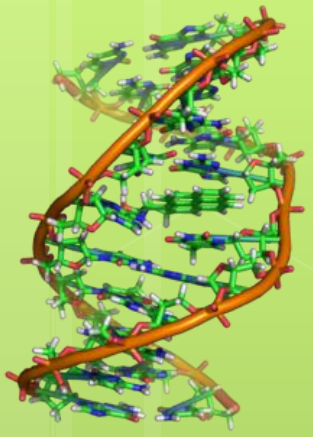
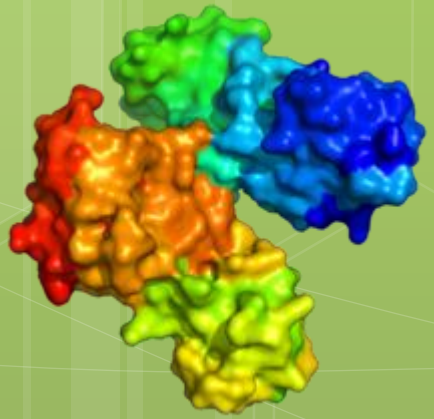




α-shapes στη βιολογία

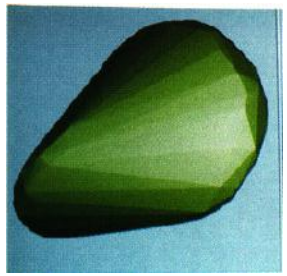
Λογισμικό CGAL για α-shapes

Νικόλας Μπεγέτης

Πώς παίρνουμε το περίβλημα σημείων στο χώρο;

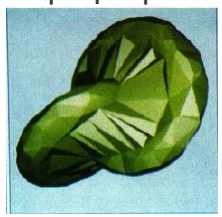
- Βρίσκοντας το Convex Hull
- π. χ:



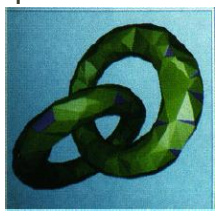
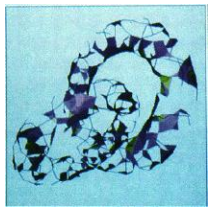


- Θα μπορούσαμε να βρούμε μία καλύτερη προσέγγιση;
- Ναι, με α-shapes…
- π. χ:





ή

Κρέμα παγωτό με λαχταριστά κομμάτια σοκολάτας (παράδειγμα)

- Έστω ένα παγωτό με κομματάκια σοκολάτα. Θέλουμε να αφαιρέσουμε με ένα κουτάλι παγωτού όσο παγωτό μπορούμε χωρίς να ακουμπήσουμε τα σοκολατάκια. Ακόμα και να αφαιρέσουμε τις εσωτερικές τρύπες (μέρη που δεν τα φτάνουμε με το κουτάλι), θα καταλήξουμε σε ένα σχήμα με όρια : κορυφές τόξα και σκέτα σημεία.
  - Αν τώρα αυτά τα όρια τα βάλουμε σε ευθ. τμήματα και τρίγωνα έχουμε τα α-σχήματα

- Όπως είναι προφανές το τελικό σχήμα θα εξαρτηθεί από το μέγεθος του κουταλιού
- Άρα για κουτάλι με μικρή διάμετρο (μικρή παράμετρος α) θα μπορέσουμε να φάμε σχεδόν όλο το παγωτό και να μείνουν μόνο τα σοκολατάκια.
- Αντίστοιχα για κουτάλι με μεγάλη διάμετρο (μεγαλύτερη παράμετρο α) δεν θα χωράει το κουτάλι μεταξύ κάποιων κομματιών και έτσι το παγωτό ανάμεσα σε αυτά τα σημεία δεν θα το φάμε ποτέ.
- Αν η διάμετρο του κουταλιού (πολύ μεγάλη παράμετρος α) τείνει στο άπειρο τότε δεν θα φάμε καθόλου παγωτό.

