METK – The Medical Exploration Toolkit

Christian Tietjen[1], Konrad Mühler[1], Felix Ritter[2], Olaf Konrad[2], Milo Hindennach[2], Bernhard Preim[1]

[1]Institut für Simulation und Graphik, Otto-von-Guericke-Universität Magdeburg
[2]MeVis Research, Center for Medical Image Computing, Bremen
tietjen@isg.cs.uni-magdeburg.de

Abstract. In the following we will describe concept and realization of the Medical Exploration Toolkit – the METK. The METK is designed for loading, visualizing and exploring segmented medical data sets. It is a framework of several modules based on the free MeVisLab, a development environment for medical image processing and visualization. The framework is platform-independent and freely available. We will also present several different applications, developed with the METK.

1 Introduction

For many tasks in computer-aided medical visualization, it is not sufficient to display just one anatomical structure or to perform a simple volume rendering. Building applications for complex case analysis with advanced visualization techniques is still an extensive job. Low level libraries like VTK [1] are very flexible, but the application development for supporting a special task is very time consuming. Julius [2] and VolV [3] are based on VTK and Qt and provide a faster application development. However, the programming environment is still C++, thus, substantial programming knowledge is still required.

MeVisLab [4] provides an easy to learn, script based framework for medical image processing and visualization, based on modular visual programming. Among other libraries, VTK and Qt are also integrated. Thus, a full image processing and visualization pipeline can be used for application development without deep programming knowledge.

Currently, there is no general communication or data standard available in public with MeVisLab. Hence, newly designed visualization or exploration techniques need to be adapted for each application. Also simple tasks such as loading specific patient image data or the visualization of segmented structures require a deep knowledge of the used techniques. The Application Frame (AppFrame) facilitates the integration and testing of new algorithms and the development of application prototypes with MeVisLab that can be used in clinical environments [5]. While AppFrame indeed addresses generic issues such as DICOM image import and export, user-management, reporting, and documentation functionality, it does not provide advanced visualization and exploration techniques. In addition, the AppFrame is not available in public.

Hence, we developed the Medical Exploration Toolkit (METK) as a comprehensive collection of basic and advanced visualization and exploration techniques with a common communication protocol. These underlying structures are invisible for application developers. Thus, they can focus on application requirements and come up with complex applications using just a few METK modules. The METK is platform-independent and will be freely available at the end of 2007.

2 Materials and Methods

The METK is composed of five different layers: communication layer, database layer, visual processing layer, visualization layer and exploration layer (Fig. 2). For each layer, several METK modules are available that can be combined to networks, building up an application.

