Proc. 3rd Int. Conf. on Intelligent Systems for Molecular Biology (ISMB'95), Cambridge, UK, AAAI Press 1995.

Solvent Accessible Surface Representation in a Database System for Protein Docking[1]

Thomas Seidl and Hans-Peter Kriegel

Institute for Computer Science, University of Munich
Leopoldstr. 11 B
D-80802 München, Germany
email: { seidl | kriegel }@informatik.uni-muenchen.de

Abstract

Protein docking is a new and challenging application for query processing in database systems. Our architecture for an efficient support of docking queries is based on the multi-step query processing paradigm, a technique well-known from spatial database systems. Along with physicochemical parameters, the geometry of the molecules plays a fundamental role for docking retrieval. Thus, 3D structures and 3D surfaces of molecules are basic objects in molecular databases. We specify a molecular surface representation based on topology, define a class of neighborhood queries, and sketch some applications with respect to the docking problem. We suggest a patch-based data structure called the *TriEdge structure*, first, to efficiently support topological query processing, and second, to save space in comparison to common planar graph representations such as the quad-edge structure. In analogy to the quad-edge structure, the TriEdge structure has an algebraic interface and is implemented via complex pointers. However, we achieve a reduction of the space requirement by a factor of four. Finally, we investigate the time performance of our prototype.

Keywords: Molecular surface representation, planar graphs, surface approximation, local shape index, efficient query processing, database systems, protein docking.

1. Introduction

The fundamental 3D objects in molecular biology and computational biochemistry are large molecules with several hundreds to thousands of atoms. Various applications require access to the 3D structure of the molecules, as it is provided for proteins by the Brookhaven Protein Data Bank (PDB) (Bernstein et al. 1977). Up to now, the PDB contains 3,000 proteins, enzymes, and viruses (PDB 1994). For each entry, along with information on the chemical structure of the protein, the 3D coordinates of its atoms are stored in a text file.

In the last years, a new topic has been emerging in the area of protein engineering: the prediction of molecular interaction, called the *docking* problem. Several methods has been suggested to meet the one-to-one docking problem (Connolly 1986), (Badel, Mornon, & Hazout 1992), (Katchalski-Katzir et al. 1992), (Fischer et al. 1993), (Helmer-Citterich & Tramontana 1994): Which constellation of two given proteins represents a stable complex? Since all of the six degrees of freedom to compose two molecules together in 3D space are continuous, the constellation space is infinite. Common discretizations of protein surfaces result in thousands of points, and, at first sight, each of them can represent a possible docking site.

In our project "BIOWEPRO: a Database System for Protein Docking", we are faced with the one-to-many docking problem: Select such proteins from the PDB that are able to form a stable complex in interaction with a given query protein, and determine appropriate constellation parameters. When considering the number of proteins in the database ($3 \cdot 10^3$) and the number of possible docking sites on a protein in the database ($> 10^3$) and on the query protein ($> 10^3$), the search space at least has an overall size of billions of protein-protein-constellations (Ester et al. 1995).

Thus, docking retrieval is a new and challenging application for spatial database systems. Due to the enormous size of the search space, a multi-step query processing architecture is recommended. In spatial database systems, this paradigm efficiently supports the processing of point and region queries as well as spatial joins (Brinkhoff et al. 1993), (Kriegel, Schneider, & Brinkhoff 1993), (Brinkhoff et al. 1994). Additionally, we perform several steps of preprocessing.

From a geometric point of view, the main object of interest in protein docking is the 3D molecular surface, rather than the 3D arrangement of the atoms a molecule consists of. Moreover, the *local shape* of the surface at a possible docking site is the fundamental geometric criterion for docking retrieval, rather than the location of a docking site in space. This is analogous to similarity retrieval in CAD databases (Schneider et al. 1989). Since the interaction of molecules is restricted to the docking site, we do not require a description of the global shape (outline, contour) of molecules. Along with the geometry, also physicochemical properties determine molecular interactions and, therefore, have to be considered when representing molecular surfaces.

1. This research was funded by the German Ministry for Research and Technology (BMFT) under grant no. 01 IB 307 B. The authors are responsible for the content of this paper.

As we will see in section 3, molecular surfaces are smooth but quite bumpy (uneven). Since all bumps are of similar atomic size, the selectivity of a local shape index would be very low when based on infinitesimal neighborhoods. Instead, for each surface primitive (patch or point, resp.), we collect its neighbors within an appropriate radius, and determine the local shape via a paraboloid as a simple approximation.

