Vis Comput (2011) 27: 129–139
DOI 10.1007/s00371-010-0539-6

ORIGINAL ARTICLE

YaDiV—an open platform for 3D visualization and 3D segmentation of medical data

Karl-Ingo Friese · Philipp Blanke · Franz-Erich Wolter

Published online: 25 November 2010

Abstract In this work, we present the concept, design and implementation of a new software to visualize and segment 3-dimensional medical data. The main goal was to create a platform that would allow trying out new approaches and ideas while staying independent from hardware and operating system, being especially useful for interdisciplinary research groups. A special focus will be given on fast and interactive volume visualization, and a survey on the use of Virtual Reality (VR) and especially haptic/force feedback in medical applications will be provided.

The software will be published as Open Source and therefore be available as a rapid prototyping platform for own ideas and plugins for all members of the scientific community.

Keywords Medical visualization · Medical segmentation · Virtual reality · Haptics

1 Introduction

In recent years, 3D scans of parts of the human body became a standard procedure in the clinical workflow. Tomography techniques like Computer Tomography (CT), Magnetic Resonance Imaging (MRI) or Positron Emission Tomography (PET) produce data in the form of a regular grid, often referred to as voxel data. While direct or indirect volume visualization techniques are already valuable tools in diagnosis and research, their output can be improved by the use of additional segment information created with (semi-) automated segmentation techniques.

The sheer amount of scans and the resulting new tasks in modern electronic medicine require new ways of data interaction and manipulation. VR components like stereographic visualization and haptic interaction allow a more natural and intuitive access to the relevant information than conventional user interfaces. The new challenges require interdisciplinary groups of researchers, i.e., surgeons, radiologists, mathematicians, programmers, etc. An open software for medical visualization and segmentation being independent from hardware as well as operating system (OS) will significantly help to reduce the administrative efforts.

This paper addresses scientists from computer graphics as well as medicine and touches different fields of active research. Therefore, the attempt was made to write this paper in a style that the different groups of readers can understand all parts, by giving at least a short introduction to the basic principles and concepts of each chapter.

2 Goals

The special requirements on a software for the tasks described above can be summarized in five points:

1. **(Open Source)** To develop completely new approaches, access to the complete source code is required.
2. **(Platform Independence)** To reduce administrative efforts in large interdisciplinary research teams, a platform independent software is mandatory.

K.-I. Friese (✉) · P. Blanke · F.-E. Wolter
Leibniz Universität Hannover, Welfengarten 1, Hannover, Germany
e-mail: kif@welfenlab.de

P. Blanke
e-mail: blanke@welfenlab.de

F.-E. Wolter
e-mail: few@welfenlab.de

3. **(Fast)** While high quality visualization looks impressive, in most cases a convenient, fast and interactive visual access is needed, especially by medical users.
4. **(Extendable)** Many applications in medical research are geared towards specific clinical problems, e.g., preoperative planning or cancer diagnosis. The software has to be adaptable to support such use cases.
5. **(Easy to Use)** As the software should not only be a rapid prototyping tool for programmers in the field of medical visualization/segmentation, but also be tested and used in clinical research, the interface has to be intuitive and efficient on behalf of clinical requirements.

The last point also implies that the software has to be sufficiently stable/mature (no proof-of-concept solution). At the beginning of the development of YaDiV, no (free) software fulfilled these requirements.

3 Overview of free DICOM software

Today, many free programs for interacting with DICOM data are available, from simple tools (like anonymizers or converters), plain 2D- or 3D-viewers to complex software systems that allow interactive or high-end visualization, segmentation support as well as data analysis. Since YaDiV belongs to the last category, a short overview over currently active projects in this field should be given.

Almost all software systems are build using the Insight Segmentation and Registration Toolkit (ITK) and Visualization Toolkit (VTK). ITK is developed since 1999, being an Open Source Toolkit for Registration and Segmentation, containing data structures as well as algorithms. VTK was originally part of the book "The Visualization Toolkit—An Object-Oriented Approach to 3D Graphics" by Will Schroeder, Ken Martin and Bill Lorensen [10], which became one of the standard books in education. VTK contains methods from the fields of 3D Graphics, 3D Modeling, Image Processing, Volume Rendering and Scientific Visualization and can visualize polygonal as well as discrete data. Both VTK and ITK are written in C++.

3D Slicer

3D Slicer (current version 3.6) is an Open Source project (BSD License) for Visualization and Image Analysis. The program is in development since 1998, the first stable version was released in 2008. The module based software system allows integrating external plugins which can make use of internal modules for visualization, segmentation, registration and filters. 3D Slicer is written in C++ and available for several OS, such as Windows, Linux and Mac OS X. Slicer makes strong use of standard libraries such as ITK/VTK or the NA-MIC Kit for the modular architecture.

OsiriX

OsiriX is one of the oldest and most successful open source tools for the visualization of multimodal medical images. Similar to 3D Slicer, it is built upon ITK/VTK, but is written in Objective-C and is only executable with Mac OS X. The 32 bit version of OsiriX is developed under the GNU Public Licence (LGPL), the 64 bit version is commercial. OsiriX has a very user friendly GUI and contains a large number of modules; however, due to the fact that it is developed only for one OS, its use, especially in larger interdisciplinary teams, is restricted.

MeVisLab

As opposed to the two other programs, MeVisLab itself is not Open Source, but there exists a Software Development Kit (SDK) that allows developing own modules for 3D image processing and analysis using the innovative build-in graphical editor. MeVisLab also uses ITK/VTK and comes with four different license models (commercial, non-commercial, evaluation and unregistered). The well designed module editor makes MeVisLab a good choice for rapid prototyping without the necessity of mature programming skills.

