^{i,L}(l+1)\right) < f\left(P^G(l)\right)$ then $f\left(P^G(l+1)\right) = f\left(P^{i,L}(l+1)\right)$.

The underlying design of classical PSO is the incorporation of the two above-mentioned perspectives into individual particles behaviour, while searching the N dimensional space for the optimal solution. This is accomplished by the Equations (4-4) and (4-5):

$$\begin{aligned} V_n^i(t+\Delta t) = {} & wV_n^i(t) + C_1\varphi_1\left[P_n^{i,L} - R_n^i(t)\right]\Delta t + \\ & C_2\varphi_2\left[P_n^G - R_n^i(t)\right]\Delta t \end{aligned} \tag{4-4}$$

$$R_n^i(t+\Delta t) = R_n^i(t) + V_n^i(t)\Delta t \;\Big|\; n \in \left[1 \ldots N\right] \tag{4-5}$$

Here, Δt is the simulation clock tick (or iteration number), C_1 and C_2 are two positive constants known as cognitive and social parameters, w is the Inertia Weight, φ_1 and φ_2 are two **independent** random variables uniformly distributed in the interval of $[0,1]$. The

convergence of the particles may only be guaranteed by the proper tuning of w, C_1 and C_2. The Inertia Weight controls the influence of the pervious velocities on the current one. A large w facilitates a global search whereas smaller ones ensure local searches. Interested readers are referred to [9], which provides insights on the procedures to choose the optimal values for the constants in Equation (4-4). Assigning proper values to the constants w, C_1 and C_2 subjects the design of classical PSO to inherent complexity. In addition, many other variants of classical PSO introduce more constants. For example, Parsopoulos and Vrahatis [35] have included two additional constants, namely a Constriction Factor, $\chi,$ and a Velocity Maximum, $V_{\max}$. These two constants are used to control the growth and to prevent the explosion of the velocities into unreasonably large values. Such values might supersede the compilers' precisions, and/or cause the swarm to rapidly diverge due to the extremely large velocities. Hence, in the variant of classical PSO used in [35], which we have also considered in this study, the Equation (4-4) is modified to (4-6):

$$R_n^i\left(t+\Delta t\right) = R_n^i\left(t\right) + \chi\, V_n^i\left(t\right)\Delta t \tag{4-6}$$

In addition, Parsopoulos and Vrahatis [35] and Mikki [9] update the new velocity in (4-6) only if $\left|V_n^i\left(t+\Delta t\right)\right| \le V_{\max}$. In short, the optimal operation of classical PSO may involve the complexity of fine tuning of up to five constants. Such complexity might be quite prohibitive for the solutions of very complex problems involving large dimensions (such as the equation to register isometry among deformable objects). Nevertheless, the correct fine tuning of the five constants is essential in avoiding the swarm from diverging. In fact, as our experiments shortly illustrate, in very large dimensions, the swarm operating under the governance of Newtonian dynamics may exhibit instability, and the swarm's explosion may become inevitable. The proper selection of the values for the five constants may be characterized as attempts to ensure the bounded state for the particles individually, and for the swarm collectively. By bounded state it is meant a collective property by which the individual particles and the swarm either have a cyclic trajectory or converges to a focus.

Kennedy [37] has shown that for any variant of PSO to converge, the particles must approach $P_n^i = \frac{\varphi_1 P_n^{i,L} + \varphi_2 P_n^G}{\varphi_1 + \varphi_2}$ where φ_1 and φ_2 are two random number generators with two different seeds. To reiterate, in classical PSO, in order to ensure convergence and to avoid the swarm's explosion, the value of the constants should be selected such that all particles are pulled to P_n^i.

A question then arises whether a variant of PSO exists that may guarantee the bounded state for the particles? If bounded state is guaranteed, it may then be stipulated that the swarm eventually converges either to a suboptimal or to optimal solution, and the swarm never diverges, or never flies away from the region of the solution space wherein global optimum is located. In conclusion, we are motivated to investigate for a variant of PSO in which the bounded state of the particles is guaranteed. This will lead us to Quantum Particle Swarm Optimization which will be discussed in the next section.

4.2.2. Quantum Particle Swarm Optimization (Quantum PSO)

Recall that in classical PSO individual particles' positions and movements are controlled by the Newtonian dynamics. In contrast, in quantum PSO the particles operate under the auspices of quantum mechanics laws. The Newtonian velocity, and the complexities associated with the control of the velocities are no longer relevant in quantum PSO (the particles dynamics are governed by Schrödinger equation [38] in quantum mechanics[1]). There are four compelling reasons that attract us towards quantum PSO in this research. First, previous investigations [39] have shown that quantum PSO exhibits superior performance. Second, in contrast to classical PSO which has up to five control parameters, the number of control parameters in quantum PSO is reduced to one. The third reason is that quantum PSO is inherently more appropriate for exhaustively searching the solution space in very complex problems. The third reason, which will be further explained shortly,

[1] The distinction between the two types of physics exists only to accommodate our sense of "realism". In reality Quantum mechanics laws operate in both micro-world and macro-world.

is due to the stochastic quantum nature enabling the quanta (or the swarm in this context) to conduct far more exhaustive global search, within the boundaries of the solution space, for the optimal solution. The fourth, and the most compelling, reason is that quantum PSO guarantees a bounded state for the particles in a swarm, preventing the particles from flying away from the region of the solution space where the optimal or the suboptimal solutions are located. As discussed above, in classical PSO the particles may diverge and the swarm may explode unless control parameters are properly fine tuned so that the particles approach $P_n^i = \frac{\varphi_1 P_n^{i,L} + \varphi_2 P_n^G}{\varphi_1 + \varphi_2}$. Enforcing this condition for very large dimensions in classical PSO is very difficult (we will illustrate that in very large dimensions classical PSO will exhibit instability that control parameters fail to mitigate[1]).

In classical PSO knowing the location and the velocity of the particle at time t provides adequate insights to deterministically locate the same particle's attributes at time $t+1$, in the N dimensional time-space continuum (Equation (4-6)). Further, in classical PSO, two independent random variables along the gravitational pulls of individual and global perspectives determine the velocity (magnitude and direction) of the swarm according to Newtonian dynamics. In quantum mechanics, attributes such as the velocity do not apply due to the Heisenberg uncertainty principle: when the position is known, the velocity is completely unknown and vice versa. That means, projecting the trajectory of a particle using the direction and magnitude of velocity is not possible. In the quantum time-space framework, we may speak of particle's state which is described by the wave function $\Psi(x, y, z, t)$. The wave function does not have an immediate interpretation but the modulus square of the normalized wave function is associated with the probability distribution of the particle's position. This means in three dimensional space we have the

[1] Newtonian dynamics schemes the evolution of physical system using a linear model. This system is adequate in simulating simple collective behavior such as movement of school of fish or birds. Newtonian system is inherently inadequate to account for complex collective behavior among humans. Individual behavior may be better modeled using uncertainty with respect to his/her future behavior. Therefore, a fifth reason in this case is that quantum PSO is more appropriate for simulating complex social behavior wherein aggregated number of parameters may simply be modeled by the uncertainty.

Equation $|\Psi|^2\, dxdydz = \mathbb{Q}dxdydz$ where the right side of the relation, $\mathbb{Q}dxdydz$, is the probability for the particle to appear in an infinitesimal volume around the point (x, y, z) [36]. In other words, $|\Psi|^2 = \mathbb{Q}$ represents the probability density function satisfying $\int_{-\infty}^{+\infty} |\Psi|^2\, dxdydz = \int_{-\infty}^{+\infty} \mathbb{Q}dxdydz = 1$ which provide a statistical interpretation of the wave function. We may simplify the wave function, $\Psi(x, y, z, t)$, describing the state of in $\mathbb{R}^3$, to $\Psi(r,t)$ which describes the state of particle in a single dimension $\mathbb{R}$. In the simplified model, r may denote the particle's position in any single dimension x, y, or z. In quantum mechanics, the Schrödinger equation (4-7) describes the evolution of wave function with respect to time. In Schrödinger equation, $H(r)$ is Hamiltonian operator (4-8) where $\hbar$ is the Planck constant, m is the mass of the particle, ∇ is the gradient operator $(\hat{x}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z})$ where $(\hat{x}, \hat{y}, \hat{z})$ are the unit vectors, and $U(r)$ is the potential constraining the dynamics of the particle.

$$i\hbar \frac{\partial \Psi\left(r,t\right)}{\partial t} = H\left(r\right)\Psi\left(r,t\right) \tag{4-7}$$

$$H\left(r\right) = -\frac{\hbar^2}{2m}\nabla^2 + U\left(r\right) \tag{4-8}$$

The Schrödinger[1] equation is as fundamental to quantum mechanics as the Newtonian laws are to classical physics. As stated, equation (4-7) is time dependent describing the one dimensional state of the particle with respect to time. In many situations, nonetheless, the forces acting upon the particle are only dependent on the position (spatial position) rather than time. For example, an electron is in a bounded state around a proton in the atom's nucleus. The potential energy U of the electron does not vary with respect to time; yet, the potential energy U does change with respect to the spatial

[1] For simplification, the Schrödinger equation as shown in Equation (4-7) describes the evolution of a physical system in single dimension. The general version of equation in three dimensional space is given by $i\hbar\frac{\partial\Psi}{\partial t} + \frac{\hbar^2}{2m}(\frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} + \frac{\partial^2\Psi}{\partial z^2}) = U(x,y,x,t)\Psi(x,y,z,t)$

position. Under such circumstances, it may be possible to separate the time and the position [39]. Therefore, it may be possible to restate the wave function as $\Psi(r,t) = \Psi(r)f(t)$. In light of this, we may reformulate equation (4-7) to (4-9).

$$\frac{i\hbar}{f(t)}\frac{df}{dt}\Psi(r) + \frac{\hbar^2}{2m}\left(\frac{d^2\Psi(r)}{dr^2} - U(r)\Psi(r)\right) = 0 \tag{4-9}$$

Yet, as shown in [39] the term $\frac{i\hbar}{f(t)}\frac{df}{dt}$ is equivalent to the energy of the system E. Therefore, the equations (4-7) and (4-9) may be reformulated as (4-10)

$$\frac{\hbar^2}{2m}\left(\frac{d^2\Psi(r)}{dr^2}\right) + (E - U(r))\Psi(r) = 0 \tag{4-10}$$

Equation (4-10) is known as time independent Schrödinger equation in one dimension[1].

The solution to Equation (4-10) depends on what assumption one makes in regard to the potential energy U. In the most simple case U or the potential energy may be assumed to be zero everywhere. In this case, known as free particle, the Equation (4-10) may be written as $\frac{\hbar^2}{2m}\left(\frac{d^2\Psi(r)}{dr^2}\right) + E\Psi(r) = \frac{d^2\Psi(r)}{dr^2} + \frac{2mE}{\hbar^2}\Psi(r) = 0$. By setting $k = \frac{2mE}{\hbar^2}$ then, we may have Equation (4-11) describing the state of the free particle in single dimension.

$$\frac{d^2\Psi(r)}{dr^2} + k\Psi(r) = 0 \tag{4-11}$$

We can immediately verify that three potential solutions for (4-11) are given by (4-12).

[1] Extending the argument to three dimensions will also give the time independent version of equation as $\frac{\hbar^2}{2m}(\frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} + \frac{\partial^2\Psi}{\partial z^2}) + (E - U(x,y,z))\Psi(x,y,z) = 0$

$$\Psi(r) = A\begin{cases} sin(kr) \\ \cos(kr) \\ e^{\pm ikr} \end{cases} \tag{4-12}$$

A fourth possible solution is given by their combinations (or quantum superposition[1]) shown in (4-13).

$$\Psi(r) = A\sin(kr) + B\cos(kr) \tag{4-13}$$

In the physical world, nonetheless, the particle is bounded in some space. Consider for example, a metallic crystal. In the metallic crystal the particle (for example, the electron) may be moving anywhere within the crystal. However, at the crystal surface, the particle experiences very large (or let us assume infinite) potential that prevents the particle from crossing the surface. The potential barriers preventing the particle from moving outside may also be known as *potential well*. In the case of the metallic crystal, nonetheless, the potential well is three dimensional box as the particle moves in three dimensional space. Yet, to simplify, let us assume that the particle is moving in one dimension r in the one dimensional potential well. As shown in Figure 4-1, on either side of the potential well, there is a barrier with infinite potential energy preventing the particle to move outside. Inside the potential well, we may assume that the particle has constant potential energy. For the sake of simplicity, let us assume that the particle's potential energy inside the potential well is zero. As shown in Figure 4-1, the particle may move freely along the r axis from $r = 0$ to $r = a.$ In other words, the particle may bounce back freely along the r axis while its potential energy U is constant.

[1] In Hilbert space the superposition is possible by linear combination of the unit vectors of other possible solutions.

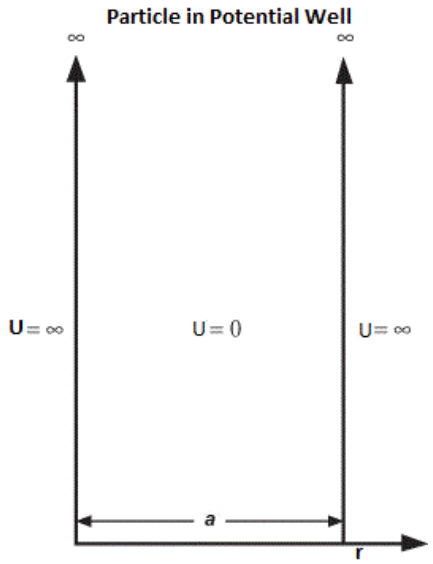


Figure 4-1 Particle in potential well

We can immediately conclude that Equation (4-11) may describe the state of the particle inside the potential well. Therefore, while inside the potential well, (4-12) and (4-13) are solutions for the particle moving inside the potential well.

However, at the two barriers of the potential well, the potential energies are infinitely large[1]. Hence, the particle may never cross at $r = 0$ or $r = a$ to exit the potential well. Therefore, we may state that $\Psi(0) = \Psi(a) = 0$ (and outside the well). From Equation (4-13), it means that $A\sin(0) + B\cos(0) = 0$. Therefore, we may conclude that $B = 0.$ Also, it means that $A\sin(ak) + B\cos(ak) = 0.$ However, since $B = 0$, then the logical conclusion is $A\sin(ak) = 0$ which implies that $ka = n\pi$. To avoid absurdity, n must be an integer and $n \neq 0$ (if $n = 0$, then it means that the particle does not exist anywhere). Recalling from earlier when we set $k = \frac{2mE}{\hbar^2}$, we may restate $ka = n\pi$ as $\frac{n\pi}{a} = \frac{2mE}{\hbar^2}$ or

$$E_n = n^2 \frac{\hbar^2 \pi^2}{2ma^2},\ n = 1, 2, 3, \ldots$$

[1] In reality the potential energy cannot be infinite but very large. Yet, there is a phenomenon known as quantum tunneling which gives a non zero probability for the particle to cross the barrier. We shall not discuss quantum tunneling in this research; however, Schrödinger equation has this possibility accounted for.

As shown above in the last two cases, the solution to the time independent Schrödinger equation in Equation (4-10) is premised upon on how the potential energy, $U(r)$, of the particle is assumed. There are generally three[1] recognized potential wells that are known to guarantee bounded states for the particles in quantum mechanics [9]. In Delta Potential Well [9, 40], $U(r) = -\gamma\delta(r)$ where γ is a positive number proportional to the depth of the potential well, and δ is the Dirac delta function-a function on a real number line that is zero everywhere except at origin, with an integral of one over the entire real line. It is shown[2] that in this potential well, we may get a probability density function of the position as shown in Equation (4-14).

$$\mathbb{Q}(r) = (1/L)e^{-2|r|/L} \tag{4-14}$$

The second potential is known as Harmonic Oscillator Potential [9, 40]. This potential is extensively used in quantum mechanics. The potential energy of the particle is modeled by $U(r) = \frac{1}{2}kr^2$ wherein k is the parameter defining the well depth. It is shown[3] that this potential well has the probability density function of the position as shown in Equation (4-15)

$$\mathbb{Q}(r) = (a/\sqrt{\pi})e^{-a^2r^2} \tag{4-15}$$

It should be noted that in Equations (4-14), (4-15), and (4-16) there are many additional coefficients involved. The derivation of the exact value of the coefficient are involved in many considerations. See Appendix A for the significance of the coefficients and the methods to choose their corresponding values.

[1] A fourth one is also considered in Appendix B. There are other potential wells that are outside the scope of this research. Interested readers may refer to [48, 2] for more coverage of other potential wells.
[2] See Appendix A
[3] See Appendix A

The third potential is known as Square Potential [9, 40]. Its potential energy is modeled by $U(r)=\begin{cases}0, & |r|\le W/2\\ V_1, & else\end{cases}$. This potential represents energy walls that are used to confine all particles with energy levels less than V_1 inside the walls located between the points $r=\pm W/2$ [9]. This potential has the probability distribution function shown in Equation (4-16).

In quantum PSO, we require the measurement of the particles' positions in localized space of measurement. This is the fundamental problem in quantum mechanics. Specifically, the measuring device follows Newtonian laws whereas the particle obeys the laws of quantum mechanics.

$$\mathbb{Q}(r)=\begin{cases}\frac{a}{W}\cos^2\left(\frac{\xi}{W}r\right), & |r| \le W/2\\ \frac{b}{W}e^{-\frac{\eta}{W}r}, & |r| \ge W/2\\ \frac{b}{W}e^{\frac{\eta}{W}r}, & |r| \le -W/2\end{cases} \quad (4\text{-}16)$$

Before we address this dilemma of the interface between the two world-views, we recall Kennedy's analysis [37] that in all versions of PSO (including quantum) the particles converge if they all approach point P in Equation (4-20) (see section 4.2.1). In our single dimension model, let us assume that r denotes the particle position along the X axis, or $r=x.$ Given Kennedy's analysis [9, 37], as x approaches P, then r must approach to 0, or $r=x-P.$

To provide an interface between the quantum and the classical domains, one needs to collapse the wave function of the particle into the localized space of measurement. This is done by the simulation procedure known as the Monte Carlo method [9, 41, 36]. The procedure is done in three steps

Step 1: Generate a random variable u uniformly distributed in the local space.

Step 2: Equate u to the pdf estimate by quantum mechanics (one of the pdfs in (4-14), (4-15), and (4-16)).

Step 3: Solve for the position r in terms of u.

Once the three steps are performed then r may be replaced by $x - P$.

Three update equations [9, 36], by which the dynamics of the particles may be described[1], are considered in this research. These are the delta well potential update, the harmonic oscillator potential update, and the square well potential update.

Delta Well Potential:

$$x_{k+1} = P \pm \frac{\ln\left(\frac{1}{u}\right)}{2g \ln\sqrt{2}} \left| x_k - P \right| \tag{4-17}$$

Harmonic Oscillator Potential:

$$x_{k+1} = P \pm \frac{\sqrt{\ln\left(\frac{1}{u}\right)}}{0.47694\ g} \left| x_k - P \right| \tag{4-18}$$

Square Well Potential:

$$x_{k+1} = P + \frac{0.6574}{g\ \xi} \arccos\left(\pm\sqrt{u}\right) \left| x_k - P \right| \tag{4-19}$$

As illustrated there are no velocity or time parameters involved to locate the position of the particles in either of the potentials. In every case the next position of the particle is a function of its current position, a stochastic average of the best local and best global optimization P, and a uniform random variable u. The particle position, nevertheless, must exhibit the characteristics of swarm. As stated earlier the movement of swarm is highly choreographed under the gravitational forces of individual particle's perspective

[1] See appendix A for more details

and global swarm's perspective. Therefore, as shown in equation (4-20), P is configured so that the two perspectives are accorded for.

$$P = \frac{\varphi_1 P_L^m + \varphi_2 P_G^m}{\varphi_1 + \varphi_2} \tag{4-20}$$

Algorithm Quantum PSO

Initialize $x^m, P_{local}^m, P_{global}$
Do $i = 1, N_{itr}$
  Do $\mathrm{m}=1,\mathrm{N}_{population}$
    Update $\mathrm{P}_{local}^m, P_{global}$
    $\varphi_1 = rand(0,1), \varphi_2 = rand(0,1)$
    $p = \frac{\varphi_1 P_{local}^m + \varphi_2 P_{global}}{\varphi_1 + \varphi_2}$
    $u = rand(0,1)$
    $L = L(g, u, | x^m - p |)$
    if $rand(0,1) > 0.5$
      $x = p + f(L, u)$
    else
      $x = p + f(L, -u)$
    endif
  endDo
endDo

Figure 4-2 Quantum PSO algorithm, obtained from [9]

As shown in Quantum PSO algorithm [9]Figure 4-2, at each iteration the particles "jump" into new quantum position under the influences of probability distribution as well as the gravitational forces of the particles' individual perspectives and global perspectives. Further, the particles' movement are governed by one of the variant of the Potential Well Distribution, L, whose update functions may be any of the equations in (4-17), (4-18), and (4-19). In so, the particles' trajectory within the localized solution space is determined and calculated for.

In our analysis so far, we have used the Schrödinger equation as the source to derive the update functions describing the position of the particles. It is also possible to derive the update functions using only probability theory, and without reference to any argument in physics [9]. In Appendix C, we have shown Mikki's [9] analysis of a generalized PSO algorithm using Markov chains.

4.3. Quantum PSO vs. Classical PSO

Four test functions were used to investigate and to compare quantum PSO against classical PSO. Laskari, and Parsopoulos [34] had already used these functions to measure the performances of classical PSO in optimizing Integer programs up to only 30 dimensions [34]. We were motivated to expand this research to larger dimensions and to compare and to contrast classical PSO with quantum PSO using these test functions.

Test Problem 1) : $F_1 = \|X\|$, $X = [X_1 \dots X_D] \in [-100,100]^D$ and D is the corresponding dimension of the problem. The solution is $F_1(0) = 0$. This test problem was considered for dimensions 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 100, 150, and 200.

Test Problem 2) : $F_2(X) = XX^T$ $X = [X_1 \dots X_D] \in [-100,100]^D$. The solution is $F_2(0) = 0$. This test problem was considered in 5 dimensions.

Test Problem 3) : $F_3(X) = \left(9X_1^2 + 2X_2^2 - 11\right)^2 + \left(3X_1 + 4X_2^2 - 7\right)^2$, $X \in [-100, 100]^2$ with solution $F_3(1,1) = 0$. This problem is bi-dimensional.

Test Problem 4) : $F_4(X) = \left(X_1 + 10X_2\right)^2 + 5\left(X_3 - X_4\right)^2 + \left(X_2 - 2X_3\right)^4 + 10\left(X_1 - X_4\right)^4$ where $X = [X_1 \dots X_4] \in [-100,100]^4$ and the solution is $F_4(0,0,0,0) = 0$.

The simulation programs were written in Cincom Smalltalk 7.8 and Mathematica 8.0. All results generated were stored in an instance of SQL Server 2008. Each simulation involved 100 attempts to optimize the targeted solution using classical PSO and all three variants of

quantum PSO for the corresponding test function. Every attempt to optimize involved up to a maximum of 10000 to 200000 iterations by the swarm to evaluate the corresponding test function. Should the swarm be successful in locating the targeted solution values, then the exact number of iterations leading to the targeted solution value was recorded. The mean number of test function evaluations and the corresponding standard deviations were calculated over all successful convergences out of the overall 100 attempts. For classical PSO we used equations (4-4) and (4-6), following the approach recommended in [34]. This was achieved by setting $C_1 = C_2 = 2.0$, and $X = 0.729$. The value of w (the Inertia Weight), was linearly increased/decreased to locate the optimum values leading to convergence rates of at least 90%, and with the least number of function evaluations. Our experiments indicated that the combination of $X = 0.729$ and $0 < w \leq 0.82$ were effective to prevent the explosions of the velocities for dimensions less than, or equal to, 200. However, as we will illustrate later, in dimensions above 200, in classical PSO, velocities will explode and the incorporation of threshold $V_{\max}$ becomes necessary.

With respect to quantum PSO, all three variants namely quantum PSO-Delta, quantum PSO-Harmonic, and quantum PSO-Square were implemented. The quantum PSO is comparatively less complex, as there is only a single constant $g > 0$ involved in the algorithms. The optimal value of g depends on the function type, the dimension size, and the swarm size. As was the case for Inertia Weight in classical PSO, the optimal value for g was sought by running repeated simulations, observing the convergence rates, and the mean number of function evaluations.

Next, we discuss the simulation results by dividing them into small, medium, large, and very large dimensions.

4.3.1. Small-Sized Dimensions

In this category, we compared the performance of classical PSO against the three variants of quantum PSO. In our experiments we tested F_1 with the number of dimensions of 5,

10, 15, 20, 25, and 30, and equipped with swarm sizes of 20, 20, 50, 50, 100, 100, 100, and 150, respectively. Our objective was to identify the number of F_1 evaluations required in order to achieve at least 90% convergence rates.

Dimension Sizes	Swarm Size	CPSO				QPSO Delta-Well			
		Convgce	# F1	StdDev	IW	Cong	# F1	StdDev	g
5	**20**	100%	39.07	5.84	0.700	97%	32.05	10.49	1.300
10	**20**	99%	72.91	11.07	0.700	66%	179.24	54.36	1.153
15	**50**	99%	81.46	8.29	0.700	92%	233.83	63.90	1.160
20	**50**	98%	106.70	14.31	0.700	65%	531.35	148.20	1.153
20	**100**	100%	88.36	8.43	0.700	91%	110.88	23.52	1.300
25	**100**	99%	112.62	12.33	0.700	92%	439.23	106.48	1.170
30	**100**	91%	139.70	19.58	0.700	90%	1731.77	722.66	1.170
30	**150**	100%	116.42	10.91	0.700	98%	417.77	78.53	1.230
Dimension sizes	**Swarm Size**	**QPSO Harmonic**				**QPSO Square Well**			
		Convgce	# F1	StdDev	g	Cong	# F1	StdDev	g
5	**20**	92%	32.71	15.64	1.83	95%	27.32	20.95	1.06
10	**20**	50%	158.14	83.11	1.78	65%	766.86	438.88	0.92
15	**50**	86%	227.62	117.44	1.77	94%	481.04	255.64	0.94
20	**50**	61%	413.03	166.02	1.77	68%	2468.69	921.50	0.94
20	**100**	96%	113.06	39.10	1.82	91%	100.26	32.58	1.02
25	**100**	89%	538.07	243.73	1.77	93%	375.70	182.54	0.98
30	**100**	87%	1221.49	939.35	1.79	90%	1466.18	987.25	0.97
30	**150**	92%	216.86	89.77	1.84	94%	266.90	106.22	1.00

Table 4.1 Results of testing F_1 in small dimensions

As Figure 4-3 and Table 4.1 illustrate, classical PSO significantly outperformed all three variants of quantum PSO in small-sized dimensions. As shown in Table 4.1, in all categories, the mean number of F_1 evaluations never exceeded 140 for classical PSO to achieve 90% convergence rate. The worst performance was exhibited by quantum Square

for the swarm size of 50 and dimension size of 20, which required nearly 2500 F_1 evaluations to achieve its best performance.

Figure 4-3 also illustrates that all three variants of quantum PSO are extremely sensitive to the ratio of swarm size to dimensions. As the ratio increased (swarm 20 and dimension 5, swarm 100 and dimension 20, swarm 150 and dimension 30) the performances of all four became comparable. Nevertheless, as the ratio decreased (e.g. swarm 50 and dimension 20) then the performances of all variants of quantum PSO worsened. Further, as the data in Table 4.1 show, in all dimension sizes, we were able to achieve a 90% convergence rates for classical PSO.

However, achieving 90% convergence rates were not possible for all dimensions sizes in the variants of quantum PSO. A careful examination illustrates that, in small-sized dimensions, variants of quantum PSO gave comparable performance to classical PSO when the ratio of swarm size to dimension size was greater than 4. Our experiments concluded, however, that increasing the ratio above 6 would only improve the performances of quantum PSO variants. However, they could hardly outperform classical PSO except for very small dimension (less than 5).

We have included the results for F_2, F_3, and F_4 in Appendix D, due to space limitations. Nevertheless, in all of the above-mentioned test functions the dimension sizes were smaller than 5. Our results confirmed similar conclusions indicating classical PSO outperforming quantum PSO in small sized dimensions in all four test functions F_1, F_2, F_3, and F_4.

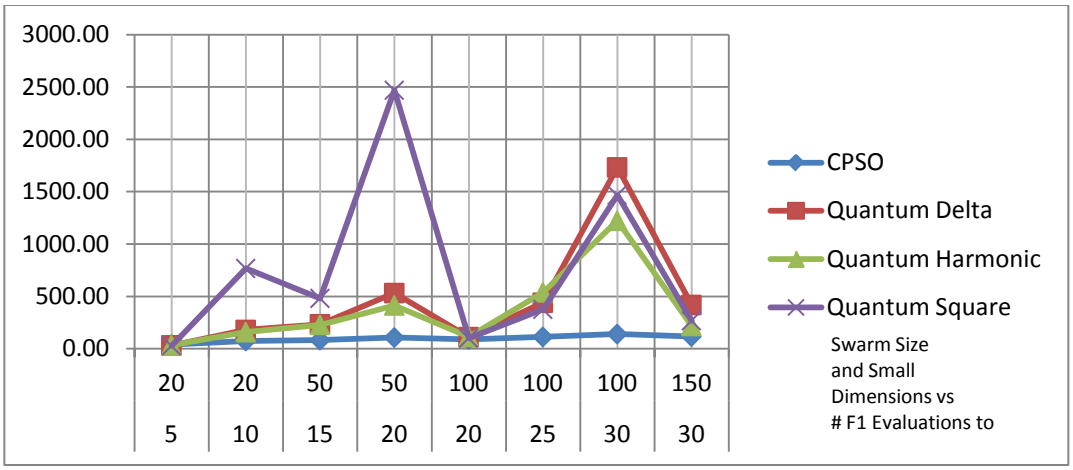


Figure 4-3 Results testing F_1 in small dimensions

4.3.2. Medium-Sized Dimensions

In this category we pursued the same objectives as the preceding one. However, here we employed dimensions of size 35, 40, 45, and 50 against the swarm sizes of 150 and 200. As illustrated in Figure 4-4 and shown in Table 4.2, the conclusions reached for the medium-sized dimensions were nearly identical with those of the small sized dimensions. In general, we witnessed that classical PSO outperformed all three variants of quantum PSO. Further, similar to the small-sized dimensions the quantum PSO performance significantly deteriorated as the ratio of swarm size to dimension neared 3, and improved as the ratio increased to 5.

The data in Table 4.2 illustrates that, when the ratio approached 3, it became difficult for quantum PSO Harmonic, and impossible for quantum PSO Square, to achieve a 90% convergence rate. Further, quantum PSO Square could only converge at 56% and 80% when the ratios of swarm size to dimensions were 3 and 4, respectively. Finally, it is noteworthy that quantum PSO Delta required over 10000 F_1 evaluations to achieve 90% convergence rate for ratio 3. In contrast, classical PSO required less than 800 F_1 evaluations.

