Vol.2016-DPS-167 No.6
Vol.2016-MBL-79 No.6
Vol.2016-ITS-65 No.6
2016/5/26

ChainVoxel: 3Dモデルのためのスケーラブルな協調編集を実現するデータ構造

今江 健悟[1,a)]　林原 尚浩[1,b)]

概要：協調編集は分散システム上の共有データにおける一貫性保持の問題を抱えている．これは分散システムにおける最も重要な課題の１つある．この問題を解決するために様々なアプローチが提案されてきたが，それらのほとんどがドキュメントのためのものである．本論文では，ChainVoxel という 3D モデルを対象としたスケーラブルなリアルタイム協調編集のための Conflict-free なデータ構造を提案する．ChainVoxel は，Shapiro らによって提案された Conflict-free Replicated Data Type (CRDT) を元にしており，このデータ構造上で 3D モデルを編集するための操作の集合も定義している．提案手法は複数のユーザによって同時に 3D 空間の同じ位置への操作による衝突の解決策のためのチェインハッシュテーブルを用いており，各ユーザが持つローカルデータの結果一貫性を保証している．提案手法の有効性を評価するために，ChainVoxel を実装した 3D モデルの協調編集システムのプロトタイプを作成し，操作数に対するデータサイズや一貫性保持に必要なメッセージ数などを計測した．

キーワード：Conflict-free データ構造，協調編集，結果一貫性，3D モデル，分散システム，スケーラビリティ

ChainVoxel: A Data Structure for Scalable Distributed Collaborative Editing for 3D Models

KENGO IMAE[1,a)]　NAOHIRO HAYASHIBARA[1,b)]

Abstract: Collaborative editing is one of the most significant topics in distributed systems because it includes a problem of consistency maintenance on shared data in distributed systems. There are so many papers which have proposed various approaches for this problem so far. Most of solutions that have already proposed are for document editing but there are a few for 3D models. We propose *C*hainVoxel, a conflict-free data structure for lock-free distributed collaborative editing of 3D Models. It is an instance of the conflict-free replicated data type (CRDT) and is defined as a set of operations. As a result, it guarantees the eventual consistency without exclusive control. ChainVoxel contains a chained hash table for collision resolution while editing voxels in the same position by multiple users. We also analyze our proposed algorithm and data structure in terms of data size and the number of operations.

Keywords: Conflict-free data structure, Collaborative editing, Eventual consistency, 3D models, Decentralized systems, Scalability

1. 背景

一貫性の保持は分散システムのデータ共有サービスの設計のための最も基本的な問題の１つである．分散システム上のリアルタイム協調編集は一貫性の更新で多数の論文 [1], [2], [3], [4], [5], [6], [7] で議論されてきた．これらはユーザのグループに同時に同じドキュメント，画像や 3D モデルに操作・閲覧を可能にするものである．しかし，これらのほとんどがドキュメントのためのものであり，3D モデルのためのものはほとんどないのが現状である．

最も簡単な共有オブジェクトの一貫性保持のための解決

[1] 京都産業大学大学院 先端情報学研究科
[a)] i1558048@cse.kyoto-su.ac.jp
[b)] naohaya@cc.kyoto-su.ac.jp

策は，排他制御アルゴリズムを使うことである．しかし，排他制御は一貫性保持のためのメッセージ交換により，システム全体のボトルネックになるリスクを抱えている．これにより排他制御はシステム全体のパフォーマンスを低下させたり，スケーラビリティの向上を妨げてしまう．

Operational Transformation は分散協調編集システムで用いられているアプローチである [6], [8], [9], [10]．これらのアプローチは排他制御を行なわない楽観的並行性制御である．全ての site でローカルな操作は即座に実行される，そして remote site にそれらは送信される．Remote Operations は競合解決のための実行より前に変換される．これらはアルゴリズムにおける一貫性保持のためのスケーラブルなアプローチである．

Shapiro らは分散システム上の一貫性のある更新のための Conflict-free Replicated Data Type (CRDT) を提案した [11], [12]．この研究はデータ構造における楽観的な同時実行制御であり，共有データの結果一貫性を保持する事が出来る． このアプローチでは可換可能な操作のデータ構造としている．したがって，全ての site は任意の操作の系列の変換と排他制御を行なわずにローカル操作とリモート操作を実行する． その結果，このアプローチではクラウドシステムのような大規模分散システムのスケーラビリティを保証している．

