IMPLEMENTATION OF BI-INCIDENCE FEASIBILITY REGIONS IN HELIX PACKING VIA CONSTRAINT SOLVING

By

ZHONGJIE LI

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2008

ACKNOWLEDGMENTS

I thank Dr. Jorg Peters for his advisory on my work. His in-depth knowledge in computational geometry and mathematics helped me a lot in completing my research. His working style and philosophy were also been beneficial to me.

TABLE OF CONTENTS

page

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

IMPLEMENTATION OF BI-INCIDENCE FEASIBILITY REGIONS IN HELIX PACKING VIA CONSTRAINT SOLVING

By

Zhongjie Li

August 2008

Chair: Jorg Peters
Major: Computer Engineering

By introducing computational geometry and computer science methods to solving the helix packing problem in biochemistry, we can assist the research. An algorithm of helix packing via constraint solving is being explored by the researchers at the University of Florida, which is based on a parametrization of the configuration space. As the first step of the algorithm, an angle based approach has been brought out to parametrize the first two incidences between two helices and therefore locate the third incidence. The algorithm successfully reduces our search space to four-dimensional space. Later on, the algorithm has been implemented in C++ in two phases. Phase 1 processes the input helix data with the algorithm then outputs the result as a text file. Phase 2 takes the output of Phase 1, and displays it with OpenGL.

CHAPTER 1
INTRODUCTION TO THE PROBLEM OF HELIX PACKING

Motivation

Since different structure aspects of proteins have been discovered by biologists in the mid 20th Century, scientists started to focus on revealing the essential mechanism of the transitions between these structure forms, such as folding that enables proteins to be able to perform their biological function. Fig. 1-1 shows an alpha helix which acts a critical role in understanding helix packing, a type of protein folding. However, because experimentally determining the three dimensional structure of a protein is often very difficult and expensive, scientists started to research on the possible ways of making predictions on the protein structure.

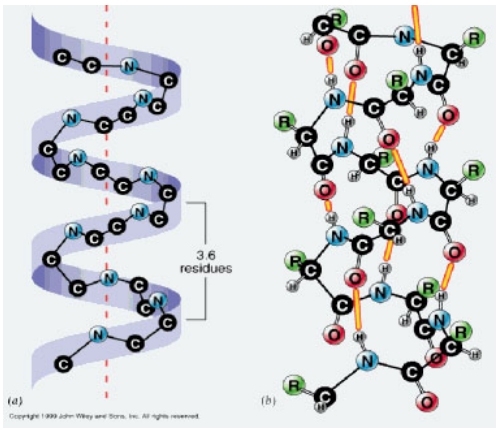

Figure 1-1. The α-helix.
(Source: http://www.uic.edu/classes/bios/bios100/lecturesf04am/lect02.htm . Last accessed July, 2008).

A number of factors exists that make protein structure prediction a very difficult task. The two main problems are that the number of possible protein structures is extremely large, and that the physical basis of protein structural stability is not fully understood. As a result, any protein structure prediction method needs a way to explore the space of possible structures efficiently (a search strategy), and a way to identify the most plausible structure (an energy function). Thus predicting the protein structure accurately remains a big challenge today.

In atomistic level, helices are often modeled as rigid 3-D objects. Hence due to their geometric nature, Computational Geometry methods can be introduced into this field. However, because of the complex structure of the helix, brutal force collision detection is often used in exploring the packing space. As a trade-off, this method consumes high computational power. In contrast to improving the computation capacity of computers, scientists are also seeking more efficient algorithms to solve the packing space with constraints.

Literature Review

The packing of transmembrane helices was modeled at the residual and atomistic levels, respectively. For predictions at the residual level, the helix-helix and helix-lipid interactions were described by a set of knowledge-based energy functions. For predictions at the atomistic level, the CHARMM19 force field (Chen and Xu, 2005) was employed.

In 2002, Fleishman and Ben-Tal developed a scoring function and a computational methodology for predicting the tertiary fold of a pair of alpha helices such that its chances of being tightly packed are maximized. The scoring function was constructed based on the qualitative insights gained in the past two decades from the solved structures of transmembrane (TM) and soluble proteins (Fleishman and Ben-Tal, 2002). Basically, they reward the formation of contacts between small amino acid residues such as Gly, Cys, and Ser, that are known to promote dimerization of helices, and penalize the burial of large amino acid residues such as Arg and Trp. Thus in computation, they can maximize the number of contacts between residues. This kind of solution is often called knowledge-based approach because of using known contact potentials, and the packing is modeled at the residual level instead of atoms level. A similar solution has been introduced in 2007 which uses coarse-grained knowledge-based potentials to score the mutual configuration TM helices (Wendel and Gohlke, 2008).

Direct simulation of protein folding in atomic detail, via methods such as molecular dynamics with a suitable energy function, is also a promising direction of the research. Because of the high computational cost of it, distributed computing technology has been introduced into this field. In 2000, a distributed computing project designed to perform computationally intensive simulations of protein folding and other molecular dynamics has been launched by Stanford University. The project is named as Folding@home and with the computational power volunteered by thousands or even millions private computer users across the globe (Shirts and Pande, 2000). It is one of the most powerful distributed computing clusters in the world. The goal of Folding@home is aimed on understanding protein folding, misfolding, and related diseases. With the assistance of the project, a number of achievements have been reached till today, such as the simulation of folding of a small α-helical protein in atomistic detail (Zagrovic et al., 2002) and atomistic protein folding simulations (Pande et al., 2003).

Problem Description, Definitions and Assumptions

In order to explore the space of possible structures in helix packing more efficiently, we are seeking for an algorithm with formal guarantees for sampling the configuration space of 2 packed helices with collision avoidance constraints by introducing computational geometry and other computer science technologies into this field. In our problem, according to the assumption, *helix* is a rigid 3-D object consisting of a set of m spheres (atoms) in fixed positions (Fig. 1-2), and the spheres can be intersected with others in the same helix. But in the process of packing, the spheres of different helices must **not** intersect. This collision avoidance is called *packing constraint*. It increased the difficulty of our problem due to the nonlinear feature it brought.

A *configuration* is defined by the relative position and orientation of 2 helices for a fixed position of the first helix. Hence a configuration has 6 independent degrees of freedom, thus can

be viewed as a point in $\Re^6$ space. If a configuration satisfied packing constraint, then we call it a *packing configuration*. A configuration of 2 helices is an *extreme configuration* if its incidences rigidly fix one helix to the other one. Generically, 6 incidences fix two helices respect to each other, but in some specific cases, 6 incidences may not be enough to form an extreme configuration (Fig. 1-3). Here we will mainly focus on the general case.

