Fast and Memory Efficient Feature Detection using Multiresolution Probabilistic Boosting Trees

Florian Schulze
VRVis Center for Virtual Reality and Visualization Research
fschulze@vrvis.at

David Major
VRVis Center for Virtual Reality and Visualization Research
dmajor@vrvis.at

Katja Bühler
VRVis Center for Virtual Reality and Visualization Research
buehler@vrvis.at

Abstract

This paper presents a highly optimized algorithm for fast feature detection in 3D volumes. Rapid detection of structures and landmarks in medical 3D image data is a key component for many medical applications. To obtain a fast and memory efficient classifier, we introduce probabilistic boosting trees (PBT) with partial cascading and classifier sorting. The extended PBT is integrated into a multiresolution scheme, in order to improve performance and works on block cache data structure which optimizes the memory footprint. We tested our framework on real world clinical datasets and showed that classical PBT can be significantly speeded up even in an environment with limited memory resources using the proposed optimizations.

Keywords: Feature Detection, Machine Learning, Decision Trees

1 INTRODUCTION

In the past years various methods for automatic processing and understanding of medical 3D image data have been developed. One important building block is the automatic detection of anatomical landmarks. Detection of these features stands often at the beginning of the processing pipeline: it transforms the dense volume representation into a sparse set of possible landmark locations, allowing a significant acceleration of subsequent high level segmentation methods.

However, making the transition from pure research algorithms which focus often solely on detection performance to real world radiology applications brings a number of additional requirements into consideration. The algorithm has to be able to deal with possibly limited technical resources - not all workstations in a hospital might be equipped with the newest hardware, and the algorithm shall run in the context of radiology workstation software which already occupies resources. Excellent time performance is required because automatic algorithms often substitute manual workflows while the result must be authorized and/or adjusted by the radiologist. In this case an automatic algorithm will only be used if the execution time of the algorithm is considerably shorter than the manual approach would be.

This work presents a highly optimized general purpose feature detection framework for the effective reduction of possible feature candidate positions in 3D image data as preprocessing step for more expensive object detection methods. Referring to the clinical application context, we designed our method according to the following requirements:

1. Time Performance: The result must be calculated in relatively short time (e.g. within seconds) in order to be usable in a clinical environment.

2. Memory Performance: The algorithm must also execute on standard PCs with limited technical resources.

Thus, the focus of the proposed algorithm and its implementation is on a small adaptable memory footprint while retaining as much execution speed as possible.

2 RELATED WORK

Object recognition, and local feature detection as a subdiscipline of it, are since many years core topics of computer vision research.

Point based methods beginning with the Harris corner detector [HS88] try to automatically extract points of interest from an image. Exact control of which points are extracted is not supported, therefore recognition of complex structures/areas is done by combining sets of feature points. The most prominent point detector is the SIFT algorithm [Low99] which overcomes the limitations of previous solutions by being scale, rotation and perspective invariant. However, translating SIFT, which is aimed for 2D images, to 3D volumes suffers from dramatic performance problems. Niemeijer et al. report in [NGL$^+$09] that SIFT feature extraction on a

$200 \times 200 \times 1024$ volume downscaled by 50% takes 10 minutes to compute.

Machine learning based approaches use a (learned) classifier to decide if a specific region of an image belongs to an object. A prominent example for this class of algorithms is the method for real time face detection presented by Viola and Jones [VJ01] that uses a cascade of boosted weak classifiers. A more general approach has been proposed by Tu et al. [Tu05] by introducing Probabilistic Boosting Trees (PBT). PBTs are decision trees which use boosted learners as classifiers in each tree node. Violas boosted classifier cascades are a special case of a PBT. An alternative to PBTs is the popular d-tree forests method [MDUA07] which produces higher detection rates, but with the drawback of much higher execution costs [LK08].

PBTs have been successfully applied on tissue classification on medical images: Militzer and Vega-Higura [MV09] use PBT for bone removal in CT angiography. The volume is first split into segments using the watershed algorithm, then each segment is classified with PBT.

Fast preselection of feature candidates for more expensive high level methods is the topic of the paper of Langer and Kuhnert [LK08]. They integrate classical decision trees with simple color based features and a multiresolution scheme for candidate computation for the expensive SIFT feature detection.

