CONTENT-ADRESSABLE MULTIMEDIA DATABASE
SERVER FOR MEDICINE

The development of “PANORAMIX”, an Electronic Health Record

Pedro Miguel Nunes Santos
Instituto Superior Técnico
INESC-ID (DMIR Group)
pedro.santos@tagus.ist.utl.pt

ABSTRACT
The main objective of this work was to develop an electronic health record (called "PANORAMIX") featuring a content-addressable database through CBIR search methods (like the Hierarchical Linear Subspace). We aim to demonstrate the benefits that this intelligent type of information retrieval can bring to the actual real practice of medicine, when integrated to an everyday use application such as an EHR.

The system is a client-server (thin client) web application, based on WebObjects (Java). It was built accordingly to official standard patient record forms (in paper), currently being used in the Portuguese Public Primary Health Care Centers. The system is characterized by being modular, portable, interoperable, easy to scale (up and out) and ubiquitously accessible.

Keywords: Patient clinical information, EHR/EMR (Electronic Health/Medical Records), CBIR (Content Based Image Retrieval), Hierarchical Linear Subspace search method, GEMINI (Generic Multimedia Indexing), WebObjects.

1. INTRODUCTION
The clinical record is a key element for planning the delivery of health care to a patient. It is a dynamic informational entity that continuously monitors/reflects the evolution of the health status of a patient, containing all the clinical information relating to him, from the clinical history and treatments to the considerations and findings resulting from the medical examinations.

One of the areas where the information technologies can be more usefully applied in medicine is the management and preservation of clinical data from patients through the so-called electronic medical records – Electronic Health / Medical Records (EHR / EMR). [1]

One of the major advantages of EHR over their paper-based counterparts (besides the more commonly known provided by the electronic management of data) is that all clinical/diagnostic tests (especially medical imagery) become attached to the patients profile (and available on electronic format). [1] Most of those tests consist of unstructured media formats (like the images). This fact complicates the search and retrieval of information since its semantic structure is not directly interpretable. So this type of data can not be searched by “conventional” queries of a relational database (as any normal text field), unless their files are annotated with external descriptors containing metadata describing and qualifying their content (such as MPEG7 files for the videos) [2; 3; 4; 5; 6].

The problem is the construction of those annotations (specifically the extraction of image features) and their semantic value. For descriptors with a high semantic value, which are directly usable in the medical context (e.g. search for specific tumour images) have to be built usually by a manual tagging process. This is a very expensive process, especially considering the daily volume of generated imagery1. The resulting descriptors usually have a very specific and limited use, as well as their classification is subjective to human interpretation. [3; 7; 8; 4; 5; 6] For instance the diagnosis using radiological imagery may not be uniform, due to doctors’ different interpretations.

Some of the existing systems are already able to perform automatic identification and annotation of objects, through statistical models and machine learning techniques, based on sets of categorized (clustered) training images that represent a certain concept. But these procedures are computationally very heavy, and operate only in very specific areas (and the classification of diseases may have a very wide degree of variability). [8]

An alternative to high-level descriptors is the search through the low-level structural features of the image (like texture, colour, shape), through a process called CBIR (Content Based Image Retrieval). These characteristics are easily retrievable, and if properly combined in sets (through standardized patterns), they can translate into a high semantic value and therefore become directly usable on a given context. [2; 9; 5]

An example of CBIR search is the method of “querying-by-example”, which consists in defining the search parameters through an image example (query-image), similar to what we want to find. So basically we create some kind of an “abstraction layer” by setting a "temporary pattern" of low-level features that represent what we are looking for. [7; 9]

1 For instance, the Radiology Department of the University Hospital of Geneva alone produced more than 12 000 images a day (in 2002). [7]
The combination of visual features (which translate the otherwise uninterpretable image content) with query-by-example techniques can create a search engine truly content addressable, that facilitates the search and management of unstructured information (such as the clinical imagery).

The main objective of this work was to introduce and demonstrate the potential that this type of intelligent access to information (CBIR) can bring to medical applications used directly in the delivery of health care to patients. This was accomplished by building an electronic health record, which includes a multimedia database, in order to “attach” the medical lab/imagery tests of each patient, to their own clinical profile. It was also implemented some CBIR algorithms, that improves the searching for those test’s images.

