筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

海底コアCTスキャンイメージ可視化のための クラウドサービスの開発 －DICOMスライス画像のための 3次元レンダリングサーバソフトウェア－

坂本 侑一郎

修士（工学）

（コンピュータサイエンス専攻）

指導教員　田中 二郎

2013年3月

概要

本プロジェクトは，筑波大学の高度 IT 人材育成のための実践的ソフトウェア開発専修プログラムにおける特定課題研究として，筆者を含む学生 3 人のチームにより，独立行政法人海洋研究開発機構（JAMSTEC）との共同研究のもと,「海底コア CT スキャンイメージ可視化のためのクラウドサービス」の開発を行うものである．JAMSTEC の所有する地球深部探査船「ちきゅう」により採取される掘削コア試料は，船上に搭載されている X 線 CT スキャナにより，DICOM フォーマットのデジタルデータとして保存される．このデジタルデータを用いれば，実物のコアが手元になくとも，コアの観察，内部構造の調査が可能となる．しかしながら，このデータの扱いにはいくつかの問題点がある．それは，データサイズが大きいため，データのダウンロード及び 3 次元描画が困難であることと，掘削コア閲覧のための専用ソフトウェアが存在しないことである．本プロジェクトにおいて，コアの CT スキャンイメージの保存と描画を担うリモートサーバを構築し，掘削コア試料の分析に適したユーザインタフェースを有した閲覧ソフトウェアを開発することにより，これらの問題を解決する．本論文は，そのシステムの内，DICOM スライス画像のための 3 次元レンダリングサーバソフトウェアに関して述べている．本プロジェクトの貢献により，海底コア CT スキャンイメージを，端末性能に依存せずかつ掘削コア試料閲覧のための専用インタフェースを利用してコアの分析を行うことができる．我々の開発するシステムを，地球科学の研究者をはじめとする多くの人々が利用することにより，地球科学研究の発展や理解が促進されるものと期待できる．

目 次

図 目 次

第1章　はじめに

本プロジェクトは，筑波大学の高度 IT 人材育成のための実践的ソフトウェア開発専修プログラムにおける特定課題研究として，筆者を含む学生 3 人のチームにより，独立行政法人海洋研究開発機構（JAMSTEC）高知コア研究所との共同研究のもと,「海底コア CT スキャンイメージ可視化のためのクラウドサービス」の開発を行うものである．

JAMSTEC は，国際的な海洋掘削プロジェクトの IODP[1] の一機関であり，その主力船である地球深部探査船「ちきゅう」を保有している．JAMSTEC は，この「ちきゅう」による科学航海により，これまで人類未踏であるマントルの掘削という壮大な科学目標を掲げている [4]．掘削船により掘削された柱状の試料は「掘削コア試料」と呼ばれ，研究者はこの掘削コア試料を分析することにより，震源物質調査や地球環境の変化，地下生物圏の解明などの研究をしている [4].

「ちきゅう」船上にはこの掘削コア試料を分析するための様々な設備が搭載されており，他の掘削船にない特徴的な設備として，X 線 CT スキャナがある．掘削されたコアはすぐにこの X 線 CT スキャナによりデジタルデータ化され，それを用いれば，デジタルデータとしてコアの観察が可能となる．しかしながらそのデジタルデータは，様々な理由から扱いが難しい現状がある．主な問題点として，掘削コア 1.5 メートル当たりのデータサイズが数 400MByte あり，さらにその本数が数千にのぼること，そうした大きなデータサイズの画像を 3 次元描画するためにはハイスペックな PC が必要であること，閲覧するための適切な専用ソフトウェアが存在しないことが挙げられる．

そこで本プロジェクトでは，前述の問題を解決するために,「海底コア CT スキャンイメージ可視化のためのクラウドサービス」の開発を行う.「海底コア CT スキャンイメージ可視化のためのクラウドサービス」は，海底掘削コア試料の CT スキャンデータを画像処理して 2 次元画像を生成するレンダリングサーバと，携帯情報端末や PC でその画像を閲覧するためのクライアントソフトウェアである *Virtual Core Viewer* から成る．レンダリングサーバにコアデータを配置し，クライアントからのリクエストに応じて画像を生成することにより，データの大きさと描画の負担の問題を解決する．Virtual Core Viewer は，掘削コア試料を閲覧するための各種機能を備え，これまでに存在しなかった未踏の掘削コアを閲覧するための専用ソフトウェアインタフェースを提供し，コア解析のための新しい手段を提供する．

本プロジェクトの貢献は，Android OS 上で動作する専用アプリケーションである Virtual Core Viewer を Google Play で公開し，また同様の機能を持つ Web アプリケーションを高知コア研究所の運用する Virtual Core Library で公開したことである．我々の開発するシステムを，地球科学の研究者をはじめとする多くの人々が利用することにより，地球科学研究の発展や

理解が促進されるものと期待できる．

本プロジェクトは，以下のメンバーで行った．

- 坂本侑一郎（レンダリングサーバ開発）
- 岡本昂也（ミドルウェア開発）
- 佐々木慎（アプリケーション開発）
- 山際伸一准教授（プロジェクト担当教員）
- 和田耕一教授 （プロジェクト担当教員）
- 久光敏夫技術主任（JAMSTEC 高知コア研究所 共同研究者）

開発の期間は 2012 年 5 月から 2013 年 1 月までであり，その詳しい開発スケジュールは付録 A に記載してある．我々は開発内容を 3 つの段階に分けて段階的に機能を追加し，その都度，久光技術主任を通じて地質学者の方々に利用していただき，本システムのフィードバックを得た．そのフィードバックを基にユーザインタフェースの改良や機能追加を重ねた．

本プロジェクトにおいて，筆者はレンダリングサーバの開発を担当した．具体的には，レンダリングエンジンの設計・開発と，それをクライアントアプリケーションから呼び出すための API を設計した．

本報告書は，全 5 章で構成されている．第 2 章では本プロジェクトの背景について述べる．第 3 章では要求定義とシステムの全体設計について述べる．第 4 章では筆者が担当した DICOM スライス画像のための 3 次元レンダリングサーバソフトウェアについて述べる．第 5 章では本プロジェクトのまとめと今後の発展について述べる．付録には，本プロジェクトの進行計画や，開発・運用に関するドキュメントを掲載する．

第2章　本研究の背景とあらまし

本章では，本研究の背景となる，海洋掘削プロジェクトの現状や関係する技術について述べる．

2.1　海洋掘削プロジェクトと掘削コア試料

東日本大震災のような巨大地震の断層調査や，地球の歴史を探る手段の一つとして「掘削コア試料」と呼ばれる，海底から掘削された柱状の地層サンプルが利用され，世界の研究者が掘削コア試料を使って様々な研究と調査を行っている．国際的な海洋科学掘削プロジェクトとして，日本と米国が主導する IODP がある [1]．IODP は，地球環境変動，地球内部構造及び地殻内生物圏等の解明を目的とした国際的な海洋科学掘削計画である．掘削船は米国の JOIDES Resolution 号と日本の地球深部探査船「ちきゅう」による科学研究航海が実施されている [2].「ちきゅう」は，船上にはこの掘削コア試料を分析するための様々な設備が搭載されており，他の掘削船にない「ちきゅう」の特徴的な設備として，X 線 CT スキャナがある．

「ちきゅう」により掘削されたコア試料は，図 2.1 の流れで扱われる．(1）掘削されたコアを 1.5 メートルずつ（これを 1 セクションと呼ぶ）に切断し，セクションごとに航海番号やセクション番号などの ID を割り振り，データベースに登録して管理する．また，船上に搭載されている X 線 CT スキャナにより掘削コアをスキャンし，デジタルデータを得る．掘削コア試料をデジタルデータ化することにより，採取したばかりのコア試料の物性や構造を半永久的に保存することが可能となる．(2）コアの実物は高知コア研究所において保管・管理される．コアを用いて研究を行うためには，利用したい掘削コアを選定し，コアが保存されている高知コア研究所へ利用申請する必要がある．(3）一方，デジタルデータは同研究所の Virtual Core Library[3] を介して公開されており，掘削から 1 年経過したコアのデータを自由にダウンロードできる．Virtual Core Library では，掘削コアの CT スキャン画像が 1 セクション毎にまとめられた zip ファイルをダウンロードできる．CT スキャンによる掘削コアの断面画像の 1 枚 1 枚は，CT の標準画像フォーマットである DICOM フォーマットで記録される．地質学者は，DICOM フォーマットの画像を読み込むことのできるソフトウェアを用いて，掘削コアの 3 次元画像を閲覧することが可能となる．

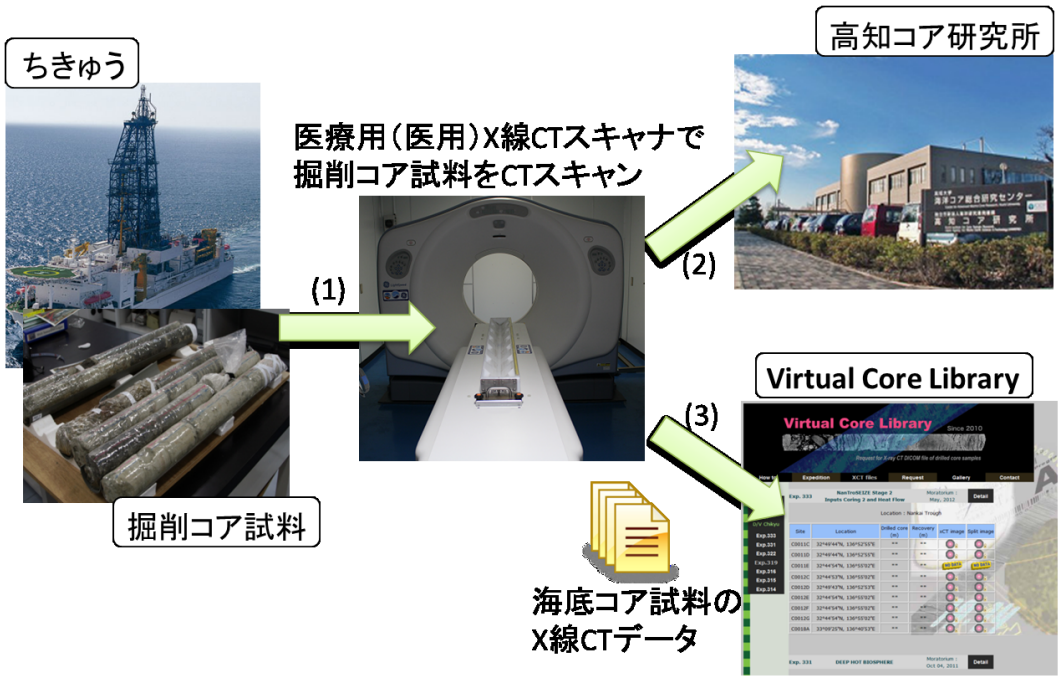

図 2.1: 掘削コア試料の保存

2.2　X-CT と DICOM フォーマット

X-CT は，物体へ X 線を照射してその吸収度合いを測定することにより，物体の断面画像を得る技術である．この断面画像のことをスライスと呼び，この画像が 1 枚の DICOM フォーマットの画像ファイルとして保存される．CT スキャンされた画像のピクセル値に対応するものは，CT 値と呼ばれる．CT 値は空気をマイナス 1000HU，水を 0HU と定義した HU（Hounsfield Unit）という単位が利用され，ほかの物質はこれらとの相対値により表される．元素番号が高く密度が高いほど高い CT 値を示し，逆に元素番号が低く密度が低いほど低い CT 値を示す．

また，DICOM（Digital Imaging and COmmunication in Medicine）は，米国放射線学会 (ACR) と北米電子機器工業会 (NEMA) が開発した，CT や MRI，CR などで撮影した医用画像のフォーマットと，それらの画像を扱う医用画像機器間の通信プロトコルを定義した標準規格のことである [5]．DICOM ファイルの内容は，スライス画像のピクセルデータと患者の名前や生年月日等のメタデータから構成されている．DICOM フォーマットの画像を閲覧するためのソフトウェアとして，OsiriX 等が挙げられる．これらのソフトウェアでは，2 次元（スライス）画像に対する画像処理や，連続したスライス画像から 3 次元画像を構成することなどが可能となる．

DICOM フォーマットの画像ファイルから 3 次元イメージを再構成する手順を図 2.2 に示す．連続したスライス画像を積み重ねることにより，3 次元モデルを作り，さらに CT 値と色の組み合わせを決めることで，3 次元イメージを再構成する．

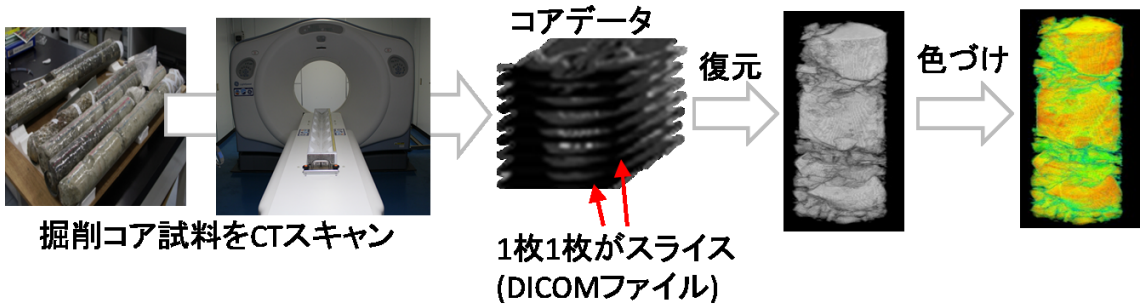

図 2.2: 掘削コア試料の CT イメージの 3 次元描画

2.3 クラウドコンピューティング

従来はユーザのローカル環境で利用・管理を行っていたデータやソフトウェアを，リモートのサーバで管理し，インターネットを介してユーザがそれを利用するコンピュータの利用形態を，クラウドコンピューティングと呼ぶ [6]．クラウドコンピューティングについて明確な定義は存在しないが，何らかのコンピュータリソースを，ネットワークを介して利用するコンピュータの利用形態のことを総称してこのように呼ばれる．

クラウドサービスには，有償や無償，ソフトウェアやハードウェア等，色々な種類が存在している．例えば，Google の Google Apps はメールやオフィススイート等のソフトウェアを無償または有償で提供し，Amazon の Amazon Elastic Compute Cloud は仮想化サーバを無償または有償で提供している [7][8].

