Formal extraction of biomedical objects by subgraph matching in attributed hierarchical region adjacency graphs

Christian Thies[a], Volker Metzler[b], Thomas Lehmann[a], and Til Aach[b]

[a]Department of Medical Informatics, Aachen University of Technology, Germany
[b]Institute for Signal Processing, University of Luebeck, Germany

ABSTRACT

Extraction of objects from biomedical images is the fundamental task for many high level applications in medical image processing such as cytometry or diagnostic decision support. Therefore, a formal specification of sought objects is required along with an extraction procedure. On the basis of a hierarchical image decomposition objects are described by image regions of characteristic shape, texture, and visual context. For example, a cell consists of a circular core, a surrounding body containing organelles, which is in turn surrounded by the nutrition agent, and other cells. This is modeled by hierarchical graph representation of the region topology as nodes and the region properties as node attributes. In a hierarchical region representation, an object is described by subregions which again may contain subregions, thus object extraction becomes the matching of the respective region nodes. Obviously, graph matching is a NP-complete problem and therefore, it requires heuristics to become computable. This even holds for subtree matching. We propose a new approach which makes strongly use of the inclusion property of regions in a hierarchical image decomposition along with the visually descriptive attributes. The algorithm iterates a top-down bottom-up sequence over the region hierarchy to restrict the search space. Hence at each step, a layer of tree-node attributes must be compared to the attributes of the sought objects root node description. The bottom-up analysis is only invoked for the subtree depending on those nodes. Thus, each node is visited according to the topology of its visual occurrence in an image.

Keywords: Object extraction, content description, image retrieval, region adjacency graph

1. INTRODUCTION

Quantitative object extraction from images is an important task in high level image understanding, which is strongly linked to the segmentation problem. In typical applications, image segmentation is applied to separate an image into object and background information.[1] Such algorithms depend on the one hand on the extraction of visually comprehensible image features, e.g. inflection contours[2] and on the other hand on their formal description, which is the knowledge about the sought objects.[1] In the last decades, three different levels of abstraction for formal object description have emerged, viz. regional description via feature vectors, structural components and their topology, e.g. as half-circle above horizontal line, and finally complete models which describe an entire complex shape.[3] Each object formalization requires a set of rules with an increasing level of abstraction and specificity.[4] Specialized applications can fall back on a-priori domain-specific knowledge to extract the sought types of objects with a specialized extraction procedure.[5] But in general purpose applications for object extraction such as image retrieval, the type and formalization of objects cannot be predicted a-priori. A general purpose algorithm must respect the multi-scalar character of visual information[6] along with an easy-to-use parameterization scheme.

Content-based image retrieval is a good example for the need of high level object extraction. Here, the extraction of images which contain a certain object from a heterogeneous database is one of the fundamental tasks.[7,8]

Correspondence: Christian Thies, Department of Medical Informatics, RWTH Aachen, Pauwelstrasse 30, 52057 Aachen, Germany E-mail: cthies@mi.rwth-aachen.de, Telephone: +49 241 8088795

Medical Imaging 2004: Image Processing, edited by J. Michael Fitzpatrick, Milan Sonka, Proceedings of SPIE Vol. 5370 (SPIE, Bellingham, WA, 2004) · 1605-7422/04/$15 · doi: 10.1117/12.536019

Following classical object extraction approaches, the user has to apply a special object extraction algorithm to each image in the database in order to find the sought object. The number of those algorithms will grow each time another object is sought. Furthermore, those algorithms require fundamental knowledge in basic image analysis and software development. Besides the administrative and software-engineering efforts, a large number of costly low-level computations is required for each new query. But image retrieval depends on fast response times to queries since the user must decide whether the result matches his needs within a feedback cycle.[9] Therefore, a flexible as well as efficiently computable description is required which can be easily extended during runtime.[10] The necessary preparation of the image data can be computed off-line at the data-entry time and is not that time critical.[9]

There are approaches which decide the similarity of images on the basis of contained regions and which invoke single scale segmentation techniques.[7,8] But those systems do not consider an object as an entity instead of they regard the extracted regions as indices in their global image feature description. Furthermore, they do not consider an object as a combination of sub objects thus their decision is only based on the descriptive power of the features and makes no use of structural properties. Content description consists of descriptive features and of topological relations.[11]