The application of the CBIR concept in medicine is relatively new. There are indeed some academic studies and projects including developed applications (e.g. medGIFT) that implement and studies the CBIR in medicine. However there still isn’t any integrated application fully functional in the real clinical practice. [7]

2. ARCHITECTURE AND TECHNOLOGY

2.1. Functional Architecture

We choose to implement some of the most typical functions of this kind of systems (as well as the new CBIR search methods). For instance: searching, browsing and management of patient data sheets; registration of consultations, new patients and doctors; upload of medical tests and other documents; definition of personal system settings; among others. Nevertheless the system is modular, so that new features can be added as well as the existing ones gradually refined in future work.

For the CBIR search, we choose to implement three algorithms: the histograms method; the basic list matching method (here called “Brute Force”); and a Hierarchical Linear Subspace method [8].

2.2. Information Architecture

There were a couple of restrictive factors to consider when deciding which patient information should be stored in the system. On one hand the actual data contained in medical record is heavily dependent on the medicine practice at hand, and the information that each doctor and/or the medical institution considers relevant. On the other hand it does not exist yet a standard electronic patient record data model, defined by the Portuguese Health Authorities ². However there are official standard patient record forms (in paper), being used in the Public Primary Health Care Centers.

The general practice form model is composed by several semantic subgroups:

- Personal Clinical Sheet – Contains the patient’s clinical information (e.g. blood group, vaccination data, personal habits, prolonged treatment, personal history, among others);
- Social and Family background Sheet – Contains the patient’s socio-family background information (e.g. household composition, living conditions among others). It may be shared by the other family members;
- Consultation Record Sheet – Contains the records of all the medical appointments, organized according to the SOAP structure (Subjective, Objective, Assessment and Plan);
- Medical Tests summary Sheet – Contains relevant clinical test results (only the more relevant values) as well as the date when they were performed.

The system’s patient data model was based on those official forms (for General Practice). This not only ensures that the stored data is in accordance with the definitions accepted by any Portuguese Public Health Institution, as well as the system forms are familiar to the largest possible number of doctors.

We choose the most important fields to represent the various semantic groups, according to a professional medical opinion. The Figure 1 shows part of the patient data stored in the system.

2.3. Application Architecture

The system is a web based client-server (thin client) application composed by three layers (presentation, logic and data). This type of architecture has a growing popularity among applications of this kind, especially for smaller or independent medical practices (like small clinics). This is because the cost of implementation is much smaller, and the system is easier to build and less demanding in terms of hardware needed (e.g. on the client side the interface is made through web browser). Furthermore these systems are much more ubiquitous in terms of access. They can be accessed by any device connected to the internet.

One of the main requirements for the system is modularity, so that future functionality can be added or improved gradually in different versions, without affecting those already been built. In order to accomplish this requirement a loosely coupled components separation strategy amongst functionalities was devised. By ensuring object oriented practices such as class encapsulation, amongst others, we were able to achieve those objectives.

Security is another important feature required for systems that handle sensitive data such as the patient clinical information. In this case it is being addressed by another parallel Master Thesis (see the ongoing work [10]). It is basically composed by login via username and password pair, the use of communication security protocols (like SSL), the encryption of data and the separation of databases (for sensitive information).

Interoperability is also an unavoidable subject when considering medical applications, since a substantial part of their functionalities rely on their capacity to communicate with each other (and being most of them heterogeneous systems). The “Panoramix” besides being based on Java technology (which is widely used, facilitating the integration of other software) it is built in conformity with typical industry communication standards such as HL7 (also being addressed by the ongoing Master Thesis [10]) and in the future the DICOM (for images).

---

² It is a very current issue that has being addressed by the ACSS (Administração Central do Sistema de Saúde).
2.4. Technology Architecture

The system consists on a central server where the application is deployed, as well as the database with the patient information (we use MySql 5) and the file systems where the stored medical tests are kept. The clients’ machines access it remotely over the Internet through a web browser.

The physical storage of the images was decided to be through file system instead of a database (DB) (although some of their metadata and smaller feature vectors are kept in the DB). Despite the performance increase gained when the images and their metadata were stored in the database, this method was unmanageable for large volumes of information. Because each time that information is extracted from the DB, the corresponding entity was encapsulated in a Java object (by an ORM tool used\(^3\) – Object Relational Mapping). This consumed too much RAM, causing the Java Heap to overfill very easily. Besides that, the system is supposed to operate on a network (or at least it’s ready to), interacting with other disparate applications, and therefore the images should be stored on a file system, to facilitate the access (as in the PACS – Picture Archiving and Communication System).

