Comvi - Comparative Visualization of Molecular Surfaces using Similarity-based Clustering

Wilhelm Buchmüller, Shoma Kaiser, Damir Ravlija, Enis Spahiu

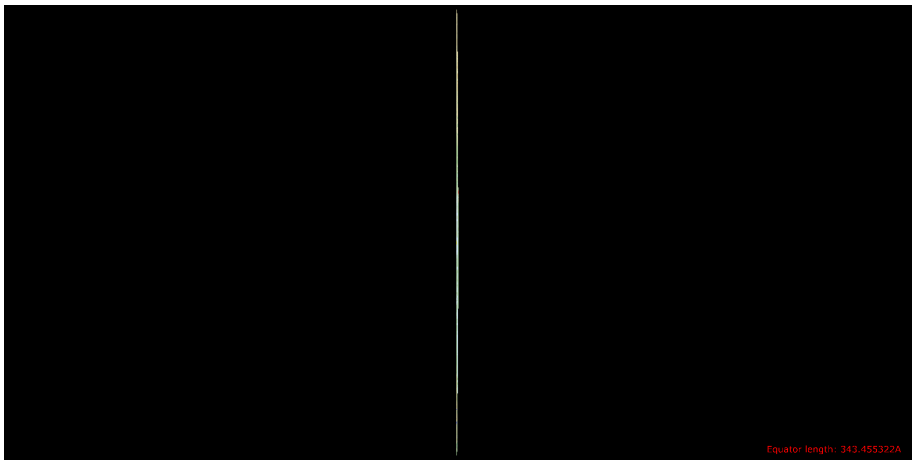

Figure 1: Screenshot of a running MegaMol/MSMCLUSTER/comvi instance

Abstract— The goal of this paper is to show the reader the abstract methods and concrete applications that were used to cluster similar looking molecular maps of proteins. The maps that were provided by Krone et al. [8] represent the topology of an protein surface with
extract and compare features and rank the similarity of the molecular protein maps. Further we present a new method of how the won data can be visualized on high resolution and large displays with dynamic interactions. The paper describes the process and the approaches that were taken to solve this task.

Index Terms—Clustering, Similarity, feature extraction, Visualization, high-resolution display, Powerwall,MegaMol, VISUS

1 Introduction

Over the span of 6 months we, the authors of this paper, have researched and implemented a comparative clustering process for images of molecular maps. The task consisted of several parts: To generate images with the existing `molecularmaps` plugin for MegaMol [1] to build up a dataset, to extract a strong feature vector from the image, to find a way to cluster similar feature vectors and to visualize the clustering on the Powerwall with a tracking device.

This work was based on the MegaMol[1] project. MegaMol is a simulation tool developed by the Universität Stuttgart and the TU Dresden. It can be used to visualize particle data, simulations on atomic scale and other molecular processes such as the one that were presenting with our work. Due to its modular nature, it can be extended with modules to either build standalone plugins or build modules that interact and interface with each other. In this paper we guide you through the `MSMCLUSTER` plugin, its capabilities and inner workings.

One tasks was to retrieve molecular image data through existing MegaMol plugins [8]. For this a special binary of megamol was compiled and will be released in a separate project that will be publicly available starting May 25, 2018[1]. The next task was to extract a expressive feature vector from those images and to find a metric to cluster them by similarity.

The last task which was also developed in a plugin in MagaMol was the visualization on the VISUS Powerwall. The Powerwall is a very high definition display [2] that can be used to visualize large data(sets). Due to its size and resolution its possible to display much more information than on a regular consumer grade screen.

The Powerwall also supports a tracker device that can transmit 6 degrees of freedom, so for the interaction step we had more possibilities than with traditional human interface devices (HIDs), if you would exclude devices like smartphones. For this last step we also researched the possible interactions with the unique tracking device which can be used with the Powerwall.

It is clear that the task required from us that we learn how to compare the images, measure the distances between the images, and cluster these images. The given task required that we use a similarity based clustering algorithm.

Initially we were given the choice we could either chose to find similarities and cluster the proteins in the `.pdb` format or given as bitmap image generated by the `MolecularMaps` plugin in MegaMol [8].