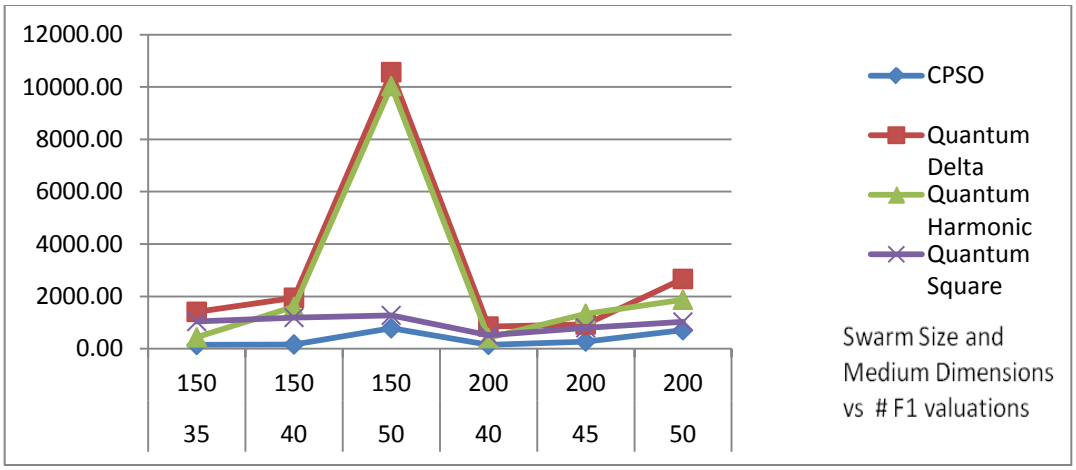


Figure 4-4 Results testing F_1 in medium dimensions

Dimension Sizes	Swarm Size	CPSO				QPSO Delta-Well			
		Congnce	# F1	StdDev	IW	Congnce	#F1	StdDev	g
35	**150**	91%	144.98	19.61	0.700	95%	1407.91	500.40	1.180
40	**150**	90%	163.89	19.02	0.700	90%	1945.41	610.75	1.180
50	**150**	90%	781.33	85.34	0.800	90%	10564.70	5361.13	1.170
40	**200**	95%	148.67	18.31	0.700	94%	844.29	185.04	1.210
45	**200**	95%	276.25	33.47	0.750	90%	927.73	239.57	1.230
50	**200**	96%	707.09	79.07	0.800	90%	2664.34	1008.6	1.200
Dimension sizes	**Swarm Size**	**QPSO Harmonic**				**QPSO Square Well**			
		Congnce	# F1	StdDev	g	Congnce	# F1	StdDev	g
35	**150**	91%	436.48	274.35	1.81	91%	1036.21	537.73	0.98
40	**150**	94%	1598.96	1015.83	1.79	91%	1190.71	712.58	0.99
50	**150**	88%	10035.00	5873.71	1.79	56%	1273.88	432.24	1.00
40	**200**	95%	413.03	119.93	1.82	97%	513.37	212.57	1.00
45	**200**	94%	1338.71	594.48	1.81	93%	793.79	445.12	1.00
50	**200**	94%	1872.99	1111.8	1.81	80%	1023.92	471.8	1.00

Table 4.2 Results testing F_1 in medium dimensions

The results obtained so far with respect to small sized dimensions and medium sized dimensions demonstrate that for the equation to calculate similarity among deformable objects (Equation (3-5)) classical PSO might be the more promising option. Yet, we have to remind the reader that the scalability of the optimization technique to very large dimensions and very large solution spaces are of paramount significance, which we will address in the next sections. Therefore, we defer a comprehensive analysis to after presenting the results for large and very large dimensions.

4.3.3. Large-Sized Dimensions

Experimental results obtained for large sized dimensions overwhelmingly support the performance superiority of Quantum PSO against Classical PSO. Further, the results also point out to destabilization of Classical PSO as the dimensions enlarge. As Figure 4-5 illustrates and Table 4.3 shows, all variants of quantum PSO significantly outperformed classical PSO by at least 40% improvement.

In the worst case for classical PSO (swarm 4000, dimension 200), quantum PSO Harmonic outperformed classical PSO by a large margin, namely 76%. Amongst the variants of quantum PSO, the Harmonic outperformed Delta Well and Square Well, although Square Well gave comparable performance to Harmonic in all dimensions.

A careful examination of Figure 4-5 depicts that all four curves have made concave up formations. Amongst the four curves, however, classical PSO has a steeper angle of descent as well as a sharper angle of ascent, compared with all three variants of quantum PSO.

This is indicative of classical PSO being very sensitive to the ratio of swarm size to dimension, in domains with large dimensions. On the other hand, the quantum PSO Harmonic and quantum PSO Square have much flatter curves. This behavior is indicative of their stability and less sensitivity to moderate changes in the ratio of swarm to dimension in this category.

Dimension sizes	Swarm size	CPSO				QPSO Delta-Well			
		Congnce	# F1	StdDev	IW	Congnce	# F1	StdDev	g
100	**600**	98%	2930.44	317.11	0.820	90%	2124.64	567.19	1.300
100	**700**	95%	1444.61	135.91	0.800	99%	1962.35	574.94	1.300
100	**800**	94%	1375.9	131.41	0.800	94%	1356.03	329.65	1.300
100	**1000**	99%	1256.02	110.88	0.800	94%	973.28	165.24	1.320
100	**2000**	97%	399.12	34.49	0.750	93%	438.59	46.73	1.360
100	**3000**	92%	229.16	21.15	0.700	96%	321.39	35.17	1.380
150	**3000**	100%	1720.53	129.18	0.800	97%	804.36	122.01	1.400
150	**3500**	92%	891.17	62.56	0.775	99%	738.95	102.58	1.400
150	**4000**	97%	870.59	65.53	0.775	94%	588.56	62.08	1.420
200	**4000**	96%	2330.6	202.5	0.800	98%	1337.68	345.66	1.420
Dimension sizes	**Swarm size**	**QPSO Harmonic**				**QPSO Square Well**			
		Congnce	# F1	StdDev	g	Congnce	# F1	StdDev	g
100	**600**	94%	859.65	286.98	1.88	84%	1068.54	391.33	1.04
100	**700**	93%	703.30	207.12	1.88	94%	959.73	360.04	1.04
100	**800**	95%	543.40	175.97	1.89	98%	764.48	215.97	1.04
100	**1000**	90%	388.27	91.25	1.90	96%	471.34	117.35	1.05
100	**2000**	99%	200.90	39.87	1.92	100%	202.99	24.52	1.07
100	**3000**	94%	141.02	14.87	1.94	95%	141.81	19.22	1.09
150	**3000**	98%	356.51	66.67	1.93	100%	424.70	40.28	1.06
150	**3500**	91%	307.63	47.59	1.94	98%	340.04	50.36	1.07
150	**4000**	98%	287.43	41.70	1.94	99%	309.21	34.32	1.07
200	**4000**	98%	546.88	142.67	1.94	99%	616.19	144.28	1.08

Table 4.3 Results testing F_1 in large dimensions

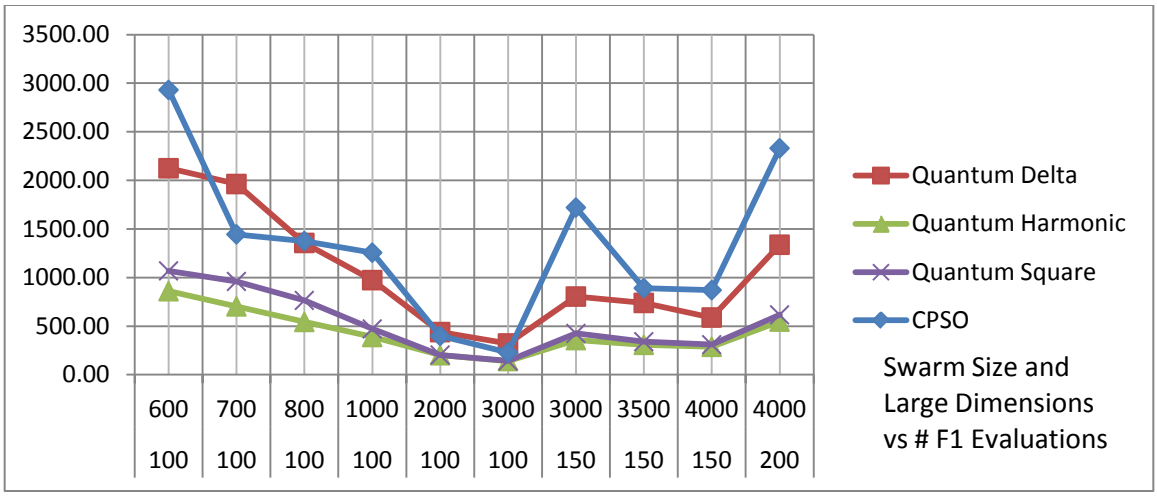


Figure 4-5 Results testing F_1 in large dimensions

4.3.4. Very Large Sized Dimensions

In this section we use analytical approaches to conclude the trend we have witnessed so far. That is, we are interested in discussion whether quantum PSO will also outperform classical PSO when the number of dimensions are larger than 200.

We discussed, in the preceding sections, that the design of classical PSO may involve fine tuning of up to five constants. These are the Inertia Weight, C_1, C_2, χ and $V_{\max}$. In the preceding sections, we set constant values to C_1, C_2, and χ. We did not require a threshold set for $V_{\max}$, as velocities for classical PSO never exploded into astronomical values. In fact, the only variable utilized in the preceding classical PSO experiments was Inertia Weight. Likewise, the only variable in quantum PSO counterparts was g. Accordingly, from parameter complexity perspective, both algorithms enjoyed the same level of complexity given that each had to use at most one control parameter. However, if the circumstances warrant to treat C_1, C_2, or χ as variables, then we can conclude that classical PSO will have worse variable complexity than quantum PSO. In fact, the data obtained in the preceding section indicates that, for domains with more than 200 dimensions, classical PSO would potentially suffer from increased parameter complexity,

exhibit instability, and endure deteriorated performance compared with its quantum PSO counterparts.

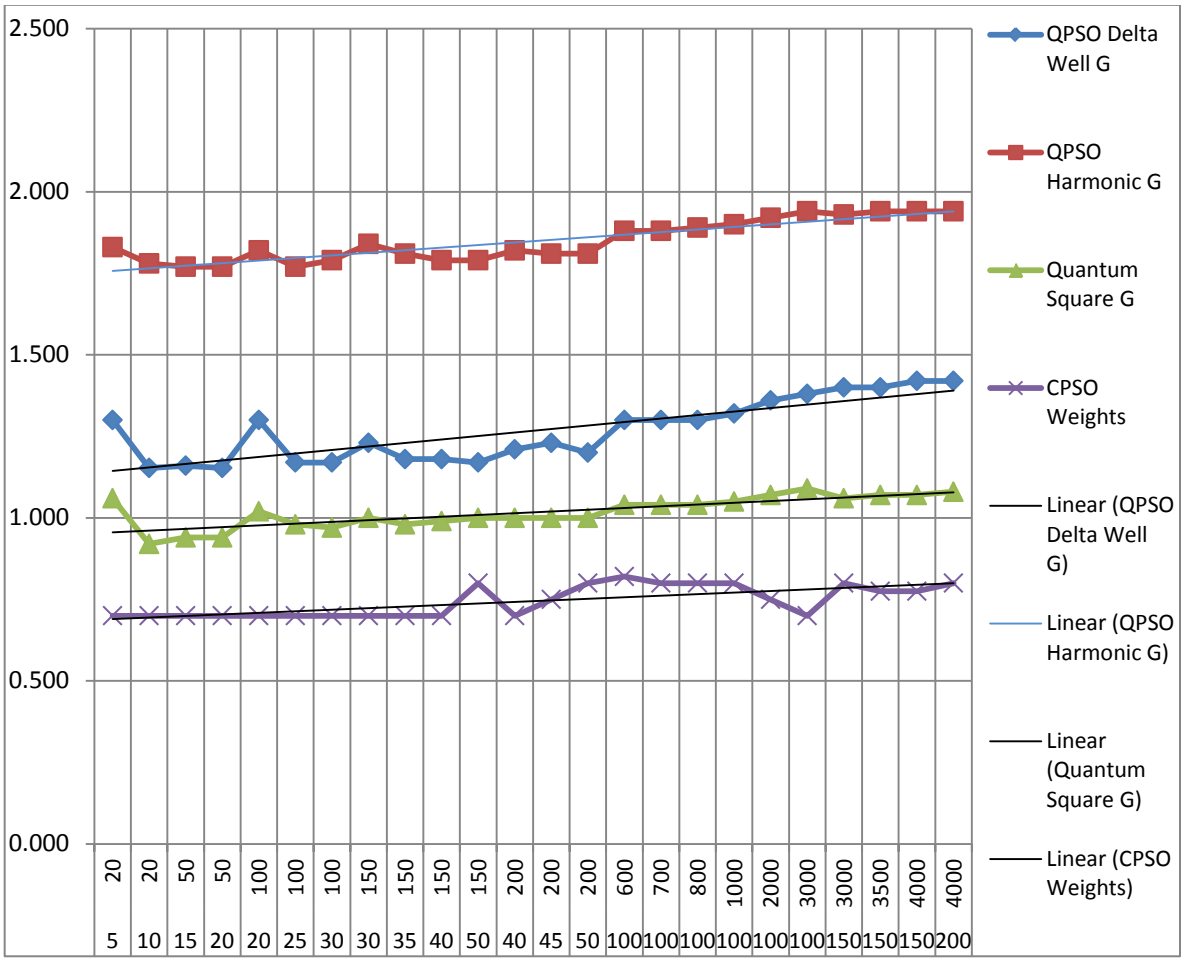


Figure 4-6 Inertia weight and g

Figure 4-6 depicts an upward trend in the values of Inertia Weight and g as dimensions enlarged. A careful examination shows that all four curves have positive slopes. A question might arise is "how large could these values go"? May the values of Inertia Weight and g increase indefinitely? Theoretically, there is no limit on how large value g may assume [9], as g is a control constant. However, the same may not be assumed for Inertia Weight. Recalling equation (4-6), the significance of Inertia Weight is to control the effect of the historical velocity upon the current one. Accordingly, the Inertia Weight may not assume a value larger than 1.0. This implies that the full percentage of the historical velocity should be applied when calculating the current velocity, in equation (4-4). Thus, a

theoretical limit on the maximum value of Inertia Weight is established. A careful study of Inertia Weight in Figure 4-7 illustrates that its value gradually increases as the dimensions enlarge. We may conclude that, in dimensions above 200, the value of Inertia Weight may have to be enlarged further. Nonetheless, our simulations results indicated that for F_1, the values of Inertia Weight larger than 0.82 caused the corresponding velocities to explode into extremely large values.

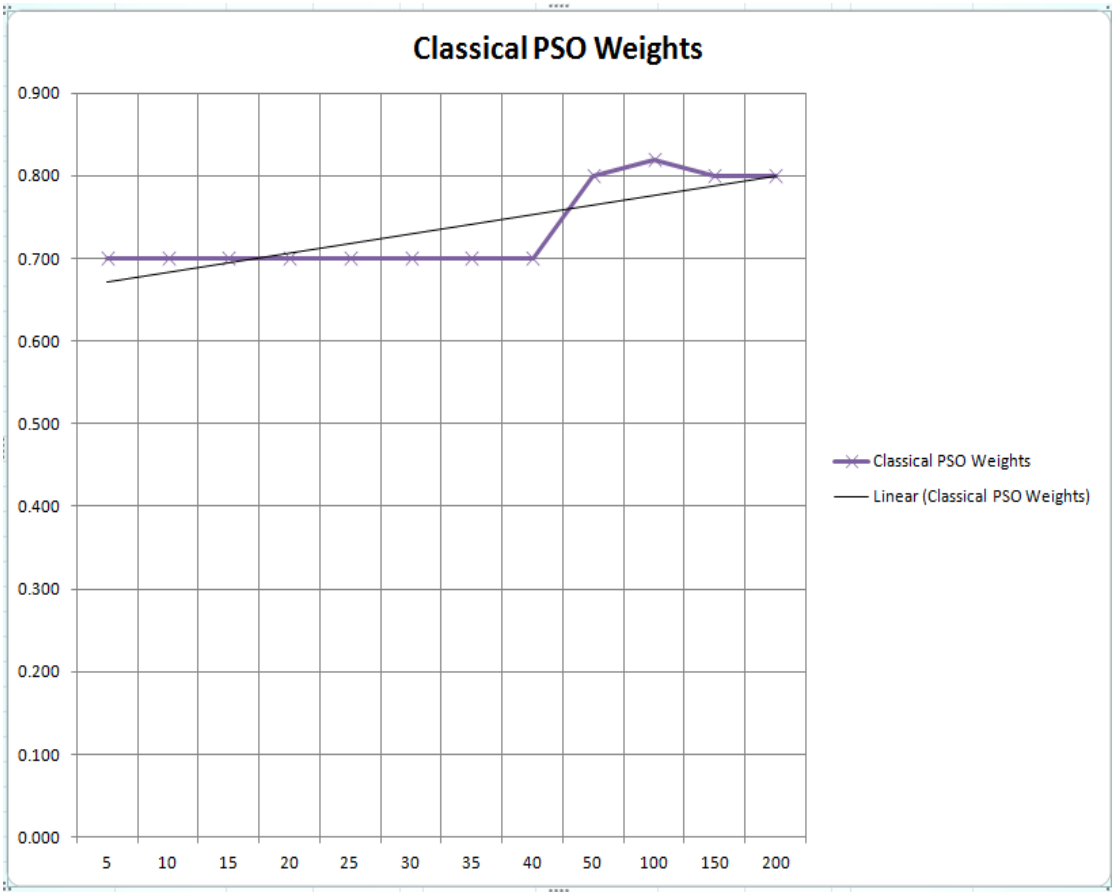


Figure 4-7 Inertia weight versus dimensions in classical PSO

Thus, for dimensions above 200 we may no longer be able to proceed without imposing a threshold $V_{\max}$ to clamp down on exploding velocities. Introducing $V_{\max}$ brings not only a second variable in the design of classical PSO (and thus worsening the variable complexity of classical PSO), it also neutralizes the effects of enlarging Inertia Weight. In conclusion, as the number of dimensions exceeds 200, classical PSO will experience instability given

the counter-effects of enlarging Inertia Weight against $V_{\max}$. To cope with the inherent instability, one may have to revisit the policies that set constant values to C_1, C_2, and χ in order to treat the latter three as variables instead as constants, and accordingly to further deteriorate the variable complexity of classical PSO.

4.3.5. Varying Inertia Weights and g to Improve Performance

In classical PSO, varying Inertia Weights would have significant effects on improving the mean number of evaluations required while maintaining above 90% convergence rates.

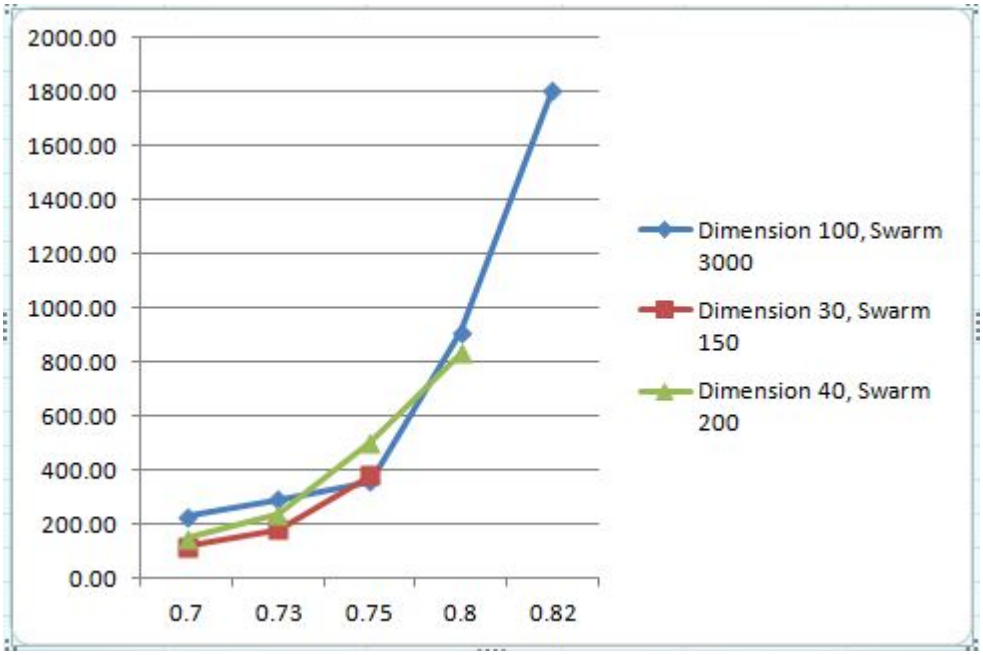


Figure 4-8 Varying weight against # F_1 in classical PSO

Figure 4-8 illustrates three cases wherein the varying Inertia Weights made significant improvements on mean number of evaluations required. For example, in dimension 100 and Swarm size 3000, the mean number of F1 evaluations was reduced from 1800 at Inertia Weight 0.82 to just over 200 at Inertia Weight 0.7. As Figure 4-9 illustrates, the similar improvements were also experienced in quantum PSO by varying the value of g.

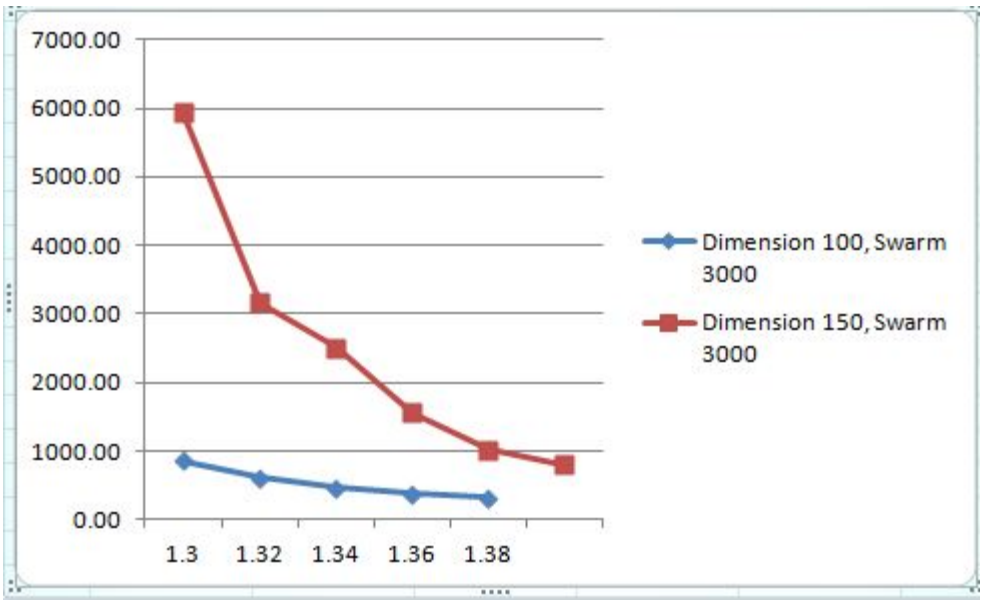


Figure 4-9 Varying g against # F_1 in quantum PSO Delta

Figure 4-9 illustrates two examples in quantum PSO Delta in which the number of F_1 evaluations were considerably decreased by increasing the value of g from 1.3 up to 1.4 while maintaining 90% convergence rates. We conclude that fine-tuning the control parameters in both classical PSO and quantum PSO is effective in optimizing better solutions. In many cases, varying the control parameters enabled us to significantly reduce the number of function evaluations while maintaining 100% or plus 90% convergence rates. Quantum PSO has a single control parameter and there is no restriction on how it may be used in optimization of function with very large dimensions. The same, however, may not be concluded for classical PSO. As stated previously, there is a theoretical limit on how large value Inertia Weight may assume. To prevent the explosion of velocities $V_{\max}$ and/or constriction factor may have to be utilized. The use of the two as variables in the design of classical PSO not only worsens the variable complexity, but also introduces instability by neutralizing the impacts of Inertia Weight. Accordingly, increasing the size of the swarm appears to be the only method for optimizations of functions with very large dimensions. Increasing the size of the swarm implies more function evaluations by the members of the swarm and thus degradation of

the overall performance. Therefore, we may conclude that for very large dimensions, quantum PSO is the preferred option.

4.4. PSO and the Equation to Register Isometry

In the preceding chapters we formalized mathematical definitions to register isometric regions on deformable three dimensional objects. We developed the equation to register isometry (Equation (3-5)) by which the isometries on three dimensional objects may be determined and established. We showed, however, that the solution for the equation required a brute-force search in its corresponding solution space in order to determine if isometric regions exist. Given the size of the solution space, no search method with polynomial time complexity can effectively and exhaustively search the entirety of the solution space in reasonable computational time. Finally, we concluded that we may have to opt for optimization techniques in order to stochastically search the solution space to locate an optimal solution within the acceptable error boundaries.

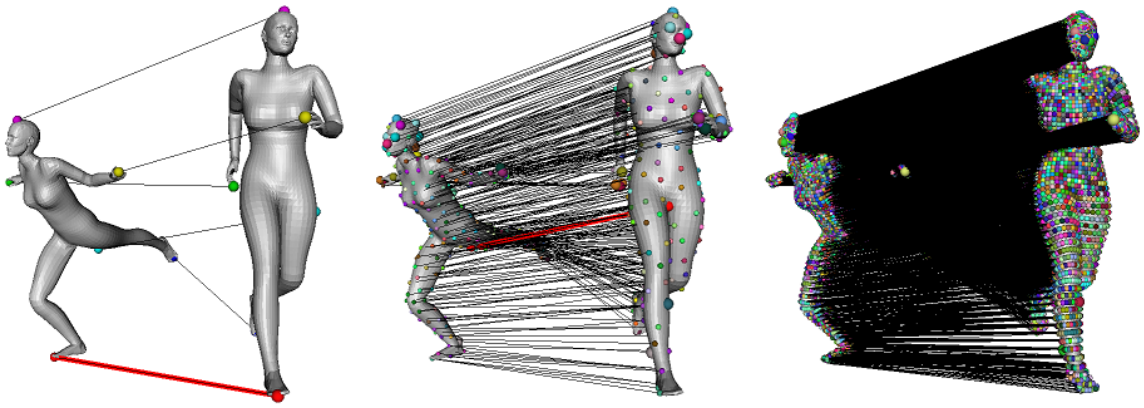


Figure 4-10 n vertices and $n!$ solution space [42]

Recall from the previous chapters that the size of the solution space for the distance calculator equation (Equation (3-5)) is directly related to the number of vertices n of the point cloud representing the three dimensional objects. As n increases to $n+1$ then the size of the solution space goes from $n!$ to $(n+1)!$ (Figure 4-10). Accordingly, the optimization algorithm for equation to register isometry (Equation (3-5)) must exhibit

scalability for small, medium, large, and very large values of n. Our analysis in this chapter indicated that although the classical PSO provided more favorable results for small and medium sized dimensions, the PSO operating under the governance of Newtonian dynamics was not scalable for large and very large sized dimensions. On one hand, quantum PSO provided less favorable results in small and medium sized dimensions; yet, the PSO operating under the governance of quantum mechanics laws outperformed its rival in large and very large dimensions, and was scalable for all dimension sizes.

Further, as discussed above, quantum PSO ensured that the optimization effort remained focused in the bounded region wherein the optimal and/or suboptimal solutions were located, irrespective of the value of n and/or the size of the solution space. In classical PSO, guaranteeing bounded state involves supplementary optimization involving proper values for the five additional control parameters in order to guaranteeing convergence. The complexity associated with this effort renders classical PSO unsuitable for a complex problem such as the equation to calculate similarity distance (Equation (3-5)) among the deformable objects.

Finally, both classical PSO and quantum PSO are highly adaptable to the characteristics of the solution space. Yet, classical PSO conducts its search using a Newtonian model. This means the particle's navigation in any interval of time is constrained by the trajectory of the particle governed by the particle's position and its velocity. In quantum PSO, the particle's state is governed by the Schrödinger equation, which signifies a probability distribution for the particle's next position. This property appears to give the swarm the ability to sample and to survey the heterogeneous solution space more effectively.

Accordingly, our analysis supports the conclusion that quantum PSO is more appropriate optimization technique for complex problems such as the equation to calculate the distance between three-dimensional deformable objects (Equation (3-5)).

4.5.Conclusion

In this chapter, we investigated an optimization technique known as Particle Swarm Optimization. We provided detailed dynamics of two PSO variants namely classical PSO and Quantum PSO. The former operates under Newtonian laws whereas the latter behaves according to quantum mechanical principles. Our analysis indicated quantum PSO was scalable for the problem statements of all dimension sizes. Specifically, for problem statements involving large and very large dimensions, quantum PSO provided superior results both in convergence rates and computational efficiency. The results of our experiments as well as our analysis indicated that classical PSO exhibited instability in the form of the swarm disintegration, and the swarm diverged in the n dimensional spaces when the value of n went above certain threshold levels. Finally, quantum PSO design enabled a bounded state for the optimization effort to search only the solution space wherein a solution (optimal or suboptimal) is located. Accordingly, we concluded that for the problem statements such as the equation to register isometry (Equation (3-5)) wherein the number of dimensions could grow very large, quantum PSO was the more suitable candidate to stochastically search the solution space.

In Chapter 5, we apply quantum PSO to the problem statement to calculate the isometry distance between two deformable objects (Equation (3-5)).

Chapter 5 Design of Quantum PSO for Isometry Registration

In this chapter we describe the design and the implementation of a quantum PSO algorithm for the solution of the equation to register isometry (Equation (3-5)) between two non-rigid three-dimensional objects. The design and the implementation should address all the constraints that are imposed on the solution space, i.e., any potential solution that infringes upon the constraints stipulated, is consequently invalidated. The formulation of the problem statement incorporated with the constraint statements constitutes the *objective function* for the isometry detector (Equation (3-5)). We will explain the reformulation and the transformation of the computational model (described by Equation (3-5)) to an objective function that quantum PSO may operate upon in order to identify a solution. Other important elements of the design such as initialization methods, random number generators, topology representations, and so on will be discussed. The design must also uphold the Scalability, Efficiency, Complexity, and Adaptability (SECA) guidelines described in Chapter 4. We will present the experiments conducted and the results obtained. Finally, we conclude this chapter with an analysis of the results, challenges encountered, and the solutions devised to overcome the problems confronted.

5.1.Quantum PSO Design

In this section, we outline the design and programming specific considerations by which quantum PSO conducts the search for the optimal solutions for the equation to register isometry.

5.1.1. Quantum PSO and Stochastic Processes

Recall from Chapter 4 that quantum PSO is an optimization process that operates under quantum mechanical laws. The state of the swarm particles in the quantum space is

determined by a probability distribution. In section 4.2.2 we provided the general algorithm (Figure 4-2) for quantum PSO. As shown there, the algorithm relies extensively on the sequence of random numbers generated. The health and the effectiveness of the algorithm in simulating the behavior of the quantum domain relies principally on the health and the goodness of the random numbers generated. Among the qualities that determine the goodness of a random number generator are the uniform and long periodicity.

For the experiments conducted in this thesis we made use of primarily two random number generation algorithms: Extended Cellular Automation (ExtendedCA) and Mersenne Twister Shift Generator.

The ExtendedCA [43] is based on the theory of cellular automata and uses a genetic algorithm to generate high quality random bit sequences. It relies on five cell rules wherein each cell is dependent on the non-adjacent cell from the preceding steps. The Mersenne Twister Shift Generator [44] is based on matrix linear recurrence over a finite binary field. The name is driven from its underlying algorithm which utilizes Mersenne prime numbers. Both algorithms have periodicity above $2^{19937}-1$.

Given the usage of some parallelization in the experiments conducted, the programming aspects were accordingly designed to ensure that the sequence of pseudorandom number generations remained independent across the parallel kernels. The design of the experiments utilized a master seed to guarantee reproducibility of the simulation events. In order to do so, the user launched the simulation in the master kernel by supplying a large integer as the master seed. The master seed was then used to generate unique local seeds per slave kernel prior to launching a slave kernel. Each slave seed was then supplied as input parameter to the slave kernel. Finally, the slave seed in the slave kernel was utilized to generate additional seeds used by all random number generators inside each and every slave kernel. This design ensured not only reproducibility of the simulation

events but also eliminated the possibility of any behavioral codependency across parallel kernels.

