Interaktives System zur Darstellung funktionaler Bilddaten

I. Erki (*), H. Kleines (*), K. Ziemons (**), K. Zwoll (*)
Forschungszentrum Jülich (KFA)
Zentrallabor für Elektronik (*), Institut für Medizin (*)
52425 Jülich
I.Erki@KFA-Juelich.de

Im Rahmen der Hirnforschung kommen derzeit die Bildaufnahmesysteme MRI, SPECT, PET und MEG im Forschungszentrum Jülich zum Einsatz. Studienspezifische Analysen dieser Modalitäten stehen im Mittelpunkt der Forschungsaktivitäten. Ein Bestandteil des aktuellen Projekts M-FIBRe ist die Implementierung eines interaktiven Systems zur Darstellung funktionaler Bilddaten. Die Voraussetzung einer netzweiten Anbindung dieser Applikation führte zu der Entscheidung die Programmiersprache JAVA in einer Machbarkeitsstudie zu untersuchen. Funktionale Elemente der integralen Oberfläche werden vorgestellt und anhand der Zeitkomplexität eingeordnet.

Keywords:
verteiltes System, JAVA, funktionale Bildgebung, Aufwandsabschätzung

1 Einleitung

Im Rahmen des M-FIBRe-Projekts (Multi-Modality Functional Imaging for Brain Research) mit dem Institut für Medizin des Forschungszentrums Jülich werden die verschiedenen, bildgebenden Modalitäten MR, SPECT, PET und MEG über eine gemeinsame Datenbank in ein Gesamtsystem integriert. Weitere Schlüsselkomponenten des Systems sind die audio-visuelle Kommunikation (Videokonferenz), das kooperative Arbeiten an gemeinsamen Dokumenten (White-Board) und die Verwendung der Breitbandkommunikation ATM (Asynchronous Transfer Mode). Konzeptionell basiert das System auf einer heterogenen, verteilten Umgebung. Darstellungen sowie Kommunikation werden über Bildarbeitsplätze (Clients) mit integralen Oberflächen durchgeführt. Verschiedene Visualisierungsfunktionen kommen in der Oberfläche zum Einsatz. Anfragen der Clients werden durch funktionalitätsspezifische Server in nebenläufigen Prozessen (Threads) bearbeitet. Zur Zeit wird evaluiert, ob die plattformunabhängige Programmiersprache JAVA für die Implementierung geeignet ist. Zu diesem Zweck wird ein Prototyp der Bildverarbeitungs- und visualisierungsfunktionen in JAVA implementiert.

2 Infrastruktur von M-FIBRe

Die Infrastruktur des M-FIBRe-Systemarchitektur entspricht konzeptionell einer verteilten Umgebung. Die Verwaltung großer Bilddatenmengen erfordert die Verwendung von Datenbanken. Im laufenden Projekt ist die Integration einer relationalen Datenbank (ORACLE) vorgesehen. Die Verwaltung von Dienstanfragen an die Datenbank wird durch zentrale Instanzen durchgeführt. Im allgemeinen wird die Realisierung der Systemarchitektur auf einem Client/Server-Modell beruhen. Benutzerorientierte Anfragen an Systemkomponenten werden durch eine integrale Oberfläche, welche in der plattformunabhängigen Programmiersprache JAVA [1, 2, 3, 4] implementiert wird, durchgeführt. Die Integration verschiedener Plattformen unter den Betriebssystemen *SOLARIS 2.5*, *DIGITAL UNIX 4.0, IRIX 6.2*, *Windows 95* und *Linux 2.0* sind für die Realisierung des Systems vorgesehen. Funktionen des Systems sind die *Darstellung der Bilddaten*, *Verwaltung und Administration* von Daten, *audiovisuelle Kommunikation* und *kooperatives Arbeiten* an Dokumenten. Die Übertragung der Daten wird über ATM (*Asynchronous Transfer Mode*) mit einer Übertragungsrate von 155 MBit/s realisiert. In der Abbildung 1 sind die verschiedenen Systemkomponenten dargestellt.