Αντιστοίχιση παραδείγματος με παγωτό στη CGAL

=

μικρή παράμετρο α

=

μεγάλη παράμετρος α

- Οπότε η παράμετρος α καθορίζει τη διάμετρο κουταλιού

Τριγωνοποιήσεις CGAL σε α-shapes

- Η CGAL χρησιμοποιεί
  - Delaunay triangulation
  - Regular triangulation
- Σε βασικά α-σχήματα:
  - Delaunay triangulation
- Σε ζυγισμένα α-σχήματα:
  - Regular triangulation

Βιβλιοθήκη CGAL για α-shapes

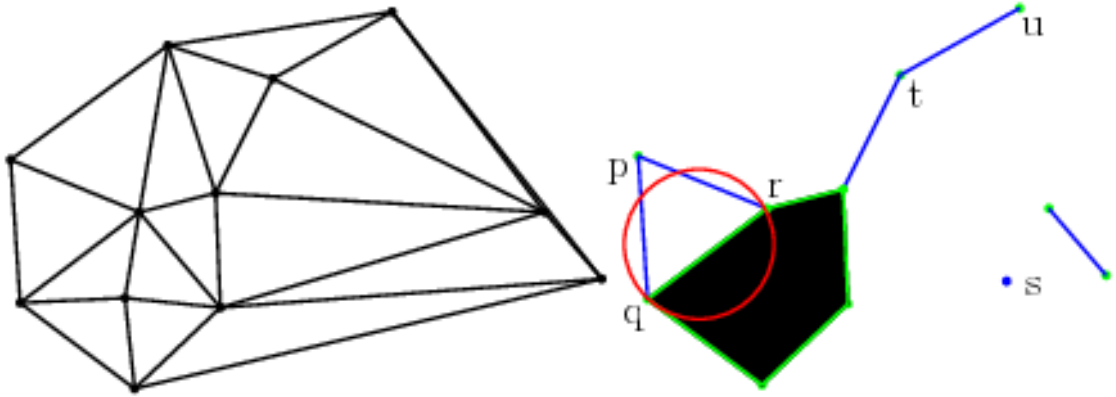
- `CGAL::Alpha_shape_3<Dt,ExactAlphaComparisonTag>`
  - Γενική βιβλιοθήκη της CGAL για τα α-shapes
- `CGAL::Fixed_alpha_shape_3<Dt>`
  - υποσύνολο που δίνονται ως δεδομένα εξ αρχής τα σημεία

    - `Dt` = η τριγωνοποιήση που χρησιμοποιείται κάθε φορά
    - `ExactAlphaComparisonTag` = true/false ανάλογα αν η όψη ανήκει στο α-σύμπλεγμα σημείων

CGAL::Alpha_shape_3<Dt,ExactAlphaComparisonTag>

- Παρέχει το `CGAL::object` σαν τύπο με το οποίο διατάσσει τα στοιχεία του α-συμπλέγματος με τη σειρά που μπαίνουν στο σύμπλεγμα ενόσω μεγαλώνει το α.
- Επίσης παρέχει συνάρτηση για να προσδιορίζει το μικρότερο α που:
  - 1. υπάρχει έστω και μία ζεύξη,
  - 2. τα συστατικά είναι λιγότερα ή ίσα με τον αριθμό των σημείων

Παράδειγμα - Σχήμα



- Αριστερά: μία τριγωνοποίηση Delaunay για σύνολο σημείων στο δυσδιάστατο χώρο.
- Δεξιά: Κατάταξη με χρωματική διοαφοροποίηση των απλόκων με βάση μία δεδομένη παράμετρο α που ισούται με το τετράγωνο της ακτίνας του κόκκινου κύκλου.