\(^3\) Enterprise Objects (as part of WebObjects).

Therefore the images, as well as some of their feature vectors (the bigger ones) had to be stored outside the database.

The system is based on WebObjects, which is the web applications development platform from Apple. This technology helps giving some beneficial properties to the system\([11; 12]\):

- Portability and interoperability – The application can be deployed over different platforms (e.g. Linux, Windows, Mac) since the basic technology is Java (both the application main programming language and the frameworks). This also allows the integration of many existing libraries and other Java applications;
- Modularity – The functionalities and the user interface were built from reusable and loosely coupled (relatively independent of each other) web components. Which is the basic construction element of the of a WebObjects applications structure;
- Ease of access and ubiquity – The system is a web application accessible by web browser (web portal), from any device with Internet connection;
- Scalability – The application can have multiple instances running in one or more servers, almost seamlessly and
transparency to the programmer. He only has to choose one of many available load balancing algorithms (or implement his own).

3. CBIR SEARCH METHODS

All CBIR search algorithms (using query-by-example approach) are based on the same general idea: The comparison of an image-example (the query-image) representation (the feature vector) with all the equivalent representations of the images in the search space. This comparison is made in an abstract space (the feature space) where the images are represented through points, and the distance between them gives the level of similarity between their corresponding images. Than the images, whose distance is below a certain limit (threshold), are returned (sorted by similarity with the query-image).

The greatest difference between the various algorithms is the image features (low-level characteristics) used to build their feature vectors (and sometimes also the distance formula used). The feature vectors are an abstract representation of an image, through a numeric n-dimensional vector (within the feature space), which characterizes it according to a given parameter (or several combined). We can think of them as a "signature" of an image.

The dataset of images used was composed of 12 000 images in shades of grey in the Portable Network Graphics format (.PNG), resulting from several X-rays randomly collected. The images came from 116 different categories (different views of radiographed body parts), belonging to persons of both sexes, various ages and different diseases. So the quality and size of the images varies widely, and all of them were rescaled to fit in a 512x512 Bounding Box (but they do not necessarily have this dimension) keeping the same aspect ratio. [13]

We can see one example of the CBIR search in Figure 2, with the results to a given query. Note that the hits are organized according to decreasing similarity, and in this case the first hit corresponds to the query image (thus the distance is zero).

3.1. Histograms

This method consists in the comparison of the colour histograms of the images. This histogram gives the pixels count with the same colour in each image. It is computed and stored when an image is added to the system (through the JAI library – Java Advanced Imaging).

The number of comparisons is always the same for every query (for a certain number of images stored in the system). We use histograms with 256 bins, hence capable of counting 256 different shades of grey (because of the colour depth of the images being used). This means we have 256 comparisons per image stored. Since we are using 12 000 images (for testing purposes), the number of total comparisons per query is 3 072 000.

As we will see in the results section, this method is extremely efficient in processing queries, but is not very effective in finding similar images. This can be explained because the histograms only take into consideration the absolute pixel count with a particular colour, ignoring their spatial distribution. For that reason it was decided to implement the algorithm shown below.

3.2. “Brute Force” (List Matching)

This method is a common list matching. This is basically a straight comparison of all the pixels of the images. The features vectors used here are the pixels colour vectors of the images in their original size (sometimes scaled to 256x256).

The algorithm consists of calculating the Euclidian distance between the pixels vectors of the query-image and each image in the system. All images whose distance is less than the established threshold are returned in ascending order.

Since the original pictures have different sizes, which don’t allow the direct comparison between them, first we have to resize the images to a fixed square resolution (256x256).

The number of comparisons is also fixed in this method, since each rescaled image (256x256 pixels) is compared to the query-image (65 536 comparisons per image). Since currently there are 12 000 images in the system, the total number of comparisons per query is 786 432 000.

As it will be shown later in the results section, this algorithm is much more effective than the histograms. Because the pixels compared between images are equivalent (in terms of their spatial position in the image), thus the distance value reflects much more accurately the degree of similarity between the images.

But in terms of efficiency, the algorithm is extremely heavy, since the number of comparisons per query increases very much compared to the histograms. For that reason it was decided to implement the following algorithm, which is based in the same principle as this one, but has a mechanism to reduce the number of comparisons.