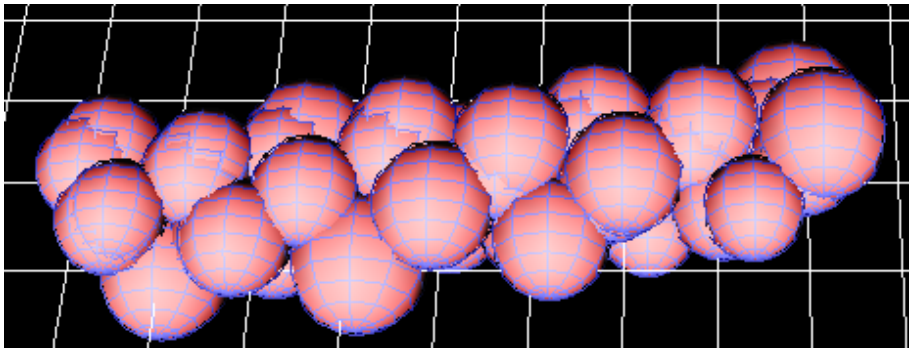

Figure 1-2. A helix is a rigid 3-D object consisting of a set of m spheres (atoms) in fixed positions.

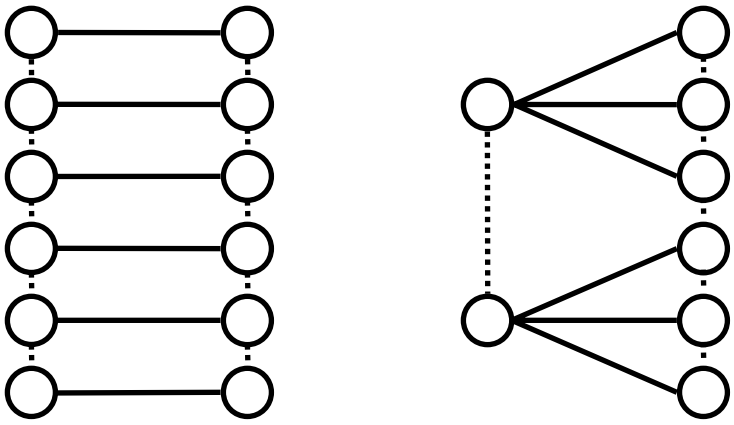

Figure 1-3. General case and a special case of incidences between 2 helices. Doted lines represent the connections within a helix. Solid lines represent the connections between helix. (a) is the general case where 6 incidences can fix the configuration of 2 helices. (b) is a special case where although 6 incidences happened, the configuration is still not fixed because right helix can rotate about the hinge of the connection between 2 atoms in left helix. (Source: Meera Sitharam, Heping Gao, Jorg Peters. 2007. Helix packing via Constraint Solving).

Our algorithm should contain every extreme packing configuration in the sample result, or a configuration arbitrarily close to it (within an input desired precision). For other configurations, we should be able to sample them with any desired (user input) precision. On the other hand, one of the noticeable drawbacks of existent algorithm on exploring the feasible regions of helix packing is that it will sample one extreme configuration multiple times. As a main factor in increasing the efficiency, our algorithm should outputs each sample configuration exactly once or at least same number of times (constant times).

In the following chapters, we will first introduce the algorithm developed by a group of researchers at the University of Florida. Then we will focus on the first step of the algorithm and an implementation of it.

CHAPTER 2
REVIEW OF BI-INCIDENCE FEASIBILITY REGIONS SOLVING IN HELIX PACKING

Review of Helix Packing via Constraint Solving

In early 2007, the idea of exploring the feasible regions of helix packing via constraint solving was explored by Dr. Meera Sitharam, Heping Gao, and Dr. Jorg Peters at the University of Florida. In their algorithm, the incidences between 2 helices will be parametrized, and based on this parametrization, the next incidence can be located in a range described by one or a set of equations achieved by applying collision detection between spheres. The algorithm will start from the parametrization on 2 incidences, then by recursively applying the principle, the number of contacts will be increased and hence the extreme configurations can be found.

As the first step of this algorithm, 2 incidences between 2 helices will be parametrized. To be more specific on the 2 incidences, the term of "Dumbbell" has been defined as a pair of intersecting spheres (a_1, a_2) within a helix a. A bi-incidence between a pair of helices a, b is hence defined as dumbbells (a_1, a_2) and (b_1, b_2) touch each other: a_1 is incident on b_1 and a_2 is incident on b_2 (Fig. 2-1). Note that an incidence between two spheres does not pin down the point of incidence. Therefore each such incidence removes only one degree of freedom, not three.

Within the bi-incidence, the packing constraint should also be enforced. Thus we can derive equations according to this constraint to parametrize the bi-incidence configuration space. The parametrization can be done in both the distances and the angles between the dumbbells. In the distance based approach, the constraint can be reflected by the distance between a_1 and b_2, and a_2 and b_1. But here we will focus on the angle based approach, which will be reviewed later.

After the bi-incidence has been fixed by the parameters, we can further rotate the helices about the hinge of the dumbbells respectively, and use certain parameters to locate the next

incidence. Therefore, in the next step, we can seek for a method to parametrize the 3-incidences then locate the 4th incidence accordingly, and so on.

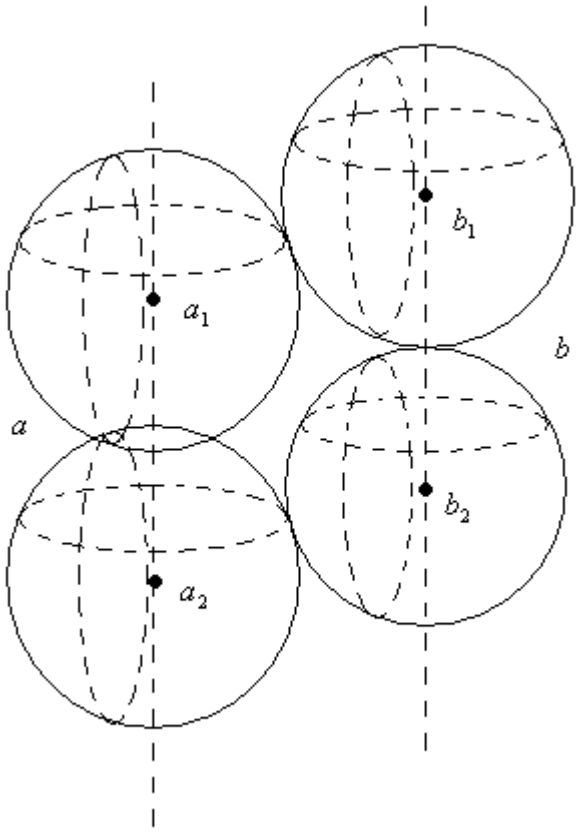

Figure 2-1. Dumbbell (a_1, a_2) from helix a and dumbbell (b_1, b_2) from helix b form a bi-incidence. (Source: Meera Sitharam, Heping Gao, Jorg Peters. 2007. Helix packing via constraint solving).