Collecting neighbors requires access to adjacency information, as it is provided by well-known data structures like the quad-edge structure (Guibas & Stolfi 1985). In this paper, we propose a new basic technique to represent molecular surfaces, reducing the storage requirement by a factor of four. Since our method is based on topology, it supports effective and efficient processing of queries on the molecular surface structure. We formally introduce the conditional neighborhood query, and give an algorithm to process it. As an application, local surface approximation is considered in detail. We implemented our method in C++ on top of the object-oriented database management system ObjectStore (Orenstein et al. 1992).

The paper is organized as follows: In section 2, we shortly present our architecture to process docking queries. In section 3, we give a specification to represent molecular surfaces and, in section 4, we introduce the neighborhood query and some applications with respect to the docking problem. In section 5, the TriEdge data structure is presented that supports efficient processing of neighborhood queries. Section 6 contains first evaluation results, and section 7 concludes the paper with a summary and an outlook to future work.

2. A Query Processing Architecture for Protein-Docking

The enormous size of the constellation space implies a great challenge to efficient query processing. To reduce the set of candidate complexes step by step in an efficient way, we provide our docking system with a multi-step query processing architecture. This technique causes very small cost per constellation in the filter step, reducing the set of possible docking candidates drastically when compared to the overall size of the constellation space. Since the quality of the remaining candidates is not expected to be very good, further refinement steps have to be performed. These may cause higher cost per candidate pair, but have to ensure high quality results.

Further reduction of the constellation space is achieved by managing groups of points instead of the points themselves. We call such groups *segments*, and create them by collecting points with similar features. In analogy to the knobs and holes of (Connolly 1986), we use points with local extreme features as centers of the segments. We assume these segments to represent potential docking sites.

The preprocessing steps begin with calculating geometric and physicochemical features for a set of points on the surface in a location independent way. The features of the segments, aggregated from the features of the points, are complemented. Then, the segments are inserted into a multidimensional access method, called *feature index* (Aldinger et al. 1994) for an efficient retrieval of docking partners from the set of segments of all the proteins in the database. Obviously, representation methods for surface segments are the more appropriate, the more they are independent from location and direction in space, thus being invariant with respect to translation and rotation. Figure 1 illustrates the steps of our query processing architecture as described below in more detail.

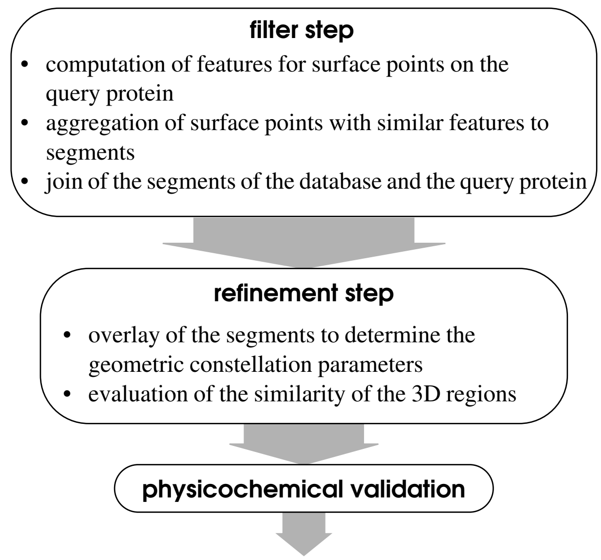


Figure 1: query processing architecture

(1) Filter step using geometric feature vectors. In analogy to the preprocessing steps, features and segments are determined on the query protein. For the retrieval of segments from the database that are complementary to the query segments, we store the segments in a second feature index without performing complementation. A multidimensional join is performed with the two feature indices, yielding candidate pairs. Each pair consists of a segment from the database and a segment of the query protein, both having similar features where similarity is defined by a given threshold.

(2) Refinement step using 3D geometry. For the comparison of the 3D geometry, we need the exact geometry of the constellations. Therefore, the segments of a candidate pair are fitted together to determine the best relative position representing a complex in 3D space. Then, some simple evaluations of the geometric similarity are performed, and constellations are discarded for which the protein pairs overlap on some site. The result of the refinement step is a set containing geometric constellation parameters for pairs of proteins.

(3) Validation of the constellations with respect to physicochemical properties. In the first refinement step of query processing, only geometric criteria are considered for docking candidates. The physicochemical properties of the surface like hydrophobicity and electrostatic potential also have a high degree of relevance for docking. This is due to the fact that these interactions depend on the relative position of two proteins which is only determined in the refinement step. In the last step of query processing, constellations are returned that fulfill both, the geometric and the physico-chemical criteria for the docking of two proteins.

Since no algorithm is yet known to classify a predicted complex to be stable in reality with a high probability, the results delivered from the docking system finally have to be tested by an biomolecular laboratory experiment. Thus, we cannot find final results in an early step of query processing, but may only discard constellations which are recognized to represent no valid complex.