3.3. Hierarchical linear Subspace

This method is also based in the comparison of pixels values, but through hierarchical search in smaller vectorial subspaces of the images. This particularity combined with a limitation through threshold, reduces the search space by eliminating different images prior to their comparison in the original dimension (thus reducing the total number of comparisons).

This approach is based on an indexing system called Generic Multi Media Indexing (GEMINI). The main idea is to find a feature extraction function that maps the object (images) from its original dimensional space into a smaller feature space. At that size is faster to apply a "quick-and-dirty" test in order to discard items not qualified for the query (whose distance is greater than the threshold) – pruning process. To accomplish this, the mapping has to be according with the so-called lower bounding lemma, which ensures that the distance in the feature space is always inferior or equal than the original, and therefore ensuring that any qualified object will not be lost (false dismissals). [8; 14]

4 The authors of this dataset (V1.0, 2007) are Lehmann T.M., Schubert H., Ott B. e Leisten M. The dataset was built for IRMA Project and for the ImageCLEFmed 2007 competition [13] (courtesy of TM Deserno, Dept. of Medical Informatics, RWTH Aachen, Germany).

5 This algorithm was created by Prof. Andreas Wichert, from INESC DMIR group (IST) [8].
One proved consequence (or extension) of this lemma states that any distance in a subspace of the feature space will always be less inferior or equal to the distance in that feature space, which it self is also inferior or equal than the original. (see Lemma 1) [8]

**Lemma 1** – Let $O_1$ e $O_2$ be two objects; $F(O_2)$, the mapping of objects into $f$ dimensional subspace $U$ should satisfy the following formula for all objects, where $d$ is a distance function in the space $V$ and $d_U$ in the space $U$.

$$d_U(F(O_1), F(O_2)) \leq d(F(O_1), F(O_2)) \leq d(F(O_1), F(O_2))$$

Therefore we can define a sequence of hierarchical subspaces $U$ (where $U_0 \supset U_1 \supset \cdots \supset U_n$), whose dimension is also lower $(\dim(U_0) > \dim(U_1) > \cdots > \dim(U_n))$, and consequently use them to successively refine the search space (filter dismissals). In the end we get a result set without any false dismissals, but it might have some unqualified objects (false hits). To filter these false hits, it is necessary to compare the objects in their original space. [8; 14]
This implementation begins to compare the image in lower subspaces (which correspond to images "reduced" from the original dimension) and verify if the distance exceeds the threshold (duly multiplied by the adequate normalization constant for that dimension). If the threshold is exceeded, it means the image has not the degree of similarity required and that therefore there is no need to compare it in higher subspaces. This way we eliminate the set of images that are not hits, saving up unnecessary comparisons in larger subspaces. In the end we proceed to the comparison of image vectors of 256x256, corresponding to the last filtering in the original space.

The comparison of the Euclidean distances in the different subspaces uses a threshold value from the original image space (256x256). We project the distances from the lower dimensional spaces into the original image space. The projection corresponds to the multiplication of the distance in a given subspace by a constant. That constant depends of the subspace dimension, and it is calculated through (1).

\[ C_{\text{subspace}} = \frac{\text{Original dimension}}{\sqrt{\text{Subspace dimension}}} \]  

(1)

For the mathematical background and further explanation of the theory behind this algorithm, please consult [8].

This time the feature vector is an “image pyramid” whose levels (composing vectors) correspond to the original image resized (image subsamples). Each image pyramid is formed by four vectors: 256x256, 16x16, 64x64, 8x8 (4x4 was initially used but now deprecated, we will see why in the next section). These dimensions were chosen because the closest square resolution to the average original sizes of the dataset is of 256x256 (they had to be standardized to allow comparison with each others). We take the highest dimension level (256x256) and than apply \( \frac{1}{4} \) reduction factor, to build the lower levels of the hierarchy. The comparison is made top-down from the pyramid (from the lower 8x8 to the highest 256x256). (See Figure 3)

The image down-sampling is made through the arithmetic mean (or average) of the pixels surrounding in the precedent higher space. This because the average guarantees that each "sub-image" in the pyramid is an orthogonal projection of the precedent higher dimension, and therefore corresponds to a linear reduction from the original image. This means that the distances are preserved and will be kept equivalent in all dimensions (respecting the extension to the Lower Bounding Lemma). [8] The rescaling methods of Java image utilities and native image classes, uses interpolation formulas (e.g. Nearest-neighbour, Bilinear, Bicubic methods), for which it would be necessary to prove that they respect the lower bounding lemma. Thus, for instance to compute the 64x64 subspace, we divide the precedent subspace (256x256) in square windows with four pixels wide (16 pixels), resulting in 4 096 windows (256x245 pixels / 16). In each window we calculate the average of its pixels, which will be the value of the corresponding pixel in 64x64 subspace.