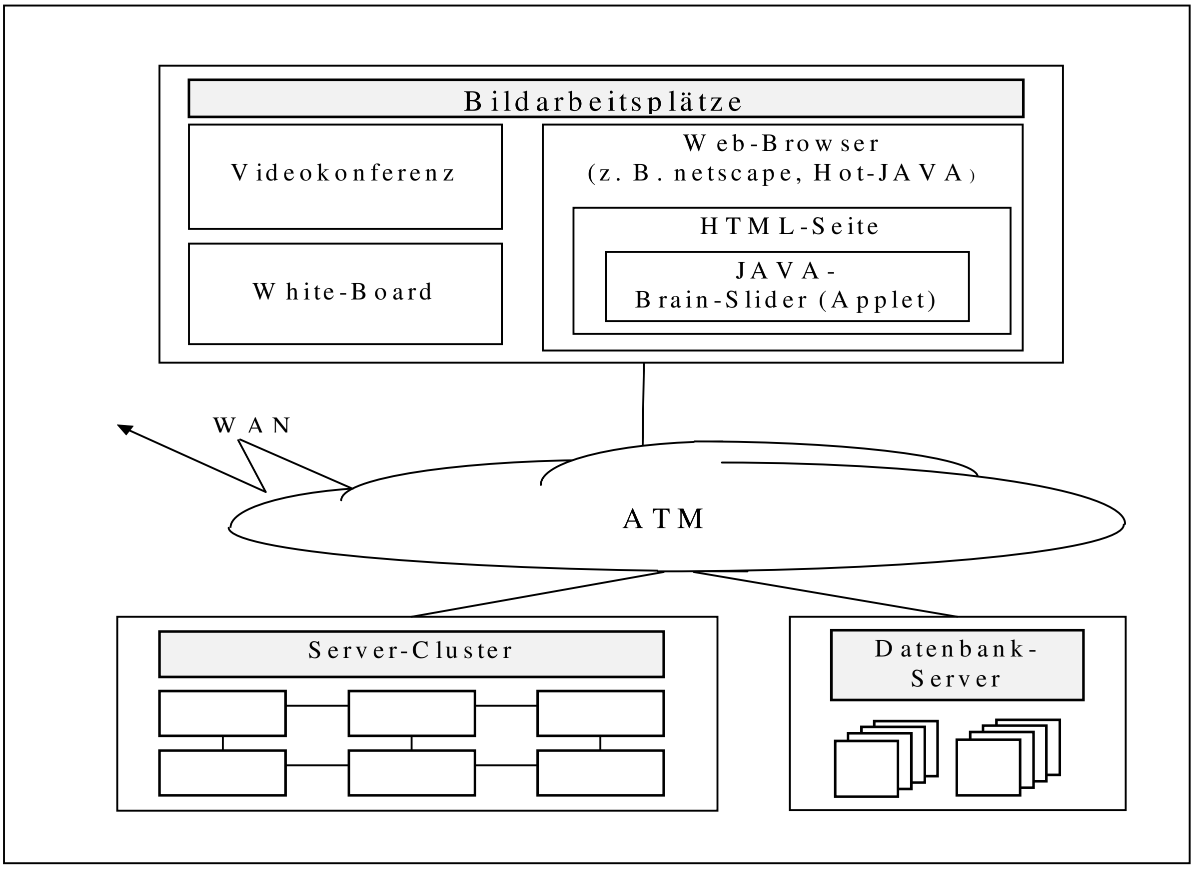

Abb. 1.: Infrastruktur von M-FIBRe

Es wird angestrebt, auf den Bildarbeitsplätzen eine integrale Oberfläche zu realisieren, welche die funktionalen Komponenten Videokonferenz, White-Board und Web-Browser beinhalten. Ruft ein Benutzer über den Web-Browser den JAVA Brain-Slider auf, so wird die Kommunikation über das Hypertext-Protokoll (http) [3] durchgeführt. Die Anfrage wird an einen Web-Server (CERN w3-Webserver) im Server-Cluster gestellt. Dem Client wird eine HTML-Seite (Hypertext Markup Language) übermittelt, in welcher sich die Referenz auf die JAVA-Applikation befindet. Der Client lädt die bytecodierte Applikation und führt diese lokal aus. Interaktionen des Benutzers werden an die Server weitergeleitet und die Schnittdaten werden zur Laufzeit an den Client übermittelt. Gleichzeitig soll der Benutzer in einem späteren Projektstadium über dieses System mit anderen Benutzern über die Videokonferenz- und die White-Board-Komponente kommunizieren und aktuell geladene Bilddaten diskutieren.

