Local approximate 3D matching of proteins in viral cryo-EM density maps

Stefan Burkhardt, Kimmo Fredriksson, Tuomas Ojamies, Janne Ravantti, Esko Ukkonen

Abstract— **Experimental structure analysis of biological molecules (e.g, proteins) or macromolecular complexes (e.g, viruses) can be used to generate three-dimensional density maps of these entities. Such a density map can be viewed as a three-dimensional gray-scale image where space is subdivided in voxels of a given size. The focus of this paper is the analysis of virus density maps. The hull of a virus consists of many copies of one or several different proteins. An important tool for the study of viruses is cryo-electron microscopy (cryo-EM), a technique with insufficient resolution to directly determine the arrangement of the proteins in the virus. We therefore created a tool that locates proteins in the three-dimensional density map of a virus. The goal is to fully determine the locations and orientations of the protein(s) in the virus given the virus' three-dimensional density map and a database of density maps of one or more protein candidates.**

I. Introduction

There are many variants of 3D matching. They differ in the representation of data they work on and the type of matching required. The differences between discrete 3D objects and discretized continuous data are very significant for the choice of algorithm. It is also important to note that approximate matching is far more challenging than exact matching. Another significant problem parameter is whether matching of complete objects against each other (*global matching*), smaller objects against parts of larger objects (*semi-local matching*) or parts of objects against parts of objects (*local matching*) is necessary. Semi-local and local matching are inherently more complex than global matching. It is, for example, quite pointless to compute global invariants and use them as a base for local matching.

Our research originated in the analysis of the 3D structure of viruses. Its goal is the determination of the physical structure of a virus, i.e. the type of protein molecules it consists of and their location and orientation. Since viruses are too large for X-ray crystallography, cryo-EM reconstruction is used to determine their density distribution. This generates a voxel-based representation in which every voxel is assigned a density value. However, due to limited resolution of these density maps, it is impossible to directly derive the location and orientation of hull proteins from this data set. Therefore, the goal of our work was an algorithm that solves this problem. Since we need to match parts of proteins with parts of the virus under rigid motion and have to compensate for measurement errors, we are faced with local approximate 3D matching. It is not semi-local since we only get sufficient information about the surface of the virus to compare surface regions with parts of the surface of proteins. Those surface areas of the proteins in the virus that are within the virus and border other proteins in the virus are impossible to directly detect in the experimental data of the virus. Figure 1 shows 3 instances of a cut through a viral density map. The left is the map itself, in the center a protein is marked and in the right cut the surface of the protein which

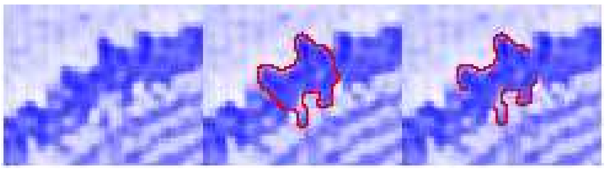


Fig. 1. 3 copies of a cut through the density map of a virus with a protein and the surfaces coinciding with the surface of the virus marked in the center and right cut.

S. Burkhardt is with the Max-Planck Institute for Computer Science, Stuhlsatzenhausweg 85, D-66123 Saarbrücken, Germany and visiting the University of Helsinki. E-mail: stburk@mpi-sb.mpg.de

K. Fredriksson is with the University of Joensuu, Department of Computer Science, P.O. Box 111, FIN-80101, Finland, E-mail: Kimmo.Fredriksson@cs.joensuu.fi

T. Ojamies and E. Ukkonen are with the University of Helsinki, Department of Computer Science, P.O. Box 26 (Teollisuuskatu 23), FIN-00014, Finland, E-mails: {Petri.Ojamies, Esko.Ukkonen}@helsinki.fi

J. Ravantti is with the University of Helsinki, Institute of Biotechnology and Faculty of Biosciences, P.O. Box 56 (Viikinkaari 5), FIN-00014, Finland, E-mail: Janne.Ravantti@helsinki.fi

Supported by the Academy of Finland grant 201560

coincides with the surface of the virus is marked. Only the portion of the protein's surface marked in the right cut can be used to detect similarity and thereby assign this location and translation of the protein a high similarity score.

