Interactive and dynamic web-based visual exploration of high dimensional bioimages with real time clustering

Magnus Rathke[1], Jan Kölling[1], Karin Gorzolka[2], Karsten Niehaus[2], Tim W. Nattkemper[1]

[1]Biodata Mining Group, Faculty of Technology
[2]Proteome and Metabolome Research, Faculty of Biology
Bielefeld University
Universitätsstraße 25
D-33615 Bielefeld
mrathke@cebitec.uni-bielefeld.de
koelling@cebitec.uni-bielefeld.de

Abstract: Web browsers and web applications have become common tools in bioinformatics over the past decades. Many existing web applications revolve around server-client interaction, where heavy computational tasks are often outsourced to the server and the presentation is handled on the the client-side. However more recent additions to the web browser technology embrace the capability of handling more complex operations on the client-side itself, cutting out most of the server-client interaction except for data loading.
This paper contributes to the exploration of the potential of approaches to implement and speed up computational expensive tasks, like image cluster analysis, within a client-side web browser environment. The experimental results, incorporating the well known k-means algorithm which serves as a platform for various parallelization approaches, indicate the possibility to achieve real time image clustering. Especially for the available MALDI-MSI data set the results look promising. Despite good results of multithreading approaches, algorithmic approaches appear to be relevant too. Therefore advancements in accelerating the k-means algorithm itself are considered.

1 Introduction

Imaging systems have become a relevant factor in biological disciplines. Especially sophisticated imaging methods such as Matrix-Assisted Laser Desorption/Ionization Mass Spectrometry Imaging (MALDI-MSI) or Raman imaging, but also the enhancement of traditional optical light microscopic methods through fluorescence microscopy, provide the possibility of visual analysis of biological specimen on a cellular and subcellular level. MALDI-MSI and Raman imaging in particular not only provide high fidelity molecular compound analysis but also acquire regiospecific molecular measurements, which allows a visual representation of tissue biology on the basis of specific molecules, e.g., proteins, metabolites or peptides [NC13].
Although modern imaging techniques are able to give deeper insight into the biological system of an analyzed specimen and greatly contribute to the understanding of biologi-

cal molecules, the evaluation and analysis of such generated data sets can be complex. Multivariate images generated by biological imaging methods usually contain hundreds (multi-spectral) to several thousands (hyper-spectral) images, depending on the complexity of a probed sample and the chosen imaging method [F+13].

To provide the required computational resources for the analysis of complex data sets and to make the results easily available to researchers in different settings, modern analysis and visualization methods often involve server-client applications. The server side of an application handles expensive computational tasks for multivariate data analysis, employing either computing clusters or cloud computing solutions to ensure reasonably fast computation. The client side of such applications handles the visualization of the computational results and offers an interface to the analysis methods. The drawback of server-client applications is asynchronous processing, since it is not ensured that a task submitted to a server is computed immediately. This prevents server-client applications from true interactivity, which is desired especially in the context of data mining and exploratory data analysis [L+11] [HLN11].

Modern desktop computers are able to use extensive computational resources. Due to technological advancements, multi-core processors as well as powerful graphic cards are common assets. Furthermore, advancements in web browser technology enable them to use the provided resources even from within a standard web browser. Modern web browser implementations provide multithreading capabilities and also allow accelerated 3D graphics support by accessing graphic cards. As a result, client side applications with access to more and more computational capabilities of current desktop computers, which are also easy to access and update, are possible.

In the following, the focus is laid on achieving real-time cluster analysis of spectral image data within a client-side web browser environment. This aims at interactive exploration of high dimensional data sets, with respect to different clustering parameters, e.g., number of prototypes, or varying input, e.g., subsets of the data, for visual display of different clustering results (clustermaps) [K+12]. Therefore, it is investigated how much speedup, if any, can be achieved for k-means using the current multithreading capabilities of web browsers. To better evaluate the effects of multithreading and further accelerate the analysis, several established algorithmic enhancements of k-means were considered in addition to a standard implementation. For convenience, a short introduction into the clustering problem with k-means and a definition of the data structure is given in Section 2. Section 3 introduces two approaches to accelerate the k-means algorithm, whereas the first approach is based on using so called *web workers*[1] for parallel execution and the second approach refers to the strategy to reduce the amount of distance calculations needed, by applying the triangle inequality [Elk03]. The experimental results of these approaches are presented in Section 4 and discussed in Section 5. Section 6 summarizes the findings and gives an outlook on improvements for parallel computation using future web browser features.

