Nanosuperficies
El mundo a escala molecular

Ángela Hidalgo Valverde
Víctor González García
Cloe Biedma López
Jorge Rueda de Gracia

Grupo Bioquímica II

29 de mayo de 2024

Proyectos Interdisciplinares en Ciencias
del Centro Mediterráneo (2ª ed.)

Facultad de Ciencias

Universidad de Granada

Índice

Resumen

En este proyecto se describe brevemente tres tipos de superficies de proteínas y tres posibles métodos para calcularlas. Seguidamente se detalla el algoritmo que hemos seguido para su obtención, la estructura de datos que definen y parametrizan dicha superficie y las ecuaciones conseguidas. Por último se presentan los resultados alcanzados y se discuten problemas encontrados en el proceso y posibles mejoras.

1. Introducción

Uno de los principales problemas en la biología estructural y la biocomputación es determinar cómo interacciona una proteína con otra molécula, ya sea una proteína (*docking*) o un ligando (*binding*). Una forma de estudiar estas interacciones es con la geometría de la proteína, pues su rugosidad, forma y huecos determinan los procesos que pueden ocurrir.

Así, podemos definir tres tipos de superficies:

1.1. Superficie de Van der Waals

Si consideramos los átomos de la proteína como esferas, en primera instancia podemos definir su superficie como la mera intersección y unión de esferas. Tomando el radio de Van der Vaals de cada átomo, esta superficie es una propiedad intrínseca de la proteína.

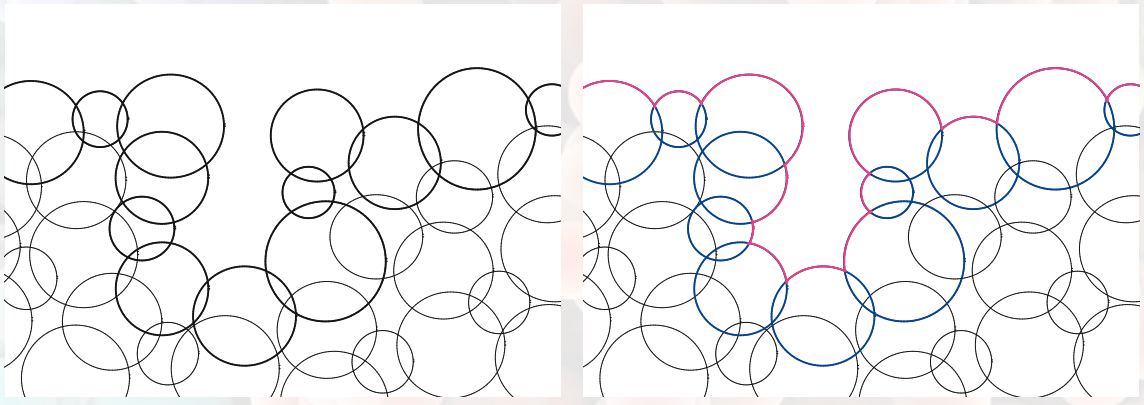

Figura 1: Comparación entre la proteína y su superficie de Van der Vaals

1.2. Superficie Accesible al Solvente (SAS)

Si bien la superficie de Van der Vaals es importante, puede ser más interesante estudiar la superficie como el resultado de la interacción entre la proteína y una molécula externa, a la que llamaremos solvente. El solvente recorre, se desliza por encima de la proteína, y la superficie será solo aquella que el solvente "ve.º pueda detectar. Así, si existe algún hueco o cavidad donde no puede entrar, esa zona deja de considerarse como parte de la superficie: si el solvente no lo ve, es que no existe.

Definimos la superficie accesible al solvente (SAS) como la superficie marcada por el centro de un solvente esférico que recorre la superficie de Van der Vaals. Ahora la SAS no es una propiedad intrínseca de la proteína, sino que depende del radio considerado para el solvente. Por tanto, un solvente menor detectará una SAS más rugosa, mientras que un solvente mayor detectará una SAS más lisa, habiendo eliminado las zonas que no son relevantes para él.

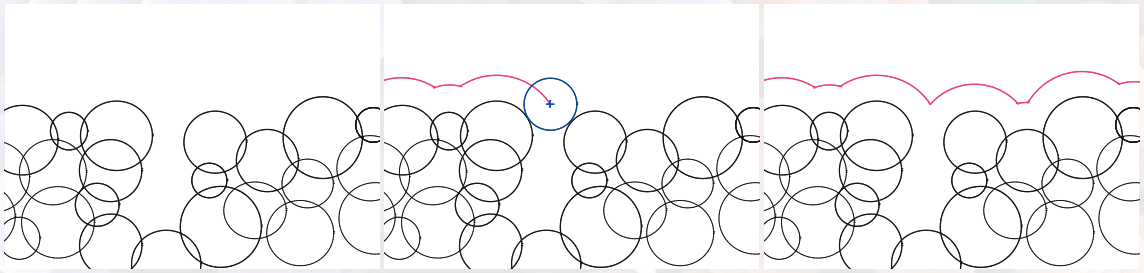

Figura 2: Proceso de obtención de SAS

Esta geometría aporta información sobre qué procesos de binding o docking podrían ocurrir y se puede estudiar las diferencias para varias moléculas de interacción.