3. Molecular Surface Representation

We consider the molecular surface to be the solvent accessible surface for a solvent probe radius α following (Richards 1977). Various implementations for the calculation of molecular surfaces are published (Connolly 1983a), (Varshney, Brooks, & Wright 1994), (Halperin & Overmars 1994). Molecular surfaces have a strong regularity, and they only consist of three types of patches: Convex spherical patches, saddle-shaped rectangles, and concave spherical triangles (figure 2). These types depend on the number of atoms that the probe sphere is in simultaneous contact with when rolling over the molecule. Since the algorithms enforce the atoms to be in general position, ensuring the probe sphere never being in simultaneous contact with more than three atoms, the complexity of the algorithms is reduced drastically. Obviously, this also holds for the representation of molecular surfaces.

At first sight, a patch-based representation of molecular surfaces seems to be difficult to manage: There are different types of patches with a different number of vertices, etc.. Thus, triangulated surface representations are quite common to be used for docking purposes, and also for visualization. However, there are strong advantages for the patch-based representation: First, for a particular molecule and a given probe radius α, the solvent accessible surface is well defined in its structure and its shape. Second, for any given point density, the patchwork can be refined to a dotted surface and a triangulation, but not vice versa. Third, every patch is homogeneous with respect to the (infinitesimal) curvature, and the normal vector of a surface point is determined by the geometric parameters of the patch it belongs to, i.e. via the associated atom sphere, torus, or probe sphere, respectively. Overall, we prefer a patch-based surface representation rather than a point-based method or an arbitrary triangulation.

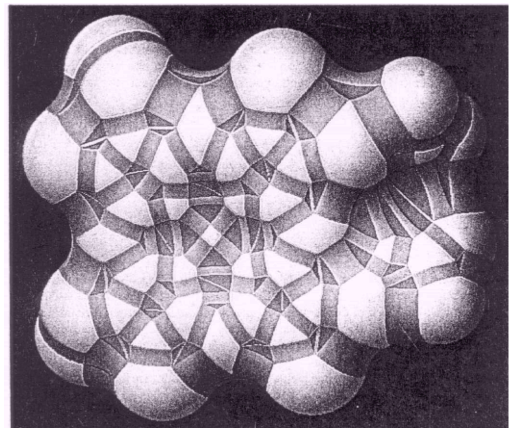


Figure 2: surface of a portion of hemoglobin, from (Connolly 1983b)

The articles describing molecular surface calculation as found in molecular biology (Connolly 1983b), computer graphics (Varshney, Brooks, & Wright 1994), and computational geometry (Halperin & Overmars 1994), do not mention their surface representation method. In (Halperin & Overmars 1994) only, there is a hint that extended van-der-Waals surfaces are represented by the quad-edge structure. In the following, we present a patch-based representation for molecular surfaces providing access to the patches via an edge algebra. Such an edge algebra has been suggested for the quad-edge structure by Guibas and Stolfi (Guibas & Stolfi 1985) to represent planar graphs such as molecular surfaces.

As an interface for our surface representation, we specify a set of edge functions. An illustration of the operations is given in figure 3. We implement this specification as a set of C++ classes for the object-oriented database system ObjectStore. However, to demonstrate the relations between the functions, we give an algebraic specification with axioms. From the `EDGE` $\rightarrow$ `EDGE` functions, `Sym` and `LEdge` or `Sym` and `Lnext` are primitive, the others can be derived with respect to the axioms:

```
SPEC MolSurface;
 TYPES EDGE, FACE, VERTEX;
 FUNCTIONS
   Sym, LEdge, REdge,
    Lnext, Lprev:  EDGE -> EDGE;
   Left, Right:    EDGE -> FACE;
   Orig, Dest:     EDGE -> VERTEX;
   any_edge:       FACE -> EDGE;
   any_edge:       VERTEX -> EDGE;
```

```
  AXIOMS (for EDGE e, FACE f, VERTEX v)
    e.Sym().Sym() == e;
    e.Redge().LEdge() == e;
    e.Lnext() == e.Sym().REdge();
    e.Lprev() == e.LEdge().Sym();
    e.Sym().Orig() == e.Dest();
    e.LEdge().Orig() == e.Orig();
    f.any_edge().Left() == f;
    v.any_edge().Orig() == v;
END MolSurface.
```