ユーザの利点として，パーソナルコンピュータや携帯情報端末等のクライアントとそれらの上で動作する Web ブラウザとインターネットに接続するための通信回線を用意すれば良く，開発や管理・保守を，ユーザ自身がする必要がなくなる．ゆえに，サービスを受ける者は外出先でも自宅でもどこに居ても，好きなタイミングでサービスを受けることができるようになる．

2.4 議論

海洋地質学の研究は地球の歴史，地球のシステムを解き明かす重要なものであり，我々人類の今後の発展に深くかかわっている．「ちきゅう」の科学掘削航海により得られる掘削コア試料は日々増え続けており，掘削コアの CT スキャンデータも同様に増え続けていく．情報技術の進歩に伴い，CT スキャンデータの活用が今後ますます見込まれる．掘削コア試料の CT スキャンデータを自由に活用して分析することは，実物のコア試料を用いた分析の効率や効果を高めることができる．そのため，掘削コア試料を用いた研究には，掘削コア試料の CT スキャンデータの活用が非常に重要となる．

しかしながら，掘削コア試料の CT スキャンデータの活用は十分なされていない現状がある．例えば，以下の状況を想定してみる．

- 船上にて，今掘削されたコアと過去に掘削されたコアをすぐ比較したい．
- 学会や会議等の複数人数が集う場で，掘削コアを見ながら議論がしたい．

これらのことを実行しようと思うと，以下の 3 点が主な問題となる．

1) データサイズ

掘削コア試料は 1 本（1 セクションとも言う）は 1.5m に区切られており，1 セクションあたりのデータサイズは大きいもので約 400MByte ある．コアデータを扱うためには PC にダウンロードする必要があるが，そのデータサイズゆえに通信の負担が大きい．また，現在 Virtual Core Library において公開されているコアデータは，約 4000 セクション分に上る．複数本のコアデータを扱いたい場合，さらに通信の負担が増えることになり，またデータを保存するための PC のストレージの容量も確保しなければならない．

2) 端末の処理能力

前述のデータサイズのコアデータを 3 次元描画するためには高速なプロセッサと大容量のメモリが必要となる．OsiriX[11] において 800 スライスを超える画像を扱いたいときは，6GB 以上の RAM を搭載しているマシンが推奨されている．携帯情報端末の場合，メインメモリは最新のものでも 1GB であることから，1 セクション分のデータを扱うことも難しいか，または不可能である．

3) 専用ソフトウェアが存在しない

コアデータは医用画像の標準フォーマットである DICOM であるため，DICOM ファイルを読み取れる専用ソフトウェア（OsiriX 等）を用いれば 3 次元化が可能である．しかし，DICOM ビューワは医用画像の閲覧に特化したソフトウェアである．骨や筋肉，靭帯などの人体の組織に対応する CT 値は既に知られており，その CT 値に対応する色付けも容易に行うことができる．コアを分析する際にはユーザがコアの構造を表現する CT 値に対して適切な色付けを行い，可視化しなければならない．

大容量の DICOM ファイルをリモートサーバに保存する方法として，ネットワークに連結されたストレージを利用する研究がある [12]．しかし，このサービスは DICOM ファイルを提供するだけのサービスであるため，データサイズの縮小はできない．DICOM ファイルを直接閲覧する環境 (ソフトウェアや表示用機材) が存在しないため，問題点 2)3) が解決できない．リモートサーバで DICOM ファイル描画し，ローカル端末で閲覧する方法として，MPEG ビデオストリーミングを利用した研究がある [13]．この研究ではリモートサーバで視覚化した 3 次元モデルを，MPEG ビデオストリーミングを通してクライアントのモバイル端末に表示している．ストリーミングは，通信するデータサイズが大きく，ネットワーク帯域を多く利用するため，サービスの質が，ネットワークの帯域やレイテンシ等のネットワークの品質に大きく左右されてしまう．この研究では，1) データサイズにおける問題点が解決できない．

そこで，我々は 2 つのアプローチをとり，上で述べた 3 つの問題を解決する．

i) コアデータの画像処理を行うレンダリングサーバを構築する

ii) 掘削コア試料の観察，画像操作に適したユーザインタフェースを有した閲覧ソフトウェアを開発する

i）レンダリングサーバにおいて，コアデータの 3 次元再構成を行い，それを任意の平面で切り取ったコアの 2 次元の画像を生成する．ユーザは，描画結果を軽量な 2 次元画像として受け取るため，ユーザ自身の端末にコアデータを保存し描画をする必要がなくなる．3 次元描画が可能な高性能な端末を必要としないため，モバイル端末からもコア画像の閲覧が可能となる．ユーザは，インターネットを通じて，レンダリングサーバに対して，閲覧したいコアやコアに対する操作をリクエストして，コア画像を得る．これにより，ユーザは従来のようにローカルの端末に DICOM ファイルをダウンロードする必要がなくなり，1 つ目の問題点である 1) データサイズに関する問題を解消することができる．加えて，DICOM 形式のデータ描画処理がリモートのサーバで行われるため，ローカル端末での DICOM ビューワを使った描画処理が不要になり，端末性能に依存せずにコア画像を閲覧できる．これにより，2 つ目の問題点である 2) 端末に要求される処理能力を解消できる．3) 専用ソフトフェアが存在しない問題に関しては，地質学者からコア試料の分析に関する方法をヒアリングし，コア試料の分析に特化したインタフェースを提供することで問題解決を図る．

本プロジェクトで開発するシステムを導入する前後での利用イメージの違いを図 2.3 に示す．従来では，Virtual Core Library からコアデータをダウンロードし，高性能な端末により描画処理をしていたが，本システムではリモートサーバ上でコアデータを処理してコア画像を生成するため，コア画像の閲覧する端末の性能は問われない．また，大容量のコアデータをダウンロードするため，高速回線が必要だったが，クライアントは軽量化された 2 次元画像をダウンロードするため，低速な回線でも利用可能となる．閲覧用のインタフェースも，これまでは OsiriX 等の医用画像向けのものが利用されてきたが，我々のシステムにより，コア閲覧に特化したインタフェースにより，コアの内部構造の観察が可能となる．

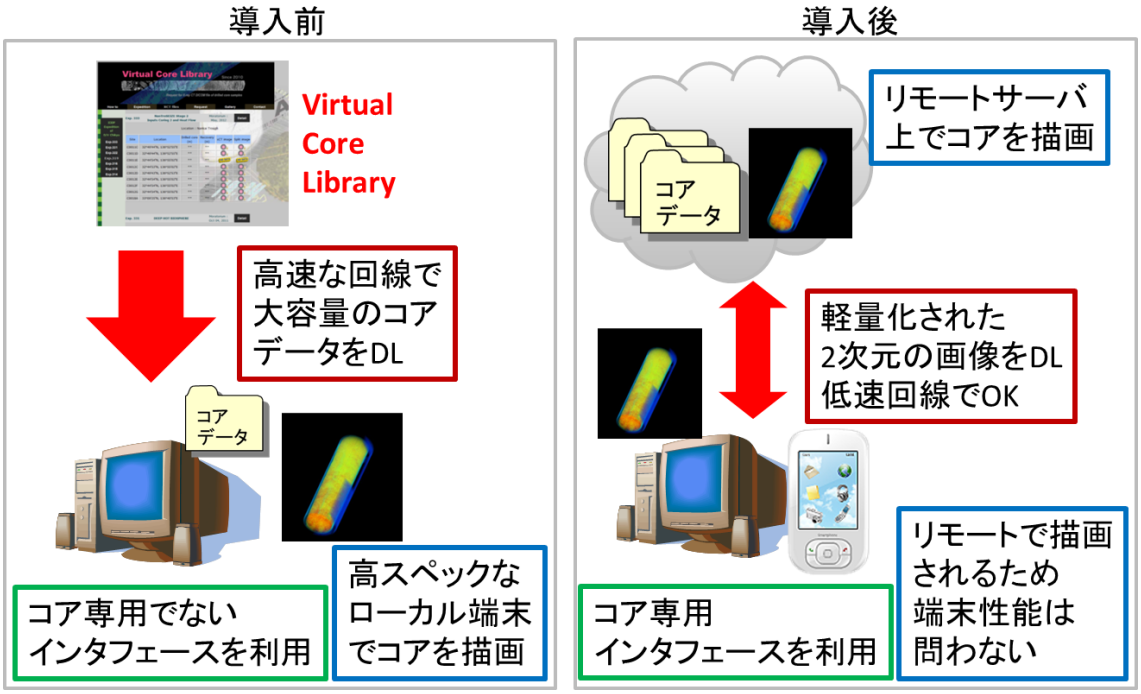

図 2.3: 本システムの導入前と後の利用イメージ

第3章　海底コアCTスキャンイメージ可視化のためのクラウドサービス

この章では，要件定義と利用シナリオについて述べ，それを実現するためのシステムの全体設計について述べる．

3.1　サービスの概要

前章において，デジタルデータ化された掘削コア試料の利用とその問題点について議論した．それらと地質学者へのヒアリングを基に，以下の要件を定義した．

1) 端末性能に依存せず，掘削コア試料の閲覧が可能であること

2) 掘削コア試料の分析に適したインタフェースであること

3) コアに対してインタラクティブな操作が可能であること

これら3つの要件から，タブレット等のモバイル機器上にてコアの閲覧を可能にするクライアントアプリケーション及びコアデータを処理するクラウドサービスを開発する．その全体像を図3.1に示す．(1) PCやタブレット等のクライアントから，レンダリングサーバへ画像描画をリクエストする．(2) レンダリングサーバは，コアデータをを2次元のコア画像に変換するために，まずはDICOMスライス画像を読み出し，3次元モデルの生成をする．(3) 次に，ユーザからのコア操作に関する操作を適用して2次元画像を生成する．(4) 最後に，クライアントアプリケーションに2次元画像を返すことにより，ユーザはコア画像を閲覧することができる．

クライアントでは，コアを分析するためのユーザインタフェースを有したアプリケーションにより，画像生成のリクエストを行う．コアの分析のために，CT値に対応する色を自由に選択できる機能の他に，コアに対する操作として，拡大，縮小，回転，移動，切断機能を実装する．クライアントからの操作に応じて，その操作に対応する画像を，レンダリングサーバにて生成する．具体的には，次のような処理がある．1. クライアントからのリクエストを処理してレンダリングエンジンの各ノードに画像処理を割り振る．2. リクエストされたコアのDICOMファイルからCT値を読み込む．3.CT値に色を付け，3次元画像を構成し，任意の平面での2次元画像を生成する．4. 生成した2次元画像をクライアントに返す．以上のことにより，携帯端末に対する負荷を軽減し，ユーザからの要望を解決する．

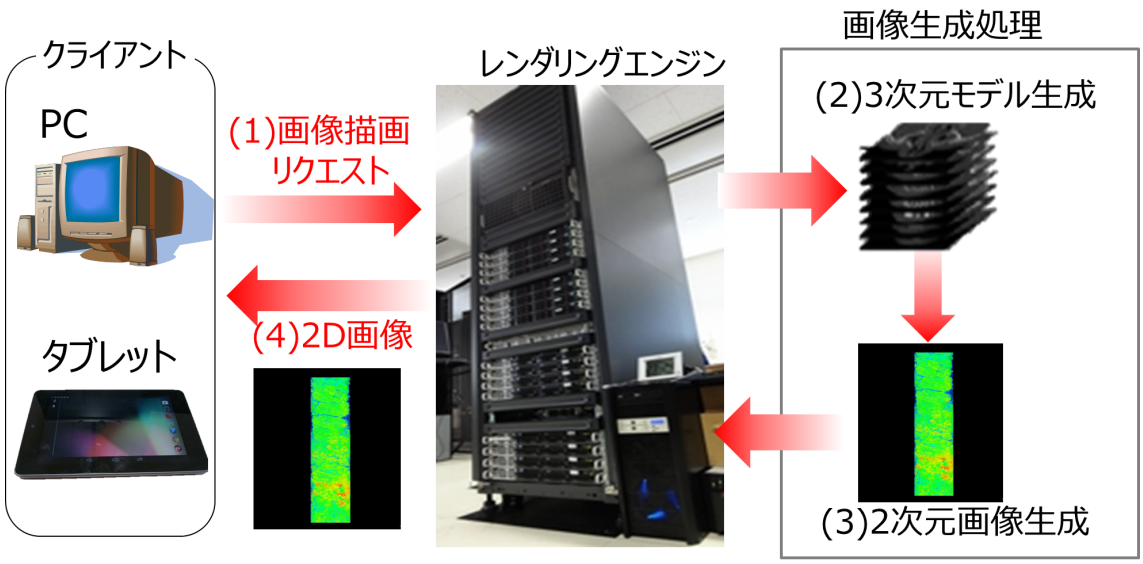

図 3.1: システム概要

利用シーンとして，デジタル化されたコアデータと実物のコアを目視しながらちきゅう船上で比較検討を行えること，携帯端末を用いてコアデータの操作・分析を行えることを主眼とする．また，利用者は地質学者だけでなく，様々な分野の研究者，教育担当者や学生，報道関係者，イベント企画者等をユーザとして，掘削コア試料の展示，報道，教育，普及活動，遊び等を目的とした利用も考慮する．

3.2 提供する機能

提供する主な機能を，1) コア試料の検索，2) コア試料の分析，3)CT 値に対する色づけとした．

1) コア試料の検索

現在公開されているコアセクションの数は 4000 にものぼる．掘削されたコアは，1.5 メートルずつに区切られ，それが 1 コアセクションと定義されている．1 つのコアセクションには，航海番号，掘削サイト番号，ホール番号，コア番号，ビットタイプ，セクション番号の 6 つの要素からなる識別番号が割り当てられている．地質学者は，これらの識別番号にもとづいて，閲覧したいコア試料を探し出す．しかし，現在閲覧可能なコア試料は 4000 セクション以上あるため，自身でひとつひとつ確認しながらコア試料を探し出すのは手間がかかる．本サービスを提供するにあたって，できるだけ簡単にコア試料を選択できるような検索機能を提供する．検索機能では，航海プロジェクト，掘削された場所などの情報を入力することで，ユーザが閲覧したいコア情報を探し出す．