4 YaDiV

YaDiV (Yet Another DICOM Viewer) was developed by Karl-Ingo Friese [4], in cooperation with medical researchers from the Hannover Medical School (MHH). The complete software was developed from scratch and is not built on ITK/VTK. The development started in 2006 with a strong focus on 3D visualization and segmentation. At that time, only a few Open Source DICOM systems were available, and those did not have the desired functionalities.

4.1 Concepts

Many algorithms working on (sometimes very large) medical volume data sets exhibit very long run times. Since they may produce (false) results due a wrong choice of parameters, these may have to be corrected by the user either by selecting other parameters and re-running the computation or correcting the results manually. Therefore, one of the core concepts of YaDiV is the heavy use of multi-threading for every time-consuming method, thus not blocking the graphical user interface (GUI). The user can observe the process with an animated visualization of the intermediate results,

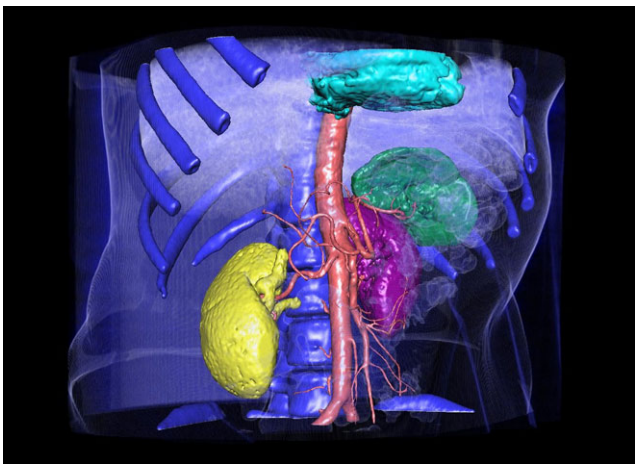

Fig. 1 Interactive visualization (segment and volume data) with YaDiV

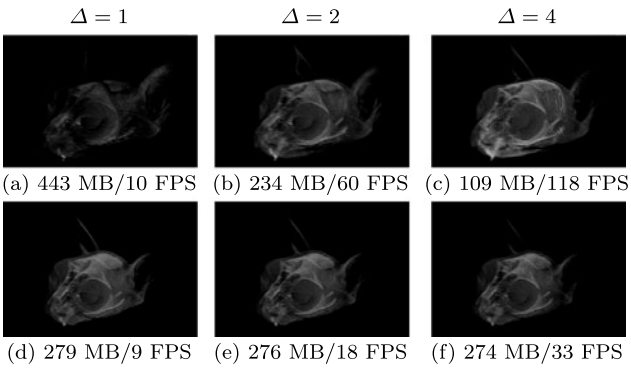

Fig. 2 Influence of projection quad number in 2D/3D texturing, measured on a notebook graphic card (NVIDIA GeForce 9600M GT)

e.g., a segmentation method, and manually abort them at every stage.

The software is designed to be an open platform that allows the rapid development of new modules and applications in medical data processing. It contains a large set of standard operations and integrates a plugin concept to allow different scientists to develop their own (usually more specific) modules.

4.2 Visualization

"*As a tool for applying computers to science, visualization offers a way to see the unseen. As a technology, Visualization in Scientific Computing promises radical improvements in the human/computer interface and may make human-in-the-loop problems approachable*." [6]

We differentiate between two different visualization strategies: *interactive visualization* is designed for very short response times, e.g., a fast validation of a segmentation result, while *high quality visualization* focuses only on the quality of the resulting image. Furthermore, we distinguish between methods that visualize only the original topographical data (raw data), those that use already identified structure/segment information and methods that use both kinds of information to generate a visual impression.

The visualization modules of YaDiV include fast and interactive volume visualization using shaders and 2D or 3D textures if the hardware supports this, as well as multiple forms of segment visualization, e.g., surface representation using an interactive self-refining marching cube with a modified lookup table. Figure 1 demonstrates interactive volume and segment visualization with YaDiV. Also included is a high quality ray casting module, using normal transfer functions as well as segment information and optional artistic lighting models.

2D/3D texturing

As their name already suggests, texturing methods use the 2D/3D texture abilities of modern graphic cards to visualize discrete volume data.

In the 2D texturing approach, 2D textures are mapped onto parametrized quadrangular plane segments (quads) that correspond to the tomographic images in the volume stack. To avoid visual gaps, frontal, sagittal and transversal projection quads can be used. This results in a high memory load but is usually very fast, since even modern graphic cards are highly optimized in dealing with 2D textures.

A conceptional problem of 2D texturing are artifacts appearing when using transparent textures. Even modern 3D graphic cards can draw transparent polygons only correct when they are ordered "back to front" (painter algorithm). Since the different orthogonal projection planes are overlapping, this ordering cannot be done without cutting the planes at their intersections. This would not only be time expensive but also create new interpolation artifacts. By using a 3D texture, it becomes also possible to use only projection quads that are orthogonal to the viewing direction. When the object or the point of view is moved, not the quad geometry but the texture coordinates are modified. Additional benefits are the optional use of trilinear filtering to enhance image quality or to use shaders, e.g., to simulate lighting effects.

To increase frame rate, YaDiV allows reducing the number of projection quads. In the 2D texturing case, this also decreases the memory consumption. In the 3D texturing situation, the original 3D texture and with that the memory requirement remains the same, so there is only the speedup due to the reduced number of projection planes. Figure 2 shows the influence of the number of projection planes in 2D (a)–(c) and 3D (d)–(f) texture visualization on visual impression, frame rate and memory consumption ($\Delta = 1$ means every volume slice is used, $\Delta = 2$ every second, etc.). To allow the visualization of very large volume data, it is also possible to force YaDiV to use a lower texture dimension.