1.3. Superficie Excluyente al Solvente (SES)

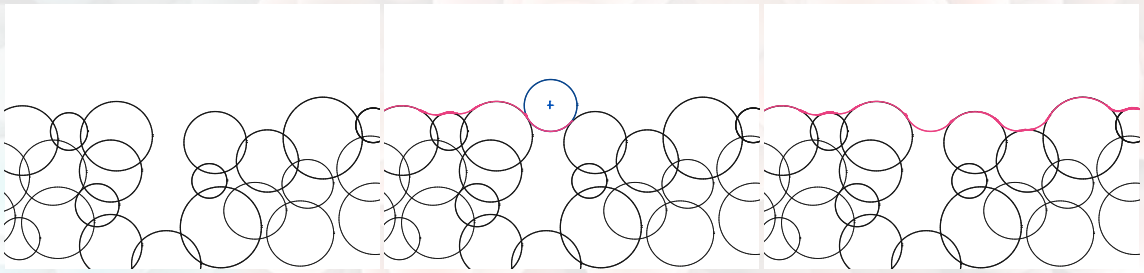

Figura 3: Proceso de obtención de SES

Siguiendo el mismo análisis que en la superficie anterior, definimos la superficie de Connolly o SES como la superficie que marca el final de la esfera del solvente al recorrer la superficie de Van der Vaals de la proteína. Así, la SES estará formada por zonas de contacto (trozos de la propia superficie de Van der Vaals) y zonas reeentrantes (donde el solvente no "toca" la proteína pero tampoco puede acercarse más).

Trozos de SES se pueden clasificar como:

- parches esféricos, al igual que las dos superficies anteriores.
- Toroides y "sillas de montar".
- Triángulos no geodésicos.

Las superficies VdW y SAS solo cuentan con parches esféricos.

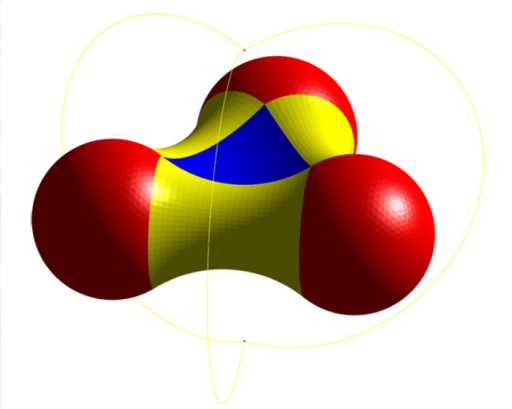

Figura 4: Ejemplo de SES de 3 átomos: parches esféricos (en rojo), toroides (en amarillo) y triángulos no geoésicos (en azul).

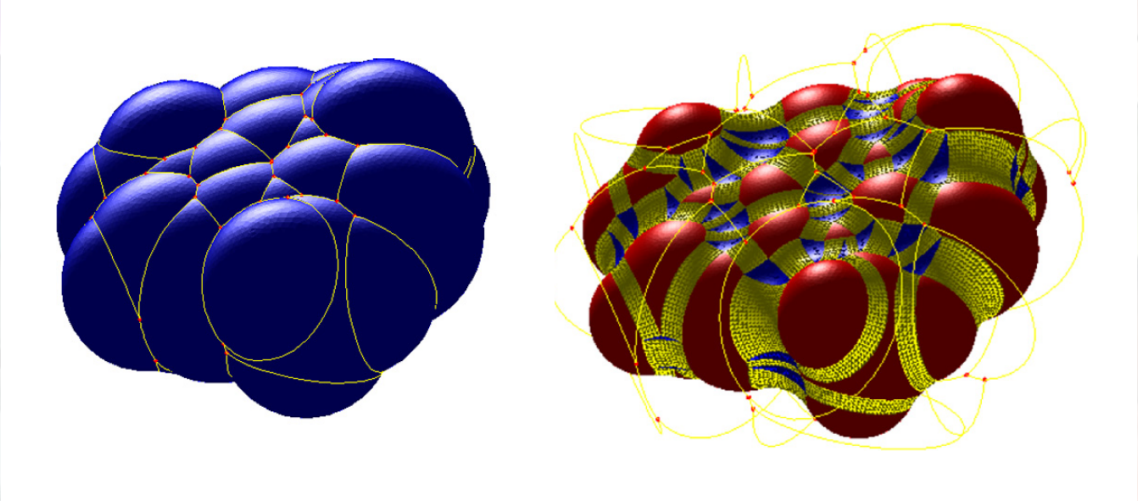

Figura 5: Comparación de SAS (izquierda) y SES (derecha).

El objetivo de este proyecto es construir un algoritmo capaz de parametrizar y definir con exactitud las diferentes superficies, calcular el área total de la proteína y construir una APP que pueda ser usada por un público general, donde el usuario solo tenga que seleccionar la proteína y obtenga la superficie calculada.

2. Metodología

2.1. Método numérico

Se secciona la proteína a lo largo de un eje, obteniendo planos de cierto grosor. Esto proporciona un figura plana (que no tiene por qué ser conexa) que se puede dibujar como la unión de varios círculos. Para cada sección se superpone una cuadrícula y se cuenta el número de celdas que están sobre la frontera, obteniendo una medida aproximada de la longitud de esta. El valor obtenido se multiplica por el grosor considerado, obteniendo una aproximación del área que aporta dicha sección a la proteína. Se repite para todas las secciones.