Review of Bi-incidence Feasibility Regions Solving in Angle Approach

As we stated before, the bi-incidence space consists of one dumbbell from each of the two helices. Dr. Jorg Peters at the University of Florida derived the parametrization of the bi-incidence with angle approach in his draft "Bi-indicence feasibility regions in helix packing via constraint solving". For simplicity, we position one dumbbell (D_a) on the x-axis and one of its spheres centered at the origin. The rest of Helix a will be translated accordingly. Then we move the dumbbell (D_b) of the other helix b to a position that form a bi-incidence contact with D_a.

By specifying the two angles (ϕ and ψ) of the bi-incidence, we fix the configuration between these two dumbbells. For each sampled configuration, we will also translate the rest of Helix b accordingly. Then by sampling on β which denotes the angle Helix b rotates about the hinge of D_b, we can determine the feasible region of α, the angle Helix a rotates about the hinge of D_a. Therefore, the output of solving bi-incidence feasibility regions will be sets of angles ($\phi, \psi, \alpha, \beta$).

Parametrization of a Dumbbell Sliding on a Fixed Dumbbell (Solving for ϕ and ψ)

We fix one of the dumbbells to have its axis aligned with the x-axis and one endpoint at the origin (Fig. 2-2).

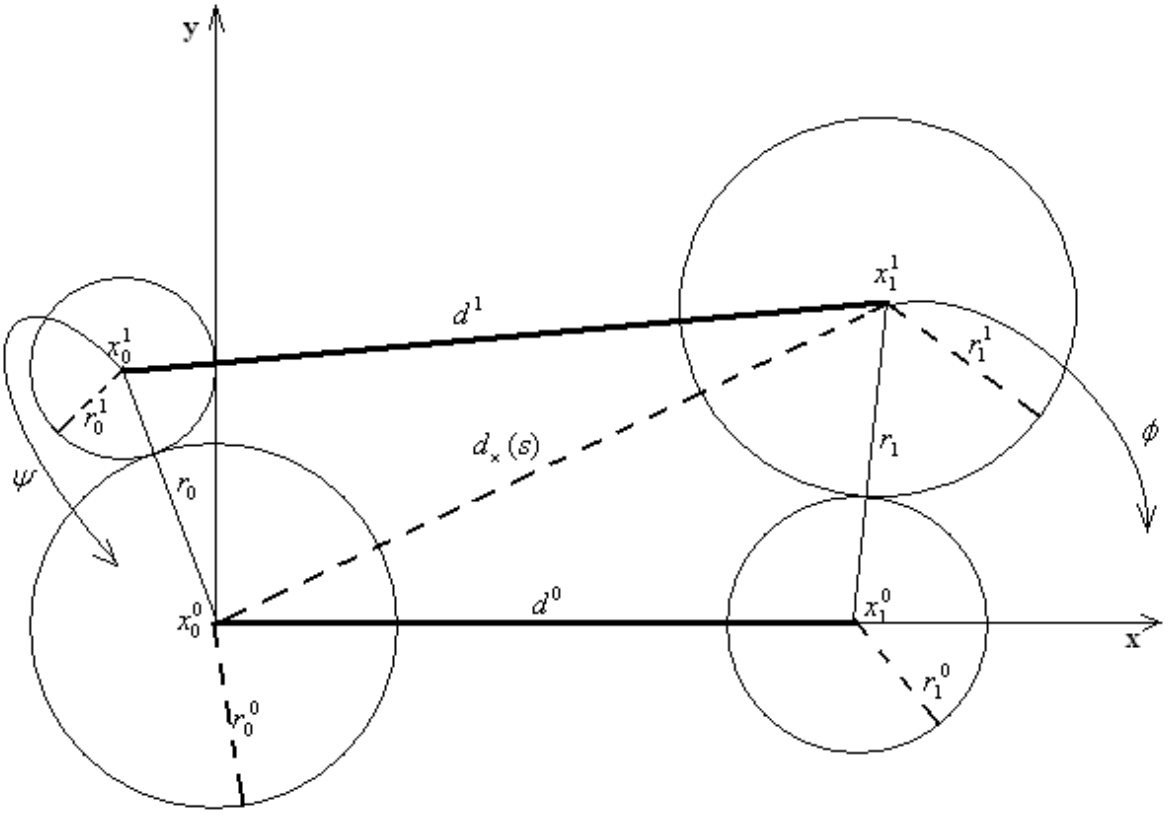

Figure 2-2. Bi-incidence of a pair of dumbbells and the parametrization (Source: Meera Sitharam, Jorg Peters, and James Pence. 2007. Bi-incidence feasibility regions in helix packing via constraint solving).

x_0^i, x_1^i, the endpoints of dumbbell i, (2-1)

r_0^i, r_1^i, the corresponding radii, (2-2)

r_j, the sum of radii of x_j^0 and x_j^1, (2-3)

d^i, the distance between x_0^i and x_1^i within dumbbell i, (2-4)

$d_\times$, the axis formed by connecting x_0^0 and x_1^1. (2-5)

For a clear view of the figure, although in practice a dumbbell should consist of two intersecting spheres, here we do not require that constraint. That is, $d^i \geq r_0^i + r_1^i$ need not hold. And since our algorithm here does not rely on that constraint, the result is still valid.

The parametrization places

- x_1^1 on a circle in the xy-plane with parameter ϕ of radius r_1 about x_1^0.
- x_0^1 on a circle with parameter ψ of a cone around $d_\times$.

$$d_x := \left\| x_1^1 - x_0^0 \right\| = \sqrt{(d^0)^2 + 2d^0 \cos\phi \cdot r_1 + (r_1)^2} \quad (2\text{-}6)$$

To keep the bi-incidence property, the feasible region for the parameters ϕ and ψ is constrained by

$$\phi:\ d_\times^2 \geq (r_0^0 + r_1^1)^2 \quad (2\text{-}7)$$

$$\phi:\ d_x^2 \leq (r_1 + d^0)^2 \quad (2\text{-}8)$$

$$\phi:\ d_\times^2 \leq (r_0 + d^1)^2 \quad (2\text{-}9)$$

$$\psi:\ \left\| x_0^1 - x_1^0 \right\|^2 \geq (r_1^0 + r_0^1)^2 \quad (2\text{-}10)$$

Since (2-8) is trivial which always holds, by combining (2-7) and (2-9), we have the region of ϕ:

$$\frac{(r_0^0 + r_1^1)^2 - (r_1)^2 - (d^0)^2}{2r_1 d^0} \leq \cos\phi \leq \frac{(r_0 + d^1)^2 - (r_1)^2 - (d^0)^2}{2r_1 d^0} \quad (2\text{-}11)$$