Table 1 YaDiV marching cube performance, CS = cube size, T = number of triangles, ms = time in milliseconds, T/s = triangles per second (notebook with Intel Core 2 Duo (2.4 GHz), 4 GB RAM, NVidia 9400M)

Data Set	Segment	CS	T	ms	T/s
CT_Head	bone	4	26568	135	197k
256 × 256 × 113		2	125616	203	618k
		1	543192	765	710k
	brain	4	13288	36	369k
		2	78384	240	327k
		1	474152	578	821k
	jaw	4	984	15	64k
		2	6300	10	610k
		1	30924	49	635k
VIX	bone	4	112192	322	348k
512 × 512 × 250		2	511588	1012	506k
		1	2148580	3183	675k
	flesh	4	89680	93	961k
		2	372896	394	946k
		1	1517008	2111	719k
Obelix_261	body	4	240932	652	370k
512 × 512 × 520		2	1017592	2449	416k
		1	4202888	8308	506k
	bone	4	186112	926	201k
		2	902504	2015	448k
		1	4009216	6818	588k

Interactive segment visualization

To visualize segments in 3D, a self-refining modified Marching Cube Method (MC) is used to approximate a triangulated segment surface: Beginning with a cube size of 4, the cube size is halved until the finest level or a configurable maximum number of triangles is reached. The original MC algorithm [5] visualizes iso-surfaces in regular grid data and uses the grey value differences to smooth the surface. Since segment data is binary, a geometric smoothing is necessary to achieve a natural visual impression without artifacts due to the grid structure of the data. The ambiguity problem of the original MC was solved by using a modified lookup table, resulting in a closed (waterproof) segment surface. Table 1 demonstrates that the visualization speed is truly interactive: even on a notebook, the first refinement step of larger data sets is reached in less than one second.

Ray Casting

Ray Casting belongs to the class of direct volume visualization algorithms. A so-called transfer function maps each intensity value to a color and opacity value. By casting an imaginary ray from the viewpoint through each pixel of the resulting image, the color value is computed by sampling the ray as it traverses the volume data. By using different sub-voxel resolution interpolation methods (linear, cubic, sinc, etc.), very high quality results can be achieved.

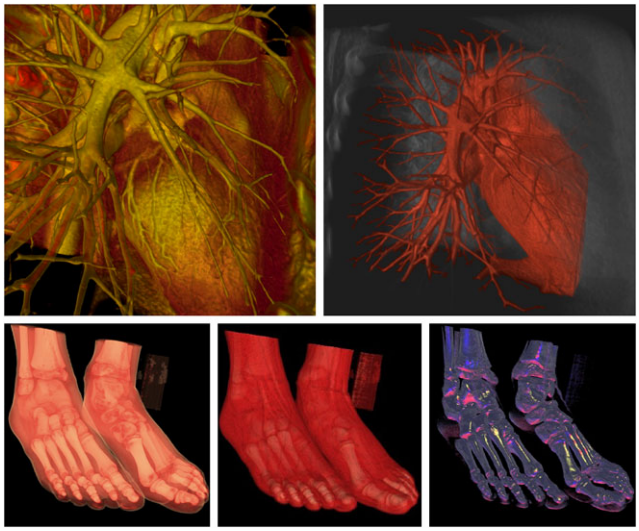

Fig. 3 High Quality Ray Casting Module of YaDiV: Standard Phong Shading and Lit Sphere Mapping

The Ray Casting Visualization of YaDiV was implemented as modular and includes several options. In the most simple case, a classic color/opacity transfer function and phong lighting is used. Additionally, YaDiV supports individual transfer functions for each segment and artistic lighting models such as Lit Sphere Mapping (see [2] and [11]), where a 2D image of a lit sphere (hand drawn or computed) is used to generate different shading effects, shown in Fig. 3. During the manipulation of the transfer function, the 2D/3D Texturing mode can be used as a fast preview for the outcome. The code is modular and extendable so that new ideas and features can easily be integrated.

Similar to the more complex segmentation routines, the ray casting algorithm is written multi-threaded and profits from modern multicore architectures. Strategies like early ray termination, empty space skipping and an intelligent gradient normal caching are also included and guarantee fast high quality results.

4.3 Segmentation

A segment describes a meaningful subset of the voxel data, in a medical context, for example, an organ, muscle or bone. This subset does not have to be connected, e.g., a segment could contain individual bone parts. The process of creating a segment is called “segmentation”.

The most basic way to create a segment is doing it by hand: a medical expert marks the relevant structure(s) on each voxel layer, similar to a classic paint program. Depending on the structure and the resolution of the volume data,

this process can take from several minutes to several hours, also the result may differ when done by different experts, especially on low resolution "blurry" data. The goal of every automatic method is therefore to be either faster or more precise (or at least better reproducible). From the computer vision point of view, several attempts have been made to classify segmentation methods into subclasses (e.g., "voxel-based", "edge-based", "histogram-based", etc.). Since most of the more advanced methods fall into several categories, we will only separate between "model-based" and "general" algorithms in this paper. Model-based algorithms rely on a priori information about the to-be-segmented structures and are therefore designed for specific use cases, such as segmenting the liver. In contrast, general algorithms are comparable to tools that allow (usually in a creative combination of several of them) segmenting many different structures.

YaDiV contains several built-in general segmentation methods: range, region growing, moving contour and atlas-based. Additionally it is possible to perform morphological operations, remove small connected parts to eliminate noise, or to correct the results of the general methods manually in a free draw mode. All methods are implemented as threads, some even use multiple cores (if available) and visualize the intermediate segmentation results interactively during the process. This is especially useful for computationally complex methods, such as the moving contour algorithm which can take up to several minutes, while more simple methods like range or region growing segmentation usually take only a few (milli-) seconds on a normal PC with standard clinical CT/MRT resolution volume data.