これらの Treedoc [7] のような共有ドキュメントのための，いくつかの CRDT を元にした MANET [5] 上の協調編集システムが存在している．

本論文では，3D モデルのためのリアルタイム協調編集に着目している．現在のところ，3D モデルを使用した複数のアプリケーション (e.g., VR システム, 遠隔教育システムなど) が開発されている．そして，3D モデルの編集と閲覧のためのいくつかのファイルフォーマットも提案されている (例えば，COLLADA [13])．そのため，3D モデルはコンピュータ上の主要なコンテンツになりつつあり，ドキュメント，写真や動画のようになろうとしている．加えて，SketchUp [14] や Open3D [15] のような 3D モデルの協調編集サービスなどが開発されている．これらの背景を踏まえると，3D モデルの協調編集は非常に関心が持たれている分野であることがわかる．

本論文では，*ChainVoxel* と呼ばれる共有 3D モデルの協調編集のための Conflict-free なデータ構造を提案する．ChainVoxel は CRDT の概念を元にしたデータ構造であり，特徴的なところはチェインハッシュテーブルで共有データ上の操作の集合を持っていることである．そして，共有 3D モデルに対する操作を可換可能にしている．そのため同時に操作が起こった場合でも，排他制御を行なわずに共有データの結果一貫性特性を保証することができる．

本論文では ChainVoxel を元にした 3D モデルのための協調編集のプロトタイプを実装し，ChainVoxel のふるまいの評価を行なった．このプロトタイプは COLLADA フォーマットを想定している．

2. 関連研究

本節では，リアルタイム協調編集システムの関連研究を紹介する．

2.1 ドキュメントのための協調編集

ドキュメントのためのリアルタイム協調編集システムを実現するために多数のアルゴリズムとデータ構造が提案され，これまで議論されてきた．

Operational Transformation アプローチは一貫性保持のための主要な解決策である．Sun らは因果関係を保持した変換を提案: ドキュメントの協調編集のための Inclusion Transformation と Exclusion Transformation [3] である．

Vidot らにより提案された SOCT アルゴリズム [1] では，操作は sequencer によって与えられたタイムスタンプを用いて全体の順序付けを行なう．この sequencer は単調に増加する正の整数を生成する．その結果，これらの操作はタイムスタンプを元に同じ順序で，個々の site に配信される．

Li ら [16] によって提案された State Difference based Transformation (SDT) アプローチ は任意の変換パスの収束を保証している．SDT は peer-to-peer システムでの協調編集のための始めての Operational Transformation アルゴリズムである．最悪時間計算量は $O(n^3)$ で， 平均計算量は $O(n^2)$ である．

対照的に，Shapiro らが提案したリアルタイム協調編集のための Conflict-free Replicated Data Type(CRDT)[12] では，可換可能なデータ構造を元にした楽観的並行性制御アプローチである．その理由としては，非同期的に複数の site で操作が起こった場合に，共有データの site 間の一貫した状態の収束を保証するためである．Letia らは CRDT ベースのシステムの例を与えで，そしてパフォーマンスとスケーラビリティを示した [11].

2.2 3D モデルのための協調編集

Ha らは Three.js を元にした協調 3D エディタを実装した [17]．このエディタは Web ブラウザ上で実行され, XMPP メッセージングシステムを用いて remote sites と通信をする．このシステムはライトウェイトで, Openfire XMPP サーバを用いて共有データを同期させている．この提案システムは並行性制御のためのクライアント・サーバシステムとしてデザインされている．それはつまり，ロックベースの協調編集システムであることを意味している．その結果，システムのスケーラビリティが限られていることが明らかである，なぜならロックはボトルネックになることと，システム全体のパフォーマンスを損ねるためである．

Kang らはスマートデバイス上の 3D CAD システムを提案

Vol.2016-DPS-167 No.6
Vol.2016-MBL-79 No.6
Vol.2016-ITS-65 No.6
2016/5/26

した [18]．これは Samsung Galaxy Tablet 10.1 with Android 3.2 上で実装されている．この研究の重要な点はスマートデバイスのタッチパネルを用いて 3D モデル編集のための洗練された操作を提供することである．このシステムはタッチパネル上の 10 のジェスチャー (e.g., 一本指ドラッグ，二本指スワイプなど) と 3D のための 10 の操作関数 (e.g.，切り取り，ソリッドオブジェクトの作成など) のマッピングを提供している．それはスタンドアローンソフトウェアシステムではあるが，とても興味深いユーザインタフェースである．

3. システムモデル

私たちは双方向ネットワークによって接続されたプロセスの集団をシステムとして仮定する． ネットワークは以下に定義されるような Quasi-reliable なネットワークを前提とする．

Definition1 (Quasi-reliable network) Quasi-reliable network [19] は以下の３つの特性によって定義する．