Then for each feasible ϕ, we only need to find out ψ that will not cause intersection between dumbbells. We note that $\left\| x_0^1 \right\|^2 = (r_0)^2$. Define a as $a := \dfrac{d_\times^2 - (d^1)^2 + r_0^2}{2d_\times^2}$

Then the first coordinate of x_0^1 is

$$(d^0 + r_1\cos\phi)a - r_1\sin\phi\sin\psi\sqrt{(\frac{r_0}{d_\times})^2 - a^2}\ .$$

Therefore the left hand side of (2-10) becomes

$$\left\|x_0^1 - x_1^0\right\|^2 = (r_0)^2 - 2d^0((d^0 + r_1\cos\phi)a - r_1\sin\phi\sin\psi\sqrt{(\frac{r_0}{d_\times})^2 - a^2}) + (d^0)^2$$

Here we define k_ϕ as:

$$k_\phi := \frac{r_1\sin\phi\sqrt{(\frac{r_0}{d_\times})^2 - (\frac{d_\times^2 - (d^1)^2 + r_0^2}{2d_\times^2})^2}}{(d^0 + r_1\cos\phi)a - \frac{(r_0)^2 + (d^0)^2 - (r_1^0 + r_0^1)^2}{2d^0}} \qquad (2\text{-}12)$$

The constraint on ψ will be:

$$\frac{1}{k_\phi} \le \sin\psi \qquad (2\text{-}13)$$

Moving a Helix to a Position of Its Dumbbell

Since in our current stage, the dumbbell lies on x-axis, D_a, will be a test dumbbell which is preset, we need only translate D_b. The main idea is to first move Helix b from its original position O_b to where the sphere x_1^1 overlapped with current x_1^1. Then we calculate the rotation matrix to rotate vector (x_1^1, x_0^1) to make x_0^1 also overlapped with current one. At last we apply this matrix to the whole helix. The detailed translation process will be provided in the next chapter.

Helix Hinge Motion and Intersection (Solving for α and β)

The dumbbell axes serve as hinges around which the remainder of the helix can rotate by angles (α, β). Here we only give each dumbbell a and b one additional sphere a_3,

respectively b_3, rather than checking all m spheres. To extend the parametrization to a full helix, we can test all the spheres start from the nearest ones to the dumbbell, then find the intersection of all the regions.

For a fixed feasible (ϕ, ψ), we seek, for each choice of a_3 and b_3, the intersection of the feasible (α, β) regions (hinge angles so that the two additional spheres do not intersect one another) (Fig. 2-3).

$P^i(\gamma)$: the position for sphere attached to dumbbell i, where γ indicates the rotation angle. $\gamma = 0$ represents the initial position of P^i (2-14)

δ_i: the origin of the rotation of P^i (2-15)

ρ^i: corresponding radius of the additional sphere (2-16)

To avoid intersect between a_3 and b_3, the following equation should be satisfied:

$$\left\| P^0(\alpha) - P^1(\beta) \right\|^2 > (\rho^0 + \rho^1)^2 \tag{2-17}$$

Set

$$P := P^1(0) - x_0^1 \tag{2-18}$$

$$h := \frac{x_1^1 - x_0^1}{\left\| x_1^1 - x_0^1 \right\|} \tag{2-19}$$

$$c_\beta := \cos\frac{\beta}{2} \qquad s_\beta := \sin\frac{\beta}{2} \tag{2-20}$$

$$v(\beta) := x_0^1 + (c_\beta^2 - s_\beta^2)P + 2c_\beta s_\beta P \times h + 2s_\beta^2 (h \cdot P)h - \begin{bmatrix} x_0^1 \\ 0 \\ 0 \end{bmatrix} \tag{2-21}$$

$$k(\beta) := \frac{1}{2\delta_0}(\left\| v(\beta) \right\|^2 + (\delta_0)^2 - (\rho^0 + \rho^1)^2) \tag{2-22}$$

As a constraint between α and β, the feasible region is characterized by

$$\cos\alpha \cdot v_2(\beta) + \sin\alpha \cdot v_3(\beta) < k(\beta) \tag{2-23}$$

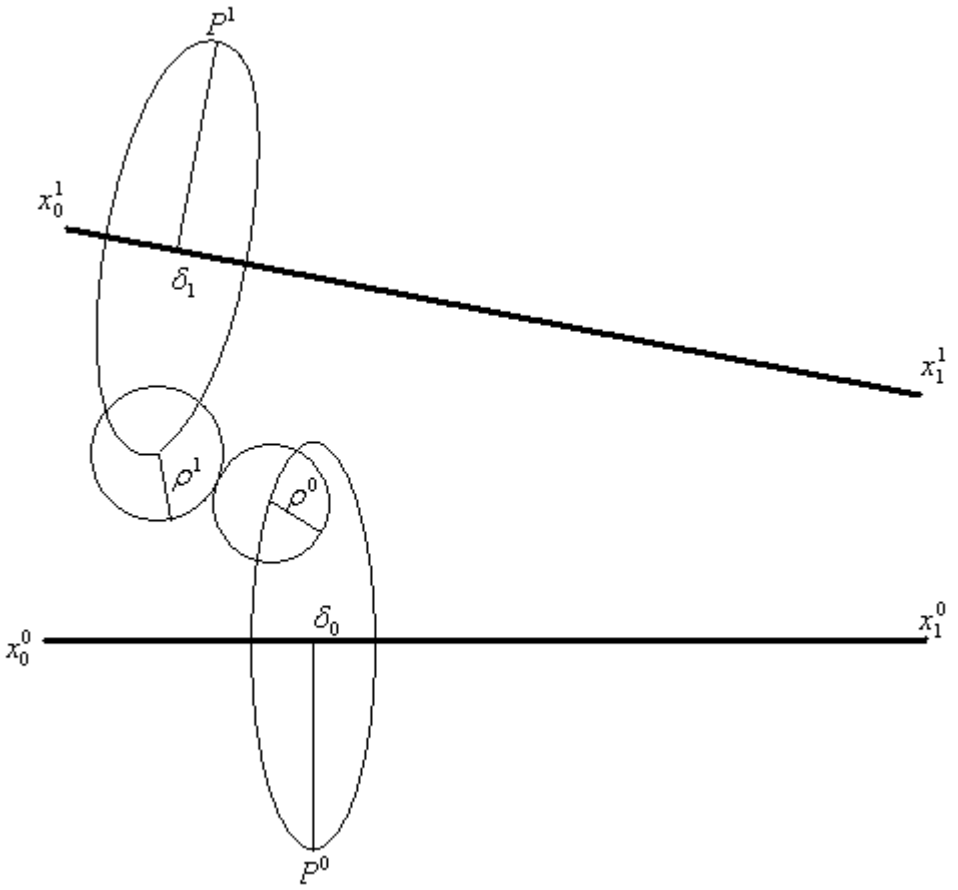

Figure 2-3. Sphere intersection when rotating about the hinge (Source: Meera Sitharam, Jorg Peters, and James Pence. 2007. Bi-incidence feasibility regions in Helix packing via Constraint Solving).