All segmentation algorithms will not directly create or modify a segment but always create a so called "selection" of voxels. The selection can be copied into a (new) segment or added to/removed from an existing one. Also, segments can be copied back into the selection for further data processing. Figure 4 illustrates this in an artificial example: a sphere-shaped selection is removed from the three existing segments for flesh, bone and brain.

As a design decision, no specific model-based methods are included within the core version of YaDiV, to avoid a large monolithic software which can do everything and nothing. To keep the software easy to use and the interface simple, model-based algorithms are implemented as plug-ins which can be installed or removed at runtime, allowing YaDiV to fit to the individual requirements of surgeons, neuroradiologists, orthopedists, etc.

Range and region growing segmentation

Range segmentation is a very simple but for many situations still useful tool. By specifying an upper and lower intensity value limit, e.g., the Hounsfield Values for bone if the data is a CT scan, the user selects a range in the histogram defining the segment. The algorithm is very fast and uses no additional memory.

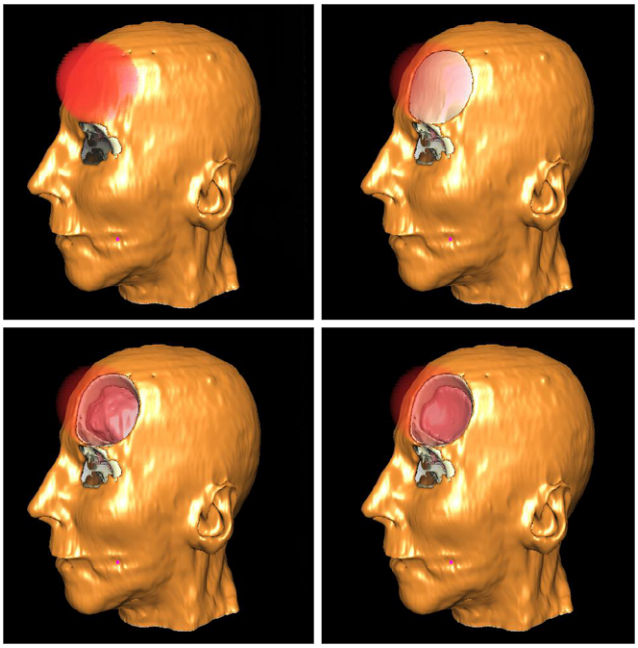

Fig. 4 Example for Boolean Segment operations (Screenshot of YaDiV)

Region growing starts from a user defined starting point (seed) as segment and iteratively collects voxels from its neighborhood. If the intensity values of the neighborhood of the seed lie within a given variance of the seed intensity, the voxels are added to the segment and considered as new seeds. Region growing works also very fast and uses (depending on the implementation) comparatively less additional memory. In practical clinical use cases, a common problem of region growing segmentation is that the algorithm tends to expand over small connected tubes into neighboring structures, e.g., two individual bones that are close together. This can be avoided by the use of the so-called blocking segments, defining "no-go" regions for the expansion process. Alternatively, small connections can be removed in a post processing with a morphological erosion that splits the segment into different connectivity regions.

(*Level set*) *moving contour methods*

Instead of a single point, moving contour methods evolve an initial contour line of a segment. In each evolution step, the contour expands (or shrinks) depending on "inner" or "outer" parameters. The classic problem of the moving contour algorithms was that the initial contour's topology has to match the possibly unknown topology of the final segment. The solution of this is the implicit definition of the contour as a level set of a higher dimensional function. In [8], Osher and Sethian proposed defining the contour as the

(zero-)level set of its own signed distance function (with negative distance values in the inside of the segment). Instead of the contour itself, the distance function is evolved, allowing the contour topology to change during the process.

YaDiV contains two different active contour modules. The first is an edge stopping based approach which goes back to the original idea of [8] but has been extended to the 3D case with different stopping functions. The basic idea is that a contour C expands in every contour point in the normal direction with a (non-negative) evolution speed F. During the evolution, F is influenced by an inner factor, the curvature of C and an outer factor, the so-called stopping function g. To stop the expansion of the contour at edges in the voxel image, g is usually defined using the gradient of the volume data. The user can choose the curvature weight (resulting in sphere-shaped contours or allowing more "fingers"), the expansion speed as well as the stopping function. Due to its design, this approach is useful when looking for segments in an image with sharp edges. The curvature factor controls whether the evolving contour may squeeze through small holes—or not. Due to the fact that the method has to calculate the gradient, an effective storage and caching mechanism has to be used to reduce the memory footprint.

The second moving contour approach is based on the energy minimization idea, trying to minimize a functional of the so-called "inner" and "outer" contour energy, consisting of the mean intensity values of the voxels inside or outside the contour. In [3], Chan and Vese suggested (for the 2D case) including the volume of the inside and the area of the contour as additional weighted parameters. With these weights, the user can control if the evolution aims for more compact or complex branching figures. The energy-based moving contour approach uses again the level set of the signed distance function. As opposed to the edge stopping method, it is also suitable for segments with blurry edges. As no gradients have to be calculated, the energy method is faster and also less memory consuming than the edge stopping algorithm.

Figure 5 shows the 2D visualization of the intermediate results at different evolution steps of both approaches.

To speed up both contour evolution approaches, several optimization strategies have been used. The first is the so-called narrow band strategy, limiting the recalculation of the distance function to a narrow area around the segment contour. Additionally, the considered volume data can be limited to a small, user defined box containing the to-be-segmented structure. In the last step, the calculation of the concrete distances is approximated by our own fast "onion-ring" method. Nevertheless, both methods remain time expensive.