The problem of the large memory footprint of volume data is often discussed in context of volume rendering. LaMar et al. [LHJ99] use an octree structure with blocks containing different resolutions, where only the needed subvolume is downloaded to graphics hardware. However, the whole volume data still has to fit into main memory. This has been improved by Guthe et al. [GWGS02] who proposed to hold the data 30:1 wavelet compressed in memory and extract needed data on demand block-wise and cache the data as long as possible.

The purpose of our feature detection method is similar to that of Langer and Kuhnert [LK08] since we also aim to reduce the list of possible candidate position as much as possible for later more expensive methods. Langer and Kuhnert tailored their algorithm especially for pre-filtering for SIFT feature computation. In difference to them we decided to use the more general PBT [Tu05]. This has several advantages: first, it is independent from SIFT features and easily adaptable to any kind of landmark/structure. Second, decision tree methods can capture large image variabilities while only need to execute $\log n$ weak classifiers. Third, they are robust against over-fitting unlike classic decision algorithms.

Our contribution. To satisfy the high performance requirements to the algorithm in a clinical environment, we extend the original PBT by integrating cascading tree nodes into normal tree building and introduce the concept of classifier sorting (Section 3.1). Both result in higher execution speed of the classifier. A second performance optimization is achieved by integrating the PBT into a multiresolution classification scheme (Section 3.2). An effective postprocessing step is introduced that applies particle filters to compute probability maps for candidate features for outlier detection (Section 3.3). The memory footprint of our feature detection framework is optimized by the introduction of a multiresolution, multi-derivative block cache data structure (Section 4). The performance of our method has been evaluated on a real world clinical usecase (Section 5).

3 ALGORITHM

In the following we explain in detail the classifier and our extensions on it (Section 3.1), the multiresolution feature detection framework (Section 3.2) and the postprocessing step based on candidate probability (Section 3.3).

3.1 Probabilistic Boosting Tree with Partial Cascading and Classifier Sorting

Probabilistic Boosting Trees. A PBT [Tu05] is a special kind of decision tree which holds at each tree node a boosted classifier. PBTs are trained top down. Based on a set of positive and negative samples a boosted classifier with a limited number of weak classifiers is trained for each tree node. On each recursion level the sample set is split using the generated classifier and the new subsets are used to train positive and negative child branches. Although multi-class classifiers are possible, we limited our implementation to the simple two-class model.

Classical Cascading. If the boosted classifier in each tree node is trained in a way that it does not produce false negative results, the resulting decision tree consists of positive child nodes only. Traversing this tree has only one sequential path and degenerates to the cascade of boosted classifiers of Viola and Jones [VJ01] (see figure 1 left). Cascading improves execution speed. It allows the classifier to early terminate and reduces

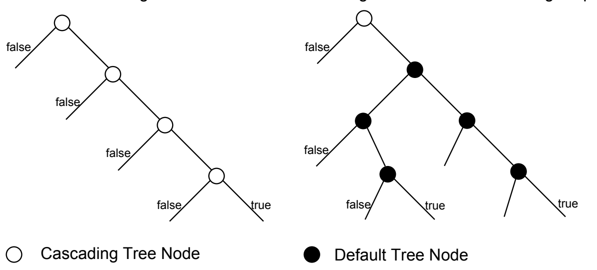

Figure 1: Probabilistic Boosting Tree. Left, tree with cascading nodes only. Right, one cascading node at the tree root followed by a default PBT.

in this way the number of classification tests, but it reduces also the flexibility of the original PBT to capture a high variability of features.

PBT with Partial Cascading. We observed that a high number of samples can be classified as false by executing only one boosted classifier (see section 5.2). This allows to combine the speed-up of cascading with the flexibilty of the PBT by placing one cascaded classifier in front of the PBT: Our tree model contains one cascading node at the root level. A negative outcome stops the classification immediately, a positive outcome is further processed using the full PBT (figure 1 right).

Classifier Sorting. We also observed that a high amount of samples can be early terminated with a cheap and fast performing classifier (see section 5.2) and that it is advantageous to use expensive classifiers only in places which are executed less often. In our model the most visited place is the cascading node at the root of the tree which can discard a large amount of samples as false. The rest of the tree is visited less frequently. Hence, we sort the expensive classifiers into the later tree nodes while the first node can only use fast executing classifiers.

Image Features. The classifier decides on a per voxel basis if the current voxel belongs to the searched structure or not. Since PBT is a so called ensemble classifier, basically every possible classification method can be integrated. However, the selection of image features has influence on detection performance and execution speed.