In this paper we present an algorithm based on geometric hashing techniques for solving this problem. While relatively slow, it produces good results for a test dataset obtained from public databases (EMDB [1] and PDB [2]).

II. Algorithm

The algorithm we developed is faced with two different data representations. While the virus is described by a 3D-density map, the proteins we want to detect in it are given as sets of point clouds describing the spatial arrangement of their atoms, for example as PDB files. Therefore we convert the PDB representation of the protein into a 3D density map using a tool called EMAN [3]. Another such tool is contained in the SITUS package [4]. The scale of the experimental data set is known and it is possible to generate a density map with the same scale using EMAN. However, a much more complicated problem arises from differences in the density value curves. EMAN does not produce density values that are directly comparable to the experimental data. We therefore decided to use a feature based approach for comparing the two data sets. Feature based methods have two main advantages over direct methos. They are relatively density scale invariant and reduce the amount of data that needs to be processed by the matching algorithm.

Previous work has already shown that edge detection can be used to improve quality and efficiency of matching [4]. In principle this means computing gradients for all voxels and considering all voxels above a user-defined threshold as edges/surfaces. The sets of these voxels are then treated as point clouds that represent the virus' and the protein's surface. We use an algorithm by Monga et al. [5].

EdgeDetect($DensityMap$)
$EdgeCloud \leftarrow \emptyset$
for $p \in DensityMap$
  do $G \leftarrow$ Gradient(p)
    if $G \geq Threshold$
      then $EdgeCloud \leftarrow EdgeCloud \cup p$

The actual amount of data reduction depends on the surface of the molecules. For a sphere enclosed in a cube with n voxels on each side the point set generated using edge detection would be of size $O(n^2)$. For roughly spherical objects with an uneven surfaces the exponent increases somewhat. For our data sets we observed values in the range $[n^{2.2}, n^{2.6}]$.

After preprocessing the data we want to compare, and converting it into compatible formats, we are faced with the problem of local approximate matching of 3D point clouds. In order to achieve this, we have to determine the rotation and translation which correctly overlays one cloud onto the other.

A classical approach for comparing one point cloud with another is the so-called *geometric hashing* [6]. It is based on partitioning space into *cells* and assigning each point of the target point cloud to its corresponding cell. The partition it uses is the straightforward discretization into voxels. In a second stage, for each point in the query cloud we determine its cell and determine if there is a matching point from the target in that cell. Counting the number of matching points provides us with a method to approximate the similarity of the two point clouds. This technique can easily be extended to matching one point cloud with several others. In this case, for every cell of the partition we have to determine and store the list of point clouds that have a point in the respective cell.

In order to determine the translation and rotation required to match one point cloud to another, we separate the determination of translation and rotation by selecting an arbitrary *base point* in the target point cloud and translating the point cloud to place the base point into the origin. We then generate a set of *rotational samples*, i.e. instances of the target point cloud rotated around the base point, and store this set in a database (see the pseudocode for PreprocessProtein). In order to determine the correct translation we then proceed to select every point in the query as its base point, translate it and the point cloud to place it into the origin and attempt to match it with the database of rotational samples using geometric hashing (see the pseudocode for MatchVirus). In order to reduce the amount of work in this phase we only consider points of the query in a neigborhood N around the base point. The size of this neighborhood is twice the diameter of the protein. In essence this moves evaluation of all rotations into preprocessing, while the evaluation of all translations is done at runtime through the selection of the base point.

```
PreprocessProtein(EdgeCloud)
  Select base point b ∈ EdgeCloud
  for p ∈ EdgeCloud
     do p ← p − b
  for i ← 1 to RandomSampleNumber
     do Generate Random Rotation R
          AddToDatabase(R, i)
          for p ∈ EdgeCloud
             do p ← Rotate(p, R)
                  s ← Cell(p)
                  AddToSampleList(s, i)
```

```
MatchVirus(EdgeCloud)
  for p ∈ EdgeCloud
     do N ← ∅
          for q ∈ EdgeCloud
             do if Distance(p, q) ≤ ProtDiam
                     then N ← N ∪ q
          for q ∈ N
             do s ← Cell(p)
                  for r ∈ SampleList(s)
                     do VoteForHit(q, r)
  HitList ← SortHitsByVotes()
```