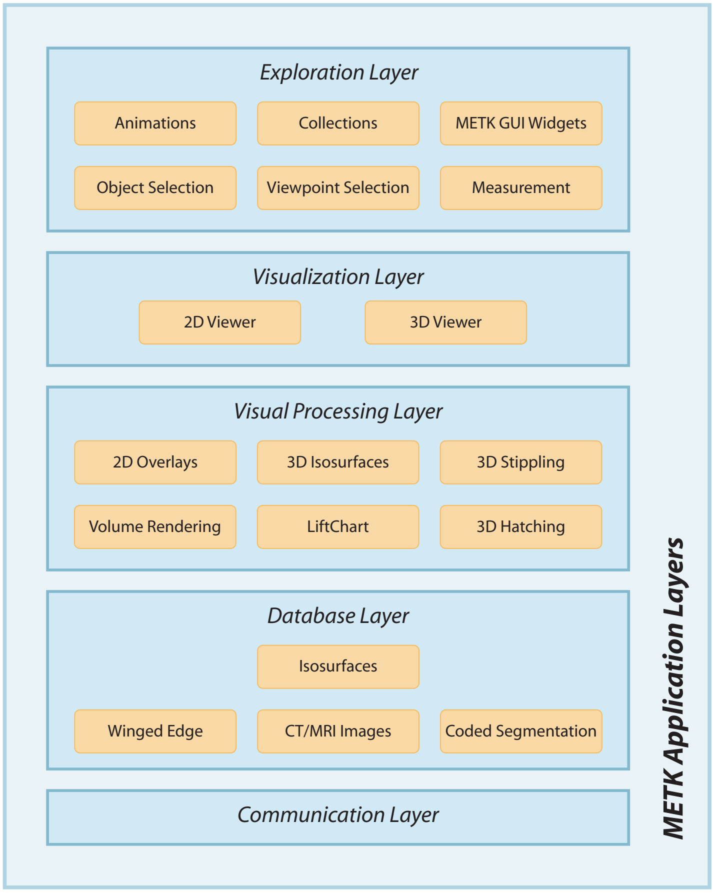


Fig. 1. Layer schema of the METK

The *communication layer* enables all modules to share events with other modules and to be notified about global events, like loading or closing a case. A case consists of original CT or MRI datasets with additional segmentation masks, descriptions about their anatomical affiliation and patient information. Also, visualization parameters may be stored. All METK modules can access and manipulate these data. So, if one module changes the color of a structure, all other METK modules may react on this event and may adjust their own settings. The main visualization properties are always shared to provide a synchronized visualization in all modes and viewers.

The *database layer* contains all necessary data: image data and segmentations masks, as well as isosurfaces and their winged edge data. The isosurfaces are generated by default, because they are needed at most. All other data may be made available optionally, if one visualization module needs it. Since the computation of an isosurface may be computationally expensive for some structures, it is possible to compute those surface only if necessary, e.g., if they are treated as visible. For setting up a new case or expanding an existing one, there are still several modules available in MeVisLab. The METK does not provide particular segmentation algorithms, like LiveWire or Watershed, that are already present in MeVisLab. It just provides functions for storing the retrieved segmentation information in a METK readable manner. The entire case must be provided in form of an XML file. The case data contains information about the segmented structures (e.g., type and color). The image data itself may be provided in one of many standard formats like DICOM, Analyze or RAW.

The *visual processing layer* interprets the data. Several visualization techniques are implemented, which are not part of the MeVisLab library. Isosurfaces may be displayed by shaded surfaces, silhouettes, stippling or hatching. Image data may be displayed as orthogonal 2D slices or as volume rendering. In both modes, the segmentation masks may be displayed by colored overlays or by tagged volume rendering. More advanced techniques like displaying LiftCharts [6] or distances to other structures are also available in 2D and 3D [7]. Every visualization technique may be turned on or off for every single structure. Thus, it is possible to render the whole scene with isosurfaces, but the pathologic structures with an additional silhouette.

The *visualization layer* displays and combines the visualization of each structure provided by the visual processing layer. It is possible to combine different visualization techniques in separated 2D and 3D viewers as well as in combined 2D/3D viewers. Simple screen shot facilities are also provided.

The *exploration layer* contains high level exploration techniques. The entire state of the METK database may be stored along with a short comment, a caption, a small thumbnail and a screen shot, referred to as a *collection*. A collection is later accessible for further explorations or presentations. Furthermore, the datasets may be explored by using automatically generated animations [8]. Those animations are described in a high-level manner and can be automatically reused for many similar cases. The development of application interfaces is supported with well designed METK widgets on the basis of Qt, e.g., for structure lists or panels to change visualization parameters. The navigation in 3D is supported by advanced techniques for automatic viewpoint selection [9], by intelligent object selection algorithms (if multiple semi-transparent structures are overlapping each other) and by facilities for structure measurement.

Only the communication and database layers are mandatory. All other modules are optional. However, it is possible to combine the functionalities of different modules. For example, it is possible to smoothly switch from one collection or the current viewing state to another collection by using animations.

We also developed a new technique for the efficient management of segmentation masks. Normally, segmentation masks of structures are saved in single files for each structure. This results in extensive memory consumption, because for complex operations like combined 2D overlay visualization of multiple structures, all segmentation masks needs to be hold in memory. To avoid that, a *coded segmentation* module was developed. The coded segmentation combines all segmentation masks derived from the same image in only one image. Each voxel value represents a combination of structures that the voxel belongs to. The information of relation between voxel values and belonging structures is stored in the case. The coded segmentation is especially useful for liver surgery, where all segmented structures are overlapping each other (e.g., tumor, Couinaud segment and vessels).