In the current work we integrated classifiers which make decisions based on five different image features.

1. Haar-like features with different patterns and sizes.
2. Image intensity
3. Gradients and principal curvatures
4. Region histograms based on image intensity and derivatives with different sampling resolutions and sizes.
5. Structure tensors

Haar-like features and image intensities are the features with the lowest computational costs and are therefore used for building the cascaded root. Gradients need to be computed by filtering as well as principal curvatures which need an additional Hessian analysis step. Region histogram classification multiplies the cost by the number of samples. Structure Tensors require the convolution of the gradient image with a Gaussian kernel and subsequent eigenanalysis of the structure tensor matrix. These three types of classifiers are exclusively used for the non-cascaded part of our PBT.

The chosen weak classifiers are scale variant which is adequate for our application scenario because we expect anatomical structures to have a specific size (small variations in size should be accepted anyway, larger variations because of age or gender can be covered with different detectors and pre-classification based on patient background data).

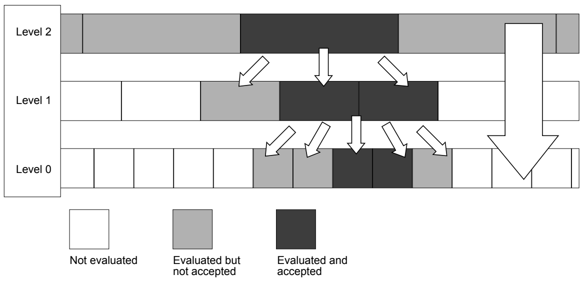

Figure 2: Multiresolution Algorithm

3.2 Multiresolution Feature Detection

The PBT with Partial Cascading is embedded into a multiresolution scheme based on a power of two Gaussian image pyramid [AAB+84] to further reduce the number of voxels to be processed.

A separate classifier C_i is trained for each resolution level. Multiresolution classification starts at the lowest resolution level n by applying classificator C_n on image I_n. Classification results in a set positively marked voxels $(p_0^{+,n}, ..., p_m^{+,n})$. These voxels are propagated into the next higher resolution level $n-1$ where each positive lower resolution voxel marks the voxels within the corresponding filter kernel in level $n-1$ as candidates. Classification of the current level is only computed on the remaining candidate voxel. The propagation is repeated until the original resolution (level 0) is reached. Figure 2 depicts the algorithm with a 1D example. Note that most of the high resolution voxels do not need to be checked using this scheme.

In the case of overlapping kernels some higher resolution voxels have two or more parent voxels and it can happen that a voxel is marked as positive and negative. In this case the positive mark is kept. This leads to a slight over-segmentation, but on the other hand the effect of false negative samples might be reduced, which is a wanted effect.

3.3 Filtering of Results Using Fast Probability Computation

The direct result of our feature detection algorithm is a bit mask of candidates which still might contain false positives. One method to reduce the number of false positives is to assign a probability to each candidate that reflects the confidence in its classification. The resulting probability map can then be further processed by thresholding which effectively removes outliers and/or non-maximum-suppression which only leaves the candidates which are at the center of the expected shape.

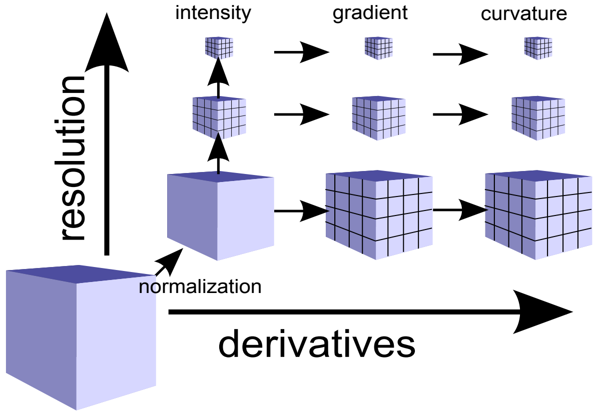

Figure 3: Datastructure: Only the base intensity volume is kept completely in memory. All other data, derivative and lower resolution volumes are computed block wise on demand.

Probabilistic boosting trees can deliver such a probabilistic classification. Drawback of this straight forward approach is the low time performance.

We observed that the result of feature detection form clusters at the feature location resembling already the searched structure (e.g. the intervertebral discs in figure 9). Thus, we propose to assign probabilities to candidates by comparing the shape of its surrounding cluster with the searched shape.