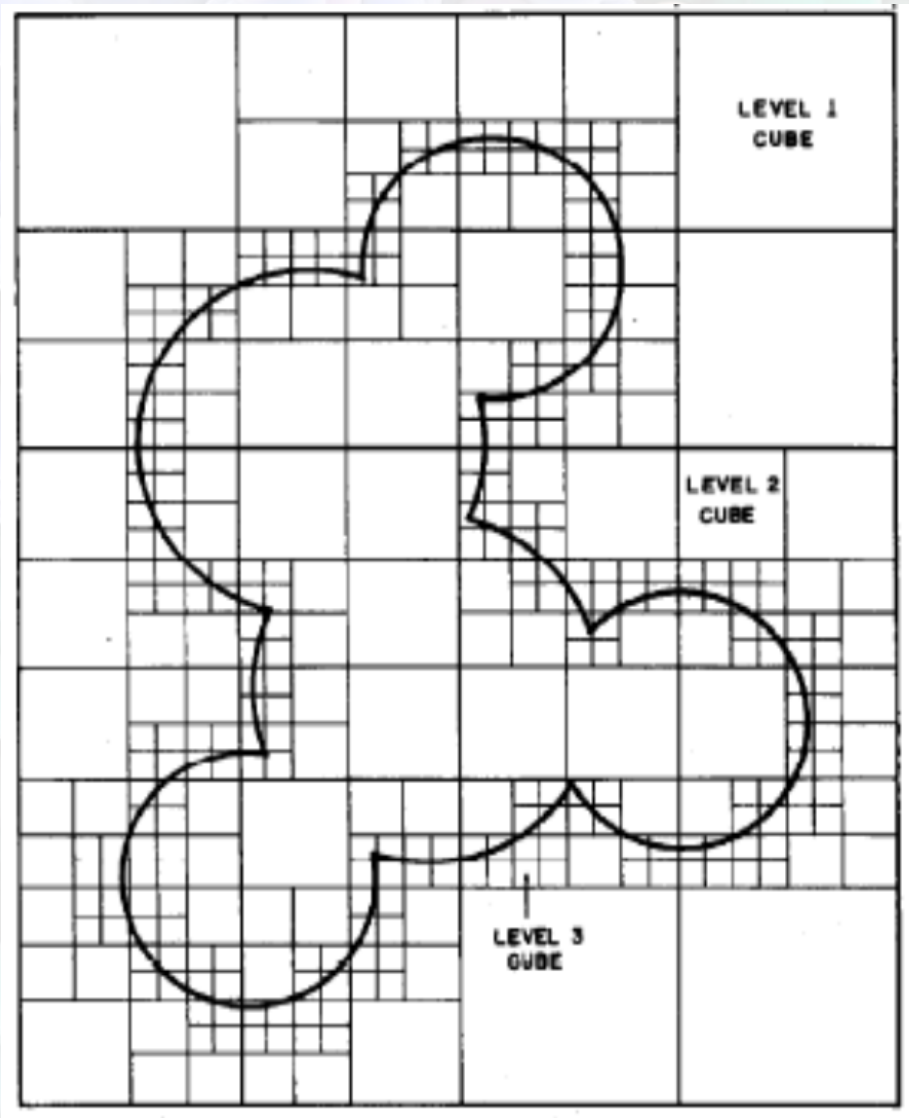

Figura 6: Resultado ejemplo del método numérico

Este método es más sencillo, en principio, de implementar que los otros dos; sin embargo la precisión de este recae en el grosor considerado y cómo de ajustada sea la cuadrícula. Este algoritmo se logró implementar el año pasado y se detectó una fuerte dependencia con el ángulo de orientación de la proteína, puesto que al seccionar la proteína las esferas se aproximan a varios cilindros orientados.

2.2. Método semianalítico

Se secciona la proteína como en el método anterior y sobre esa intersección plana se calculan los arcos que conforman la frontera; estudiando la intersección de esos círculos de forma analítica. Esta vez calculamos la longitud de cada uno de los arcos para obtener la medida exacta de la longitud de la frontera. De nuevo, multiplicando la longitud que hemos obtenido en cada plano por el grosor considerado se obtiene la medida aproximada del área total de la proteína. Este método también depende de la orientación escogida.

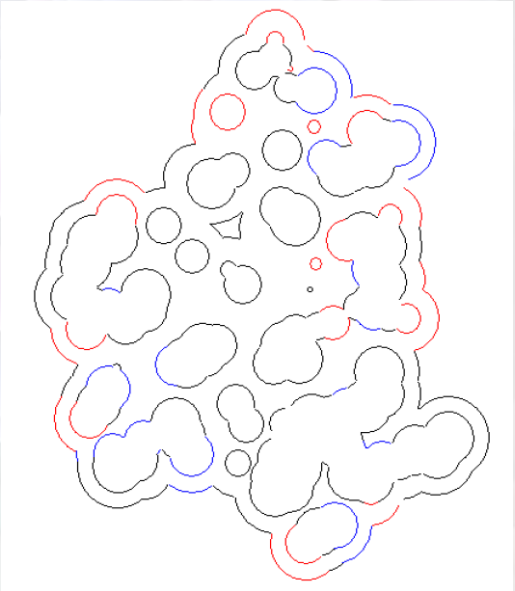

Figura 7: Sección de una proteína, formada por arcos y circunferencias.

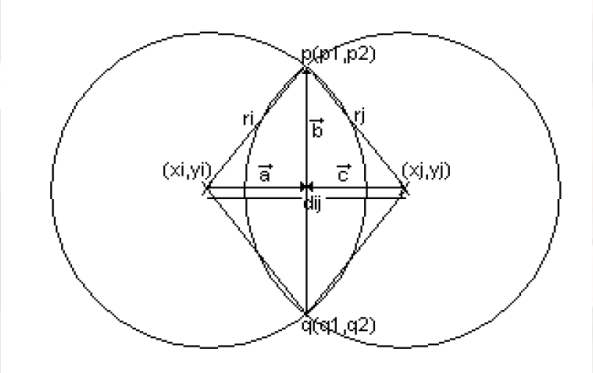

Figura 8: Representación geométrica de dos círculos que se cortan entre sí, a partir de la cual se pueden obtener las ecuaciones de puntos de corte y arcos externos.