[1] http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html

2 Data- and Problem Definition

This section presents the clustering problem with the k-means algorithm on multivariate images. Therefore definitions are given for high dimensional image datasets and the k-means algorithm.

2.1 Data definition

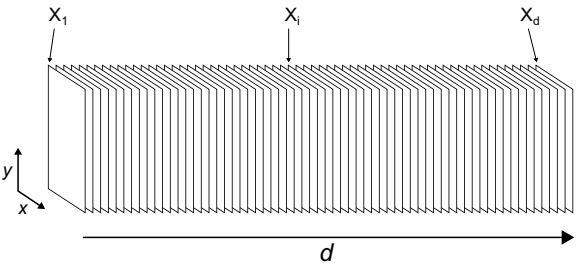


Figure 1: Multivariate image X with dimensionality d. In the context of imaging d is often referred to as the number of *channels*. Each channel represents one image $X_i \in X_1, \cdots, X_d$, with dimensions $x \times y$.

Biological multivariate images generated with, e.g., MALDI-MSI or Raman Imaging combine information on molecular composition and position and thereby enable localization of molecular compounds. The underlying concept is to fixate a biological specimen or a section of one and probe it in a regular order (rasterisation), acquiring a spectrum of information per sampled position. This not only allows the correlation of identified features to a specific location within the sample, but also visualization.

A multivariate image X contains an ordered set of greyscale images $\{X_1, ..., X_d\}$ (see Figure 1) where each image $X_i = (x_d)$ is a $x \times y$ matrix of intensity values. Each image X_i visually represents the spatial distribution of a selected feature i in the original sample. Concatenating all intensity values for a fixed x, y coordinate over all images X_i recovers the original spectrum.

2.2 k-means

The k-means algorithm is one of the most famous unsupervised learning algorithms for cluster analysis [W$^+$08] [Jai10]. Since its publication in the middle of the 20th century, several variations were developed. Lloyd's algorithm is often referred to as the standard k-means algorithm [Llo82].

The goal of k-means is to partition a set $X = x_1, x_2, ..., x_n$ into k subsets ($k < N$). Each subset is represented through a prototype μ_k. These subsets are referred to as clusters $C = c_1, c_2, ..., c_k$ with their respective prototypes μ_k. To partition a set X into k subsets the algorithm locates similarities by minimizing $J(C)$ (see equation 1), the sum-of-squared errors, between each datapoint x_i and its corresponding prototype μ_k of a cluster c_k.

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} \|x_i - \mu_k\|^2 \tag{1}$$

To achieve a local minimum $J(C)$, the algorithm cycles through two steps which can be described as:

1. *Find the closest prototype*
2. *Update prototypes*

The runtime of the k-means algorithm is thereby primarily influenced by the distance calculations, which are performed to find the closest prototype μ_k to a datapoint x_i. Since this action needs to be performed for each datapoint x_i, the runtime is denoted as $\Theta(n \cdot k \cdot d)$, where d depicts the dimensionality of the datapoint and the prototype. The recalculation of the centroids is the second factor in the k-means algorithm. The runtime of this step is denoted as $\Theta(n \cdot k \cdot d)$. However these two parts do not increase cardinality. Thus, the overall runtime of the k-means algorithm is linear in all its factors and is described as $\Theta(n \cdot k \cdot d \cdot i)$, where i denotes the number of iterations.

3 Methods

This section introduces the approaches which were considered to achieve real time clustering in a client-side web browser environment.