Therefore we has constrained the feasible region for each of $\phi, \psi, \alpha, \beta$. By combining them together, the feasible region will be in $\Re^4$ space.

CHAPTER 3
IMPLEMENTATION OF BI-INCIDENCE FEASIBILITY REGIONS SOLVING ALGORITHEM

Implementation Environment

Here we use C++ to implement the algorithm. The develop environment is Microsoft Visual C++. To display the result of ϕ, ψ, α and β, we included the library of OpenGL into our program to draw them on the screen.

Overview on the Structure of Implementation

The program is divided into two phases.

The first phase reads the data of the helix, and with a specified preset test helix we apply the algorithm of bi-incidence feasibility regions solving on them. It will have a formatted text file of the sampled ϕ, ψ, α and β as the output. It can be used to solve all the valid dumbbells in the input helix as well as any user specified dumbbell only.

The second phase takes the output of the first one, then with the data on both helices (the input one and the preset test helix), it displays the result in a 3-D scene with the shapes of helices.

Implementation: Phase 1

The input data of a helix is organized as a text file, specifying the position and the radius of each sphere in the helix. For a sample input, please refer to the section of Appendix. First of all, we open the file in C++ and read all the data into memory. Also, we specify our test helix. In the form of *Sphere(center_x, center_y, center_z, radius)*, the three spheres of test helix are*:* *testSphere_1(0, 0, 0, 0.5), testSphere_2(1, 0, 0, 0.5), and testSphere_3(0, 1, 0,* $\frac{\sqrt{2}}{2}$*).* As for the algorithm, *testSphere_1* serves as the sphere located on x_0^0, *testSphere*_2 will be located on x_1^0, and *testSphere_3* will be α_3. Then we do a search on the intersection between each pair of

spheres in the input helix. If we found an intersection pair, they will form the dumbbell that we will work on.

Solving ϕ

For each pair of dumbbells, the first read in sphere will serve as the one located on x_0^1, and the second one will be at x_1^1. Also we get the radius of both and the distance between their centers. Since solving ϕ and ψ doesn't require the dumbbells to be actually touched, we do not need to translate the chosen dumbbell of the helix to its configuration for now. Then by applying equation (2-11), we can have the range of $\cos\phi$. In C++, due to the precision of calculation, the result can be slightly out of the allowed range. For example, in some of the cases $\cos\phi$ can be equal to 1.0000001. Therefore, after each calculation requires meaningful boundary such as sin and cos, we do a manual round up, such as:

```
if (lowerBoundCosPhi < -1){
	lowerBoundCosPhi = -1;
}
if (upperBoundCosPhi > 1){
	upperBoundCosPhi = 1;
}
```

We noticed that in C++, the acos function will give out the result in the region of $[0,\pi]$. However, because the constraint only apply to cos value, there should be two feasible regions. So here we use the result of acos to get the other region also.

Solving ψ

Based on the result regions, we make an even sampling on the feasible values. Then for each feasible ϕ, we apply equation (2-13) to get the feasible region of $\sin\psi$.

An issue here is that the divider of (2-13) can possibly be 0, which will cause unexpectable result of ψ. The 0 factor can be from $\sin\phi$ or $\sqrt{(\frac{r_0}{d_\times})^2-(\frac{d_\times^2-(d^1)^2+r_0^2}{2d_\times^2})^2}$. In either case, we go back to check the original constraint

$$(r_0^1+r_1^0)^2 \le (r_0)^2 - 2d^0 d_\times \frac{d_\times^2-(d^1)^2+r_0^2}{2d_\times^2} + (d^0)^2 \qquad (3\text{-}1)$$

If it holds, then we assert that all of ψ values are valid, thus $\psi \in [0,2\pi]$. Otherwise, there is no valid ψ. Also we will need to check if the equation asks $\sin\psi > 1$. If so, we also assure that there is no valid region for ψ.

After all boundary checks have been done, we use asin function in C++ to solve ψ. Then we project the result to the region of $[0,2\pi]$. It may also cut the region into two. Then we sample in the region of feasible ψ and work for each one.

Translate the Whole Helix to Make Sure the Chosen Dumbbell Lies in the Configuration

Before we proceed with solving α and β, we need to translate the whole helix to the position specified as the chosen ϕ and ψ.

Based on previous process, we now can get the position of x_0^1 and x_1^1. Now we refer the original position of the dumbbell spheres as $x_0^1\_o$ and $x_1^1\_o$. By subtracting $x_0^1\_o$ from $x_1^1\_o$, we are going to have a vector denoted as V_o. Correspondingly, subtracting x_0^1 from x_1^1 will give out another vector denoted as V_c. Because the length of V_o and V_c are same, the process of translating V_o to V_c consists of only two parts, translation which refers to moving the vector, and rotation. In the rotation process, we use x_0^1 and $x_0^1\_o$ as the origins respectively. Our first step is to compute the rotation matrix to put V_o into the plane $x-y$. Then we rotate V_o about z-axis

to align it to x-axis. At last we rotate it about y-axis and z-axis to make it align with V_c. After rotation, the process of translation is trivial since we only need to compute the difference between $x_0^1\_o$ and x_0^1.

Then by applying the 4 rotation matrices and 1 translation matrix, we can successfully move the whole helix. In order to ensure the correctness of the transformation, we deployed a piece of code to check the result.

Solving α and β Pair

Recall equation (2-23)

$$\cos\alpha \cdot v_2(\beta) + \sin\alpha \cdot v_3(\beta) < k(\beta)$$

To translate this equation into a form more suitable for solving, set

$$R := \sqrt{(v_2(\beta))^2 + (v_3(\beta))^2} \tag{3-2}$$

If $R \neq 0$, we calculate an offset angle λ as

$$\lambda := a\cos(\frac{v_2(\beta)}{R}) \tag{3-3}$$

Then (2-23) is equivalent to

$$R \cdot \cos(\alpha - \lambda) < k(\beta) \tag{3-4}$$

Our approach to get the (α, β) pair is to sample on β in the region of $[0,2\pi]$. Therefore, for each β sample value, we have a range of $\cos(\alpha - \lambda)$, and we are able to get the feasible region of α. If there is no valid α, the sample value of β is consequently not valid.

Since α is constrained by a cos function, the result can still have two regions. We use a similar technique as in solving ϕ to handle the situation.

User Specified Dumbbell Solving

As an additional feature of the implementation, besides solving all the dumbbells of a helix, our program allows the user to specify one dumbbell and get its result accordingly. User will need to indicate the index number of the two spheres of the dumbbell. If the two cannot form a dumbbell, an error message will be out.

