

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

Wilhelm Buchmüller, Shoma Kaiser, Damir Ravilija, Enis ...

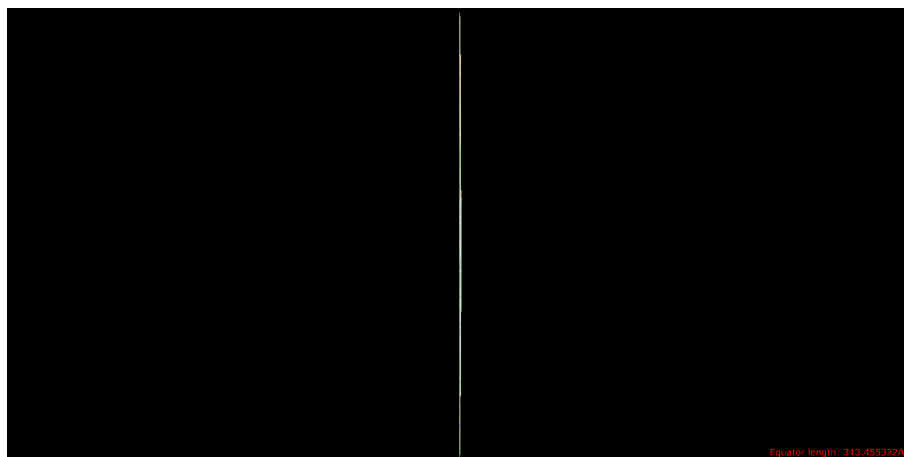


Figure 1: Screenshot of a running comvi instance

Abstract—The goal of this paper is to show the reader the abstract methods and concrete applications that were used to 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 a 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

TODO: add sections to tasks in this section Over the span of 6 months we, the authors of this paper, have researched and implemented a comparative clustering of molecular maps. The task consisted of several parts: This work was based on the MegaMol project. MegaMol is a simulation tool developed by the Universitt Stuttgart and the TU Dresden(?) **TODO: cite megamol**[1]. It can be used to visualize particle data, simulations on atomic scale and other molecular processes. Due to its modular nature, it can be extended with modules to interact with other modules. In this paper we guide you through the **TODO: verbose** MSMCLUSTER plugin, its capabilities and inner workings.

The next task was to retrieve molecular image data through existing MegaMol plugins **TODO: cite molecuLr maps**[6]. For this a special binary of megamol was compiled and will be released in a separate project **TODO: make github link tocomvi public**. 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 that can be used to visualize

large data(sets). Due to its size its possible to display much more information than on a regular screen.

The POWERWALL also supports a tracker device that can transmit 6 degrees of freedom, so for the interaction step we had more freedom to work with than with traditional human interaction devices **TODO: HID abbreviation correct (?)**, for this last step we also researched the possible interactions with the tracking devices on the POWERWALL.

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

2 RELATED WORK

TODO: cite kolesar for clustering Clustering proteins by similarity or at least comparing individual proteins has been subject of existing work.

The paper [3] already had similar approaches to our results. Kolesar et al. used a 10 dimensional feature vector based on invariant image moments defined by hu **TODO: cite hu image moments** [2]

Another approach for 3D protein data were 3D zernicke moments explored by **TODO: 3d surfer** [7]. The approach is basically the same as in Kolesar et al. but the TODO used Zernicke moments instead of traditional image moments and extended them to three dimensions.

2.1 Initial Challenges and encountered problems

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.

Wilhelm Buchmüller Shoma Kaiser
buch.willi@googlemail.com example@example.com
Enis ... Damir Rwilja
example@example.com example@example.com

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

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 **TODO: cite megamol protein image**[6].

Since handling image files which give information about the protein in two dimensions was easier than dealing with the pdb file format which results in three dimensional visualizations we decided to start with a two dimensional approach.

2.2 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 **TODO: give section label** to determine the similarity between two protein maps because we didnt manage to train a custom model in the given timeframe, due to inexperience **TODO: this can be said better** and non existance of labeled data.

But our relatively spartan results with a pretrained Imagenet **TODO: cite imagenet** [5] **TODO: cite darknet publication** model let us to believe that given the knowledge on the subject and humanly labeled data (based on known featuresit **TODO: LEFT OFF HERE** should be definety possible for this specified task to find a machine learning solution using neural networks/autoencoders.

2.3 Approaches to the Visualization-Task

Our approaches to visualizing the given clusters were the following, the reader is reminded that we are not just visualizing the clusters on a "normal machine" but rather the POWERWALL, a projected display with effectively 6-24 times the resolution of a consumer grade display. Details on the POWERWALL can be found **TODO: cite powerwall publicatoin here if available**. [8]

If we are given so much pixel real estate we are given the freedom to draw smaller pixels since we still will be able to see them on the POWERWALL.

During the duration of the project the idea of a 3D visualization was discussed among the team, but we settled for a 2D visualization. This had a couple of reasons.

While the interactions with the data in 3D would have been more fun since we had more degrees of freedom to work with. 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.

... So we decided to settle for a 2D visualization. Our approach is rather boring but it works. On startup we display nothing, if the user chooses his supplied image data and the algorithms used to cluster the pictures he gets the option to start the visualization.

The visualizatoin consists of displaying the image of a cluster representative with a simple rectangle. This way the user knows exactly what to expect to be in the cluster.

If the user wishes to have a closer look at the images in the given cluster he can click onto the representative and will get a view of all the images in that cluster.

We decided after testing with toy test and real datasets that one level of subclustering is enough, after 2 levels of subclustering the clusters get **TODO: find better expression** noisy and ambiguous.