Since handling image files which give information about the protein in two dimensions was easier than dealing with the pdb file

Wilhelm Buchmller
buch.willi@googlemail.com
Enis Spahiu
enis.spahiu@hotmail.de
Shoma Kaiser
shoma.kaiser@googlemail.com
Damir Ravlija
st144286@stud.uni-stuttgart.de

Student Project started at 1 December 2017; Manuscript received 31 June 2018; accepted XX June 2018; posted online XX June 2018. For further information about this article, please contact one of the authors.

[1] `https://github.com/aiosin/comvi`
Last retrieved: May 25, 2018

[2] `https://www.visus.uni-stuttgart.de/institut/visualisierungslabor/technischer-aufbau.html`
Last retrieved: May 25, 2018

format which results in three dimensional visualizations we decided to start and carry out the task with the two dimensional approach.

To prevent confusion further down the paper the reader should be familiar with the following two terms: comvi - internal name of the clustering engine that can be used by the MegaMol Plugin
MSMCluster - name of the Megamol plugin that was developed as the main part of the project

Over the course of the next few pages you will learn how we approached these challenges and how we (attempted) solved them. You will find out what worked and what didn't.

2 Related Work

Clustering proteins by similarity or at least comparing individual proteins has been subject of existing work.

The paper [5] already had similar approaches to our results. Kolesar et al. used a 10 dimensional feature vector based on invariant image moments defined by Hu [2]. Hu-Moments are set of invariant moments defined over a two dimensional signal, that give unique features about the the object in the image. More about image moments and the Hu Moments in section 3.3.1.

Another approach for 3D protein data were 3D zernicke moments explored by La et al. in [9]. The approach is basically the same as in Kolesar et al. but La et al. in [9] used Zernicke moments instead of traditional image moments and extended them to three dimensions. More about the comparison of the different methos in 5.1

For further information and more complex approaches to this problem one should consult the Smith Waterman algorithm from Smith and Waterman "Identification of Common Molecular Subsequences"[11]. This work consideres the protein as a sequence of amino acids, rather than extracing two or three dimensional features over a two or three dimensional signal.[3]

3 Clustering

3.1 Approaches to the Clustering-Problem

Right of the start we had several ideas of how we could approach this problem. With the recent trend in machine learning we had a couple of ideas of how we could determine a similarity metric between two images or classify an image into a more usable vector of data.

We ended up using a higher dimensional feature vector described in section 3.2 to compute the similarirty between two protein maps because we didn't manage to train a custom model in the given timeframe, partly due to lack of knowledge in the field of machine learning and neural networks and also due to the lack of labelled data.

If one would have had labelled data one could easily train an autoencoder, SVM or simple neural net (if the number of classes is relatively low).

Our relatively spartan results with a pretrained Imagenet [7][4] model let us to believe that given the knowledge on the subject and humanly or algorithmically determined labeled data (based on known features or descriptors) it shoud be definetly possible for this specialized task to find a machine learning solution using neural networks/autoencoders.

3.2 Finding a feature vector to cluster the images

The challenge of finding a good feature vector was/is to find good features which can give a lot of information about the image. If the extracted image features are bad the dimensionality reduction will give noisy results and the knowledge gained from the clustering will be not that much.

The following procedure after finding/determining/calculating the feature vector for a given image is to apply some sort of dimensionality reduction to project a higher dimensional vector onto a 2D or 3D plane.

This has multiuple advantages. First If the dimensionality reduction works as intended one can easily find out after applying the dimensionality reduction if similar looking items are positioned next to each other.

3.2.1 Curse of dimensionality

Alternatively one could also find a similarity measure for higher dimensional vectors, but this problem has turned out harder than it seemed partly due to the *curse of dimensionality*[5]

As we all know in higher/infinite dimensional spaces, otherwise unexpected things start to happen such as the euclidian distance or mathematically put the L_2-norm loses relevance since

$$\lim_{n\to\infty} x^n = 0 \quad \forall x \in [0,1) \tag{1}$$

Simply put, otherwise very dissimilar values get pushed to zero. For example between a vector with 0.99 in every dimension and a vector with 0in every dimension the eucludian distance is: $0.99^{100} = 0.33660$ which would put them closer than they really are.

Feature vector
So we have to come up with other solutions to this problem discussed in section **??**

After looking at a small subset of molecular map images from a variety of proteins we determined that we needed to extract feature information about the follwoing properties of a given image:

- Color distribution
- Shape
- Texture
- Invariant image moments

The initial idea was that the color distribution gives information about the color palette in the image, the extracted shape features should give information about the the biggest n shapes in the picture, the texture feature should differentiate between smooth and rough and coarse texture and everything in between.