3 Results

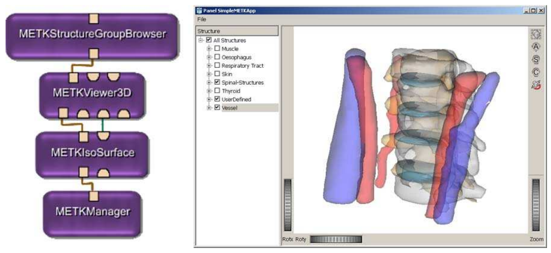
As shown in Fig. 3(a), only four modules are necessary to load and display a spine case dataset with 34 different structures. The METKManager loads the case data and provides the obligatory isosurface data (Communication and Database Layer). METKIsosurface renders the isosurfaces with shaded surfaces (Visual Processing Layer) and the METKViewer3D finally displays the shaded isosurfaces (Visualization Layer). The METKStructureGroupBrowser enables the user to show or hide the different groups of structures or separate structures (Exploration Layer).

Currently, two applications based on METK are published, while more are in a pre-release development state. The NeckSurgeryPlanner [7] is a software assistant for pre-operative planning and visualization of neck dissections (Fig. 3(b)). The LiverSurgeryTrainer [10] is a training system to learn specific tasks for liver donor transplantations and tumor resections.

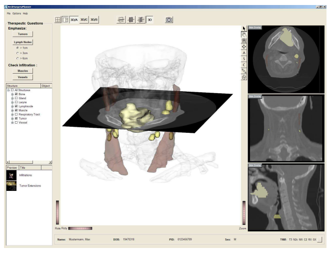
4 Discussion

The METK helps to develop basic applications very fast. It provides basic facilities like case management as well as advanced visualization and exploration techniques like stippling or viewpoint selection. No programming knowledge is necessary to set up high level applications. The resulting applications are as slim as possible, because the basic database and visualization layers are optional. The METK is platform-independent and will be freely available at www.metk.net.

Fig. 2. (a) Network of an application to explore segmented structures in 3D and to change their visibility. (b) A screen shot of the NeckSurgeryPlanner



(a) Simple network showing a spine dataset.



(b) NeckSurgeryPlanner

A great deal of work still needs to be done. A kind of METK self-management would be helpful. Some modules need the guaranteed existence of another, for example, the animation module needs a 3D viewer to render. When developing complex applications, the separation between case and application data is still a problem. For an application with multiple viewers, the viewer parameterization should be stored in the case, like image windowing information in DICOM, or separately, because it is undefined on loading the case in another application.

Acknowledgement. This work was supported by the BMBF in the framework of the SOMIT-FUSION project (FKZ 01|BE 03B) and by the Deutsche Forschungsgemeinschaft (DFG) (Priority Programme 1124, PR 660/3-1).

References

1. Kitware Inc . VTK Home Page; 2007. http://www.vtk.org.
2. Keeve E, Jansen T, Krol Z, et al. JULIUS: An extendable software framework for surgical planning and image-guided navigation. Proc MICCAI. 2001; p. 1336–7.
3. Pfeifle, M et al . VolV: Eine OpenSource-Plattform für die medizinische Visualisierung. In: Proc. CURAC; 2007. p. 193–6.
4. MeVis Research. MeVisLab Home Page; 2007. http://www.mevislab.de.
5. Rexilius J, Kuhnigk JM, Hahn HK, et al. An application framework for rapid prototyping of clinically applicable software assistants. In: GI Jahrestagung; 2006. p. 522–8.
6. Tietjen C, Meyer B, Schlechtweg S, et al. Enhancing Slice-based Visualizations of Medical Volume Data. In: EuroVis; 2006. p. 123–30.
7. Tietjen C, Preim B, Hertel I, et al. A Software-assistant for pre-operative planning and visualization of neck dissections. In: Proc CURAC; 2006. p. 176–7.
8. Mühler K, Bade R, Preim B. Adaptive script based animations for intervention planning. Proc MICCAI. 2006; p. 478–85.
9. Mühler K, Neugebauer M, Tietjen C, et al. Viewpoint selection for intervention planning. In: EuroVis; 2007. p. 267–74.
10. Cordes J, Mühler K, Preim B. Die Konzeption des LiverSurgery-Trainers. In: GI-Jahrestagung; 2006. p. 514–21.