The total comparisons number per query depends on the threshold and the query-image itself. Depending whether is a higher or lower threshold the number of qualified images for a query (the hits) varies. When an image has a lower similarity value than the threshold, in a query, thus not being qualified as a hit (dismissal), the system continue immediately to next image, saving the comparisons of the higher dimensions. Of course this depends from image to image (even if the threshold is the same) and the corresponding similarity level (with the query-image), therefore is not possible to estimate a fixed number of comparisons per threshold.

![Figure 3 – Image Pyramid and the maximum number of comparisons per level (for 12 000 images dataset).](image-url)
4. EXPERIMENTAL RESULTS

The tests were run locally on two Apple iMacs with a 2.0 GHz (and 1.8 GHz) processor Intel Core 2 Duo, 1 GB of RAM (2 DIMM DDR2 SDRAM, 512 MB of 667MHz). In terms of software, both machines were the same: operating system Mac OS X 10.4.11; WebObjects (5.3.3); Java Platform Standard Edition (J2SE) 1.5.0_13. The Java Heap was configured with 32 MB of initial space, extensible to 512MB. The thread priority was set to “MAX_PRIORITY”.

For testing purposes, a sample of 600 images was randomly selected from the original dataset. With this sample, initially we estimated the proper thresholds for this dataset. First, we calculated the Euclidean Distance between each picture and those stored in the dataset (in all sub-spaces – 4x4, 8x8, 16x16, 64x64 and 256x256). Afterwards the mean value between the distances for each sub-dimension was calculated. In Figure 4 we can see the chart that maps the average Euclidean distance per a certain number of images.

As we can see there is a considerable distance reduction in the 4x4 subspace, considering the other dimensions. This means that for any threshold value chosen, there would not be enough dismissals elimination (in the 4x4 subspace), so that the overall performance gain would be significant. Therefore we decided to deprecate this size and use bigger one instead (8x8).

The thresholds for this dataset were chosen based on the average number of expected hits for a query. The estimate can be seen in the magnified threshold chart (see Figure 5). For each number of images “X” the average Euclidean distance is “Y” units. Therefore one query with a certain threshold “Y” is expected to return “X” results.

For evaluating the systems efficiency performance, in the following table (see Table 1) we can see the averages of values recorded (as well in the charts of the 50 first queries of the 600 performed – Figures 6, 7 and 8):
Table 1 – CBIR methods comparison. The values marked (*) are always the same.

<table>
<thead>
<tr>
<th>Search Method</th>
<th>Average Num. Hits</th>
<th>Average Num. Comparisons</th>
<th>Time (ms)</th>
<th>Standard Deviation of Time (ms:ms)</th>
<th>Time (ms:ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute Force (threshold 0)</td>
<td>1(*)</td>
<td>786 432 000(*)</td>
<td>130 048.7717</td>
<td>00:11:138</td>
<td>02:10:049</td>
</tr>
<tr>
<td>Histograms (BD)</td>
<td>12 000(*)</td>
<td>3 072 000(*)</td>
<td>1 084.878333</td>
<td>00:00:255</td>
<td>00:01:085</td>
</tr>
<tr>
<td>H. Subspace (threshold 6x10^8)</td>
<td>26</td>
<td>3 312 536</td>
<td>1 191.726667</td>
<td>00:01:684</td>
<td>00:01:192</td>
</tr>
<tr>
<td>H. Subspace (threshold 8,516x10^8)</td>
<td>348</td>
<td>29 551 591</td>
<td>9 494.616667</td>
<td>00:11:324</td>
<td>00:09:495</td>
</tr>
<tr>
<td>H. Subspace (threshold 9,487x10^8)</td>
<td>668</td>
<td>54 605 747</td>
<td>18 035.87667</td>
<td>00:18:407</td>
<td>00:18:036</td>
</tr>
<tr>
<td>H. Subspace (threshold 1,012x10^9)</td>
<td>962</td>
<td>76 874 233</td>
<td>26 502.27667</td>
<td>00:24:222</td>
<td>00:26:502</td>
</tr>
<tr>
<td>H. Subspace (threshold 1,115x10^9)</td>
<td>1 587</td>
<td>123 511 252</td>
<td>41 102.36833</td>
<td>00:33:737</td>
<td>00:41:102</td>
</tr>
<tr>
<td>H. Subspace (threshold 1,225x10^9)</td>
<td>2 471</td>
<td>187 914 922</td>
<td>61 354.11167</td>
<td>00:41:132</td>
<td>01:01:354</td>
</tr>
</tbody>
</table>