The image moments were chosen as a goto approach to extract invariant features from the image which has been proved to yield valid results as described in Kolesar et al. [5]

The exact image features that we extract from the image are the following:

3.3 Image Features/Descriptors

3.3.1 Invariant Image Moments by Hu

The set of invariant Image Moments discovered by Hu et. al are rotation, translation, scale and trasformation invariant. This allows us to determine if an imageI_a is similar to another image I_B if I_B is equal I_A and simply rotated by 30.

The continouus Image, defined over the domain $\mathbb{R}^2$ Moments over two dimensions at their core are defined as such:

$$M_{p,q} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dydx$$

Where $f(x)$ is the signal of the image.

When dealing with discrete values like we find them in an discretized RGB/ greyscale (GS) Image we use sums insteads of integral so we get this:

[3] An overview can be found here:
https://www.ebi.ac.uk/Tools/sss/
Last retrieved May 25, 2018

[4] https://pjreddie.com/darknet/imagenet/
Last retrieved: May 25, 2018

[5] https://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/tutorials/dimensionality.pdf
Last retrieved: May 25, 2018

$$M_{i,j} = \sum_{i=0}^{n} \sum_{j=0}^{m} x^i y^j f(x,y)$$

Hu further defines his invariant moments as such:

Kolesar ended up using a collection of 10 experimentally determined moments to cluster the image. In our case the moments we used the standard Hu et al. invariant moments.

3.3.2 Color palette/histogram

The goal when extracting color palette was to reduce the big color space that is present in any of the molecular maps and get a few distinct colors that yield a lot of information from that range.

To achieve this we extract a histogram of each color channel of the RGB images. Each channel is then represented as a greyscale image. We then create for each channel a histogram of the luminance intensities with in our case 16 bins. This "reduces" the $128 * 128 = 16384$ dimensions to just 48 dimensions for our image.

An alternative approach we had to extract color information about the image is to split the image in to equally sized chunks and to extract simple color distribution features like average, min and max values.

The results from both approaches will be discussed in **??**

3.3.3 Texture

TODO: better introduction After looking at a toy dataset of molecular maps we noticed that many samples had a distinct roughness that looked like they could be used to classify their texture.

We ended up using the haralick textural features. The haralick features work with a grey level correlation matrix (GCM). The GCM for a given greyscale image is defined as such:

$$C_{\Delta x,\Delta y}(i,j) = \sum_{x=1}^{n} \sum_{y=1}^{m} \begin{cases} 1, & \text{if } I(x,y) = i \text{ and } I(x+\Delta x, y+\Delta y) = j \\ 0, & \text{otherwise} \end{cases}$$

Also we can take more features from that GCM which gives us more features to work with **TODO:** expand.

we also computed this on every channel yielding us another $3 \cdot 14$ features.

Alternatively one can also use the tamura texture features. the tamura features are different to the haralick featues because they **TODO:** expand tamura

3.3.4 Shape

We did not end up using the shape features described here but, since we spent quite some time researching this problem, I think it is important show how and why they were created.

The shape features we hoped to use were fourier descriptors[6] [7]. In short, we get the centroid-distance curve of a distinct shape and take the fast (discrete) fourier transform of that curve. The technique has been proven to work for shape recognition in earlier works [3].

Our approach was the following. Since our molecular maps are extremely complex (image-wise) we first need to divide the image into discrete image region. We preferably want to segment our image into n colors. To recap: our everyday monitor does support 24 bit colors, 8 8 bit per channel. that yields us $2^{24} = 16777216$ colors. We'd be lucky to find a countour in this color mess. So our approach is to squash the color space down to a couple of colors. This proces is called color segmenatation. We use the k means algorithm to put every

Alternatively the mean-shift algorithm can be used. The algorithm is a kernel based clustering algorithm which operates sort of like the gradient descent algorithm. Each point has a weight and initally all the points are laid out on a grid (or your dimensions next best spatial representation) each value attracts with a constant force all other neighboring values so they all suck each other in discrete clusters depending on the parameters (bandwith size and kernel weight). A comparison of color segmentation of mean shift and k means can be seen here[8].

But our problem of color segmenatation was stil not solved. Especially in the k means algorithm we encountered a lot of smaller "irrelevant" shapes that we didnt want. So one way of getting them out of the way was to squash the color space further down to two or three distint colors in the image.