2) コア試料の分析

掘削コアは，海底から採取された地層サンプルであり，地層は鉛直方向に重なったものである．ゆえに，掘削コア試料の断面を観察することにより様々な情報を得ることができる．本サービスにおいても，コアの断面を観察するための，縦方向切断を機能として提供する．閲覧のための基本的な機能として，回転と拡大・縮小機能も提供する．コアの全体像や細部表示，任意の視点からの閲覧を可能とする．コアは細長いため，視点の上下移動も行えるようにする．

3) CT 値に対する色づけ

コアデータには，水や筒といった分析に不要な物体が写りこんでいる．表示する CT 値幅を設定することで，分析に関係のない物体を取り除くことができる．これにより，可能な限りコアの分析に関係のないものを取り除くことが可能となる．表示する CT 値幅を設定しただけでは，コアを分析するには不十分である．設定した CT 値に対して色づけがされていなければ，物体の違いや構造が画像として表現されない．そこで，コアの分析に適した CT 値に対して色づけ・透過設定ができるインタフェースを提供する．コアデータを閲覧するのに適した CT 値幅を憶測で設定するのは，非常に困難である．CT 値とその発生頻度を CT 値ヒストグラムとしてユーザに表示することで，CT 値幅の設定を容易にする．

3.3 システムの全体設計

本サービスを実現するために，1) クライアントアプリケーション，2) ゲートウェイ，3) レンダリングエンジンから成るシステムを設計した．その構成を図 3.2 に示す．

1) クライアントアプリケーション

クライアントアプリケーションは，コアの分析に必要な機能を搭載したユーザインタフェースを提供する．タブレット端末からの利用を想定した Android アプリケーションと，PC からの利用を想定した Web アプリケーションの 2 つをクライアントアプリケーションとして提供する．ユーザからの操作に応じて，ゲートウェイが有する各モジュールにリクエストを送信し，コア画像や各種情報を取得する．

2) ゲートウェイ

クライアントアプリケーションからの要求を初めに受け付けるサーバである．その中に，コアの情報を取得するためのコア検索モジュールと，コアの画像を取得するためのレンダラ制御モジュールを有する．コア検索モジュールは，クライアントからのクエリに一致するコア情報を検索用 DB から選択する．レンダラ制御モジュールはクライアントから，選択されたコア番号と，コア画像に対する操作の情報を受け取り，レンダリングエンジンにあるレンダラに処理を依頼する．ただし，レンダラに対する依頼はロードバランサを介して行われる．ロードバランサは，負荷の少ないノードを選択し，レンダラにコア画像生成処理を依頼する．

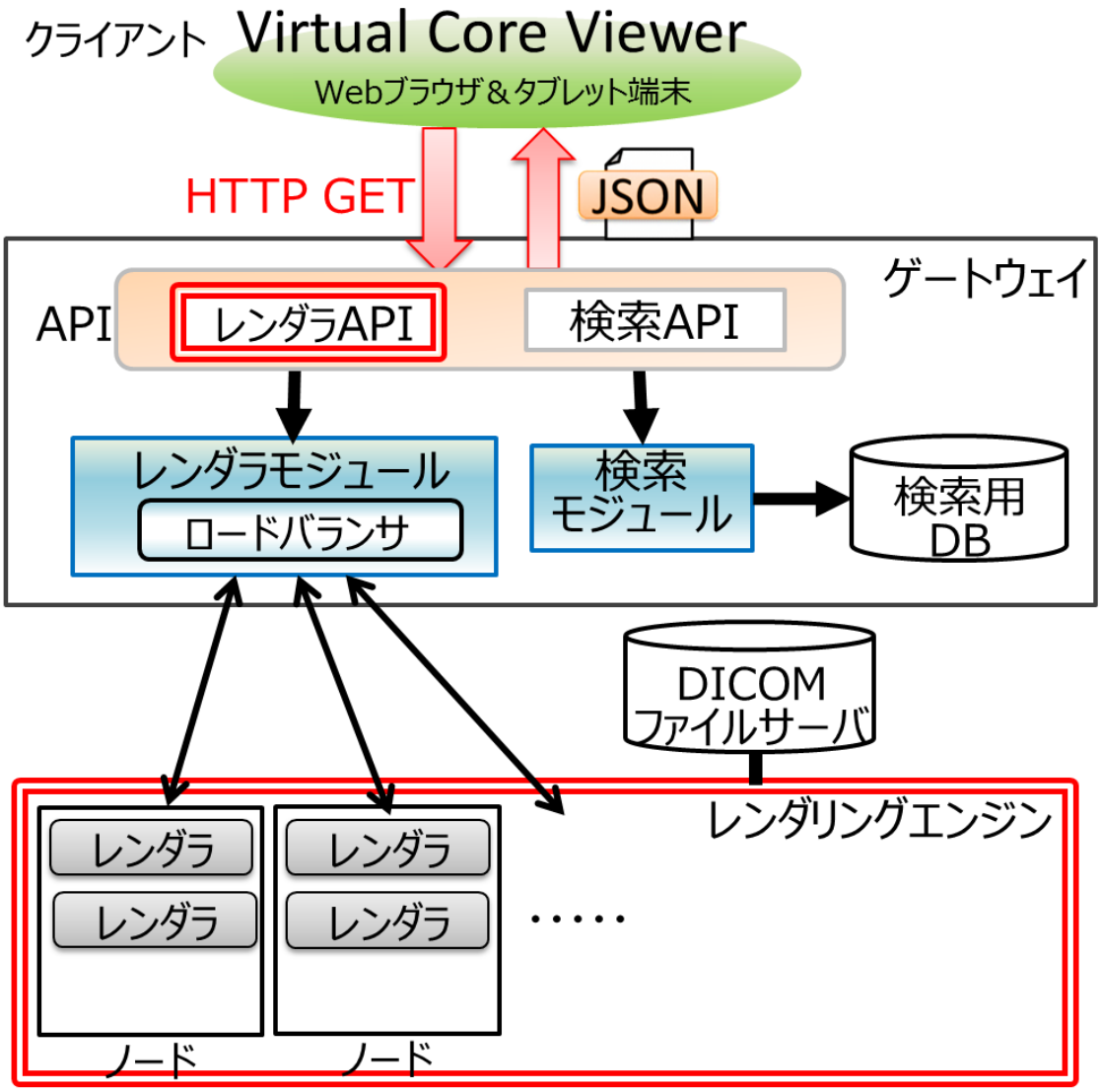

図 3.2: システムの全体構成

3) レンダリングエンジン

レンダリングエンジンは，画像生成を行うレンダラが稼働するノード群である．レンダラは，受信したリクエストにもとづいてコアデータの画像を生成する．画像生成に用いるコアデータはすべて DICOM ファイルサーバに格納されている．適宜，DICOM ファイルサーバからコアデータを読み込むことで，画像を生成する．

設計したサービスにおけるコアデータ閲覧までの流れは次のとおりである．コアデータを閲覧するために，まずコアデータの情報を取得する．コアデータの情報は, クライアントアプ

リケーションから, 検索 API を介して検索モジュールにリクエストを送信することで取得できる．検索モジュールでは, 受信したリクエストをもとに検索用 DB にアクセスし, 閲覧したいコアデータの情報を特定し, クライアントアプリケーションに返す．次に, 取得したコアデータの情報を用いて, レンダラ API を介してレンダラモジュールにリクエストを送信する．レンダラモジュールは, 受信したリクエストに基づいて，レンダリングエンジンのレンダラに画像生成処理を依頼する．画像生成処理が終了した後に, クライアントアプリケーションにコアデータの画像が表示される．クライアントアプリケーションからコア分析のための操作を実行した場合, コアデータの情報とコアに対する操作情報をリクエストとして送信することで, 操作情報が反映されたコア画像が生成・表示される．レンダラへの画像生成処理の依頼はロードバランサを介して行い, 画像生成処理によりかかる負荷を可能な限り分散する．

3.4　筆者の担当

筆者の担当は，レンダリングエンジンの設計・実装と，クライアントからレンダリングエンジンの機能を利用するためのレンダラ API の設計である．これは，図 3.2 中の赤色の二重線で囲われた部分に該当する．

レンダラ API は，クライアントアプリケーションからレンダラへ指示を出し，目的となる画像や情報を取りだすためのアプリケーションプログラミングインタフェースである．クライアントアプリケーションは，Android アプリケーションと Web アプリケーションとして実装されるため，その両方にとって利用しやすい API を作成する．また，ゲートウェイ－レンダリングエンジン間のデータの受け渡し方法も定義する．

レンダリングエンジンでは，提供すべき機能を実現し，クライアントアプリケーションが要求するコアの 2 次元画像を生成する．

リモートのレンダリングサーバを利用したボリュームデータの閲覧に関する研究がある [14][15]．これらと違い，本サービスでは掘削コア試料の閲覧を対象としていることと，低速な回線でも利用可能であることを念頭に，システムを構築していく．

詳細は，次章にて述べる．

第4章　DICOMスライス画像のための3次元レンダリングサーバソフトウェア

本章では，クライアントアプリケーション－レンダラ間の通信プロトコルの設計と，レンダリングエンジンの設計・実装について述べる．

4.1　クライアントアプリケーション－レンダラ間の通信プロトコル

クライアントアプリケーションからレンダラへ指令を与えて2次元画像を得るために，まずはクライアントアプリケーション－レンダラ間の通信プロトコルの設計を行った．前章において，本サービスのクライアントアプリケーションは，Webブラウザと Androidアプリケーションにより提供すると定めた．それぞれのクライアントを開発するプログラミング言語，動作するプラットフォームに依存しない汎用的な通信方法として，HTTPを用いる Web APIを作成した．HTTP GETを利用した Web APIは，Googleや Yahooが公開する検索 APIや Twitterなど，昨今の Webサービスにおいてしばし用いられる．取得したい情報は，パラメータとその値の組の羅列として表現される．リクエストを受け取ったレンダラモジュールは，レンダリングエンジンを制御して目的の画像や情報を取得し，クライアントへ JSON(JaveScript Object Notation)[9] 形式のデータとして送信する．JSONは，JavaScriptにおけるオブジェクトの表記法をベースとしたデータ記述言語である．JSONを採用した理由は以下の2点である．

- XMLと比較して記述が簡潔であり，データ量が少ない
- Webアプリケーション開発によく用いられる JavaScriptでの扱いが容易

JSONと XMLの表記の違いを図 4.1に示す．例えば，“id=1” を表現したいときは，JSONは “id:1”，XMLは “<id> 1</id> ” となる．XMLの場合は，終了タグが必要であることと，“format” や “formats” のような値を囲うためだけのタグが必要となる．一方 JSONは属性と値を “ {”，“} ” で囲うため，XMLと比較して簡潔な表現となる．また，JSONは，JavaScriptだけでなく Javaや PHPなど様々な言語で利用可能である．クライアントアプリケーションの開発者は，Web APIを用いて情報を取得し，その情報を利用して，目的のアプリケーションを開発する．

機能要件から，やり取りすべきパラメータを抽出し，リクエストパラメータを表 4.1に，レスポンスを表 4.2のように定めた．不正なリクエストパラメータを受信した場合には，不正理由を返す．

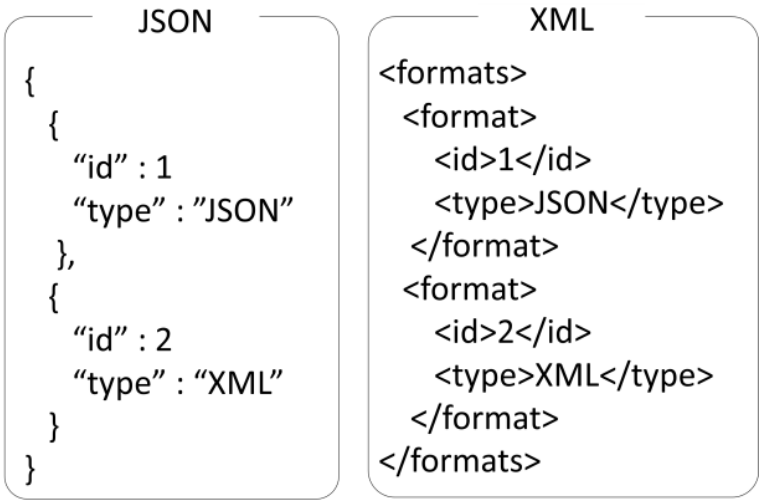

図 4.1: JSON（左）と XML（右）の比較

クライアントから描画パラメータを受け取ったレンダラモジュールは，ソケット通信により，描画パラメータをレンダリングエンジンに送信する．その際は，各パラメータを “\\” をデリミタとして区切る．描画パラメータとは別に，レンダリングエンジンが生成する画像を保存する場所の絶対パスも指定する．

4.2 レンダリングエンジン

レンダリングエンジンの各ノードのスペックは CPU：Xeon E5645 2.40GHz × 2，メインメモリ 12GB であり，計 16 台稼働している．コアデータの 3 次元描画には，OpenGL を用いる．レンダリングエンジン用のマシンにはディスプレイが接続されていないため，仮想フレームバッファを描画結果の出力先として指定する．レンダリングエンジンの設計書と起動に必要なソフトウェアや起動オプションは付録 B，付録 C，付録 D に記載する．レンダリングエンジンは，以下の手順により 2 次元画像を生成する．

1) 初期化処理

2) 描画パラメータの受信

3) 描画処理

4) 描画結果の送信

それぞれの項目について，以下に詳細を述べる．

表 4.1: レンダラ API のパラメータと取りうる値

パラメータ	意味と値
corenum	検索 API で調べたコア番号．0 以上
zoom	拡大率．0.0-
depth	コアの上下移動．符号付整数
rot_ type	回転方法
rotx, rotz	前後回転，縦軸回転．0-360
resolution	解像度，5 段階
color_ type	ヒートマップかモノクロか
cf	色関数．CT 値と色値の組の配列
af	透過関数．CT 値と透過値の組の配列
eye_ x,eye_ y,eye_ z	現在のカメラ座標
up_ x,up_ y,up_ z	現在のカメラの上方向ベクトル
degree	回転角度
mouse_ x, mouse_ y	マウスの移動ベクトル