5.1.2. Quantum PSO and Particle Initialization

The swarm in quantum PSO constitutes particles that collectively search the solution space. There were a number design decisions involved that could have wide range of implications with respect to the behavior of the swarm. The swarm size and the swarm initialization were two influential factors whose variations were explored in the experiments conducted in this thesis.

The choice of the swarm size will be explored in more detail as the corresponding experiments are elaborated; nonetheless, we may state here that the size of the swarm in all experiments conducted remained finite per experiment; that is, there were no possibility for the swarm size to increase or decrease during a simulation experiment.

In the experiments conducted, we employed different mechanisms by which the swarms were initialized prior to the commencement of their navigations in the solution space to locate the optimal solution (or the targeted permutation matrix, P, as discussed in Chapter 3). The optimization of the equation to register isometry (Equation (3-5)) may be paraphrased as the heuristic search among permutations of the identity matrix, to locate P-the targeted permutation matrix. Therefore, the individual members of the swarms had to be initialized either to random permutation matrices, or the individual members had to be initialized to an instance of a stochastic matrix.

5.1.2.1. Random Permutation Matrix Initialization

Recall the discussions from the previous chapters on topology representations by a point cloud of size n. In such topologies, there are $n!$ different ways to express the permutations of the identity matrix, i.e., the size of the solution space is $n!$. Three different approaches were employed to initialize the swarm with the permutation matrices from the solution space.

In the *Non-even Distribution Approach* the individual particles were assigned to a permutation matrix randomly selected from the solution space. Given the randomness involved, there were no guarantees that the particles were evenly distributed across the solution space prior to the commencement of the swarm navigation. In this form of initialization, certain segments of the solution space might be more heavily represented as particles density in those segments was higher. On the contrary, other segments of the solution space might be underrepresented due to the lower density of the particles from the corresponding regions.

The second method, in this category, was the *Even Distribution Approach* wherein the algorithm ensured that all segments of the solution space were evenly represented by the even density of the swarm members. For example, a solution space of size $n!$ was divided into r regions. Subsequently a swarm with m particle members was divided into m/r parts, assigning each part to one of the $n!/r$ regions. The division of the solution space into equal regions was possible by lexicographically ranking (please see section 5.4.1) of all permutations of the solution space in ascending or descending order.

The third method was the Monte Carlo Distribution Approach (Figure 5-1). In this method a probability distribution function was established and only those particles which scored with higher than acceptable threshold level would be admitted into the swarm membership. In initializing the members of swarms, the Monte Carlo Distribution

Approach ensures that all members of the swarms have an acceptable probability of success in converging to the optimal solution.

5.1.2.2. Random Stochastic Matrix Initialization

In this method each individual members of the swarm was initialized with a stochastic matrix whose entry values ranged randomly in the interval of [-1, 1].

Monte Carlo Distribution Approach

$$\Pr(P) = \frac{1}{\sqrt{2}} \exp\left(-\frac{1}{2} (\| PXP^T - Y \|_F + \| P^T YP - X \|_F) \right)$$

1 – Start with a random permutation matrix P_{old} that scores a probability value higher than certain threshold, t

$\Pr(P_{old}) > t$. Make P_{old} the first member of swarm

2-Generate a new random permutation matrix P_{new}

3- Accept P_{new} into the swarm if

$$A(\Pr(P_{old}), \Pr(P_{new})) = \min\left(1, \frac{\Pr(P_{new})}{\Pr(P_{old})}\right) > u$$

where u is a uniform probability distribution

4- If P_{new} were accepted then $P_{old} = P_{new}$.

5- Repeat the 1-4 until all members of the swarm are initialized.

Figure 5-1 Monte Carlo distribution initialization approach

5.1.3. Quantum PSO and the Gravitational Pull

In Chapter 4 we illustrated that in Quantum Swarm Intelligence the swarm navigates within the boundaries of a quantum domain under the influences of two gravitational forces. The first gravitational force is the best local position, P_L^m, that each individual member of the swarm has acquired since the commencement of the swarm navigation. The second gravitational force is the best global position, P_G^m, that the entire swarm has acquired since the swarm exploration of the solution space commenced. There is a third

factor of randomness under a uniform probability distribution that complements the two gravitational forces in order to conduct the evolution of the swarm in the quantum space. The combination of all three factors were captured by the Equation (4-20) which plays a central role in the overall quantum PSO algorithm (in Figure 4-2) to map the state of the swarm in the quantum space framework.

$$P = \frac{\varphi_1 P_L^m + \varphi_2 P_G^m}{\varphi_1 + \varphi_2}$$

A careful examination of Equation (4-20) reveals that the two random variates, φ_1 and φ_2, may only introduce the element of randomness into P if and only if $P_G^m, P_L^m \in \mathbb{R}$. When $P_G^m, P_L^m \notin \mathbb{R}$, then φ_1 and φ_2 cannot on their own introduce randomness to the structures that P_G^m, P_L^m represent. Consider, for example, when P_G^m, P_L^m are permutation matrices with dimension of 3 such that $P_G^m = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, P_L^m = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$. Placing P_G^m, P_L^m in (4-20) will result in an aberration from the intended randomness $\left(\frac{\varphi_1 \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} + \varphi_2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}{\varphi_1 + \varphi_2} = \frac{\begin{pmatrix} 0 & \varphi_1 & 0 \\ 0 & 0 & \varphi_1 \\ \varphi_1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} \varphi_2 & 0 & 0 \\ 0 & 0 & \varphi_2 \\ 0 & \varphi_2 & 0 \end{pmatrix}}{\varphi_1 + \varphi_2} = \frac{\begin{pmatrix} \varphi_2 & \varphi_1 & 0 \\ 0 & 0 & \varphi_1 + \varphi_2 \\ \varphi_1 & \varphi_2 & 0 \end{pmatrix}}{\varphi_1 + \varphi_2} \right)$. A careful examination of the result, $RE = \frac{\begin{pmatrix} \varphi_2 & \varphi_1 & 0 \\ 0 & 0 & \varphi_1 + \varphi_2 \\ \varphi_1 & \varphi_2 & 0 \end{pmatrix}}{\varphi_1 + \varphi_2}$, illustrates that irrespective of the values for φ_1 and φ_2, RE can only be randomized through the indices that have

non zero values in $P_G^m = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ or $P_L^m = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$. In other words, the two random variables φ_1 and φ_2 as well as non-zero values in P_G^m and P_L^m limits the swarm's navigation to the confines of the subset of the solution space that have structural affinity with those permutation particles of P_G^m or P_L^m. To further elucidate, consider the region of the solution space that is represented by the identity matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ which has "1" in indices p_{11}, p_{22}, and p_{33}. The identity matrix lacks structural affinity with both $P_G^m = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ or $P_L^m = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$ as neither of the latter two have a '1' in indices p_{33} and p_{22}. Therefore, it is be impossible for Equation (4-20) to produce a result, $RE = \dfrac{\begin{pmatrix} \varphi_2 & \varphi_1 & 0 \\ 0 & 0 & \varphi_1 + \varphi_2 \\ \varphi_1 & \varphi_2 & 0 \end{pmatrix}}{\varphi_1 + \varphi_2}$, which has non zero values in indices p_{22} and p_{33}. Accordingly, Equation (4-20) can never create attraction towards the region of solution space represented by the identity matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, as neither P_G^m nor P_L^m has a 1 in indices p_{22} or p_{33}. The combination of $\varphi_1, \varphi_2, P_G^m$, and P_L^m in Equation (4-20) does not necessarily create structural vigor to exert gravitational pull to all the regions of the solution space. The randomness by φ_1 and φ_2 may only randomize the trajectory of swarm navigation to the confines of the space that P_G^m, P_L^m have structural affinity with.

To rectify the dilemma above, we need to augment the random behavior in (4-20), φ_1 and φ_2, with an additional mechanism to ensure that not only the elements in RE are

randomized (as done by φ_1 and φ_2) but also the structure of RE is scrambled by randomly exchanging rows and/or columns. In other words, the randomization mechanism should ensure that randomness, over large number of experiments, brings about affinity to the entirety of the solution space in the resultant RE, and all structural variations in the solution space are accordingly accounted for. This objective may only be brought about by augmenting the randomness of φ_1 and φ_2 with a mechanism that induces a commotion to the structure of the resultant matrix, RE.

In doing so, when swarm consists of permutation matrices, we have to modify Equation (4-20) to Equation (5-1).

$$P = [\varphi_1(\Gamma_1 P_L^m) + \varphi_2(\Gamma_2 P_G^m)][\varphi_1\Gamma_1 + \varphi_2\Gamma_2]^{-1} \quad (5\text{-}1)$$

In Equation (5-1), Γ_1, Γ_2 are two random permutation matrices of the identity matrix. As shown, the introduction of the two random permutation matrices will introduce the element of randomness to the structure of the resultant matrix, and will accordingly account for the intended appeal towards all structural variations in the entirety of the solution space.

5.2. Surface Representation

In sections 2.6, 3.1, and 3.2, we explored different variations to represent deformable objects envelope surfaces. Among the methods described, point cloud with geodesic distances among every two pairs of landmarks was identified as the topology selected to represent candidate deformable objects. The landmarks on the topologies could be the coordinates of the constituents of the corresponding deformable objects (e.g., in the case of proteins the coordinates of carbon atoms imported from protein data banks). Or, the landmarks could be the vertices of the tessellated three dimensional objects. In some of our simulation experiments we made use of the Gamma distribution [45] random variables and uniform distribution random variables to generate distance matrices. However, in the majority of the experiments we made use of vertices of real three

dimension shapes and calculated the distance matrix among them with the Dijkstra algorithm as described in Table 2.1.

5.3.Constraint Based Objective Function: Lagrange and Penalty

In navigating the solution space using quantum PSO, we are interested in identifying the permutation matrix P which may relate, and establish isometry between, the two topologies representing two deformable objects (as specified by Equation (3-5)). Accordingly, as quantum PSO navigates through the solution space, the intermediary permutation matrices must maintain their structural integrity, i.e., they must adhere to the fundamental structural integrity of a permutation matrix (as defined by Equation(3-2)). The enforcement of the integrity of the permutation matrix (Equation (3-2)) on the equation to register isometry (Equation (3-5)) is accomplished by means of constraints imposed upon the intermediary P generated while the swarm evolves in the quantum space. In some of the experiments conducted for this research effort, we made use of two constraint mechanisms to sustain the integrity of P over the course of swarm's evolution; namely, Lagrange Multipliers and the Penalty Method. These two methods were chosen as they are widely used in mathematical optimization. The two methods are suited for our objective as both methods strategize to locate the extrema of a function subject to the constraints.

Further, inherent in our optimization effort is the normalization pertinent to the two statements in the equation to calculate isometry (Equation (3-5)). In all our constraint based experiments conducted in this chapter, we made use of Frobenius [46] norm as the de facto standard for matrix norm. The Frobenius norm of a $m \times n$ matrix A may be defined as the square root of the sum of the absolute squares of all its elements (as defined by Equation (5-2)) [46].

$$|| A ||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} | a_{ij} |^2} \qquad (5\text{-}2)$$

Alternatively the Frobenius norm may be defined by square root of the matrix trace of AA^H wherein A^H is the conjugate transpose[46]. Hence Equation (5-2) may be restated by the Equation (5-3).

$$|| A ||_F = \sqrt{tr(AA^H)} \tag{5-3}$$

The Frobenius norm is widely used in numerical linear algebra operations. It has the useful property of being invariant under matrix operations such as rotation [47]. We also experimented with Max norm [47]; however, as will be explained in the next sections, the choice of the norm did not have significant influence on the results obtained.

5.3.1. Lagrange Multipliers

In optimization, the theory of Lagrange Multipliers [48] is a mechanism which provides strategy to locate the extrema of a function subject to some constraints (as defined by Equation (5-4))

$$\arg\min_{x,y} f(x,y) \mid g(x,y) = c \tag{5-4}$$

By introducing a new parameter, λ or Lagrange multiplier, the Equation (5-4) may be reformulated in .

$$\Lambda(x,y,\lambda) = f(x,y) \pm \lambda\,(g(x,y) - c) \tag{5-5}$$

Accordingly, we are searching for (x,y) assuming that $g(x,y) = c$ and $f(x,y)$ has continuous first derivatives.

In our case, however, $g(x,y)$ should enforce the formation of a permutation matrix. Neither $g(x,y)$ nor $f(x,y)$ have partial first derivatives. We may, however, formulate the enforcement of permutation matrix by utilizing two multipliers-one for the enforcement of at most one unique single instance of 1 per column, and the second constraint to enforce at most one single instance of 1 per row (as formalized by Equation (3-2)).

$$\begin{matrix} C_{col} : PI_N = I_N \\ C_{row} : P^T I_N = I_N \end{matrix} \mid I_N = [1_1, ..., 1_N]^T \tag{5-6}$$

Thus, the enforcement of the integrity of the permutation matrix may be formulated by two Lagrange multipliers, λ and μ as stated by Equation (5-7).

$$\lambda C_{col} + \mu C_{row} = c \mid \lambda \in \mathbb{R}, \mu \in \mathbb{R} \tag{5-7}$$

Having incorporated Equations (5-2) and (5-7), we may proceed to define an objective function for isometry distance calculator (Equation (3-5)) with the two corresponding constraints.

$$\| PX_d P^T - Y_d \|_F + \lambda \| C_{col} \|_F + \mu \| C_{row} \|_F \leq \varepsilon \tag{5-8}$$

Equation (5-8) is the objective function that may be supplied to a Quantum PSO algorithm for optimization in order to detect the permutation matrix that provide the optimal solution given the constraints imposed. It is noteworthy that there are three variables, $P, \lambda,$ and μ to optimize in Equation (5-8).

5.3.2. Experimental Results with Lagrange Multipliers

The experiments conducted in this chapter consisted of generating a distance matrix X_d of dimension n using a probability distribution such as gamma distribution. We then made use of a random permutation matrix P in dimension n to transform X_d by way of relation PX_dP^T to generate Y_d or ($PX_dP^T = Y_d$). Then, X_d and Y_d were the input parameters to quantum PSO in order to retrieve the original P.

Extensive experiments were conducted to optimize variables $P, \lambda,$ and μ in Equation (5-8). The range of experiments, however, were limited to topologies represented by number of vertices whose sizes ranged from 5 to 12. The result predominately indicated that regardless of the values assumed by λ and μ, the constraints C_{col} and C_{row} may enforce the integrity of P as permutation matrix only for topologies represented by

vertices smaller than 9. As the size of vertices increased to values larger than 8, it was observed that $\lambda,\ \mu,\ C_{row},\ \text{and}\ C_{col}$ lacked the necessary constraining energy to enforce and to sustain the integrity of permutation matrix $P.$ Accordingly, for topologies represented by point clouds $n \times n$ wherein $n > 8$, it was not possible to produce a P that would satisfy the Equation (5-8).

Further, a surface represented by a point cloud of $n \times n$ geodesic distances wherein $n > 8$ constitutes a matrix of more than 8×8 entries. Consider when $n = 9$; we are dealing with optimization efforts involving matrices with 81 entries (the geodesic distances between n vertices). As the swarm traverses through the solution space, each swarm member which is a matrix with 81 constituent entries would have to be updated (by Harmonic Oscillator Potential-Equation (4-18)). Accordingly, if the swarm size is m, after each iteration, $81m$ entries would have to be updated, and ultimately optimized. The overall size of the variables involved demands significant computational resources in order to successfully fulfill the optimization objectives. In addition, enforcing the integrity of permutation matrix is bequeathed to the Lagrange multipliers $\lambda \text{ and } \mu$ in Equation (5-8). As noted earlier, for cases wherein $n \leq 8$ the multipliers could successfully maintain the integrity of produced P. Nonetheless, as n assumes larger values the constraining energy of the multipliers dissipate as the constraining energy would have to be divided among increasingly larger constituent members of the $P.$ Our results pertaining to the experiments to optimize Equation (5-8) indicated that the constraining energy of the Lagrange multipliers may successfully produce the desired P only for cases of $n \leq 8$, and may not be scalable for $n > 8$.

5.3.3. Quadratic Penalty Method

In our second method to optimize Equation (3-5), we make use of the Quadratic Penalty [49] method. In the Penalty method we introduce the original objective function plus a new term for each constraint imposed. The additional term for the constrain imposes a penalty every time the function deviates from the constraints. The severity of the penalty,

nevertheless, amplifies as the magnitude of deviation increases. Accordingly, the penalty term always has some positive value. The magnitude of the penalty is controlled by a penalty coefficient. In this effort, we make use of the *quadratic penalty function*, in which the penalty terms are the squares of the constraint violations. In the context of the simulation performed, the first few iterations may be penalized lightly for violating the constraints. Nevertheless, as the simulation proceeds towards convergence then even minor violations may be severely penalized by increasing the magnitude of the constraint coefficient. The penalty method may be formalized according to the Equation (5-9).

$$\begin{aligned} &\arg\min_{x,\mu} \Phi(x,\mu) = f(x) + \mu \sum_i c_i^2(x) \\ &\mu > 0 \text{ is the penalty cooefficient} \end{aligned} \tag{5-9}$$

As shown, we may penalize the constraint violations by driving $\mu \to \infty$. Therefore, we may consider a sequence of $\{\mu_k\}$ with $\mu_k \to \infty \text{ as } k \to \infty$ [49] in order to locate the optimal value of x_k and μ_k to optimize $\Phi(x,\mu)$.

It is noteworthy that the Penalty method has one primary advantage over the Lagrange Multiplier given that there is only one additional parameter to optimize in the Penalty approach.

Having incorporated Equation (5-9), we may now proceed to define the objective function for the equation to register isometry (Equation (3-5)) using the Penalty method in Equation (5-10).

$$|| PX_d P^T - Y_d ||_F + \mu(|| C_{col} ||_F^2 + || C_{row} ||_F^2) \le \varepsilon \tag{5-10}$$

Alternately, we may impose an inverse relationship with respect to the penalty coefficient between the two terms by introducing Equation (5-11).

$$f(P,\mu) = (1-\mu) || PX_d P^T - Y_d ||_F + \mu(|| C_{col} ||_F^2 + || C_{row} ||_F^2) \le \varepsilon \tag{5-11}$$

Equation (5-11) was the objective function that was given to the quantum PSO algorithm for optimization in order to detect the permutation matrix that provide the optimal solution given the constraints imposed. As illustrated, there are two variables to optimize P and μ.

5.3.4. Experimental Results with Penalty Method

We conducted extensive experiments to optimize P and μ in Equation (5-11). Our experiments ranged over topologies represented by $n \times n$ matrices wherein $4 \le n < 15$. The experimental results proved that penalty method had superior performance over Lagrange multipliers in producing the targeted permutation matrix P. In all cases, not only the targeted P was successfully identified by the quantum PSO but also the constraints were able to successfully sustain the integrity of the permutation matrix over the course of the swarm trajectory across the solution space.

The obtained results, nonetheless, were rendered at high computational costs. For $n \times n$ surfaces where $n = 12$ we had to utilize swarm sizes of more than 800 particles. Given the large swarm size coupled with the size of the solution space (12! =479,001,600), the targeted permutation matrix was located after 5 hours computation on a single core machine. For $n = 13$ and $n = 14$ the computational costs significantly increased and necessitated the modification to the programming logic to use multi-core CPUs. The parallel processing logic enabled to obtain results with comparable computational times to $n = 12$ for the cases where $12 < n < 15$.

In conclusion, the results indicated that the Penalty method is a more effective approach than Lagrange Multipliers, since the Penalty enabled us to locate the targeted P with reasonable amount of computational resources when $n \le 14$ (or the size of solution space was smaller than 14! or 87,178,291,200). However, the computational resources required (the size of the swarm and the CPU cycles) significantly increased as the value of n enlarged. Accordingly, for cases where $n \ge 20$ (or the size of the solution space is larger than 20! or 2,432,902,008,176,640,000) we anticipate that we may require massive

parallel CPU farms in order to optimize the objective function with the penalty method (Equation (5-11)).

5.3.5. Lagrange and Penalty Analysis

In this section we provide more in-depth analysis identifying subtler causes for both Lagrange and Penalty to fail in scaling up for larger values of n.

First, as n assumes larger values the solution space correspondingly expands by factorial order, that is as $n \to n+1$, the solution space spreads out by $n! \to n!(n+1)$. This implies that the probability of randomly identifying the targeted permutation matrix also diminishes by factorial order ($1/n!$ to $1/(n+1)!$). Consider the objective functions defined by Equations (5-8) and (5-11) wherein we endeavor to isolate the global minimum, and in both Equations the global minimum is achieved by the targeted permutation matrix P. When n assumes larger values, the probability for the swarm to encounter a wider array of local minimums amplifies, while the probability for the swarm to encounter the P which gives the global minimum deteriorates. One may consider to increase the size of the swarm in order to offset for the enlargement of the solution space. Although increasing the size of the swarm is theoretically possible, from a computational perspective the size of the swarm cannot be increased by factorial order. One needs to be reminded that the swarm size of m constitutes of m particles where each particle is represented by a permutation of identity matrix of dimension n. The computational complexities associated with matrix operations deteriorates performance as the size of the swarm increases. The computational consideration forbids admitting more particle members to the swarm by factorial order. Yet, the probability for the swarm to encounter more local minimums increases by factorial order as n enlarges. Accordingly, the increased vastness of the solution space admits more opportunities for the swarm to deviate from the trajectories that would terminate in the global minimum, or the targeted permutation matrix P.

Second, as stated earlier, the objective for the swarm is to follow the trajectories that would eventually terminate in the global minimum (by the targeted permutation matrix P). Consider the objective function with the penalty method (Equation (5-11)). Each point u_k in the trajectory is considered closer to the global minimum if $f(u_k, \mu) < f(u_{k-1}, \mu)$. Yet, although u_k may bring a numerically more attractive value $f(u_k, \mu)$ and closer to the global minimum at $f(P, \mu)$, it may also trap the swarm into a local minimum from which the swarm may have to exert enormous energy to escape.

$$\mathrm{X}_d = \begin{matrix} 0 & 2.86571 & 2.62673 & 4.01039 & 3.92631 \\ 2.86571 & 0 & 5.02911 & 3.25239 & 5.28119 \\ 2.62673 & 5.02911 & 0 & 3.23432 & 4.24928 \\ 4.01039 & 3.25239 & 3.23432 & 0 & 5.07305 \\ 3.92631 & 5.28119 & 4.24928 & 5.07305 & 0 \end{matrix}$$

$$\Upsilon_d = \begin{matrix} 0. & 2.62673 & 3.92631 & 2.86571 & 4.01039 \\ 2.62673 & 0. & 4.24928 & 5.02911 & 3.23432 \\ 3.92631 & 4.24928 & 0. & 5.28119 & 5.07305 \\ 2.86571 & 5.02911 & 5.28119 & 0. & 3.25239 \\ 4.01039 & 3.23432 & 5.07305 & 3.25239 & 0. \end{matrix}$$

$$\Gamma = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{matrix}$$

Figure 5-2 Two distance matrices transformed by permutation matrix Γ

Speaking in algorithmic terms, the energy required to escape from a potential local minimum may only be realized by more iterative search in the solution space to find a trajectory not only to escape from the trap in a local minimum, but also to re-orient the swarm towards the targeted global minimum.

Let us explain this second point with an example. Consider the surface X represented by the geodesic distance matrix X_d in dimension 5. Now consider the second geodesic distance matrix Υ_d which is a transformation of X_d produced by the permutation matrix Γ in Figure 5-2. Thus, we have $\Gamma \mathrm{X}_d \Gamma^T = \Upsilon_d \Rightarrow \| \Gamma \mathrm{X}_d \Gamma^T - \Upsilon_d \|_F = 0$. As stated earlier, Γ

is an arbitrary permutation of the identity matrix of dimension 5. Next, let us examine some other permutations, Γ_α and Γ_β, of the identity matrix of dimension 5 and the sort of transformations they might produce when they are applied to X_d.

As shown in Figure 5-3, the permutation Γ_β produces a smaller real value than the real value produced by permutation Γ_α. Naturally the swarm will be more strongly attracted toward Γ_β rather than Γ_α as the former produces value closer to the global minimum produced by the permutation Γ.

$$\Gamma = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{matrix} \qquad \| \Gamma \mathrm{X}_d \Gamma^T - \Upsilon_d \|_F = 0$$

$$\Gamma_\alpha = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{matrix} \qquad \| \Gamma_\alpha \mathrm{X}_d \Gamma_\alpha{}^T - \Upsilon_d \|_F = 3.02485$$

$$\Gamma_\beta = \begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix} \qquad \| \Gamma_\beta \mathrm{X}_d \Gamma_\beta{}^T - \Upsilon_d \|_F = 2.31167$$

Figure 5-3 Three permutations of dimension 5 and the real values that they produce

Let us now examine the structural similarity of Γ_α, Γ_β, and Γ (By structural similarity we are referring to the number of row changes that a permutation matrix must make in order to transform to another). A careful examination of the three permutation matrices reveals that Γ_α must make two row changes (rows 4 and 5) in order to transform to Γ. On the other hand, Γ_β must make four row changes (rows 1,2, 4, and 5) in order to transform to Γ. Hence, Γ_α is structurally closer and is structurally more similar to Γ. Notwithstanding larger structural distance, Γ_β produces real value closer to the global

minimum than the real value which Γ_α produces. The revelation here signifies that structurally similar permutation matrices to the target may produce larger real values than the structurally dissimilar permutation matrices. Yet, structurally similar permutation matrices require fewer Harmonic Oscillator Update energy in order to transform to their target. In other words, Γ_α demands less search energy (only two row changes) than Γ_β (which needs four row changes) in order to transform to Γ.

Consider the situation wherein the swarm is gravitated to and is settled at Γ_β as its last best global position. In attempting to extend its trajectory towards the global minimum, the swarm searches the solution space in order to find a permutation matrix that produces a closer value to the global minimum. Yet, the swarm would exhibit no gravitation towards Γ_α as the latter produces a value larger than Γ_β in spite of the fact that Γ_α is structurally very similar to the global minimum produced by Γ. In this situation the swarm is trapped in Γ_β. The swarm may only escape from this trap if and only if the swarm can locate Γ directly and bypassing altogether Γ_α. Assuming that $S = \bigcup_{i=1}^{v} \Gamma_i$ is the set of all Γ_i that have structurally closer distance than Γ_β to Γ but produce larger real values than Γ_β, the question arises as how large v can be, or what is the size of S. It is logical to deduce that as the solution space expands, the size of S also increases. Consequently, in larger solution space there is higher likelihood for the swarm to trap in the vast array of local minimums. Every escape from a potential trap in a local minimum demands mitigating iterations by Harmonic Oscillator Update in the form of solution space search so that the swarm may continue its trajectory towards the global minimum. However, repeated traps in the abundance of the local minimums deteriorates the optimization performance.

We may conclude that in order to make quantum PSO more scalable for larger surfaces, the algorithm must be improved on two fronts. First, the algorithm must ensure the integrity of permutation matrices is sustained for large topologies. And second, a measure

of structural similarity must also be taken into account as the swarm trajectory is charted through the solution space towards the global minimum.

5.4.Particles with Permutation Ranking Numbers

In the previous section we identified two factors inhibiting quantum PSO to scale up for surfaces represented by a large number of vertices. To address these factors, we introduce a paradigm shift in the policy by which we admit particles into the swarm.

In our experiments so far we had assumed that the swarm members constituted of either permutations of the identity matrix or stochastic matrices (recall section 5.1.2). In this section, we propose using an integer value associated with the permutation ranking as members of the swarm particles. As we shortly illustrate, swarms that consist of arrays of unique permutation ranks (where a permutation rank is an integer value and a distinct member of the swarm) can address the inhibiting factors identified and analyzed in section 0, and thus enabling quantum PSO to scale up for topologies represented by larger number of vertices.

5.4.1. Lexicographic Permutation

We start by providing a definition for Lexicographic Ordering before extending this definition to Lexicographic Permutation. Lexicographic Ordering is an ordering [50] for Cartesian product of any two sets A and B such that $\forall a_i, \forall a_j \in A$ and $\forall b_i, \forall b_j \in B$ then (a_i, b_i) and $(a_j, b_j) \in A \times B$ and $(a_i, b_i) < (a_j, b_j)$ *iff* $a_i < a_j$, or $(a_i = a_j)$ and $(b_i < b_j)$. This definition may be extended to a Cartesian product of arbitrary length by recursion, e.g. $(A \times B) \times C$. Lexicographic ordering when applied to a list, produces all possible permutations in which case the elements are arranged in increasing order. Consider the list {123} wherein the lexicographic order of the permutations produce L ={{123}, {132}, {213}, {231}, {312}, and {321}}. An examination of the permutation list L demonstrates that the definition of Lexicographic Ordering given above is held by the order in which the elements in L are arranged. As such, we may define that the Lexicographic Permutation

is the scheme that accords the total ordering of the permutation list L based on their alpha-numeric values . For example, in the case of list {123} the numerical value signifies the criteria to order the permutation. However, the list may constitute the elements of alphabets. Therefore, the list {a, b, c} would produce L ={{a, b, c}, {a, c, b}, {b, a, c}, {b, c, a}, {c, a, b}, and {c, b, a}} as its permutation ordering accordingly.

Another quality which is of significance in lexicographic permutation ordering is that each successive permutation in L is generated based on the principle of minimum change; this means that successive permutations in L differ by the minimum number of transpositions possible. For example, in the two examples given above each permutation differs from its neighbor by exactly one transposition. Therefore, to reiterate, in lexicographic permutations ordering the successive permutations are generated based on the principle of least transposition.

Associated with each permutation in the generated list L, we may assign a ranking number in the form of an integer value from 0 to $n! - 1$ where n is the size of the original list [50]. The integer value or the rank identifies not only the exact permutation in L but also its location within L.

Expanding this definition to permutation matrices, we may proceed to identify all $n!$ permutations of the identity matrix of dimension three as follow.

0	1	2	3	4	5
$\begin{matrix}1&0&0\\0&1&0\\0&0&1\end{matrix}$	$\begin{matrix}1&0&0\\0&0&1\\0&1&0\end{matrix}$	$\begin{matrix}0&1&0\\1&0&0\\0&0&1\end{matrix}$	$\begin{matrix}0&1&0\\0&0&1\\1&0&0\end{matrix}$	$\begin{matrix}0&0&1\\1&0&0\\0&1&0\end{matrix}$	$\begin{matrix}0&0&1\\0&1&0\\1&0&0\end{matrix}$
{123}	{132}	{213}	{231}	{312}	{321}

Figure 5-4 Permutations of identity matrix dimension 3

Noteworthy in the Lexicographical ordering is the principle of minimum transposition where successive permutations differ by the least number of transpositions possible. Recalling the definition of structural similarity from the preceding section, we may conclude that successive permutation matrices are arranged in structurally similar manner

(to the extent possible). The notion of structural similarity comes to more illustrative prominence for larger values of n; yet, due to the space limitations we do not show all permutations when $n > 3$. It suffices to reiterate that the generation of each successive permutation is based on principle of minimum change which means successive ranks represent matrices that are structurally similar to their neighbors.

Moving to {1,2,3,4}, its lexicographic permutation ordering is given by L = {{1,2,3,4},{1,2,4,3},{1,3,2,4},{1,3,4,2},{1,4,2,3},{1,4,3,2},{2,1,3,4},{2,1,4,3},{2,3,1,4},{2,3,4,1},{2,4,1,3},{2,4,3,1},{3,1,2,4},{3,1,4,2},{3,2,1,4},{3,2,4,1},{3,4,1,2},{3,4,2,1},{4,1,2,3},{4,1,3,2},{4,2,1,3},{4,2,3,1},{4,3,1,2},{4,3,2,1}}. For the sake of brevity, only 8 of the permutations of the identity matrix of dimension 4 is shown below (Figure 5-5).

