Grant Agreement Number: 257528

KHRESMOI

www.khresmoi.eu

Report on the anatomical structure identification and localization

Deliverable number	*D2.5*
Dissemination level	*Public*
Delivery data	*due 31.8.2013*
Status	*Final*
Authors	*René Donner, Georg Langs, Dimitrios Markonis, Matthias Dorfer, Henning Müller*

This project is supported by the European Commission under the Information and Communication Technologies (ICT) Theme of the 7th Framework Programme for Research and Technological Development.

Executive Summary

This deliverable describes the computational framework for the identification and localization of anatomical structures in medical imaging data. It is a core process in the KHRESMOI indexing framework of radiological imaging data in the clinical context. The focus of the deliverable is the framework that integrates several algorithmic approaches into a unified scalable indexing system. It relies on methodology developed in KHRESMOI. The system estimates coarse position of imaging data in relation to human anatomy. It estimates a mapping between a whole body reference space and each individual volume. This mapping is then used to establish location correspondence across cases, and to propagate anatomical structure labels to individual imaging data. The deliverable concludes with a discussion of the current status, and its limitations and suggests the direction of research for the remaining project period.

Table of Contents

List of Figures

Abbreviations

API	Application Programming Interface
CPU	Central Processing Unit
CT	Computed Tomography
DICOM	Digital Imaging and Communications in Medicine
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
MR(I)	Magnetic Resonance (Imaging)
MRF	Markov Random Field
MUW	Medical University Vienna
PACS	Picture archiving and communication system
RadLex	unified language / ontology of radiology terms
ROI	region of interest
SLIC	Simple Linear Iterative Clustering

1 Introduction

This deliverable describes the system for computational structure localization and identification during indexing of large scale medical imaging data. It summarizes the results of a number of lines of work performed in the course of the KHRESMOI project ranging from feature extraction, to model learning, group-wise registration, and landmark localization. The premise of the KHRESMOI indexing system is that data can be indexed without manual user interaction. Therefore a computational localization and identification framework parses the data during indexing, to map anatomical labels to the imaging data.

Localization and structure identification is a core process of large scale biomedical image mining and search. In the context of clinical radiology it serves as a means to constrain search spaces (e.g., a query consisting of a lung pathology should typically result in search results of lung pathologies), and to define domains for feature extraction and learning (e.g., features that are helpful for lung pathology matching, are not necessarily suitable for abdominal pathologies). In the context of literature search it is an implicit process of matching structures across images.

Previous deliverables hold in-depth information regarding methodology for feature extraction from imaging data (Deliverable D2.2), the evaluation of all individual components (Deliverable D2.3), and the underlying learning frameworks that form the core for localization and identification (Deliverable D2.4). In this deliverable we describe how these components are integrated to obtain a scalable framework that can process medical imaging data to identify the anatomical structures present.

The KHRESMOI localization and identification framework comprises general methodology instead of a collection of specialized approaches. This is motivated by the aim to explore scalable approaches that aim at capturing the entirety of the imaged observations, instead of focusing on a growing set of well modeled individual anatomical structures. The KHRESMOI indexing and retrieval prototype is a modular system. This deliverable will only briefly outline the localization and structure identification approaches that have been developed in the course of KHRESMOI. It will focus on the description of the computational framework that integrates these approaches in a general indexing and retrieval framework. The deliverable is divided into two main parts:

1. **Scalable computing framework**: We describe the computing framework, that allows for a scalable, parallelized computation of the localization during indexing of large amounts of medical imaging data. Given a set of volume data such as MRI, or CT volumes, the system assigns each voxel in each volume a position in a reference space, that establishes correspondence across volumes. It labels those voxels that are part of indexed organs.

2. **Localization and structure identification algorithms**: We outline the components that are part of this framework. There are two directions of algorithms relevant for localization, both of which rely on a learning- and a mapping phase:

   (a) atlas mapping of entire medical imaging volumes;

   (b) identification of individual landmarks in the volumes.

In addition to describing the current status of the system, we list its limitations, and draft the coming work for finalizing the indexing and retrieval prototype.

2 Map Reduce Framework for Structure Identification and Localization

In this section we describe the framework which was developed to perform the indexing. It allows for structuring the localization components, and the execution of localization and structure identification. The system allows for further development of image processing and analysis algorithms independently from scaling, parallelization, dependency checking, data persistence or computation scheduling.