表 4.2: レンダラ API の返り値

送信するパラメータ	返り値
img_ url	生成された画像の URL
histgram	CT 値のヒストグラムの配列
scale	1 ピクセルあたりの長さ
eye_ x,eye_ y,eye_ z	現在のカメラ座標
up_ x,up_ y,up_ z	現在のカメラの上方向ベクトル

4.2.1 初期化処理

レンダラは，起動時に引数として与えられたパスに保存されている複数の DICOM ファイルの読み出しを，GDCM（Grassroots DICOM library）[10] を用いて行う．DICOM に格納されているデータには 2 種類あり，画像データとその画像に関するメタデータである．画像生成に用いるためのメタデータを表 4.3 に示す．I_{NUM} は，その DICOM ファイルが何番目のスライスに当たるかを意味している．この値を基にスライスをソートし，ボリュームデータを作成する．読み込まれたボリュームデータはレンダラが終了するまでそのプロセスが保持する．そうすることにより，描画リクエストが来てからすぐに描画を開始できる．

表 4.3: レンダラが利用する DICOM ファイルのメタデータ

パラメータ	単位	DICOM 中のタグ名
I_{NUM}	枚目	Image Number
S_W	mm	Pixel Spacing
S_H	mm	Pixel Spacing
T	mm	Slice Thickness
P_W	ピクセル	width
P_H	ピクセル	height

4.2.2 描画パラメータの受信

レンダラは，視点移動，画像の品質，画像保存先に関する情報を元に描画を開始する．それまでは，起動時に指定されたポート番号をソケットに割り当てて通信相手からの接続を待ち受ける．

4.2.3 描画処理

描画処理では，(a) 視点の決定，(b) ボリュームデータの圧縮，(c) CT 値に対する色の設定，(d) CT 値に対する α 値の設定，(e) 縦方向切断の各処理を適用し，2 次元画像を生成する，描画のイメージを図 4.2 に示す．まずは，初期化時に読み込んだボリュームデータを 3D テクスチャデータに変換する．それを，複数枚重ねた長方形ポリゴンに対してテクスチャマッピングをすることにより，ボリュームレンダリングを実現する．表 4.3 に示すメタデータを利用して，長方形ポリゴンの縦横及び間隔を計算し，実物の掘削コアの縦・横・高さの比を保つ．長方形ポリゴンの辺の長さは，ピクセル数とピクセル間隔を用いて，幅 P_W・S_W，高さ P_H・P_H となる．この長方形ポリゴンの中心を z 軸が通り，かつ z 軸とポリゴンが直交するように配置する．ポリゴン間の間隔は，スライスの厚さ T とする．デフォルトの視点の座標は，y 軸上にあり，y 軸の＋方向から原点方向を向いている．視点の上方向は z 軸の＋方向と一致する．

(a) 視点の決定

コアを任意の方向から閲覧するために，2 種類の視点計算の方法を実装した．1 つ目はオイラー角のよる視点の決定と，2 つ目はクオータニオンによる視点の決定である．

オイラー角は，x 軸，y 軸，z 軸の回転角度を指定することによる視点の計算法である．コアと視点の位置関係を，図 4.3 に示す．コアは円柱状であるため，側面と上面・底面の閲覧を考えれば良い．ゆえに，コアの軸に沿った回転により側面を閲覧することが可能となり，前後方向への回転により底面の閲覧が可能となる．よって，デフォルトの視点を y 軸上の+から原点とし，回転を z 軸回転 (コア軸回転)，x 軸 (前後回転) の順序で適用することとした．

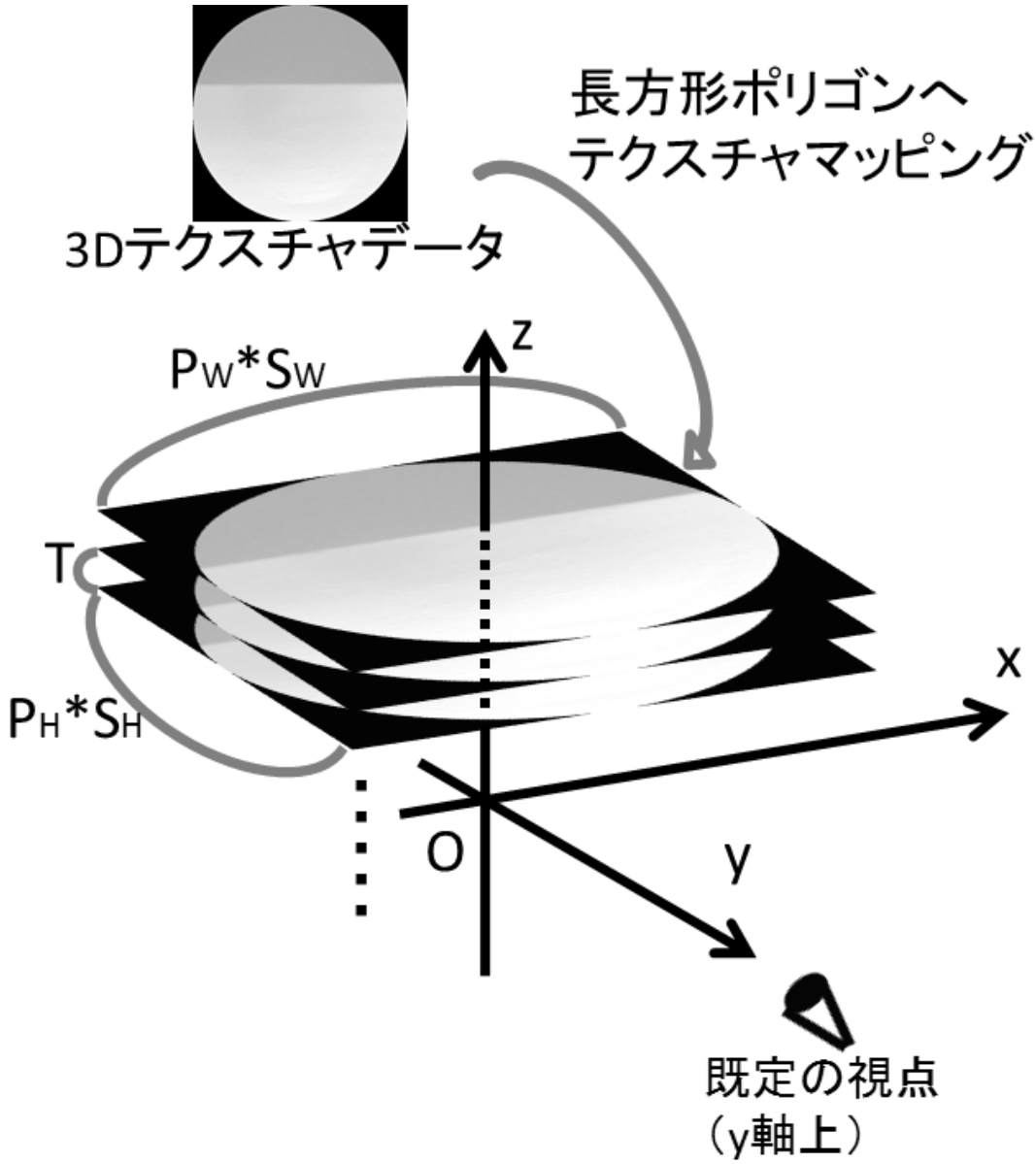

図 4.2: レンダリングの流れ

クオータニオンは，回転軸ベクトルと回転角度を指定することによる視点の計算法である．コアを閲覧するためには前述の 2 軸考慮すれば十分であったが， ここに y 軸を加えて 3 軸を指定した時の視点がどこに移動するかは，ユーザにとってイメージしにくい．その解決案がクオータニオンによる視点の決定であり，画像平面に対するマウスドラッグを用いた回転操作を想定している．この操作法による回転軸ベクトル a は，現在の視線とマウスの移動ベクトルに直交してなおかつコアの描画空間内における原点を通る直線となる．現在の視点位置

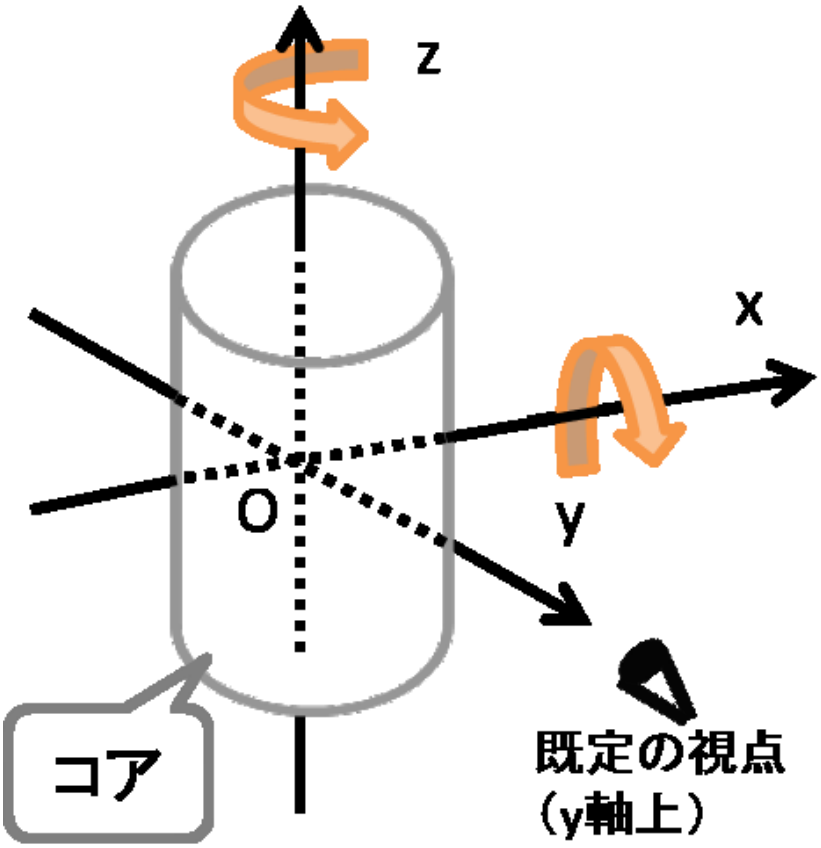

図 4.3: 視点の回転（オイラー角）

から原点への単位ベクトルを $e = (e_x, e_y, e_z)^T$ 視点の上向きベクトルを $u = (u_x, u_y, u_z)^T$，画像平面におけるマウスの移動ベクトルを $m = (m_x, m_y, 0)^T$，マウスの移動ベクトルと x 軸のなす角を θ とすると，回転軸ベクトルを $a = (a_x, a_y, a_z)^T$ は以下の式で表される．

$$S = \begin{pmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{pmatrix} \tag{4.1}$$

$$R = ee^T + (\cos\theta)(I - ee^T) + (\sin\theta)S \tag{4.2}$$

$$a = Ru \tag{4.3}$$

現在の空間座標に対して a を軸とした座標変換を適用することにより，新たな視点が決定される．図 4.4 のような回転となる．

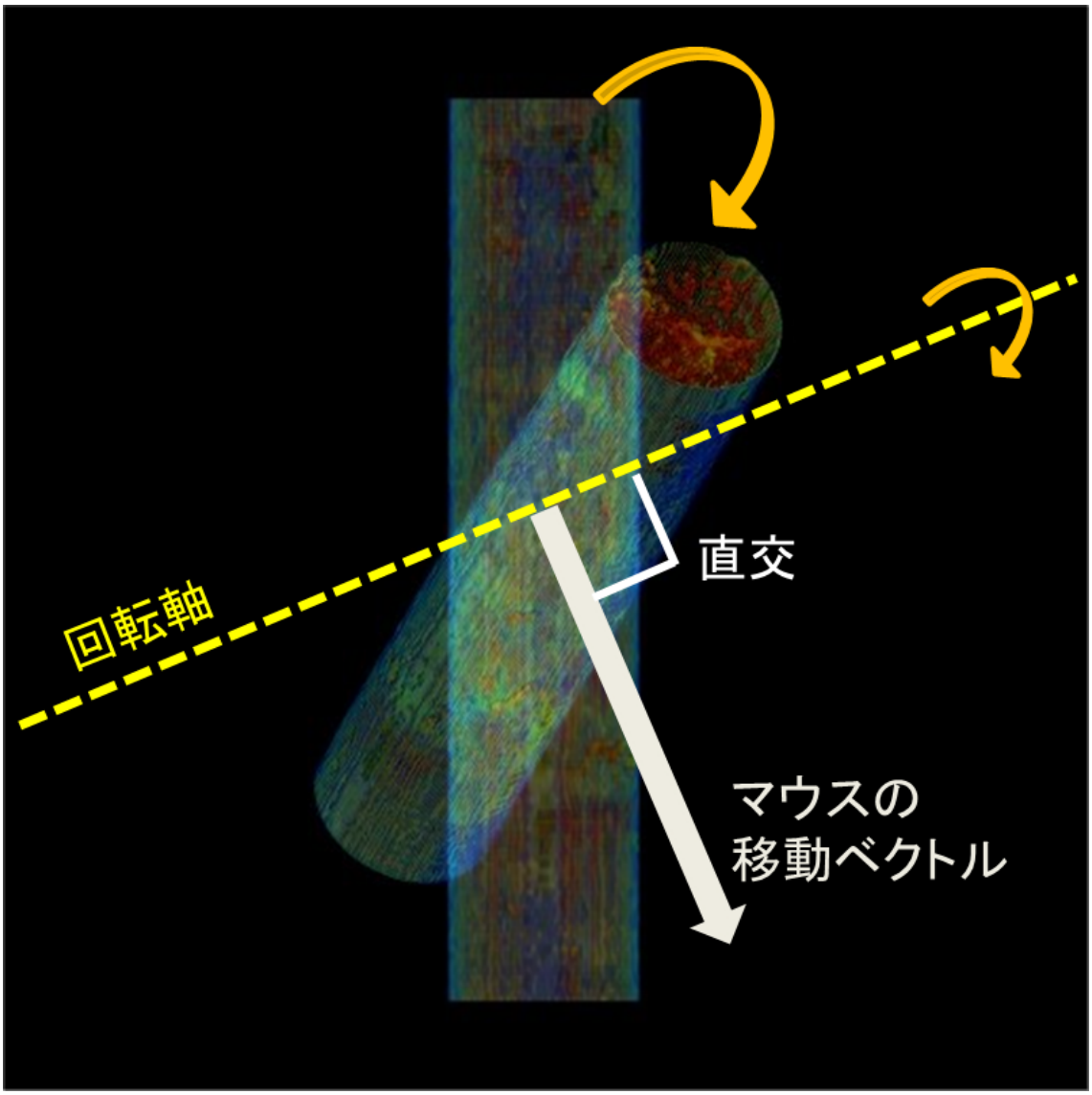

図 4.4: 視点の回転（クオータニオン）

(b) ボリュームデータの圧縮

本プロジェクトで扱うコアデータの 1 枚の DICOM ファイルに格納されているスライス画像のサイズは，幅 512 ピクセル，高さ 512 ピクセルであり，1 ピクセルあたり 16 ビットである．また，1 コアセクションあたりの DICOM ファイルの数は，およそ 400〜2500 である．ゆえに，ボリュームデータは大きいもので 1GByte を超えるデータサイズとなる．これをそのままテクスチャデータとして用いる場合，描画が終わるまでに時間を要することになる．そこで，描画処理の時間短縮のために，ボリュームデータの圧縮を行う．ボリュームデータの解