Generating the rotational samples in 3D-space is non-trivial. Our first approach was to use evenly distributed sets of the three rotational angles, however, this leads to a very uneven sampling and is highly inefficient since some areas are heavily oversampled. We therefore used a simple, more promising approach, *random sample generation* [7]. Deterministic sample generation with good properties is challenging and application of existing results could be a topic of further research.

Comparison of points from a query point cloud with a rotational sample that is off by a few degrees of rotation in one or more angles but shares the same base point is faced with two types of error: *rotational* and *radial error*. While the radial error (i.e. the difference in distance from the respective base points) is independent of the angular difference between the point clouds, this is not the case for the rotational error. Here the absolute error value linearly increases with the distance of a pair of points from their base point. The classical partition of space into equally sized voxels does not address this phenomenon. We therefore decided to implement a more advanced spatial partition, the *cube-based partition*. First we divide space into 6 pyramidal cones by centering a cube around the base point of a point cloud and defining the cones through the base point and the 4 corners of each of the faces of the cube. We then proceed to lay a rectangular grid over the faces of the cube by shooting rays with equal angular spacing from the base point through the face of the cube. This approach results in smaller cones which are not fully symmetrical anymore but still of roughly similar shape and size. We then divide these cones into intervals with respect to the distance from the base point. This provides us with a spatial partition that allows for a constant rotational *and* a nearly constant radial error since the angle covered by all the cells in one cone remains constant independent of the distance from the base point and only varies slightly between different cones. Figure 2 contains a cut through a cube-based partition that divides each face of the cube into 5 by 5 cones.

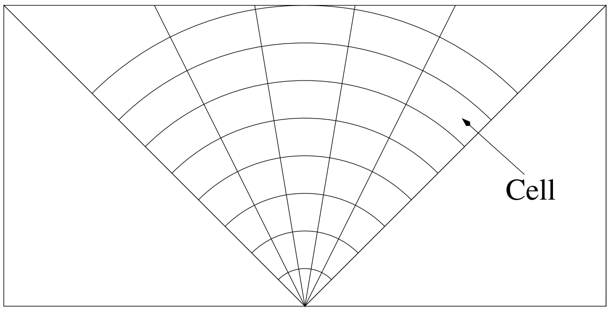


Fig. 2. A vertical cut through the center point of a half-cube showing the cube-based partition of its upper face into 8 radial intervals and 5 by 5 cones.

The use of a single designated base point for a target point cloud can lead to problems in the case of local matching. Since it is possible that a base point from one point cloud is missing in the second point cloud, we could miss a match if no other points are sufficiently close. An example for such a base point would be one that is marked in the center cut of Figure 1 but not in the right cut. A simple solution to this problem is the use of two or even more base points. This leads to a linear increase in the number of sample point clouds in the database and increases the probability of a common base point. If we assume that a fraction m of the surface of the protein coincides with the surface of the virus, the probability of missing a match when using just one base point is $1 - m$. When using n base points, it is $(1 - m)^n$.

So far we have not described in detail the counting scheme used for determining the degree of similarity in the two point clouds. In geometric hashing, this process is often called *voting*. For the trivial spatial partition into sufficiently small voxels it is guaranteed that there will be at most one point from the query and any rotational sample per cell. This property significantly simplifies the voting process. For each point in the query there can be at most one point in the target point cloud so we simply have to loop over all query points and count the number of matching points in their respective voxels. However, with larger voxels or when using the cube-based spatial partition, it is possible that two or even more points of a point cloud are within the same cell. In this case the voting algorithm becomes more complex. Assuming there are q points of the query point cloud and t points of the target point cloud in the same cell, the trivial implementation would be a loop over all q points that increases the number of matching points by t in each iteration. This would lead to $q \cdot t$ votes, clearly an exaggeration. The cleanest implementation would be to count $\min(q, t)$ matching points. However, this requires slightly more computation. In our current implementation we use a compromise and count q matching points if $t \geq 1$. Since we typically use relatively small cells, the exaggeration in the number of votes caused by this simplification turned out to be acceptable.