Output the Result

As we running the program, after each valid α region is given out, we will write a line to the output text file. The structure of the output content is organized as a tree (Fig. 3-1).

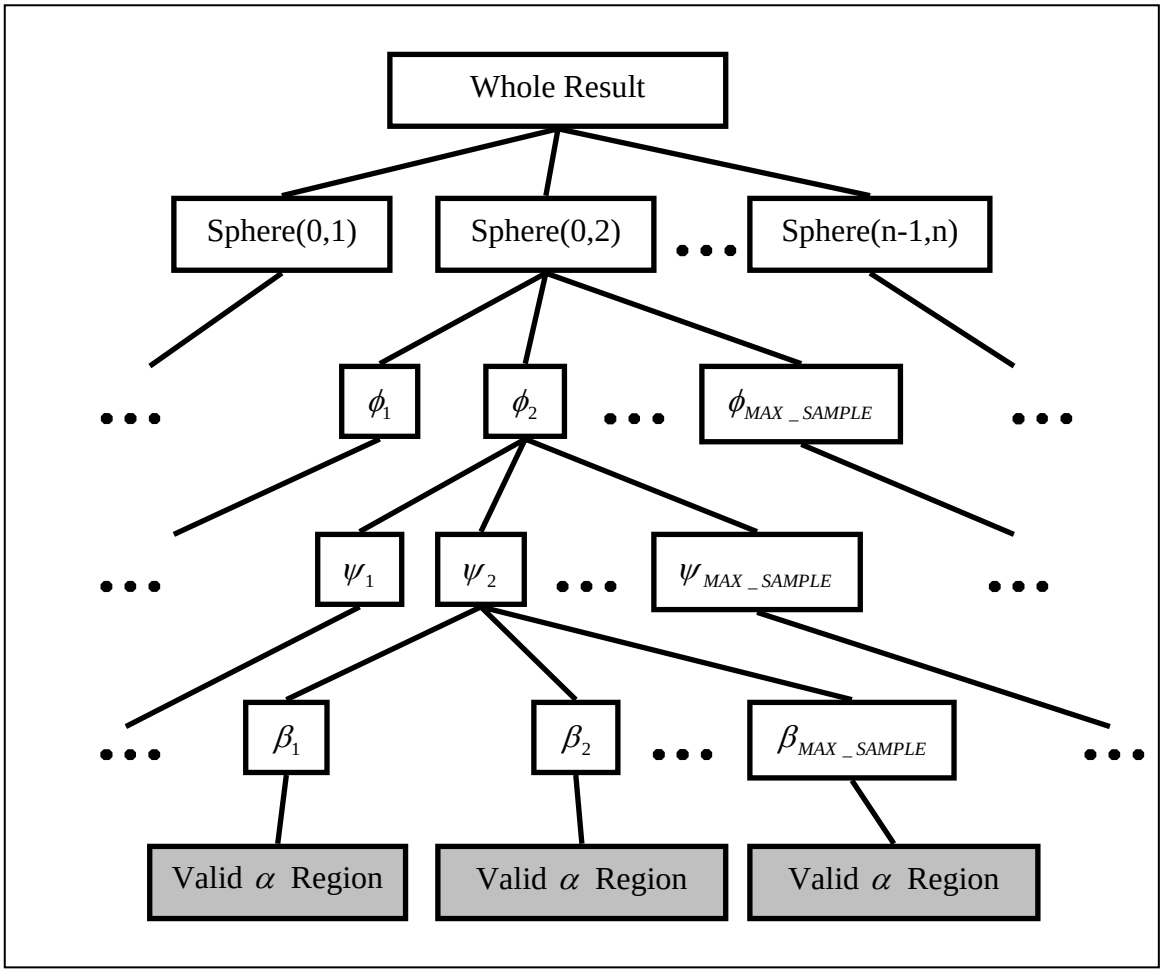

Figure 3-1. Structure of output file

Implementation: Phase 2

Since the result (ϕ, ψ, α, β) forms a space in $\Re^4$, a shape in 3-D is not enough to represent the whole feasible region. Therefore we first map the selected ϕ, ψ, and β to x, y, and z axis respectively. Then we use a colored line segment to represent the valid α region for each 3-D space selected point (Fig. 3-2). Red represents not valid regions, while blue represents valid regions.

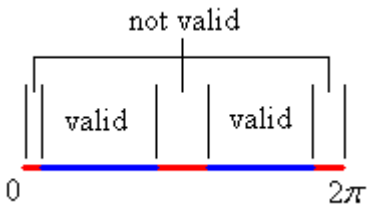

Figure 3-2. Line segment to represent the α region

Besides the feasible region, the program also displays the input helix with solid red spheres as selected dumbbell and wired blue spheres as other spheres. And the test helix is displayed in its own coordinate system as well.

For a better view, the program can be switched between showing (ϕ, ψ, β) regions, which will only consists of points, and showing the whole space.

In Fig 3-3, the input helix is on the left side, and the test helix lies on the right side. Centered is a wired yellow cube represents the boundary of the whole space. The strong red line is x-axis representing ϕ, the strong yellow line is y-axis representing ψ, and the strong blue line is z-axis representing β. Except $y-\psi$, others are mapping $[0,2\pi]$ region to [-5, +5], while $y-\psi$ is mapping $[0,2\pi]$ to [0, +10]. In Fig. 3-4, we only display (ϕ, ψ, β) regions as points in the same coordinate system.