2.3. Método analítico

Este método supone un cambio conceptual respecto de los otros dos. En este pasamos a realizar un cálculo tridimensional estudiando las intersecciones de -ya no círculos- sino de esferas en el espacio. La idea del método es ir esfera por esfera sumando el área "visible" (área de regiones que no se encuentran en el interior de ninguna otra) sobre esta. Para ello es primordial conocer cuáles son las regiones

visibles de la esfera. Nuestro trabajo se ha centrado en intentar calcular estas regiones esféricas puesto que en el trabajo en el que nos hemos basado (Connolly[1][2], Quan-Stamm[3][4]) no vienen detallados estos algoritmos. Suponiendo que se conocen estas regiones visibles, se separan en componentes conexas mediante un algoritmo de árbol binario que las clasifica. En cada una de estas regiones conexas (que en el artículo de Quan se denominan *parches*) se aplica el Teorema de Gauss-Bonnet para calcular su área. Repitiendo esto para cada parche y para cada esfera se obtiene el área exacta de la proteína.

Primeramente, pretendíamos seguir el enfoque semianalítico. Sin embargo consideramos que por un poco más de esfuerzo podíamos tener el método analítico, que se supone más preciso.

3. Resultados y Discusión

Nuestro trabajo ha consistido en la comprensión y aplicación del método analítico para el cálculo de la superficie de Van der Waals y superficie accesible al solvente, a partir de los artículos ya mencionados ([1],[2],[3],[4]) y de una biblioteca en C++ proporcionada por los tutores. Hemos creado un algoritmo que calcula y almacena toda la información requerida para definir las superficies, hemos ampliado la biblioteca inicial para incluir las estructuras (clases) que hemos considerado relevantes para el cálculo, y hemos conseguido las ecuaciones necesarias para cada paso del algoritmo.

A continuación explicamos con más detalle el proyecto conseguido y los resultados del mismo.

3.1. Entorno de trabajo

Este trabajo ha sido construido en el entorno C++ Builder[7]; lo elegimos por su versatilidad para crear una App sencilla, que pueda ser usada por un público general a modo de calculadora de superficies. El usuario selecciona un archivo PDB y la App le proporciona un texto con cierta información sobre la proteína. Además, se despliega un menú con varios botones, para acceder a información sobre temperatura, átomos...etc.

3.2. Algoritmo seguido

El algoritmo que hemos implementado realiza los siguientes pasos:

1. Lectura del archivo PDB[6] y guardado de las coordenadas y radio de los áto-

mos (*parsing*). Estos archivos son bibliotecas que contienen toda la información relevante de una proteína, sus átomos, residuos (aminoácidos incompletos que esta´n unidos entre sí dentro de la proteína) y subunidades.

2. En caso del cálculo de la superficie accesible al solvente (SAS), se le suma el radio del solvente a los radios de los átomos. Este paso no se hace si se está calculando la superficie de van der Vaals.

3. Ordenado de los átomos según el eje de coordenadas que más .[a]lineado.[es]té con la proteína. Para ello se buscan los valores máximos y mínimos de las coordenadas de los átomos. El eje con mayor rango será el escogido para la ordenación.

4. Selección del átomo i (según el orden anterior). Se toma un subconjunto de átomos $\{j\}$ "cercanos". Por ejemplo, si se ha ordenado en el eje X:

$$\mid c_{jx} - c_{ix} \mid < r_{max}$$

siendo $\vec{c}_i = (c_{ix}, c_{iy}, c_{iz})$ el centro del átomo i, $\vec{c}_j = (c_{jx}, c_{jy}, c_{jz})$ el centro del átomo j y r_{max} el radio máximo que pueda tener un átomo de la proteína. Esto nos asegura que cualquier átomo que no pertenezca a $\{j\}$ no puede cortar a i. Hemos tomado $r_{max} = 5$ Å.

5. Recorre el subconjunto $\{j\}$ y encuentra cuáles cortan a i, y calcula el correspondiente círculo de corte. Conseguimos así un conjunto de círculos del átomo i.

6. Para los círculos del átomo i, se calculan los puntos de corte entre círculos y los arcos resultantes de estos cortes.

7. Se clasifican los arcos en *loops*: curva cerrada formada por la unión de varios arcos. Se calculan también los ángulos α_s y la longitud e_s de cada arco que constituye el loop.

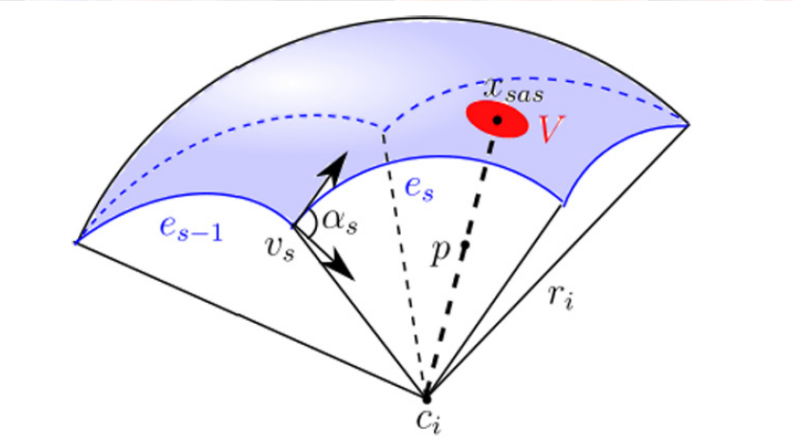

Figura 9: Región de la esfera (átomo i) que delimita un loop. El loop está formado por un conjunto de vértices $\{v_s\}$ y arcos $\{e_s\}$, que forman ángulos $\{\alpha_s\}$ entre sí.