A fast option to compute this are shape particle filters which are applied in our framework. The likelihood that a candidate belongs to the searched structure is computed by applying a shape approximating the structure of interest around each candidate and by measuring the ratio of overlap of neighborhood and shape.

4 IMPLEMENTATION

4.1 Data Preprocessing

The spatial resolution of medical 3D images in a clinical environment is generally highly anisotropic. Especially the slice distances show high variability from modality to modality, from scanner to scanner depending on the used imaging protocol. The scale variant nature of the image features described in section 3.1 requires the same spatial resolution of all images to be processed.

Thus, training data as well as unseen data is preprocessed by resampling the original volume data to an isotropic voxel size that is selected based on the targeted anatomical landmark. The current implementation uses bilinear interpolation for resampling.

The resampled volume (in the following denoted as "base volume") is the basis for all following computations and the original data can be discarded at this point.

4.2 Data Management and Derivative Computation

The data management component is responsible for efficiently providing the necessary data to compute the requested weak classifiers on all resolution levels while keeping the memory footprint small and flexible.

The supported weak classifiers require intensity, gradient and principal curvature data for all positively marked voxel positions on the different levels of resolution. It is obvious that the performance of the weak classifiers decides on the performance of the whole PBT.

It is well known that filtering volume data with separated filters for derivative computation is much faster than applying a three dimensional filters per voxel individually. We currently use a $3 \times 3 \times 3$ Sobel for gradient computation, which can be replaced by any other appropriate separable filter. However, applying a separable filter for derivative computation requires to keep the whole filtered volume in memory, which might be problematic having our initial requirements in mind.

To overcome this limitation and to make the memory footprint manageable also in an environment with limited resources, we introduce a cached block structure (see Figure 3). The intensity base volume is entirely located in memory. Lower resolution volumes, gradients, structure tensors and principal curvature are organized into smaller blocks that are only computed on request. After computation, block data remains cached in memory. If the memory for allocation of new blocks gets low, the cache is partially cleaned by removing data which was accessed the longest time ago.

For fast computation of Haar-like features an additional data structure, an integral volume, is needed. This data is currently computed as a whole and kept in memory. This is due to the more complicated generation method of this data which makes it hard to compute the value block-wise on demand.

4.3 Optimized Classifier Execution

Generally each voxel can be classified individually by executing the whole boosting tree starting from the lowest resolution. However, having in mind that one voxel in a lower resolution volume has influence on a number of voxels in the higher resolution and that the data is arranged in a cached block structure, it is worth to consider a optimal execution order.

Detection of features on the whole volume or of a sub volume follows two strategies. First, feature detection is done in resolution level order. This means that the PBT for one level is executed on the whole region of interest and then all positive classified voxels are propagated to the next higher level.

Second, all per level classification is performed block wise. In this way only a small number of data blocks must be in cache. Any other execution order (for example line wise) would cause a lot of cache misses and would likely lead to often re-computation of block data. If multiple classifiers must be applied on the same volume all classifiers are executed on each block sequentially. After the first classifier is executed the block cache remains in (partially) filled state. Data which is already cached must not be computed if the next classifier tries to access this data. Parallelization is implemented using a worker thread-pool. Classification of one block is fed into a job queue which distributes the work to the worker threads.

5 EXPERIMENTS

Our multiresolution PBT framework was tested in a real world scenario as preprocessing part for a semi-automatic annotation algorithm for the vertebral column. The task was to preselect appropriate candidates for the location of the intervertebral discs and the spinal canal.

For the intervertebral discs, three different detectors were trained to cover the different appearance of lumbar, thoracic, and cervical disks. The spinal canal could be detected by using only one detector.

5.1 Setup and Training

The algorithm has been trained and evaluated on 19 CT datasets (13 for training 6 for evaluation only) containing different parts of the vertebral column. The datasets have up to 1112 axial slices with a slice resolution of 512×512 and a slice distance between 0.62 *mm* and 3.0 *mm*. Some of the data contains pathologies (broken vertebrae, collapsed disc, scoliotic spines) as well as one cervical dataset from a child.

Experiments have shown that the thinnest intervertebral discs in the cervical section can still be distinguished if the slice distance is at least 1.5 *mm*. We therefore fixed the base volume voxel scale for this experiment as 1.5 *mm* isotropic and the datasets were resampled accordingly.