After presenting the aims of the framework we provide an overview of how these aims are met, followed by an in-depth description of the programming model of the system. Subsequently some implementation considerations are described and a reference section for the entire programming model is provided, together with a walk-through of the current state of the graph in regards to the clinical 3D retrieval prototype.

2.1 Aims of the scientific computing framework

Provide a clear structure of the entire system The many parts of complex, state-of-the-art computer vision processing pipelines need to be easy to define, to represent and to communicate across developers.

Allows to focus on the core methods Researchers should be able to focus on their core competence and develop novel algorithms. The systems engineering aspects required in large scale systems should be handled by the system.

Ensure that results match the current input / parameter settings In complex computing systems that process different data sets, and contain a variety of components, dependencies, and parameters, reliable dependency and parameter tracking is necessary. Managing this manually is error prone. In practice it often leads to uncertainties regarding intermediate algorithm-, data-set-, parameter- and result versions. The developed framework should track the changes of the algorithms, data-sets and parameters and guarantee that the results returned are computed exactly as specified in the system definition.

Allow to reuse computations As one of the consequences of the above uncertainties, intermediate results are often recomputed to verify that they are up to date. The framework should automatically and reliably reuse intermediate results where possible.

Handle caching / persistence The handling of data persistence shall be done automatically. Researches shall never have to load a certain file with a certain naming convention which (hopefully) contains the right intermediate results from last year.

Be easy to use The framework programming model should be easy to understand and use even by novice programmers. System definition should be possible with a minimum number of different semantic concepts.

Allow to quickly build and evaluate variants of the system One of the requirements of day to day research is the ability to run only sub-parts of the system, with different parameters, on different data-sets. The framework should make this as easy as possible and not pose any versatility restrictions.

Reproducibility The result of a run should be completely determined by the systems definition. E.g. persistent intermediate results can be deleted at will with the next request from the system returning exactly the same results, triggering intermediate computations as necessary.

Scheduling / parallelism / scalability The user should not be concerned with starting or scheduling the computations. The inherent parallelism in the systems should be automatically exploited and computations should be scheduled on as many cores and machines as are available at any point.

Suitable for off-line and on-line tasks The framework should optimally support long-running, off-line tasks as encountered in large-scale machine learning, as well interactive on-line tasks such as those occurring in the back-end of an image retrieval application.

Insight into computing activity The framework needs to provide the researcher with a detailed overview of the currently running computations and the status of each part of each system.

Insight into (intermediate) results The results generated in each node of the system should be visualized so the researcher can inspect and diagnose the behavior of the system.

Access to intermediate results Even while computationally costly processes with long run times are still running, the framework should provide easy access to the results of the already performed intermediate computations.

Computing platform The framework should be operating system agnostic and also work in heterogeneous environments. The Matlab programming language should be used, as it is the most widely used language in the computer vision and machine learning community for algorithm development and rapid prototyping.

Performance / framework overhead The time and space overhead required by the framework in addition to the actual node runtimes / storage requirements should be minimal.

2.2 Overview of the scientific computing framework

The proposed computing framework is based on the idea of structuring the calculations as a *map-reduce graph*, as popularized by the seminal publication [1] by Google and widely implemented in the Hadoop framework [1]. It is strongly influenced by systems aiming to separate the algorithmic idea from the practical details of data-handling, persistence and scheduling such as the Halide programming language[2].. We extend the basic idea of map reduce to fit within the specific requirements of practical research on the one hand and large scale computer vision systems on the other hand. Structuring the computations as a map-reduce graph allows to address all the objectives raised in section 2.1:

[1]http://hadoop.apache.org
[2]http://halide-lang.org

Clear structure The map-reduce graph models the data-flow of the system, which is an intuitive way of looking at the entire system of computations.

Focus on core methods The framework drastically reduces the gap between the core algorithm and its application in practice. The nodes directly implement the scientific core of the system, while the entire management of data and computations is handled by the framework.

Results matching definition As there is no manual interaction for running experiments and handling data the framework can ensure that the results provided match a given system definition. This is ensured even in the case of complex and long-running tasks with large scale data.

Reuse computations The framework is dependency-tracking, i.e. when the algorithm in a node or a parameter governing a node's behavior changes, only the computations affected by this change are rerun.