3.1 Multithreading in web browsers

Modern web browsers allow JavaScript (JS) web applications to spawn operating system level threads via the **Web Worker**-interface, which enables actual concurrent computing. Applications can thus execute time consuming operations in the background without interfering with user interfaces. Threads invoked by the application execute a JS script, which is called a worker. Each worker operates in its own global scope, thus does not share any resources with other threads. Communication, e.g., messaging or data transfer, between threads is done in a *Message Passing Interface* (MPI)[2] like manner and allows bidirectional messaging between parent and child threads. Transferring data can be done

[2] http://www.mcs.anl.gov/research/projects/mpi/

in in either two ways. The first way is structural cloning, where objects can be cloned and copied to other threads. The second way is transferable objects. Incorporating another more recent feature of modern web browsers in the form of typed arrays[3], this technique passes a reference to the buffer of an object to other threads. Generally workers can be separated into two different categories, depending on their way to communicate with other threads [Fla11].

Dedicated worker is the standard worker which is implemented by all modern browsers. The dedicated worker can send to and receive from one parent thread, but is not able to send messages sideways. Thus the dedicated worker can't interact with other worker threads.
Shared worker is an implementation which is not communicating via messages alone, but each worker also has specific *ports* it listens to. The shared worker can receive messages from more than one application and allows worker-threads to communicate with each other.

Thus, the actual parallelization scheme becomes similar to approaches using MPI. The idea for dedicated workers is to distribute data evenly between the worker-threads. Speedup is gained, since each thread has to do less calculations:

```
#Parent thread:
    Distribute data evenly between the child threads.
Repeat until convergence:
    #Child threads:
        Calculate the best matching unit.
        Return indices.
    #Parent thread:
        Calculate new prototypes.
```

3.2 Accelerated k-means through geometric reasoning

A focus of research concerning k-means is acceleration by using additional information available at runtime through geometric reasoning [PM99]. The central concept is to avoid unnecessary distance calculations by using the triangle inequality. For any three points x, y and z the following equation applies:

$$d(x, z) \leq d(x, y) + d(y, z) \tag{2}$$

Especially in later iterations Lloyd's algorithm tends to undergo unnecessary distance calculations when cluster centers are almost settled. This is particularly time consuming for higher dimensional datasets. Furthermore, it is not necessary to know the exact distance from a datapoint x_i to its corresponding center c_j, as long as the triangle inequality holds true.

[3] https://www.khronos.org/registry/typedarray/specs/latest/

Additional information is passed from one iteration to the next iteration in the form of *upper bounds* and *lower bounds*. The *upper bound* $u(x)$ denotes the distance of a datapoint x_i to its closest corresponding center c_j. The *lower bound* $l(x, c')$ denotes the distance of a datapoint x_i to each centroid c', where $c_j \neq c'$ [Elk03].
Obtaining useful *lower bounds* that help to skip unnecessary distance calculations will not be discussed here, detailed information can be found in C. Elkan's paper "Using the Triangle Inequality to Accelerate k-means" [Elk03]. Elkan proposes to utilize the triangle inequality by computing an *upper bound* $u(x)$ and keeps track of $k-1$ *lower bounds* $l(x, c')$ for each datapoint x_i. Although the algorithm achieves massive speedup compared to the standard k-means algorithm, it needs additional memory for *lower bounds*, which is used to keep track of the distances of a datapoint x_i to each centroid c' (where $c' \neq c$ and c denotes the closest center).
Variations of the accelerated k-means algorithm proposed by G. Hamerly and Drake & Hamerly's *adaptive k-means* algorithm show that it is possible to skip most of the distance computations by keeping track of fewer *lower bounds*. Hamerly's algorithm keeps track of only one *upper bound* and one *lower bound* [Ham10]. The adaptive k-means algorithm keeps track of a variable number b of useful *lower bounds*, which are considered for distance computations. Generally good runtime of the algorithm is achieved for an interval of $\frac{k}{8} \leq b \leq \frac{k}{4}$ [DH12]. Thus, both variations have a much lower memory profile but exceed Elkan's algorithm for lower (Hamerly's algorithm) to mid range (adaptive k-means) dimensional datasets [DH12].
Adaptive k-means is used here as an algorithmic approach to faster execution and was implemented in JavaScript. Additionally, a threaded version using web workers was developed.