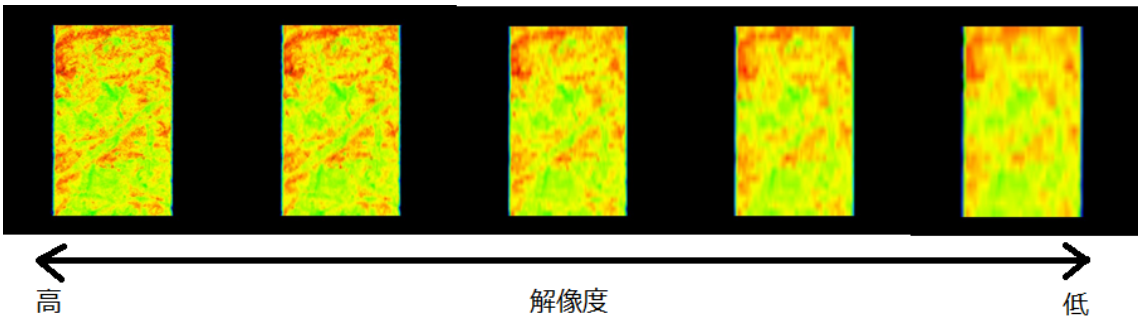

図 4.5: ボリュームデータ圧縮時の画像の違い

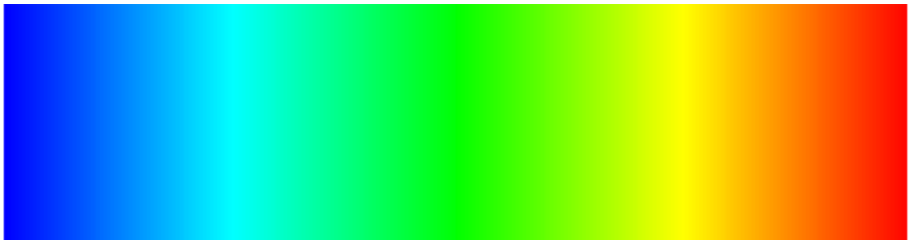

図 4.6: ヒートマップのグラデーション

像度を 5 段階（データ量 1/1, 1/8, 1/64, 1/216, 1/512）用意し，ユーザの求める品質，応答速度に応じて選択できるようにした．圧縮には，平均画素法を用いた．これは例えば，$3 \times 3 \times 3$ ボクセルが $1 \times 1 \times 1$ ボクセルに圧縮されるとき，元のボクセル値の平均値を新たなボクセル値として用いるものである．この手法を用いることにより，単純にボクセルを間引いた圧縮手法に比べて，粗さを抑えることができる．図 4.5 にボリュームデータ圧縮時の画像を示す．データ量を 1/512 に圧縮した時でも大まかな形が保持できていることが分かる．また，ズームの度合いに合わせて扱うスライス数も変更している．例えば，ズームアップした場合には，ごくわずかしか描画されないにも関わらず 2000 枚分のデータを扱うことを避けるため，ズーム倍率の逆数× 512 スライス分の領域を取り扱うこととした．

(c) CT 値に対する色及び透過の設定

CT 値に対するの色付けには，ヒートマップを用いる．ヒートマップとは，値の低いものを青色に，値の高いものを赤色にマッピングするものであり，図 4.6 のようなグラデーションとなる．

CT 値の範囲は-1000〜4000 と定義されていが，0 以下の数値は液体もしくは気体であるため，それらは掘削コア試料の中にはほとんど含まれていない．そのため，本システムにおいて扱う CT 値の範囲を 0 以上 4000 以下とした．RGB 値は，CT 値 v($v_{min} \leq v < v_{max}$) を用いて，4.4 式，4.5 式，4.6 式で表される．グラフ化すると図 4.7 の関数で表される．

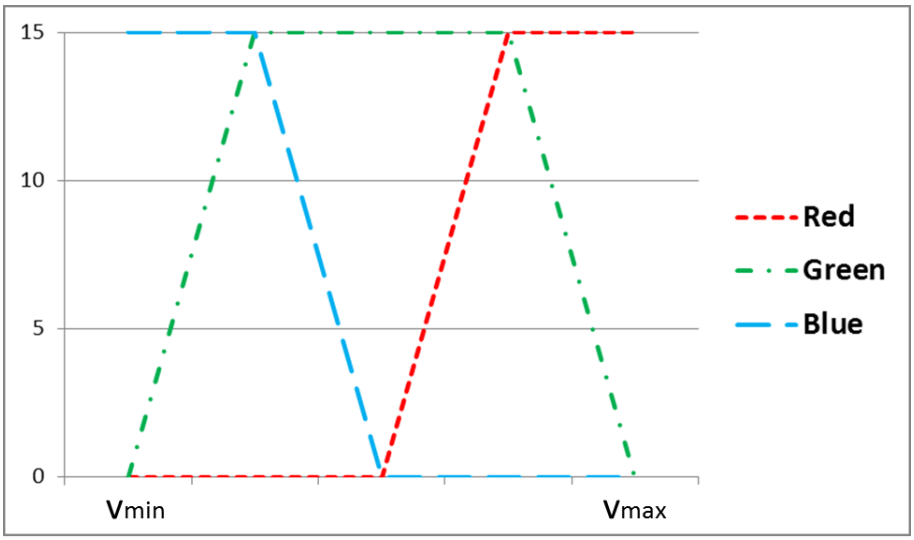

図 4.7: ヒートマップ関数

$$R(v) = \begin{cases} 0 & (v_{min} \leq v < \frac{v_{min}+v_{max}}{2}) \\ 0.25v - 32 & (\frac{v_{min}+v_{max}}{2} \leq v < \frac{3(v_{min}+v_{max})}{4}) \\ 15 & (\frac{3(v_{min}+v_{max})}{4} \leq v < v_{max}) \end{cases} \tag{4.4}$$

$$G(v) = \begin{cases} 0.25v & (v_{min} \leq v < \frac{v_{min}+v_{max}}{4}) \\ 15 & (\frac{v_{min}+v_{max}}{4} \leq v < \frac{3(v_{min}+v_{max})}{4}) \\ 63 - 0.25v & (\frac{3(v_{min}+v_{max})}{4} \leq v < v_{max}) \end{cases} \tag{4.5}$$

$$B(v) = \begin{cases} 15 & (v_{min} \leq v < \frac{v_{min}+v_{max}}{4}) \\ 31 - 0.25v & (\frac{v_{min}+v_{max}}{4} \leq v < \frac{v_{min}+v_{max}}{2}) \\ 0 & (\frac{v_{min}+v_{max}}{2} \leq v < v_{max}) \end{cases} \tag{4.6}$$

また，この色付けを CT 値に対して線形にするだけでなく，3 次スプライン補間を用いた関数により色付けを変化させる機能も実装した．CT 値と色値の組列を指定することにより，これらの点間を 3 次スプライン補間した関数を生成する．適切な曲線を指定することにより，濃淡の調整が可能となる．例えば，図 4.8 に示す関数を適用することにより，図 4.9 のように濃淡の差を強調する画像を生成することが可能となる．

(d) CT 値に対する α 値の設定

CT 値に対する α 値には，4bit 割り当てた．RGBA を合計 16bit として扱うことができれば，システム上都合がよいからである．

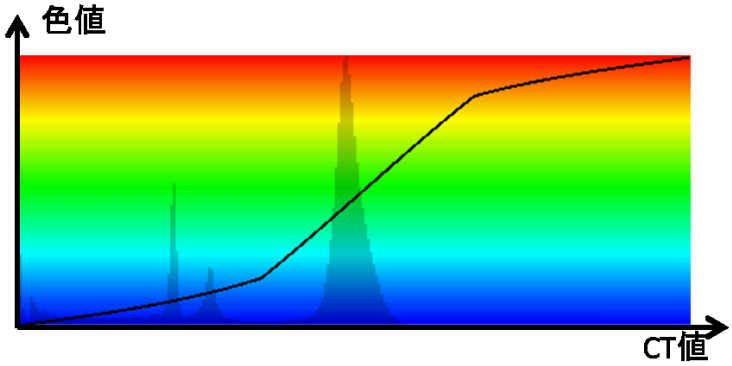

図 4.8: 色関数の例

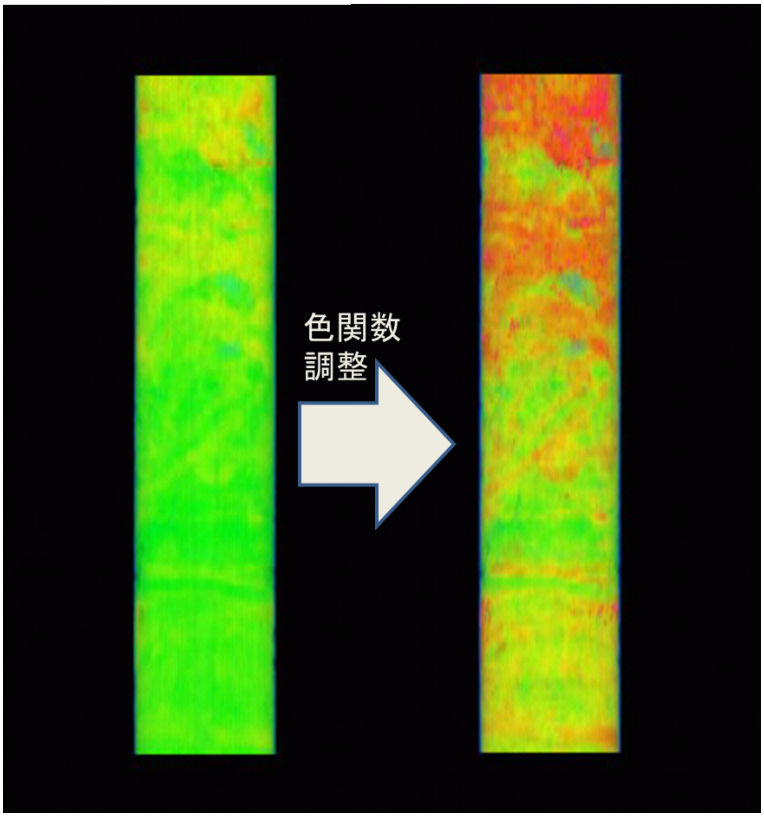

図 4.9: 色関数適用後

クライアントアプリケーションが CT 値とα値の組列を指定し，点列を線形補間することによりα値関数を生成する．図 4.10 のような関数を用いれば，図 4.11 のように，コアの表面を覆っていた CT 値の低いボクセルが透過され，内側の構造を閲覧することが可能となる．

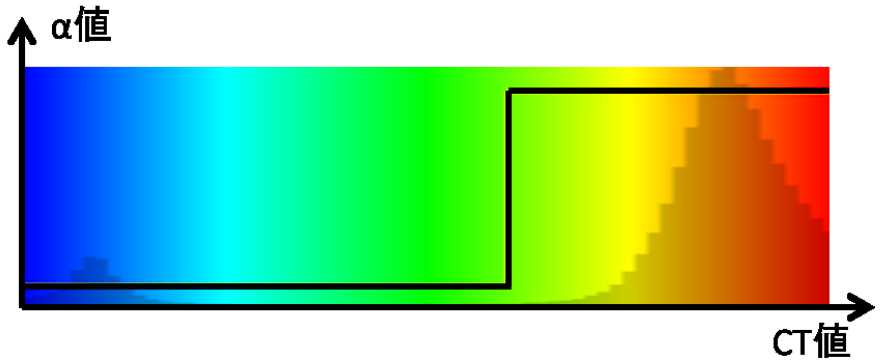

図 4.10: α 関数の例

(e) 縦方向切断

縦方向の切断は，OpenGL のクリッピング関数を用いて実現する．クリッピング処理は，4.7 式を満たす領域を描画しない．縦方向切断を実現するには，z 軸に水平で原点を通る平面を指定すれば良い．この平面の方程式は，角度 θ を用いて 4.8 式によって求めることができる．

$$Ax + By + Cz + D < 0 \tag{4.7}$$

$$sgn(\cos\theta)((\tan\theta)x + (-1)y + (0)z + (0.5 - \tan\theta)) < 0 \tag{4.8}$$

縦方向切断を適用すると，図 4.12 のような画像が得られる．

4.2.4 描画結果の送信

レンダラ起動時に，指定された送信ポートに基づいてソケット通信を行う．送信の内容は，現在の視点位置ベクトル，視点の上向き方向ベクトル，CT 値のヒストグラム，1 ピクセルあたりの長さ (mm) と描画が正常に終了したか否かを示す OK もしくは NG の文字列である．ヒストグラムはボリュームデータの圧縮をする時に計算される．1 ピクセルあたりの長さ s は，視体積のニアクリップ面の長さを L，生成する画像の 1 辺のピクセル数を W_W として 4.9 式で計算される．

$$s = \frac{2L}{W_W} \cdot P_W \cdot S_W \tag{4.9}$$

4.3 議論

各レンダラは，初期化時に描画すべき X-CT スキャンデータを読み込んでから描画指示を待機しているため，クライアントアプリケーションからのインタラクティブな操作を受け付