Handle caching / persistence For each node, i.e. a computation definition, it can be specified whether the node's output should be persistent, i.e. saved on disc. Whether a node's output should be saved or the node run every time is a space / time design choice which is controlled by a simple parameter. As a node is not concerned with persistence or scheduling no changes to the node are required.

Straightforward to use The system has been developed to make it easy to use. The definition of the graph is a simple, human readable, hierarchical data structure and the node definitions follow a very simple format.

Allow to quickly build variants of the system Often the final result of a research project is a single system with one set of data and settings. During day to day research many variants of the elements of a system are investigated, by different researchers, on different data sets. The framework makes no assumption of one master system with variants, but each use case can be its own system. Only due to the fact that parts of the systems are overlapping the benefits of i.e. the reduced number of computations are achieved.

Reproducibility As both the framework and the definition of the system are fully versioned and each step is dependency tracked, results can be reliably reproduced. For example, intermediate results can be safely deleted, as they will be recreated as necessary.

Scheduling / parallelism / scalability A major design goal of the system is the separation of the algorithms and the data handling / computation scheduling. The research is only concerned with algorithm design, not with data persistence or with running compute jobs. The systems definition and a result request provide enough information to the framework such that it is able to produce a response to the request. Due to the natural parallelism throughout much of typical map-reduce graphs, the computations can be automatically scheduled across several cores and machines in a cluster. This also provided automatic scalability even to very large scale problems.

Insight into computing activity As the framework is designed to handle many computations with long run times it is important to provide the user with information on the status of each request. This is provided through very detailed status pages which show compute progress as well as detailed error messages for failed computations.

Insight into (intermediate) results As part of the above status pages detailed information of each result of each node is provided. This includes a graphical representation of the result as well as numerical properties of the result, like minima and maxima. Additionally, for each node summary statistics of the results are provided, e.g. the mean value of all individual minima, or the presence of Not-a-number values in any of the results. This information provides in-depth insight into the systems results and provides a powerful means for diagnosis.

Access to intermediate results Each intermediate result is easily accessible as soon as it is computed. Not only in the status pages but programmatically, for each node the already computed results can be easily loaded. For example, when thousands of images need to be preprocessed, the researcher can already start working with the first few results, while the entire data set might take weeks to compute.

Computing platform The entire framework is implemented in Matlab. It can be easily used on all major operating systems and hardware platform. Its internal scheduler allows to easily deploy it on a cluster with shared storage.

2.3 Terminology

To clarify the following presentations a small set of terms is defined, in the context of the proposed framework. A graphical representation of these terms can be seen in Fig. 1.

Framework The framework allows to defined, nodes, graphs, domains and parameters and can subsequently answer the requests of the researcher.

System A system is defined using the frameworks, and consists of a graph, the nodes, one or more domains and some parameters. Once a system is defined and made known to the framework, a specific request for one of the nodes can be answered by the framework.

Item An item is the smallest entity of the systems input. For example, the filename of a filename, or a number, or a matrix.

Domain A domain is a ordered list of data items marked for processing. For example, in an image retrieval training system, the domain might be a list of image filenames, or a list of identifiers (IDs) that define a set data set of entries in a database.

Graph The graph defines the data-flow of the defined system. It consists of nodes, which are connected to form a data-flow or dependency graph. Throughout this document, graphs are drawn such that data flows from top to bottom, i.e. a node depends on the nodes above it to which it is connected to. Each item in a domain is pushed trough the graph from top to bottom.

Node A node encapsulates a specific algorithm or operation. For example in a simple image processing pipeling the graph might consist of the nodes ”Image”, ”Kernel”, ”Filtered Image”.