Else we'd have to dynamically dilate and erode [9]the image until we can detect with a matrix labeling algorithm / floodfill algorithm that we really have only a handful of shapes in the image.

For the shapes we then extract the contours and compute the centroid of that shape. The centroid is the geometric center of mass of the image which can be built up by the first central moments of the two dimensions.

Now we have the centroid and the countour of the shape that we want to classify we need to form the contour centroid distance curve. The contour centroid distance curve is simply a disrete list of values of the $x$and y values of each item of the contour or alternatively the list of the euclidian distances from the centroid, both approaches have been shown to work.

We then normalize the values by diving all the values by the biggest value in the array, resulting in a array that is in range(0,1]

Without loss of generality depending on which curve we chose we take the fast discrete fourier transform of that signal, which gives us for each sine/cosine coefficient a weight how that component contributes/weighs in the signal.

Without any proof that would go beyond the scope of this paper we can say that the gained features are rotation scale and translation invariant. Its rotation invariant because the Fourier transform is shift invariant so it doesnt matter where we begin our centroid curve the fourier transform will be the same even if its rotatted. Its scale invariant because we normalize it, its translation invariant because the distance curve is trivially translation invariant (wihtout proof).

Another shape feature can be won, under the condiciton that the shape object is given as a greyscale binary matrix. From this matrix one can extract Image Moment features like the one by Hu as described in section 3.3.1.

We did not end up using the shape features, because we could notice any improvement over the other features that we used, but we put substantial effort into getting these features to work, so we wanted to describe our process here.

3.4 Finding the best performing similarity measure

After our feature processing on an image we end up with an feature matrix where the rows are samples and the columns are the features.

$$A \in \mathbb{R}$$

TODO: remove (?)

3.5 Findin the best performing dimensionality reduction method

For testing purposes we used every dimensionality reduction method we could find that looked halfway promising. We tested on datasets of various sizes and content.

[6] http://fourier.eng.hmc.edu/e161/lectures/fd/node1.html

[7] http://demonstrations.wolfram.com/FourierDescriptors/

[8] https://spin.atomicobject.com/2015/05/26/mean-shift-clustering/
Last retrieved: May 25, 2018
https://spin.atomicobject.com/2016/12/07/pixels-and-palettes-extracting-color-palettes-from-images/
Last retrieved May 25, 2018

[9] https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html
Last retrieved May 25, 2018

We tested our results on the Oxford flower dataset and bmw dataset and our large (3000 images) molecular maps dataset and a medium version of that dataset (300 images) and a small dataset for fast testing purposes.

As you will find in our comvi repository [10] the dimensionality reduction routine consists of passing it a features array of shape $A^{n\times m}$ where n is the number of samples (i.e. the number of images) and m is the number of features per image. In the routine we additionally scale the data because we want to avoid potentially unwanted effects caused by very big values.[11] We then return the reduced feature array back to the callee. The callee can then apply further some functions on this reduced array.

After inital testing with linear dimensionality reductions we notice that our results just wouldnt "converge", so we again tried with any non linear dimensionality reduction technique we could find.

We tested our data with Local Linear Embedding (LLE), Multi-Dimensional Scaling (MDS), Kernel-PCA, Isomap and t-distributed Stochastic Neighbor Embedding (t-SNE). After testing with a small non representative user group that would judge how good the clustering was done by the technique we concluded that t-SNE performed the best.

Further testing leads us to believe that t-SNE is the best dimensionality reduction procedure for this problem. t-SNE is discussed in more detail in section 3.6

3.6 Clustering - Algorithms and strategies

Clustering of data is defined as the process of grouping n distinct values into m different classes. Clustering algorithms are algorithms that perform clustering. They can be subdivided in two different ways [10]. Because the given data was not labeled, only unsupervised machine learning techniques, like clustering, could be applied on it.

Depending on the number of classes to which one data element can belong we differ between hard and soft clustering. In hard clustering one element can belong to only one class, whereas in soft clustering it can belong to many different classes. On the other hand, depending on the way in which the data is clustered we differ between flat and hierarchical clustering. In flat clustering particular clusters have not relation to each other, whereas in hierarchical clustering considers the distance between clusters as well.