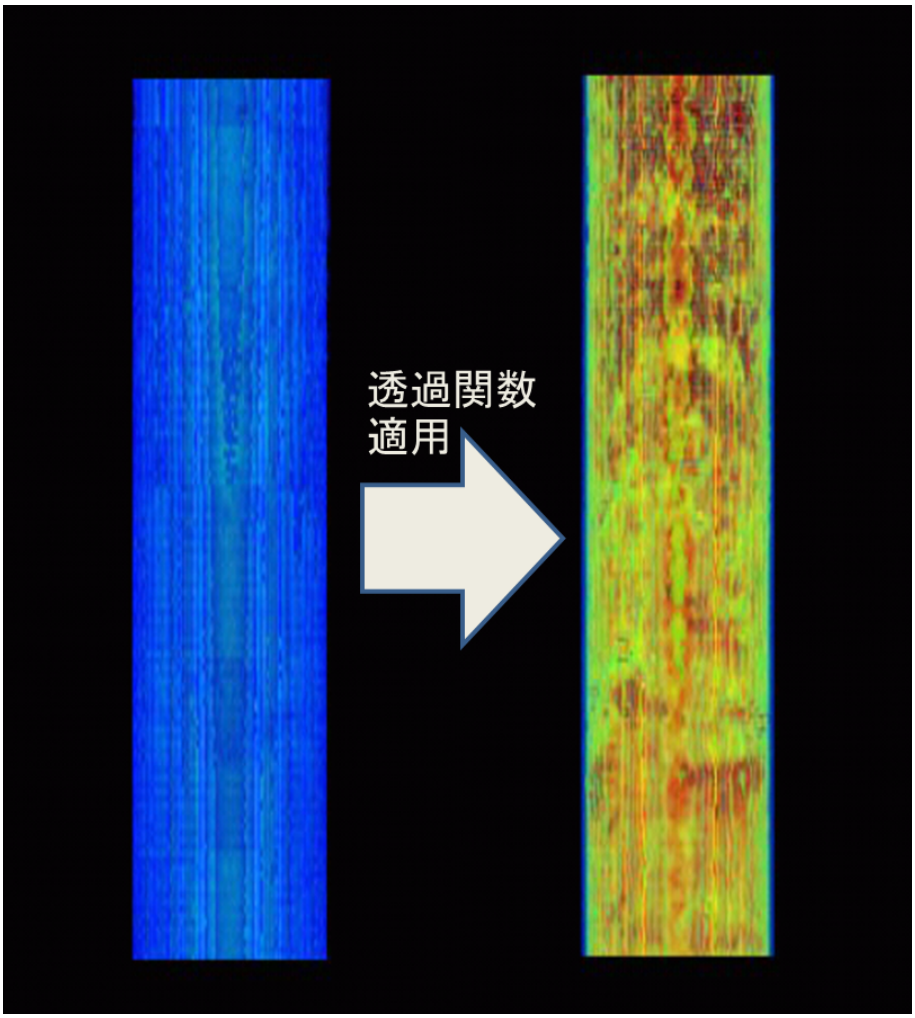

図 4.11: α 関数適用後

けることが可能となった．ボリュームデータは最大で約 1GB であるため，1 ノードあたり最大で約 6 コア分のレンダラを稼働させている．

描画処理においては，視点の変更，CT 値に対する色づけ・透過，コアの切断機能を逐次実行していくことにより，利用者が望む画像を得ることができる．応用的な使い方として，透過機能により，コアの内部に存在する特定の CT 値を持った物体のみを観察することや，解像度選択が可能であるため，初めは低解像度で閲覧してレスポンス時間を短縮し，気になるコア画像があれば高解像度で閲覧するといったことも可能となる．

3 次元の画像処理を高速化させる手段として，GPU を用いた実装が良く知られている．しかしながら，今回の実行環境では，画像の描画先に仮想フレームバッファを利用しているため，GPU を利用した画像処理を行えていない．そこで，GPU を使うことができたならどの程度処理速度が向上するかを実験した．100 枚の画像を生成・保存し，1 枚あたりに要した時間を計測

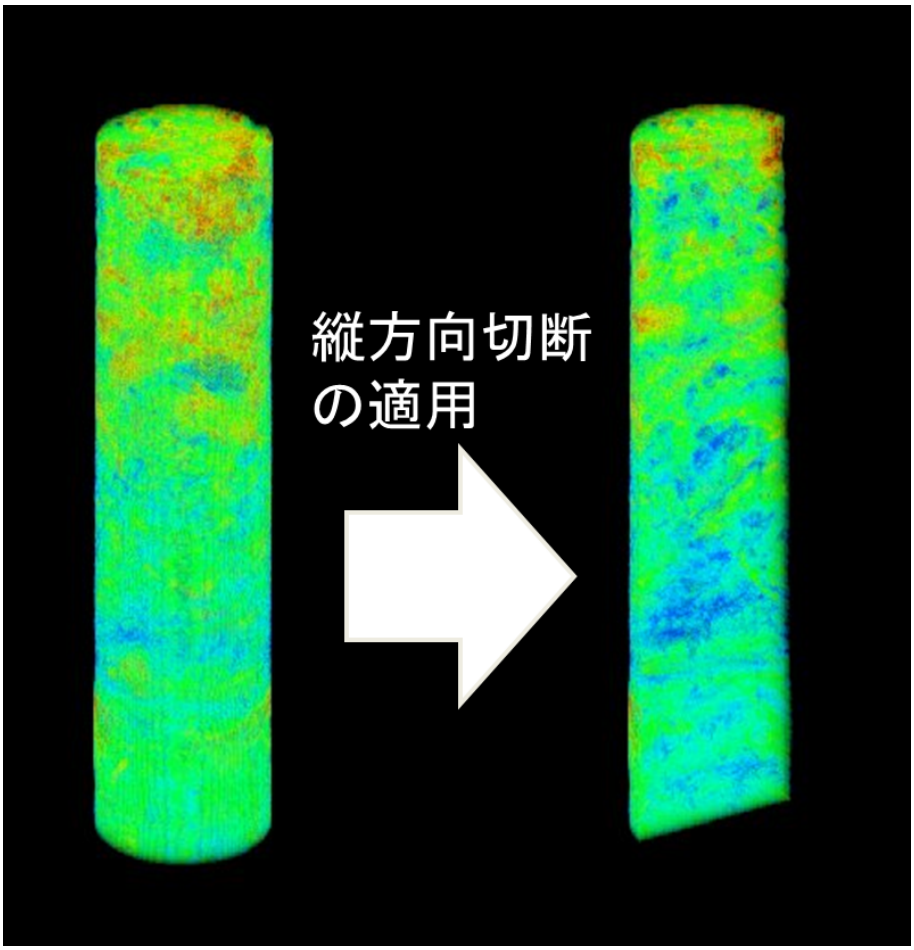

図 4.12: 縦方向切断

表 4.4: GPU 実験結果

	解像度 0	解像度 1	解像度 2
GPU 無し	7.75（秒/枚）	1.96	1.22
GPU 有り	1.80	0.44	0.31

する．計測の対象は 4.2 節における描画処理のみとし，与えるパラメータは，10 度ずつ z 軸回転を加える以外は固定とした．GPU 無しのマシンのスペックは，CPU：Xeon E5645 2.40GHz × 2，メインメモリ 12GB，GPU 有りのマシンのスペックは，CPU：Core i7 930 2.8GHz，メインメモリ 12GB，GPU：NVIDIA GForce GTX 280 において差を比較した．実験結果を表 4.4 に示す．GPU 無しの場合において，主に時間を要している箇所は描画である．GPU を利用した場合は，そうでない場合と比較して処理時間は約 1/4 となった．

第5章　まとめと今後の発展

本プロジェクトでは,「海底コア CT スキャンイメージ可視化のためのクラウドサービス」を開発し，筆者はそのサービスを構成する「DICOM スライス画像のための 3 次元レンダリングサーバソフトウェア」を開発した．本プロジェクトの成果により，データサイズの大きさやコア閲覧用の専用ソフトウェアが存在しなかったために，従来扱いが困難であった掘削コア試料の CT スキャンイメージを，データサイズを気にすることなく，しかもコアの閲覧に適したインタフェースにより，コアを分析することが可能となった．

筆者と共にプロジェクトを進めた佐々木・岡本両氏により開発されたクライアントアプリケーションである Virtual Core Viewer は，Android アプリケーション版は Google Play にて，Web アプリケーション版は Virtual Core Library にて，それぞれ公開している．Virtual Core Viewer の外観を図 5.1 と図 5.2 に示す．Android アプリケーション版はタッチ操作ピンチイン・ピンチアウトやフリックによる，タッチ操作を活かした閲覧インタフェースである．上部のタブ切り替えにより画面遷移することで，検索や色付けや切断の機能を利用するための画面へ移動する．一方 Web アプリケーション版は，検索やコアに対する操作は一画面に配置されている．広く普及した Web ブラウザ（Internet Explorer，FireFox，Google Chrome 等）から閲覧可能であり，PC やタブレットまたは OS の違いにもよらず，閲覧可能である．

今後の発展として，医用画像への応用が考えられる．筆者の開発したレンダリングエンジンは，DICOM 形式のデータであれば，それが掘削コア試料でも人体でも構わない．図 5.3 と図 5.4 は，筆者の開発したレンダリングエンジンで描画した，脳の MRI スキャン画像と腕の CT スキャン画像である．また，レンダリングエンジンを呼び出すための Web API を作成したため，閲覧したい対象に応じたユーザインタフェースを作成することにより，本プロジェクトで開発したシステムと同様の恩恵を受けられることとなる．それはすなわち，端末性能に依存しない DICOM 画像の閲覧と，それが帯域の狭い通信回線でも可能であるということである．医用画像へ応用した場合の更なる利点として，個人情報の隠匿が挙げられる．DICOM のメタデータには，患者の氏名，年齢，体重が記載されているため，オープンに扱うことは困難である．それらの個人情報を伏せたまま画像情報のみを閲覧できることは，今後の医学の発展にもつながると期待できる．