4 Experimental Results

This section presents the observations made during the experiments with the different k-means implementations. All results were produced with an Intel Core i7-3632 QM @ 2.20 $\times$4 CPU and 8GB Memory. As web browsers Mozilla Firefox version 27 and Google Chrome version 32 were used. The experimental set up primarily considers multithreading approaches with dedicated workers and algorithmic enhancements using adaptive k-means.

Table 1: Different versions of accelerated k-means clustering applied in our study.

Algorithm	Description
standard-kmeans	Standard k-means implementation
worker-kmeans	*standard-kmeans* executed from a web worker script
threaded-kmeans	Threaded k-means implementation employing web workers
trans-thread-kmeans	*threaded-kmeans modified to use transferable objects*
adapt-kmeans	k-means implementation based on the adaptive k-means algorithm
thread-adapt-kmeans	Threaded version of *adapt-kmeans*

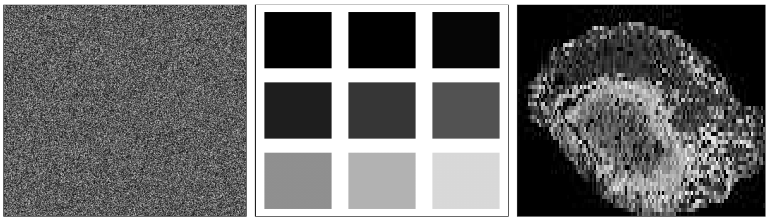
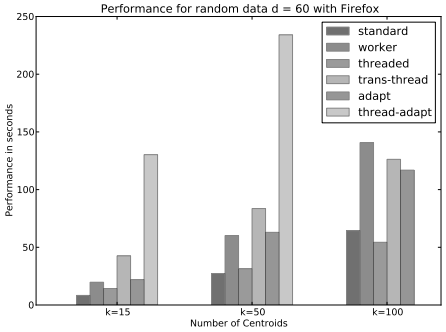
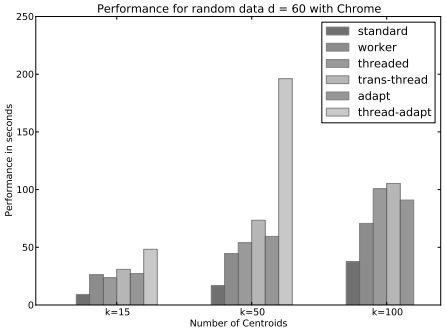


Figure 2: Sample images for three different datasets: 1. A 500×500 multivariate image of random intensity images ranging from 0 to 255 and dimensions of $d = 30$ and $d = 60$. 2. A 300×300 tiled constructed multivariate image containing 9 clusters plus background and dimensions of $d = 30$ and $d = 60$. The tiles vary their intensities between -5 and $+5$ of their average. 3. A 120×50 MALDI-MSI dataset from a study on barley seed germination [Gor13] with dimensions of $d = 54$ and $d = 94$.