In all datasets position and location of the intervertebral discs and the spinal column have been manually labeled. Based on the given annotation, positive samples have been generated randomly inside the intervertebral disc and the spinal column. Negative samples have been generated randomly all over the volume with the constraint to have a minimal distance to positive samples of 10 *mm*.

5.2 Performance Evaluation

Time performance of the algorithm has been assessed based on a set of eleven CT volumes (six evaluation and five training datasets). The properties of the data, its original and normalized size is listed in table 1. The classifier is trained using one cascading step and allow only intensity and Haar-like features in the cascade node. Influence of the different optimizations is measured against this default. Detection performance was measured based on 8 datasets containing the 6 evaluation datasets.

Volume	original size	normalized size
1	$512 \times 512 \times 202$	$106 \times 106 \times 134$
2	$512 \times 512 \times 163$	$113 \times 113 \times 108$
3	$512 \times 512 \times 361$	$144 \times 144 \times 168$
4	$512 \times 512 \times 222$	$89 \times 89 \times 148$
5	$512 \times 512 \times 249$	$170 \times 170 \times 166$
6	$512 \times 512 \times 152$	$91 \times 91 \times 101$
7	$512 \times 512 \times 277$	$245 \times 245 \times 184$
8	$512 \times 512 \times 260$	$244 \times 244 \times 179$
9	$512 \times 512 \times 1112$	$274 \times 274 \times 370$
10	$512 \times 512 \times 228$	$176 \times 176 \times 228$
11	$512 \times 512 \times 945$	$244 \times 244 \times 630$

Table 1: Properties of volumes for performance evaluation.

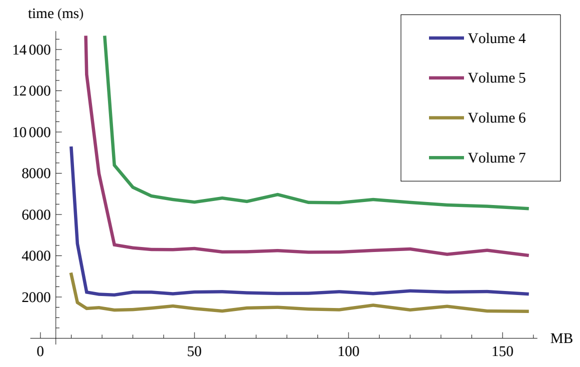

Figure 4: Memory Limits

Limited Memory The data structure is designed to cope with limited resources. However, reaching the bounds of memory provokes clearance of cache blocks that might have to be recomputed at a later stage of the algorithm. Figure 4 illustrates the time performance over different cache memory bounds for datasets $4-7$ and show a clear threshold ($\sim$ 25 MB) for all four datasets where the performance/memory ratio changes dramatically. This memory limit is slightly different for each dataset and depends on the dataset size. If the available memory falls below that threshold computation time rises heavily whereas performance remains stable if enough memory is available. The threshold marks the point where data blocks need to be frequently recomputed. As long as enough memory is available deletion of block data from the cache and occasional recomputation has almost no influence on performance.

Multithreading. We tested the multithreading performance of our algorithm on an Intel quad core CPU with 2.4 Ghz and hyperthreading. Figure 5 plots the computation speed over the number of threads again on datasets $4-7$. Time drops until 4 threads are used. For more threads no significant speed-up (but also no significant slowdown) can be monitored.

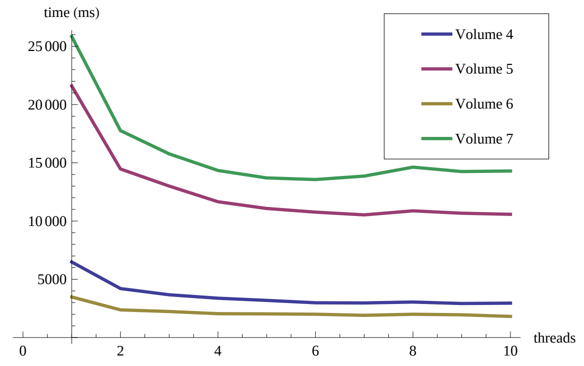

Figure 5: Plot computation time against number of threads. Tested on a quad-core with Hyperthreading.