8. Una vez conseguido un conjunto de loops que pertenecen al átomo i, se clasifican en subconjuntos llamados *fronteras*. Cada frontera delimita o define un *parche esférico*, que será un trozo de área de la esfera que aporta a la superficie de la proteína. La suma del área de los parches será la aportación del átomo i al área de la proteína.

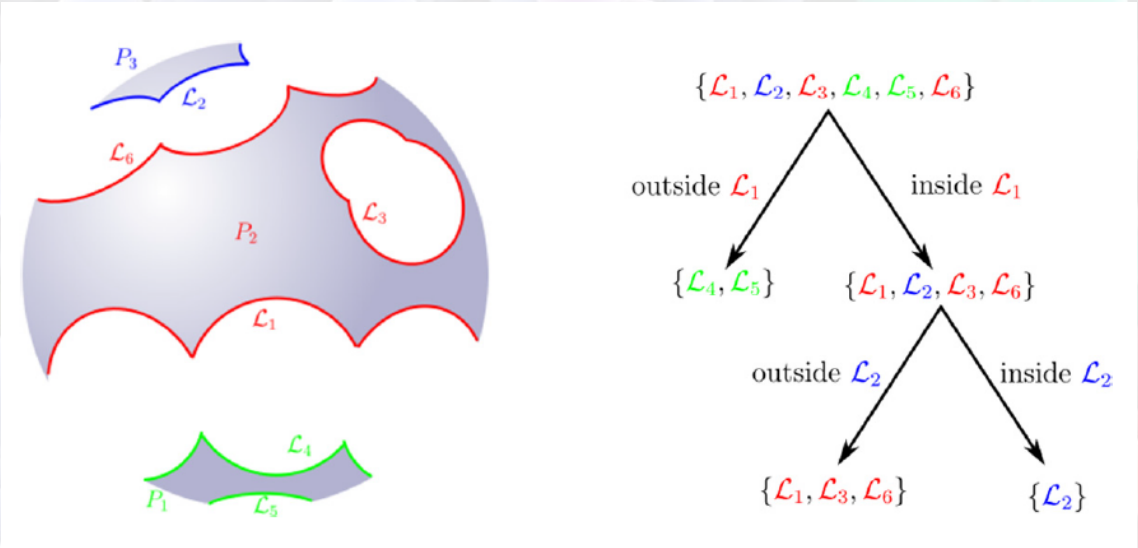

Figura 10: Clasificación de loops según un Binary Tree, consiguiendo subconjuntos que definen parches esféricos.

9. Se vuelve al paso 4, repitiendo para el siguiente átomo i, hasta recorrer la proteína por completo.
10. Si se había seguido el paso 2, devolvemos los radios originales de los átomos de la proteína.

3.2.1. Biblioteca C++

Partimos de una biblioteca inicial en C++, proporcionada por los tutores de este proyecto. Los elementos que usamos son principalmente:

Clase TPDB

- Vector de TAtomPDB.
- Número de elementos del vector.
- función de parsing.

Clase TAtomPDB

- Posición del centro del átomo
- Radio del átomo

Hemos añadido varias clases que almacenen la información descrita en el apartado 3.2, además de un método de ordenación de los átomos de la proteína. Estas son:

Clase TCircle

- Posición del centro del círculo
- Vector normal al plano que contiene el círculo
- Radio del círculo

Clase TArc

- Punto inicial
- Punto final
- Radianes del arco
- Etiqueta del círculo al que pertenece el arco

Clase Tloop

- Vector de TArc
- Número de elementos del vector

Clase TPatch

- Vector de Tloop
- Número de elementos del vector

Así, una proteína (TPDB) tendrá un vector de átomos (TAtomPDB); cada átomo tendrá un vector de parches o fronteras (TPatch); cada frontera tendrá un vector de loops (TLoop); cada loop tendrá un vector de arcos (TArc); y cada arco pertenecerá a un círculo (TCircle) del átomo.

Asimismo hemos incluido funciones que calculan y rellenan toda esta estructura de datos, para los cuales también hemos conseguido las ecuaciones necesarias.

3.2.2. Cálculo analítico

Consideramos cada átomo i como esferas situadas en un espacio tridimensional, conociendo las coordenadas (X, Y, Z) de su centro $\vec{c}_i$ y el radio r_i.

Círculo Cuando dos esferas i y j se cortan entre sí, forman un círculo sobre la superficie de ambas (análogo a que un plano corte una esfera). Con la información de las esferas podemos conseguimos las ecuaciones para el círculo:

- El vector normal al plano que contiene al círculo $\hat{d}$

$$\hat{d} = \frac{\vec{d}}{d} \qquad \vec{d} = \vec{c}_j - \vec{c}_i \tag{1}$$

  siendo $d \equiv |\vec{d}|$ (o $d^2 \equiv \vec{d} \cdot \vec{d}$).

- Las coordenadas del centro del círculo $\vec{H}$

$$\vec{H} = \vec{c}_i + t\frac{\vec{d}}{d} \qquad t = \frac{r_i^2 - r_j^2 + d^2}{2d} \tag{2}$$