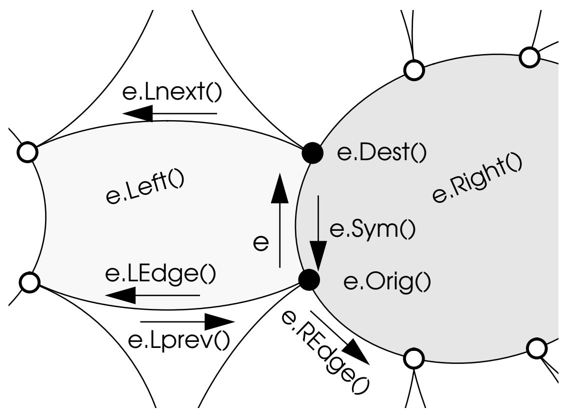


Figure 3: illustration of the MolSurface algebra

In addition to the algebra, we define iteration statements to assign all the edges e around a face or a vertex counterclockwise to an edge variable e: `forall_around_face(e,face){...}` and `forall_around_vertex(e,vertex){...}`. Thereby, the edge cycle is traversed by the `Lnext` or `LEdge` function, resp., until the first edge is reached a second time.

In this paper, we focus the specification onto the topological structure of molecular surfaces. To yield the location of the vertices and some representative points of the patches, further functions provide access to geometric attributes, e.g. `location: VERTEX` → `3D_POINT`, and `representative: FACE` → `3D_POINT`. Similar functions deliver thematic attributes and shape information. This approach results in a representation hierarchy as shown in figure 4.

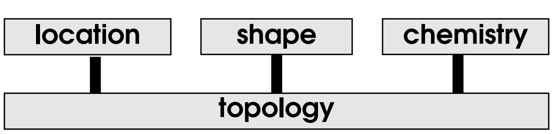


Figure 4: topology-based representation of molecular surface

Up to here, we specified our patch-based representation of molecular surfaces. In section 4, we show how neighborhood queries are performed based on this specification, and in section 5, we explain our MolSurface algebra implementation.

4. Application for Protein Docking

In this section, we present some applications that require topological information. We introduce the neighborhood query, and apply it to local surface approximation, surface segmentation, and determination of local extrema.

A *neighborhood query* is specified as a conditional neighborhood, $n_c(p)$, that will be selected from a surface in the database. Therefore, $n_c(p)$ is determined by a center patch or point p, and by the condition c that can be a euclidean distance criterion, a similarity criterion, or any other binary relation. For a surface graph g, $n_c(p)$ is defined to be the maximal connected subgraph of g restricted to the patches v (or points, resp.) that fulfill the condition c(v,p). Thus, for each $v \in n_c(p)$, one of the following properties holds: (i) c(v,p) and (v=p), or (ii) c(v,p) and a neighbor of v is in $n_c(p)$. In other words, $n_c(p)$ contains only patches p that (1) are reachable from p within $n_c(p)$, and (2) fulfill the condition c together with p: c(v,p).

Various applications of the general concept of neighborhood queries can be thought of. The following examples show some of them in the domain of protein engineering.

(1) Local approximation of the molecular surface. For the calculation of a local shape index at a patch p on our patch-based surface representation, we follow (Zachmann et al. 1992) and approximate the neighborhood of p by a paraboloid. As mentioned above, infinitesimal curvatures are not specific. Thus, we provide control over the locality of the approximation by a radius parameter r. A so called euclidean neighborhood query, $n_r(p)$, consists in the selection of all the patches that are reachable from p within the euclidean distance r. In this case, the condition c simply is a comparison: c(v,p) = '$dist_{euclid}(v,p) < r$' (cf. figure 5).

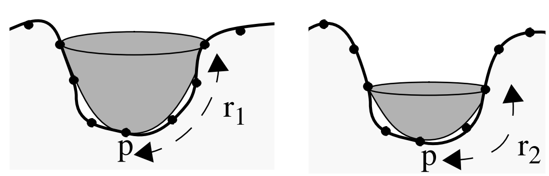


Figure 5: surface approximation for different neighborhood radii r_1 and r_2

(2) Segmentation of molecular surfaces. We perform segmentation via a region growing algorithm. For each of the patches in a sprout set, we perform a neighborhood query specified by a similarity criterion, e.g. two patches are similar if they both are saddle-shaped, or if they have the same sign for the electrostatic potential. The resulting segments may have various extensions on the surface and in 3D space, since they are not bound by a spatial distance criterion. Also, they may overlap. When providing a similarity criterion with a very low significance, a single segment can cover the whole surface.

(3) Determination of local extrema. In (Connolly 1986), knobs and holes are defined to be surface points having an extreme solid angle with respect to their neighbors. As a generalization, our neighborhhood query supports specifying arbitrary neighborhood radii leading to different degrees of locality. Thus, various radii can be investigated with respect to their appropriateness for the purpose of docking retrieval.

