Section: **Visualisierung**

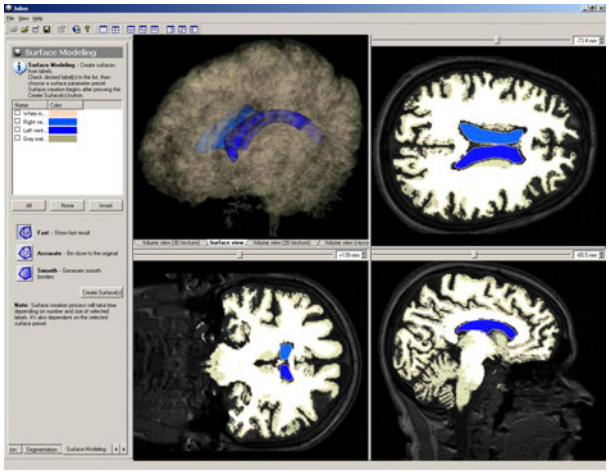
ID: 117

Abstract-Title:
JULIUS - A SOFTWARE FRAMEWORK FOR MEDICAL APPLICATIONS

Authors:
F. Fracassi[1], J. Mueller[1], M. Wedekind[1], B. von Rymon-Lipinski[1], T. Jansen[1], E. Keeve[1]
[1] Forschungszentrum caesar

Abstract-Text:
Purpose: Most medical software systems need standard functionality for processing and visualization of medical datasets. A lot of work is needed to set up this infrastructure before one can begin to concentrate on the actual task: implementing specific and novel medical knowledge into the software system. The Julius Software Framework provides well tested and optimized components for such tasks while being easily extendible as well. Such an approach effects in rapid and high quality application development for medical applications.
Method: Julius is designed consequently as a flexible component system. The components are organized in three software layers: The "core layer" provides basic functionality, like the low-level component architecture and an inter-component messaging system. The following layers are composed of a configurable set of components, called plug-ins. The "data-processing layer" contains data plug-ins that provide abstractions for medical data sets (volumetric or surface data). Medical operations like polygonal mesh generation, segmentation or registration and data import and export (DICOM, STL, VTK, etc.) are encapsulated in controller plug-ins. Finally, the "visualization layer" provides functionality for the development of medical visualization components, including slice views, high-quality raycasting and hardware-accelerated approaches such as 3D-texture-mapping and point-based splatting. All components can be accessed via abstract interfaces, facilitating the replacement and extension of functionality. A specific medical application is constructed by reusing an appropriate configuration of available plug-ins and development of application-specific components. The user interface (UI) can be adapted by XML configuration files. Therefore UI designers can rapidly develop different UI-views without changing the application code. Julius is written in C++ using Qt, resulting in an efficient cross-platform framework.
Results: The Julius development started in 1999 with a scientific background and has evolved to an industrial-strength project. The implementation of an industrial-driven software development process as well as feedback from different scientific and industrial partners has made Julius a stable base for medical applications. The strict use of the modularity concept guarantees that medical workflows are freely configurable and thereby Julius can be applied in a variety of medical application areas. One example is its successful utilization for dental implantology by our commercial partner. Conclusions: Julius is a flexible, well-proven framework for the development of medical applications. Its multi-layer architecture allows rapid application development and gives the user the freedom to concentrate on the actual problems that come with the specific requirements of the respective medical field.www.julius.caesar.de

Bild 1/JPG


Surface Modeling
All
None
Invert
Fast
Accurate
Smooth
Create Surface(s)
Segmentation
Surface Modeling