Since each permutation matrix may be identified by an integer value, we may define Equation (5-12). Let us define $\aleph = [0, n!-1] \subset \mathbb{Z}^+$. Also, let us define Γ to be the set of all permutation matrices of dimension n. Then we may define the bijective mapping Q such that

$$Q : \aleph \leftrightarrow \Gamma$$
$$\forall x_i \in \aleph, \exists\, \Gamma_i \in \Gamma \mid Q(x_i) = \Gamma_i \tag{5-12}$$

0
$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$
{1,2,3,4}

1
$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix}$$
{1,2,4,3}

2
$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$
{1,3,2,4}

3
$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{matrix}$$
{1,3,4,2}

20
$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{matrix}$$
{4,2,1,3}

21
$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{matrix}$$
{4,2,3,1}

22
$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$
{4,3,1,2}

23
$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{matrix}$$
{4,3,2,1}

Figure 5-5 Permutations of identity matrix dimension 4

Equation (5-12) indicates that there is a one to one mapping Q between an integer in the interval $\aleph$ and a permutation matrix in Γ. Consider the example in Figure 5-5 wherein n or the dimension is 4. As shown, in light of the Equation (5-12), there is a one to one mapping between an integer in the interval of [0 , 4!-1] and a unique permutation matrix of dimension 4.

In the next section, we describe how permutation ranking may enable quantum PSO to scale up for topologies larger than those that Lagrange Multipliers and/or Penalty methods could deliver.

5.4.2. Particles with Permutation Ranking

In section 5.3 we had conducted our experiments using two procedural techniques. First the swarm was initialized to an array of particles wherein each particle was a permutation matrix or stochastic matrix (see section 5.1.2 for details). Second the swarm gravitated towards point x_p if and only if, in Equation (5-11), $f(x_p, \mu) < f(x_{p-1}, \mu)$ wherein x_p and x_{p-1} were two successive position over the course of the swarm's trajectory towards the global minimum. As stated in section 5.3, we made use of Frobenius norm in order to produce real numbers (from the intermediary matrix results) to compare and to rank the resultant $f(x_p, \mu)$ and $f(x_{p-1}, \mu)$.

In this section, we propose to abandon both these procedural techniques in favor of the permutation based ranking. By employing the new techniques (described below) we suggest that the quantum PSO can overcome the inhibiting factors described in section 0. Recall the analysis in section 0 wherein we identified factors preventing the quantum PSO to scale up for surfaces with large number of vertices. As we will illustrate, after describing in detail the permutation ranking techniques, we may enable quantum PSO to scale up for larger number of vertices.

15							33							44						
1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0

127							555							3015						
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0

Figure 5-6 Arbitrary swarm of size 7

First, we propose to initialize the swarm to the integer values randomly selected from the set of integers $\{0,1,2,3,...,n!-1\}$. In this initialization scheme, the swarm constitutes an array of integer values selected randomly from $[0,n!-1]$. As per the definition given in the preceding section (Equation (5-12)), each integer value identifies a unique permutation matrix of dimension n. For example, consider (Figure 5-6) a swarm made of 6 particles, and $n=7$. An arbitrary swarm may constitute of the array of particles such as #(15, 33, 44, 127, 555, 3015) or any other randomly selected integer from $[0,7!-1]$. Each particle, through its integer value, nevertheless, identifies its corresponding permutation matrix (Figure 5-6) of dimension 7 according to the lexicographic ordering defined in section 5.4.1. Further, the integer value of each particle also identifies the location of the permutation matrix it represents in the lexicographic ordering. For example, particle 15 represents the 15th permutation matrix, particle 33 represents the 33rd permutation matrix and so forth in the list of all permutations of identity matrix of dimension 7.

Second, we propose to evaluate a point x_u in the swarm's trajectory not by its Frobenius norm (in the objective function) but by the number of zeros that it produces (again in the objective function). To clarify this point let us revisit the example given in Figure 5-2 and

Figure 5-3. In that example three permutation matrices $\Gamma, \Gamma_\alpha,$ and Γ_β produced real values 0.0, 3.02485, and 2.31167 respectively (using Frobenius norm). However, modifying the objective function to count the number of zeros that the permutation matrices produce we get totally different result as shown in Figure 5-7.

$$\Gamma = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{matrix} \qquad Count[\Gamma \mathrm{X}_d \Gamma^T - \Upsilon_d, 0] = 25$$

$$\Gamma_\alpha = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{matrix} \qquad Count[\Gamma_\alpha \mathrm{X}_d {\Gamma_\alpha}^T - \Upsilon_d, 0] = 13$$

$$\Gamma_\beta = \begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix} \qquad Count[\Gamma_\beta \mathrm{X}_d {\Gamma_\beta}^T - \Upsilon_d, 0] = 5$$

Figure 5-7 Three permutation matrices and the number of zeros they produce

As the results in Figure 5-7 illustrate, at position x whose permutation matrix is represented by Γ, the objective function produces 25 zeros. Since the dimension $n = 5$, it is indicative that the objective function has produced $n^2 = 25$ zeros. Given that the maximum number of zeros is realized, we may conclude that the targeted permutation matrix is identified when the quantum PSO has identified Γ that produces n^2 zeros. Similarly, we may conclude that position x_α articulated by permutation matrix Γ_α is preferred, and closer to target, than the position x_β represented by Γ_β (the former produces 13 zeros whereas the latter produces only 5). Γ_α may produce more zeros as Γ_α is structurally very similar to Γ (Γ_α differs from Γ by only two rows). However, Γ_β produces fewer zeros because Γ_β differs from Γ by four rows (thus, Γ_β is structurally less similar to Γ than Γ_α is).

We hypothesize that by introducing these two modifications we will have addressed the factors identified in section 0 inhibiting the Quantum PSO to scale up for surfaces represented by topologies comprised of more than 15 vertices (size 15 was the best result we could achieve with Quadratic Penalty Method).

To clarify this hypothesis, in section 0 it was shown that when the swarm made of the particles represented by permutation matrices of $m \times m$ size, then the optimization energy of the Harmonic Oscillator Update at each iteration would have to be divided among all the constituent $m \times m$ members of each and every particle in the swarm. Further, it was argued that the Lagrange and the Penalty multipliers could not exert the necessary constraining energy to enforce the integrity of the permutation matrices along the swarm's trajectory when the matrices enlarged beyond dimension 15.

We suggest that by replacing the permutation matrices in the swarm with their corresponding rank in the lexicographical ordering, we have addressed both these issues. The rank in the lexicographical ordering is just an integer value and not a permutation matrix. Accordingly, the effort in quantum PSO is reduced from optimizing an $m \times m$ matrix to a single integer value per particle in the swarm. Second, by utilizing the rank (and not the corresponding permutation matrix), the services of the constraining mechanisms and the multipliers to enforce the integrity of the permutation matrices are no longer warranted. Accordingly, we may re-formulate the objective functions by using the integer rank corresponding to its permutation matrix (as opposed to using permutation matrix itself). The other inhibiting factor was the structural dissimilarity causing the swarm to trap in a local minimum prevalent in the solution space (recall section 0 for an in-depth analysis). As illustrated in Figure 5-2 and Figure 5-3 , Γ_β traps the swarm in a local minimum-the corresponding real value produced by the Frobenius norm of the objective function had eliminated the propensity for the swarm to gravitate towards Γ_α, and/or similar permutation matrices, that were structurally more similar to the targeted global minimum. Yet, by modifying the objective function to count for the resultant zeros (rather than using the real value produced by the Frobenius norm), this

issue is also mitigated. As Figure 5-7 illustrates, the quantum PSO is more attracted towards Γ_α than towards Γ_β given that the former produces more zeros than the latter does (because the former is structurally more similar to the target).

Given the argument presented in this section, we proceed to formulate a new objective function as shown in Equation (5-13). Let's recall Q to be mapping defined in Equation (5-12) and $x_r \in \aleph$ as defined in (5-12).

$$f(x_r) = Count[Q(x_r)^T Y_d Q(x_r) - X_d, \# \leq \varepsilon] \leq Dim[Q[x_r]]^2 \quad (5\text{-}13)$$

The notion $Count[M, \# \leq \varepsilon]$ states, "Count all entries in the matrix M that are smaller or equal to ε [1]". However, as opposed to the objective functions in (5-8) and (5-11) the goal in (5-13) is to maximize $f(x_r)$ such that $f(x_r) = Dim[\Gamma]^2$.

5.4.3. Experimental Results with Particles with Permutation Ranking

The wide range of experiments that we conducted overwhelmingly confirmed the improvements in results obtained for the new objective function (Equation (5-13)). The modified quantum PSO was able to scale up and match up topologies up to 20 vertices at significantly shorter computational times compared with the results obtained for Lagrange Multipliers and Quadratic Penalty method (Table 5.1).

Method	Number of Vertices (n)	Dimension of the Solution Space ($n!$)	Swarm Size	Average time to complete experiment
Lagrange Multipliers	8	40,320	300	1 hours , 20 minute
Penalty Method	12	479,001,600	800	5 hours, 40 minutes
Permutation Ranking	20	2,432,902,008,176,640,000	120	15 minutes

Table 5.1 Comparing the three methods: Lagrange, Penalty, and Permutation Ranking

[1] See section 2.5 for the significance of distortion ε in isometry

For Permutation Ranking we conducted 30 experiments wherein the global seeds in each experiment was unique. We were able to achieve 100% success rates over all 30 experiments.

5.4.4. Permutation Ranking Analysis

As Table 5.1 shows, Permutation Ranking has superior performance over the other two methods. The average run to completion time was not only significantly shorter but also the swarm size needed was also much smaller (see section 5.4 for an analysis of the merits of the Permutation Ranking over the other two).

One factor whose influence comes to prominence in Permutation Ranking is the choice of value for g. We need to recall Delta Oscillator Potential (Equation (4-18)) and the influence of g in overall quantum PSO behavior (recall section 4.3.5). The choice of g has a significant influence on the behavior of Permutation Ranking method. Small values of g enable the swarm to make large jumps across the solution space whereas the larger values of g make the jumps smaller in size. Naturally, it would be preferable for the swarm to conduct large jumps when it commences its navigations across the solution space. Large jumps enables the swarm to perform a global survey of the entirety of the solution space; yet, as the swarm approaches the regions where the target is located, the swarm should conduct more local searches, and thus smaller jumps are more desirable. Accordingly, we propose that g should vary from a small value at the beginning of swarm navigation to a large or very large value prior to identifying the swarm's target. In fact for the experiments that we conducted we had to vary the value of g from the initial 1.6 at the start of the optimization to the final value of 500000 just before identifying its target. Questions arise as to how does the swarm know to switch from the global search to local search. Also, by what factor g should be increased in order to conduct local search?

5.4.4.1. Intelligent Sampling of the Solution Space

Let us formalize the notion of the "swarm's trajectory" from the preceding sections in light of the objective function articulated by Equation (5-13). In navigating the solution space to locate its target, the swarm commences its search from an arbitrary location x_0 in the solution space. At x_0, then we have $f(x_0) \geq n$ [1]. Also let's identify x_p to denote the location corresponding to the target. Therefore at x_p, $f(x_p) = n^2$. Accordingly[2], the set $ST = \bigcup_{r=0}^{p} x_r$ constitute the swarm trajectory $(x_0, x_1, x_2, ..., x_{p-1}, x_p)$ in the solution space such that $f(x_h) < f(x_{h+1}),\ 0 \leq h < p.$ Given the analysis, in the preceding sections, we may state that the swarm moves from any x_h to x_{h+1} if and only if $f(x_{h+1}) > f(x_h).$ We may conclude that $\forall x_i \in ST \mid 1 < i \leq p-1,$ then $n < f(x_{i-1}) < f(x_i) < n^2$. Accordingly, as the swarm moves from x_{i-1} to x_i, it means the swarm is getting closer to the target at x_p where $f(x_p) = n^2.$

In the solution space there is exactly one $x_p \mid f(x_p) = n^2$. Again, recalling Equation (5-12), we may say that there is exactly one $Q(x_p) = \Gamma_p$. The question arises as to how many x_{p-1} or Γ_{p-1} exist in the solution space? Given that Γ_{p-1} is the second best position compared to Γ_p, it signifies that there are exactly two rows in Γ_{p-1} that are different from the rows in Γ_p. The question of in how many ways two rows can differ from Γ_p is of Binomial coefficient or $\binom{n}{2}$. As such, irrespective of the value of n, there is always $\binom{n}{2}$ or $\frac{n!}{2(n-2)!} = \frac{n(n-1)(n-2)!}{2(n-2)!} = \frac{n^2-n}{2}$ of x_{p-1} in the solution space. Alternatively we may

---

[1] $f(x_0) = n$ is the minimum possible value corresponding to n zeros in the diagonal of the resultant matrix. In other words, $f(x_0) < n$ is an impossibility.

[2] $f(x_p) > n^2$ is an impossibility as the distance matrix is of size $n \times n$.

paraphrase that irrespective of the value of n, there are always $\frac{n^2-n}{2}$ permutation matrices of type Γ_{p-1} in the solution space that are different from the target Γ_p by exactly two rows. This proves to be very encouraging as it would be much more computationally efficient for the swarm to search for one of the $\frac{n^2-n}{2}$ permutation matrices of type Γ_{p-1} in the solution space, i.e., the probability of finding one of the Γ_{p-1} is $\frac{n^2-n}{2n!}$ whereas the probability of finding Γ_p is only $1/n!$.

As the swarm navigates the solution space, the swarm may recognize a second best position x_{p-1} when $f(x_{p-1}) = (n-2)^2 + 4 = n^2 - 4n + 8$ [1]. Finally we need to recall the definition of lexicographic ordering given in section 5.4.1. The positions identified by $(x_0, x_1, x_2, ..., x_{p-1}, x_p)$ are none other than the integer values corresponding to the rank of their permutation matrices. The ranking of the permutations ordering are produced based on the principle of minimal transpositions. This implies that x_p is located very close to a narrow cluster of the $\frac{n^2-n}{2}$ of x_{p-1} in the solution space, i.e. wherever x_p lies in the solution space, its immediate neighbors are structurally very similar. In other words, x_p is always surrounded by a cluster of x_{p-1}. Let us consider some examples in order to clarify

[1] This formula was derived experimentally. Alternatively we may use the following

A. generate a distance matrix for surface X

B. generate a Permutation Matrix

C. generate distance matrix for surface Y from A and B (P.X.Transpose[P])

D. Exchange any two rows of the Permutation Matrix in 2 and name it NewP.

E. Calculate Count[Flatten[NewP.X.Transpose[NewP] - Y], 0.0]

how permutation ranking based on the principle of minimum transposition can concretize the formation of cluster of high value targets, x_{p-1}, and x_{p-2} around x_p.

Recall the two distance matrices X_d and Y_d that were related by the permutation matrix Γ from Figure 5-2. As shown there, the dimension of the distance matrices is $n=5$. Accordingly, the size of the solution space is $n!=120$ permutation matrices . Figure 5-8 illustrates the list of the results obtained when X_d, Y_d and all $n!=120$ permutation matrices in their ranking order, 0 to $n!-1$, are evaluated by Equation (5-13). As seen in Figure 5-8, there is exactly one $x_p \mid f(x_p)=n^2=25$ in the results. There are also $\frac{n^2-n}{2}=10$ of $f(x_{p-1})=(n-2)^2+4=13$ in the results. We may also observe a cluster of 6 $f(x_{p-1})=13$ surrounding the single $x_p \mid f(x_p)=n^2=25$.

{7,5,5,5,5,13,9,7,5,5,7,9,7,5,5,5,7,7,9,5,5,7,5,13,9,7,5,5,7,25,13,13,7,7,13,13,7,7,5,9,7,13,5,7,9,5,7,7,5,5,5,5,13,9,9,7,7,9,7,5,5,5,5,7,5,7,5,5,7,5,5,5,9,7,7,5,7,5,5,5,5,7,13,7,7,5,5,9,9,5,7,5,5,7,5,5,7,9,5,7,5,7,5,5,7,5,7,13,5,7,9,5,5,9,5,7,7,5,7}

Figure 5-8 Results when n = 5 with X_d, Y_d and all $n!$ permutations evaluated in Equation (5-13)

Moving to $n=6$ and two arbitrary X_d, Y_d in Figure 5-9, we can draw the very same conclusions. Figure 5-9 contains all $n!=720$ results. As illustrated, there is exactly one $x_p \mid f(x_p)=n^2=36$ in the results. There are also $\frac{n^2-n}{2}=15$ of $f(x_{p-1})=(n-2)^2+4=20$ in the results. We may also observe a cluster of 7 $f(x_{p-1})=20$ surrounding the single $x_p \mid f(x_p)=n^2=36$.

{8,8,6,6,8,20,12,12,8,8,12,12,8,8,6,6,8,8,12,6,8,8,6,20,12,12,8,8,12,36,20,20,12,12,20,20,12,12,8,8,12,12,20,8,12,12,8,12,8,8,6,6,8,20,12,12,8,8,12,12,8,8,6,6,8,8,12,6,8,8,6,8,6,12,8,8,6,12,8,8,6,6,8,20,12,12,8,8,12,12,8,8,6,6,8,6,8,8,12,6,8,8,12,6,8,8,6,12,20,8,12,12,8,8,12,6,8,8,6,8,8,6,6,6,6,12,8,8,6,6,8,8,6,6,6,6,8,6,8,6,6,8,6,12,8,8,6,6,8,20,12,12,8,8,12,12,8,8,6,6,8,8,12,6,8,8,6,8,8,6,6,6,6,12,12,8,8,8,8,8,8,6,6,6,6,8,8,6,6,6,6,6,6,8,6,8,6,8,8,6,6,6,6,12,8,8,6,6,8,12,8,8,6,6,8,6,6,6,8,6,8,8,8,6,6,6,6,8,12,6,8,8,6,8,12,6,8,8,6,12,12,8,8,8,8,20,12,12,8,8,12,12,8,8,8,8,12,8,12,8,8,12,8,8,8,6,6,6,6,12,8,8,6,6,8,8,6,6,6,6,8,6,8,6,6,8,6,12,8,8,6,6,8,8,8,6,6,6,6,6,6,8,6,8,6,6,6,6,8,6,8,8,6,6,6,6,8,6,6,8,8,6,6,8,6,12,8,8,6,8,6,8,6,6,6,6,8,6,6,8,6,6,6,8,8,6,6,6,8,8,12,6,8,6,8,6,8,6,6,8,8,6,6,6,6,12,8,8,6,6,8,8,6,6,6,6,8,6,8,6,6,8,6,12,12,8,8,8,8,8,8,6,6,6,6,6,6,8,8,6,6,6,6,8,8,6,6,20,12,12,8,8,12,12,8,8,6,6,8,8,6,12,8,8,6,6,8,8,12,6,8,12,8,8,8,8,12,8,6,6,6,6,8,6,6,8,6,8,6,6,8,6,8,6,6,8,12,8,8,12,8,6,8,6,6,8,6,6,6,6,8,6,8,8,6,8,6,6,6,6,6,8,8,6,6,8,6,6,6,6,8,12,8,8,6,6,8,8,6,8,6,6,6,8,8,6,6,6,6,6,6,8,6,8,6,8,6,12,8,8,6,8,6,12,8,8,6,12,8,8,6,6,8,8,6,12,8,8,6,12,8,20,6,6,8,6,8,6,8,8,12,6,8,6,8,8,12,6,8,8,12,6,8,8,6,6,8,8,12,6,8,8,12,12,20,8,12,6,8,8,12,6,8,6,8,6,6,8,6,6,6,6,8,6,8,8,8,8,12,8,12,6,6,6,8,6,8,8,6,6,6,6,8,8,6,6,8,8,12,6,6,6,8,6,8}

Figure 5-9 Results when n = 6 with X_d, Y_d and all $n!$ permutations evaluated in Equation (5-13)

Finally, moving to $n = 7$, we may realize that $n! = 5040$. Therefore, we only show partial results in Figure 5.10 for $n = 7$ to demonstrate the same trend we observed in the previous two examples as $n \to \infty$. As shown, Figure 5-10 depicts partial results when $n = 7$ which demonstrates the formation of a cluster of high value results around $x_p \mid f(x_p) = n^2 = 49$. In fact, a careful examination illustrates that there is no occurrence of minimum possible result, 7, in the formed cluster. Likewise in the preceding cases where $n = 5$ and $n = 6$, there were only two occurrences of minimum possible result, 5 and 6, in the corresponding formed clusters (see Figure 5-8 and Figure 5-9). This indicates that as $n \to \infty$ then the cluster formed around $x_p \mid f(x_p) = n^2$ becomes populated with only the higher value results, and smaller value results are pushed further away from $x_p \mid f(x_p) = n^2$.

{11,13,9,9,13,29,17,19,13,13,19,17,11,13,9,9,11,11,17,9,13,11,9,29,17,19,13,13,19,49,29,29,19,19,29,29,19,19,13,13,17,19,29,13,19,17,13,19,11,13,9,9,13,29,17,17,11,11,17,19,13,13,9,9,11,13,19,9,13,11,9,13,9,17,11,11,9,19,13,13,9,9,11,29,19,19,13,13,17,17,13,11,9,9,11,9,13,11,17,9,11,13,19,9,13,11,9,19,29,13,19,17,13,13,17,9,11,11,9,13,9,9,7,7,9,19,11,13,9,9,13,11,7,9,7,7,9,7,11,7,9,9,7,19,11,13,9,9,13,29,17,19,13,13,19,17,11,13,9,9,11,11,17,9,13,11,9,13,9,9,7,7,9,17,13,11,9,9,11,11,9,9,7,7,7,9,11,7,9,7,7,9,7,11,7,9,7,11,9,9,7,7,7,17,11,13,9,9,11,13,9,9,7,7,9,7,9,7,11,7,9,9,11,7,9,7,7,11,17,9,13,11,9,9,13,7,9,9,7,17,13,11,9,9,11,29,17,17,11,11,17,17,11,11,9,9,13,11,17,9,11,13,9,13,9,9,7,7,9,19,11,13,9,9,13,11,7,9,7,7,9,7,11,7,9,9,7,19,11,11,7,7,11,13,9,9,7,7,9,9,7,13,9,9,7,7,9,9,13,7,9,13,9,9,7,7,9,9,7,11,9,7,7,13,9,19,13,11,9,9,7,11,9,7,7,9,13,7,9,9,7,7,9,9,11,7,7,9,13,...}

Figure 5-10 Results when n = 7 with X_d, Y_d and all $n!$ permutations evaluated in Equation (5-13). Partial Results are shown here

The discovery of this cluster may be the basis for the swarm to change from global search to local search. Accordingly given the proximity of the x_p to x_{p-1}, the swarm switches from the global search to the local search by significantly increasing the value of g once any of the $\frac{n^2-n}{2}$ of x_{p-1} is located. In addition, as the swarm locates the cluster, it breaks into segments where each segment conducts local search in the vicinity of newly discovered x_{p-1}. In employing this approach, we were able to the produce the results shown in Table 5.1.

5.4.4.2. Variations in g

The transition of the swarm from global search to local is accomplished by increasing the value of g. By what factor should g be increased? As the swarm moves along the trajectory $(x_0, x_1, x_2, ..., x_{p-1}, x_p)$ the value of g should accordingly be increased to make the jumps smaller and the search more local. We conducted extensive experiments for value of n ranging from 8 to 20. When $n = 20,$ we found out that the initial value of $g = 1.75$ should gradually grow until the cluster is identified, and then g should be assigned a very large value such as 5000000 in order to ensure the local search for target

x_p is limited to within the cluster. Although we were able to reproduce the results illustrated in Table 5.1 consistently over more than 30 experiments, for reasons described in section 5.4.5, we decided to end further studies on cluster targeting and cluster local analysis. Nonetheless, we acknowledge that more experiments are warranted to more thoroughly investigate the pertinent behavior of g in influencing the trajectory of the swarm along $(x_0, x_1, x_2, ..., x_{p-1}, x_p)$.

5.4.5. Permutation Ranking, the Final Word

Table 5.2 depicts some attributes of the solution space.

n	Size of Solution Space: $n!$	$\frac{n^2-n}{2}$ or # x_{p-1}
20	2,432,902,008,176,640,000	190
21	51,090,942,171,709,440,000	210
22	1,124,000,727,777,607,680,000	231
23	25,852,016,738,884,976,640,000	253
24	620,448,401,733,239,439,360,000	276

Table 5.2 Some characteristics of the solution space

As $n \to n+1,$ we may observe that the size of the solution space enlarges from $n! \to n!(n+1)$. However, the number of x_{p-1} in the solution space is only added by n which is much smaller than $n+1$ factor by which the size of the solution space[1] has been multiplied. Alternatively, we may conclude that the size of the solution space is enlarged by much larger margin than the margin increase of x_{p-1} in the solution space. In light of this observation, we may arrive at the conclusion that the cluster containing the high value targets becomes insignificant in size as the solution space enlarges. Eventually as the solution space gets larger and larger, the cluster becomes more invisible to the swarm navigating the solution space.

[1] $\frac{(n+1)^2-(n+1)}{2} - \frac{n^2-n}{2} = \frac{n^2+2n+1-n-1-n^2+n}{2} = \frac{2n}{2} = n$

In fact in the simulation experiments that we conducted we could no longer locate x_p with 100% success rate for $n > 20$ unless we proportionally increased the size of the swarm. Nevertheless, increasing the size of the swarm resulted in severe performance degradations causing the run to completion time to exceed 180 minutes per experiment for $n = 21$. For swarm sizes of smaller than 2000 and for $n > 20$, the immensity of the solution space and the abundance of local maximums rapidly diminishes the effectiveness of the swarm to veer off the local maximums, and pursue its navigation towards its target at global maximum.

In our efforts to enable Quantum PSO to scale up for large number of vertices we have so far directed our efforts on three separate avenues. First, we employed Lexicographic permutation ordering whereby we reoriented the energy of Harmonic Oscillator Update function to optimize a single integer (an opposed to an $m \times m$ matrix per particle). Integer value is the most atomic type possible we may use. Second, we made more effective usage of g so that Quantum PSO not only to distinguish between global and local searches but also to adopt the two search types as the swarm navigates through the solution space.

n	$\frac{n^2 - n}{2}$	$P(x_{p-1}) = \frac{n^2 - n}{2n!}$
21	210	$210/21!$
22	231	$231/22!$
23	253	$253/23!$
24	276	$276/24!$

Table 5.3 Probabilities of finding second best as n enlarge

Visiting the third column of Table 5.3, we may formalize the ratio of the number of x_{p-1} to the size of the solution space as $P(x_{p-1}) = \frac{n^2 - n}{2n!}$. Yet, we see that as $n \to \infty$, then

$\underset{n\to\infty}{P}(x_{p-1}) = \lim_{n\to\infty}\frac{n^2-n}{2n!} = \lim_{n\to\infty}\frac{n(n-1)}{2n(n-1)(n-2)!} = \lim_{n\to\infty}\frac{1}{2(n-2)!} \approx \lim_{n\to\infty}\frac{1}{n!} = 0$. This signifies that as $n\to\infty$, then the distinction between x_{p-1} and x_p becomes unimportant and inconsequential ($\underset{n\to\infty}{P}(x_{p-1}) \approx \underset{n\to\infty}{P}(x_p)$). Accordingly, the improvements brought about by engineering the value of g to more effectively navigate through the local maximums towards any of the x_{p-1} will deteriorate as $n\to\infty$. Our third, and the last remaining, option remains to manipulate the size of the swarm to account for the enlargement in the solution space. The requirement for large swarm sizes had been anticipated for quantum PSO by Sun [36] and Mikki [9]. In addition, recall from section 4.3.3 that for dimension size of 200 we had to use swarm size made of 4000 particles. Yet realizing that at least a swarm size of 2000 along a computational time of 180 minutes is required for $n=21$, we come to conclusion that increasing the size of the swarm to account for $n\geq 22$ would bring about unacceptable performance with our 3.4 GHz CPU platform. For topologies $n\geq 22$ we may have to resort to specialized hardware CPU, and or parallel processing techniques in order to improve performance. The swarm in quantum PSO is extremely sensitive to the size of the solution space. As the solution space expands (n goes larger) the swarm size should also be enlarged by relative factor in order for quantum PSO to remain computationally efficient.

In light of the arguments presented, we conclude that we have investigated all available avenues in scaling up quantum PSO within the framework of our investigation and considering the parameters we can maneuver with. To scale up our efforts to beyond $n>21$, we will have to investigate other research routes within the domain of evolutionary computing but certainly outside the realm of quantum PSO.

In Chapter 6, we will investigate a class of Genetic algorithm that we employ to implement a synchronistic algorithm with quantum PSO. This possibility come to the fore considering that the mutation distance between any instance of Γ_{p-1} and the target Γ_p (recalling (5-12) wherein $Q(x_p)=\Gamma_p$) is only 2. Accordingly, we may move from Γ_{p-1} to

Γ_p without the aid of quantum PSO by simply repeating the mutations between any two rows in Γ_{p-1} until the Γ_p is identified. In Chapter 6, we investigate if the Genetic algorithms may render help for any of the permutation matrices in the set $\bigcup_{i=0}^{p-1}\Gamma_i$ and not only the Γ_{p-1}.

5.5. Conclusion

In Chapter 3, a computational model for the equation to register isometry (Equation (3-5)) was formulated as the mathematical model by which isometry between two candidate topology may be established. In this chapter we explored three methods to transform the computational model for isometry registration (Equation (3-5)) to objective functions. The transformed objective functions were the operational domains of the quantum PSO engines in order to detect optimal solutions within the bounded constraints and error tolerance. The first two objective functions were based on Lagrange Multipliers (in section 5.3.1) and Quadratic Penalty methods (in section 5.3.3). The third objective function relied upon the lexicographically structured permutation ranking of the swarm and the solution space (in section 5.4). A thorough design and analysis of quantum PSO in optimizing a solution based upon the three different methodologies were covered. The results of the analysis suggested that quantum PSO might be utilized for isometry registration among topologies that are represented by fewer than 20 vertices on regular CPU machines. For topologies with larger than 20 vertices the particles in the swarm were slowed down frequently by the local extrema rendering unacceptable computational performance to locate the global target. To improve performance the swarm sizes would have to be increased relative to the enlargement of the solution space. Further, the computational platforms should be boasted up with CPU with faster clock speeds, and/or making use of parallel CPU cores.

Chapter 6 Genetic Algorithm and Quantum PSO: A Synchronistic Approach

In the previous chapters we outlined four characteristics of Scalability, Efficiency, Complexity, and Adaptability (SECA) for the optimization method to solve the equation to register isometry (Equation (3-5)) among deformable objects. In Chapter 5 we showed that the quantum PSO based optimization design may meet all the characteristics except for Scalability. We showed that quantum PSO is extremely sensitive to the size of solution space and the ratio of swarm size to the size of solution space should be taken into design consideration for topologies with $n > 20.$ We concluded that to account for this ratio in the design of quantum PSO, we may have to resort to more powerful hardware platform which would compromise the Efficiency objective in SECA guidelines. Hence, we argued to augment the quantum PSO with another mechanism from evolutionary computing that might mitigate the quantum PSO sensitivity to the size of the solution space.

In optimization of complex problems another class of approach known as Genetic Algorithms may render assistance in enabling an inherently parallel processing methodology towards the realization of an optimal solution (the parallel processing is also a feature in quantum PSO wherein a group of particles collaborate in parallel). Yet, the prominent feature of Genetic computing is the emulation of evolutionary strategy inspired by natural selection in biological systems. Genetic algorithms have been employed for search strategies to solve multi-dimensional, multi-objectives, and multi-constraints problems that are also frequently slow, complex, and extremely difficult to articulate and formulate using conventional and analytical techniques [11, 12].