Due to their design, moving contour methods are very flexible and can be adapted to many use cases. A typical use case could be to get a good starting segment (generated, e.g., by a more simple method or manual segmentation) and use its contour to initialize the moving contour algorithm with a low expansion speed. But the mathematical background also leads to a lack of user acceptance, as, for example, most surgeons do not want to think about curvature factors or contour area weights when performing a concrete segmentation. Here there is much room for future developments, using these powerful methods in settings where parameters can be estimated and automatically fitted to a given problem, thereby unburdening the user.

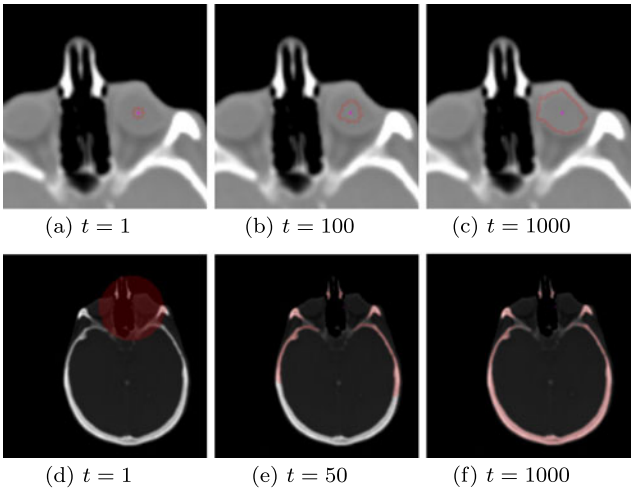

Fig. 5 Moving Contour Segmentation with YaDiV: Edge Stopping (**a–c**) and Energy Minimization (**d–f**)

Atlas based segmentation

Atlas-based segmentation is a somewhat special case for general segmentation methods, as the atlas could be understood as an implicitly given model. But since the method itself is generic, it still belongs to the general class. A good overview over atlas based segmentation is given in [9].

The basic idea is to use previously segmented volume data of the same body part (atlas) to find similar looking structures in new tomography scans. This is done in three steps: (i) choosing an atlas, (ii) registering the atlas to the scan, (iii) using the registration transform to identify segments.

The first step requires user interaction—an appropriate atlas could be the scan of a patient of the same gender, age or with the same disease. If the scan is part of a study, previously segmented scans from the same patient can be used.

The registration step is performed automatically; in an iterative process, a transform T is found that maximizes the similarity with respect to some similarity measure, e.g., the mutual information of both volume data sets. In the current implementation, the registration is performed in two steps; an affine (rigid) and an elastic (non-rigid, using cubic b-splines) transformation. To speed up the process, it is

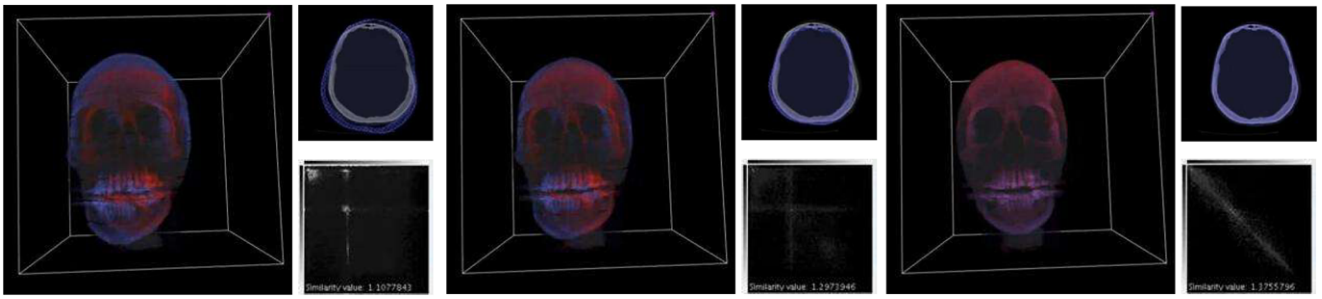

Fig. 6 Atlas based segmentation: using mutual information as similarity measure

possible to use only a configurable percentage of the grid data to compute the similarity measure. In our experiments it turned out that considering 10% of the voxel data were enough. Figure 6 shows the registration of two CT scans (red and blue) together with the visualization of their joint histogram.

If a sufficient similarity has been reached, the same transformation is applied to the atlas segments. If an appropriate atlas was chosen, the segments will already be valid in the new scan. If such an atlas was not available, the identified segments can be used in a post processing refinement step, e.g., with a moving contour approach. It is also possible to use more then one atlas (see, for example, [1]).

4.4 Virtual reality support

Tomography scans produce three dimensional data, but output (monitor) and input devices (mouse) are still two dimensional. This leads to a lack of intuitive interfaces. Even the three dimensional looking images produced by 3D texturing or ray casting methods are still two dimensional, similar to classic photographs. The recent advancements in Virtual Reality (VR) technology raise the hope of a more natural and intuitive access to the relevant information.

Stereographic visualization

Stereographic visualization techniques allow an immediate scene understanding. While on 2D monitors scene understanding comes usually from shading, moving the scene and eye-hand navigation, stereography allows seeing even complex objects in an intuitive human way, allowing deeper understanding and diagnosis. Similar to popular movies like "Avatar", the observer gets the impression that a "real" object is directly in front of (or behind) the monitor. For stereo visualization, it is necessary to compute not only one, but two perspectively correct images: one for the left and one for the right eye. This usually halves the frame rate.