- El radio del círculo s

$$s = \sqrt{r_i^2 - t^2} \tag{3}$$

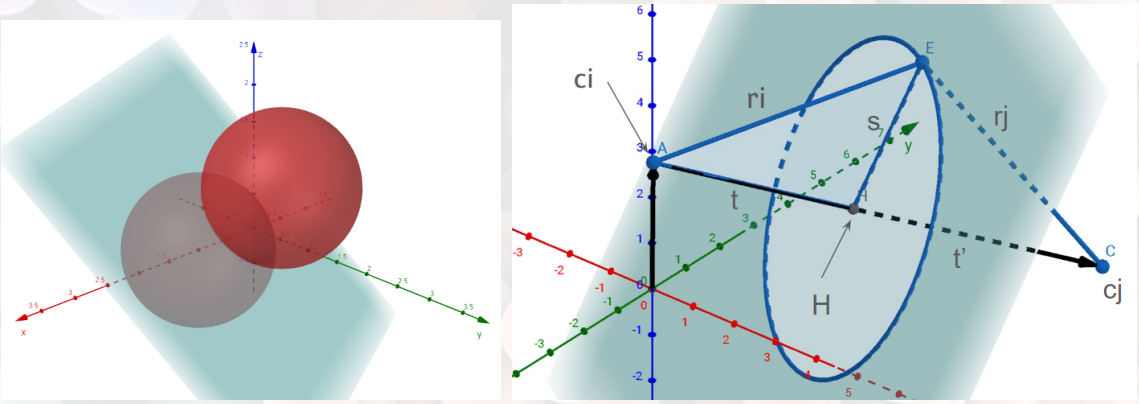

Figura 11: Círculo resultante de la intersección entre dos esferas

Puntos de corte entre círculos El procedimiento de cálculo parte de que se conocen el centro y los radios de las circunferencias y el normal del plano donde están contenidas. Las circunferencias que no son tangentes se cortan en dos puntos que determinan de manera unívoca una recta. La recta debe tener vector director perpendicular a los normales.

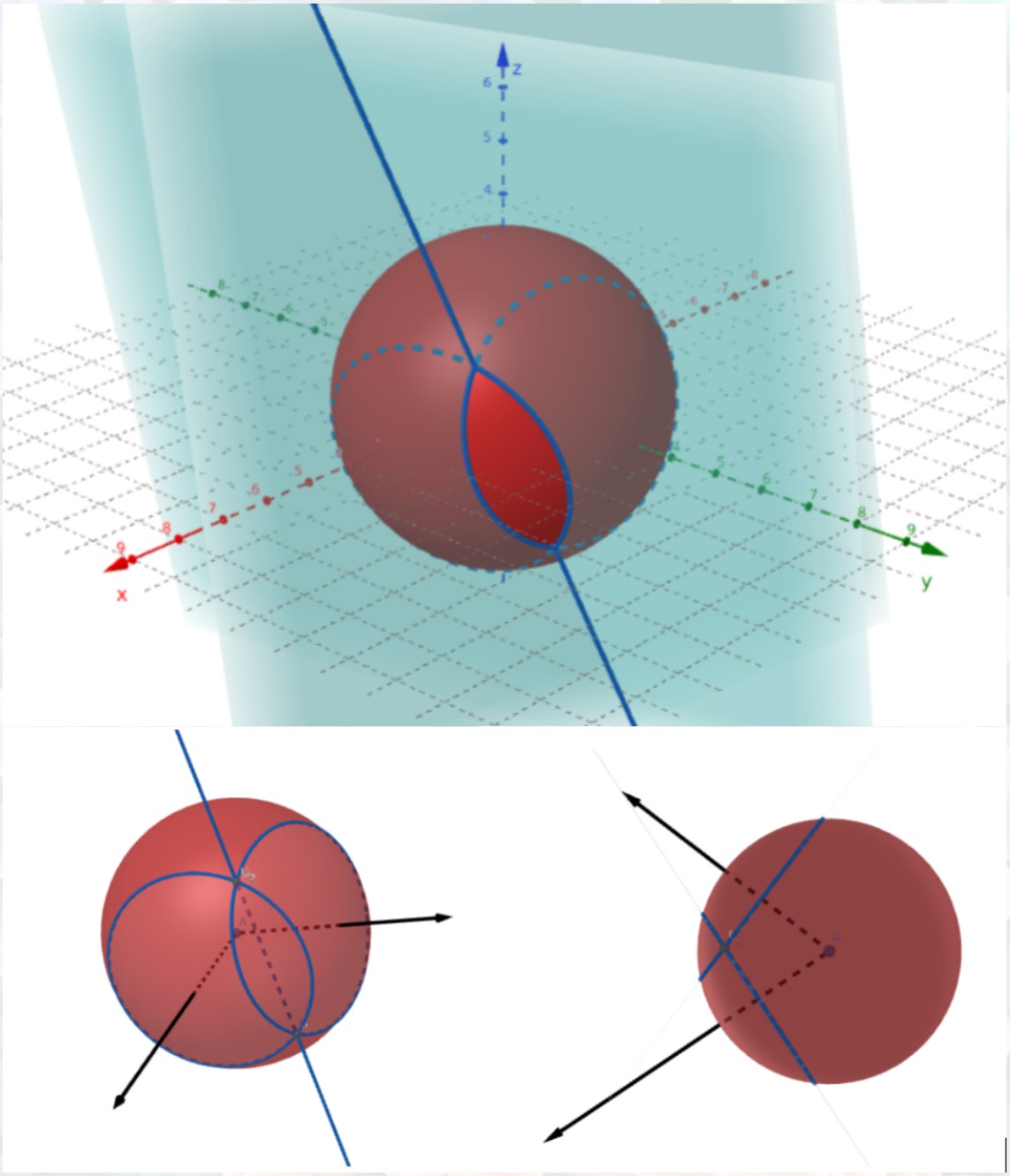

Figura 12: Recta resultante del corte entre dos círculos (inscritos en la misma esfera)

Para obtener la recta necesitamos un punto base. Para ello imponemos que alguna de las coordenadas x, y ó z sea 0 y resolvemos para el que sea posible. Obtenemos por tanto la parametrización de la recta e imponemos que la distancia del punto sobre la recta hasta C sea r, o equivalentemente:

$$\langle P + vt, P + vt \rangle = r^2 \tag{4}$$

Resolviendo la ecuación (4) (usando que $|| v || = 1$) tenemos que:

$$t_{\pm} = -\langle v, P_C \rangle \pm \sqrt{\langle v, P_C \rangle \text{`}2 - (|| P - C ||^2 - r^2)} \tag{5}$$

y por tanto los puntos que buscamos son:

$$Q_{\pm} = P + vt_{\pm} \tag{6}$$

Arcos El siguiente paso es limpiar la parte de los círculos que no cuenta para el loop, como en la figura 13.