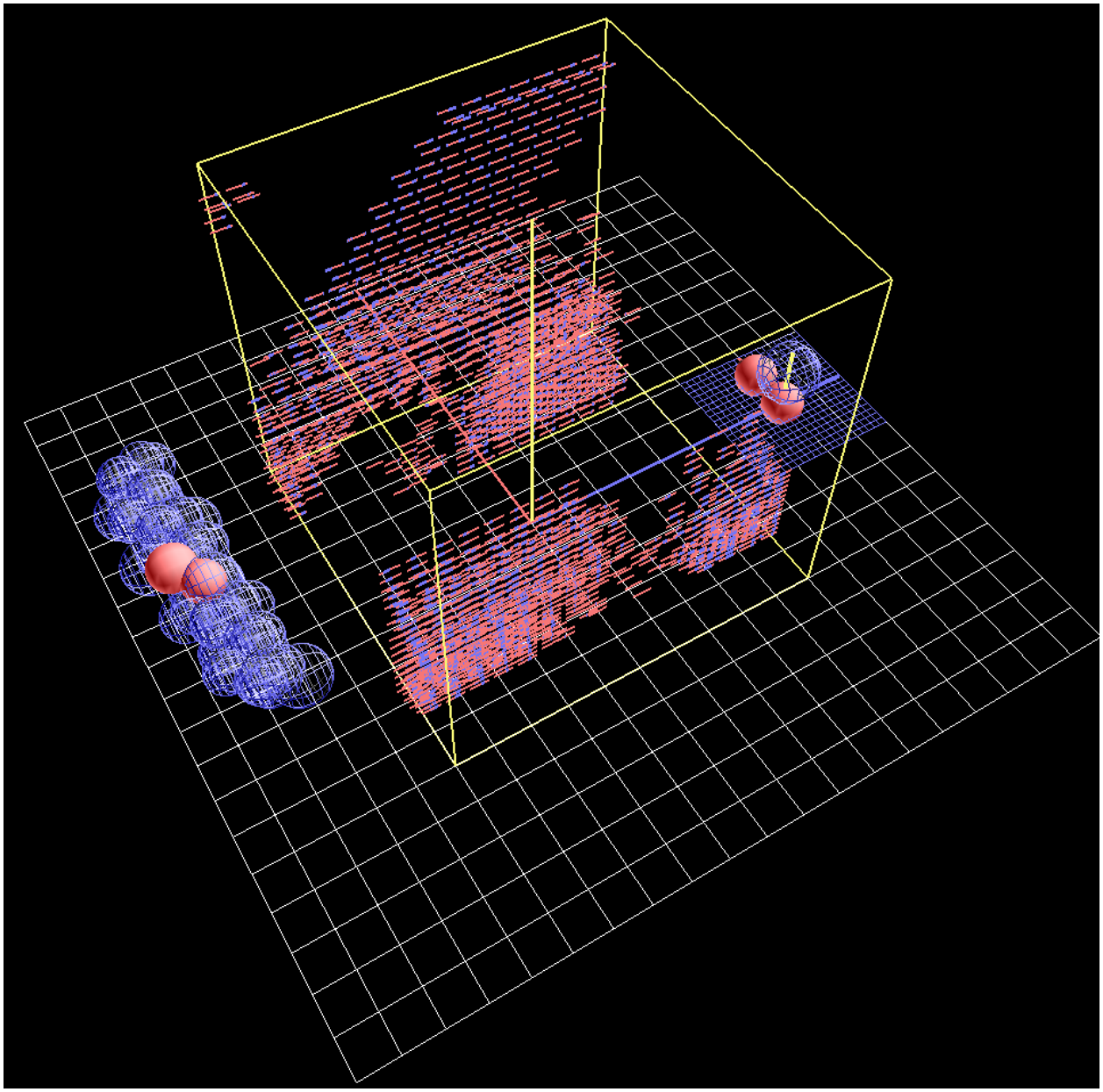

Figure 3-3. Sample output of Phase 2 with α regions.

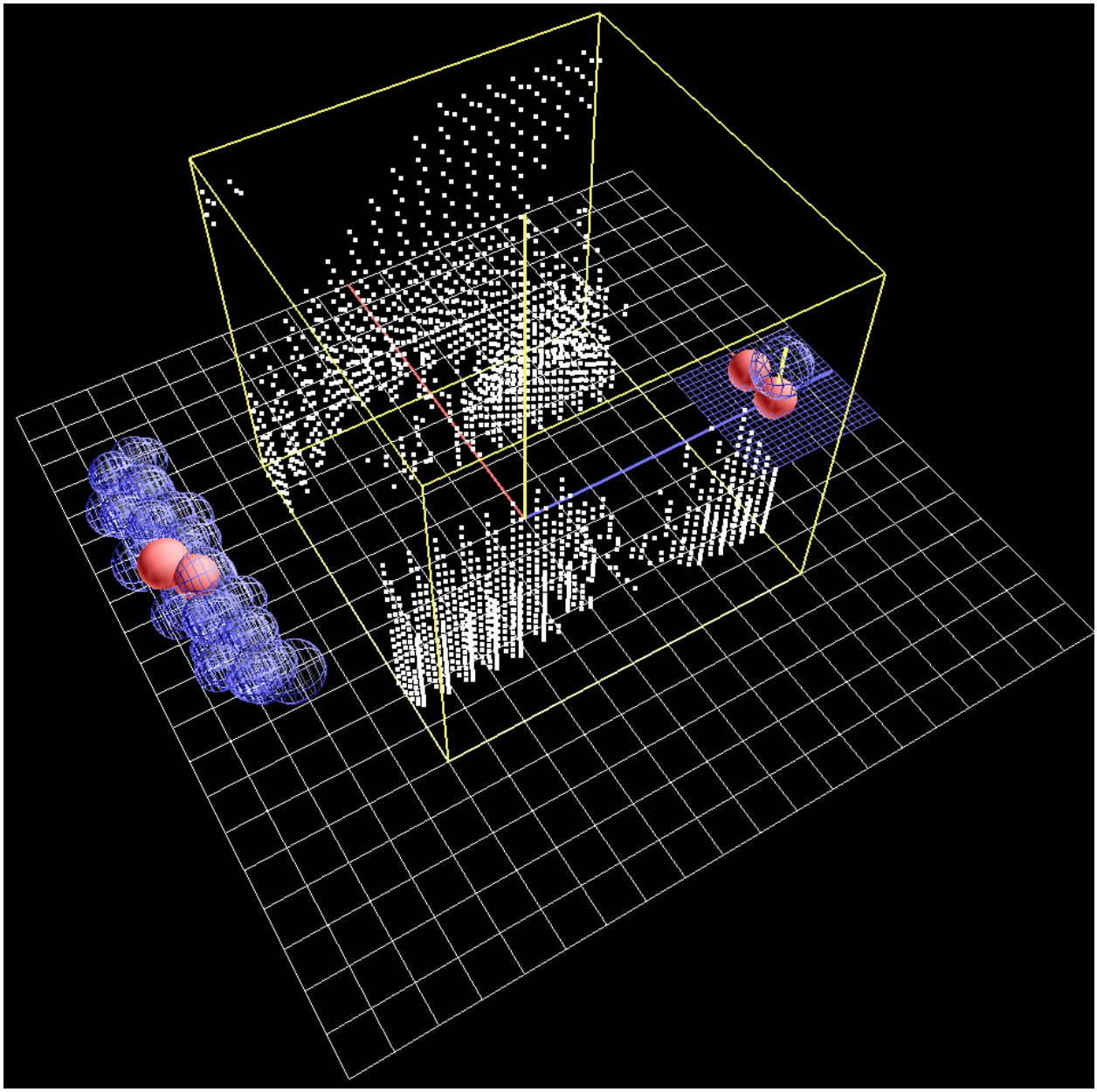

Figure 3-4. Sample output of Phase 2 without α regions.

CHAPTER 4
CONCLUSION

Effectiveness of Bi-Incidence Feasibility Regions Solving

In the process of sampling the configuration space of 2 helices so that they do not intersect, we are particularly interested in including those extreme configurations in our samples. By bi-incidence parameterization, we can effectively locate the feasibility regions and hence the first 3 incidences between the 2 helices avoiding sampling of collision detection. In general, this process will leave the configuration space with 3 degrees of freedom.