A further problem we have to handle is the occurrence of duplicate matches. Our algorithm generates a list of hits, i.e. pairs of base points and rotational samples which are sorted by their votes. In case we have two or more very similar hits, we have to detect and remove all but one of them. A second source of duplicate matches is the use of multiple base points. In this case, each occurence of the pattern typically matches at least one rotational sample for each base point. We use a simple greedy algorithm to remove duplicate matches from this list. It generates an empty 3D image the size of the virus and places the best matching protein in the list in the 3D image and deletes it from the list. It then proceeds to greedily select the best matching protein and attempts to place it into the 3D image. If there is a significant amount of overlap with proteins already in the image, the match is scrapped, otherwise it is added to the image. In principle, the list of matches contains all possible translations and rotations, but processing them would be far too time-consuming. We decided to use a user-defined maximum list length to limit the number of candidates that have to be processed with the duplicate removal algorithm.

```
DuplicateRemoval(HitList)
ResultMap ← Empty Density Map
for h ∈ HitList
    do if h ∩ ResultMap ≤ MaxOverlap
          then ResultMap ← ResultMap ∪ h
```

III. Algorithm Analysis

The naive algorithm to solve our problem would consist of translating the protein into each possible position in the virus, rotating it into a number of rotational samples and comparing each instance generated in this way with the virus. Its runtime could be described as:

$$O({n_{virus}}^3 * RS * {n_{protein}}^3)$$

Where n_{virus} and $n_{protein}$ are the edge lengths in voxels of cube around the virus and the protein and RS is the number of rotational samples used. While the translations would be relatively cheap, the rotations of the protein involve large constants.

Our approach has a somewhat different complexity. We select every edge point in the virus as base point. The number of such points is roughly ${n_{virus}}^{2.6}$. For each base point we determine the edge points in its neighborhood in time $O({n_{protein}}^3)$. We then access the sample lists of all these $O({n_{protein}}^3)$ points. The expected length of a sample list can be computed as:

$$SL_{len} = \frac{RS * {n_{protein}}^3}{6 * FN * n_{protein}}$$

Where RS is again the number of rotational samples used and FN is the number of faces we subdivide a cube in the spatial partition into. Due to our use of edge detection this value drops (for our data) to:

$$SL_{len} \approx \frac{RS * {n_{protein}}^{2.2}}{6 * FN * n_{protein}} = \frac{RS * {n_{protein}}^{1.2}}{6 * FN}$$

For decent results it is adviseable to select $FN = s^2, RS = s^3$ where s is a parameter describing rotational sampling resolution. Due to our spatial partition's independence of angular error this parameter is independent from the size of the objects density maps, i.e. from n_{virus} and $n_{protein}$. This leaves us with a total runtime of

$$O({n_{virus}}^{2.6} * {n_{protein}}^{3} * \frac{RS * {n_{protein}}^{1.2}}{6 * FN})$$
$$= O({n_{virus}}^{2.6} * {n_{protein}}^{4.2} * s)$$

This results in significantly smaller runtimes in practice. The reasons are threefold:

- Problem size reduction through edge detection
- Use of the cube-based partition which allows the use of a constant number of rotational samples independent of the size of the density maps
- Precomputation of the expensive rotational sample generation

The combination of these factors allows the solution in a limited time frame for the problems we are currently facing.