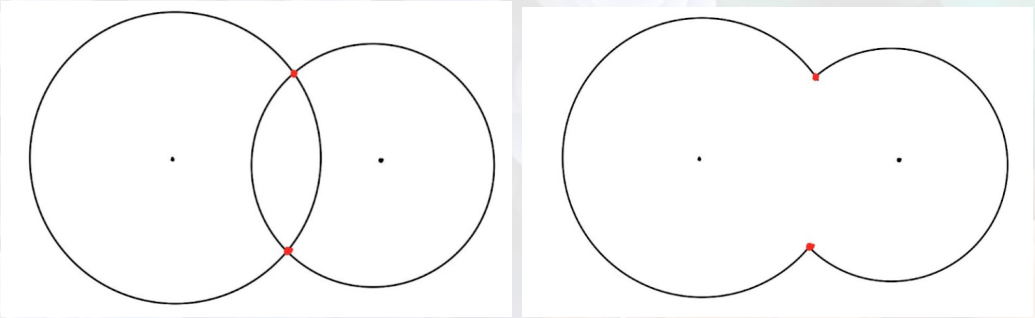

Figura 13: Puntos de corte entre dos círculos. Selección de arcos que forman un loop

Nuestra idea para representar los arcos era guardar el centro de la circunferencia donde estaba contenido el arco, el radio, el vector normal a la circunferencia y los puntos de intersección con el convenio de escribirlos en sentido positivo, es decir, recorrer la circunferencia en sentido antihorario e ir guardando los puntos de intersección. Tenemos varias propuestas pero ninguna está acabada.

Área de un parche esférico se hace uso de la fórmula de Gauss-Bonnet[1] para calcular el área de casquetes esféricos A_P:

$$\sum_s \alpha_s + \sum_s k_{e_s} \mid e_s \mid + \frac{1}{r_i^2} A_p = 2\pi\chi \tag{7}$$

siendo α_s el ángulo que forman dos arcos que se cortan; k_{e_s} la curvatura geodésica del arco e_s; $\mid e_s \mid$ la longitud del arco e_s; χ la característica de Euler (como estamos sobre superficies esféricas, χ vale 2 menos el número de loops que tenga la frontera del parche). En la figura 9 se puede apreciar geométricamente algunos de estos parámetros.

Toda esta información debería estar debidamente calculada y almacenada, si bien solo hemos conseguido calcular los vértices v_s.

3.3. Resultados obtenidos

Si bien no hemos conseguido seguir con la parametrización de arcos y loops, sí que se han calculado los círculos de todos los átomos de una proteína.

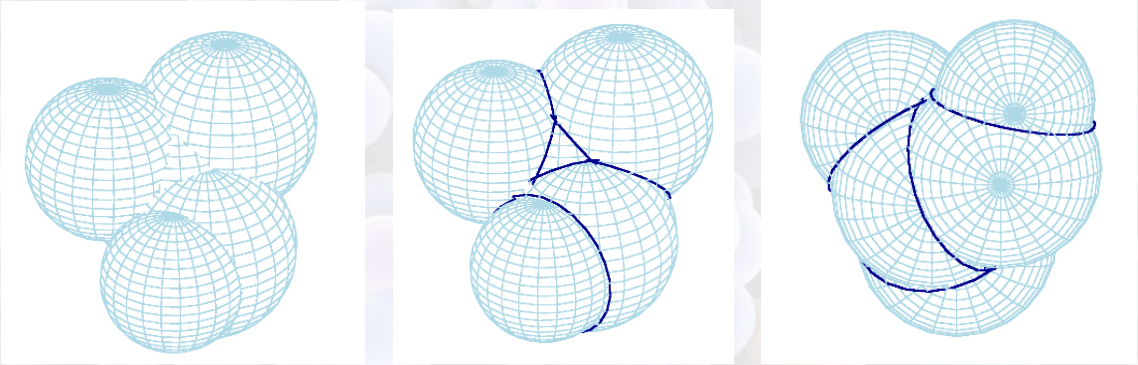

Figura 14: Proteína ejemplo de 5 átomos (izquierda). Círculos calculados por nuestro algoritmo sobre la proteína (medio y derecha).

En la figura 15 (izquierda) se pueden observar los círculos calculados de la proteína '1FDS.PDB', de 2152 átomos. El algoritmo consigue 7 345 círculos. Para mayor claridad, también se han representado los círculos 200-300 proyectados en el plano XY (figura 15, derecha).