Since Java3D, which is used by YaDiV to render 3D objects, directly supports stereographic visualization, it is possible to explore DICOM data on different stereographic devices, e.g., stereo monitors, head mounted displays (HMD) or stereographic projectors. No special stereo version of the software is needed. When YaDiV detects stereographic hardware, stereo visualization is enabled by default.

Haptic interface devices

While the stereographic visualization allows seeing virtual objects as if they were real, haptic interfaces allow even touching and manipulating them. Sometimes referred to as haptic input devices, this terminology is somewhat misleading as mechanical energy is not only entered by the user but also returned by the device. Haptic devices can be classified by their number of degrees of freedom, their workspace dimension, the number of haptic interaction points and the maximum force.

In the field of medicine, haptic input devices have currently three major use cases: pre-operative planning, robot assisted (minimal invasive) surgery (a good overview is given in [7]) and training simulations. Only little research has been done to enhance the human–computer interface, especially in 3D tasks like volume segmentation, registration or navigation, where a 3D interface could allow complete new intuitive tools.

There exists a prototype version of a haptic support module for YaDiV which implements surface-based as well as voxel-based haptic scene rendering. As currently only few end users own haptic interfaces, this part will be redesigned to become a plugin and is planned to be published early next year.

4.5 Using YaDiV with non-medical data

Although YaDiV was developed with medical data in mind, this is not the only field of science that produces regular grid data. Engineers, for example, use tomography data for materials research, while mineralogists analyze crystals and vesicles shown in Fig. 8. We were approached by several scientists who wanted to analyze their data and found existing specialized systems insufficient. Since their data was not stored in the medical DICOM format, an image import function was developed, that allows YaDiV to convert image

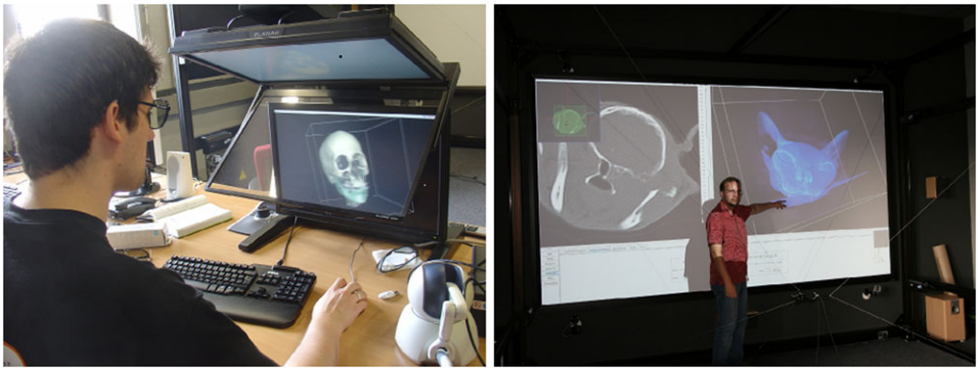

Fig. 7 Support of Stereographic Hardware

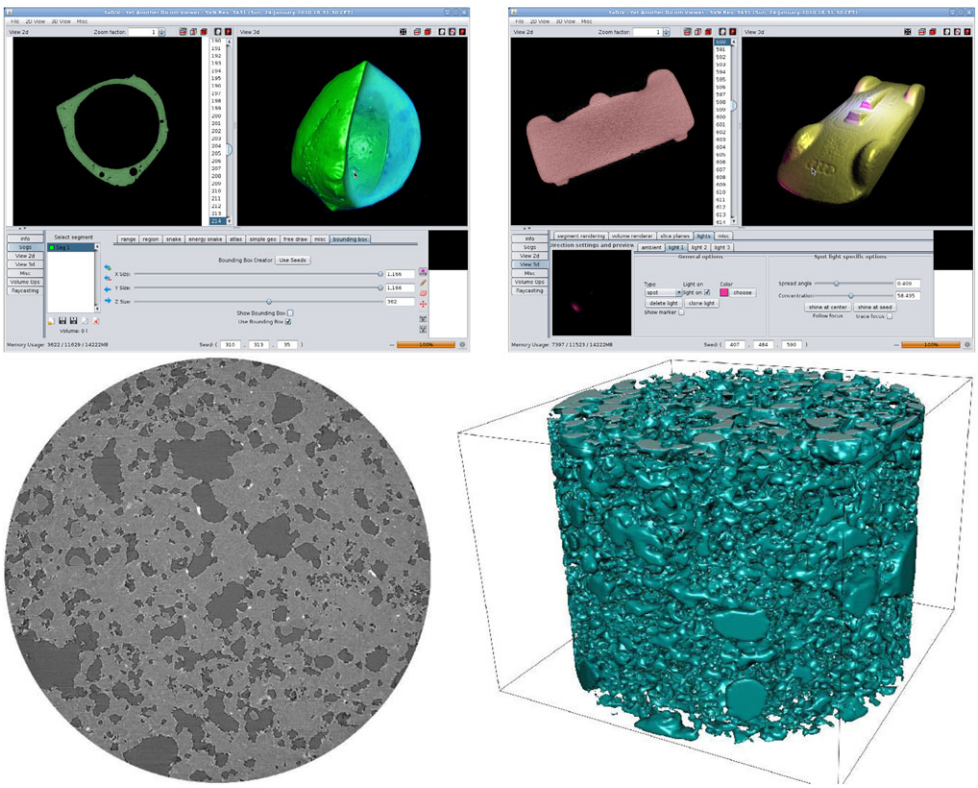

Fig. 8 Non-medical volume data examples from mechanical engineering and mineralogy

data from the most common formats (JPG, TIFF, PNG, etc.) to a volumetric representation. Thus, the complete array of functions for segmentation and visualization of YaDiV were available for the analysis of the data.