The architecture of genetic algorithm constitutes of three components [12]: a population of entities that each may exhibit a potential solution, a genetic operator mechanism by which new generations are produced from the old, a fitness function which evaluates the health and rigor of the individual offspring (or the solution that each entity offers) within

the newly generated population. The seamless ensemble of the three components enables a cohesive architecture in emulating an evolutionary course towards the optimal solution. In this architecture, the genetic operator produces new generation of entities from the old. The individual members in the new generation are evaluated by the fitness function to identify the fittest (based on some criteria that is articulated in the function). The fittest entity selected by the fitness function becomes the progenitor of the next generation (or population). In biological realms this process may perpetually progress. However, in the optimization domain each and every generation is examined against some formulated criteria in order to evaluate and to measure the state of the evolution towards fulfilling and rendering the optimization objectives. Realizing the objective within the boundaries of the predicated error tolerance signifies the termination of the optimization effort.

6.1. Architecture of Genetic Algorithm

In this section, we explore specifically the components of the architecture of the Genetic Algorithm that we identified in the preceding section.

6.1.1. Population

The population constitutes the entities that individually carry some attributes and/or characteristics that are of interest and are the focus of the observation. In biological domains, for example, the population of interest may be comprised of a colony of bacteria. Some attributes of bacteria such as size or response to certain environmental effects may come under consideration in the members of the colony and/or across the generations. The population may be selected randomly from a larger one based on some criteria. However, the individual members of the population possess some attributes that are meticulously observed among the current and future populations.

In the optimization domain, the population refers to the aggregates of the instances of some class. The instances of the class may possess any attributes or states that change

reflecting the dynamism of their behaviors in response to not only some stimuli but also the diversification in the successive regenerations.

6.1.2. Genetic Operator

In producing a new generation, the genetic operator carries out the production of the new and the successive generations. The operator ensures some attributes are inherited from the parent generations; yet, the operator also ensures that a subset of attributes differ in the successive generations. Therefore, each individual member of the successive generations is diversified not only from their parents, but also from their siblings.

Two primary genetic operators are known [12]; namely, Mutation and Crossover. In Mutation individual entities may mutate from one type to another. Consider a genome and the individual chromosomes. In successive generations, for example, one nucleic acid type may mutate to another. In an optimization problem, however, the mutation is often modeled after binary representation of the attributes within the population, or the mutation of a '1' to a '0' and/or vice versa. There are various methods on how and which bit in a typical string to select in order to mutate. In Bit String mutation, individual bits are randomly selected. In Flip Bit mutation, the bits in the string are simply inverted. In Non-Uniform mutation, each bit is given a different probability value indicative of the chance of the bit being inverted. In Uniform mutation, each bit has the same probability to mutate across the generations.

In the Crossover operator, genetic material or attributes are taken from parents to recombine and to produce new genetic material for the new population. The prime example of this operator in biological system is the process of meiosis wherein homologous chromosomes exchange genetic material to form a recombinant baby chromosomes [51].

In both types of operators within the optimization domain, nevertheless, a pronounced stochastic process is embedded ensuing random behavior based on some probability distribution model is manifested.

6.1.3. Fitness Function

In each successive generation, a fitness process is required to evaluate the makeup of the new population against some criteria. In biological models the criteria may include adaptability to the environmental peculiarities. Certain environmental factors, such as temperature and acidity of the water may be the determining factor to accord fitness measure to the individuals in a population. A new population tested by the fitness process will be evaluated to select the fittest member of the new population. The genetic material or the attributes of the fittest are passed to the next generation.

In the optimization domain, the criteria for fitness are formulated according to the constraints and/or the objectives of the optimization problem. The fitness function's main roles include guidance towards the optimization objectives and observance of the constraints imposed. Further, the fitness function evaluates the state of the optimization at each generation to determine if the objectives have already been realized, whether the objectives are realizable in the future generations, or impossible to be realized.

6.2.Genetic Algorithm and Quantum PSO

Recall that in Chapter 5, quantum PSO algorithm was used in conjunction with three variant formulation of the objective functions. In the first two variants, quantum PSO operated upon two constraint based objective functions. The constraint based methods, Lagrange Multipliers and Quadratic Penalty, were employed to enforce the integrity of the intermediary resultant permutation matrices over the course of optimization processes. In the third variant, Lexicographic Permutation Ordering was used. All three variants had limited scalability for large surfaces due to the enormity of the sizes in the solution spaces. In all three variants, the swarm would be trapped eventually in the large number of local extrema preventing the swarm to pursue and to realize its optimization objective towards the global extremum. Yet, as described and analyzed in sections 5.4.3, 5.4.4, and 5.4.5 the Lexicographic Permutation Ordering produced the best results of the three

enabling the quantum PSO to realize its optimization objective for surfaces when $n \leq 20$, and with reasonable and acceptable computational performance.

In section 5.4.4.1 we formalized the notion of swarm trajectory $ST = \bigcup_{h=0}^{p} x_h = \{x_0, x_1, x_2, ..., x_{p-1}, x_p\}$ wherein x_p is the target objective or the global maximum. As analyzed in section 5.4.5, for $n > 20$, the swarm could not successfully reach its targeted objective, the global maximum at x_p, wherein $f(x_p) = n^2$. In fact for $n > 20$, the swarm traps at some local maximum $x_h \mid 0 < h < p-1$ such that $f(x_h) < n^2$, and from which the swarm may not escape in a computationally efficient manner.

In this section, and in light of the discussions on Genetic Algorithms, we propose that the swarm trapped at $x_h \mid 0 < h < p-1$ constitute a population that an instance of Genetic Algorithm may operate upon in order to release the swarm from the local maximum at x_h. The population may then continue traversing its trajectory to $x_{h+1}, x_{h+2}, ...x_{p-1}$, and eventually to the global maximum at x_p. The Equation (6-1) (which is a variation of Equation (5-13)) may serve as the fitness operator to evaluate the population and to identify the fittest particle within the swarm. Note that (6-1) is different from (5-13) by the input parameter ((6-1) takes a permutation matrix and (5-13) takes an integer). The fittest particle is determined by evaluating all particles of the swarm in Equation (6-1). The particle that produces the largest value is the fittest.

$$M(\Gamma) = Count[\Gamma^T Y_d \Gamma - X_d, \# \leq \varepsilon] \leq Dim[\Gamma]^2 \qquad (6\text{-}1)$$

Lastly, we propose a variation of Bit String mutation as the genetic operator in order to generate the future generations of the population. As an example, consider a swarm $SW = \{pl_1, pl_2, pl_3, ..., pl_{s-1}, pl_s\}$ that constitute of s particles. Let us assume that SW is trapped at x_h such that $f(x_h) < n^2$ as defined by Equation (5-13). Recalling the discussions on Particles with Permutation Ranking (section 5.4.1), we know that each

particle $pl_i \in \mathbb{Z}^+$. Yet, by virtue of Equation (5-12), we know that each particle in SW may also be represented by a unique permutation matrix. The swarm SW may therefore be transformed to a population of permutation matrices $PO_v = \{\Gamma_{v1}, \Gamma_{v2}, \Gamma_{v3}, ..., \Gamma_{vs}\}$ where v indicates the population's generation number, and $\forall pl_i \in SW \; \exists \; \Gamma_i \in PO_1$, and $PO_1 = \{\Gamma_{11}, \Gamma_{12}, \Gamma_{13}, ..., \Gamma_{1s}\}$ is the first generation of the population. Note that the SW and PO_v are of equal size s. Once the first generation PO_1 is known the genetic algorithm takes over of the PO_1. The first generation is evaluated by the fitness operator to identify the fittest member $\Gamma_{1i} \,|\, 1 \le i \le s$. The fittest is the one among all entities in PO_1 that once evaluated in Equation (6-1) produces the largest value. Once the fittest Γ_{1i} among the population is known, then the genetic operator takes over by randomly selecting and exchanging two rows from Γ_{1i} to produce the second generation of the population $PO_2 = \{\Gamma_{21}, \Gamma_{22}, \Gamma_{23}, ..., \Gamma_{2s}\}$. The second generation PO_2 is then evaluated by the fitness function to identify the fittest member $\Gamma_{2i} \,|_{1 \le i \le s}$. Like the case in the first generation, the fittest is the one among all entities in PO_2 that once evaluated in Equation (6-1) produces the largest value. Note that the Γ_{2i} may also represent x_{h+1} in the swarm trajectory. This means that once Γ_{2i} is identified, the population has escaped from the point where the swarm had been trapped (in quantum PSO before genetic algorithm takes over). It also means that $M(\Gamma_{2i}) > M(\Gamma_{1i})$ where M is defined by Equation (6-1). Once the fittest Γ_{2i} among the population PO_2 is known then the genetic operator takes over by randomly selecting and exchanging two rows from Γ_{2i} for s number of times to produce the third generation of the population $PO_3 = \{\Gamma_{31}, \Gamma_{32}, \Gamma_{33}, ..., \Gamma_{3s}\}$.

The process outlined above will continue for all successive generations PO_w where $w > 3$, and until the fitness operator identifies the fittest Γ_{wi} in PO_w such that

$M(\Gamma_{wi}) = Dim[\Gamma_{wi}]^2$ wherein Γ_{wi} corresponds to the permutation matrix providing the global maximum in M (Equation (6-1)), and the targeted optimization objective.

We propose to combine Quantum PSO with mutation based Genetic algorithm to produce Mutation Quantum PSO (MQPSO) as a synergistic and synchronistic algorithm to optimize solutions for $n > 20.$ In the experiments that we conducted MSPSO proved to be extremely fast and efficient. In all our experiments MQPSO could successfully match isometries on topologies represented by $10 \le n \le 1000$ vertices. Accordingly we present that MQPSO is highly scalable for wide varieties of topology sizes. In the section to follow we provide pseudo-codes and provide detailed experimental results.

6.3.Mutation Quantum PSO (MQPSO)

In this section we provide pseudo-code and detailed explanation about the MQPSO. As shown in Figure 6-1, the Quantum Mutation method accepts two surfaces that are represented by two geodesic distance matrices, and returns the permutation matrix that transforms one surface into the other. The two input parameters "surfaceX" and "surfaceY" must be of the same dimensions and globally accessible to all methods called hereon. The dimension of "surfaceX" may only be between 6 to 1000 as the algorithm was not tested for higher dimensions. The size of the solution space is therefore between 1 to Factorial[Dimension[surfaceX]]. Swarm size must be between 20 to 100. For large dimensions, the choice of the swarm size does NOT affect the performance of the algorithm.

Quantum Mutation

Input: surfaceX, surfaceY

Output: permutationMatrix

Pseudo-code:

```
dimension = Dimension[surfaceX];
swarmSize = an Integer between 20 to 100.
swarm = InitializeSwarm(swarmSize,dimension);
{bestGlobalValue, bestGlobalCoordinate} = QuantumSwarmOptimization(swarm);
If targetReached {
        permutationMatrix = The permutation matrix associated with bestGlobalCoordinate;
}
Else {
        permutationMatrix = MutationOptimization(bestGlobalCoordinate, bestGlobalValue);
}
return permutationMatrix
```

Figure 6-1 Quantum Mutation pseudo-code

This method first searches the solution space using quantum PSO to locate the targeted permutation matrix. If the solution space is small (dimensions smaller than 15) then the quantum PSO will most likely locate and find the targeted permutation matrix. For larger dimensions, however, if the quantum PSO fails (produces only a local maximum), then the best location value obtained by the quantum PSO is passed to the Mutation Optimization Method in order to take over the search for the targeted permutation matrix. It is imperative to execute quantum PSO before Mutation so that the latter starts its mutation loop on a coordinate which is as close as possible to the targeted permutation matrix. If quantum PSO is removed, then Mutation must randomly select a location in the solution space to start its mutation loop. Due to the vastness of the solution space, Mutation without quantum PSO will result in failure.

Initialize Swarm

Input: Swarm size, dimension

Output: Swarm

Pseudo-code:

```
swarm = Generate an array of swarmSize.
middle = Find the integer value associated to the middle of the solution space
    (Factorial[dimension]/2)
swarm = Initialize the particles in swarm to random integer values around the value of middle
    (±1,000.000, e.g.)
return swarm;
```

Figure 6-2 Initialize the swarm pseudo-code

The method Initialize Swarm (Figure 6-2) creates a swarm which constitute integer values between 1 to the size of the solution space. The swarm will subsequently be utilized in quantum PSO to search the solution space for the targeted permutation matrix. Later on, if quantum PSO is unsuccessful in locating the targeted permutation matrix, then the swarm will be passed on to the Mutation Based Genetic algorithm.

How to initialize the swarm with respect to the various locations in the solution space? This will be very important question whose answer will have considerable performance ramifications. Given the vastness of the solution space, and the fact that the solution space is not homogeneous (recall section from 5.4.5), then answering this question may be pursued as an independent study. However, results of our experiments have consistently shown that the swarm should not be initialized too dispersed across the solution space. The swarm whose individual particles remain in a cluster (or flock or swarm) have consistently demonstrated to be more effective than those swarms whose individual particles are dispersed too far away from one another.

Quantum Swarm Optimization

Input: Swarm

Output: Best Global Value, Best Global Coordinate

Pseudo-code:

```
{bestGlobalValue, bestGlobalCoordinate} = evaluate the swarm by the cost Function to establish an
        initial bestGlobalValue and an initial bestGlobalValue.
g = A small integer value;
Loop
      For k = 1, k <= particlesSize do
            particle[k] in Swarm = StandardQuantumUpdate;
            Evaluate particles[k] in cost Function and update bestGlobalValue and
                  bestGlobalCoordinate;
            If target is reached, then go to End Loop
      End For
      If trapped in local minimum, then go to End Loop
End Loop
return bestGlobalValue and bestGlobalCoordinate;
```

Figure 6-3 Quantum Swarm Optimization pseudo-code

The QuantumSwarmOptimization method (Figure 6-3) uses the standard Quantum Particle Swarm Optimization (Figure 4-2) in order to locate the permutation matrix by which surfaceX may be transformed to surfaceY (or vice versa). If it fails in its objective, then the best global value and the best global coordinate (refer back to section 4.2 for details of the significance of the two in Quantum PSO) are returned. The method will fail in its objective if and only if the swarm is trapped in a local minimum. In general, however, this may be determined by establishing a threshold level for the maximum number of iterations allowed before failing to update the best global value. If the threshold is reached, then the swarm is likely trapped for the foreseeable number of future iterations. Therefore, the method terminates and returns the best global value and the best global coordinate located so far.

Again, the choice of g has considerable impact on the success/failure of the method (recall sections 4.3.5, 5.4.4.1, and 5.4.4.2 for the discussion on this topic). Given the vastness of the solution space, the value of g must be small in order to ensure large update jumps across the solution space. Accordingly, a small g will ensure that the

solution space is surveyed as far as possible. In the experiments conducted g remained constant to an initially set value between $0 < g \leq 2$.

Large swarm sizes and a good value of g will enhance the success rates of the swarm. Nevertheless, for large size dimensions, the solution spaces are extremely vast. Doubling the size of the swarm in dimensions of 100 or above is similar to doubling the size of the swarm of birds, and then expecting the new swarm to be better equipped to locate a specific prey that may be located anywhere on the face of earth. Obviously success rates gained by increasing the size of the swarm cannot possibly counter-effect the performance degradation ensued by the swarm size enlargement.

Cost Function

Input: Particle

Output: Integer value representing the number of matches

Pseudo-code:

```
permMat = Find permutation matrix associated with particle;
matrix = permMat.surfaceX.Transpose[permMat] - surfaceY;
integerValue = Count all elements in  matrix whose values are smaller than the maximum value of
    noise;
return integerValue
```

Figure 6-4 Cost function pseudo-code

The Cost Function method (Figure 6-4) is a part of the fitness operator by which each particle (or entity) member of the swarm (or population) is evaluated in Quantum PSO and/or the Mutation based Genetic Algorithm method. This method is modeled after the objective function defined by Equation (6-1). A large value returned by this method is a definite indication that the particle represents a coordinate in the solution space that is closer than any previous location to the targeted permutation matrix. The largest possible value corresponds to the coordinate of the target. Accordingly, the largest possible value is Dimension[surfaceX]^2. The maximum value of allowed noise may be set globally. The choice of noise or ε is of paramount significance. This value indicates the maximum noise or distortion allowed between the two surfaces given that a perfect isometry is rare.

Mutation Optimization

Input: Best Global Coordinate, Best Global Value

Output: Permutation Matrix

Pseudo-code:

```
Loop
    new Coordinate = MutateCoordinate(bestGlobalCoordinate);
    newValue = Evaluate newCoordinate by the costFunction;
    If newValue > bestGlobalValue
            bestGlobalValue = newValue;
            bestGlobalCoordinate = newCoordinate;
            If targetReached
                    permutationMatrix = bestGlobalCoordinate;
                    goto EndLoop;
            EndIf
    EndIf
EndLoop
return permutationMatrix;
```

Figure 6-5 Mutation Optimization pseudo-code

This MutationOptimization method (Figure 6-5) mutates over the permutation matrix associated with bestGlobalCoordinate until it locates the targeted permutation matrix. This method is modeled after the instance of Genetic Algorithm discussed in sections 6.1 and 6.2. The target is reached if and only if the Cost Function (Figure 6-4) returns the largest possible value. See the comments for the method Cost Function for more details.

Mutate Coordinate

Input: Best Global Coordinate

Output: A New Mutation

Pseudo-code:

```
For all entities in population do
    twoRows = Select two rows of the permutation matrix associated with best global coordinate;
    Remember the twoRows so that they will not be reselected in the future mutations;
    aNewMutation = Exchange the twoRows of the permutation Matrix;
Evaluate the new population and assign the best value to aNewMutation
return aNewMutation;
```

Figure 6-6 Mutate Coordinate pseudo-code

The Mutate Coordinate method (Figure 6-6) mutates the permutation matrix associated with the bestGlobalCoordinate. The mutation takes place by selecting two rows and then

exchanging their locations. Duplicate selections are not allowed, e.g., $(1,4) \Leftrightarrow (4,1),\ (3,6) \Leftrightarrow (6,3),\ (4,8) \Leftrightarrow (8,4)$, and so on. There are various implementation ways to ensure duplications are avoided. For all selections of (i, j) then $i \neq j$. A selection (i, j) may not be repeated in future mutations. This means any mutation can only takes place at most once. Again, there are many ways to implement this feature.

6.4. Experimental Design and the Experimental Results

The design and runtime environment of the experiments were predominantly done in instances of Wolfram Mathematica 8. Certain experiments were also conducted in Microsoft Visual Studio C++ 10 in order to gauge performance comparisons between the two.

Mathematica is a computational platform designed and dedicated to symbolic, functional, and procedural programming with scientific and/or engineering applications. Mathematica provides extremely rich library of functional capabilities for overwhelming wide areas in mathematical and scientific computations.

All experiments targeted were programmed and conducted inside Mathematica, running over Mathematica's virtual machine. In doing so, the experiments were shielded from machine dependent considerations pertaining to numerical and floating point operations. The ability to run the experiments on the virtual machine enables the Mathematica programs to supersede the precision levels of the running hardware core. Further, given that the desired precision level of the floating point operations may be specified at programming time, independent of the hardware platform, the programs can be easily cross-ported and cross-run over different hardware platforms. This capability is extremely desirable in consideration for setting up parallel heterogeneous distributed farm data centers wherein Mathematica instances running on heterogeneous platforms may collectively cooperate over a network grid (and oblivious to any intermediary firewalls) to do parallel processing solving the same problem.

As stated previously the Mathematica core provides a wide range of functional assets pertinent to the diverse range of computational requirements. The core functional assets may also be augmented with additional libraries specific to certain vertical areas. Combinatorica [52] is a package enriching Mathematica programs with over 450 functions pertinent to the Combinatorial Mathematics. The experiments in the context of this thesis relied heavily on the wide range of assets provided by this package. In particular, functional services for matrix operations, list permutations and partitions, and/or graphs were extensively utilized.

We conducted two different sets of the experiments based on the type of input data. The first set was comprised of simulated data wherein two geodesic distance matrix of arbitrary dimension was produced using a Gamma Distribution and according to the pseudo-code illustrated in Figure 6-7.

Generate Two Distance Matrices with Simulated Entries

Input: an integer value between 8 to 1000

Output: Two Distance Matrices

Pseudo-code:

1. n **= an integer between 8 to 1000**
2. X **= Generate an** $n \times n$ **geodesic distance matrix using Gamma Distribution.**
3. P **= Generate a random permutation matrix of dimension** n **.**
4. $Y = PXP^T$
5. **Use** X and Y **as input to MQPSO to retrieve and to locate** P

Figure 6-7 Generate two distance matrices with simulated entries pseudo-code

In the second sets we made use of 3 dimensional shapes wherein the geodesic distance matrix was constructed by extracting an arbitrary number of coordinates and calculating the geodesic distances among all the extracted coordinates according the code in Figure 6-8.

Generate Two Distance Matrices with Real Data entries

Input: an integer value between 8 to 1000, 3D Shape

Output: Two Distance Matrices

Pseudo-code:

1. n = an integer between 8 to 1000
2. X = Generate an $n \times n$ geodesic distance matrix using n vertices extracted from 3D shape.
3. P = Generate a random permutation matrix of dimension n.
4. $Y = PXP^T$
5. Use X and Y as input to MQPSO to retrieve and to locate P

Figure 6-8 Generate two distance matrices with real data entries pseudo-code

The results presented in this section are produced by the simulation runs on single core machine with 3.4 GHz clock speed. No multiprocessing or parallel processing is performed in order to run the experiments in this section.

Table 6.1 illustrates the results for surfaces represented by vertices of dimension 25 to 1000. In all cases a geodesic distance matrix of corresponding dimension was generated by using a Gamma Distribution with $\alpha = 90$, $\beta = 5.5$. This geodesic distance matrix was then transformed by a randomly generated permutation matrix of the same dimension in order to generated the second surface (see Figure 6-7).

#Vertices	# Experiments	Success Rate	Avg Run Time (s)	Min Run Time(s)	Max Run Time (s)
25	30	100%	0.14	0.08	0.19
50	30	100%	0.67	0.51	0.78
100	30	100%	6.61	5.80	7.60
200	20	100%	93.35	79.36	102.10
300	10	100%	453.22	422.34	485.33
500	10	100%	3,797.36	3,702.87	3,822.22
1000	5	100%	43,568.65	43,234.70	43,822.67

Table 6.1 Results using simulated data for vertices sizes between 25 and 1000

The two surfaces were then given as input parameters to an instance of MQPSO. As shown in all cases MQPSO was able to successfully identify the permutation matrix by which the two surfaces were related.

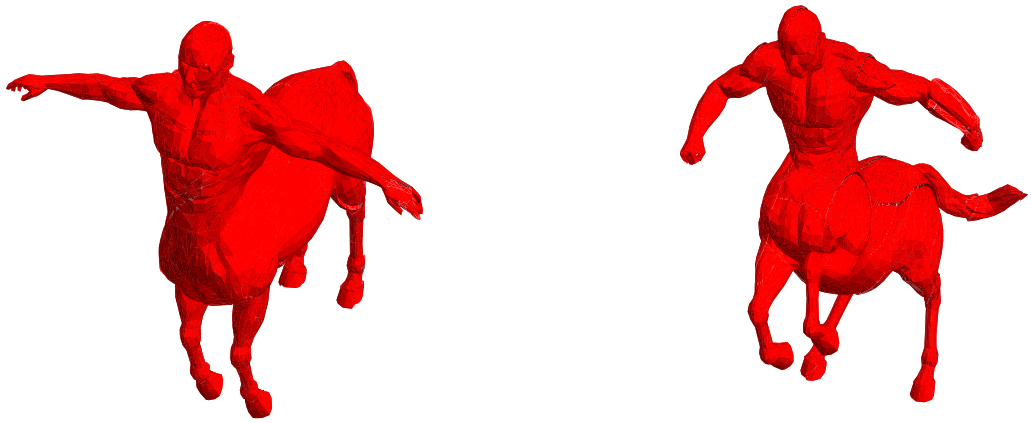


Figure 6-9 Three dimensional shapes used to extract coordinates

In our second set of experiments, we were motivated to make use of real data as opposed to simulated input data. Accordingly, we utilized the three-dimensional shapes shown in Figure 6-9 in order to extract random coordinates to construct distance matrices with the coordinates. The three dimensional objects used for the experiments were in OBJ Wavefront formats [53]. The OBJ file format is an open standard and has been adopted by many 3D graphics application vendors. It is one of the universally accepted file formats for the definition of the three dimensional geometrical objects.

The three dimensional shapes in Figure 6-9 are of two poses of the same Centaur. Each pose is expressed by 3600 vertices in Wavefront format. We extracted random vertices from the two and calculated the geodesic distances among the vertices obtained. We then made use of random permutation matrices to obtain the matrices corresponding to their transformed pairs. Each pair of the geodesic distance matrices was then given to MQPSO to locate the permutation matrix that would relate the pair.

#Vertices	# Experiments	Success Rate	Avg Run Time (s)	Min Run Time(s)	Max Run Time (s)
25	30	100%	0.13	0.07	0.21
50	30	100%	0.69	0.49	0.76
100	30	100%	6.61	5.60	7.80
200	20	100%	92.35	76.56	104.50
300	10	100%	443.51	412.64	497.11
500	10	100%	3,787.55	3,693.87	3,847.75

Table 6.2 Results from vertices of the 3D shape for sizes between 25 to 500

The results illustrated in Table 6.2 are nearly identical to those obtained for the simulated data (Table 6.1). In all cases MQPSO could successfully identify the permutation matrix that would relate the two input matrices. Since the two results are only different due to the element of randomness, then all subsequent experiments were done by means of real data.

Given the results obtained and illustrated in Table 6.1 and Table 6.2, we may conclude that MQPSO delivers far superior performance than those methods investigated in Chapter 5. In addition, MQPSO provides versatility and scalability for small and large size vertices alike.

Hitherto, the two input distance matrices investigated were related by only a permutation matrix. Nevertheless, we were motivated to test MQPSO tolerance for noise[1]. In the case where noise is involved, the two surfaces are not only different by a permutation matrix but also by some percentage of noise or distortion. Therefore, in the forthcoming results we subjected the two input surfaces to some percentage of noise or distortion. The amount of noise applied to a surface was based on a percentage of the standard deviation of the length of all edges (an edge was identified to be a direct link between any two vertices) in that surface.

[1] See section 2.3 discussing about distortion $\mathcal{E}$

Noise (σ)	# Expmnts	Success Rate	Avg Run Time (s)	Min Run Time(s)	Max Run Time(s)
0.25	30	100%	0.14	0.12	0.16
0.50	30	100%	0.17	0.12	0.75
0.75	30	100%	0.16	0.12	0.48
1.00	30	100%	0.17	0.12	0.48
2.00	30	100%	0.22	0.14	0.86
3.00	30	100%	0.20	0.14	1.12
4.00	30	97%	0.20	0.14	0.51
5.00	30	100%	0.27	0.14	1.53
6.00	30	90%	0.29	0.14	1.22

Table 6.3 Results obtained for 25 vertices

As illustrated in Table 6.3 when the two surfaces were represented by 25 vertices, the algorithm exhibited error tolerance for up to 6 standard deviation where the success rates began to decline. In all cases, nonetheless, the algorithm could identify the targeted permutation matrix in fraction of seconds.

Moving to 50 vertices in Table 6.4, it is evident that the MSPSO exhibits tolerance for noise for up to 3 standard deviations. However at error levels above 4 standard deviation, then the successes rates starts declining. As shown the average runtime over 50 experiments are impressive for 50 x 50 matrices, all experiments running in less than 2 seconds.

Noise (σ)	# Expmnts	Success Rate	Avg Run Time (s)
0.25	50	100%	0.76
0.50	50	100%	0.76
0.75	50	100%	0.84
1.00	50	100%	1.06
2.00	50	100%	1.12
3.00	50	100%	1.51
4.00	50	90%	1.38

Table 6.4 Results obtained for 50 vertices

In vertices of size 50, we begin to observe that, as the error level increases, the average run time also increases. This is due to the fact that the size of the solution space as well as the error levels introduced present more opportunities for the swarm to deviate into trajectories that are not destined for the optimal target. Therefore, the efforts required for reorientation and self correction translates to more iterations and/or mutations by the swarm. More mutations/iterations translate to longer computational times.

Vertices = 100				Vertices = 200			
Noise (σ)	**# Expmnts**	**Success Rate**	**Avg Run Time (s)**	**Noise (σ)**	**# Expmnts**	**Success Rate**	**Avg Run Time (s)**
0.25	50	100%	6.65	**0.25**	50	100%	99.55
0.50	50	100%	8.22	**0.50**	50	100%	96.25
0.75	50	100%	8.02	**0.75**	50	100%	134.97
1.00	50	100%	9.06	**1.00**	50	100%	160.14
2.00	50	100%	12.86	**2.00**	50	100%	226.53
3.00	50	100%	11.86	**3.00**	50	100%	323.17
4.00	50	96%	18.20	**4.00**	50	98%	373.24

Table 6.5 Results obtained for 100 & 200 vertices

Moving to vertices of size 100 & 200, MQPSO v1 sustains its impressive error tolerance. Yet, in vertices of size 200 we may observe the increase in average run times as error level enlarges. We may remind the reader that in comparing the size of the two solution spaces the larger (200!) one is $200!/100! = 2.4506 \times 10^{216}$ times larger than the smaller one (100!). Accordingly, the swarm must exert enormous amount of computational energy to rectify any deviations in the trajectory towards the targeted permutation matrix.

Table 6.6 illustrates the results for vertices of sizes 300 and 500. We may notice the remarkable loss of error tolerance in both cases. For vertices of size 300, the error tolerances start to dissipate at 2 standard deviation. Further the jump from 1 standard deviation to 2 standard deviation comes at the cost of over 100% increase in average run time. In case of vertices of size 500, the error tolerance may not be sustained for above 3% of the standard deviation.

Vertices = 300				Vertices = 500			
Noise (σ)	# Expmns	Success Rate	Avg Run Time (s)	Noise (σ)	# Expmns	Success Rate	Avg Run Time (s)
0.25	10	100%	433.83	0%	5	100%	3797.36
0.50	10	100%	471.78	1%	5	100%	3861.89
0.75	10	100%	640.10	3%	5	100%	3797.15
1.00	10	100%	947.60	5%	5	25%	5152.53
2.00	10	90%	1919.00				

Table 6.6 Results obtained for 300 & 500 vertices

In all experiments presented in this section, the swarm or population size was less than 100 particles.

# Vertices	Swarm or Population Size
V = 25	25
V= 50	30
V= 100	60
V= 200	80
V= 300	80
V >=500	100

Table 6.7 Vertices, and swarm size or population size

In the experiments conducted we used the swarm sizes set to those values shown in Table 6.7; nevertheless, the MQPSO performance proved to be insensitive to dramatic variations to the swarm sizes other than those used.

6.5.Conclusion

In this chapter we developed a MQPSO algorithm which implemented a symbiosis between quantum PSO and a Genetic Algorithm in order to register correspondence on deformable objects. The simulation experiments presented in this chapter demonstrated that MQPSO may deliver superior results meeting all the guidelines of Scalability, Efficiency, Complexity, and Adaptability. With regard to scalability, MQPSO demonstrated highly scalable performance for topologies represented by $n \leq 1000$ vertices. As far as efficiency, MQPSO delivered its results on a 3.4 GHz CPU. We may expect that on more powerful CPU equipped with parallel processing, MQPSO will render improved

performance beyond that already delivered. With respect to avoiding complexity, MQPSO simply requires two distance matrices as input parameters. The algorithm requires no further interaction from the user. Finally the algorithm is highly adaptable to the characteristics of the solution space. The algorithm can deliver results in spite of presence of noise or distortion in the solution space.

In the next chapter, we will use the MQPSO algorithm to register isometry among three dimensional deformable objects.