We applied the following clustering algorithms to image descriptors that were extracted from the protein images.

k-Means takes the number of clusters as an input and if the data set contains more elements than the wanted number of clusters groups elements into that many clusters. Since it always groups data into the specified number of clusters k-Means is a flat clustering algorithm. Owing to the fact every image is clustered into only one cluster it belongs to hard clustering algorithms.

To cluster the data k-Means uses the concept of centroids. Centroid is a point in vector space that is located in the middle of the corresponding cluster. There is therefore one centroid for every cluster. At first k-Means initializes the same number of centroids the same to the number of clusters. This can be done by choosing randomly that many points in the vector space. Since k-Means is an iterative algorithm it it then repeats reassignment and recomputation steps until it minimizes the distance between the centroid and all of the elements of the corresponding cluster as follows.

Let $D = \{x_i'\}_{i=1}^{n}$ be a set of n vectors x_i that represents some data. This data should be clustered into k clusters whose centroids are $\mu_{c(i)}$. Now let $c : i \mapsto k$ be the assignment of the element i to the cluster k. K-Means minimizes the following loss function L:

$$L = \min_{c,\mu} \sum_{i=1}^{n} (x_i - \mu_{c(i)})^2 \tag{2}$$

This minimization is done by repeating the (4) reassignment and (5) recomputation step.

$$\forall i : c(i) := \arg\min_{k} (x_i - \mu_k)^2 \tag{3}$$

$$\forall k : \mu_k := \frac{1}{|c^{-1}(k)|} \sum_{i \in c^{-1}(k)} x_i \tag{4}$$

The main weakness of k-Means is that it does not converge to a global minimum. In order to circumvent this problem centroids are usually initialized randomly, but this then makes the algorithm non-deterministic. To find the best possible clustering peforming several restarts of the algorithm is sometimes needed. Its main advantage on the other hand are that it often gives good results and is easy to setup. It is also efficient because its time complexity is linear in the number of elements, number of clusters and number of iterations.

MSMCluster plugin contains two implementations of k-Means. We implemented one version and the other version comes from the clustering part of the dlib library [4].

Hierarchical agglomerative clustering is a type of hierarchical clustering that runs bottom up.

Mean-shift

3.7 Finding the best performing clustering algorithm

Our approach to finding a clustering algorihtm was the same as the one that we used to find the best performing dimensionality reduction algorithm.

Except with a little modification. Kolesar et al. [5] used the k nearest neighbors algorithm over their 10-dimensional feature vector to cluster their data, as they discuss their results. We decided to further use this approach to cluster our molecular maps instead of the molecular protein tunnels which Kolesar et al. analyzed.

In the early stages of the projcet we intended to use the DBSCAN clustering algoirtm. The key feature to this algorithm is that it automatically determines the optimal number of clusters that it detects in the given data.

The algorithm is a density based spatial clustering algorithm that uses a similarity metric between two items that are to be clustered defined over the domain $[0, 1]$. So the ways of computing the similarity is given more freedom with this approach.

You could use the euclidian distance between two feature vectors or the cosine distance or feed the two images to a siamese neural network/autoencoder. Siamese neural network are, for the sake of simplicity two identical black box oracle machines which yield different valued vectors if you feed the two machines two different signals and an autoencoder is a machine which will have learned to classify data into a few distinc categories. The main disadvantages two both of these techniques is that both the siamese neural network and the autoencoder have to be trained.

You could also usethe householder/hausdorff distancebetween two higher dimensional vectors.**TODO:** expand

We opted against DBSCAN for our primary clustering algorithm because in the early stages of the project the computation of the $n \times n$ similarity matrix was very computationally expensive and took quite a bit more time than the approaches that we demonstrate in the next paragraphs.

For our initial testing we used k-nearest neighbors with the elbow method and a cutoff at 85 % **TODO:** elaborate on elbow method

[10] https://github.com/aiosin/comvi

[11] https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e
Last retrieved: May 25, 2018

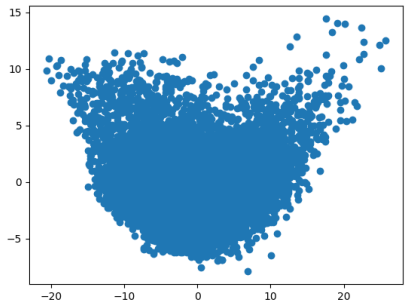

Figure 2: Early test wihh the Oxford flower dataset **TODO:** cite oxford flower dataset

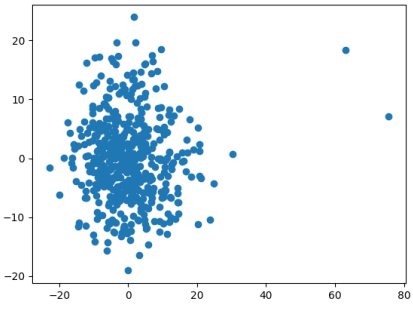

Figure 3: Early test with the bmw car dataset **TODO:** cite stanford car dataset[6]

spectral clustering, chinese whispers **TODO:** put in clustering algorithms descriptions...