Furthermore the extraction of objects in large image series depends on appropriate indexing techniques for fast processing.[7] Both requirements do mutually influence each other. A flexible content description leads to an information explosion and consequently to large and complex-structured data, while fast indexing requires small and well-ordered data. Current image retrieval approaches integrate content description and extraction as well as indexing into one step, where the indexing order directly depends on the low-level image processing operations.[7,12,13]

In this paper, an efficient object description and query approach is proposed that integrates visual information organization, efficient object description, and fast object extraction. The approach consists of a multi-scale graph representation of visual content, an appropriate hierarchical query object formulation in a domain-specific language (DSL), and the appropriate evaluation strategy which is executed on a virtual machine. The reminder of the paper is structured as follows. In Section 2, the graph based organization of objects is described followed by the language definition (Sec. 3). The strategy to execute queries is proposed in Section 4. Experiments and their results are shown in Section 5 and the paper closes with a discussion and an outlook in Section 6.

2. HIERARCHICAL OBJECT REPRESENTATION

Natural scenes consist of objects which are separated from each other in a physical way. Thus, separating properties of the surrounding objects can be defined for each single object. Furthermore, real objects consist of sub-objects consisting of sub-objects and so on. Those two basic facts form the paradigm for the hierarchical object representation.

2.1. Visual perception and extraction

In image processing, visual object extraction means the detection of the interfaces between two objects which consequently leads to edge detection. But the problem can also be formulated vice-versa as a connected component analysis[14] since each natural object forms an entity with intrinsic connectivity properties. Now, object extraction becomes a region detection. Each detection algorithm has to consider this duality.[15]

However, a general extraction approach that is based on a low-level object entity formulation has to detect any visual plausible entity which might form an object. Such an approach does not segment an image in a semantic way, where the objects are described in terms like background or foreground. On the contrary, the whole image is completely partitioned in visually distinguishable regions. This is the reason why classical edge detection algorithms such as the Canny-operator are basically not suitable for region-based object analysis. They do not provide reliable closed contours, i.e. the complete object interfaces. In this context, it is helpful to distinguish between image regions and objects. Image regions reflect a visual property without any semantic meaning and an extraction algorithm must extract such visually relevant features of an image. Image contrast is a major feature for human perception that can be modeled as the zero-crossings of the second derivative of the intensity surface.[2] In two dimensional (2-d) images this leads to closed inflection contours which form the respective image regions. Inflection contours can be extracted without any high-level knowledge on actual objects. This allows a complete separation of visual information extraction and content description.[16]

2.2. Hierarchical organization of visual information

Visual information is organized in a hierarchical way because each corresponding object is again build from sub objects. This coarse-to-fine reduction principle can be transferred to visual information extraction (Fig. 1). Partitioning of an image can be conducted from the entire image down to image pixels, as the atoms of digital image decomposition. In the fine-to-coarse direction, the geometric size of an image is the limit. In between, each possible combination of image pixels can be arranged in a hierarchical way since small regions are included in large regions if and only if they are a union set. Combining the inclusion and the connected component property of image regions the basic principle of hierarchical image decomposition algorithms is defined.[14] Operators that respect connectivity and union construct a causal image region hierarchy.[16,17] From a set theoretical point of view, this data structure forms a complete lattice if the empty set is added. A helpful observation is that if an algorithmic problem operates over a lattice, it can be solved by greedy approaches. This is useful for the design of efficient algorithms.

2.3. Representation of objects as subgraphs

A multiscale decomposition is obtained by operators which reduce image details by extension of their zone of influence. They have to extract connected components with respect to the visual plausibility of their interface. However hierarchical representations can only be extracted by causality preserving operators. This means each contour pixel of a coarse scale must be represented at exactly the same position in finer scales.[18] Consequently a causal multi scale decomposition yields a set of regions with visually perceivable contours linked by their topological adjacency and full inter-scale inclusion relations. Since the image is completely partitioned each region can be seen as a node and each adjacent neighbor as an edge. The resulting data structure is a region adjacency graph (RAG).[19] Additionally, the inclusion relations form a tree structure over the nodes. Thus, the hierarchical aspect of multi scale object information is also reflected. The RAG is extended to a hierarchical attributed region adjacency graph (HARAG). Besides the structural information, a region is described by a set of basic attributes such as size, element array, contour, and number of sons, which reflect the raw data of the region. Those attributes can be aggregated to more complex attributes, such as mean grey value, convexity, or Fourier descriptors. The set of descriptive attributes is infinite, redundant and ambiguous, but those attributes form an important part in object description. In the concept of object extraction, they are part of the observer's knowledge on image content. He must be enabled to select and manipulate them in order to obtain the sought results. Thus, attribute aggregation forms the first part of this approach to object extraction.