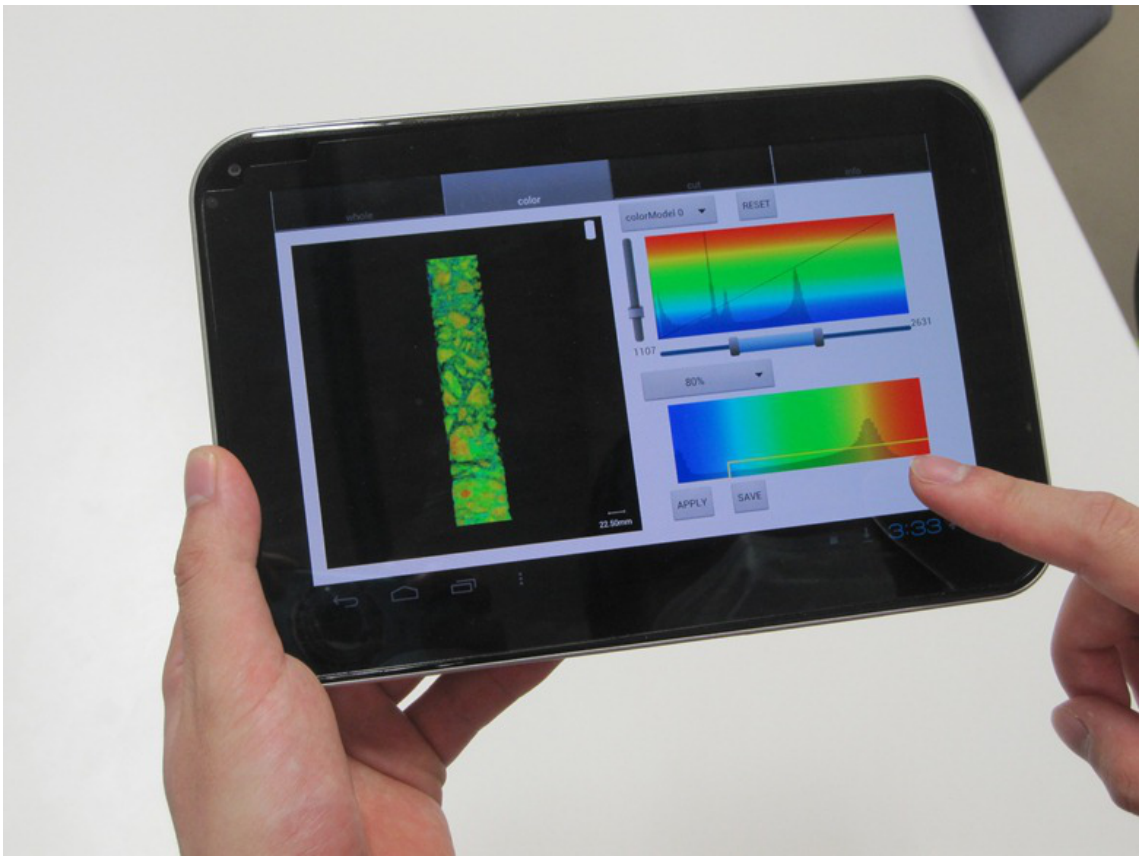

図 5.1: Android アプリケーション版の Virtual Core Viewer

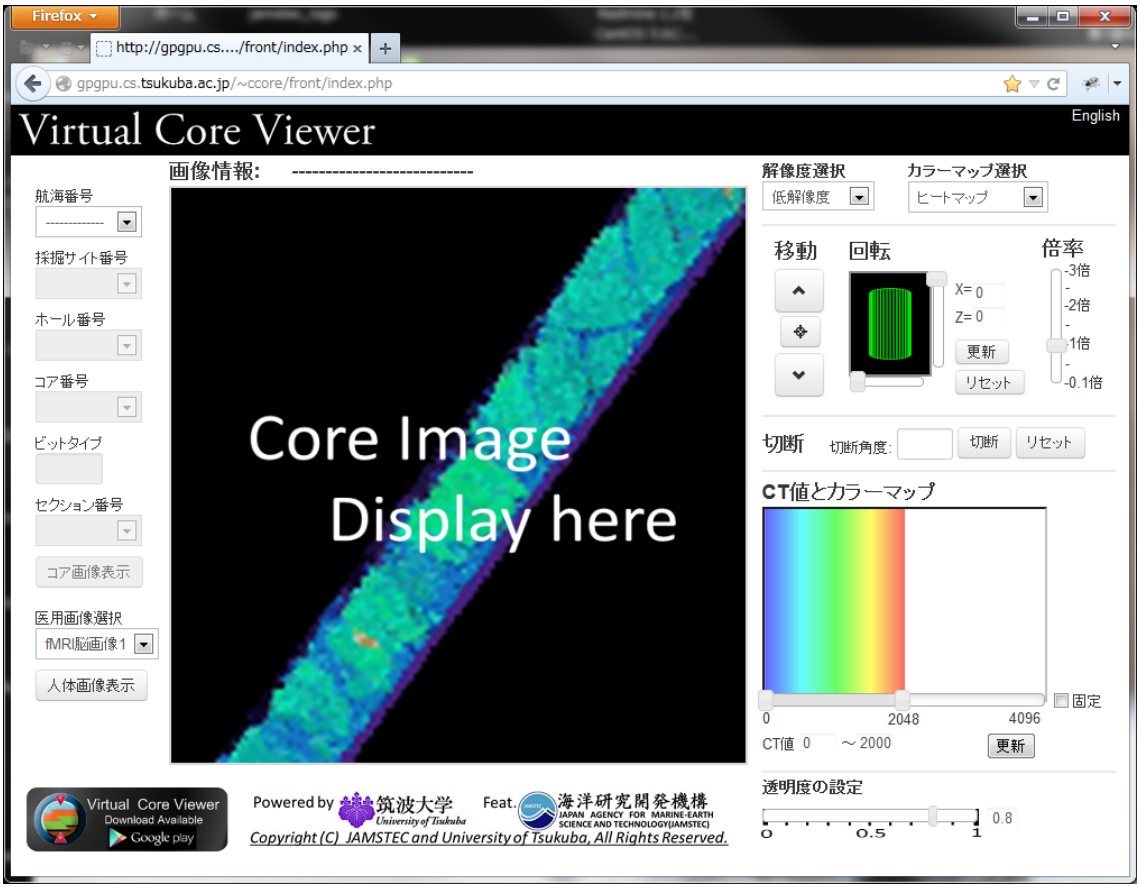

図 5.2: Web アプリケーション版の Virtual Core Viewer

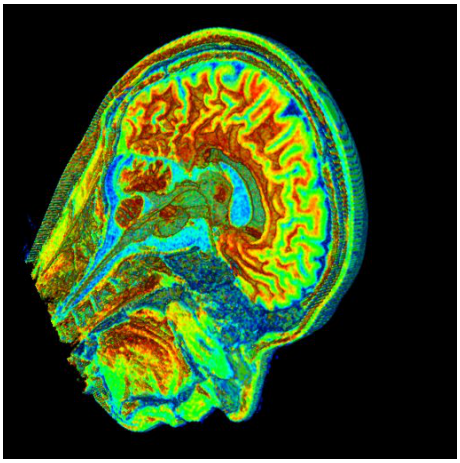

図 5.3: 脳の MRI 画像

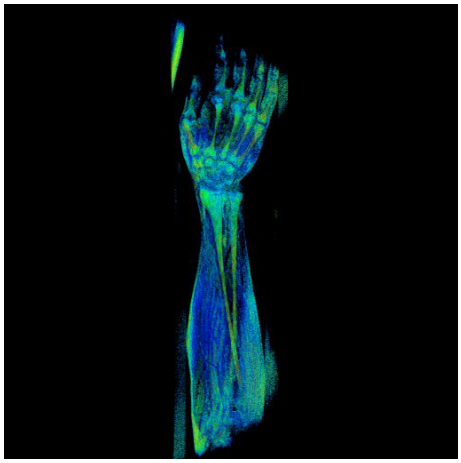

図 5.4: 腕の CT 画像

謝辞

本プロジェクトの担当教員である山際伸一准教授，和田耕一教授には，日々ご指導ご助言をいただき，感謝しております．また，高知コア研究所の久光敏夫先生の精力的な協力により，プロジェクトを円滑に進めることができました．本当にありがとうございました．IPLAB の田中二郎教授，志築文太郎准教授，三末和男准教授， 高橋伸准教授，Simona Vasilache 助教は，私が IPLAB に配属されて以来，ゼミや発表等を通じてご指導していただきました．高度 IT 人材育成のための実践的ソフトウェア開発専修プログラムの担当教員である山戸昭三教授，中沢研也教授には 2 年間通じて多くのアドバイスをいただきました．山際研究室の秘書の櫻井美知代さんには，本プロジェクトの事務的な仕事をサポートしていただきました．電気通信普及財団及び栢森情報科学振興財団の助成により，プロジェクトを推進することができました．本プロジェクトのチームメンバーである佐々木慎君，岡本昂也君のおかげで，1 年間楽しくプロジェクトを進めることができました．

これまで支えてくださった，先生，友人，家族全員にこの場を借りて感謝申し上げます．

参考文献

[1] IODP INTEGRATED OCEAN DRILLING PROGRAM, http://www.iodp.org/

[2] Drilling ship Chikyu returns deepest seabed core samples yet: Nature News Blog, http://blogs.nature.com/news/2012/09/drilling-ship-chikyu-returns-deepest-seabed-core-samples-yet.html

[3] Virtual Core Library, http://www.kochi-core.jp/VCL/

[4] JAMSTEC 地球深部探査センター, http://www.jamstec.go.jp/chikyu/jp/CHIKYU/index.html

[5] DICOM: About DICOM, http://medical.nema.org/Dicom/about-DICOM.html

[6] クラウドコンピューティングとは【cloud computing】, http://e-words.jp/w/E382AFE383A9E382A6E38389E382B3E383B3E38394E383A5E383BCE38386E382A3E383B3E382B0.html

[7] Google Apps for Business, http://www.google.com/intl/ja/enterprise/apps/business/

[8] Amazon EC2 （仮想サーバー Amazon Elastic Compute Cloud）, http://aws.amazon.com/jp/ec2/

[9] JSON, http://www.json.org/

[10] Gdcm Home Page, http://www.creatis.insa-lyon.fr/software/public/Gdcm/

[11] A. Rosset, L. Spadola, and O. Ratib, Osirix: an open-source software for navigating in multi-dimensional dicom images.," Journal of Digital Imaging, vol. 17, no. 3, pp. 205-216, 2004.

[12] M. Yakami, K. Ishizu, T. Kubo. T. Okada. and K. Togashi, "Development and Evaluation of a Low-Cost and High-Capacity DICOM Image Data Storage System for Research," Journal of Digital Imaging, Springer, vol. 24, no.2, pp. 190-195, April 2011.

[13] F. Lamberti and A. Sanna. A streaming-based solution for remote visualization of 3D graphics on mobile devices. IEEE Transactions on Visualization and Computer Graphics, 13(2):247.260, 2007.

[14] Sagar Saladi, Joerg Meyer, " Wavelets and Textures with Illumination for Web-based Volume Rendering ", High-Performance Computing Symposium 2003, SCS Advanced Simulation Technologies Conference, pp. 171-177, March 30-April 3, 2003.

[15] M Moser, D Weiskopf. “ Interactive volume rendering on mobile devices ”, Vision, Modeling, and Visualization 2008, pp. 217-226, October 8-10, 2008.

付 録A　プロジェクトの進行計画

開発スコープをS0，S1，S2の3段階として，図A.1に示すスケジュールで開発を進めた．10月4日は学内における本プロジェクトの中間発表であり，12月5日はICNC ’12（International Conference on Networking and Computing）にて本プロジェクトの成果を発表することとしたため，このようなスケジュールとなった．

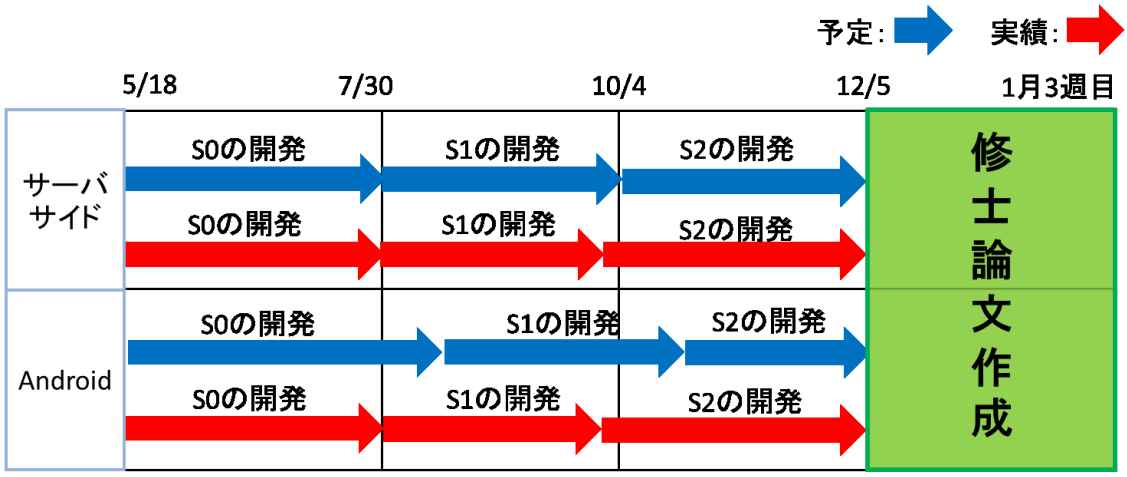

図 A.1: 開発スケジュール

各スコープで実装する機能の一覧を以下に示す．S0では，コアの分析に必要となる基本的な機能を実装する．S1では，コアの検索機能と，新機能の実装をする．S2では，基本機能の拡張とインタフェースの改善をする．

- S0
  - 拡大，縮小
  - 回転
  - 移動
  - コアの選択
  - 色付け機能
  - CT値と色の対応の表示

  - 縦方向切断
  - 透過
- S1
  - 検索機能
  - スケールの表示
  - CT 値のヒストグラム表示
- S2
  - CT 値と色の対応を選択する機能
  - CT 値と色の対応を保存しておく機能
  - 斜め方向切断

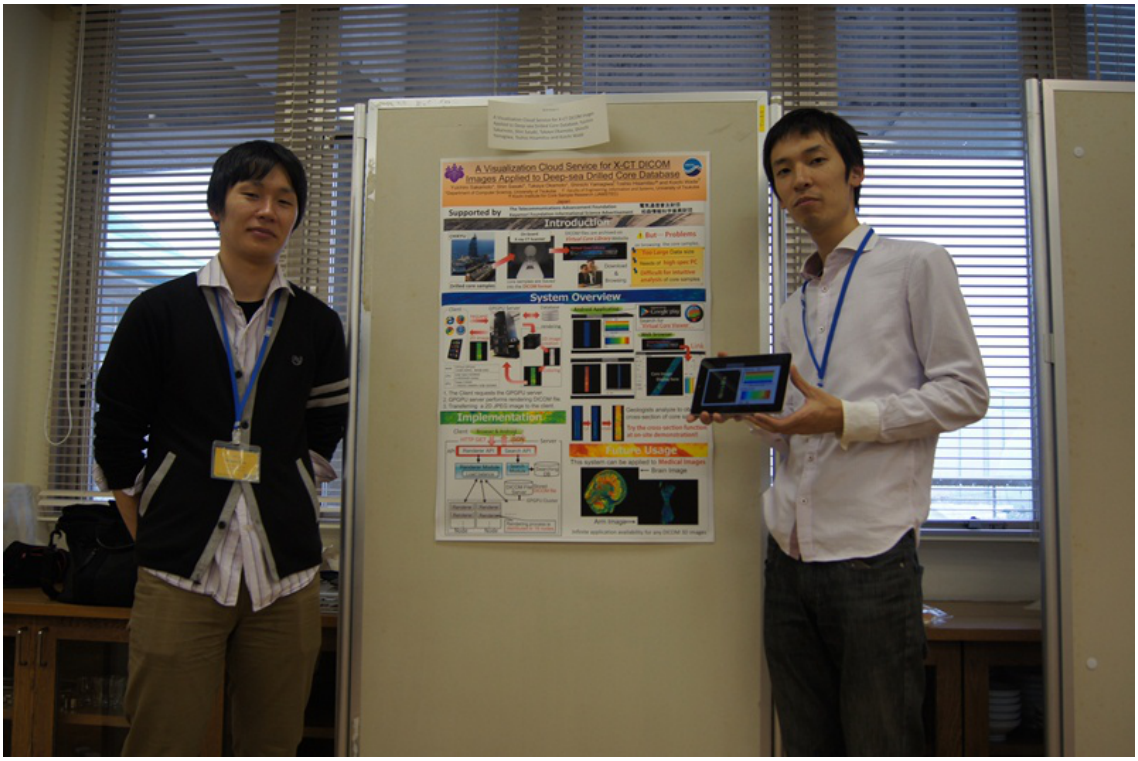

図 A.2: ICNC でのポスター発表の様子

図 A.3: Google Play にて公開されている

付 録B　レンダリングエンジンの実行方法

B.1　必要となるソフトウェア一覧

1. Xvfb（仮想フレームバッファ）
2. GDCM（DICOM 読み込み用ライブラリ）
3. import（画面キャプチャ，ImageMagick に含まれている）

Xvfb と import については，yum でインストール可能である．GDCM は，git リポジトリからソースコードをダウンロードして，自分で make する．その手順は以下のとおりである．

1. cd SOMEWHERE
2. git clone git://git.creatis.insa-lyon.fr/gdcm
    - (Optional, for test suite) git clone git://git.creatis.insa-lyon.fr/gdcmData
3. mkdir gdcmInstall
4. mkdir gdcmBin
5. cd gdcmBin
6. ccmake ../gdcm
    - Toggle and adjust the required options and parameters (see below for more info)
    - hit c (a couple times, until no stars appear, in order configure)
    - hit g (generate makefiles)
    - hit q (quit ccmake)
7. make
8. make install

B.2 コンパイル

レンダラのコンパイル時には，GDCM のライブラリファイル一式と，インクルードファイルへのパスと，OpenMP のオプションを付加する．

レンダラのコンパイルオプション

-I/usr/include/ -I/home/sakamoto/gdcm/include/gdcm -L/usr/lib64/ -L/home/sakamoto/gdcm/lib/gdcm -lglut -lGLU -lGL -lX11 -lm -lgdcm -lgdcmjpeg16 -lgdcmjpeg12 -lgdcmjpeg8 -lgdcmopenjpeg -fopenmp

B.3 実行手順

ゲートウェイのレンダラモジュールから Xvfb とレンダリングエンジンを起動することを想定している．

起動手順の具体例を以下に示す.