Neighborhood query processing. Based on the surface representation specified in section 3, we perform neighborhood query processing via a graph traversal algorithm:

```
void Neighborhood
  ( PATCH* p, CONDITION* cond,
    os_Set <PATCH*>& result)
{
  os_List <PATCH*> open;
  PATCH *h, *v;
  result.clear();

  if (cond->eval(p,p))
    {
      open.insert(p);
      result.insert(p);
    }

  while (h = open.remove_first())
    // expand current patch h:
    forall_around_face( e, h )
    {
      v = e.Right();
      if (not result.contains(v)
          and cond->eval(v,p))
        {
          result.insert(v);
          open.insert(v);
        }
    }

  return;   // result is refparam
}
```

The behavior of the algorithm is controlled by the insertion method into `open`: When inserting new patches at the end, a breadth-first traversal is performed, whereas an insertion at the beginning would result in a depth-first traversal.

5. The TriEdge Data Structure

Up to here, we declared our objects of interest, specified access operations, and presented a basic query class with applications. Now, we illustrate our approach to implement the objects and their operations in a very efficient way.

The quad-edge structure (Guibas & Stolfi 1985) is quite common for the representation of planar graphs such as molecular surfaces. The implementation is based on edge records containing four pointers to the two bounding vertices and the two co-bounding faces, and four complex pointers to the edges that are adjacent via the origin and destination vertex in clockwise and counterclockwise direction. This concept is similar to the doubly-connected-edge-list (DCEL) from (Preparata & Shamos 1985), with two differences: First, the quad-edge structure is defined via an algebra, leading to a more comfortable access to edges, their symmetric and dual edges, in comparison to the DCEL. Second, the complex pointers of the quad-edge structure contain a simple pointer to an adjacent edge record, and additionally, unlike a DCEL entry, an offset value representing the view to the referenced edge. Exactly this idea of an algebraic interface together with a complex pointer concept is used for our TriEdge structure as described below. This approach is well supported by the object-oriented data model and leads to an easy integration.

However, the quad-edge structure requires 36 bytes per record, when adjusted to a multiple of four. Real molecular surfaces have a size of some thousand patches and edges, for a common probe radius of 1.4 Å which is the size of a water molecule. Therefore, the storage requirement for the surface topology information of a single molecule is hundreds of kilobytes (cf. table 1). Since the number of proteins in the PDB currently is 3,000 (PDB 1994), the size of our 3D protein database for docking retrieval comes into the range of gigabytes. Therefore, an efficient implementation is crucial.

protein	number of atoms	number of patches	number of edges	size of quad-edge records
2pab.a	872	3710	7380	260 kbytes
2pab.b	872	3776	7524	265 kbytes
2ptc.e	1629	5606	11160	393 kbytes
2ptc.i	454	1990	3972	140 kbytes
2sni.e	1983	6220	12396	436 kbytes
2sni.i	513	2116	4224	149 kbytes

Table 1: surfaces of real proteins

Our approach to implement the MolSurface specification consists of two steps: First, we reduce the complexity of the representation by mapping the molecular surface graph to an equivalent but simpler graph structure. Second, we exploit the strong regularity of the new structure as a key property for an efficient representation. Overall, we save space and, therefore, time for the reduced volume of data transfer within the database system. In this section, we explain our technique, whereas in section 6, the space reduction factor is shown in detail, and a first runtime evaluation of our prototype is presented.

The main observation concerns the structure of molecular surfaces and the role of the saddle patches: Every saddle patch connects two convex patches and two concave triangles. Both types, the convex and the concave patches, are surrounded by cycles of saddle patches. This is analogous to faces and vertices of a planar graph that are surrounded each by a cycle of edges. We exploit the analogy to develop a new storage method for molecular surfaces, and call the new structure the *simplified surface graph* (SSG, cf. figure 6), which is topologically equivalent to the accessible surface as defined in (Richards 1977).

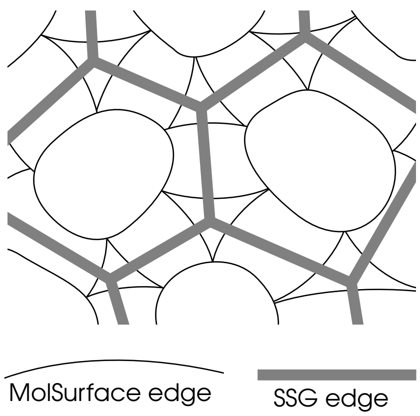


Figure 6: detail of a molecular surface: MolSurface edges and SSG edges

Like the molecular surface, also the SSG is a planar graph, consisting of vertices, edges, and faces. For molecular surfaces, the vertices are mapped to corner points, the edges to arcs, and the faces to patches. However, for the SSG, we associate the vertices to the concave triangles of the molecular surface, the edges to the saddle patches, and the faces to the convex patches (cf. table 2). Accessing a patch or vertex of the molecular surface via this mapping can be performed in O(1). The number of edges to be stored is reduced drastically: Every edge of a molecular surface belongs to exactly one saddle, and every saddle is bound by four (molecular) edges. Obviously, a molecular surface graph has four times as many edges as saddle patches, and, therefore, four times as many edges as the SSG.