We ended up with the mean shift algorithm. It also is a density based algorithm and much like the DBSCAN algorithm is also determines the optimal number of clusters for the given parameters but it operates with a kernel sampled on each point in the plot and calculates the center of mass so to speak of the points clouds[12] . For a demonstration of the mean shift algorithm look at Figure 4.

3.8 Testing the feature vector with other datasets

4 Visualization and Interaction

To show up our algorithm results, we needed a suitable representation and a intuitive interaction for our generated clusters. In favor we used a configuration bar where you can set relevant parameters like the clustering algorithm or the images to cluster. **TODO:** megamol graphic The represented clusters are shown in a square area. This application is finally shown on the POWERWALL in Visus. With a tracking stick it was possible to interact with the clusters on the POWERWALL.

4.1 Background MegaMol and Powerwall

The visualtion of MegaMol is displayed on the Powerwall. The Powerwall is a merged Large High-Resolution Display made of five portraitoriented strips . The setup has a resolution of 10,800 x 4,096 pixels for each eye, and is projected onto a physical screen size of about 6 x 2.2 metres. Two 4K LCoS projectors are projecting the

[12] https://spin.atomicobject.com/2015/05/26/mean-shift-clustering/
Last retrieved: May 25, 2018

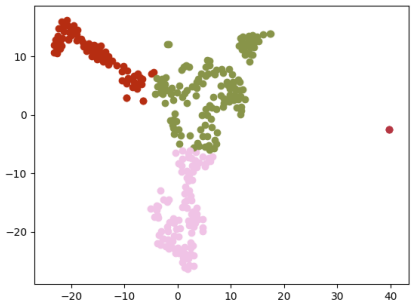

Figure 4: final test with the tsne dimensionality reduciton and the mean shift clustering algorithm

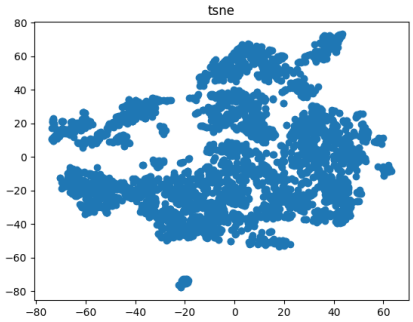

Figure 5: final test with the tsne dimensionality reduciton and the mean shift clustering algorithm

images for each strip. A pixel is about 0.56 mm in size, which corresponds to about 50 ppi . The users interacted with the system using a 6DOF mid-air pointing Stick with two buttons (1 on each side), which was tracked by 14 camers using an optical tracking system from NaturalPoint. All the transformations were synchronised over all views, while the interaction is runnig.

4.2 Goal of Visualization and Interaction

The main goal was to visualize the clusters as clear and seperated blocks in square area. In context to a good clairity it was suitable if we cluster the images into about three to ten clusters. For our subjective feeling it is clear to generate the count of the clusters in this area. Therefore the size of the clusters were adjusted to the square area. For the parameter assignment the configuration bar on the left can be changed by the user. Therein should be found all relevant parameters to manipulate the clustering and the visualization.

To interact with the Application intuitively on PC and on the POWERWALL it was important that we use movements for the interactions from our everyday life. If the Application is used on PC, the MegaMol Project is mainly interacting with the mouse. If we run Application at the Powerwall it is interacting with the tracking stick.

With the four button states of the tracking sticks we focused on the most important interactions like “selection” or “drag and drop” To localize the aimung of the user we should calculate a intersection coordinates of the tracking stick in direction to the Powerwall. While moving the tracking stick for interactions, these coordinates should be scaled for a user- friendly interactions.