Especially the data from scientists of the Institute of Mineralogy at the Leibniz Universität of Hannover was technically demanding: in a common project, YaDiV is used to visualize and analyze very high resolution data (2048 × 2048 × 2048). For this, new memory saving storing techniques with lossless reduction were developed. As a result, the possibility of fast 3D visualization lead the mineralogists to a deeper understanding of so far unseen structures. A statistical analysis plugin developed for this use case allows for a much better calculation of statistically relevant key parameters than conventional 2D methods, as well as demonstrating the plugin capabilities of the system.

4.6 Implementation

YaDiV is implemented in Java, therefore it is completely independent from hardware and operating system and has been tested on several platforms. The 3D graphic interface uses the open source API of Java3D and supports many VR components such as stereographic visualization (Fig. 7) on multiple devices.

Due to efficient programming and the consequent use of multi-threading, YaDiV profits from modern multicore architectures—the de facto standard in every new PC or laptop. The result is not only a very fast implementation of,

Table 2 YaDiV Code Statistic (without plugins)

Number of packages	26
Number of classes	431
Total lines of code	42630

e.g., complex segmentation methods, but also an interactive behavior, allowing to animate all processes and enabling the user to abort them at an early state, e.g., when it is clear that the final outcome of a moving contour method will not match the desired anatomical structure.

The native data structure for segments is a binary regular grid with the dimensions of the voxel data, implemented internally as a one-dimensional `int`/`long` array with a three-dimensional access interface. All segment related data structures and methods use native bitwise integer arithmetic which is directly executed on the processor (see [12]) and results in a speedup factor of up to 32 (up to 64 on 64 bit computers).

An intuitive GUI with a rich integrated help system allows even untrained users to achieve the desired results. The support for haptic input devices as well as stereographic visualization guarantees a natural access to the information and allows the development of complete new work flow approaches in interaction with the 3D anatomical data.

A clear separation between core and module packages and the use of design patterns allows an easy extension of the software. To prevent that customized modules "bleed" into the main API, message passing and interface concepts are used and the core library already contains a rich set of standard operations.

5 Summary and outlook

The YaDiV system allows fast and interactive visualization and segmentation of voxel data. The program is truly platform independent and can be started even from an USB stick. At the beginning of the development, it was not sure if Java would be fast/memory efficient enough for large volume data, yet it turned out that in practical tests YaDiV appears to be as fast as or even faster than many other free and commercial products.

YaDiV supports stereographic output devices but also runs on every normal PC/Laptop. During the development, the software was constantly tested by our medical partners whose valuable feedback greatly influenced concepts and interfaces.

The software is a good choice for interdisciplinary research teams, not only in the medical field. Due to the import abilities, the software is usable for a wide range of applications that deal with regular grid data, providing the scientific community with a free platform which comes with a rich tool set and many unique features.

By becoming open source, this software can serve as a platform for trying out new ideas in all fields of science that deal with regular grid data and become a valuable contribution to the scientific community. A regularly updated closed source version of YaDiV is already available at http://www.welfenlab.de/yadiv. The open source release is scheduled for January 2011.

5.1 VR and man–machine interaction

As mentioned before, only few projects focus on the new possibilities haptic and stereographic visualization offer to the man–machine interaction.

A first example could be a human assisted registration step in an atlas based segmentation. All known registration methods are very good in optimizing a transformation to get a local similarity maxima, but sometimes miss a (human obvious) better global solution. Humans, on the other side, are very good at detecting global maxima but lack the ability (and patience) to do a fine voxel-by-voxel optimization. A system that would allow the medical expert not only to visualize the situation stereographically but also to grab and manipulate the data could be used as a fast, rough pre-registration which is then automatically refined by algorithms.

The authors of this paper believe that the new haptic interface will influence the clinical workflow with 3D volume data in a similar way as the invention of the classic computer mouse influenced 2D tasks like paint programs or word processing. Therefore, it is planned to use YaDiV as a platform to develop and test new approaches in close contact with our medical partners. The necessary modules (stereographic and fast visualization, segmentation algorithms, support for haptic interfaces) are already implemented.

5.2 Current projects

YaDiV is still in active development. As opposed to the recent years, when the platform itself was extended and redesigned, the development now focuses on specialized research projects which are implemented as plugins.

An ongoing project in combination with Laboratory of Biomechanics and Biomaterial (LBB) from the Hannover Medical School (MHH) focuses on the automatic computation of knee joint kinematics by 3D in-vivo analysis using an upright MRT. Together with the medical experts from the MHH, a plugin is developed that will combine existing registration methods for patella, femur and tibia with in house developments to analyze the 3D bone position with and without natural loading.

Another project is focusing on the (semi-) automatic segmentation of the orbita for post-surgical analysis. This work

is a part of an international clinical research project by the AO foundation (Davos, Switzerland) with principal investigator Prof. Gellrich, from the department for craniomaxillo-facial surgery, MHH. Together with Dr. Harald Essig from the same department, our goal is to implement a plugin for reliable orbita segmentation, that will allow measuring certain statistics, e.g., the volume or the angle between orbita floor and the medial wall.

5.3 What is left to do?

Even after reaching a mature state, there are still many things left to do. After cleaning up the code and releasing the software as open source (scheduled for October 2010), an important usability feature would be to offer translations of the dialogues, menus, etc. into different languages. Another important step will be to rewrite the existing haptic device module to work with the new plugin interface.

Even if YaDiV is already capable of dealing with many different forms of regular grid data, it would also be interesting to see if this could be extended to non-regular (rectilinear, structured, maybe even unstructured) grids. Also, the current implementation of the core data structures are only capable of dealing with numbered grid data and it could be worth to make tests, to see if a more general approach (e.g., by using generics) would decrease the performance drastically.