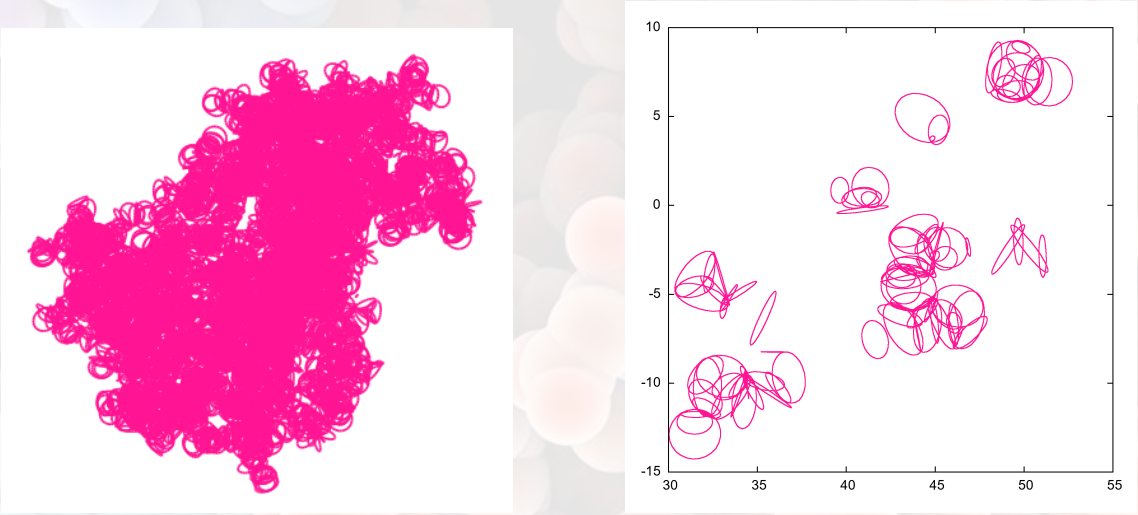

Figura 15: Círculos calculados de la proteína '1FDS.PDB'. Selección de 100 círculos, proyectados en el plano XY

Podemos considerar que los resultados de los círculos son satisfactorios.

3.3.1. Problemas encontrados

Uno de los primeros problemas que tuvimos fue entender la estructura de datos de la biblioteca inicial, puesto que ninguno de los integrantes teníamos mucha experiencia con clases y objetos de C++. Al intentar enfrentarnos a la primera parte del algoritmo, la ordenación de los átomos, no sabíamos si esto podría afectar al etiquetado, puesto que no todos los átomos que aparecen en los archivos PDB[6] pertenecen a la proteína. Esto lo solucionamos creando un vector *AtomIndex*: ordenamos este vector que enumera a los átomos en vez de ordenar los átomos en sí, evitando perder información relevante.

Al calcular y almacenar la información de los círculos, el vector normal $\hat{d}$ tiene una dirección y sentido. Si consideramos que el mismo círculo pertenece al átomo i y al átomo j, $\hat{d}$ sale de uno de ellos y entra hacia el otro. Esto presentaba problemas en el cálculo de la ecuación de la recta (4) (el vector director v es el producto vectorial de $\hat{d}$ y $\hat{d}'$ de dos círculos diferentes). Decidimos almacenar el mismo círculo como dos TCircle (uno para el átomo i y otro para el átomo j), estableciendo el convenio de que $\hat{d}$ siempre sale hacia afuera de la esfera.

En adición, la clasificación por Binary Tree de los loops (obteniendo fronteras) nos llevó bastante tiempo de comprender, puesto que no teníamos claro cuál era el criterio para considerar qué es "dentro"de un loop y qué es "fuera". Tras varios días de análisis conseguimos llegar a un convenio, si bien no hemos podido añadirlo al programa por falta de tiempo.

El mayor problema encontrado ha sido la parametrización de los arcos, pues finalmente no hemos conseguido seleccionar qué arcos son válidos y cuáles no. Nuestro programa queda incompleto.

3.3.2. Posibles mejoras propuestas

La ordenación de átomos se realiza en una dimensión, según el eje de coordenadas que esté más alineado con la proteína. Es posible ordenarlos en las tres dimensiones, consiguiendo regiones cúbicas en vez de capas, que minimicen los subconjuntos $\{j\}$ para cada i. Esto parece mucho más difícil de implementar (y por ello elegimos una ordenación en una dimensión) pero puede resultar interesante y disminuir el tiempo de ejecución.

Otra posible mejora sería un único guardado de TCircle, puesto que actualmente tenemos la información repetida y puede ser costoso en memoria.

Asimismo, el algoritmo sólo calcula el área de parches esféricos, por lo que quedaría incompleto para el cálculo de la superficie excluyente al solvente (superficie de Connolly). Se podría ampliar el algoritmo para calcular toroides y triángulos no geodésicos.

4. Conclusiones

A pesar del estado incompleto del proyecto, la realización de este nos ha aportado una mayor compresión en bioquímica, biología molecular, topología y geometría, así como un aprendizaje en programación, trabajo en equipo interdisciplinar y comunicación.

Si bien nuestro avance ha llegado hasta los círculos y puntos de corte y no conseguimos parametrizar los arcos, los pasos restantes son pocos y más sencillos, por lo que con más tiempo se alcanzarían los objetivos marcados.

Nuestra participación en esta actividad del Centro Mediterráneo ha sido enriquecedora y agradecemos a los tutores Fernando Reyes Zurita, Aureliano M. Robles Pérez, Hilario Ramírez Rodrigo, Margarita Arias López y María José Sáez Lara por su colaboración.

Referencias

[1] Michael L. Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16(5):548–558, Oct 1983.

[2] Michael L. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221(4612):709–713, 1983.

[3] Chaoyu Quan and Benjamin Stamm. Mathematical analysis and calculation of molecular surfaces. *Journal of Computational Physics*, 322:760–782, 2016.

[4] Chaoyu Quan and Benjamin Stamm. Meshing molecular surfaces based on analytical implicit representation. *Journal of Molecular Graphics and Modelling*, 71:200–210, 2017.

[5] Peter Van Weert Ivor Horton. *Beginning C++17, from Novice to Professional*. Apress, fifth edition edition, 2018.

[6] Protein Data bank. https://www.wwpdb.org/.

[7] Embarcadero Cross-Platform App Development Software C++ Builder. https://www.embarcadero.com/es/.

[8] Jmol: an open-source Java viewer for chemical structures in 3D. https://jmol.sourceforge.net/.