IV. Alternative Applications

While our algorithm was designed for the specific problem of matching proteins in viral structures, it is quite generic and could be applied to other problem instances in computational biology and beyond. We are currently starting to use it to match proteins and other molecules in cryo-EM density maps of larger complexes. A good example for this application is given in work by Rath et al. [8]. They present an algorithm targeted at a problem similar to ours. It is based on computing cross-correlation coefficients between query and target using FFT (fast Fourier transform). They use it to locate ribosomal proteins and RNA in a (unfortunately unpublished) cryo-EM map of the ribosome 70S from *E. coli*. The main difference to our problem is that the query occurs only once in the target.

V. Experimental Results

To evaluate our algorithm in a practical setting, we selected `PRD1`, a virus for which the location and arrangement of the hull proteins is already known [1], [9]. Figure 3(b) shows a cut through its cryo-EM density map. The virus' hull consists mainly of the protein `P3`, a total of 720 copies of the protein arranged in a regular, symmetric fashion. All instances of the protein exist in a trimer, a compound of 3 instances of the protein. We therefore decided to use the `P3`-trimer as input instead of just one copy of the protein and match it against the 3D density map of `PRD1` to find its 240 instances in the hull. Figure 3(a) shows a 3D model of the trimer.

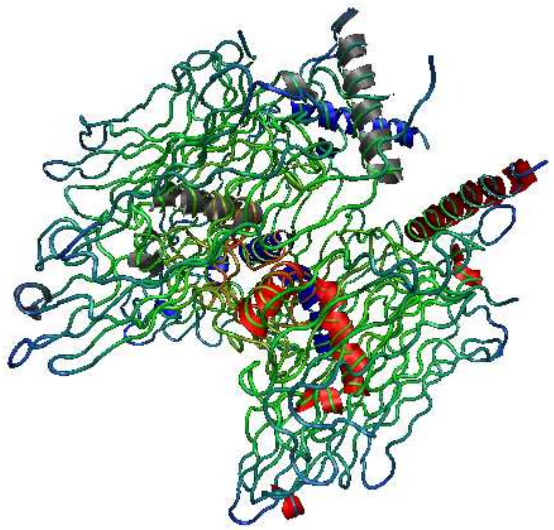
The cryo-EM density map of the virus has a resolution of 256 voxels in each dimension for a total of roughly 16 million voxels. Edge detection reduced this data set to a point cloud with 595,113 points. For the `P3`-trimer we generated a density map with 72 voxels in each dimension (373,248 voxels) which was then transformed into a point cloud containing 2086 points using edge detection. Figure 4 shows slices of the virus and protein data after edge detection.

We used 8000 rotational samples of the `P3`-trimer and either one, two or three randomly selected base points. We subdivided each side of our cube-based partition into 19 times 19 faces. Table I contains the number of successfully detected proteins and the runtime of our algorithm on a 2.66 GHZ Pentium 4 with 1 GB of RAM. The number of matches is based on a visual comparison of an overlay of the cryo-EM map of the virus and the assembly of the trimers generated by our algorithm. In each run we processed the 20000 best candidates with the duplicate removal algorithm. This explains why there are actually less matches for the case of 3 base points since there were roughly three times as many duplicates.

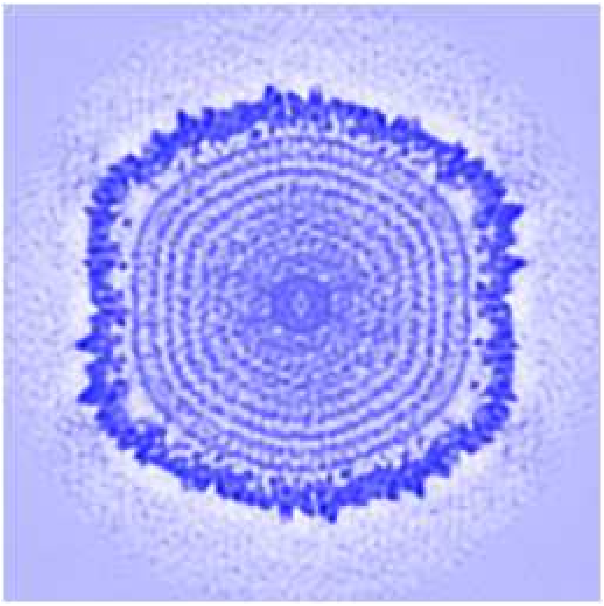
TABLE I
EXPERIMENTAL RESULTS FOR OUR IMPLEMENTATION COMPARING `PRD1` WITH THE `P3`-TRIMER.