Figure 6 – Average values charts for the Hierarchical Subspace p/threshold: (1) average number of comparisons, (2) average time, (3) average number of Hits.
As we can see the Hierarchical Subspace method can be far more efficient, depending on the threshold used. And if we consider that in normal operative conditions, a doctor wouldn’t want a large number of hits (only the images more similar to the query-image interest the user) we can use lower thresholds to limit the results, which will improve even more the performance. This happens because when the threshold is lowered, more dismissals will be filtered in the lower subspaces, saving the comparisons in the higher dimensions. However we must be careful with the threshold used, because that value will influence the number of hits returned, and it might make the method more inefficient than the very “Brute Force”.

The threshold values should be reviewed periodically, according to the images dataset present in the system (as the set of images change, the average distance values amongst them change as well, and consequently the proper thresholds also change). Note that those values are only approximate estimations, since the proper threshold for a given query would depend most on the query-image itself.

As for the histograms, as we have already seen, its high number of comparisons is due to the fact that in all queries is only necessary to compare the 256 bins of the histograms of all images in the system (from a given doctor). That is why the number of comparisons is always equal, as well as the number of results (this may be filtered at the end of the query, to facilitate results analysis).

The discrepancy in the time registered between the methods, is due to the type of storage of the feature vectors. E.g. the histograms can be saved in the database, while the pyramid of images can not be fully stored there, because it takes up too much space when extracting the vectors and converting them into Java objects (only the smaller, lower subspace vector is storable).

As for effectiveness, since we didn’t have the similarity absolute values, between the images, we had to base this evaluation upon the human observation of the query results. So we chose among some similar image groups (e.g. hand, lungs x-rays). Than we perform queries, with query-images chosen from those groups, and count the number of images returned in the first five hits, belonging to those groups (images that are visually identical to the query image).

It was concluded that the histograms method, despite extremely efficient, is not very effective in finding similar images. This happens because it only takes into account the number of similar pixels, disregarding their spatial distribution. In the Hierarchical Linear Subspace and the “Brute Force” methods, this doesn’t happen (because the pixels compared have equivalent spatial positions).
5. CONCLUSION

We built an EHR system, incorporating a "Content-addressable" multimedia database, through CBIR search methods. The system consists of a web application, accessible from any device with internet connection and a Web browser.

The CBIR methods (using query-by-example) allow the search for imagery tests to be content-oriented instead of using external metadata descriptor files, avoiding the problems of tagging. And as demonstrated, it may benefit the EHR systems, in terms of clinical decision support. For instance to better treat a disease or "predict" the development of a patient health status, based upon similar medical histories of other patients, found using the comparison of similar examinations.

There are CBIR methods more suitable for the medicine context than others, especially because of the features they use. Most of the medical images use monochromatic levels of grey, not colour, and have low contrast. So the textures and the identification of contours may have better results in medical CBIR search than the use of colours. [3; 7]

We obtained good results (both from effectiveness and efficiency performances points of view) using colours as the image feature, and the hierarchical linear subspace method (which takes into account the spatial distribution of pixels). This happens by choosing an appropriate threshold for the set of images in the system, so that limits the number of returned hits (which is ideal for a normal clinical use of the system, since the doctors are only interested in the first few most similar hits).

6. ACKNOWLEDGEMENTS

The author would like to thank Dr. TM Deserno (and TM Lehmann, H. Schubert, Ott B. and Leisten M.) for the permission to use the dataset for experimental tests purposes [13].

He would also like to thank Prof. Andreas Wichert for all his support during the development of this Master Thesis.

7. BIBLIOGRAPHY