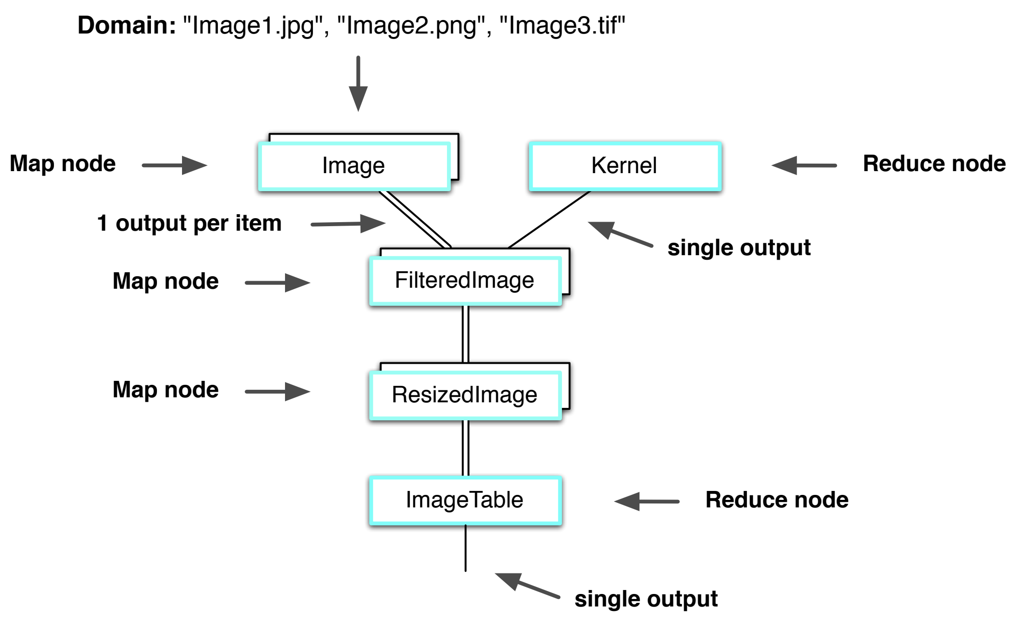

Figure 1: Simple example of a map-reduce graph. Double borders indicate map nodes and per-item connections, single borders indicate reduce nodes which only produce a single output.

Map node A map node performs an operation on a single item. Either the item is part of a domain or the output of a precursor node. It can perform any operation , while ensuring that given a specific version of the node (specified as a hash value of the definition of a node), a specific input and the same set of parameters it always returns the same output.

Reduce node A reduce node sees all items of a domain at the same time. As such it can for example compute their average, or compute an index of all the items.

Request A request tells the framework, given a graph, which node should be computed over which domain (and with which parameters). The framework can either invoke the necessary computations or load an already computed version.

Parameters Parameters are always bound to a specific node and can thus influence its behavior. The framework assumes that the output of a node can be different for two different values of a parameter.

2.4 Illustrating Example

To explain the main components of the system, in the following we will walk through an example system definition, shown in Fig. 1 before describing the components in detail in section 2.5. We will look at the most important properties described in the previous section in the context of this example. In the example, we are computing an image composed of several thumbnails, which get processed / filtered, then resized to a common height and are finally assembled into a single image where they are placed left to right.

The domain is composed of three items, in this case the ordered strings `Image1.jpg`, `Image2.png` and `Images3.tif`. Each of the images is loaded in node `Image`, it is thus a map node. It gets e.g. `Image1.jpg` as input and has to return the corresponding image as a matrix.

The next node, `FilteredImage`, whose task is to pre-process the image, depends on both the output of `Image` and `Kernel`. `Kernel` is a node which is independent of the domain, and always returns a Gaussian kernel of a certain width (specified by its parameters). It is thus a reduce node, providing a single output. `FilteredImage`, being a map node, is computed for each item and gets the loaded image and the kernel as input. It performs its filtering operation and return a filtered image. In `ResizedImage`, again a map node, the input image is resized proportionally to a certain height. Finally, `ImageTable` has the task of combining all images. It thus needs access to all items, and is a reduce node. It produces a single output: the combined image.

Framework behavior during algorithm development Let's view the framework's behavior in the context of ongoing algorithm development. This involves many variations and iterations of experiments on the system. For example, what happens if the height parameter of node `ResizedImage` is changed and the output of `ImageTable` is requested? The framework, through the dependency graph, knows that only the output of `ResizedImage` and `ImageTable` can be affected by a change to the height parameter. It can therefore safely load the results of `FilteredImage`, apply the operation in `ResizedImage` and then `ImageTable` and return the result. Note how the minimal number of necessary computations was performed. Additionally, all outputs are versioned, so a request for `ImageTable` with the original height parameter will return immediately without computations, with the loaded result for `ImageTable`.