Chapter 7 Mutation Quantum PSO and Isometry for Three Dimensional Objects

In the this chapter, we set up experiments to apply the MQPSO algorithm to different three dimensional objects in order to observe the algorithm's performance in registering any isometry/correspondence on the candidate regions on the three dimensional objects.

In Chapter 6, we presented simulation results pertinent to the performance of our MQPSO algorithm. However, our motivation has always been to use the algorithm to register regions that are isometric or are in correspondence on three dimensional deformable objects. The three dimensional objects used for the experiments were in OBJ Wavefront formats [53]. Nevertheless, the MQPSO algorithm is neutral to the input file formats. The algorithm requires solely two geodesic distance matrices as their input parameters, and the algorithm is neutral with respect to how the distance matrices are produced. In all experiments conducted in this chapter, there was a preliminary phase involved during which two distance matrices were extracted from two OBJ files corresponding to two different three dimensional objects. In all cases the geometric shapes were expressed with at least 2400 vertices at full resolution in their corresponding OBJ files.

7.1. Isometric Sites on Three Dimensional Objects

As stated earlier, our objective is to verify whether our algorithm can identify isometric regions on two three-dimensional shapes. Therefore, in this section, we elaborate on the notion of "isometric regions on three dimensional objects"[1]. Consider the two poses of sea-horse in Figure 7-1. Each pose is expressed in 3400 vertices in OBJ format. Each pose, nevertheless, is a different articulation of the very same sea-horse in three dimensional space. Further, there are many isometric regions on the two pose. Two candidate isometric sites on the heads are highlighted in yellow.

[1] For formal definition of isometry see sections 2.2 and 2.5

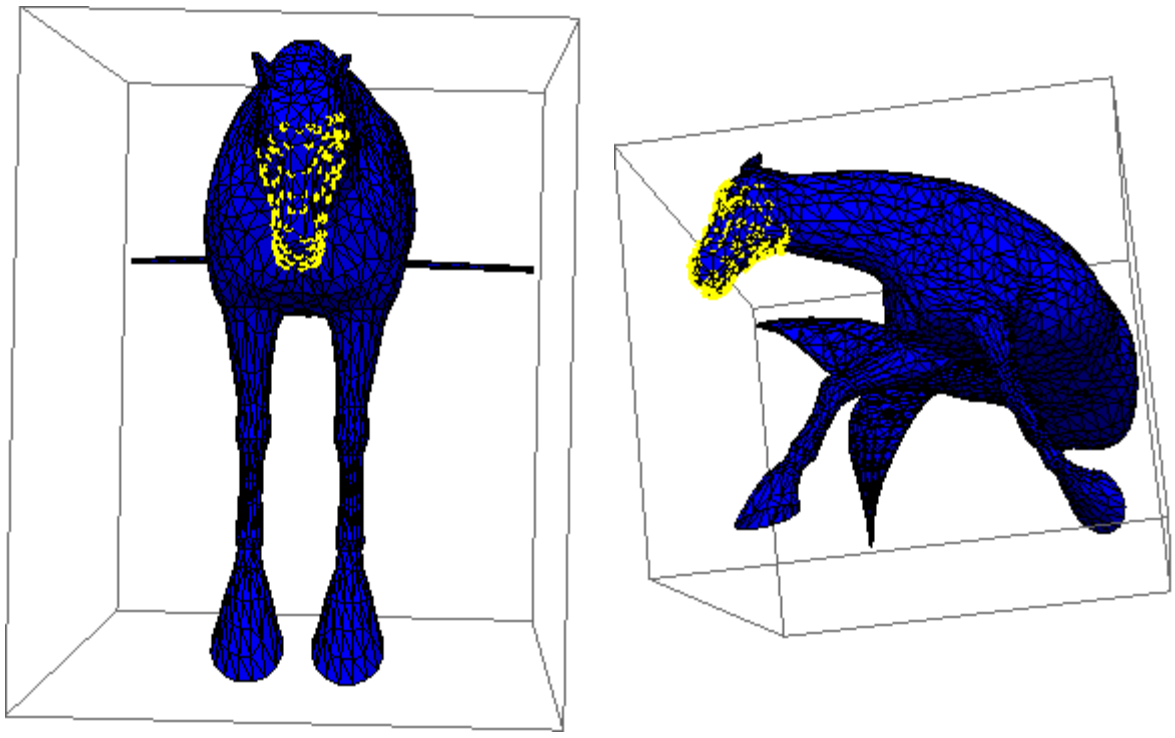


Figure 7-1 Sea-horse head poses

Figure 7-2 illustrates the very same pose, of the very same sea-horse, reoriented in the three dimensional space to highlight in yellow candidate regions for isometry on the corresponding tails. As illustrated in Figure 7-1 and Figure 7-2, one may find numerous candidate regions for isometries on the poses shown. In the two figures, however, only two regions on the heads and the tails are shown. In Figure 7-1, the areas highlighted in yellow constitute of 272 vertices on each pose. In Figure 7-2, the areas highlighted in yellow consists of 240 vertices on each pose.

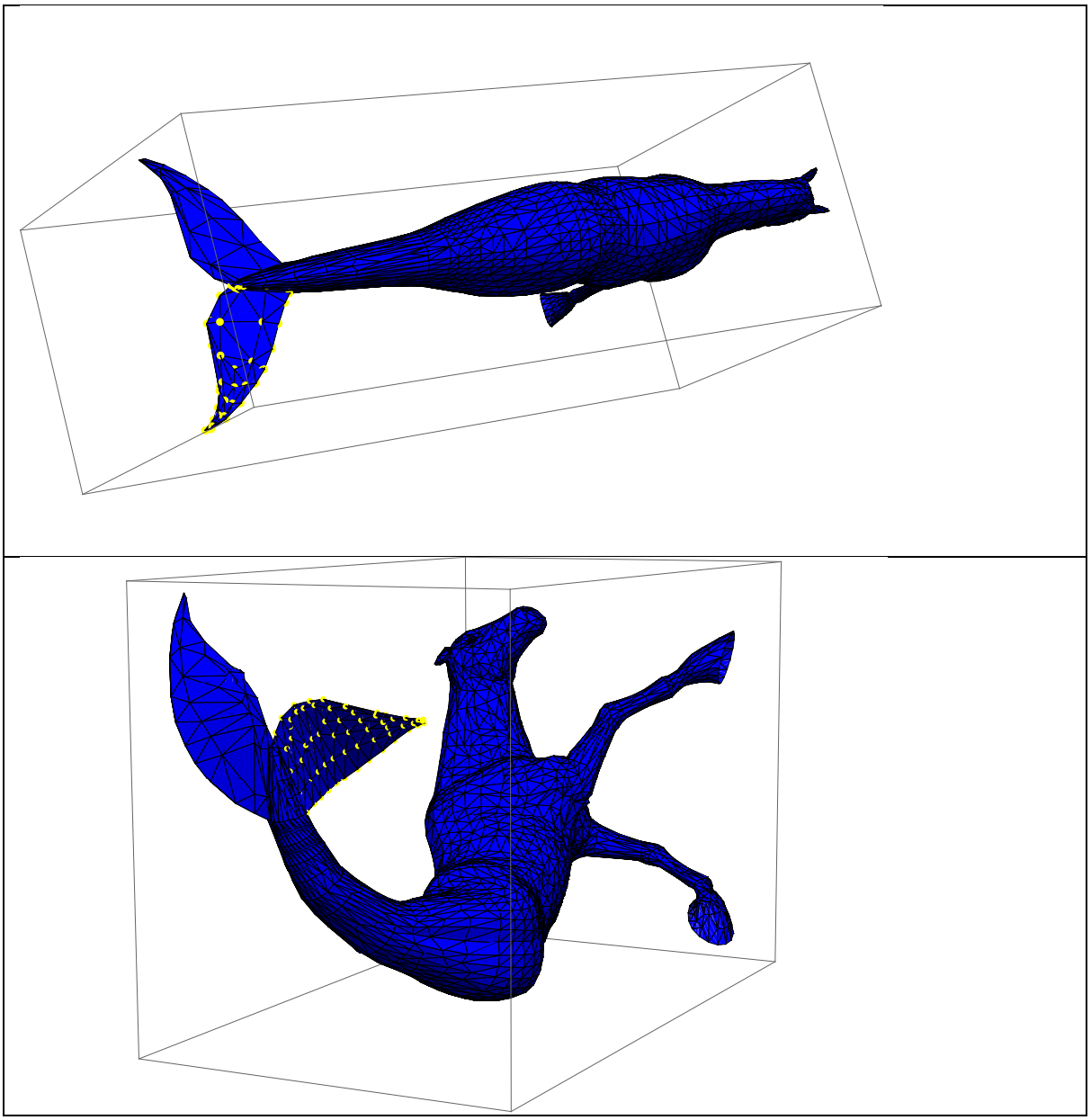


Figure 7-2 Sea-horse Tail Poses

Consider the head poses on the sea-horse in Figure 7-1. We may construct two distance matrices of sizes 272 x 272 of the two poses' heads. The two distance matrices may be given as input parameters to an instance of our MQPSO algorithm to verify if the two sites are isometric. The same procedure may be extended to the tails depicted in Figure 7-2. That is, two distance matrices of size 240 x 240 may be constructed as input parameters

to MQPSO. In each case, if the two sites are related by a permutation matrix, then MQPSO has successfully verified that the two regions are isometric.

In conclusion, in order to register the isometry on the candidate regions, a mechanism is required to select the data or vertices corresponding to the candidate regions. Once the data or the right vertices are selected, then the distance matrices among the selected vertices may be constructed in order to provide them as input parameters to MQPSO.

Before we elaborate on the data selection mechanism, we review some terms and terminologies pertinent to two-dimensional graphs theory. As we will shortly illustrate, the proposed data selection mechanism relies heavily on the two dimensional graph representation of the three dimensional objects.

7.2. Graph Representation

A three dimensional object may be represented by a two dimensional undirected graph $G = (V, E)$ such that V is the set of vertices or nodes, and E is the set of links, edges, or lines connecting the pairs of vertices in V. The vertices in V may be labeled by unique labels. For example, each vertex $v_i \in V$ may be labelled by a unique integer so that v_i is distinguished from every other vertex $v_j \in V \mid i \neq j$. Likewise, every single edge in E may assume a scalar quantity of some significance. For example, the scalar quantity may signify the weight or the distance between the corresponding two vertices that the edge connects.

Complete Graph: A complete graph is one in which every single pair of vertices in V is connected by an edge in E. Therefore, the number of edges in a complete graph of n vertices is $\binom{n}{2} = \frac{n(n-1)}{2}$ edges.

Connected Graph: A connected graph G is the one in which there is a path involving one or many edges from any vertex to every other vertex in G. This implies that a complete graph is always a connected graph; however, a connected graph is not necessarily a

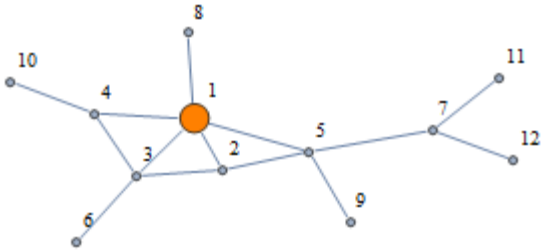
complete graph. A graph that is not connected is called a disconnected graph wherein there is at least a single pair $(v_i, v_j) \in V$ such that there is no known path from v_i to v_j, or the distance from v_i to v_j is infinite.

Consider the two poses of the sea-horse in Figure 7-1 and Figure 7-2. The first pose may be represented by a connected graph of 2194 vertices and 6505 edges, and the second pose may be represented by a connected graph of 2194 vertices and 6512 edges. Given the sheer size, the two graphs are not visually illustrated.

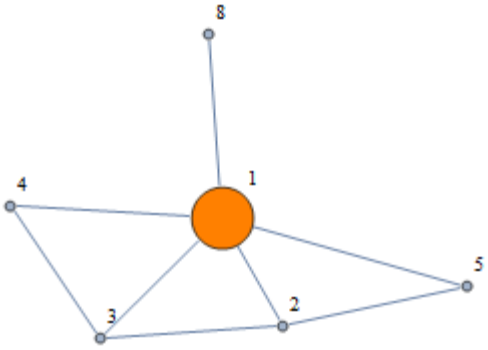
Neighborhood Graph: The neighborhood of a given graph $G = (V, E)$ from a vertex v is the subgraph $G_S = (V_S, E_S) \mid G_S \subset G$ induced by the neighborhood of vertex $v \mid v \in V_S,\ V_S \subset V$. There are many ways to construct a neighborhood graph around any given v. In the cases evaluated in this research, we made use of the integer value to indicate the distance or the depth from v to construct the neighborhood graph. Therefore, an integer depth of 1 indicated a subgraph $G_{S1} = (V_{S1}, E_{S1})$ including vertex v and its immediate vertex neighbors $V_{S1} \subset V$ such that $\forall v_j \in V_{S1}\ \exists (v, v_j) \in E_{S1}$. A depth greater than 1 would extend the subgraph by recursion to all immediate neighbours of the indicated depth. Figure 7-3 illustrates examples of neighborhood graphs of depth 1 and depth 2.

Closeness Centrality: The closeness centrality of a vertex $v_i \in V$ in a connected graph is the numerical value indicating how close the vertex v_i is to the center of graph G. To elaborate, let us assume a connected graph $G = (V, E)$ such that the size of V is n, i.e. there are n vertices in G. Then, the closeness centrality of v_i is the sum of all distances from v_i to every other vertex $v_j \in V \mid i \neq j$ divided by $n-1$. In this scheme, the vertex closest to the center of graph G receives the smallest value among all vertices in V. On the other hand, the largest closeness centrality value will be assumed by the vertex farthest away from the center of G.

Graph at Full Resolution



Neighborhood Graph of depth 1 centered at vertex #1



Neighborhood Graph of depth 2 centered at vertex #1

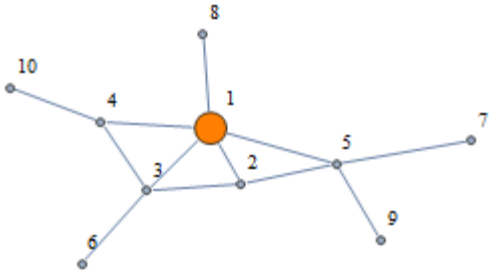


Figure 7-3 Neighborhood graphs of depths 1 & 2

Degree Centrality: The degree centrality of a vertex $v \in V$ is the size of the list of edges incident to v. Considering a typical connected graph, the vertices located on the perimeter have naturally lower degree centrality compared with those vertices that are located in the interior of the graph.

Perimeter Vertices: The perimeter vertices of the graph $G = (V, E)$ constitute the set of vertices $V_p \subseteq V$ such that V_p circumnavigate the graph G. The perimeter vertices of a graph G may be obtained by listing the Degree Centrality of all vertices in ascending order, and then selecting the first n vertices that encompass the entirety of the graph. Consider the graph that appears in Figure 7-4. Sorting the degree centrality of the vertices in ascending order will produce the list {{12,1}, {11,1}, {10,1}, {9,1}, {8,1}, {6,1}, {7,3},

{4,3}, {2,3}, {5,4}, {3,4}, {1,5}}. As shown, vertices 12, 11, 10, 9, 8, and 6 are on the top of the list with degree centrality of each being 1.

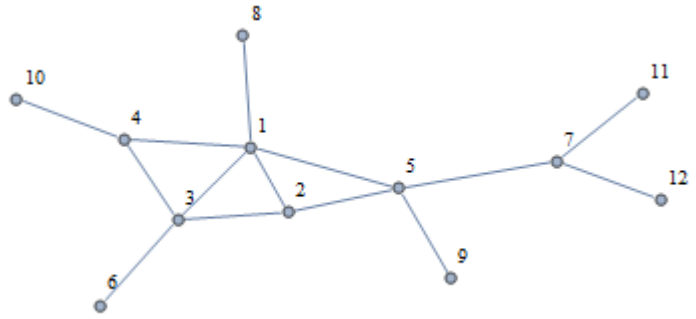


Figure 7-4 Graph with 12 vertices

As illustrated in Figure 7-5, the 6 vertices form the perimeter of the graph that encompass not only all other vertices but also the overall contour of the graph

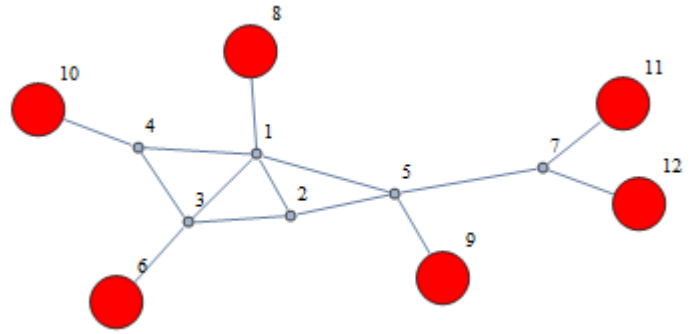


Figure 7-5 Perimeter vertices in a graph

Further, in the three dimensional space, these vertices form the volume of the space which encompasses all other vertices with higher degree centrality.

7.3. Vertices Encompassing Candidate Sites for Isometry

In section 7.1, we argued for a data selection mechanism by which vertices corresponding to the candidate regions for Isometry verification may be selected. Consider the two poses of the Centaur in Figure 7-6. Let us assume that the tails of the two poses are the candidate regions, and they are isomeric. Accordingly, a data selection mechanism is

required to select the vertices corresponding to the tails from the two poses. Once the vertices are known, then two distance matrices representing the geodesic distances among the vertices on each tail will be constructed. The two distance matrices are then given as input parameters to an instance of the MQPSO algorithm to verify if the two tails are isometric.

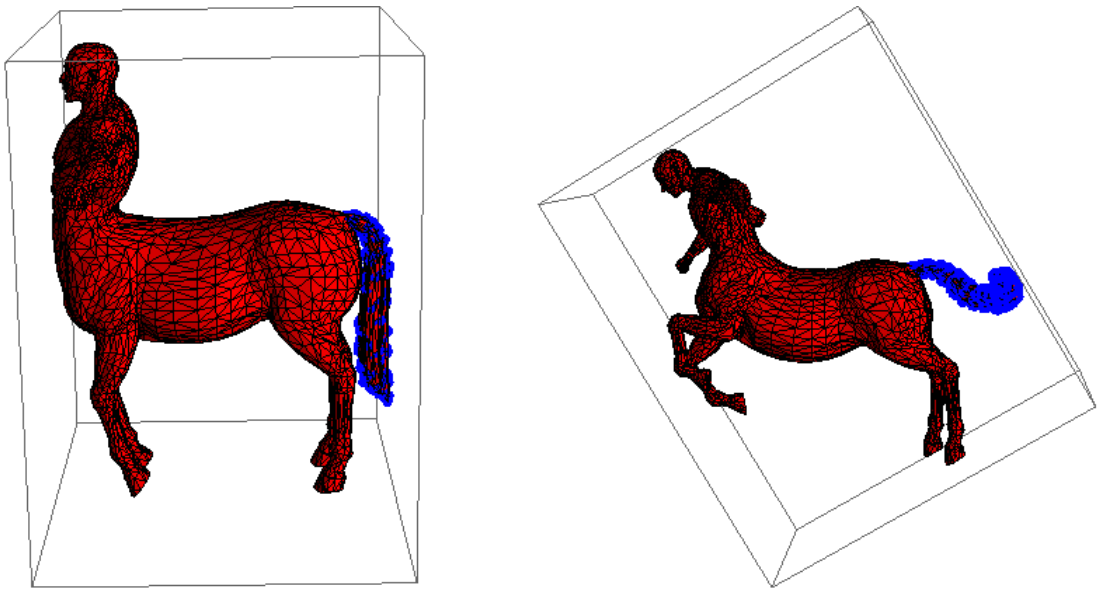


Figure 7-6 Isomeric Tails on Centaur

In light of the review on graph theory in section 7.2, each pose of the Centaur in Figure 7-6 may be expressed by a two-dimensional graph. Let us assume that pose 0 (on the top) is expressed by graph $G_0 = (V_0, E_0)$ and pose 1 (on the bottom) is expressed by $G_1 = (V_1, E_1)$. Then, the tail in pose 0 may be expressed by a neighborhood graph $G_{T0} = (V_{T0}, E_{T0}) \mid G_{T0} \subset G_0$. Likewise, the tail in pose 1 may be expressed by a neighborhood graph $G_{T1} = (V_{T1}, E_{T1}) \mid G_{T1} \subset G_1$. Once G_{T0} and G_{T1} are known, then their corresponding vertices V_{T0} and V_{T1} may be retrieved in order to construct the two geodesic distance matrices as input parameters to MQPSO.

7.4.Constructing Neighborhood Graphs

In this section, the procedure to construct the neighborhood graph of the regions of interest on a three dimensional object is outlined.

Constructing Neighborhood Graph of Tail

Input: Three Dimensional Shape in OBJ Format

Output: Neighborhood graph of tail

Step 1: Obtain the two dimensional graph of the Centaur at full resolution. Each vertex in the graph must be labelled by a unique integer.

Step 2: Inspect the graph from step 1. Identify the area in the graph that correspond to the tail.

Step 3: Select any vertex in the tail area by noting its label. Ideally this vertex should be as far away from the center of graph as possible.

Step 4: Construct a neighborhood graph of depth one or two around the vertex selected in step 3.

Step 5: List all the vertices of the graph obtained in Step 4. Calculate the closeness centrality of the vertices (see section 7.2).

Step 6: Identify the vertex with highest value of closeness centrality from step 5. This vertex is the tip of the tail, and is farthest away from the center of the graph.

Step 7: Construct a new neighborhood graph of the required depth around the tip. Start with small depth. Gradually increase the depth until the neighborhood graph encompasses the entirety of the tail and no more.

Figure 7-7 Steps in constructing neighborhood graph of tail

We have tailored the procedure to describe the steps for the neighborhood graph of the tail on Centaur (two poses of which are in Figure 7-6). The procedure may be extended to construct any neighborhood graph on any three dimensional object.

Figure 7-8 illustrates the neighborhood graph of the tail in pose 1 (bottom pose) of the Figure 7-6, constructed by means of the steps outlined in Figure 7-7. The tail is made of 281 vertices and 832 edges. The pose 1 of Centaur in Figure 7-6 at full resolution has 3600 vertices and 10194 edges. In our case, however, we are interested in the 281 vertices of the tail by which we may proceed to construct the 281 x 281 distance matrix. The distance matrix contains the geodesic distances among every two pairs of the vertices among the 281 vertices. Note that the neighborhood graph in Figure 7-8 is a connected graph in which every given vertex pair is connected via some path. However, the distance matrix contains the geodesic distances which amount to the shortest path between every pair.

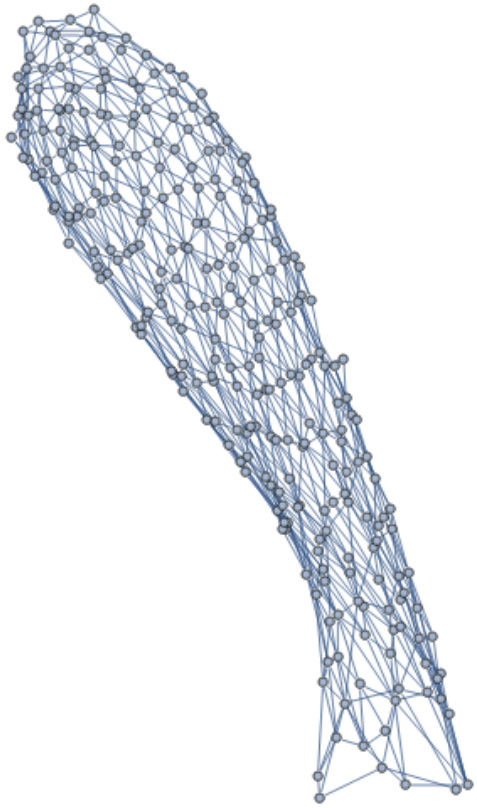


Figure 7-8 Neighborhood graph of tail

As stated earlier, the procedure outlined in Figure 7-7 may be utilized for any area on the three dimensional shape. Figure 7-9 illustrates the neighborhood graph of the head of the same Centaur pose. The head is constructed with a neighborhood graph which has 246 vertices and 714 edges. Note that the neighborhood graph in Figure 7-9 does not have homogeneous density of vertices. In particular, one side of the neighborhood graph has a higher concentration of vertices than the other. The higher density on one side corresponds to the articulation of eyes, ears, mouth, etc. on the head which amount to a higher density of vertices. Therefore, every member, e.g., eye, or ear, may be expressed by its own neighborhood graph, if warranted.

Figure 7-9 Neighborhood graph of head

7.5. Tessellation Effect on Neighborhood Graphs

In section 7.4, we outlined the steps to construct the neighborhood graph of a specific region on a three dimensional shape. A dilemma, however, arises when the neighborhood graphs are finally constructed. Due to the different tessellations, the two corresponding regions (for example, in Centaur, the tail in Pose 0 and the tail in Pose 1) are not necessary expressed in an equal number of vertices. Therefore, their corresponding neighborhood graphs are not of the same size (in the number of vertices). For example, in Pose 0 and Pose 1 the tails are represented by 211 and 281 vertices,

respectively. Likewise, the heads are represented by 246 and 242 vertices, respectively. Accordingly, the method by which we have constructed the neighborhood graphs will not produce distance matrices of equal dimensionality. We need a complementary method to construct two sub-graphs from the constructed two neighborhood graphs, that are not only of equal sizes, but also preserve the overall geometry of their respected regions. In this section, we describe the steps required in order to construct the sub-graph that preserve the geometry of its corresponding region.

Constructing sub-graph of the neighborhood graph

Input: Three dimensional shape in OBJ format

Output: Sub-graph of the neighborhood graph

Step 1: Construct the neighborhood graph as described in section 7.4.

Step 2: Obtain the perimeter vertices as described in section 7.2

Step 3: Construct a sub-graph with the vertices selected in **Step 2**.

Step 4: Check if the new sub-graph contains the entire geometry (or the perimeter) of the neighborhood graph obtained in **Step 1**. If not, then increment/decrement the number of the perimeter vertices in **Step 2**, and repeat steps 3 and 4.

Figure 7-10 Steps in constructing sub-graph which also preserves the geometry and contour of its neighborhood graph

A shown in Figure 7-10, steps 2 and 3 employ a mechanism that is biased against the centrally located vertices and favors vertices that are located in the extremities. Thus, by ensuring the inclusion of the vertices at the extremities (at the cost of the exclusion of the vertices with higher degrees that are naturally located interior to those vertices in the perimeters) the overall outline and the geometry of the neighborhood graph at full resolutions is preserved. The two subgraphs, however, should be constructed such that the mean of all the edges in the first subgraph is approximately equal to the mean of all the edges in the second subgraph.

In Figure 7-9, the neighborhood graph of the Centaur's head was shown at full resolution with 241 vertices. In Figure 7-11, we show the subgraph consisting of only 90 vertices in red that is superimposed on the neighborhood graph at full resolution. The subgraph is constructed using the procedure outlined in this section (Figure 7-10). As shown, even with 90 vertices the overall geometry and contour of the head is approximately preserved.

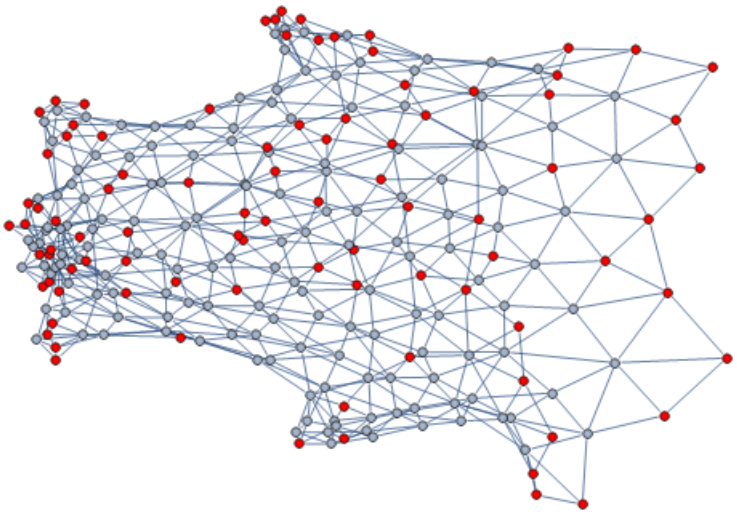


Figure 7-11 Head sub-graph with 90 vertices selected in red

Figure 7-12 illustrates the subgraph with 170 vertices in red (still 71 vertices short of the neighborhood graph at full resolution) superimposed on the neighborhood graph at full resolution.

The procedure outlined in this section may be utilized to select arbitrary number of vertices from the neighborhood graph. As such, this method may be used to select equal number of vertices from the two candidate neighborhood graphs in order to construct the two distance matrices of equal dimensionality.

The procedures described in sections 7.4 and 7.5 enabled us a data selection mechanism by which we constructed distance matrices as input parameters for our MQPSO algorithm. In the next section experimental results will be presented on registering isometry on candidate regions of various three dimensional deformable objects.

Figure 7-12 Head sub-graph with 170 vertices selected in red

7.6. Experimental Results

In this section we describe the results obtained when using our MQPSO algorithm on different three dimensional deformable shapes. Note that, in all experiments conducted we utilized the methodology described in section 7.5 to construct the two distance matrices representing the topologies of the candidate regions.

The two candidate regions were extracted from two poses of the same three dimensional object. Each pose at full resolution, however, was expressed using different tessellations. As such, the triangulated meshes that captured the local smoothness of each pose had different granularities. As discussed earlier, for example, the tails on the two poses of Centaur were expressed in different number of vertices, edges, and consequently triangles. In registering the isometry between two regions represented by the two distance matrices, we need to consider the intrinsic noise or error[1]. In other words, the MQPSO algorithm must account for the intrinsic noise pertinent to the underlying triangulated meshes. In view of the intrinsic noise, the MQPSO algorithm searched for the

[1] See section 2.5 on distortion/noise in isometry

permutation matrix to relate the two regions given a measure of error tolerance. As such, in all experiments, the σ or error tolerance was defined as $\sigma = Max[\sigma_0, \sigma_1]$ wherein σ_0 was the standard deviation of the lengths of all the edges in neighborhood graph from the first pose, and σ_1 was the standard deviation of the lengths of all edges in the neighborhood graph from the second pose. Therefore, in the following experiments the objective was to identify a permutation matrix that may relate the two distance matrices given the error tolerance of σ or some percentage of σ. If the two distance matrices could be related by a permutation matrix, given the error tolerance, then their associated topologies were concluded to be isometric[1].

The non-rigid deformable three dimensional objects we used for our experiments included Centaur, Gorilla, and Sea-horse. Candidate regions to register for isometry were Centaur Tail (Figure 7-6), Gorilla Right Hand (Figure 7-13), Centaur Left Leg (Figure 7-14), Sea-horse Tail (Figure 7-15), and Centaur Head (Figure 7-16). Table 7.1 shows the results obtained for the all experiments conducted.

3D Shape	Candidate Region	Figures	Vertices	Number of experiments	Success Rate	Error Tolerance	Avg Time (s)
Centaur	Tails	Figure 7-6	65	10	100%	2σ	6.17
			100	10	100%	1.5σ	43.16
			150	10	100%	1.25σ	643.56
Gorilla	Right Hands	Figure 7-13	100	10	100%	1.35σ	48.12
			120	10	100%	1.30σ	113.56
Centaur	Left Legs	Figure 7-14	56	10	100%	1.50σ	5.27
			120	10	100%	1.14σ	285.97
Sea-horse	Tails	Figure 7-15	220	10	100%	1.75σ	1698.21
Centaur	Heads	Figure 7-16	242	10	100%	1.50σ	3938.53

Table 7.1 Results of the experiments for isometry registration

[1] We re-emphasize that the noise is intrinsic due to the distortion and/or variations of tessellations of the two poses. Therefore, the cost function must account for the intrinsic noise by raising its error/noise tolerance. In here, we did not add any noise to the distance matrices.