molecular surface	MolSurface graph	simplified surface graph
corner point	vertex	—
patch border arc	edge	—
concave triangle	face	vertex
saddle rectangle	face	edge
convex patch	face	face

Table 2: relationship between molecular surfaces and the graph structures

A second key observation leads us to a further reduction of the storage space. Since an SSG is a planar graph, we could use the quad-edge structure to store it in the database. However, we can exploit a basic property of an SSG: All of the SSG vertices have a degree of three, since there are only triangles among the concave patches, due to the general position of the atoms: `e.LEdge()`^{3} = `e` for every `EDGE e`. Therefore, we store the edge cycles around the SSG vertices in arrays with a fixed length of three, rather than using any dynamic structure. This leads us to the TriEdge structure which consists of records that contain three SSG half-edges, all of them belonging to the same SSG `LEdge` cycle. Similar to the quad-edge structure, an edge is represented by a complex pointer that consists of a simple pointer to the TriEdge record, and of a view that specifies which SSG half-edge of the `LEdge` cycle is represented (cf. figure 7).

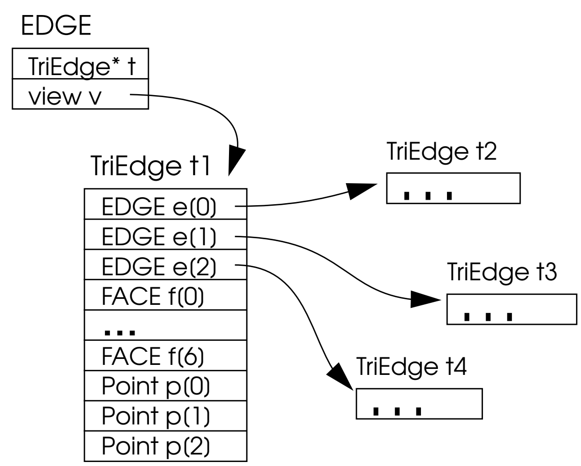


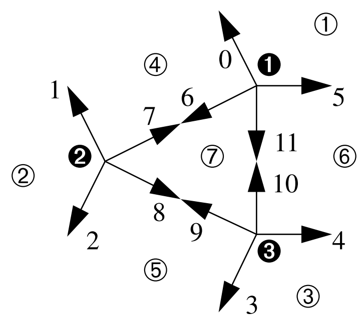
Figure 7: the TriEdge data structure

A TriEdge record contains three complex edge pointers to represent the connectivity information. Due to the mapping from SSG to molecular surface, a TriEdge record has to provide access to seven patches: One concave triangle as an image of the SSG vertex, three saddle patches as images of the three SSG edges, and three convex patches as images of the SSG faces that are adjacent to the saddles. Since the corner points are shared between the patches, we associate them to the TriEdge records rather than to the patches.

For molecular surfaces, a TriEdge record represents twelve half-edges (cf. figure 8a) and, therefore, requires the view v being in the range of `[0..11]`. Since all of the elements of an `LEdge` cycle are represented by the same TriEdge record, the `LEdge` operation requires no dereferencing of the TriEdge pointer of an edge, but only a change of the view v. This function is the same for all edges and, therefore, can be carried out via lookup in a constant table `LEDGE: [0..11]` → `[0..11]`, implemented simply as an array. For the `Sym` operation, dereferencing a TriEdge pointer is required in six cases, in the other six cases a change of the view is sufficient. When providing further arrays SYM, LEFT, and ORIG, the basic MolSurface operations are implemented as follows (cf. figure 8b):

```
LEdge MolSurface :  (t,v) → (t, LEDGE[v])
Sym MolSurface :    (t,v) → if (v > 6)
                              then (t, SYM[v])
                              else sym((*t).e[v/2],v)
Left MolSurface :   (t,v) → (*t).f[LEFT[v]]
Orig MolSurface :   (t,v) → (*t).p[ORIG[v]]
```

a)



b)

view	0	1	2	3	4	5	6	7	8	9	10	11
LEDGE	6	2	8	4	10	0	11	1	7	3	9	5
SYM	*	*	*	*	*	*	7	6	9	8	11	10
LEFT	④	②	⑤	③	⑥	①	⑦	④	⑦	⑤	⑦	⑥
ORIG	❶	❷	❷	❸	❸	❶	❶	❷	❷	❸	❸	❶