What happens if we change the domain? Let's assume a forth image filename, `Image4.jpg` gets added. And again we request `ImageTable`. The only computations performed by the framework are the invocations of `Image`, `FilteredImage` and `ResizedImage` for the new item, as well as a call to `ImageTable`. If we change the domain to consist only of `Image1.jpg` and `Image2.png`, only `ImageTable` will be invoked, as this is the only node whos output is not yet known.

All requests and changes to parameters can be performed independently by different researchers at the same time, without any conflicts, and the framework ensures to always return the results corresponding to the system's definition, with the minimal number of necessary computations performed.

2.5 Graph and node definition

In the following we describe the definition of the graph and its nodes, using the example described above. The definition is performed by specifying a Matlab structure, where each field corresponds to one node in the graph. Each of these fields is in turn a structure, so the definition of a graph is a single, hierarchical data structure.

Each node can specify a function handle for either a map- or a reduce function, as well as the nodes it depends on and, optionally, default parameters. Additionally, each node has several properties which control how the framework performs persistence and scheduling, but these properties do not change the behavior of the algorithm specified in the function handle. This is part of the reason why the algorithmic details are entirely separated from the persistence / scheduling details.

The following code specifies the algorithm presented in section 2.4. It shows the hierarchical data structure used to define a system depicted in Fig. 1, with each node specifying either a map or a reduce function and accompanying parameters as well as the dependency definitions.

```
graph.Image.map = @(p,item) fm.imread(item);
graph.Kernel.reduce = @(p,items) fspecial('gaussian',[6 6]*p.std, p.std);
graph.Kernel.params.kernel.std = 1;

graph.FilteredImage.map = @(p,item) imfilter(item.Image,item.Kernel);
graph.FilteredImage.needs = {'kernel', 'image'};

graph.ResizedImage.map = @(p,item) imresize(item.FilteredImage, p.height, ...
  size(item.FilteredImage,2)*p.height/size(item.FilteredImage,2);
graph.ResizedImage.needs = {'FilteredImage'};
graph.ResizedImage.params.ResizedImage.height = 256;

graph.ImageTable.reduce = @combineImages
graph.ImageTable.needs = {'ResizedImage'};

function r = combineImages(p,items)
r = cell(1,numel(items));
for i = 1:numel(items)
  r{i} = feval(items{i}.ResizedImage);
end
r = cell2mat(r);
```

2.6 The 3D Retrieval prototype graph

The entire 3D retrieval system is structured as one connected map-reduce graph, depicted in Fig. 2. This entails both the off-line, long-running part of indexing the visual and semantic data (above the red line) as well as the interactive, on-line back-end for the retrieval graphical user interface (GUI) (below the red line).

In Fig. 2, data flows from top to bottom, where each double-bordered box is executed once per medical volume (such as those components computing visual features for individual volumes). Single-bordered boxes are executed once (such as the indexing operations over all volumes, returning a single index).

A detailed description of the entire retrieval system shown in the graph is provided in deliverable D9.4.1. In the following, we will thus focus on the nodes which are part of the localization module.

Fig. 3 gives a more detailed view on the framework components that perform localization and anatomical structure identification during indexing.

Volume alignment and atlas registration The main tasks of the data preparation are the coarse alignment of the volume to a whole body reference, followed by an affine and non-rigid