2.4. Extraction of objects as sub-graphs

Objects are subgraphs of an HARAG. In simple cases, an object is a single region with distinct feature values. Then object detection becomes a query to the HARAG, where single regions with the defined attribute values are selected. Sometimes, different objects with similar appearance have to be extracted separately. Figure 1 illustrates the schematic extraction of cellular structures, where a cell consists of an ellipse (1,3), circular cores (4,6) and some rectangular (9,7,8) and elliptical (5) ribosomes. In one case, the core contains two sickle shaped chromosomes (10,11). Aim of the query is to extract the cell with the sickle shaped structures. Therefore, a query is defined which describes an ellipse as a root node (A), a circular substructure (B), and a rectangular substructure (X) as sons. The node (B) is supposed to have a sickle shaped sub node (Δ). This example illustrates the principle of the approach. It is obvious that there are some heuristics in the approach. The first one is the assumption that objects consist of sub objects, thus there always exists a root node which sufficiently describes the top-level object. In general this root node is formed by the entire image. The next heuristic assumes that the number of respective sub nodes is not determined. For instance the number of sickles (10,11) in the core (6).

The proposed method differs significantly from classical graph matching approaches. The costly comparison of

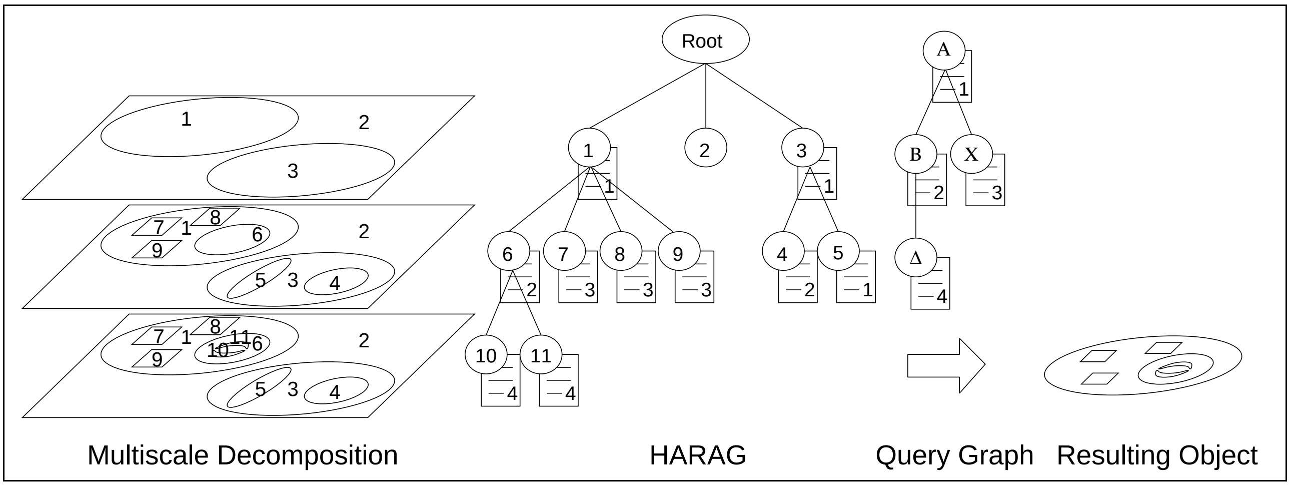

Figure 1. From a multiscale decomposition, an attributed HARAG is obtained. The HARAG describes the inter-scale inclusion relation for each region between the scales. By using the region attributes a query graph can be defined to describe a subgraph. This query graph is matched against the image graph and all matching components are extracted as the resulting object.

each node of the image graph with each node of the query graph is avoided by following the inclusion property. Furthermore, the graph representation of the object can be transformed into a formal query language.

3. THE QUERY LANGUAGE

We regard object description as the extraction of a subgraph from the HARAG. Once an extraction query has been formalized, it can be applied to arbitrary images which have been transformed into the graph structure. In medical applications, the need for formal knowledge representation on sought objects meets the non-formal character of verbal biomedical object descriptions. Thus, a language definition must be as flexible as possible to cope with the ambiguous and non-deterministic object descriptions. In contrast to specialized systems, where the knowledge can be introduced already for low-level operations (e.g. thresholds), a generalized approach depends on abstract formalization techniques. By now, model-based approaches are not applicable since designing a querying by models is a cumbersome task[20] which does not fit in the concept of fast query generation and application.