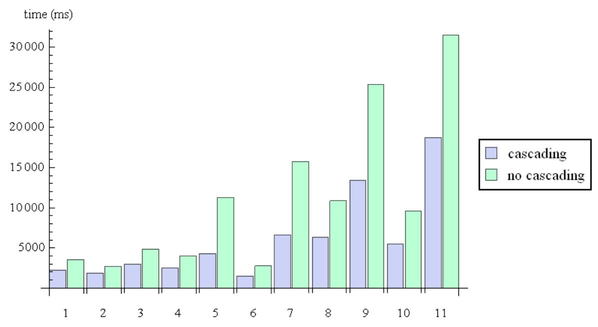

Figure 6: Detection speed comparison between PBT with (blue) and without (green) cascading on eleven different datasets.

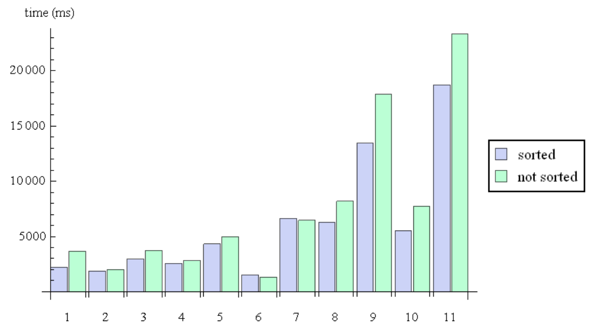

Figure 7: Detection speed comparison between PBT with (blue) and without (green) classifier sorting on eleven different datasets.

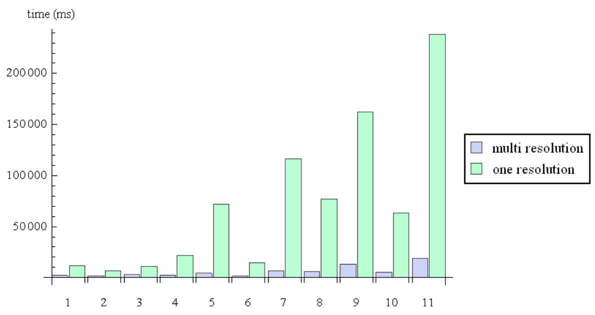

Figure 8: Detection speed comparison between multiresolution vs. one resolution.

The scaling with the number of threads below four is not linear. This is caused by the current locking strategy that prohibits accessing one block if it is currently computed by another thread. This situation mainly occurs if 2nd derivatives have to be computed that require accessing also neighboring first derivative blocks. If another thread is classifying one of these neighboring blocks at the same time it has to wait until the lock is released. This kind of collision happens more frequently as more threads are used. We expect therefore a logarithmic scaling of time performance with the number of cores as long as the locking behavior is not improved.

Cascading Speed-up. The impact of cascading on detection speed has been measured by comparing the time performance of our default detectors with detectors which are trained without including a cascading step. The result is plotted in figure 6. Over eleven datasets we measured a speed-up of $1.45 - 2.67$ for detectors including a cascading step.

Classifier Sorting Speed-up. The impact of classifier sorting is plotted in figure 7. We compare the time performance of our default detector including cascading and sorting with detectors which are allowed to use all classifiers in the cascading node. Classifier sorting results in a speed-up up to 1.65 for detectors which use sorting.

Multiresolution Speed-up. To measure the impact of multiresolution feature detection we compared detectors using three levels of resolution against detectors using only one level. The results are plotted in figure 8. The measured speed-up ranges between 3.44 and 17.51.

Detection Performance. Two feature detection results are depicted in Figure 9. The first row shows the detection of intervertebral discs in the lumbar section of the spine, the second row the detection of the spinal canal on a whole spine. The detection progress from lowest to highest resolution level is depicted from left to right.

The images illustrate well the effectiveness of the multiresolution scheme since already at the lowest resolution level the major part of the volume is excluded from higher resolution analysis.

The selected voxels (blue) reproduce the shape of the searched anatomical parts to a large extend. However outliers can be observed, for example inside the vertebral body (first row) or at the ventral side of the ribcage (second row). Moreover missing features can be observed as well (first row, ventral side of the topmost disc).