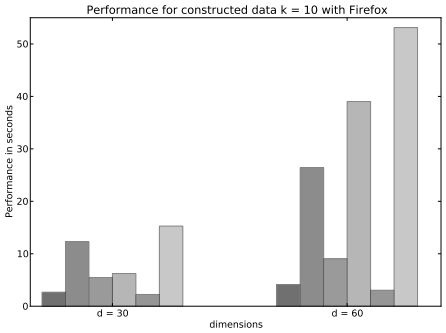
In the following, the different versions will be referenced to as shown in Table 1. The algorithms are considered to reach convergence if 90% of the centroids do not differ in the distance of an ϵ - environment of $1 \cdot 10^{-7}$ to the centroids of the previous iteration. If no convergence occurs the process is terminated after 1000 iterations. For the experiments three different datasets were employed to verify performance in terms of runtime (Figure 2.1), accuracy (Figure 2.2) and behavior on real MALDI-MSI data (Figure 2.3).
The runtime of the different implementations listed in Table 1 depends on the web browser used. Overall Chrome achieved better execution times than Firefox for the single threaded versions, i.e., standard-kmeans and adapt-kmeans. In particular the standard-kmeans method showed the best results for runtime on the datasets using Chrome, but good runtime results can also be observed using Firefox (see Figure 3). The second single threaded k-means version, i.e., adapt-kmeans, showed high variance in its runtime performance. The obtained runtime results for the Firefox browser show that the algorithmic enhancement achieved slightly better performance for the constructed and the MALDI-MSI dataset (see Figure 3 (c) and (e)), however, particularly for the Chrome browser the adapt-kmeans algorithm performs worse than standard-kmeans (see right column of Figure 3).
Firefox performs consistently better for threaded k-means versions such as threaded-kmeans, trans-thread-kmeans and adapt-thread-kmeans (see Figure 3). In the case of the threaded version of the adaptive k-means the runtime experiments had to be terminated. Due to extensive memory usage the test runs could not be finished (see Figure 3 (a) and (b)). Also the threaded k-means method using transferable objects, i.e., trans-thread-kmeans, was outperformed by the version using structural cloning, i.e., threaded-kmeans, in most cases (see Figure 3). The threaded-kmeans method achieves runtime performance comparable to standard-kmeans (see Figure 3 (c) and (e)) and also exceeds the runtime performance of the single threaded standard-kmeans version (see Figure 3 (a)). In addition, it could be observed that the single threaded k-means version pushed into a background thread, i.e., worker-kmeans, overall performed worse than the standard-kmeans version (see Figure 3).



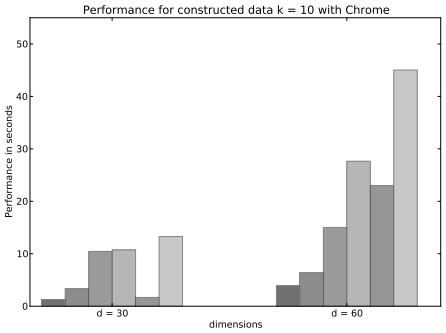
(a) Random image dataset with Firefox



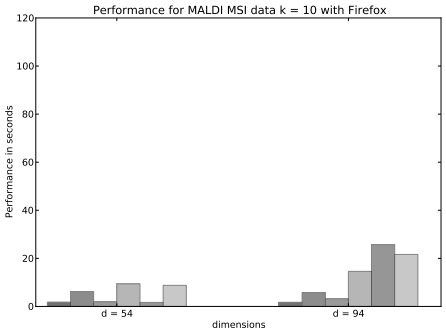
(b) Random image dataset with Chrome



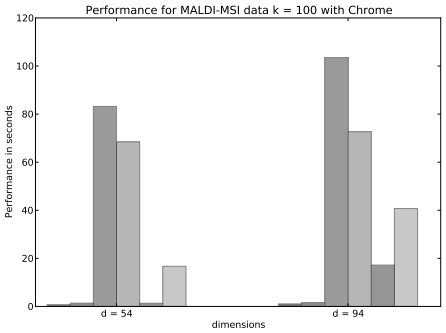
(c) Constructed image dataset with Firefox



(d) Constructed image dataset with Chrome



(e) MALDI-MSI dataset with Firefox



(f) MALDI-MSI dataset with Chrome

Figure 3: Runtime results of the employed datasets for the Firefox browser (left column) and the Chrome browser (right column). Figures (a) and (b) Runtime results of the employed k-means versions for $k = 15$, 50 and 100 on the random image dataset. Figures (c) and (d) Runtime results of the employed k-means versions for $d = 30$ and 60 on the constructed image dataset. Figures (e) and (f) Runtime results of the employed k-means versions for $d = 54$ and 94 on the MALDI-MSI image dataset.