In a general purpose object extraction mechanism a generalized formalization of objects is required. In our system this is realized via a domain specific query language with structure specific evaluation strategies. The language is organized on three different levels of abstraction which integrates three standard components of a formal language. At first, numerical expressions to generate new attributes from the interpretation free raw data is described (Sec. 3.1). Secondly, formulas in first order predicate logic (FOL) form the static definition to select respective regions (Sec. 3.2). Finally, the dynamic component as the main cycle of the virtual machine (VM) is introduced (Sec. 4).

3.1. Attribute formulation

The image decomposition process yields a data structure which is entirely self-contained and no knowledge on image content is introduced besides the decision which regions are visually comprehensible. Thus, the HARAG consists of nodes which contain a generic set of attributes that is fixed and needs no formal interpretation. This basic set ATTRIBS looks as follows:

- `area`: The size of the image partition in pixel.
- `creation`: The first occurrence of the region during scale-space build-up.

- `absorption`: The last occurrence of the region during scale-space build-up.
- `sons`: The number of successors in the tree.
- `neighbors`: The number of adjacent nodes in the tree.
- `Lsons`: The list of the successing nodes.
- `Lneighbors`: The list of the adjacent nodes.
- `Apos`: The array of positions of the connected pixels which form this region.
- `Image`: The grey-value array of the image.
- `perimeter`: The length of the partition contour.
- `Cpos`: The contour positions of the image partition.

Along with the region graph G, the image I is known from where the actual HARAG $G(I)$ originates. Each kind of knowledge on objects is an interpretation of this basic set of features. Since the features are ordinal values, their interpretation is a numerical aggregation. This meets the demand of computational formalization. Complex attributes can be generated by formal transformations of elements from the basic set of attributes. Thus, only the basic set of attributes must be stored in permanent memory, e.g. the hard disk.

In the actual implementation some redundant attributes are stored along with the basic attributes with respect to computation time. By now they consist of the contour information perimeter and `Cpos` which can also be extracted from the `Apos`-array. Those attributes are associated to each region-node n of the HARAG. The following set of arithmetic and higher order functions is currently integrated into the language.

- `hist()`: The intensity histogram of a region (FUNC1)
- `hist.size`: The size of the histogram of a region (FUNC1)
- `bins`: The non zero bins of the histogram of a region (FUNC1)
- `+, -, *, /`: The basic arithmetic expressions (ARITHOP)
- `log, sqrt`: Higher order functions (FUNC1)
- `fds(n)`: The first `n` Fourier descriptors of a regions contour (FUNC1)
- `dist(A,B)`: Computing the euclidian distance between two arrays A and B (FUNC2)
- `pow(A,B)`: Compute the power `B` of `A` (FUNC2)

New attributes can be aggregated by expressions consisting of the basic set of features, constants and functions in the following grammar.

Basic Variables:

```
01 const -> [0-9,.]+                           //Real value constants
02 var -> [A-z]+                               //Variable of type double or Array of double
03 attrib -> ATTRIBS                           //Each known predefined Attribute
```

Expressions:

```
04 exp -> attrib                            //area
05 exp -> const                             //3.1416
06 exp -> var                               //Variable name
07 exp -> FUNC1(exp)                        //log(area)
08 exp -> FUNC2(exp,exp)                    //pow (perimeter,2.0)
09 exp -> exp ARITHOP exp                   //(3.1416 * area )
10 exp -> ( exp )                           //Nested parenthesis
11 exp -> FOREACH i IN VAR ARITHOP (exp(i)) //Apply exp to each i in VAR
                                              and aggregate by ARITHOP
12 exp -> MAP (exp, VAR)                    //Apply exp to each element in VAR
13 vardecl -> VAR := exp                    //round:=area / perimeter
14 decls -> DECLS (vardecl)$*$ SLCED        //Set of new attribute declarations
```

The virtual machine handles either real values or arrays of real values, which can be addressed in the declarative section of the query. Typechecking is performed during loading. Thus, during the object learning process, complex attributes can be added without changing the underlying implementation of the query engine. This ensures the postulated simple extension of feature information. Some aggregated sample attributes are:

```
1. convexity := (4 * area * 3.14176) /(pow((perimeter),2.0))
2. entropy   := -1 * FOREACH i IN hist() + (i/hist.size * log(i/hist.size) /log)
```