Υλοποίηση

- Ακολουθεί ο κώδικας για αναπαράσταση τρισδιάστατου μορίου πρωτεΐνης με α-shapes, με σκοπό την εύρεση της κατάλληλης α παραμέτρου και των απλόκων που δημιουργούνται.
  - τετραέδρα (τριγωνικές πυραμίδες) – όψεις (έδρες) και ακμές
- Στη συνέχεια ακολουθεί και διάγραμμα κατανομής των απλόκων με βάση την αλλαγή της παραμέτρου α

Κώδικας για αναπαράσταση πρωτεΐνης, εύρεσης της κατάλληλης α παραμέτρου (αρχείο: a-shape_protein.cpp)

```
1.  /*****************************************************************************************************************************
2.  * University/Faculty          : National Kapodistrian University of Athens - Department of Informatics and Telecommunications
3.  * Course - Professor          : Computational Geomatry - Emiris Ioannis
4.  * File name                                  : a-shape_protein.cpp
5.  * Name/Surname                              : Nikolaos Mpegetis
6.  * A.M                                       : 1115200700281
7.  *****************************************************************************************************************************/

8.  #include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
9.  #include <CGAL/Delaunay_triangulation_3.h>                                   //used for simple points
10. #include <CGAL/Regular_triangulation_euclidean_traits_3.h>    //used for weighted points
11. #include <CGAL/Regular_triangulation_3.h>

12. #include <CGAL/Alpha_shape_3.h>

13. #include <fstream>
14. #include <list>
15. #include <cassert>
16. #include <time.h>


17. struct point{
18.         double* coords;                 //coordinates. matrix that in index=0 holds x coordinate, in index=1 holds y coords...etc.
19. };

20. struct points2{
21.         point* input_points;
22.         int numoflines;
23.         int dimensions;
24. };
```

Κώδικας για αναπαράσταση πρωτεΐνης, εύρεσης της κατάλληλης α παραμέτρου(συνέχεια) (αρχείο: a-shape_protein.cpp)

```
25. typedef CGAL::Exact_predicates_inexact_constructions_kernel K;

26. typedef CGAL::Regular_triangulation_euclidean_traits_3<K> Gt;

27. typedef CGAL::Alpha_shape_vertex_base_3<K>          Vb;
28. typedef CGAL::Alpha_shape_cell_base_3<K>           Fb;
29. typedef CGAL::Triangulation_data_structure_3<Vb,Fb>  Tds;

30. typedef CGAL::Alpha_shape_vertex_base_3<Gt>         rVb;
31. typedef CGAL::Alpha_shape_cell_base_3<Gt>          rFb;
32. typedef CGAL::Triangulation_data_structure_3<rVb,rFb> rTds;

33. typedef CGAL::Delaunay_triangulation_3<K,Tds>       Del_Triangulation_3;
34. typedef CGAL::Regular_triangulation_3<Gt,rTds>      Reg_Triangulation_3;
35. typedef CGAL::Alpha_shape_3<Del_Triangulation_3>         Del_Alpha_shape_3;
36. typedef CGAL::Alpha_shape_3<Reg_Triangulation_3>         Reg_Alpha_shape_3;

37. typedef K::Point_3                          Point;
38. typedef Del_Alpha_shape_3::Alpha_iterator           Alpha_iterator;

39. typedef Reg_Alpha_shape_3::Cell_handle          Cell_handle;
40. typedef Reg_Alpha_shape_3::Vertex_handle        Vertex_handle;
41. typedef Reg_Alpha_shape_3::Facet               Facet;
42. typedef Reg_Alpha_shape_3::Edge                Edge;
43. typedef Gt::Weighted_point                 Weighted_point;
44. typedef Gt::Bare_point                   Bare_point;


45. using namespace std;
```