An experiment consisted of running the MQPSO algorithm with two distance matrices representing two candidate regions on two poses of a three dimensional shape. Each experiment was repeated 10 times with different global seed values. The significance of differentiating the global seed was to force MQPSO to commence its search from a different location within the solution space in each experiment.

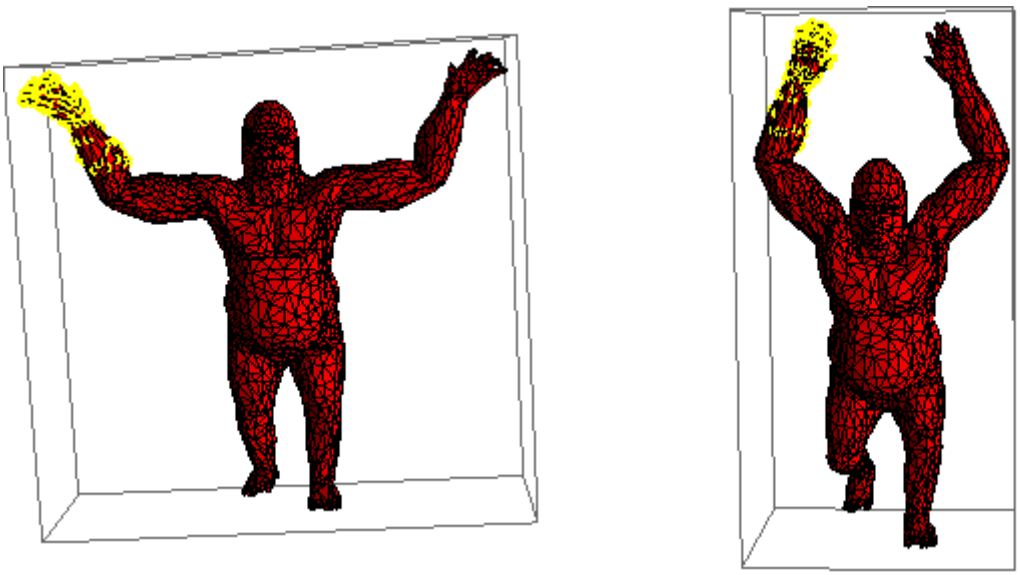


Figure 7-13 Gorilla right hand isometry registration

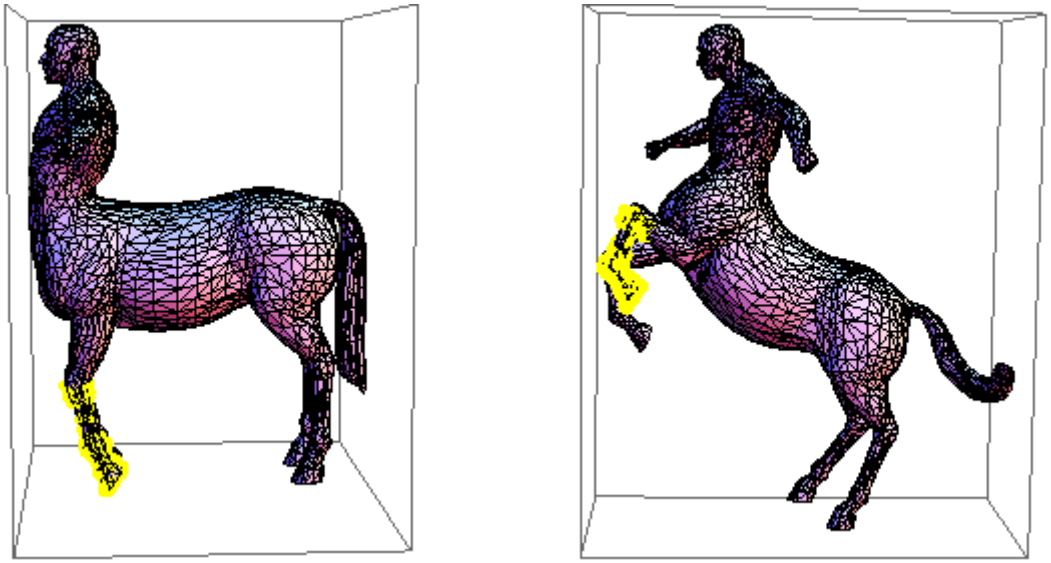


Figure 7-14 Centaur Left Leg isometry registration

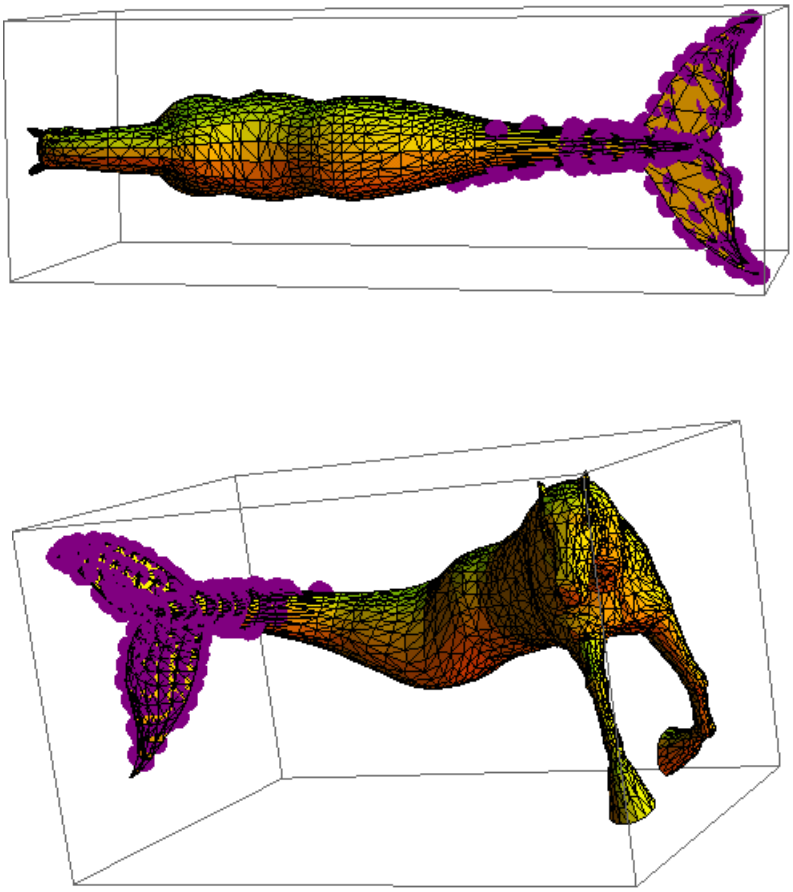


Figure 7-15 Seahorse Tail isometry registration

The success rate indicates the percentage of success out of the 10 experiments that the MQPSO algorithm could identify a permutation matrix that may relate the two input distance matrices, and register isometry between the two respected topologies. As the results in Table 7.1 show, the MQPSO algorithm was able to successfully identify a permutation matrix , and to register the isometry between the corresponding shapes, in all experiments conducted. The results indicate that by increasing the number of vertices, the error tolerance (or intrinsic distortion) was accordingly reduced at the cost of increasing the computational time to convergence.

In the course of experiments conducted, it was observed that in some circumstances, when the error tolerance was above 1.35σ, the two corresponding regions may be related by more than a permutation matrix. In fact, in those cases, no amount of fine

tuning on the degree of error tolerance could result in the production of a single permutation matrix that would relate the two topologies. Nevertheless, these permutation matrices differed from one another by one or two rows/columns. It was also observed that increasing the number of vertices could not reduce the distortion/error tolerance below those levels indicated in Table 7.1.

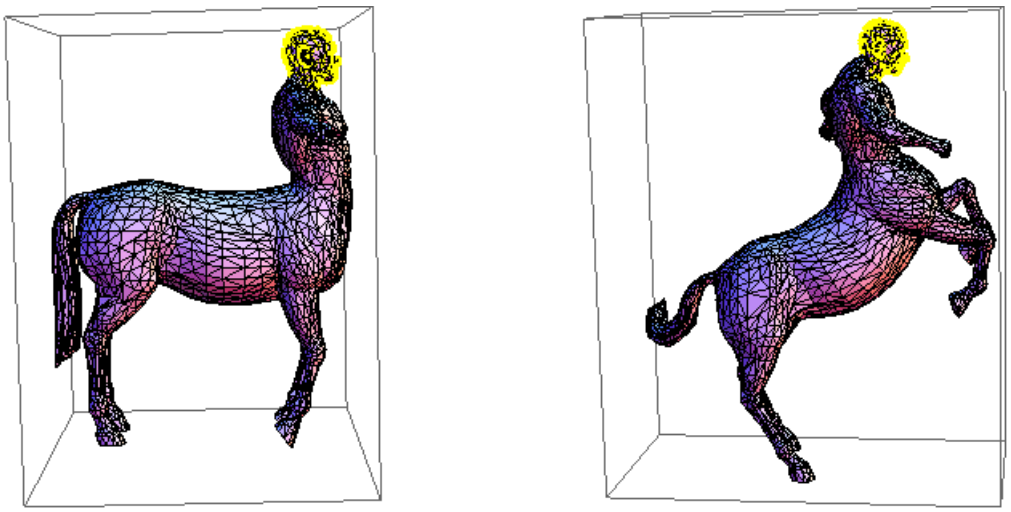


Figure 7-16 Centaur head isometry registration

To better understand the significance of intrinsic noise/distortion, let us recall the pertinent discussions from Chapter 2. A geodesic distance between any two points (x, x') is the length of the shortest path connecting the two.

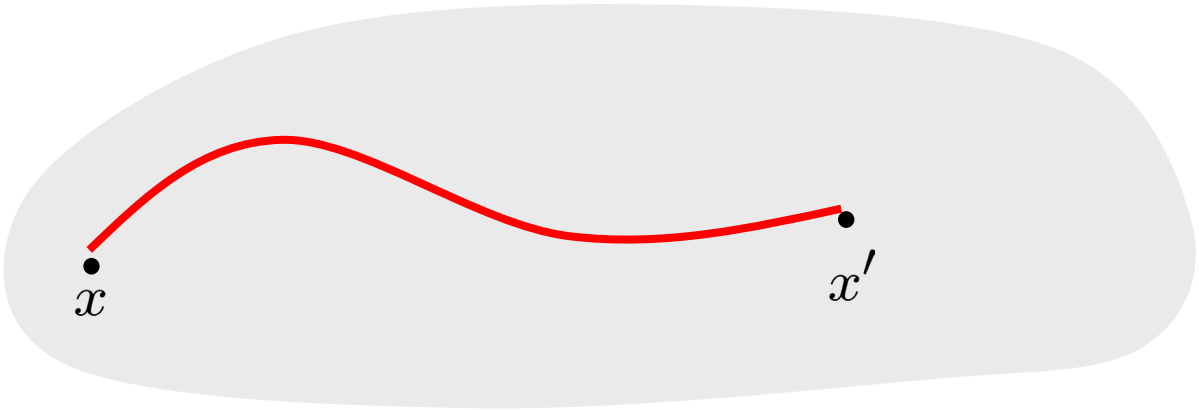


Figure 7-17 Geodesic distance is the shortest curve between two vertices

In the case of the three dimensional shapes used in our experiments, they were constructed with triangular meshes. The geodesic distances between every two vertices were approximated by the sum of lengths of the line segments that constituted the shortest path. Accordingly, every geodesic distance was constructed as $\sum_{i=1}^{N-1} d_{\mathbb{R}^3}(x_i, x_{i+1})$ where $N-1$ is the number of line segments between x_1 and x_N.

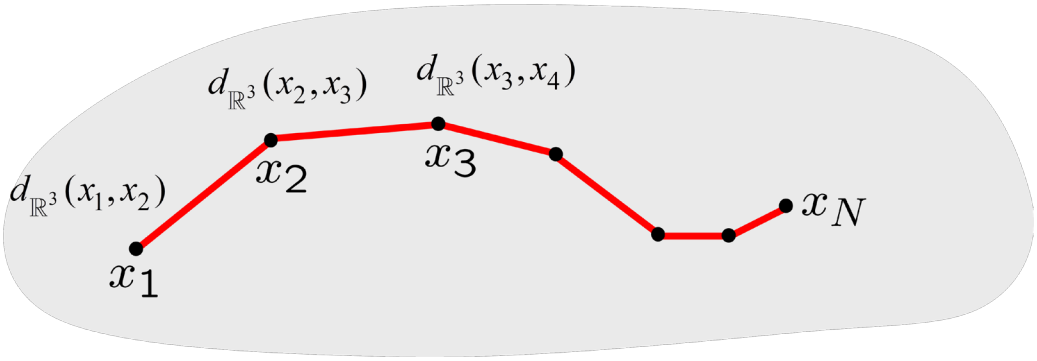


Figure 7-18 Geodesic distance is induced by sum of Euclidean distances

Every line segment, however, is measured by Euclidean distance. In other word, the geodesic distance between every two vertices is an approximation induced by the summation of the Euclidean distances of the intermediary line segments.

In ideal situations we may have perfect isometries among the two shapes with no distortion. In Figure 7-19, we have a case of distance preserving map or isometry between the two metric spaces of (X, d) and (Y, δ).

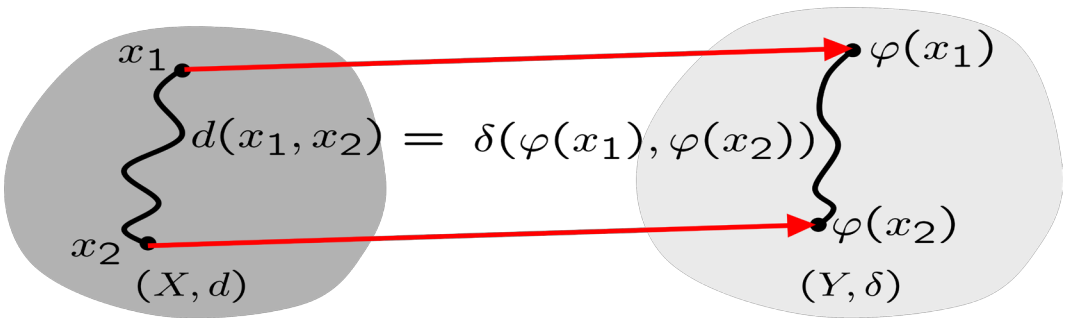


Figure 7-19 Distance preserving map or isometry [2]

In most situations, however, due to the tessellations effects and induced geodesic distances, the perfect isometry is compromised to near isometry or ε – isometry.

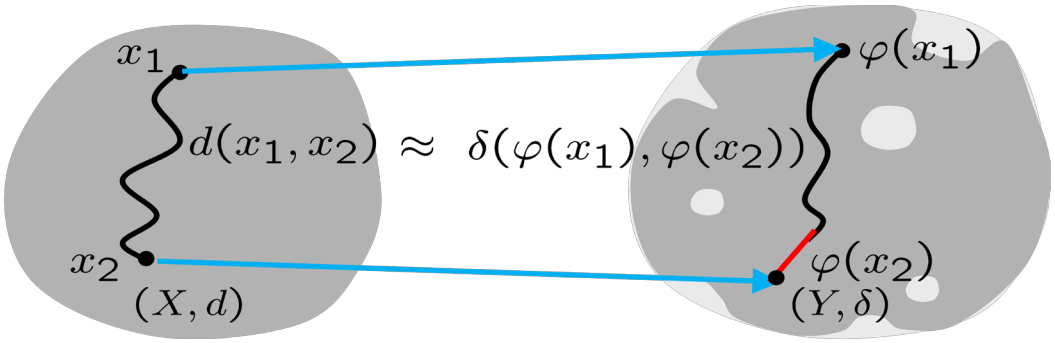


Figure 7-20 Almost isometry [2]

In the case of the three dimensional shapes used in this thesis , to counter ε or to reduce the error tolerance below those levels in Table 7.1, we may have to increase the size of triangulations representing the shapes. This will ensure the shapes become smoother and will consequently reduce the amount of intrinsic distortions between the two geometric spaces.

7.7.Summary

In this chapter, we provided experimental results of applying the MQPSO algorithm to different three dimensional deformable objects. The results provided evidence that the MQPSO algorithm can successfully register isometries on various sites on a wide range of deformable three dimensional geometries.

We also provided a data selection mechanism to construct two dimensional graphs, neighborhood graphs, and sub-graphs not only to build input parameters for the MQPSO algorithm but also to account for irregularities and noise in the underlying three dimensional deformable geometries.

Chapter 8 Conclusion

In this chapter we present a brief discussion and identify the contributions made by this thesis. We also provide some other application areas that may take advantage of this research effort.

8.1. Discussion

The world around us is populated with non-rigid deformable objects. Deformable objects are elastic, flexible, and amorphous. The quantitative representation and measurement by which the deformable objects are studied and analysed are of special significance in order to derive pertinent semantic information. Examples of semantic information may include similarity, recognition or identification, and/or distance between the deformable objects. Consider, for example, the intelligent act of "recognition" by the human mind. The act involves an intellectual apprehension of an object by an individual's intelligence. The dynamics of the act by human intelligence has baffled neurologists and philosophers alike (the latter since antiquity to present day) [54, 55].

In modern physics[1], wherein the paradigm of quantity reigns [56], any object must have a mathematical representation in order to be identified and/or comprehended. For example, a billiard ball may be represented by (d,δ) where d is the diameter and δ is the density. Two billiard balls may be deemed to be alike if $|d_1 - d_2| + |\delta_1 - \delta_2| \leq \varepsilon$. Likewise, the act of recognition, for example by a computer, is accomplished when two mathematical representations (corresponding to two physical objects) are deemed to be similar. Thus, the semantic notation of similarity and/or recognition is dependent upon a quantitative representation of the distance between the two physical objects. If the distance is small, then the two physical objects are similar. In the domain of non-rigid

[1] The metaphysical premise upon which the modern physics is based was founded by Rene Descartes when he divided the external world to *res extensaes* (extended entities) and *res cogitantes* (thinking entities). Physics has since been concerned only with quantification of the "extended entities". In this dualistic perspective, upon which modern epistemology rests, all else that are not quantifiable are thinking entities and have no ontological existence outside of one's mind [77].

deformable objects the quantitative representation assumes more complexity. The non-rigid objects may deform and assume new extrinsic geometries. The representation must reflect the invariant intrinsic geometry of the object that does not undergo a change. Further, the quantitative measurement of the distance between the two deformable objects must account for isometry-invariant-distance and/or isometry-invariant similarity.

This thesis focused on the realization of two objectives. The first objective was to review the mathematical models for the analysis and representations of the non-rigid deformable objects. The second objective was to develop optimization engines in order to register isometry among deformable objects. In summary, the four contributions of this thesis may be highlighted as:

Comparative analysis of Quantum PSO vs. Classical PSO: In this study the two variants of the particle swarm optimization were studied and compared. We aimed to identify the variant that was scalable across solution spaces whose sizes extended from small to extremely large. The topologies of the deformable objects may be expressed at wide range of resolutions which consequently determines the sizes of the solution spaces. Accordingly, the scalability of the optimization engine to effectively navigate all sizes of solution spaces was of paramount importance. Further, we looked for the PSO variant that might guarantee that the optimization engine focused its search energy in a bounded region of the solution space wherein a solution was located. Our study in this effort provided evidence that Quantum PSO was superior to Classical PSO in fulfilling both objectives.

Quantum Particle Swarm Optimization and Isometry Registration: In this study, we designed and developed Quantum PSO engines to register isometries between two deformable objects. We developed constraint based objective functions using Lagrange Multipliers and the Quadratic Penalty Method. We conducted experiments and provided analysis that supported the claim that the two constraint based methods were applicable only to the solution spaces that were comparatively small. The quantum PSO with constraint based methods was not scalable to register

isometry between deformable objects whose topologies were represented by more than 15 vertices. Given the insights gained, we developed a novel scheme to structure the solution space with Lexicographic ordering. A new version of quantum PSO was developed in which the particles were represented by integer values that denoted their ranks and/or their relative locations in the solution space. The newly developed quantum PSO with lexicographic ordering enabled far superior performance in terms of computational time to locate and to register isometries. However, experimental results and our analysis provided evidence that the quantum PSO with lexicographic ranking might not scale up efficiently for topologies represented by very large number of vertices without resorting to multi-cores/parallel processing platforms.

Quantum PSO and Genetic Algorithm: In this effort, we designed and constructed a synchronistic algorithm between Quantum PSO and a Mutation based Genetic Algorithm. In this novel approach we were able to achieve our objective of developing an optimization engine that met the requirements; namely, scalability, efficiency, non-complexity, and adaptability. We provided simulation results by which we showed the new algorithm could register isometries between topologies comprising from 10 to 1000 vertices. The simulation results provided evidence that the algorithm was robust among topologies with a wide range of distortions and noises.

Isometry Registration between Deformable Objects: In this effort we showed that Quantum PSO with Mutation based Genetic Algorithm could successfully register and identify isometries among non-rigid deformable objects. We provided experimental results demonstrating the effectiveness of the algorithm in registering isometries among five different deformable objects. We also developed a data selection mechanism based on graph theory by which various regions on any three dimensional object could be selected as candidate sites, and be used as input parameters to our algorithm.

8.2. Future Work

In this section we identify other areas that future research may provide additional insights.

Topologies Larger than 1000 Vertices: The range of experiments we conducted in Chapter 6 were limited to those topologies represented by fewer than 1000 vertices, or when $n \leq 1000$. In Table 6.1, we presented results for 5 experiments in which for $n = 1000$, MQPSO was able to successfully detect the correspondence in just over 12 hours of computation time.

Let us more closely visualize the size of solution space when $n = 1000$. Figure 8-1 depicts the integer value of 1000!. Now compare this value with the integer value 1,135,296,000,000,000,000,000 which is the diameter of the Milky Way galaxy in meters [57]. As shown, the integer value for 1000! is the evidence to the enormity of the solution space when $n = 1000$. Given the abundance of the trajectories in this space, it should become more illustrative of the efforts MQPSO must exert to locate and to identify its very single target when $n \geq 1000$. Therefore, we may conclude that for $n \geq 1000$, MQPSO may not locate its target in acceptable computational time. The second reason which exacerbates the performance is the number of digits in the integer values. For example, when $n = 1000$, then $1000!$ constitutes of 2568 digits, when $n = 1500$ then 1500! constitutes of 4115 digits, or when $n = 2000$ then 2000! is comprised of 5736 digits. These integer values are far larger than maximum size of the atomic types by any hardware platform. Processing such large values will have to be accomplished either by the virtual machines or at the programming layers. In either case, performance degrades as n gets larger. In light of this dilemma, we conclude that for $n > 1000$, the quantum PSO based upon Permutation Ranking may be approaching its ultimate limit in rendering acceptable performance. We need to remind ourselves, however, that for $n > 20$ the role expected of quantum PSO in MQPSO is not to locate its target, but to push the swarm into the solution space and thus render a best global position as close as possible to the global target. As analyzed in Chapter 5, quantum PSO will eventually get trapped in a local

maximum where Mutation Algorithm will take over to pursue the search for a global maximum. Therefore, it is our conclusion that for $n > 1000$, an alternative approach to Permutation Ranking should be sought to push the swarm into the solution space before Mutation Algorithm takes over.

40238726007709377354370243392300398571937486421071463254379991042993851239862902059204
42084869694048004799886101971960586316668729948085589013238296699445909974245040870737
59918823627727188732519779505950995276120874975462497043601418278094646496291056393887
43788648733711918104582578364784997701247663288983595573543251318532395846307555740911
42624174743493475534286465766116677973966688202912073791438537195882498081268678383745
59731746136085379534524221586593201928090878297308431392844403281231558611036976801357
30421616874760967587134831202547858932076716913244842623613141250878020800026168315102
73418279777047846358681701643650241536913982812648102130927612448963599287051149649754
19909342221566832572080821333186116811553615836546984046708975602900950537616475847728
42188967964624494516076535340819890138544248798495995331910172335555660213945039973628
07501378376153071277619268490343526252000158885351473316117021039681759215109077880193
93178114194545257223865541461062892187960223838971476088506276862967146674697562911234
08243920816015378088989396451826324367161676217916890977991190375403127462228998800519
54444142820121873617459926429565817466283029555702990243241531816172104658320367869061
17260158783520751516284225540265170483304226143974286933061690897968482590125458327168
22645806652676995865268227280707578139185817888965220816434834482599326604336766017699
96128318607883861502794659551311565520360939881806121385586003014356945272242063446317
97460594682573103790084024432438465657245014402821885252470935190620929023136493273497
56551395872055965422874977401141334696271542284586237738753823048386568897646192738381
49001407673104466402598994902222217659043399018860185665264850617997023561938970178600
40811889729918311021171229845901641921068884387121855646124960798722908519296819372388
64261483965738229112312502418664935314397013742853192664987533721894069428143411852015
80141233448280150513996942901534830776445690990731524332782882698646027898643211390835
06217095002597389863554277196742822248757586765752344220207573630569498825087968928162
75384886339690995982628095612145099487170124451646126037902930912088908694202851064018
21543994571568059418727489980942547421735824010636774045957417851608292301353580818400
96996372524230560855903700624271243416909004153690105933983835777939410970027753472000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000

Figure 8-1 Number of digits in 1000! There are 2568 digits !

Yet, for $n \geq 1000$ the single most prominent area to deliver improved and more acceptable performance remains to be the Mutation Algorithm. In MQPSO, the genetic operator (recall from sections 6.1.2 and 6.1.3) mutates by randomly selecting two rows of (the permutation matrix that represents) the fittest entity in the population to mutate and thus to proginate a future generation. In other words, the new populations are produced at the speed of two row mutations per generation. The improvement in

performance of MQPSO relies upon developing a methodology to accelerate mutation speed from two row mutation at a time to three, four, or more mutations per population. We conducted a limited experiments in this regard and the preliminary results indicated a far superior performance than those obtained and presented. However, our experiments suggested that random mutations may not provide sustained improvements over repeated experiments with different seed values. A more evolved algorithm than random mutation is warranted to accelerate the mutation speed above two rows at a time. The formulation of this evolved algorithm remains to be an open research area for future investigation.

Protein Docking: Protein molecules belong to the class of non-rigid deformable objects. Isometries on the regions of the protein molecules constitute docking sites where two protein molecules may form bonding, and form what is known as a quaternary structure. The new formed structure espouses new functional properties that may be taken advantage of in formulating new pharmaceutical drugs.

Amino acids are the building blocks of the proteins. An amino acid is a small molecule which constitute of amino group , NH_2 , a carboxyl group , $COOH$, and a hydrogen atom, H , attached to a central alpha α carbon. In addition there is side chain R connected to the central α carbon which characterizes the differences in structure and chemical properties among different amino-acids. The variations in R -group distinguish the twenty known amino acids that may be found in any protein molecules. Accordingly, the primary structure of a protein molecule constitute a chain of different types of amino acids (that are distinguishable by their R -group).

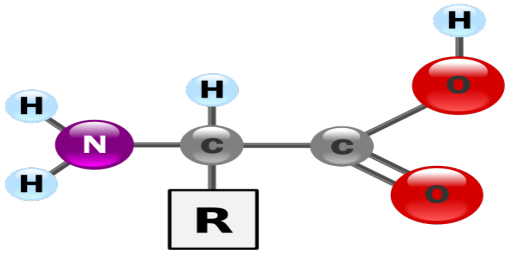


Figure 8-2 Amino acid molecule components

Amino acids may be chained together based on their chemical properties. The chain among amino acids takes place when the carboxyl group in one amino acid, and amino group in a second amino acid attract one another forming a peptide bond. When this process is repeated among many amino acids, then a polypeptide molecule representing the chain of the protein molecule is formed which constitutes the protein's primary structure.

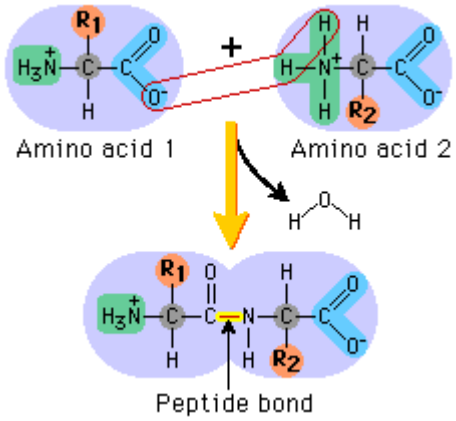


Figure 8-3 Peptide bond formation

Some amino acids are hydrophobic, others are polar, and many are electrically charged. These properties along with the locations of the individual amino-acids within the chain play significant roles on the energy levels associated with different parts of the chain. The variety of energy levels along the chain will trigger attractions and repulsions across the chain causing the molecule to fold or to bend in order to create an equilibrium state within the molecule. Combined with many other factors, the protein molecules find equilibriums in elaborate and complex three dimensional structures. When the protein

molecule is in a three dimension formation, each α carbon atom attached to its R group has assumed a distinct (x, y, z) coordinate in the three dimensional space. Therefore, the α carbon atom attached to the R group may be assumed as a vertex. Further, the polypeptide bond between two amino acids may be viewed as the edge connecting two vertices (in reality two α carbon atoms) with distinct (x, y, z) coordinates. Hence, the three dimensional structure of the protein molecule may be expressed by a graph $G_p = (V_p, E_p)$ such that V_p is the set of all α carbon atoms, and E_p is the set of all peptide bonds among the constituent amino acids.

In light of the formation of G_p, then different neighborhood graphs of G_p may be constructed and tested for isometry with neighborhood graphs from other G_q ($p \neq q$). Let us assume that $G_{sp} \subset G_p$ and $G_{sq} \subset G_q$. If G_{sp} and G_{sq} form an isometry, that is, if MQPSO find a permutation matrix that may relate the two, then G_{sp} and G_{sq} may be the candidate regions wherein the protein molecule represented by G_p may dock with the protein molecule represented by G_q (Figure 1-4). The possibility for the docking may be taken advantage of for constructing new pharmaceutical drugs.

A thorough investigation of MQPSO for the purpose of protein docking remains for future research.

Integer Programming: So far, in this thesis, the symbiosis between quantum PSO and genetic algorithm was explored in the domain of searching the solution space constituting of permutation matrices. We want to change the paradigm to the solution space $\mathbb{Z}$ wherein the constituents are integers. The symbiosis between quantum PSO and genetic algorithm may also be of utility in integer programming. Consider for example the function $F_1 = \|X\|$, $X = [X_1 \ldots X_D] \in [-100!, 100!]^D$ and D is the corresponding dimension ,with the objective of minimizing F_1. The solution space in this case is of colossal size. A swarm most likely traps in some local minimum before locating the global minimum. Now consider a swarm constituting of particles with binary integers that is

trapped in a local minimum. A mutation operator may mutate some bits in the particle with the best global position. The mutation of the bit(s) in the best global position instantly releases the swarm from the local minimum. In this case, the mutation operator will help the swarm to diversify its search strategies across the solution space. The mutation operator may select the bit(s) to mutate under the scheme of some probability distribution function.

The Choice of Potential Well Update: In the course of this thesis research we made use of Harmonic Oscillator Potential update, Equation (4-18), as our choice of attractive potential well update. This choice was made based upon the analysis and results presented in section 4.3. The analysis and results were obtained using the four test functions in section 4.3. The results in that analysis indicated that Harmonic Oscillator Potential Well update outperformed the other two potential well updates.