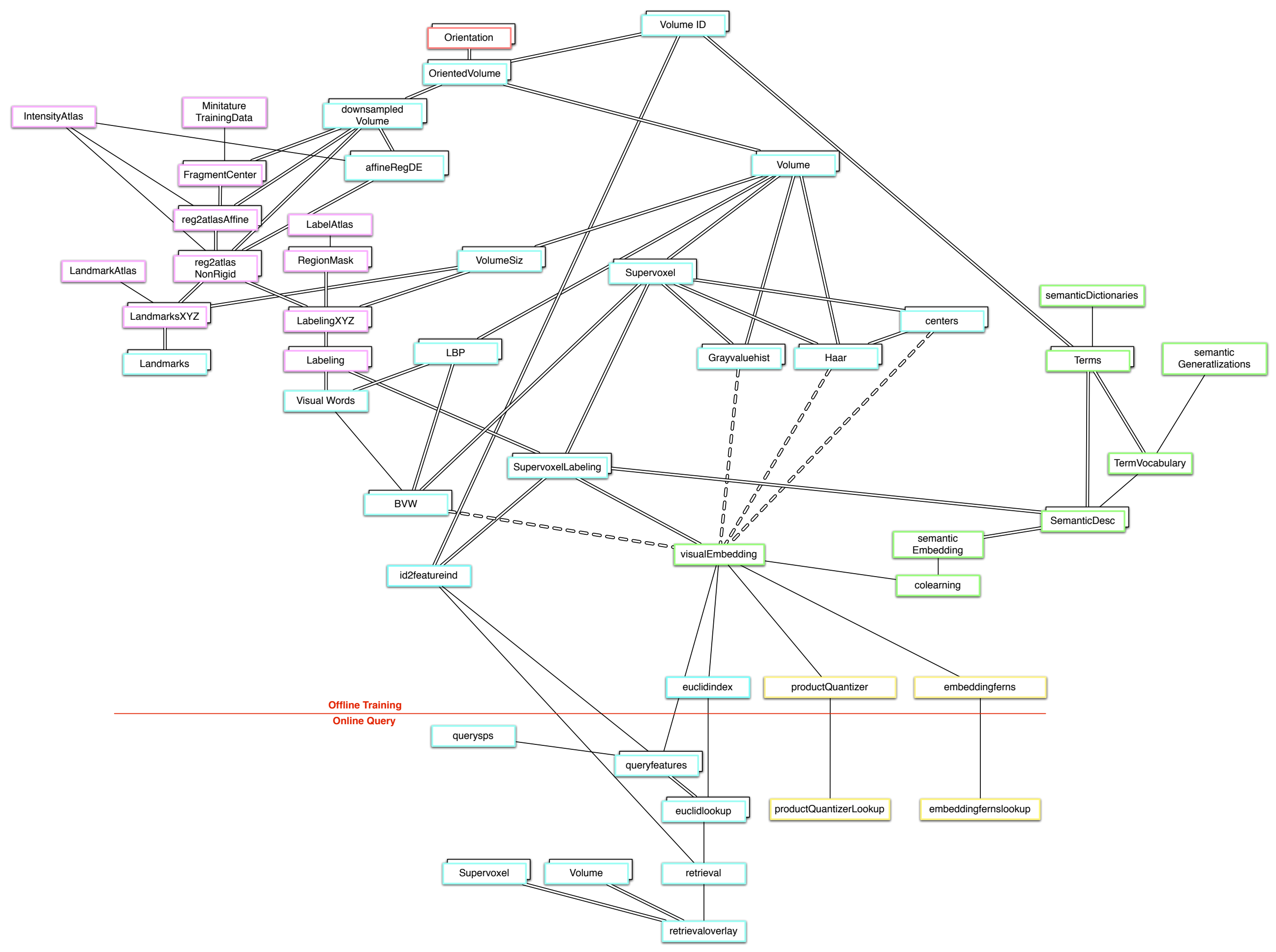

Figure 2: Current state of the MUW 3D retrieval map-reduce graph. Note how the graph describes both the long-running off-line training as well as the interactive, on-line retrieval back-end powering the user interface.

registration to the corresponding atlas. Currently, the whole body atlas consists of a whole body CT template (which is represented by node **IntensityAtlas**) and has organs labeled by medical experts on a per-voxel basis (represented by **LabelAtlas**). This atlas will be extended regarding additional modalities (MRI, contrast enhanced modalities). Separate indices can be built for differnt body regions, and the nodes **regionID** and **regionMask** are in charge of masking the relevant subset of the label atlas. Each volume is first oriented according to the DICOM header (**Orientation/OrientedVolume**) and down-sampled to match the miniature resolution necessary for the initial coarse location estimate [3]. **MiniatureTrainingData** and **Fragment-Center** estimate the approximate body region of the volume in question in the coordinate frame of the whole body reference. This information is then used to initialize an affine registration (**reg2atlasAffine**). The output of this registration is used as initialization for the non-rigid registration of the volume to the atlas.