This observation is also reflected in recall and 1-precision plots (figure 10). Recall denotes the ratio between selected voxels within the ground truth and all possible ground truth voxels. 1-precision stays for selected voxels outside the ground truth divided by all the voxels which were selected by the feature detection (also the falsely selected ones). The evaluated data involves healthy spines (1, 2, 3, 6, 7 in figure 10) and

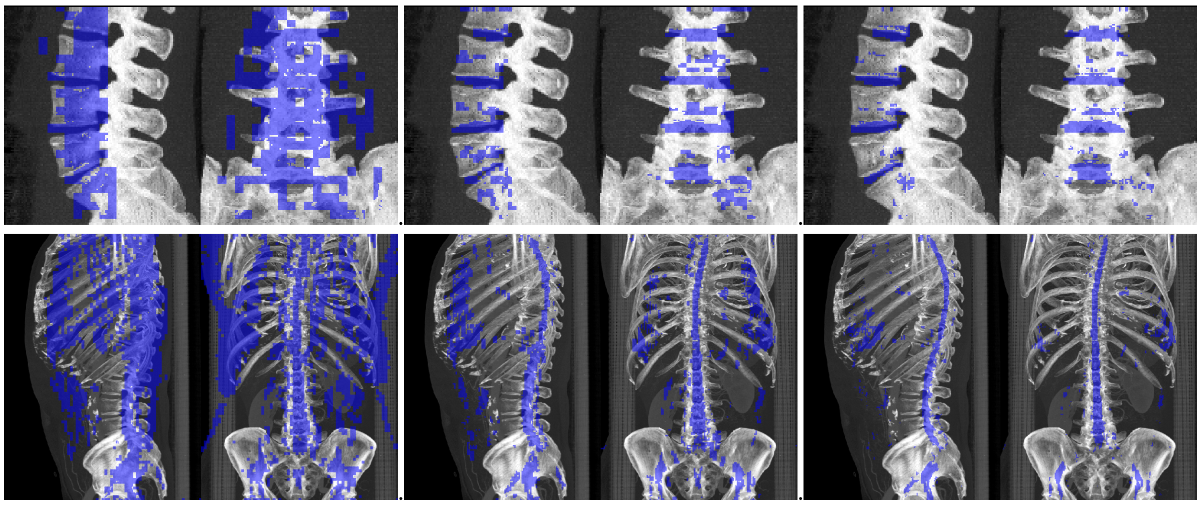
Figure 9: Coronal and sagittal images of detection results for the intervertebral disc (first row) and the spinal column (second row). Three levels of resolution document the detection process, lowest resolution left to highest resolution right.

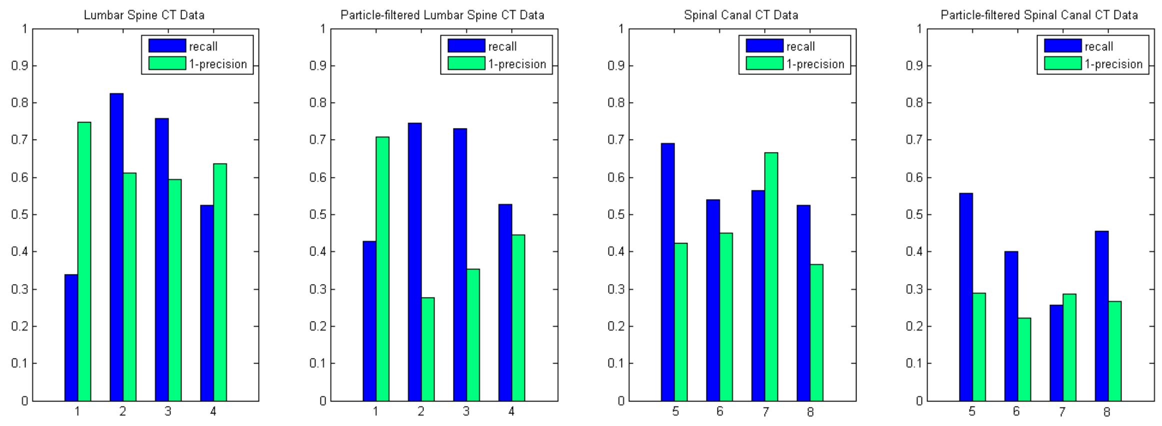

Figure 10: Recall and 1-Precision Plots

spines with diseases like scoliosis and broken vertebrae (4, 5, 8 in figure 10).