Implementation of the Algorithm

In the implementation process, we noticed that in the process of solving ψ, the divider of equation (2-13), k_ϕ can possibly be 0. This will cause unexpected result in the program. However, due to equation (2-10), we can interpret this case correctly by checking equation (3-1). Besides, our implementation handles the precision and return values of acos and asin in C++ properly.

The result of the implementation program has been tested to be correct. And there are also codes in the program for testing purpose after some important steps such as the translation of the whole helix. In phase two, the output of phase one can be handled correctly and user can also interact with the display to get a better view on the feasible regions.

Challenge and Future Development

After 3 incidences have been specified by our approach, we can do collision detection based on these constraints. This is also more efficient than doing collision detection in the earliest stage. A better way, to be explored, is to parameterize a triple incidence, then 4 incidences, etc. This would also prevent “holes” due to insufficient sampling. At present, with sampling, a β value in between of 2 adjacent sampled valid βs may not be valid so we cannot

guarantee that we have found all the extreme points in the result region. Possibly, the first step to parametrize a triple incidences is abstracting the configurations to the normal of the triangles formed by the 3 spheres of each helix respectively.

Also in the implementation, a potential improvement is that during our computation on α and β, what we did is apply the equation for each pair of the spheres. However, in practice some of the spheres, such as two far away spheres, can be trivially tested with a boundary on all of their possible locations. This method can bring improvement on the complexity of the algorithm although how much it may improve is depending on the exact configuration of the two helices.

APPENDIX
INPUT HELIX DATA

The file hlx_full.dat has 54 rows containing the data of 12 distinct amino acid sequences. Each row represents an atom. The 2nd, 3rd, and 4th columns are the *xyz* position of the spheres respectively. The 5th column is for the radius of each sphere. Contents of the file are below.

```
1   1     4.731 -1.827  0.561   2.9
1   1     5.689 -3.261  1.447   3.5
2   2     3.046  0.089  3.383   2.9
2   2     4.302  1.511  5.169   3.9
3   3     1.706  2.640  0.910   2.9
3   3     1.706  2.640  0.910   2.4
4   4     0.325 -0.141 -1.278   2.9
4   4     1.975 -2.373 -2.468   2.7
5   5    -1.381 -1.717  1.725   2.9
5   5    -0.800 -2.084  2.754   3.1
6   6    -2.934  1.622  2.652   2.9
6   6    -2.208  2.802  2.913   3.2
7   7    -4.214  2.061 -0.897   2.9
7   7    -3.782  2.091 -1.501   2.7
6   8    -5.742 -1.415 -0.822   2.9
6   8    -5.025 -2.603 -0.574   3.2
6   9    -7.457 -0.639  2.475   2.9
6   9    -6.792 -0.239  3.652   3.2
6  10    -8.860  2.583  1.038   2.9
6  10    -8.084  3.627  0.494   3.2
8  11   -10.183  0.708 -1.988   2.9
8  11    -8.981 -0.457 -3.325   3.2
6  12   -11.846 -1.852  0.270   2.9
6  12   -11.172 -2.637  1.228   3.2
7  13   -13.472  0.922  2.291   2.9
7  13   -13.100  1.448  2.662   2.7
3  14   -14.770  2.539 -0.890   2.9
3  14   -14.770  2.539 -0.890   2.4
9  15   -16.219 -0.778 -2.037   2.9
9  15   -15.282 -2.270 -2.261   3.3
10  16   -17.943 -1.235  1.315   2.9
10  16   -17.292 -1.256  2.547   3.0
11  17   -19.421  2.254  1.080   2.9
11  17   -18.808  3.220  0.963   2.7
12  18   -20.710  1.520 -2.416   2.9
12  18   -19.395  0.876 -4.189   3.7
3  19   -22.309 -1.701 -1.199   2.9
3  19   -22.309 -1.701 -1.199   2.4
```

```
<<continue>>
1  20   -23.992  0.160  1.652   2.9
1  20   -23.059  1.195  3.001   3.5
6  21   -25.334  2.753 -0.775   2.9
6  21   -24.538  3.541 -1.631   3.2
10  22   -26.711  0.012 -3.012   2.9
10  22   -25.950 -1.008 -3.579   3.0
7  23   -28.416 -1.620 -0.038   2.9
7  23   -28.063 -1.877  0.563   2.7
7  24   -29.973  1.700  0.948   2.9
7  24   -29.589  2.311  1.124   2.7
1  25   -31.254  2.200 -2.592   2.9
1  25   -30.138  2.233 -4.177   3.5
7  26   -32.777 -1.279 -2.579   2.9
7  26   -32.392 -1.908 -2.488   2.7
10  27   -34.494 -0.563  0.731   2.9
10  27   -33.837 -0.188  1.901   3.0
```

LIST OF REFERENCES

Chen, Z., and Y. Xu. 2005. Multi-scale hierarchical structure prediction of helical transmembrane proteins. *Computational Systems Bioinformatics Conference, 2005*. 203-207.

Fleishman, S. J., and N. Ben-Tal. 2002. A novel scoring function for predicting the conformations of tightly packed pairs of transmembrane α-helices. *J. Mol. Biol*. 321:363–378.

Javadpour, M. M., M. Eilers, M. Groesbeek, and S. O. Smith. 1999. Helix packing in polytopic membrane proteins: role of glycine in transmembrane helix association. *Biophysical Journal*. 77:1609-1618.

Pande, V. S., I. Baker, J. Chapman, S. P. Elmer, S. Khaliq, S. M. Larson, Y. M. Rhee, M. R. Shirts, C. D. Snow, E. J. Sorin, and B. Zagrovic. 2003. Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. *Biopolymers*. 68:91–109.

Shirts, M. R., and V. Pande. 2000. Screen savers of the world, unite! *Science*. 290:1903-1904.

Wendel, C., and H. Gohlke. 2008. Predicting transmembrane helix pair configurations with knowledge-based distance-dependent pair potentials. *Proteins*. 70:984-999.

Zagrovic, B., C. D. Snow, M. R. Shirts, and V. S. Pande. 2002. Simulation of folding of a small alpha-helical protein in atomistic detail using worldwide-distributed computing. *J. Mol. Biol*. 323:927–937.

BIOGRAPHICAL SKETCH

I received my Bachelor of Engineering degree in computer science and technology from Beihang University, one of the top schools in China. During my undergraduate study there, I have been selected as a visiting student to study at the University of Alberta in Canada for one year, where I learned computer graphics and developed an interest in related fields. After my graduate, I continued my study in computer engineering at the University of Florida as a master's candidate. Because of my excellent academic records there, I have been selected for Outstanding Achievement Awards twice.