5 Discussion

On the one hand, the results show that both Firefox and Chrome are able to perform complex tasks such as clustering multi-spectral image data in a client-side environment. The standard single threaded k-means version overall shows good performance on any of the datasets and delivers reasonable fast results even for huge datasets (see Figure 1.1 and Figure 3). On the other hand, the runtime experiments show unexpected variability in the performance of the algorithmic enhancement approaches and the threaded versions of the k-means algorithm. One possible reason for the adaptive k-means methods to not deliver the expected performance boost might be that the termination criterion does not fit and other criteria might show better results. The high memory usage observed for the threaded version of the adaptive k-means can partly be explained by the characteristics of web workers, since a web worker has high memory usage.
For the threaded approaches in general, the overhead for worker start-up and communication between the threads likely predominates the effects from parallel execution and no speedup can be measured. Another unexpected observation was made regarding the use of transferable objects and structural cloning. Transferable objects are a reference that is passed to a worker-thread. As a consequence the communication delay becomes smaller and a one to two orders of magnitude faster data transfer rate, than with structural cloning, can be achieved. Despite faster transfer rates the trans-thread-kmeans method overall took longer to finish, which is likely caused by the usage of native arrays and thus achieving higher precision in the distance computation and recalculation of the k-means prototypes.

6 Conclusion and Outlook

With respect to the initially formulated question, if it is possible to achieve online real time clustering of high dimensional datasets within a client-side web browser environment, the experimental results show that this goal could be successfully achieved for multi-spectral image datasets. The second question, whether or not it is feasible to accelerate heavy computational tasks, like clustering, with either computational approaches or algorithmic enhancements needs to be investigated further. At least for the current browser versions, trying to accelerate k-means with mutithreading or by using the adaptive version actually performs slower than the standard implementation.
Regardless of problems concerning the multithreaded versions of the k-means algorithm, the runtime experiment results show, that online real time clustering of multi-spectral datasets is possible. This enables new visualizations, e.g., a user can cluster a hyperspectral image, de-select single channels of it and get a new clustering result directly. Thereby, interactive exploration of datasets with the help of cluster analysis becomes possible.
Both web browsers achieve good performance and respond with a new clustering result in less than three seconds for either of the datasets and with all channels selected.
It remains to be seen if the threading approaches get more potential with future features for web browsers. With broader support across all browsers the shared worker API might be a

possible candidate for concurrency computing models in a web browser environment. Up until now, single threaded approaches seem to be superior to concurrency models. With later iterations of the Web Worker interface and optimized approaches for concurrency computing in a web browser environment, client-side web browser solutions can become a viable addition to current approaches for the analysis of high dimensional image datasets.

Acknowledgments:
This work was supported by the DFG Großgeräteinitiative "Bildgebende Massenspektrometrie" and the CLIB Graduate Cluster Industrial Biotechnology.

References

[DH12] J. Drake and G. Hamerly. Accelerated k-means with adaptive distance bounds. In *5th NIPS Workshop on Optimization for Machine learning*, 2012.

[Elk03] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, pages 147–153, 2003.

[F+13] J. Fonville et al. Hyperspectral Visualization of Mass Spectrometry Imaging Data. *Chemical reviews*, 85(3):1415–1423, 2013.

[Fla11] D. Flanagan. *JavaScript: The Definitive Guide*. O'Reilly Media, 2011.

[Gor13] K. Gorzolka. *Spatio-temporal investigation of the barley malting process by proteome, metabolome, and MALDI MS imaging analysis*. PhD thesis, Bielefeld University, May 2013.

[Ham10] G. Hamerly. Making k-means Even Faster. In *SDM*, pages 130–140, 2010.

[HLN11] J. Herold, C. Loyek, and TW Nattkemper. Multivariate image mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):2–13, 2011.

[Jai10] A. K Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[K+12] J. Kölling et al. WHIDE - a web tool for visual data mining colocation patterns in multivariate bioimages. *Bioinformatics*, 28(8):1143–1150, 2012.

[L+11] C. Loyek et al. BioIMAX: A Web 2.0 approach for easy exploratory and collaborative access to multivariate bioimage data. *BMC bioinformatics*, 12(1):297, 2011.

[Llo82] S. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[NC13] J. Norris and R. Caprioli. Analysis of tissue specimens by matrix-assisted laser desorption/ionization imaging mass spectrometry in biological and clinical research. *Chemical reviews*, 113(4):2309–2342, 2013.

[PM99] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 277–281. ACM, 1999.

[W+08] X. Wu et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.