The first and the third graph show results after feature detection without any postprocessing where the high recall rates give information about good detection results of structures of interest (discs and spinal canal). However, besides the high recall rates there are also high rates of 1-precisions because of the occurrence of outliers (i.e. spongy bone within vertebrae with similar features to discs). The high 1-precision rates can be reduced by postprocessing steps such as particle filters which are visible in the second and fourth graph of figure 10. The recall rates remain fairly the same, minor reductions are due to moving towards the center voxels of the discs by particle filtering.

An example for postprocessing of the resulting feature mask is depicted in figure 11. First probabilities are computed by applying a box shape particle filter with the dimensions $9 \times 9 \times 60mm^3$. The box approximates elongated shape of the spinal canal. Second, the feature

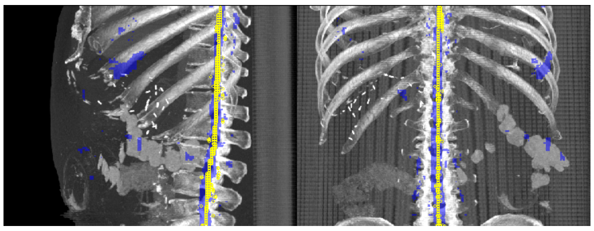
Figure 11: Feature mask (blue) post processed with particle filtering, non-maximum suppression and thresholding (yellow).

points are reduced by non-maximum suppression of the probabilities. Third, outliers are removed by thresholding the probability. The threshold is defined at $t = 0.15$.

6 DISCUSSION AND CONCLUSION

We have presented a method for time and memory efficient feature detection on medical 3D volume data. The goals and requirements formulated at the end of

Section 1 have been reached by selecting a classification based approach based on a Probabilistic Boosting Tree classifier. The classification method was improved by combining the decision tree with one cascading step and the introduction of classifier sorting. This classifier was embedded into a multiresolution framework. We could show that all optimizations together result in a huge time performance gain with an approximated speed-up factor of 20.

Multithread performance was measured to scale non linear (almost logarithmic) which is due to internal data locking. The speed-up is for state of the art quad core CPUs still significant. But to benefit from more parallelism, improvements have to be done in this section. However it is likely that more sophisticated access patterns and locking schemes can help to overcome this problem.

The behavior of the block cache data structure was evaluated in section 5.2. It is noticeable that even larger datasets require only $\sim 25MB$ for the block cache to run almost unhindered. However even under circumstances where less memory is available the algorithm will just perform slower.

Detection rate of this feature detector is not as good as it could be. We believe that other image features and filtering techniques, a finer bases scale and also a different kind of classifier could result in better detection performance. However, trading detection performance against execution speed was a conscious design decision. The results are good enough to use this method to reduce the search space for more specialized and more expensive image processing methods.

REFERENCES

[AAB+84] E.H. Adelson, C.H. Anderson, J.R. Bergen, P.J. Burt, and J.M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[GWGS02] S. Guthe, M. Wand, J. Gonser, and W. Straßer. Interactive rendering of large volume data sets. In *Visualization, 2002. VIS 2002. IEEE*, pages 53–60. IEEE, 2002.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[LHJ99] E. LaMar, B. Hamann, and K.I. Joy. Multiresolution techniques for interactive texture-based volume visualization. In *Proceedings of the conference on Visualization'99: celebrating ten years*, pages 355–361. IEEE Computer Society Press, 1999.

[LK08] M. Langer and K.-D. Kuhnert. A new hierarchical approach in robust real-time image feature detection and matching. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008.

[Low99] D.G. Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. Published by the IEEE Computer Society, 1999.

[MDUA07] Christophe Marsala, Marcin Detyniecki, Nicolas Usunier, and Massih-Reza Amini. High-level feature detection with forests of fuzzy decision trees combined with the rankboost algorithm. Technical report, Université Pierre et Marie Curie-Paris, 2007.

[MV09] A. Militzer and F. Vega-Higuera. Probabilistic boosting trees for automatic bone removal from CT angiography images. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7259 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, February 2009.

[NGL+09] M. Niemeijer, M.K. Garvin, K. Lee, B. van Ginneken, M.D. Abràmoff, and M. Sonka. Registration of 3D spectral OCT volumes using 3D SIFT feature point matching. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7259, page 51, 2009.

[Tu05] Z. Tu. Probabilistic boosting-tree: learning discriminative models for classification, recognition, and clustering. In *ICCV*, volume 2, pages 1589 –1596, 2005.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, volume 1, pages I–511 – I–518, 2001.