3 JAVA Brain-Slider

Der JAVA-Brain-Slider dient zur Navigation durch Bilddaten und visualisiert nach Eingabe des Benutzers Schnittebenen verschiedener Schnittführungen. Der Brain-Slider ist Bestandteil des Evaluationsteils des M-FIBRe-Projekts im Forschungszentrums Jülich. Die definierten Anforderungen des Projektes, insbesondere die netzwerkfähige Kommunikation der enthaltenen Komponenten, haben zu der Entscheidung geführt, den Brain-Slider als eine ernstzunehmende Alternative unter JAVA zu implementieren und für eine eventuelle spätere Produktionsumgebung zu untersuchen.

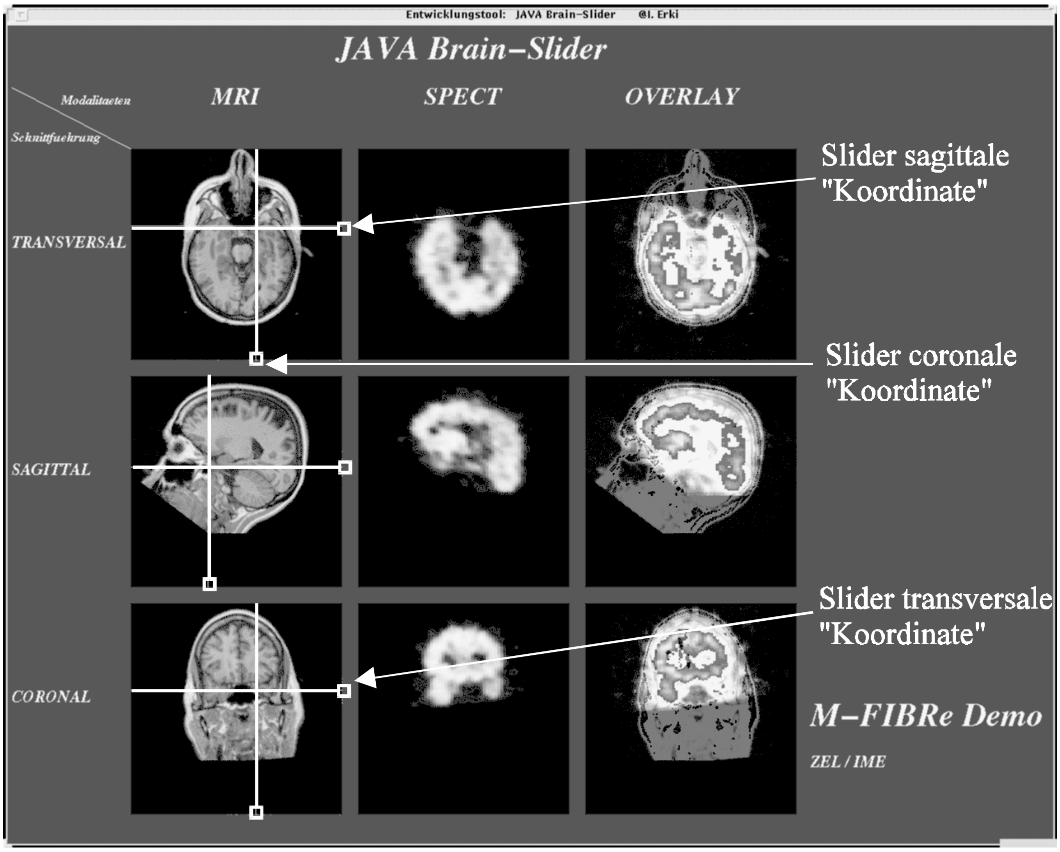

Abb. 2: Bildschirmmaske Brain-Slider

Die Applikation visualisiert Bilder verschiedener Modalitäten (hier: MRI, SPECT und PET), welche Gehirnstrukturen in den Schnittführungen transversal, sagittal und coronal abbilden können. Aus den Bildern der verwendeten Modalitäten werden zur Laufzeit Overlays generiert. In Abbildung 2 ist die Bildschirmmaske des Brain-Sliders dargestellt. In diesem Beispiel werden die Modalitäten MRI und SPECT in verschiedenen Perspektiven dargestellt. Rechts von den Modalitäten werden die Modalitäten durch einen Image-Fusion-Algorithmus zu einem Overlay zusammengeführt und die Intensitätswerte der Voxel mit einer vordefinierten Farbtabelle kodiert. Über der ersten Modalität sind in den verfügbaren Perspektiven mehrere Schieber sichtbar. Wird beispielsweise der Schieber in der transversalen Ansicht in x-Richtung gezogen, so verschiebt sich die relative Koordinate der coronalen Ansicht. Durch diese Funktionalität kann der Benutzer jeden Punkt im Raum aus 3 Ansichten betrachten. Die zur Laufzeit dargestellten Schnittebenen der Modalitäten und des Overlays entsprechen gerade den Sliderpositionen.

3.1 Betrachtungen zum Laufzeitverhalten