With the upcoming of massive parallel architectures, it will be necessary to find a way to make use of their arithmetic power without losing one of the core features, the independence from hardware and operating system. A possible solution for this might be the use of OpenCL. Currently, several groups are working on a connection between Java and OpenCL, e.g., JOCL, libCLcalc or the OpenCL-Branch of the Native Libraries For Java Project.

Acknowledgements We would like to thank our medical partners from the MHH, especially the departments of neuroradiology, the clinic for oral and facial surgery and the laboratory of biomechanics and biomaterials.

The following people helped to bring YaDiV to its current state: Benjamin Berger, Sarah Cichy, Benjamin Fleischer, Richard Guercke, Robert Meyer, Maximilian Müller, Dominik Sarnow, Björn Scheuermann, Lara Toma, Marc Christoph Vollmer, Johannes Wahle and Yifan Yu.

References

1. Aljabar, P., Heckemann, R., Hammers, A., Hajnal, J., Rueckert, D.: Multi-atlas based segmentation of brain images: Atlas selection and its effect on accuracy. NeuroImage **46**(3), 726–738 (2009). doi:10.1016/j.neuroimage.2009.02.018
2. Bruckner, S., Gröller, E.: Style transfer functions for illustrative volume rendering. Comput. Graph. Forum **26**(3), 715–724 (2007)
3. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Trans. Image Process. **10**(2), 266–277 (2001). doi:10.1109/83.902291
4. Friese, K.I.: Entwicklung einer Plattform zur 3D-Visualisierung und -Segmentierung medizinischer Daten. Ph.D. thesis, Leibniz Universität Hannover, Faculty of Electrical Engineering and Computer Science, Welfenlab, Germany (2010)
5. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, vol. 21, pp. 163–169. ACM Press, New York (1987). doi:10.1145/37401.37422
6. McCormick, B., DeFanti, T.A., Brown, M.D. (eds.): Visualization in Scientific Computing. ACM Comput. Graph., vol. 21(6). ACM Press, New York (1987)
7. van der Meijden, O., Schijven, M.: The value of haptic feedback in conventional and robot-assisted minimal invasive surgery and virtual reality training: a current review. Surg. Endosc. **23**(6), 1180–1190 (2009). doi:10.1007/s00464-008-0298-x
8. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. J. Comput. Phys. **79**(1), 12–49 (1988). doi:10.1016/0021-9991(88)90002-2
9. Rohlfing, T., Brandt, R., Menzel, R., Russakoff, D.B., Maurer, C.R., Jr.: Quo vadis, atlas-based segmentation. In: Suri, J., Wilson, D.L., Laxminarayan, S. (eds.) Registration Models. The Handbook of Medical Image Analysis, vol. III, pp. 435–486. Kluwer Academic/Plenum, New York (2005). Chap. 11
10. Schroeder, W., Martin, K., Lorensen, B.: The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th edn. Kitware, Inc., Clifton Park (1997)
11. Sloan, P.P.J., Martin, W., Gooch, A., Gooch, B.: The lit sphere: a model for capturing NPR shading from art. In: GRIN'01, pp. 143–150. Can. Inf. Proc. Society, Toronto (2001)
12. Warren, H.S.: Hacker's Delight. Addison-Wesley Longman Publishing, Boston (2002)

Karl-Ingo Friese studied Mathematics and Computer Science at the University of Hannover. He wrote his Diploma Thesis on the topic: "Surface Reconstruction of Volume Models based on Discrete Data" and finished his studies 2002. Since then he is working as a Senior Research Assistant at the Department of Computer Graphics, Institute of Man–Machine-Communication at the Leibniz Universität Hannover. During that time he wrote his PhD thesis with the title "Development of a Platform for 3D Visualization and 3D Segmentation of Medical Data" (YaDiV). In former projects, he was researching in the field of interactive landscape planing and scientific visualization with game engines. His main research focus is scientific visualization in interdisciplinary research projects.

Philipp Blanke studied Mathematics and Computer Science at the University of Hannover and wrote his diploma thesis on the topology of medial sets of polyhedra in Euclidean space. He received his diploma in Mathematics (Master's equivalent) in December 2004. Since then he is working at the Division of Computer Graphics of the University of Hannover as a research assistant and Ph.D. candidate. Currently, he is writing his Ph.D. thesis on Fast Inverse Material Flow in Hot-Forging.

Franz-Erich Wolter has been a full professor of Computer Science at the Leibniz University of Hannover since the winter term of 1994/1995 where he directs the Division of Computer Graphics and Geometric Modeling called Welfenlab. Before coming to Hannover, Dr. Wolter held faculty positions at the University of Hamburg (in 1994), MIT (1989–1993) and Purdue University in the USA (1987–1989). Prior to this, he developed industrial expertise as a software and development engineer with AEG in Germany (1986–1987). Dr. Wolter obtained his PhD in 1985 from the Department of Mathematics at the Technical University of Berlin, Germany, in the area of Riemannian manifolds. In 1980, he graduated in Mathematics and Theoretical Physics from the Free University of Berlin. At MIT, Dr. Wolter codeveloped the geometric modeling system Praxiteles for the US Navy from 1989 to 1993 and published various papers that broke new ground applying concepts from differential geometry and topology on problems and design of new methods used in geometric modeling and CAD systems. In this context, he has contributed pioneering concepts to shape and image cognition, construction and compression via tools from differential geometry employing, e.g., spectra of the Laplace operator and computations of the medial axis and geodesics in higher dimensional Riemannian spaces. During the past ten years, he has extended his research to include projects on haptic VR systems as well as mechanical-, bio-mechanical-simulation and visualization systems. Dr. Wolter is a research affiliate of MIT.