face-partitions	base points	matches	runtime
19 x 19	1	230	9.0 h
19 x 19	2	239	15.1 h
19 x 19	3	235	20.0 h

Figure 5(a) shows a slice of the reconstructed 3D image generated by the duplicate removal algorithm while using two base points. There are 5 obvious holes in the hull of the virus. The four on the corners are positions of proteins different from `P3`, the fifth is the position of the one protein we were unable to detect with our algorithm. Figure 5(b) next to it shows an overlay of the reconstruction and the



(a) Rendering of the `P3`-trimer.



(b) Slice of the cryo-EM map of `PRD1`.

Fig. 3. The input data for our experiments.

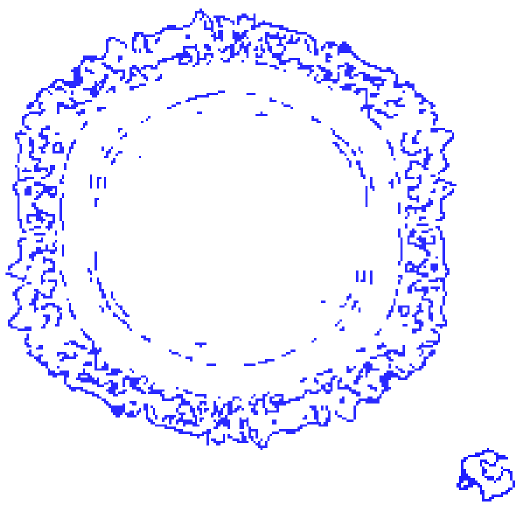
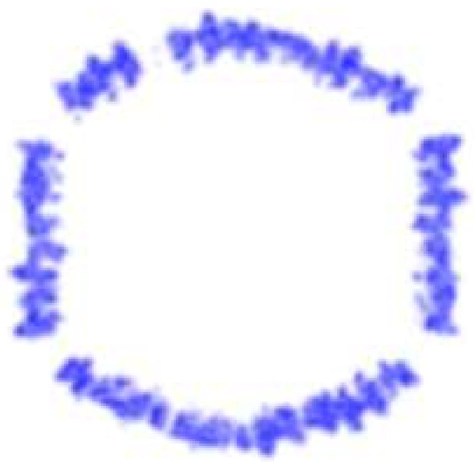


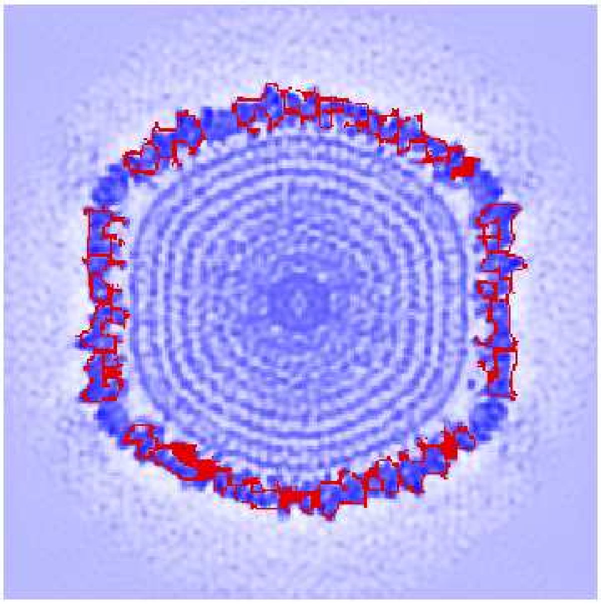
Fig. 4. The point clouds after preprocessing. The trimer is situated in the bottom right corner.

cryo-EM map which illustrates the positioning of the proteins in the hull.