3.2. Query rules

Based on the expressions defined in the last section query rules are defined as first order logic (FOL) predicates in the following predicate grammar:

```
15 pred -> TRUE, FALSE
16 pred -> exp $R$ exp                      //R from { <, >, =, !=}
17 pred -> pred $L$ pred                    //L from { ||, &&, ! }
18 pred -> '(' pred ')'                     //Nested parenthesis
19 rule -> pred
20 rule -> rule ',' pred                    //Concatenation of predicates
21 rules -> (CONST rule)+                   //Enumerated list of rules
22 query -> decls TD rules DT BU rules UB : //Entire query
```

Formulas in FOL consist of predicates and expressions, where the free variables are the attributes of each region. Thus, a formula is evaluated with respect to a specific region. The following example illustrates a rule to extract homogeneous circular areas:

```
1. entropy := -1 * FOREACH i IN hist() + (i/hist.size * log(i/hist.size) /log)
2. (entropy < 0.1) && (dist(fds(2),(1,1) < 0.1)
```

In line 1, the attribute entropy is defined as the distribution of grey-values in the region. Line 2 is the actual query Q, which is fulfilled if two predicates become true. The first predicate tests whether the entropy is beneath a threshold of 0.1, which means a homogenous grey-value distribution. The circular shape property is described by the first two Fourier descriptors of the contour. Since they represent the second area moment of the enclosed region they capture its principal axis. Thus, a value of (1,1) represents the unit circle. By computing the

Euclidian distance with a threshold of 0.1, each region with a variance of 10% between its principal axis is classified as circles.

Those rules allow a heuristic formalization of region properties by fast construction of descriptive features. This is a data-mining approach to pattern recognition because descriptive features are not selected from an initial feature set by a redundancy analysis such as the principal component analysis (PCA). On the contrary, the features are defined heuristically by the observer. By aid of the query language, he can perform the data-mining cycle of hypothesis formulation, application, evaluation, and re-formulation in a fast and simple way. The cycle can be applied automatically to a training and test dataset as well as manually integrated into a visual feedback interface.

4. EXECUTING QUERIES

By now the definition of the static components i.e the syntax of the query language is described, which expressive power can actually be regarded as a subset of current structured query languages (SQL) implementations. But unlike SQL which operates on relations i.e. tables the hierarchicyl query language (HQL) operates on tree hierarchies. In this structure subgraphs corresponds to objects in causal multi-scale decompositions as identified in Section 2). To extract single regions with sought properties from those hierarchies HQL can be applied to the tabular representation of the region set just like SQL. But in combination with the tree structure which results from the inclusion paradigm the processing sequence of the query rules is used to model respective subgraphs.

4.1. Processing order

In the hierarchical tree structure, a top-down and a bottom-up sequence is used to refer to a subset of regions. Following those two possibilities of tree traversal, each node can be visited either as a root of a subtree or as one of its leafs. Following the inclusion paradigm, this means an object definition that is based on a single region which is subsequently specialized by claiming distinct sub regions. This guarantees a more discriminant strength against similar appearing regions, which represent other objects.

The top-down sequence follows the nodes from the root to the leafs and corresponds to the coarse-to-fine processing of visual perception. Objects are identified by a super structure which consists of sub structures with respect to the inclusion relation paradigm. If a HQL-rule is fulfilled in top-down traversal, the corresponding subtree is selected as the root for a subsequent search space. The evaluation strategy is a simple recursion depending on a formula f. Its principle functionality is indicated in the following code fragment:

```
01. top_down_evaluate(n)
02.  foreach s in sons(n)
03.    if f(s)
04.      mark_node_select(s)
05.    else
06.      top_down_evaluate(s)
```

The function `top_down_evaluate` gets the actual node as a parameter, which initially is the root of the search graph (line 1). Each son s of the current node is addressed in a loop (line 2) and for each s the formula $f(s)$ is evaluated. If $f(s)$ is true, the current son is marked as relevant (line 4) and the function ends. If $f(s)$ fails, the recursion is continued until there is a node which fulfills the formula or until no subsequent node is left (line 6). This algorithm shows the principle of hierarchical queries but it does not extract sub-trees. A multi layered query tree as indicated in Figure 1 requires an extension of the simple algorithm.

```
top_down_bottom_up_evaluate(n,rulenum,mode)
 foreach s in Sons(n)
   if(mode = TOP_DOWN)
     foreach formula r(rulenum)
```

```
        if r(rulenum, s))
          mark_node_td_select(s)
          rulenum++
    if(rulenum > TD_RULE_NUM)
      top_down_bottom_up_evaluate(s,rulenum,BOTTOM_UP)
    else
      top_down_bottom_up_evaluate(s,rulenum, TOP_DOWN)
  rulenum = 0
  if (mode = BOTTOM_UP)
    foreach formula r'
      if r'(s)
        mark_node_bu_select(s)
```