Figure 8: a) a TriEdge record represents twelve molecular half-edges, b) four arrays to implement the basic operations

6. Evaluation

First, we investigate the storage requirement for the TriEdge structure in comparison to a naive representation of molecular surfaces by the quad-edge structure, and in comparison to an implementation of the SSG by the quad-edge structure. We assume that the vertices and the patches are referenced by 4-byte pointers, and the records being adjusted to a multiple of four bytes. Let e denote the overall number of edges on the molecular surface. Table 3 shows the reduction of storage space for molecular surfaces by the factor 3.9 when considering topological information as well as references to the patches and their corner points.

storage requirement		MolSurf via quad-edge	SSG via quad-edge	SSG via TriEdge
references per record to …	edges	4	4	3
	corners	2	4	3
	patches	2	4	7
bytes per record		36	52	56
number of records		e	e / 4	e / 6
total bytes		36 · e	13 · e	9.33 · e
factor		1	2.8	3.9

Table 3: storage requirement for molecular surfaces by different techniques

We integrated the TriEdge structure into our object-oriented protein database system based on the OODBMS ObjectStore. From our prototype, we obtained first results on the runtime behavior of the TriEdge structure on a HP-9000/735 workstation under HP-UX 9.01. Inserting molecular surfaces takes only a few seconds of elapsed time. This includes reading the surface structure from a text file, creating the TriEdge records as well as the patch objects, and connecting the TriEdge records, all embraced by transaction begin and commit.

From further experiments, we achieved the processing time for some selected euclidean neighborhood queries. The elapsed time is shown in table 4 for various radii. We determined the values as an average over 1,000 calls each, performed on a patch on the protein 2pab.a. As expected, the runtime grows nearly linear with respect to the number of neighbors (cf. figure 9).

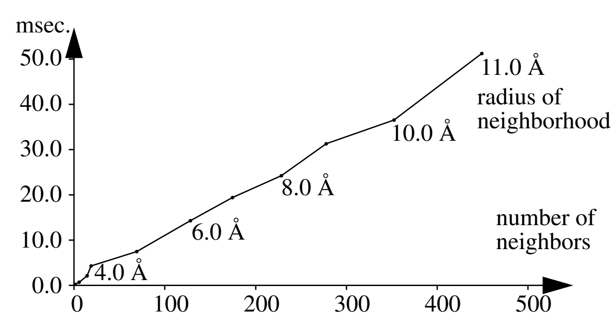


Figure 9: dependency of the runtime on the number of neighbors

radius (Å)	number of neighbors	msec per query
1.0	3	0.1
2.0	7	0.6
3.0	16	2.0
4.0	21	4.2
5.0	71	7.4
6.0	130	14.2
7.0	177	19.3
8.0	231	24.0
9.0	280	31.2
10.0	355	36.3
11.0	452	51.1
20.0	1607	164

Table 4: runtime of neighborhood queries for a patch of the protein 2pab.a (average over 1,000)

7. Conclusion

In this paper, we investigated the structure of molecular surfaces as fundamental 3D objects in database systems for molecular biology. Along with the geometric and physicochemical properties, in particular the topological structure of molecular surfaces has to be represented. We sketched our architecture for efficient processing of one-to-many protein docking retrieval, based on the multi-step query processing paradigm. A specification for surface representation is presented, together with some applications like local surface approximation and surface segmentation. The technical contribution of the paper is the introduction of the TriEdge data structure as an implementation of molecular surface graphs providing efficient support for neighborhood query processing. It leads to a considerably reduced space requirement compared to the well-known quad-edge structure. First experimental results show the query processing time to be in the range of milliseconds. The insertion time may be improved when integrating the molecular surface calculation program into the database system, thus avoiding text file input.

In future work, we will integrate the case of long side chains that yield loops within the molecular surfaces, and the case of convex patches that belong to more than one edge cycle. Currently we are working on the integration of techniques to distribute points evenly spaced on the surface. Furthermore, we investigate how dedicated clustering techniques will improve neighborhood query processing.

Acknowledgements

We thank our colleague Martin Ester for engaging and fruitful discussions on the concepts of this paper.

References

Aldinger K., Ester M., Förstner G., Kriegel H.-P., Seidl T., 1994: *'A Database System Supporting Protein-Protein-Docking: an Efficient and Robust Feature-Index',* Proc. 2nd GI Conference on Computer Science and Biology, Jena, Sept. 1994, pp. 41-52, in German.