We suggest that the choice of potential well in quantum PSO is the subject of a more thorough investigation. In particular, the choice of potential well should be studied in the context of the localized domain and the nuances of the particular solution space. Consider the case of optimization, for example, in the domain of quantum computing or quantum biology [58]. Depending on the specific domain problem, the choice of potential well may not be trivialized. The choice of potential well, for example, may relate to how the particle movement in the domain problem is envisaged. Consider a particle that is free to move in two regions of space with a barrier erected between the two regions. In this case the Delta Well Potential update (Equation (4-17)) will more accurately represent the domain problem. On the other hand, Harmonic Oscillator Potential Update may be more suitable in the domain of oscillating genes [59] in molecular biology. Or, in general any system in equilibrium that is subjected to an external force acts as harmonic oscillator for small vibrations, and may be good candidate for Harmonic Oscillator Potential Update.

A more thorough investigation of the merits of different potential wells in quantum PSO is warranted.

Appendix A: Potential Well Updates

In the Quantum domain references to particle's trajectory is inaccurate due to Heisenberg uncertainty principle[9, 36]. The principle states that the particle's position x and velocity v may not be known simultaneously. Accordingly, one may speak of particle's state (rather than velocity or position). The particle's position may only be gauged after collapsing from Quantum time-space to Cartesian four-dimensional time-space.

In Quantum domain the quantum state of the particle is characterized by the wave-function [36] $\Psi(x,y,z,t)$. The interpretation of the wave-function may be given by Equation A.1.

$$|\Psi|^2 = dxdydz = \mathbb{Q}(x,y,z)\,dxdydz \qquad \text{A.1}$$

In Equation A.1 $\mathbb{Q}\,dxdydz$ is the probability associated with the particle's position about a point (x,y,z) in the classical Cartesian framework. Therefore, $|\Psi|^2$ maybe restated as probability density function satisfying Equation A.2

$$\int_{-\infty}^{+\infty} |\Psi|^2\, dxdydz = \int_{-\infty}^{+\infty} \mathbb{Q}\, dxdydz = 1 \qquad \text{A.2}$$

In the Quantum domain, however, the state of the particle, in single dimensions, evolves in time according to Schrodinger equation in Equation A.3

$$i\hbar \frac{\partial \Psi(x,t)}{\partial t} = H(x)\,\Psi(x,t) \qquad \text{A.3}$$

In Equation A.3 $H(x)$ is the time independent Hamiltonian operator:

$$H(x) = -\frac{\hbar^2}{2m}\nabla^2 + U(x) \qquad \text{A.4}$$

where in Equation A.4 $\hbar$ is the Planck constant, m is the mass of the particle, ∇ is the gradient operator (associated with the kinetic energy), and $U(x)$ is the potential constraining the dynamics of the particle.

For $U(x)$, any potential *well* maybe employed; however, the simplest is a Delta Potential Well given by [9, 38, 40]. We assume that the particle move in the Delta Potential Well [41] in the Quantum domain of which the center is P as given by Equation (4-20). Therefore, the potential energy of the particle in one dimension is given by the Equation [39] A.5.

$$U(x) = -\gamma\delta(x-p) = -\gamma\delta(y) \quad \text{A.5}$$

where γ is a positive number proportional to the depth of the potential well. In this model , the Hamiltonian operator becomes

$$H = -\frac{h^2}{2m}\frac{d^2}{dy^2} - \gamma\delta(y) \quad \text{A.6}$$

and the Schrödinger equation may be restated as

$$\frac{d^2\Psi}{dy^2} + \frac{2m}{h^2}[E + \gamma\delta(y)]\Psi = 0 \quad \text{A.7}$$

Doing the integral $\int_{-\varepsilon}^{+\varepsilon} dx, \text{ and when } \varepsilon \to 0^+$ we may obtain

$$\Psi'(0^+) - \Psi'(0^-) = \frac{-2m\gamma}{h^2}\Psi(0) \quad \text{A.8}$$

For $y \neq 0$, the Equation A.7 may be restated as

$$\frac{d^2\Psi}{dy^2} - \beta^2\Psi = 0 \,|\, \beta = \sqrt{-2mE/h}, E < 0 \quad \text{A.9}$$

To ensure that the particles remain in potential field,

$$|y| \to \infty, \Psi \to 0 \quad \text{A.10}$$

Then A.9 may be reformulated as in A.11

$$\Psi(y) \approx e^{-\beta|y|}, y \neq 0 \quad \text{A.11}$$

It may however be proved that only even wave-functions satisfy the bound conditions stated in A.10, we may restate A.11 in Equation A.12.

$$\Psi(y) = \begin{cases} Ce^{-\beta y} \mid y > 0 \\ Ce^{\beta y} \mid y < 0 \end{cases} \quad \text{A.12}$$

where C is the normalization constant.

Having reconsidered the Equation A.8, we may obtain

$$-2C\beta = -\frac{2m\gamma}{h^2}C \quad \text{A.13}$$

Thus

$$\beta = {}^{m\gamma}\!\!\diagup_{h^2} \quad \text{A.14}$$

$$E = E_0 = -\frac{h^2\beta^2}{2m} = -\frac{m\gamma^2}{2h^2} \quad \text{A.15}$$

Determining the constant in A.12 so that Ψ satisfies the normalization,

$$\int_{-\infty}^{+\infty} \Psi(y)dy = {}^{|C|^2}\!\!\diagup_{\beta} = 1 \quad \text{A.16}$$

Therefore we obtain $|C| = \sqrt{\beta}$. L= ${}^{1}\!\!\diagup_{\beta} = \frac{h^2}{m\gamma}$ is the characteristics length of Delta Potential Well. Let $C = \sqrt{\beta} = {}^{1}\!\!\diagup_{\sqrt{L}}$. We can now represent the normalized wave-function in Equation A.17.

$$\Psi(y) = \frac{1}{\sqrt{L}} e^{{}^{-|y|}\!\!\diagup_{L}} \quad \text{A.17}$$

Accordingly, the probability density function $\mathbb{Q}$ maybe given by Equation A.18.

$$\mathbb{Q}(y) = |\Psi(y)|^2 = \frac{1}{L} e^{-2|y|/L} \quad \text{A.19}$$

What does $\mathbb{Q}(y)$ give us? Recalling Quantum state function $\Psi(y)$ and the fact that $|\Psi(y)|^2 \equiv \mathbb{Q}(y)$, we need to remind ourselves that $\mathbb{Q}(y)$ gives the probability density function that the particle appears at position y relative to p. To learn the precise position of the particle we need to gauge the position of the particle, which is called collapsing the quantum state to classical state. The Monte Carlo Simulation may be employed to identify the precise location of a particle in a classical state. Let s represent the random number uniformly distributed in the interval $(0, 1/L)$, that is

$$s = \frac{1}{L} rand(0,1) = \frac{1}{L} u, \text{ where } u = rand(0,1) \quad \text{A.20}$$

Substituting s for $\mathbb{Q}$ in A.19, we may obtain

$$s = \frac{1}{L} e^{-2|y|/L} \quad \text{A.21}$$

And

$$u = e^{-2|y|/L} \quad \text{A.22}$$

$$y = \pm \frac{L}{2} \ln(1/u) \quad \text{A.23}$$

However recalling Equation A.5, $y = x - p$,

$$x = p \pm \frac{L}{2} \ln(1/u) \quad \text{A.24}$$

Where $u = rand(0,1)$. Accordingly, Equation A.24 provides accurate position of the particle in Delta Potential Well.

The same methodology maybe employed for Quantum Harmonic Oscillator. The wave function of the Quantum Harmonic Oscillator model is given by Equation A.25.

$$\Psi(y) = \frac{\sqrt{a}}{\pi^{1/4}} e^{\frac{1}{2}a^2y^2} \quad \text{A.25}$$

And the probability density function $\mathbb{Q}$ is

$$\mathbb{Q}(y) = |\Psi(y)|^2 = \frac{a}{\sqrt{\pi}} e^{-a^2y^2} \quad \text{A.26}$$

Using the Monte Carlo simulation, the position of the particle may be measured by Equation A.27.

$$y = \pm\frac{1}{a}\sqrt{\ln(\frac{1}{u})} \quad \text{A.27}$$

Or

$$x = p \pm \frac{1}{a}\sqrt{\ln(\frac{1}{u})} \quad \text{A.28}$$

In Equation A.27 and A.28 a is the characteristic length of Harmonic Oscillator field. Both a and L (in A.23 and A.24) may be considered as the control parameters.

The next step here involves as to how to select the control parameters L and a. Let's start with A.24 and control parameter L. The fundamental condition of convergence for all particles to arrive at P is when $x^m \to P^m$ for all particles $m = 1, 2, 3, ..., M$. This can be easily inferred in A.24 that $x = p$ when $L = 0$. Hence we need to develop a time evolving L such that Equation A.29 is satisfied.

$$\lim_{t\to\infty} L(t) = 0 \quad \text{A.29}$$

To guarantee that on average the particles will converge, we need the value of $|y_{k+1}|$ at iteration k to be closer to zero. A suitable probabilistic translation of this requirements may be given by Equation [9, 39] A.30.

$$1 > \int_{-\infty}^{|y_k|} \mathbb{Q}(y)dy > 0.75 \quad \text{A.30}$$

Solving the Equation A.30 we may proceed as $1 > \int_{-\infty}^{|y_k|} \mathbb{Q}(y)dy = \int_{-\infty}^{|y_k|} |\Psi(y)|^2 \, dy > 0.75$. We may then invoke A.19 to have $\int_{-\infty}^{|y_k|} |\Psi(y)|^2 \, dy = \int_{-\infty}^{|y_k|} \frac{1}{L} e^{-2|y|/L} dy$. The right side of the equation maybe written $\int_{-\infty}^{|y_k|} \frac{1}{L} e^{-2|y|/L} dy = \int_{-\infty}^{0} \frac{1}{L} e^{-2|y|/L} dy + \int_{0}^{|y_k|} \frac{1}{L} e^{-2|y|/L} dy$ which results in Equation [39] A.31.

$$1 - \frac{1}{2} e^{-2|y_k|/L} > 0.75 \qquad \text{A.31}$$

With the condition that $g > 0$ we may reduce control variable L to g as shown in Equation A.32 for the Delta Well Potential.

$$L = \frac{1}{g \ln \sqrt{2}} |x - p| \qquad \text{A.32}$$

The derivation of a formula for a in the Harmonic Oscillator Potential (A.28) is not analytically possible [9, p. 62]. Yet, a numerical solution may be obtained as shown in Equation A.33.

$$a = g \frac{0.47694}{|x - p|} \qquad \text{A.33}$$

Appendix B: Update Equation from Diffusion Equation

In this section, we demonstrate that the diffusion equation may be considered as the source of various probability density functions for arriving at the update functions describing the dynamics of the particles in PSO. In Appendix A, we explored Schrödinger equation as the source to derive the update functions. In this section, however, we make use of the analysis provided by Mikki [9] in arriving at an update function without any reference to the collapse of the wave function (as described in Appendix A), and using only probability theory. Later, we show Mikki [9] analysis that the conditional pdf is a common theme in both quantum and classical PSO. In other words, the conditional probability distribution underlies the stochastic dynamics of both approaches. This may be used as the unifying factor for both approaches. Accordingly, a generalized PSO algorithm may be developed without reference to the physical arguments underlying both quantum and classical approaches. Therefore Mikki has concluded [9] that all PSO variations are special classes of algorithms in probability theory.

Mikki [9] begins with a diffusion equation in Equation B.1

$$\frac{\partial \tilde{N}(x,t)}{\partial t} = D\nabla^2 \tilde{N}(x,t) \qquad \text{B.1}$$

As shown, a diffusion equation is a partial differential equation which describes the density dynamics in the material undergoing diffusion. In Equation B.1, $\tilde{N}$ is the concentration function of the molecules, x is the position of the particle, and D is the diffusion constant. The diffusion equation is of significant utility in research and study pertinent to the Brownian motion. Particles in a fluid appear to be moving randomly under the influence of the bombardment of the atoms/or molecules in the fluid. Yet, direct utilization of the equation leads to the solution for the concentration of the

particles with respect to time and space. In a one dimensional space the Equation B.1 may be simplified to B.2.

$$\frac{\partial \tilde{N}}{\partial t} = D\frac{\partial^2 \tilde{N}}{\partial x^2} \qquad \text{B.2}$$

In the classical interpretation, the Equation B.2 suggests that in the short time interval Δt the position of the particle changes from x to $x+dx$ towards the particle's center of gravity. Yet, moving to PSO paradigm, the center of gravity may be replaced by location P_n^i defined in Equation B.3

$$P_n^i = \frac{\varphi_1 P_n^{i,L} + \varphi_2 P_n^G}{\varphi_1 + \varphi_2} \qquad \text{B.3}$$

As discussed in 4.2.1, 4.2.2, and shown by Kennedy in [37], P_n^i may be considered natural center of gravity for the particles in PSO environment. Therefore, it is possible to define a new position variable defined by Equation B.4.

$$r_k^{i,n} = x_k^{i,n} - P_{k-1}^{i,n} \qquad \text{B.4}$$

In Equation B.4 k is the iteration index, i is the particle number, and n is the dimension. The involvement of P_n^i in Equation B.4 signifies that the motion of the particles from k iteration to $k+1$ iteration will not be continuous. In fact, the trajectory of the particles will not depict deterministic and continuous motion. On the other hand, the trajectory will be irregular motion reflecting stochastic paths in noisy environment. In light of this development, we may state that Equation B.1 can also be interpreted as the probability density function of the particle's motion. Assuming the boundary conditions $\tilde{N}(r,0) = \delta(r),\ \lim_{r\to\infty} \tilde{N}(r,t) = 0,\ \int_{-\infty}^{\infty} \tilde{N}(r,t) = M$, then the solution of B.2 may be written as in the Equation B.5.

$$\tilde{N}(r,t) = \frac{M}{\sqrt{4\pi D}} \frac{e^{-r^2/4Dt}}{\sqrt{t}} \qquad \text{B.5}$$

Appendix B: Update Equation from...

The Equation B.5 is the basis for derivation of the probability density function describing the movement of particles in PSO environment. This Equation may be interpreted as conditional pdf as the time variable is referenced to the origin. Given that the total trajectory of the particles may be viewed as succession of short intervals of time, then the conditional pdf at any instant can be expressed by Equation B.6.

$$f(x_k^{i,n}, t_k \mid x_{k-1}^{i,n}, t_{k-1}) = \frac{1}{\sqrt{4\pi D\ (t_k - t_{k-1})}} \exp\left\{ -\frac{(x_k^{i,n} - P_{k-1}^{i,n})^2}{4D\ (t_k - t_{k-1})} \right\} \quad \text{B.6}$$

Mikki [9] has further shown that the diffusion constant may be estimated by $D = \lim_{t\to\infty} \frac{(r(t) - r(0))^2}{6t}$.

In conclusion, Mikki [9] has shown that the conditional pdf of the particle's motion may be derived from the diffusion equation. It should be noted that this pdf is based on the boundary assumptions made earlier. Different boundary condition necessitates derivation of different probability density function.

The entire trajectory of the particle may be simulated given that the pdf of the particle at kth iteration is known. This is accomplished by using the known pdf to transform one random variable X to another random variable Y. Let us denote this transformation as $y = h(x)$. Then, the cumulative density function of both X and Y will be $F_X(x)$ and $F_Y(y)$ respectively. For the transformation h to produce the distribution $F_Y(y)$, starting from uniform distribution $F_X(x)$, one must have

$$y = F_Y^{-1}(x) \quad \text{B.7}$$

As an example, Mikki shows the Laplace distribution that would be obtained from delta well potential. The pdf of the particle trajectory can be written as

$$f(r_k^{i,n} \mid r_{k-1}^{i,n}) = \frac{g \ln\sqrt{2}}{|\, r_{k-1}^{i,n} \,|} \exp\left\{ -2g \ln\sqrt{2} \frac{|\, r_k^{i,n} \,|}{|\, r_{k-1}^{i,n} \,|} \right\} \quad \text{B.8}$$

where $r_k^{i,n}$ is defined by Equation B.4 and g is the control parameter. By solving for B.7, denoting the uniform random variable u, and position r by y, Mikki [9] obtains the update equation B.8.

$$x_k^{i,n} = \begin{cases} P_{k-1}^{i,n} + \dfrac{|\, x_{k-1}^{i,n} - P_{k-2}^{i,n} \,|}{2} \ln(2u) & 0 < u \le \dfrac{1}{2} \\ P_{k-1}^{i,n} + \dfrac{|\, x_{k-1}^{i,n} - P_{k-2}^{i,n} \,|}{2} \ln\dfrac{1}{2(1-u)} & \dfrac{1}{2} < u \le 1 \end{cases} \qquad \text{B.8}$$

As shown Equation B.8 is an update function describing the dynamics of the particle, and derived using only probability theory. Therefore, as shown by Mikki [9] the pdf reflecting the stochastic behavior of the particle may be derived from the Diffusion equation, leading to the update equation being derived from the pdf without using any physical argument.

Appendix C: Derivation of PSO Algorithm Using Markov Chains

In this section a consolidated approach to PSO developed by Mikki [9] based on Markov chain is given. A consolidated approach using Markov chains is driven given that a conditional probability density function may be developed underlying the shared stochastic dynamics of both quantum mechanics approach and diffusion equation approach.

Markov chain is a collection of random variables such that

$$P[x_n \leq x_n \mid x_{n-1}, x_{n-2}, ..., x_1, x_0] = P[x_n \leq x_{n-1}] \quad \text{C.1}$$

In C.1 it is understood that the cause starts at x_0 and propagates its effects through stochastic rules. In other word, it implies that the effect at x_k is only dependent on the cause at x_{k-1}. Therefore, if the information pertinent to the dynamic behavior at x_k is known, then it is possible to describe a Markov chain as a stochastic process where the dynamic behavior only recalls its state at x_{k-1}.

In Appendix B Equation B.6 and B.8 showed that by virtue of conditional pdf the position at iteration k is dependent upon the position at previous iteration at $k-1.$ In other words, the information pertinent to the current state k is adequate to derive the information pertinent to the next state at $k+1$. This corroborates with the behavior of both classical and/or quantum PSO algorithms. On this basis, Mikki [9] has classified PSO as a stochastic Markov chain.

Mikki [9] proceeds to make use of the chain rule in probability theory as defined in Equation C.2.

$$f(x_k, x_{k-1}, ..., x_1, x_0) = f(x_k \mid x_{k-1}) f(x_{k-1} \mid x_{k-2}) ... f(x_2 \mid x_1) f(x_0) \quad \text{C.2}$$

In Equation C.2, f stands the pdf of the random variables within its arguments. The chain rule implies that evaluations up to current iteration at k may be expressed in terms of conditional pdf at previous iteration including the first iteration, $f(x_0)$. The initial condition at x_0 is customarily a uniform random distribution over the range of interest. Further, in Markov chains, the Chapman-Kolmogorov (equation D.3) may be used to establish a casual link between any three iterations that may not necessarily be successive.

$$f(x_n \mid x_k) = \int_{-\infty}^{+\infty} d^N x_m f(x_n \mid x_m) f(x_m \mid x_k) \qquad \text{D.3}$$

In D.3 $n > m > k,\ d^N x_m$ stands for volume element in the abstract N dimensional space R^N. Yet, for the general case of successive iterations, the marginal pdf of the kth state may be stated as follows:

$$f(x_k) = \int_{x_{k-1}} \int_{x_{k-2}} \ldots \int_{x_0} d^N x_{k-1} d^N x_{k-2} \ldots d^N x_0 f(x_k, x_{k-1}, x_1, x_0) \qquad \text{D.4}$$

By substituting D.3 into D.4 Mikki [9] develops the pdf of future kth iteration by incorporating the effects of all previous iterations including the one originated at initial distribution.

$$f\left(x_k\right) = \int_{x_{k-1}} \int_{x_{k-2}} \ldots \int_{x_0} d^N x_{k-1} d^N x_{k-2} \ldots d^N x_0 f(x_k \mid x_{k-1}) \ldots f(x_2 \mid x_1) f(x_0) \qquad \text{D.5}$$

Therefore, a purely stochastic PSO algorithm may be developed. Accordingly, the knowledge of conditional pdf of the sequence of iterations together with the knowledge of initial distribution would be adequate to develop the total pdf for any future iteration [9]. The overall steps may be stated as below:

Step 1: Begin with an initial distribution with known pdf $f(x_0)$.

Step 2: Develop the conditional pdf $f(x_k \mid x_{k-1})$ for arbitrary k.

Step 3: Develop the marginal pdf $f(x_k)$ using D.5.

Step 4: Update the position using pdf $f(x_k)$ to realize the random number in an update function (such as the one developed in Appendix B)

The consolidated algorithm [9] has no physical terms involved and it is purely stochastic.

Appendix D: Quantum PSO vs. Classical PSO, Test Results

F2

Dimension Sizes	Swarm Size	Convergence Rate	# of fn calls	STD Deviation	G
		QPSO-Harmonic			
5	10	60%	197.83	108.68	1.76
5	15	90%	133.11	116.05	1.76
		QPSO-Delta			
5	10	71%	386.09	208.14	1.00
5	15	91%	82.96	34.42	1.12
		QPSO-Square			
5	10	62%	253.66	140.58	0.92
5	15	89%	110.38	84.44	0.94
		CPSO			
		Convergence Rate	# of fn calls	STD Deviation	Weight
5	10	92%	29.25	5.32	0.55
5	15	91%	15.97	2.98	0.35

F3

Dimension Sizes	Swarm Size	Convergence Rate	# of fn calls	STD Deviation	G
		QPSO-Harmonic			
2	10	83%	13.3	9.3	1.90
2	20	93%	6.98	3.63	1.90
		QPSO-Delta			
2	10	92%	15.59	9.09	1.16
2	20	99%	6.93	2.92	1.38
		QPSO-Square			
2	10	89%	97.38	204.1	0.85
2	20	95%	6.43	3.7	1.08
		CPSO			
		Convergence Rate	# of fn calls	STD Deviation	Weight
2	10	92%	7.83	2.68	0.4
2	20	100%	4.9	1.92	0.3

Dimension Sizes	Swarm Size	QPSO-Harmonic			
		Convergence Rate	# of F4 calls	STD Deviation	G
4	15				
4	20	61%	197.93	103.68	1.82
		QPSO-Delta			
4	15				
4	20	60%	481.02	239.78	1.03
		QPSO-Square			
4	15				
4	20	56%	393.57	192.06	0.94
		CPSO			
		Convergence Rate	# of F4 calls	STD Deviation	Weight
4	15	97%	71.21	21.57	0.70
4	20	94%	33.48	10.51	0.50

References

[1] A. M. Bronstein, M. M. Bronstein and R. Kimmel, Numerical Geometry of Non-Rigid Shapes, New York, NY: Springer, 2008.

[2] M. Bronstein, "Project Tosca," Technion, 2013. [Online]. Available: http://tosca.cs.technion.ac.il/. [Accessed 12 May 2013].

[3] "Science Daily," 5 December 2008. [Online]. Available: http://www.sciencedaily.com/releases/2008/12/081204133855.htm. [Accessed 2 May 2013].

[4] J. Gu and P. Bourne, Structural Bioinformatics, New Jersey: John Wiley & Sons, 2009.

[5] A. Kessel and N. Ben-Tal, Introduction To Proteins, Boca Raton, FL: CRC Press, 2011.

[6] J. Fruton, Molecules and Life, New York: Wiley-Interscience, 1972.

[7] "GVIL," [Online]. Available: http://www.cs.umd.edu/gvil/visproteomics.shtml. [Accessed 3 8 2012].

[8] E. Boros and P. L. Hammer, Discrete Optimization: The State of Art, Amsterdam: Elsevier Science, 2003.

[9] S. Mikki and A. Kishk, Particle Swarm Optimization: A Physics Based Approach, Morgan and Claypool, 2008.

[10] J. Kennedy and R. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, 2001.

[11] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan

Press, 1975.

[12] S. Chakrabarti, E. Cox, E. Frank, R. H. Güting, J. Han, X. Jiang, M. Kamber, S. S. Lightstone, T. P. Nadeau, R. E. Neapolitan, D. Pyle, M. Refaat, M. Schneider, T. J. Teorey and I. H. Witten, Data Mining: Know It All, Morgan Kaufmann, 2008.

[13] A. Elad and R. Kimmel, "On bending invariant signatures for surfaces" *IEEE Trans. PAMI,* vol. 25, pp. 1285-1295, 2003.

[14] A. Dubrovina and R. Kimmel, "Matching shapes by eigen-decomposition of the Laplace-Beltrami operator" in *Fifth International Symposium on 3D Data Processing Visualization and Transmission 3DPVT*, 2010.

[15] M. Gromov, "Structures metriques pour les varietes riemanniennes," *Textes Math,* vol. 1, 1981.

[16] F. Memoli and G. Sapiro, "A theoretical and computational framework for isometry invariant," *Found. Comput. Math,* vol. 5, pp. 313-346, 2005.

[17] A. BRONSTEIN, M. BRONSTEIN and R. Kimmel, "EFFICIENT COMPUTATION OF ISOMETRY-INVARIANT DISTANCES BETWEEN SURFACES," *Society for Industrial and Applied Mathematics J. SCI. COMPUT,* vol. 28, no. 5, p. 1812–1836, 2006.

[18] A. Bronstein and M. Bronstein, "Expression-invariant 3D face recognition," in *Proc. Audio Video-Based Biometric Person Authentication, Lecture Notes on Computer Science*, vol. 2688, pp. 62-70, 2003.

[19] A. Bronstein and M. Bronstein, "Three-dimensional face recognition," *International J. of Computer Vision,* vol. 1, pp. 5-30, 2005.

[20] A. Bronstein, M. Bronstein, A. Bruckstein and R. Kimmel, "Matching two-dimensional articulated shapes using generalized multidimensional scaling," in *Proc Articulated Motion and Deformable Objects*, *Lecture notes in Computer Science,* vol.

4069, pp. 48-57, 2006.

[21] R. Geiger, R. Basri, L. Costa and D. Jacobs, "Determining the similarity of deformable objects," *Vision Research,* vol. 38, pp. 2365-2385, 1998.

[22] B. Hong, E. Prados, S. Soatto and S. Vese, "Shape representation based on integral kernels: Application to image matching and segmentation," in *Proc. Computer Vision and Pattern Recognition*, vol. 1, pp. 833-840, 2006.

[23] M. Ruggeri and D. Saupe, "Isometry-invariant matching of point set surfaces," Stanford University, 2008. [Online]. Available: graphics.stanford.edu/courses/cs468-08-fall/pdf/RuSa08.pdf. [Accessed 6 8 2012].

[24] "Dynamic Programming," [Online]. Available: http://cis.jhu.edu/education/introPatternTheory/additional/dynamic/dynamic2.html. [Accessed 15 5 2013].

[25] E. Dijkstra, "A note on two connections with graphs," *Numeriche Mathematik,* vol. 1, pp. 269-271, 1959.

[26] A. Sethian, "A fast marching set method for monotically advancing fronts," in *National Academy of Science*, 1996.

[27] R. W. Floyd, "Algorithm 97," *Comm ACM,* vol. 5, pp. 346-350, 1962.

[28] Y. Liu, Y. Fang and K. Ramani, "Deformation invariant signatures for molecular shape," *BMC Bioinformatics,* vol. 10, pp. 14-21, 2009.

[29] E. Paquet and H. Viktor, "An Exhaustive Shape-Based Approach for Proteins' Secondary, Tertiary and Quaternary Structures Indexing, Retrieval and Docking," in *Protein Structure*, 2012.

[30] A. Chiang, Elements of Dynamic Optimization, Prospect Heights, IL: Waveland Press, 1991.

[31] G. Nemhauser, A. Rinnooy Kan and M. Todd, Optimization, Elsevier Science, 1989.

[32] J. Branke, Evolutionary Optimization in Dynamic Environments, Dordrecht: Kluwer Academic Publishers, 2002.

[33] K. Deb, Multi-objective Optimization Using Evolutionary Algorithms, Chichester: John Wiley & Sons, 2001.

[34] E. Laskari, K. Parsopoulos and M. Vrahatis, "Particle SwarmOptimization for Integer Programming," in *IEEE 2002 Congress on Evolutionary Computation*, 2002.

[35] K. Parsopoulos and M. N. Vrahatis, Particle Swarm Optimization: Advances and Applications, IGI Global, 2010.

[36] J. Sun, B. Feng and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Cong. Evolutionary Computation, CEC2004*, 2004.

[37] J. Kennedy and M. Clerc, "The particle swarm: explosion, stability, and convergence," *IEEE Trans. Evolution Comput.,* vol. 6, no. 1, pp. 58-73, 2002.

[38] P. Lindsay, Quantum Mechanics for Electrical Engineers, McGraw-Hill, 1967.

[39] J. Sun, B. Feng and W. Xu, "Particle Swarm Optimization Classical and Quantum Perspective," in *CEC2004*, 2004.

[40] F. Levin, An Introduction to Quantum Theory, Cambridge: Cambridge University Press, 2002.

[41] D. J. Griffiths, Introduction to Quantum Mechanics, Prentice Hall, 2005.

[42] Y. Sahillioğlu, "Yusuf Sahillioğlu Pub," [Online]. Available: http://www.cis.upenn.edu/~yusufs/pubs/. [Accessed 10 May 2013].

[43] Z. Xuelong and W. Jiwen, "High-quality pseudo-random sequence generator based on one-dimensional extended cellular automata," in *Proceedings of the 3rd international conference on Information security*, New York, 2004.

[44] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator," *ACM Transactions on Modeling and Computer Simulation,* vol. 8, no. 1, pp. 3-30, 1998.

[45] L. Devroye, Non-Uniform Random Variate Generation, New York: Springer, 1986, pp. 401-428.

[46] R. A. Horn and C. R. Johnson, " Matrix Analysis," in *Norms for Vectors and Matrices*, Cambridge, Cambridge University Press, 1990.

[47] J. W. Demmel, Applied Numerical Linear Algebra, SIAM, 1997.

[48] G. Arfken, "Lagrange Multipliers," in *Mathematical Methods for Physicists*, Orlando, Academic Press, 1985, pp. 945-950.

[49] J. Nocedal and S. J. Wright, "Penalty and Augmented Lagrange Multipliers," in *Numerical Optimization*, New York, Springer, 2006, pp. 497-505.

[50] S. Skiena, ""Lexicographically Ordered Permutations" and "Lexicographically Ordered Subsets"," in *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica.*, Addison-Wesley, 1990.

[51] B. Alberts, A. Johnson and J. Lewis, Molecular Biology of the Cell, 5 ed., New York, London: Garland Science Textbooks, 2007.

[52] W. Mathematica, "COMBINATORICA PACKAGE GUIDE," Wolfram, 2012. [Online]. Available: http://reference.wolfram.com/mathematica/Combinatorica/guide/CombinatoricaPackage.html. [Accessed 2 8 212].

[53] Wavefront, "Wavefront Standards," Wavefront Technologies, [Online]. Available: http://www.martinreddy.net/gfx/3d/OBJ.spec. [Accessed 22 March 2013].

[54] W. Penfield, The Mystery of the Mind : A Critical Study of Consciousness and the Human Brain, Princeton University Press, 1975.

[55] A. Mooney, "Obama seeks $100M to unlock mysteries of the brain," Cable Network News, 2 April 2013. [Online]. Available: http://www.cnn.com/2013/04/02/health/obama-brain-research. [Accessed 15 May 2013].

[56] R. Guénon, The Reign of Quantity, Hillsdale: Sophia Perennis, 2002.

[57] Wikipedia, "Milky Way," [Online]. Available: http://en.wikipedia.org/wiki/Milky_Way. [Accessed 15 5 2013].

[58] "Quantum Biology," University of Illinois, [Online]. Available: http://www.ks.uiuc.edu/Research/quantum_biology/. [Accessed 20 April 2013].

[59] L. M. Tuttle, H. Salis, J. Tomshine and Y. N. Kaznessis, "Model-Driven Designs of an Oscillating Gene Network," *Biophysics,* vol. 89, no. 6, pp. 3873-83, 2005.

[60] A. Kenny, A New History of Western Philosophy, vol. 2, Oxford: Clarendon Press, 2005.