Die Verwendung einer objektorientierten Programmiersprache unterstützt den modularen Charakter der Applikation. Sämtliche Funktionalitäten des vorgestellten Systems können als Module in die vorhandene Oberfläche integriert werden, was insbesondere für die weitere Entwicklung des Systems von Bedeutung ist. Die Pflege und Administration des Systems wird hierdurch erleichtert. Die Benutzerfreundlichkeit der Applikation sollte durch angemessenes Laufzeitverhalten und die eindeutige Verwendung der angebotenen Funktionen unterstützt werden. Zu diesem Zweck wird eine allgemeingültige Abschätzung des Laufzeitverhaltens des Brain-Sliders vorgestellt. Die Funktionalität der derzeitigen Implementierung des JAVA Brain-Sliders erfordert aufgrund der großen Bilddatenmengen ein einmaliges Laden der Volumendaten in den Hauptspeicher, da sämtliche Schnittebenen zur Laufzeit aus den bytekodierten Daten generiert werden. Wird die Ladeoperation der gesamten Daten in den Hauptspeicher nicht bei Initialisierung durchgeführt, so kommt es zu Leistungseinbußen durch wiederholten Zugriff auf den Festplattenspeicher.

Derzeit ist die Funktionalität der Navigation durch Volumendaten verschiedener Modalitäten implementiert. Die Darstellung von n^2 Voxel der n' Schnittebenen (*slices*) für den Brain-Slider erfordert einen Zeitaufwand [5] der Größenordnung $O(n^3)$ für die Visualisierung aller Schnittebenen des Volumens. n' hängt hierbei unmittelbar vom Benutzerverhalten ab. Der Benutzer kann sämtliche n Schnittebenen aus allen Schnittführungen visualisieren, was jedoch aufgrund des Informationsgehalts der äußeren Schnitte nicht sinnvoll ist. Der resultierende Aufwand ergibt sich aus den tatsächlich visualisierten Schnittebenen $n' \leq n$. Es gilt für a = Anzahl der Modalitäten und b = Anzahl der Schnittführungen:

$$f_1(n) = \begin{cases} (a+1)\,b\,n^2\,n' = O(n^3) & ,\textit{für } a \geq 2 \textit{ mit Overlayansicht} \\ a\,b\,n^2\,n' \quad = O(n^3) & ,\textit{ohne Overlayansicht} \end{cases} \qquad \text{(Gleichung 1).}$$