In both the main and subcluster view the representatives are visualized with "force directed layout", to avoid

2.3.1 Approaches to the Interaction with the Powerwall

TODO: cite correct pub and use correct name

TODO: @shoma @enis @damir interaktionsmglichkeiten schreiben

2.4 Finding a feature vector to cluster the images

The challenge of finding a good feature vector was/is to find good features which are **TODO: aussagekrftig** about the image.

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

Another reason is the curse of dimensionality. As we all know in higher/infinite dimensional spaces, otherwise unexpected things start to happen such as the euclidian distance or mathmatically put the

$$L_2 \quad (1)$$

-norm loses relevance since

$$\lim_{n \rightarrow \infty} x^n = 0 \quad \forall x \in [0, 1) \quad (2)$$

Simply put, otherwise very similar values get skewed to zero.

So we have to come up with other solutions to this problem discussed in **TODO: put clustering section here**

TODO: back to finding the feature vector 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 following properties of a given image:

Color distribution, Shape, Texture and 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 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 results as described in [3]

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

2.5 Invariant Image Moments by Hu

The set of invariant Image Moments discovered by Hu et. al **TODO: find out if hu moments just him or others or et al.** are rotation, translation, scale and trasformation **TODO: find out if correct** invariant. This allows us to determine if an image I_A is similar to another image I_B if I_B is equal I_A and simply rotated by 30 **TODO: put the value in degrees there**

The (continous (spelling?) Image) Moments over two dimensions at their core are defined as such:

$$m_{i,j} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) dy dx$$

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

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

TODO: get definition of moment

Hu further defines his moments as such:

TODO: give the definition of all 7/8 moments (?) because this is going to take up a lot of space.

Kolesar ended up using these moments **TODO: put in kolsar moments**, in our case the moments **TODO: put the hu moments here**

2.6 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 **TODO: aussagekrftig** colors 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.

Alternatively you can take another approach described here

The results of both approaches will be discussed in **TODO: cite**

2.7 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:

put the formula here, this is spaceholder

Also we can take more features from that gcm than just the **TODO: what exactly** can be taken from this matrix which gives us more features to work with.

we also computed this on every channel yielding us another x features **TODO: expand texture features**.

Alternatively one can also use the tamura texture features. the tamura features are different to the haralick features because they **TODO: what exactly**. They were primarily researched/developed/created to **TODO: what exactly**.

2.8 Shape

We did not end up using the shape features described here but, since we spent quite some time getting them to work **TODO: rephrase** I think it is important show how and why they were created.

The shape features we used are fourier descriptors. 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 work **TODO: get that bibtex to that presentation on that leaf recognition script**.

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 initially 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 menas can be seen here. **TODO: give image for comparison**

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 **TODO: give link to dilation and erosion** 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 shape or in other areas of science known as the first moment

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 contour **TODO: clafiry** or the list of the euclidian (in two or

three dimensions, why this doesnt work in higher dimensions look up **TODO: curse of dimensionality** distances.

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 whhich 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/weights 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 ... its scale invariant because we normalize it, its translation invariant because the distance curve is translation invariant (wihtout proof).

Another shape feature can be won, if 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 **TODO: give section to hu moments**

We did not end up using the shape features, because we could notice any improvement over the other features that we used, but it is important to note that they exist and what their capabilities are incase someone will continue our work on this topic. **TODO: rephrase**.

2.9 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. aaaaa

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

2.10 Findin the best performign 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.

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 **TODO: give link** the dimensionality reduction routine consists of passing it a features array with **TODO: beautify shape samples*features** In the routine we additionally scale the data because **TODO: why exactly ?!?!?**. We then return the reduced feature array back to the callee. The callee can then apply further some functions on this....

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

We tested our data with LLE, MDS, kernel PCA, ISOMAP and t-SNE. After testing with a small 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 a randomized t bla Stochastic neighbor embedding process. Its works by But because the process requires random variables we decided to set the random seed of our program sow we can get deterministically the same results across different machines.

2.11 Finding the best performing clustering algorithm

2.12 Testing the feature vector with other datasets

2.13 title

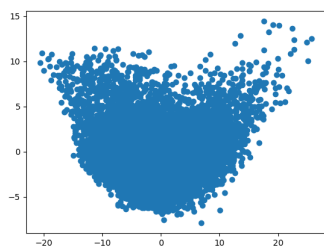


Figure 2: Early test with the Oxford flower dataset **TODO: cite oxford flower dataset**

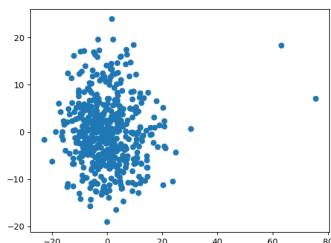


Figure 3: Early test with the BMW car dataset **TODO: cite stanford car dataset**[4]

- [2] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.
- [3] 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.
- [4] J. Krause, J. Deng, M. Stark, and L. Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] C. Müller, M. Krone, K. Scharnowski, G. Reina, and T. Ertl. On the utility of large high-resolution displays for comparative scientific visualisation. In *Proceedings of the 8th International Symposium on Visual Information Communication and Interaction*, pages 131–136. ACM, 2015.

2.14 Finding the

2.15 comparing our findings with other protein similarity practices

Laa di daa bla blubb I am a placeholder aylmao, look at me. bla [7] blah protein database similarity measure.

2.16 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.

2.16.1 Dolor

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 and learn. 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.

Table 1: lorem ipsum tabulated

dataset	full performance (fps)	half performance (ms)
balls	1,243	0.1
buckets	23	23
bolts	23,312,134.3	22.1