At first, the algorithm is extended by a processing mode which indicates a top-down or bottom-up order to reference root and leafs. At each top-down iteration the algorithm references a predicate from an enumerated list indicated by rulenum (pred grammar line 21). This addresses the actual layer in the respective query tree. This layer may consist of one or more comma separated rules, which represent different leafs at the current level (pred grammar line 20). Each of those rules is evaluated for each node of the subtree. If a rule is fulfilled the next rulenum is selected. If no other top-down rule layer exists the processing mode is switched to bottom-up processing, where again a special rule may select leafs for each entirely processed sub tree.

The algorithm extracts multi layered sub trees by following the top-down and bottom-up sequence on different levels. This is done by changing the rules application context with respect to the currently applied rule which is indicated by the rulenum value. The top-down, bottom-up strategy along with the context switch after each fulfilled predicate implements the greedy approach for region selection as implied in Section 2.2. This is based on the inclusion paradigm as derived from the nested sub-object observation. The algorithm also accepts incompletely processed sub trees as well as incompletely processed layers. Those predicates can be counted and indicate the quality of match.

5. EXPERIMENTS AND RESULTS

Two classes of experiments have been performed on a set of 10 sagittal magnetic resonance imaging (MRI) slices of a human head. At first, the extraction quality was evaluated and afterwards the extraction time.

5.1. Extraction quality

The transferability of object definitions to other applications has to be examined since defined queries must be applicable to series of images and extract the same objects. The sample task was to extract a given anatomic structure from a sagittal MRI.

Question In a sagittal MRI of a head, the brain should be extracted along with the cerebellum, the corpus callosum, and the mid-brain. Additionally, the mid-brain has to be examined for additional substructures (Fig. 2).

Execution By browsing a sample multi scale decomposition of a given MRI visually sensible regions were identified. Then, their descriptive features and respective limiting values were selected heuristically for each region. The visually sensible feature of the brain node were the first 40 Fourier descriptors. The cerebellum was detected via its convexity, entropy, and size, whereas the corpus callosum was detected via its first 15 Fourier descriptors. The mid-brain was detected via convexity and size while the substructures could simply be extracted via size in a bottom-up run since cerebellum and corpus callosum do not split up into several convex subregions.

Result All brains and substructures have been extracted in a second MRI with the given rule `BRAIN` (Fig. 2).