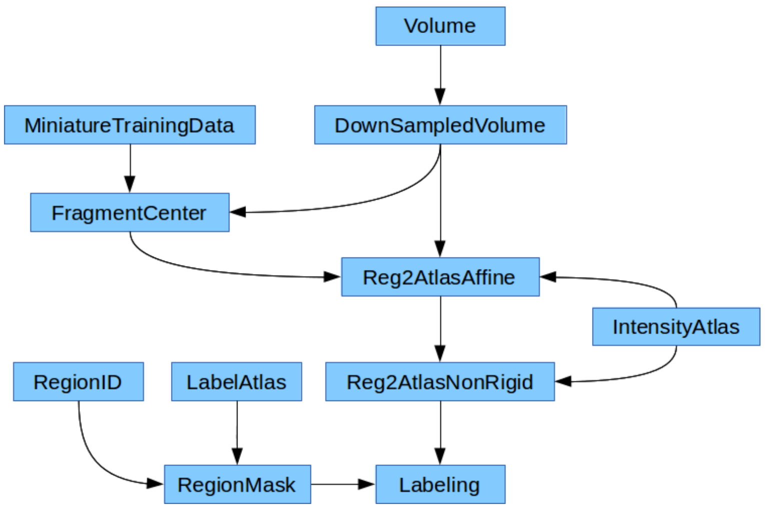

Figure 3: Graph of the localization module

Organ label mapping Once the registration is finished, the annotated organs are transferred from **LabelAtlas/RegionMask** that corresponds with the template using the spatial transform obtained by the registration steps. This provides a **Labeling**, of the same size as the input volume. Each voxel in the input volume is assigned an organ label.

3 Localization components developed in KHRESMOI

In the following we outline the algorithmic components developed that are relevant for the graph nodes performing localization. Part of them have been described in Deliverable D2.2 (features), Deliverable D2.3 (evaluation), and Deliverable D2.4 (learning). The algorithms that for the localization framework are designed with the objective to maximize generalizability, and scalability. There are three main components that have been developed in the course of KHRESMOI and are part of the localization module:

1. A miniature based coarse localization of medical imaging volumes forms the initial step in the current localization process.

2. Atlas mapping refines the initial correspondence, and assigns each voxel in each volume a corresponding position in a reference space that is linked to a label atlas.

3. Accurate localization of anatomical landmarks in the indexed volumes - in addition to organ labels - is the objective of two algorithms. The first uses local appearance together with a statistical shape model to obtain an accurate location estimate of landmarks. The second algorithm is able to localize sets of landmarks with very high speed.

3.1 Coarse miniature based localization

This algorithm consists of a training component (`MiniatureTrainingData`) that is executed during indexing , and a localization component that is executed during retrieval (`FragmentCenter`).

In this first step the indexing framework identifies the coarse position of a medical imaging volume in a whole body reference space [2]. During training, the center positions for a corpus of several thousands of volumes are annotated by experts. For these volumes miniatures are constructed that form a sampling of the possible appearances sufficiently dense for retrieval. This is similar to [8], but due to the constraint anatomical domain, a significantly lower number of examples is sufficient to sample the appearance space. The training algorithm builds a kd-tree [7] from the miniature descriptors and during localization the center position of a volume is estimated by means of k-nearest neighbor regression based on the annotated examples. A detailed description of the algorithm and a corresponding evaluation can be found in [2] and Deliverable D2.3.

3.2 Mapping anatomy labels to individual volumes

After estimating the coarse position of each volume in relation to the human anatomy a fine grained mapping between a whole body reference and the volume is calculated via fragment registration. The fragment bundling algorithm that constructs the unbiased whole body template has been published in [6] and has been explained in detail in KHRESMOI Deliverable D2.4. During volume to template mapping, each volume is registered to the segment of the whole body template that corresponds to its likely position determined by position estimates in an iterative process. Following the initial miniature based localization, an affine, and a non-rigid registration of the imaging data to a whole body template are performed. They yield a mapping of each voxel in the volume to a position in the whole body reference space. In KHRESMOI Deliverable D2.4 and [6] we describe how to learn such as template from a large set of medical imaging volumes. To assign an anatomical structure label to each voxel a label atlas that corresponds to the whole body template is mapped to the individual volumes based on the transforms obtained by the registration. This label atlas is hierarchical following the RadLex ontology. After this mapping features and indexes that cover specific anatomical structures at different levels of detail (e.g., liver, abdomen) can be built from the entire population.

3.3 Landmark configuration localization

In addition to whole organ mapping, we have developed components that can localize individual anatomical landmarks in medical imaging data. The algorithm that detects landmark configurations follows a two step approach and has been published in [5]. Its learning and localization methodology is described in detail in KHRESMOI Deliverable D2.4. Given an input volume the algorithm first generates a set of hypotheses locations for each landmark. This hypotheses are based on a global classification of image appearance based on local Hough Forests. In the second step the possible configurations of landmarks formed by all hypotheses (i.e., landmark candidates) are disambiguated based on a statistical shape model learned during training on a small annotated sample set.