Κώδικας για αναπαράσταση πρωτεΐνης, εύρεσης της κατάλληλης α παραμέτρου(συνέχεια) (αρχείο: a-shape_protein.cpp)

```
46.  void ReadFile (points2 *pts){
47.
48.          ifstream input_file("input3d_protein.txt");                    // Open the file input3d_protein.txt for reading
49.
50.          if (input_file.fail())
51.                    exit(1);
52.
53.          int dimensions;
54.          int lineCounter;
55.
56.          input_file >> dimensions;
57.          input_file >> lineCounter;
58.
59.          //dynamic allocations of matrices
60.          pts->input_points = new point[lineCounter];
61.          pts->input_points = new point[lineCounter];
62.
63.          pts->numoflines = lineCounter;
64.          pts->dimensions = dimensions;

65.          cout << endl;
66.          cout << "\nPrinting the 3D dimensions of molecule - protein and its weights..." << endl;
67.          cout << "\t\t\t\t||     1D      2D      3D      WEIGHT\t||" << endl;
68.          for (int i=0 ; i<lineCounter ; i++){                          // Get all dimensional coordinates

69.                    pts->input_points[i].coords = new double[dimensions];          //allocate matrix holding coordinates of each
     x-dimensional point
70.                    cout << dimensions-1 << "-dimensional protein point" << i << " = ||    " ;
71.                    for (int j=0 ; j<dimensions ; j++){                 // Get each points dimensional coordinates
72.                    input_file >> pts->input_points[i].coords[j];
73.                    cout << pts->input_points[i].coords[j] << "  ,  ";
74.                    }
75.                    cout << "\t||" << endl;
76.          }

77.          cout << "\nJust printed the 3D dimensions of molecule - protein and its weights...\n" << endl;
78.
79.          input_file.close();
80.  }
```

Κώδικας για αναπαράσταση πρωτεΐνης, εύρεσης της κατάλληλης α παραμέτρου(συνέχεια) (αρχείο: a-shape_protein.cpp)

```
81.  int main(){

82.          points2 pts;
83.          ReadFile(&pts);

84.          list<Point> lp;
85.          list<Weighted_point> lwp;

86.          //input: a molecule - protein
87.          for (int i=0 ; i<(pts.numoflines) ; i++){
88.
             lp.push_back(Point(pts.input_points[i].coords[0],pts.input_points[i].coords[1],pts.input_points[i].coords[2]));
89.          }

90.          for (int i=0 ; i<(pts.numoflines) ; i++){
91.
             lwp.push_back(Weighted_point(Bare_point(pts.input_points[i].coords[0],pts.input_points[i].coords[1],pts.input_point
     s[i].coords[2]),pts.input_points[i].coords[3]));
92.          }
93.
94.  //simple points -- Delaunay triangulation
95.          // compute alpha shape with delaunay triangulation
96.          Del_Alpha_shape_3 as(lp.begin(),lp.end());
97.          cout << "Alpha shape computed in REGULARIZED mode by default" << endl;
98.
99.          // find optimal alpha value for simple points with delaunay triangulation
100.         Del_Alpha_shape_3::NT alpha_solid = as.find_alpha_solid();
101.         Alpha_iterator opt = as.find_optimal_alpha(1);
102.
103.         cout << "\nSmallest alpha value for delaunay triangulation to get a solid through data points is " << alpha_solid <<
     endl;
104.         cout << "\nOptimal alpha value for delaunay triangulation to get one connected component is " << *opt << endl;
105.         as.set_alpha(*opt);
106.         assert(as.number_of_solid_components() == 1);
```