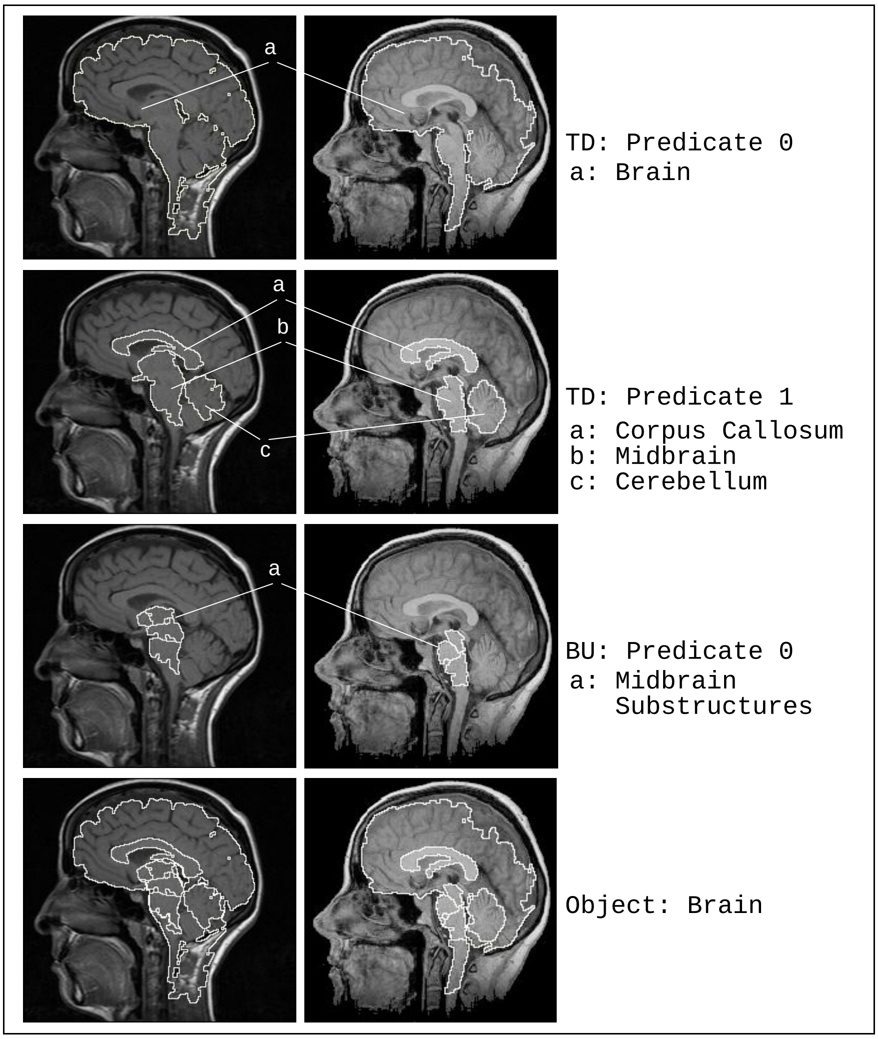

Figure 2. The application of the rule BRAIN extracts regions with different interpretations on three levels. The entire brain is the root object on level 0 in the top-down sequence. In level 1 the substrucutres were identified. The bottom-up application yielded the mid-brain sub-structures. The entire segmentation shows an overlay of the three levels.

```
DECLS
  conv    := (4 * area * 3.14176) /(pow((perimeter),2.0))
  entropy := -1 * FOREACH i IN hist() + (i/hist.size * log(i/hist.size) /log)
SLCED
TD
  0 dist(fds(40),[0.40000,0.06827,0.32890,0.04705,...]) < 0.05
  1 (conv > 0.7) && (entropy > 0.8) && (900 > size) (size > 800)  //Cerebellum,
  1 dist(fds(15),[0.10240,0.07909,0.05790,0.00966,...]) < 0.05    //Corpus Callosum,
  1 (0.55 > conv) && (conv > 0.5) && (1800 > size) (size > 1500)  //midbrain
DT
BU
  (0.4 > conv) && (conv > 0.55) && (600 > size) (size > 400) //midbrain substructure
UB
```

5.2. Computation Time

There are two questions concerning computational time. At first, the time to implement a HQL-rule and secondly the execution time. The rule BRAIN from Section 5.1 was designed manually by scanning the region features in a sample decomposition of one of the images. The observer browses the multi-scale hierarchy and points at those regions he finds relevant. Its feature values become parameters for the query rule which again is tested immediately on its precision. This cycle is repeated until the sought object is extracted. The time between the loading of the training image and the final storage of the rule was measured. The design of the rule took approximately 5 minutes.

The MRI images had a size of 256x256 pixels, 25 scales have been computed and the hierarchy consisted of approx. 2500 nodes and occupied 4 MB. The tests have been computed on an Intel Pentium III Mobile CPU with 1.0 GHz and and an IDE hard disk running Linux as the OS. The MRI hierarchies have been loaded in 1500 ms and rule evaluation took approximately 300 ms.

6. DISCUSSION AND OUTLOOK

In this paper, a novel approach to object extraction is introduced. Based on a multi scale decomposition, the visual perceivable information is transformed into a HARAG. In this graph the extraction of objects becomes subgraph matching. This is a NP-complete problem, but if the contextual information is used, the matching can be reduced to a query that is based on a domain specific language. This language along with its evaluation strategy was defined and tested with a sample application. The query design itself is highly heuristic and depends on the actual query content. Since a query is designed while testing the results, this corresponds to the feedback cycles during image retrieval. The extraction itself is computed fast since the object description is typically very small compared to the image HARAG. The results were encouraging but tests must be extended to a large database in real world conditions.