A Markov Random Field (MRF) represents the relationships among the landmarks, and the match between the appearance of each hypothesis and the corresponding prototype. The optimal labeling of the MRF yields a reliable estimate for all landmark locations that are part of the configuration. The algorithm performs global search in a volume without the need for initialization. Furthermore, the individual label weights assigned to the MRF by the observed data can serve as a means to detect outliers, or missing landmarks. Typically the number of landmark candidates is low. This is a critical property that allows the algorithm to scale well to high-resolution 3D imaging data.

3.4 Fast landmark localization

Similar to the approach described in Section 3.3 an alternative landmark localization approach focuses on improving speed during localization. The approach, first published in [4], is divided into a training phase and a localization phase. During training the algorithm creates a multi-scale codebook of image patches and landmark positions. It represents local appearance that is specific to landmarks. During localization this codebook is traversed starting from coarse scale image representation to increasingly higher resolution image patches. During this iterative process that starts with a representation of the image similar to the miniature resolution the landmark estimate becomes increasingly accurate. At each step landmark estimates are regularized by a linear statistical shape model, that represents the variability in the training set population based on their covariance structures. The resulting algorithm is extremely fast, while achieving high accuracy and reliability in landmark localization.

4 Conclusion

The present deliverable describes the computational framework for localization and anatomical structure identification in the KHRESMOI large scale biomedical image retrieval system. It draws on methodology developed in the course of the project, and described in previous deliverables. The focus of this deliverable is the description of how these algorithms are integrated in an indexing and retrieval framework.

During indexing the localization and structure identification assigns each voxel in the indexed data a location in a whole body reference space, and a corresponding anatomical label. During retrieval query regions are mapped to this atlas, and corresponding location specific indices are used for pathology specific retrieval.

Even-though the current status yields promising results there are several limitations that have to be addressed in the future. First, a single whole body reference space is limited in the representational power of the substantial variability present in the population and corresponding approaches have to be developed that accurately represent anatomical details. Multi-modal approaches have to be devised to accurately match corresponding data that is acquired in different modalities. The accurate localization of landmarks has to be integrated in the voxel labeling and organ identification framework. Lastly, the individual components have to be optimized to increase robustness with regard to pathological changes, and overall reliability and accuracy in location mapping, and labeling. We will work towards addressing these issues in the final project year.

5 References

[1] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *OSDI*, 2004.

[2] Rene Donner, Sebastian Haas, Andreas Burner, Markus Holzer, Horst Bischof, and Georg Langs. Evaluation of Fast 2D and 3D Medical Image Retrieval Approaches based on Image Miniatures. In *Proc. MICCAI Workshop on Medical Content-based Retrieval for Clinical Decision Support*, 2011.

[3] René Donner, Sebastian Haas, Andreas Burner, Markus Holzer, Horst Bischof, and Georg Langs. Evaluation of fast 2d and 3d medical image retrieval approaches based on image miniatures. In *Medical Content-Based Retrieval for Clinical Decision Support*, pages 128–138. Springer, 2012.

[4] René Donner, Björn Menze, Horst Bischof, and Georg Langs. Fast Anatomical Structure Localization Using Top-down Image Patch Regression. In *Proc. MICCAI Workshop on Medical Computer Vision*, 2012.

[5] René Donner, Björn Menze, Horst Bischof, and Georg Langs. Global Localization of 3D Anatomical Structures by Pre-filtered Hough Forests and Discrete Optimization. *Medical Image Analysis, in press*, 2013.

[6] Matthias Dorfer, René Donner, and Georg Langs. Constructing an un-biased whole body atlas from clinical imaging data by fragment bundling. In *Proc. MICCAI'13*, 2013.

[7] Beng Chin Ooi, Ken J McDonell, and Ron Sacks-Davis. Spatial kd-tree: An indexing mechanism for spatial databases. In *In Proc. IEEE COMPSAC Conf*, pages 433–438, 1987.

[8] Antonio Torralba, Rob Fergus, and William T Freeman. 80 Million Tiny Images: A large Data Set for Nonparametric Object and Scene Recognition. *TPAMI*, 2008.