Apart from this experiment we also applied our software to one of the problems in [8]. Given proteins in PDB format as queries they attemp to locate them in a cryo-EM density map of a larger complex, the *E. coli* ribosome 70S. Unfortunately their cryo-EM data set is not publicly available so we generated an artificial data set with EMAN. This of course led to nearly non-existent errors since we were matching two density maps generated by the same program from the same structure. Nevertheless it is worth mentioning that our algorithm was able to detect the correct location of a sample protein in 5 minutes. It remains to be seen, whether this will still work with the noisy experimental data set.



(a) Slice from the image using the 239 trimers found in the original map.



(b) Overlay of reconstructed image and the original cryo-EM map. The red areas are points from the protein point clouds.

Fig. 5. Results of the 2 base point run of our algorithm.

Our software can be downloaded at `www.cs.helsinki.fi/u/ojamies/gesh`.

VI. Conclusion

The algorithm and its implementation presented in this paper represent a pragmatic solution to a practical problem from computational biology. While the usefulness of the current algorithm is somewhat limited by its runtime, it is nevertheless a powerful tool for instances where the number of potential hull protein candidates is limited. In the future a faster screening approach to run on larger protein databases could be an interesting option.

As a side note, we would like to point out, that the spatial partition and the generation of rotational samples are very closely related. In principle, a spatial partition could also be used to generate rotational samples. We are currently investigating potential benefits of using the same approach for rotational sample generation and spatial partition. For example, we plan to use the cube-based partition to generate rotational samples around the central rays of each of the cones defined by the partition.

An inherent problem of any spatial partition is the discretization of space which can cause errors when matching points. While unlikely, it is quite possible, that points that are very close to each other end up in different cells of space. This problem could be alleviated by processing some or all neighborhoods of each cell that we access. It remains to be seen, whether the improvement of accuracy would be worth the increased runtime. This research would be closely related to possible improvement of the spatial partition used and the rotational sample generation.

Further possible improvements could involve a clever selection of the base points, i.e. according to chemical/physical properties of the proteins.

References

[1] M. Tagari, R. Newman, M. Chagoyen, J. Carazo, and K. Henrick, “New electron microscopy database and deposition system,” *Trends in Biochemical Sciences*, vol. 27, no. 11, p. 589, 2002.

[2] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, “The protein data bank,” *Nucleic Acids Research*, vol. 28, pp. 235–242, 2000.

[3] S. J. Ludtke, P. R. Baldwin, and W. Chiu, “EMAN: Semiautomated software for high-resolution single-particle reconstructions,” *Journal of Structural Biology*, vol. 128, no. 1, pp. 82–97, 1999.

[4] W. Wriggers, R. A. Milligan, and J. A. McCammon, “Situs: a package for docking crystal structures into low-resolution maps from electron microscopy,” *Journal of Structural Biology*, vol. 125, no. 2-3, pp. 185–195, 1999.

[5] O. Monga, R. Deriche, G. Malandain, and J. P. Cocquerez, "Recursive filtering and edge tracking: two primary tools for 3D edge detection," *Image and Vision Computing*, vol. 9, no. 4, pp. 203–214, 1991.

[6] Y. Lamdan and H. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *Proceedings of the International Conference of Computer Vision*, 1988, pp. 238–249.

[7] J. Arvo, "Fast random rotation matrices," in *Graphics Gems III*. Academic Press, 1992, pp. 117–120, 463–464.

[8] B. Rath, R. Hegerl, A. Leith, T. Shaikh, T. Wagenknecht, and J. Frank, "Fast 3D motif search of EM density maps using a locally normalized cross-correlation function," *Journal of Structural Biology*, vol. 144, pp. 95 – 103, 2003.

[9] C. S. Martin, R. M. Burnett, F. de Haas, R. Heinkel, T. Rutten, S. D. Fuller, S. J. Butcher, and D. H. Bamford, "Combined EM/X-ray imaging yields a quasi-atomic model of the adenovirus-related bacteriophage PRD1 and shows key capsid and membrane interactions," *Structure*, vol. 9, no. 10, pp. 917–930, 2001.