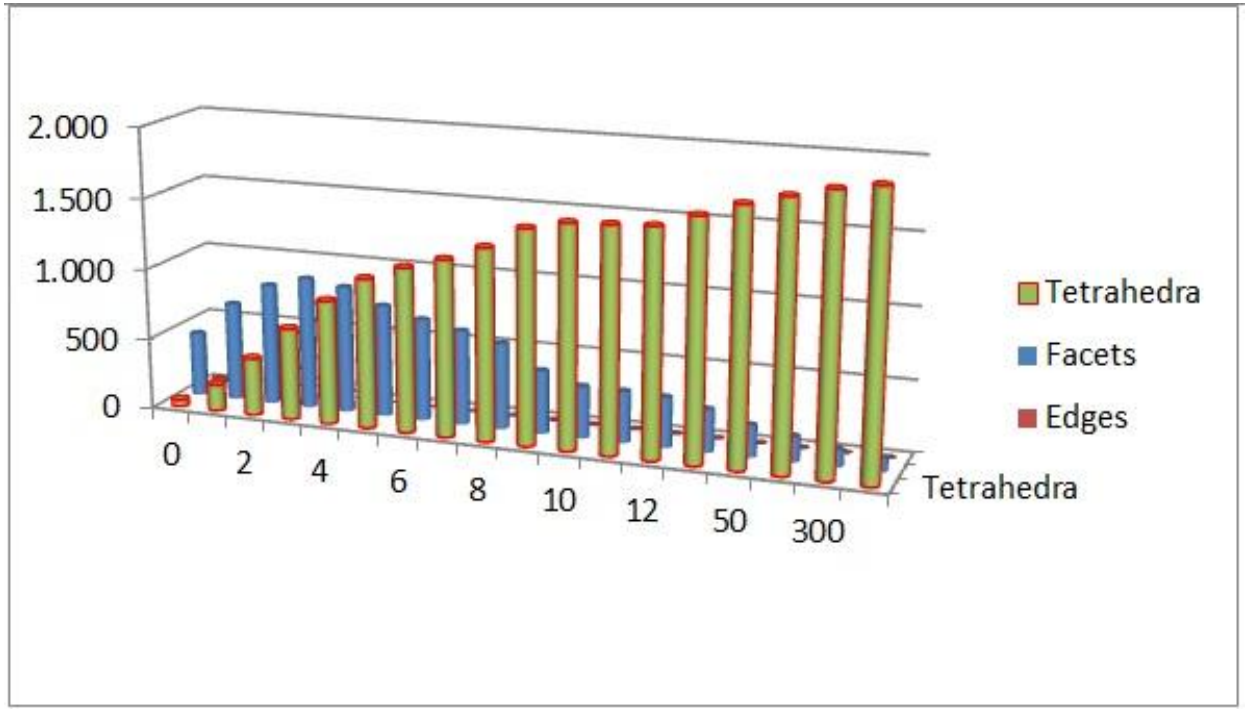
Κώδικας για αναπαράσταση πρωτεΐνης, εύρεσης της κατάλληλης α παραμέτρου(συνέχεια) (αρχείο: a-shape_protein.cpp)

```
107.    //weighted points -- Regular triangulation
108.            //build alpha_shape  in GENERAL mode and set alpha=0
109.            int a=0;
110.            cout << "Give the 'a' parameter to compute the counterpart cells, facets and edges accordings to this
        parameter!\nType : ";
111.            cin >> a;
112.            Reg_Alpha_shape_3  ras(lwp.begin(), lwp.end(), a, Reg_Alpha_shape_3::GENERAL);

113.            //explore the x-shape of weighted points with regular triangulation - It is dual to the boundary of the union.
114.            list<Cell_handle> cells;
115.            list<Facet>       facets;
116.            list<Edge>        edges;
117.            ras.get_alpha_shape_cells(std::back_inserter(cells), Reg_Alpha_shape_3::INTERIOR);
118.            ras.get_alpha_shape_facets(std::back_inserter(facets), Reg_Alpha_shape_3::REGULAR);
119.            ras.get_alpha_shape_facets(std::back_inserter(facets), Reg_Alpha_shape_3::SINGULAR);
120.            ras.get_alpha_shape_edges(std::back_inserter(edges), Reg_Alpha_shape_3::SINGULAR);
121.            cout << "\nThe 0-shape built with regular triangulation has : " << endl;
122.            cout << cells.size() << " interior tetrahedra" << endl;
123.            cout << facets.size() << " boundary facets" << endl;
124.            cout << edges.size()  << " singular edges" << endl;


125.            return 0;
126.    }
```

Διάγραμμα κατανομής των απλόκων με βάση την αλλαγή της παραμέτρου α



- Από την υλοποίηση του προγράμματος που παρουσιάστηκε προκύπτει ότι η καλύτερη τιμή που μπορεί να πάρει η παράμετρος α είναι **11.1103**.

Ευχαριστώ πολύ!

- Καλό καλοκαίρι!!!





- Νικόλας Μπεγέτης
- Προπτυχιακός φοιτητής