1. Xvfb :25 -screen 0 512x512x24 2> /dev/null &
2. ./renderer -sendPort 30001 -recvPort 30011 -dir /home/sakamoto/coredata/core1/ -display :25.0

Xvfb コマンドの実行は，ディスプレイ番号と解像度の指定が必要となる．解像度設定は，それがそのまま生成する画像の大きさとなる．今回の実装では 512x512 と定めているが，これを変更する場合は，クライアントアプリケーションの開発者と相談した方が良い．Xvfb は起動中絶えず標準出力 or 標準エラー出力にメッセージを送り続けるため，これを null デバイスへリダイレクトすることにより，メッセージを破棄する．最後に，バックグラウンド実行を指示して Xvfb を立ち上げる．

レンダリングエンジンの引数は，以下のように定められている．すべての引数は必須であり，これらが無いもしくは間違っていると，レンダリングエンジンは立ち上がらない.

-sendPort レンダリングエンジンがソケット通信を受け付けるポート

-recvPort レンダラモジュールがソケット通信を受け付けるポート

-dir DICOM ファイルがおいてあるディレクトリ，最後は/で終わらなければならない

-display Xvfb を立ち上げたときのディスプレイ番号

B.4 FAQ

Q. 出力する画像のサイズを変更したい

A. Xvfb の引数に希望するサイズを指定する

Q. ゲートウェイの IP アドレスが変わった

A. Main.cpp で定義されている SEND_IP_ADDR を書き換える

Q. import によるキャプチャができない

A. 保存先に書き込み権限があるか確かめる

Q. DICOM ファイルの読み込みに失敗する

A. メモリ確保に失敗しているか，メタデータに誤りがある

付 録C　レンダリングエンジンのプログラム説明書

C.1　VolumeRenderer クラス

VolumeRenderer()

引数 なし

返り値 なし

処理内容

メンバ変数の初期化と Reader のインスタンスを作成.

void exec()

引数 なし

返り値 なし

処理内容

縦方向切断をしたいときは，クリッピング処理を Enable すると適用される．

メタデータを利用して，コアの長さとスライスの大きさの比が一致するように計算している．現在のモデルビュー座標の逆行列を計算することにより，現在の視点座標を計算する．テクスチャマッピングとなる長方形ポリゴンの配置をする．配置の順序は，現在の視点から遠い順に配置する．

void updateVolume(int depth_from, int depth_to, int resolution, int colorfunc)

引数 **int depth_from**　何スライス目から

int depth_to　何スライス目までか

int resolution　ボリュームデータの圧縮率．0〜4

int colorfunc　0：ヒートマップ，1:モノクロ

返り値 なし

処理内容

ボリュームデータに対して色付けして，テクスチャデータを作成して登録する．ヒストグラムの計算もここで行う．

void cutVertically(int cutdir)

引数 **int cutdir** 切断する角度

返り値 なし

処理内容

cutdir から，クリッピング平面を計算する．

void init(const char *dir)

引数 **const char *dir** DICOM ファイルが保存されているディレクトリのパス

返り値 なし

処理内容

dir に指定されたディレクトリから DICOM ファイルを読み出して，CT 値データをメモリに読み込む．その際にメタデータも読み込むので，それを DrawParams のメンバ変数に格納する．3D テクスチャを利用するための設定もここで行っている．

C.2 Reader クラス

int importCtData(const char *dir)

引数 **const char *dir** DICOM ファイルが保存されているディレクトリのパス．パスの最後はスラッシュで終わること．×/home/hoge ○/home/hoge/

返り値 0:正常，－ 1:エラー

処理内容

展開済みファイル “.rawdata” が存在するときはそれを読み込む．そうでないときは，DICOM を 1 枚 1 枚読み出す．メタデータからスライス番号を取得して，その順番でメモリに格納する．DICOM の読み出しは OpenMP で並列化している．ただし，IO がネックになるため，並列数はコア数と同じくらいで頭打ちになる．

int getCtData(unsigned short *data, int resolution_w, int resolution_h, int resolution_d, int depth_from, int depth_to)

引数 **unsigned short *data** 圧縮したCT値データを格納する先．ポインタが指す領域のメモリ確保は呼び出しもとでやる．

int resolution_w 作成するボリュームデータのwidth

int resolution_h 作成するボリュームデータのheight

int resolution_d 作成するボリュームデータのdepth

int depth_from 必要とするスライスはdepth_from枚目から

int depth_to 必要とするスライスはdepth_from枚目まで

返り値 0:正常，－1:エラー

処理内容

Readerが保持しているボクセルデータ（CT値データ）を，w*h*dボクセルに圧縮する．

void printParams()

引数 なし

返り値 なし

処理内容

メンバ変数を出力する．デバッグ用．

void setWH(const char *dir, const char *file)

引数 **const char *dir** 読み込むDICOMが保存されているディレクトリ

const char *file 読み込むファイル名

返り値 なし

処理内容

dirとfileを結合してファイル名（DICOMファイル）を作り，そのファイルから描画に必要となるメタデータを読み込む．読み込むのは画像のwidth，height，pixel spacingである．

C.3 Color クラス

void setAlphaFunc(double xy[][2], int num_of_points)

引数 **double xy[** [2]] 関数を生成するための点列 xy[][0] が x の値で xy[][1] が y の値

int num_of_points 点列の数

返り値 なし

処理内容

CT 値に対応する α 値を計算したテーブルを作成する．引数に指定された隣り合う点列を線形補間した y の値をテーブルに格納する．

unsigned char getAlpha(unsigned short v)

引数 **unsigned short v**

返り値 引数 v に対応する α 値

処理内容

α テーブルを参照して，v に対応する α 値を返す．

void setColorFunc(double xy[][2], int num_of_points)

引数 **double xy[** [2]] 関数を生成するための点列 xy[][0] が x の値で xy[][1] が y の値

int num_of_points 点列の数

返り値 なし

処理内容

CT 値に対応する色値を計算したテーブルを作成する．引数に指定された点列をスプライン補間した y の値をテーブルに格納する．

void apply(unsigned short *v, int len)

引数 **unsigned short *v** CT 値データ列

int len v の要素数

返り値 なし

処理内容

色値テーブルと α 値テーブルから，v の各要素を対応する RGBA 値に変換する．

void setColorType(ColorType c)

引数 **ColorType c** 0:ヒートマップ，1:モノクロ

返り値 なし

処理内容

引数に示された値をセットし，ヒートマップもしくはモノクロの色関数を生成する．

C.4 DrawParams クラス

DrawParams()

引数 なし

返り値 なし

処理内容

変数の初期化をする．扱う変数は，表 4.1，表 **??**，表 4.3 に示す変数である．

void setParams(char *packet)

引数 **char *packet** ゲートウェイからソケット通信で受信した文字列．描画に必要なパラメータが格納されている．

返り値 0:正常にパースできた

処理内容

引数の文字列を “\\” でスプリットし，変数に値を格納する．

void print()

引数 なし

返り値 なし

処理内容

このクラスが保持する変数をコンソールに出力する．デバッグ用途.

C.5 Receiver クラス

レンダリングエンジンが利用するパラメータを，ソケット通信によりゲートウェイから受け取る．

Receiver(int port)

引数 **int port** レンダリングエンジン側の受信ポート

返り値 なし

処理内容

ポート番号をセットする．

char* receive()

引数 なし

返り値 受信したパケットの中身

処理内容

ゲートウェイからの接続を待ち受ける．パケットの内容には表 4.1 に示す値が入っているので，それをそのまま return する．

C.6 Sender クラス

void send(int port, const char * ip_addr, int error)

引数 **int port** ゲートウェイのポート

const char *ip_addr ゲートウェイの IP アドレス

int error ０：正常に描画が完了した，－１描画中に何らかのエラー

返り値 なし

処理内容

引数に指定されたポート，IP アドレスで，コネクションを確立する．DrawParams のメンバ変数にアクセスして表 4.2 に示す値をソケット通信により，ゲートウェイへ送信する．

C.7 シーケンス図

図 C.1 にシーケンス図を示す．MainLoop クラスは存在しないが，表記の便宜上そう書いている．また，それぞれのメソッドの引数は書いていない．

まずは main 関数にて Receiver と Sender のインスタンスが生成される．次に InitVolumeRenderer 関数が呼ばれ，初期化が完了する．次に，Display 関数は描画を行うための関数であり，Idle 関数は描画が終わった後にクライアントからのリクエストを待機する関数である．初期化後は，この Display 関数と Idle 関数のループに入る．終了条件は，Receiver が exit を受け取った時か，強制終了である．

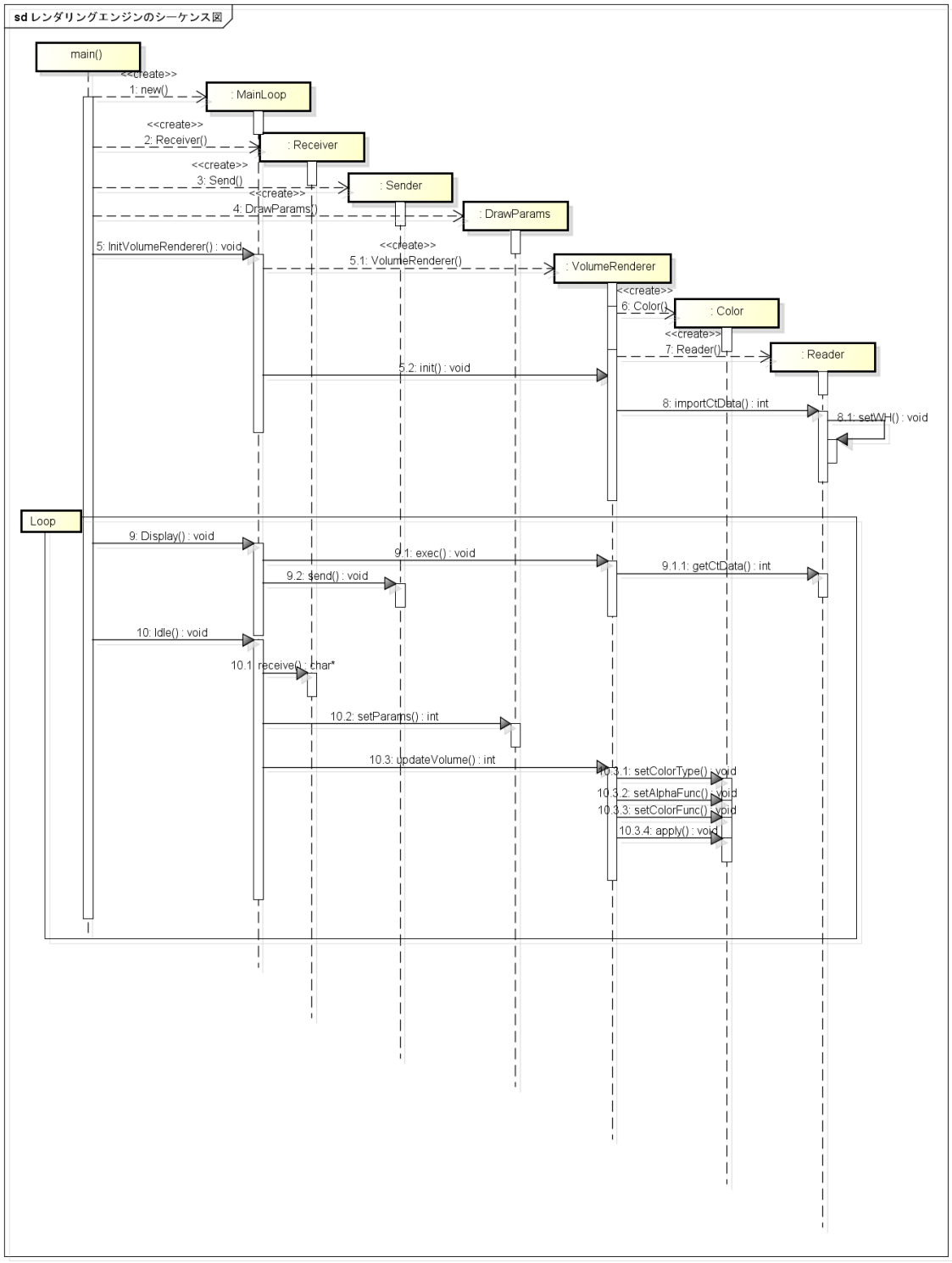
sd レンダリングエンジンのシーケンス図
main()
<<create>>
1: new()
: MainLoop
<<create>>
2: Receiver()
: Receiver
<<create>>
3: Send()
: Sender
<<create>>
4: DrawParams()
: DrawParams
5: InitVolumeRenderer() : void
<<create>>
5.1: VolumeRenderer()
: VolumeRenderer
<<create>>
6: Color()
: Color
<<create>>
7: Reader()
: Reader
5.2: init() : void
8: importCtData() : int
8.1: setWH() : void
Loop
9: Display() : void
9.1: exec() : void
9.1.1: getCtData() : int
9.2: send() : void
10: Idle() : void
10.1: receive() : char*
10.2: setParams() : int
10.3: updateVolume() : int
10.3.1: setColorType() : void
10.3.2: setAlphaFunc() : void
10.3.3: setColorFunc() : void
10.3.4: apply() : void

図 C.1: シーケンス図

付 録D　dicom2raw

dicom2raw は，コアデータをデコードし，ボリュームデータのみを格納したファイルを作るためのコマンドである．処理内容は，付録 C の Reader クラスの importCtData とほとんど同じである．第 1 引数に指定されたディレクトリにある DICOM ファイルを順次読み込んで，デコード済みのボリュームデータを作成し，第 2 引数に指定したファイル名で保存する．

dicom2raw コマンドの Usage

./dicom2raw コアデータが保存されているディレクトリ 出力ファイル名

注意事項として，保存先に write 権限が必要である．