Counting and comparing the fullfilled predicates is not yet implemented in the extraction algorithm but can be added with no conceptual change. Yet adjacency analysis is not integrated in the current method but there exist appropriate approaches e.g. by Petrakis et.al. which can be adopted.[11] The query concept itself depends on a suitable and general hierarchical decomposition of an image which is based on a model of perception to provide visually comprehensible regions. Multi scale decomposition has been in the focus of research in the last decade but stable, causal, and self-dual approaches are hard to find.[18] This is an ongoing task. Because of the tree structure of a HARAG, a complete graph-matching is not required. Rules extract subtrees with respect to the inclusion paradigm of natural objects. In addition, the necessary heuristics are formulated directly by the observer and can easily adopted to different tasks. In this setting, it becomes obvious why fast rule generation must be integrated with fast rule evaluation. The parameterization cycle is the most repeated computation in the application because it must give an immediate feedback. During serial evaluation, the file input and output (IO) is the runtime bottleneck which can be improved by fast IO-techniques or pre-loading a large part of hierarchy files in memory.

An important extension of the language is the addition of adjacency constraints for the sought regions. The data organization and query can even be integrated into complex indexing schemes for feedback speed-up such as inverted files.[7]

REFERENCES

1. Haralick RM, Shapiro LG: Image segmentation techniques; Computer Vision, Graphics, and Image Processing 19: 100–132 1985.
2. Koenderink JJ: The structure of images; Biological Cybernetics 50: 363–370, 1984.
3. Ullman S: High Level Vision: Object Recognition and Visual Cognition; MIT Press, Cambridge, MA, 1996.
4. Sengupta K, Boyer KL: Organizing large structural model-bases; IEEE Transactions on Pattern Analysis and Machine Intelligence 17(4): 321–332 1995.
5. Zahn Q, Molenaar M, Templfi K: Hierarchical image object-based structural analysis toward urban land use classification using high-resolution imagery and airborne LIDAR data; Proceedings of the Remote Sensing and Data Fusion over Urban Areas, IEEE/ISPRS Joint Workshop 2001, pp. 35–39, 2001.
6. Lindberg R: Scale Space Theory; Kluwer Academic Publishers Dordrecht, 1994.
7. Djeraba C: Association and content-based retrieval; IEEE Transaction on Knowledge and Data Engineering 15(1): 118–135, 2003.
8. Wang JZ: Integrated region-based image retrieval; Kluwer Academic Publishers Boston, 2001.
9. Tagare HD, Carl Jaffe C, Duncan J; Medical image databases: A content-based retrieval approach; Journal of Applied Mathematics and Image Analysis 4: 184–198, 1997.
10. Barr S, Barnsley M: A region-based, graph-theoretic data model for the inference of second-order thematic information from remotely-sensed images; International Journal of Geographical Information Science 11(6): 555–576 1997.
11. Petrakis EGM: Design and evaluation of spatial similarity approaches for image retrieval; Image and Vision Computing 20: 59–76, 2002.
12. Chen JY, Bouman CA, Dalton JC: Hierarchical browsing and search of large image databases; Image Processing 9(3): 442–455, 2000.
13. Xu Y, Duygulu P, Saber E, Tekalp AM, Yarman-Vural FT: Object-based image labeling through learning-by-example and multi-level segmentation; Pattern Recognition 36(6): 1407–1423, 2003.
14. Braga Neto U, Goutsias J: Connectivity on complete lattices: New results; Computer Vision and Image Understanding 85: 22–53, 2002.
15. Haris K, Efstratiadis SN, Maglaveras N, Katsaggelos AK: Hybrid image segmentation using watersheds and fast region merging IEEE Transactions on Image Processing 7(12): 1684–1699, 1998.
16. Metzler V, Thies C, Aach T: A novel object-oriented approach to image analysis and retrieval; Procs. 5th IEEE Southwest Symposium on Image Analysis and Interpretation, pp. 14–18, 2002.
17. Thies C, Malik A, Keysers D, Kohnen M, Fischer B, Lehmann T: Hierarchical feature clustering for content-based retrieval in medical image databases Procs. SPIE 5032(1): 598-608, 2003.
18. Metzler V: Morphological Decomposition as Universal Methodology in Medical Image Processing (in German); Fortschrittsbericht 10/698, VDI–Verlag, Duesseldorf, Germany, 2002.
19. Sonka M, Hlavac V, Boyle R: Image Processing, Analysis and Machine Vision; Chapman & Hall, London, 1993.
20. Cootes TF, Taylor CJ, Cooper DH, Graham J: Training models of shape from sets of examples; Procs. British Machine Vision Conference, Springer Verlag, pp. 266–275, 1992.
21. Adams NJ, Williams CKI: Dynamic trees for image modeling; Image and Vision Computing 21: 865–877, 2003.