Table 1: placeholder comparisons of different protein similarity comparisons/algorithms

*3p3cm dataset	full performance (fps)	half performance (ms)
big (3k images)	1,243	0.1
medium (300 image)	23	23
small (12 images)	23,312,134.3	22.1

4.3 Realization

To realize the ideas and goals we first discussed about the dimension of the cluster visualization. But a couple of reasons spoke up against a 3D visualization so we decided to do transform the 3D molecule into a 2D image. We would have more degrees of freedom to work with 3D molecules on the Powerwall. But we could not find a way to present the data in a way such that with just a glance the user could intuitively interpret the data that would be displayed on the screen. Furthermore the clustering and the interaction would be demanding. So we decided to settle for a 2D visualization. The visualizatoin consists of displaying the image of a cluster representative with a simple rectangle. With clicking into a rectangle the user is able to get a view of all clustered images in that cluster. Even subclustering can be executed by splitting one cluster into two more. After testing we decided that one level of subclustering is enough. After 2 levels of subclustering the clusters get too **TODO:** begriff With the configuration bar the user is able to change several variants of clustering and execute some interactions too.

But mainly the interactions are executed by mouse or by the tracking stick. The tracking stick got As already mentioned four states. In the first state, if no button is pressed, should be no interaction. In the second state, if the first button is pressed, should be a 'drag and drop'-interaction as we know it from the Computer. With pressing the other button, the third state, the user can select an object like we know it from the Computer 'leftclick'. If both buttons are pressed, the user is able to zoom. **TODO:** wenn es klappt For the zooming we used the third dimension of the tracking area in front of the Powerwall. With walking towards the Powerwall in the fourth state the user zooms in. Equally with walking away from the Powerwall the user zooms out. All movements of the tracking stick in the tracking area have to be scaled for a user-friendly usage. To calculate them we tested the interactions exprimentally by changing the scalings. We adjust these parameters so that the user can operate with the whole interface without walking around in the tracking area. As a conclusion of this part we focused on user-friendly and intuitive interactions like we know it from our everyday life.

TODO: where to put this ? We wanted to be able to

1. have a cursor
2. zoom into the picture and explore the different clusters
3. navigate into a cluster
4. display a cluster as a gridview if a) the cluster size allowed this and b)there was enough screen real estate available for this.

5 Discussion of results

As we saw on the last few pages, we have written a plugin for MegaMol that takes pdb datafile or already generated molecular maps as an input and clusters the corresponding individual protein maps by similarity. **TODO:** expand

5.1 comparing our findings with other protein similarity practices

TODO: expand bla [9] protein database simlilarity measure.

6 Conclusion

TODO: expand

Figure 6: Our implementation of k-means applied on the medium-sized data set.

Figure 7: Dlib k-means applied on the medium-sized data set.

Figure 8: Bottom up agglomerative clustering algorithm (dlib) applied on the medium-sized data set.

Acknowledgments

We would like to thank our supervisors Michael Krone and Florian Fries as well as our project examiner Prof. Ertl, for giving us this opportunity to work on this project. We are grateful that we were able to improve our knowledge and learn new things. We are also grateful for the feedback we recieved on our work.
This work was partially funded by cake and cookies.

References

[1] S. Grottel, M. Krone, C. Müller, G. Reina, and T. Ertl. Megamola prototyping framework for particle-based visualization. *IEEE transactions on visualization and computer graphics*, 21(2):201–214, 2015.

[2] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.

[3] T. Karrels. Fourier descriptors for shape recognition, 2006.

[4] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[5] I. Kolesár, J. Byška, J. Parulek, H. Hauser, and B. Kozlíková. Unfolding and interactive exploration of protein tunnels and their dynamics. In *Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 1–10. Eurographics Association, 2016.

[6] J. Krause, J. Deng, M. Stark, and L. Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] M. Krone, F. Frieß, K. Scharnowski, G. Reina, S. Fademrecht, T. Kulschewski, J. Pleiss, and T. Ertl. Molecular surface maps. *IEEE transactions on visualization and computer graphics*, 23(1):701–710, 2017.

[9] D. La, J. Esquivel-Rodríguez, V. Venkatraman, B. Li, L. Sael, S. Ueng, S. Ahrendt, and D. Kihara. 3d-surfer: software for high-throughput protein surface comparison and analysis. *Bioinformatics*, 25(21):2843–2844, 2009.

[10] H. Schütze, C. D. Manning, and P. Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press, 2008.

[11] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.