Bernstein F. C., Koetzle T. F., Williams G. J., Meyer E. F., Brice M. D., Rodgers J. R., Kennard O., Shimanovichi T., Tasumi M., 1977: *'The Protein Data Bank: a Computer-based Archival File for Macromolecular Structures',* Journal of Molecular Biology, Vol. 112, pp. 535-542.

Brinkhoff T., Horn H., Kriegel H.-P., Schneider R., 1993: *'A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems',* Proc. 3rd Int. Symp. on Large Spatial Databases (SSD '93), Singapore, Lecture Notes in Computer Science, Vol. 692, Springer, pp. 357-376.

Brinkhoff T., Kriegel H.-P., Schneider R., Seeger B., 1994: *'Multi-Step Processing of Spatial Joins',* Proc. ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, MN, 1994, pp. 197-208.

Badel A., Mornon J. P., Hazout S., 1992: *'Searching for geometric molecular shape complementarity using bidimensional surface profiles',* Journal of Molecular Graphics, Vol. 10, pp. 205-211.

Connolly M. L., 1983a: *'Analytical molecular surface calculation',* Journal of Applied Crystallography, Vol. 16, pp. 548-558.

Connolly M. L., 1983b: *'Solvent-Accessible Surfaces of Proteins and Nucleic Acids'*, Science, Vol. 221, pp. 709-713.

Connolly M. L, 1986.: *'Shape Complementarity at the Hemoglobin $\alpha_1\beta_1$ Subunit Interface',* Biopolymers, Vol. 25, pp. 1229-1247.

Ester M., Kriegel H.-P., Seidl T., Xu X., 1995: *'Shape-based Retrieval of Complementary 3D Surfaces from a Protein Database',* Proc. GI Conf. on Database Systems for Office Automation, Engineering, and Scientific Applications (BTW '95), Informatik aktuell, Springer, pp. 373-382, in German.

Fischer D., Norel R., Nussinov R., Wolfson H. J., 1993: *'3-D Docking of Protein Molecules',* Proc. 4th Annual Symposium on Combinatorial Pattern Matching (CPM '93), Padova, Italy, in: Lecture Notes in Computer Science, Vol. 684, Springer, pp. 20-34.

Guibas L., Stolfi J., 1985: *'Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams',* ACM Trans. Graphics, Vol. 4, No. 2, pp. 74-123.

Halperin D., Overmars M. H., 1994: *'Spheres, Molecules, and Hidden Surface Removal',* Proc. 10th ACM Symp. Computational Geometry, pp. 113-122.

Helmer-Citterich M., Tramontano A., 1994: *'PUZZLE: A New Method for Automated Protein Docking Based on Surface Shape Complementarity',* Journal of Molecular Biology, Vol. 235, pp. 1021-1031.

Katchalski-Katzir E., et al., 1992: *'Molecular Surface Recognition: Determination of Geometric Fit between Proteins and their Ligands by Correlation Techniques',* Proc. National Academy of Science USA, Vol. 89, pp. 2195-2199.

Kriegel H.-P., Schneider R., Brinkhoff T., 1993: '*Potentials for Improving Query Processing in Spatial Database Systems',* invited talk, Proc. 9èmes Journées Bases de Données Avancées (9th Conference on Advanced Databases), Toulouse, France.

Orenstein J., Haradhvala S., Margulies B., Sakahara D., 1992: *'Query Processing in the ObjectStore Database System',* Proc. ACM SIGMOD '92, pp. 403-412.

Protein Data Bank, 1995: *'Quarterly Newsletter No. 71 (Jan. '95)',* Brookhaven National Laboratory, Upton, NY.

Preparata F. P., Shamos M. I., 1985: *'Computational Geometry—An Introduction',* Springer.

Richards F. M., 1977: *'Areas, Volumes, Packing, and Protein Structure',* Annual Reviews in Biophysics and Bioengineering, Vol. 6, pp. 151-176.

Schneider R., Kriegel H.-P., Seeger B., Heep S., 1989: *'Geometry-based Similarity Retrieval of Rotational Parts',* Proc. Int. Conf. on Data and Knowledge Systems for Manufacturing and Engineering, Gaithersburg, ML, pp. 150-160.

Varshney A., Brooks F. P., Wright W. V., 1994: *'Computing Smooth Molecular Surfaces',* IEEE Computer Graphics & Applications, Vol. 14, No. 5, pp. 19-25.

Zachmann C.-D., Heiden W., Schlenkrich M., Brickmann J., 1992: *'Topological Analysis of Complex Molecular Surfaces'*, Journal of Computational Chemistry, Vol. 13, No. 1, pp. 76-84.