3.1.1 Funktionale Fokussierung

Derzeit ist die Einbindung funktionaler Bilddarstellungen in der Implementierungsphase. Die Funktionalität des Brain-Sliders wird zunächst um die funktionale Darstellung über den zeitlichen Verlauf erweitert. Es werden studientypspezifisch Volumendatensätze einzelner Modalitäten über die Zeitachse dargestellt. Der Benutzer kann hierdurch die zeitliche Veränderung spezifischer Regionen des Gehirns darstellen. Limitierender Faktor dieser Funktionalität wird die entsprechende Zunahme der Datenmenge sein. Die Anzahl der zeitdiskreten Volumina einer Modalität und die Darstellung der gesamten Schnittebene könnte zu Restriktionen im Laufzeitverhalten der Applikation führen. Aus diesem Grund wird die funktionale Darstellung der Bilddaten um eine weitere Funktionalität erweitert, die *funktionale Fokussierung*. Mit der funktionalen Fokussierung ist die funktionale Darstellung eines Ausschnitts aus der Schnittebene gemeint, welche interaktiv durch den Benutzer zur Laufzeit definiert werden kann. Der mit dem Mauszeiger markierte Bereich aus einer Schnittebene wird in den oben genannten Perspektiven über die Zeit dargestellt. Sei t_i mit i = 0,..., T-1 das zeitdiskrete Intervall für die funktionale Darstellung einer Modalität und b = Anzahl der Schnittführungen, so folgt für den Zeitaufwand der Darstellung aller Schnittebenen mit Schnittlänge n:

$$f_2(n) = \quad b\, n^3\, T \; = \mathrm{O}(n^3) \qquad \text{(Gleichung 2).}$$

Wird die fokussierte Schnittlänge in Abhängigkeit von T gewählt, so kann bei geschickter Auswahl der Schnittlänge der reale Zeitaufwand der Darstellung in der gleichen Größenordnung liegen, wie die Navigationsfunktionalität des Brain-Sliders.
Sei $n = 2^m$ mit m = r + s, dann gilt für einen quadratischen Ausschnitt mit der Schnittlänge $p = 2^r$ für eine Bildreihenfolge mit $T \leq 2^{2s}$:

$$f_3(n) = \quad b\, p^2\, n\, T = b\, 2^{3r+s}\, T \leq f_1(n) \qquad \text{(Gleichung 3).}$$

Dies entspricht gerade dem gleichen realen Zeitaufwand, wie die Navigationsfunktionalität des Brain-Sliders.

4 Ausblick

Durch den modularen Charakter der Applikation können weitere Funktionalitäten in einem späteren Projektstadium ohne großen Aufwand in die Oberfläche eingebunden werden. Die Verwendung einer objektorientierten Programmiersprache erleichtert hierbei die Erstellung eines modularen Systems mit funktionalitätsspezifischen Objekten.
Neben der derzeitigen Implementierung der funktionalen Fokussierung, ist die Einbindung der White-Board-Funktionalität unter JAVA geplant. Als weitere funktionale Module können gängige 3D-Rekonstruktionsalgorithmen und 3D-Visualisierungsalgorithmen in späteren Projektstadien integriert werden. Eine weitere Möglichkeit der Funktionalitätserweiterung ist die vorhandene Schnittstelle JAVAs zu verschiedenen höheren Programmiersprachen,

insbesondere *C* bzw. *C++*. Vorhandene Bildverarbeitungsprogramme können in das Gesamtsystem integriert werden und auf den Servern (Abb. 1) plattformspezifisch ausgeführt werden. Ausgabedaten der Applikationen werden in den Bytecode konvertiert und über das zugrundeliegende Netz an den aufrufenden Client übertragen. Von besonderem Interesse werden Untersuchungen bestimmter laufzeitintensiver Funktionen sein, um die interpretative Programmiersprache JAVA mit konventionellen, imperativen Programmiersprachen zu vergleichen. Ein entsprechender Laufzeitgewinn bestimmter Funktionen würde durch den Verlust der Plattformunabhängigkeit bezahlt werden.

Literatur

[1] Flanagan: JAVA in a Nutshell, O'Reilly & Associates, März 1996

[2] Damesh: Teach Yourself JAVA in 21 Days, Sams & Co. Inc., 1996

[3] December: JAVA - Einführung und Überblick, Markt & Technik/ Buch- und Software-Verlag GmbH, 1996

[4] Gosling, Mc Gilton: The JAVA languuage Environment - A White Paper, Sun Microsystems Inc., 1995

[5] Bronstein, Semendjajew: Taschenbuch der Mathematik, 24 Auflage, Verlag Harri Deutsch - Thun, Frankfurt/M., 1989