Property1 (No creation) プロセス q がメッセージ m をプロセス p から受信した場合，p は q に m を送信する．

Property2 (No duplication) q は p からメッセージ m を 1 つしか受信しない．

Property3 (No loss) p と q が正常で p が q に m を送信した場合，q はいつか m を受信する

本論文では”*process*”と同じ意味で”*site*”という用語を用いる．そのため仮定する分散システムは site の有限集合 $S = \{s_1, s_2, ..., s_n\}$ で定義される．想定する故障は停止故障のみで，ビザンチン故障は起こらないものとする．また，個々の site はユニークな識別子 $sid \in \mathbb{Z}$ を持っているとする．

共有データはこれらの site で複製され，操作は各 site で非同期に起こる．故障していない site では正しいデータが保持されていると仮定する．

3.1 データの一貫性

共有データは全ての site で複製され，それらはローカルに保持されている．これらの複製されたデータは矛盾無く更新されるべきである．私たちは以下の定義の一貫性モデル [12] を想定している．

Definition2 (結果一貫性) 結果一貫性は以下の３つの特性によって定義される．

Property4 (Eventual delivery) 任意の site の正しいデータで配信された操作は，いつかは他の全ての site に正しいデータが配信される．

Property5 (Convergence) 異なる site の正しいデータは同じ操作によって伝搬され，常に同じ状態のものが届く．

Property6 (Termination) 全ての操作は終了する

4. ChainVoxel: Conflict-free なデータ構造

ChainVoxel は CRDT [12] を元にした 3D モデルの協調編集のためのデータ構造であり，チェインハッシュ構造で表現されている．ChainVoxel は可換可能な操作の系列をチェインハッシュ構造によって保持し，これらの構造を元に共有データの結果一貫性を保証している．そのため，ChainVoxel を用いる事で 3D モデルのためのロックフリーな協調編集を実現する事が出来る．

4.1 Voxel

CRDT の元の定義では，CRDT によって表現されるデータ構造の最小の単位として *atom* が定義されている．これに対して，ChainVoxel では atom に相当するものとして，*voxel* を定義している．このため，本論文では voxel の集合から成る 3D モデルを想定している．各 voxel は sid とタイムスタンプ ts を持ち $voxel^{ts}_{sid}$ として表現される．最新のタイムスタンプは存在するタイムスタンプ間で最大の値が割り当てられる．

3D 空間の各 voxel の位置は $posID$ と呼ばれるタプル $(x, y, z) \in PI \subset \mathbb{Z}^3$ で示され，$posID$ に対応する voxel はその位置に配置されているものとする．そのため，各 voxel の位置は $posID$ によって表現され，voxel はサイズ 1 の立方体である．$posID$ は写像 $V \overset{f_{pos}}{\mapsto} PI$ によって voxel と関連付けられる．V は空間上に割り当てられた voxel の集合で，PI は voxel の $posID$ の集合である．同じ位置に対して，２つ以上の異なる voxel の競合が起こった場合，それらのうちの１つだけが空間上に存在する voxel として扱われることになる．これは ChainVoxel が同じ位置の voxel の競合を解決するために重要である．

4.2 ChainVoxel の構造

ChainVoxel は $posID$ に対応する voxel の順序集合を保持し，これらはチェインハッシュ構造によって管理される (図 1 参照).

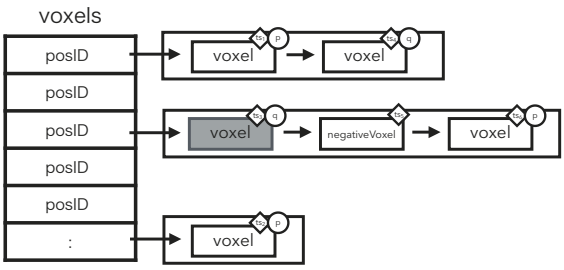

図 1 ChainVoxel の構造

voxel の順序集合として管理する目的は voxel の競合を避けるためである．これらの voxel の順序集合は ts の昇順に

よって並べ替えられる．しかし，複数の site は同じ位置に同時に操作を実行することがある．この状態を *collision* と呼ぶことにする．

collision が発生した状態では，ユーザによる意図を考慮した順序付けとしていくつかの解決策がある [3]. 今回は，複数のユーザ間で collision が起こった際に sid の昇順によって voxel の順序付けをすることにしている．これは非常にシンプルな解決策であるが，協調編集システムのパフォーマンス上の何らかの影響（例えば，オーバーヘッドなど）を負わないという利点があり，純粋に提案手法を評価するために有効である．

上記のように，3D 空間に割り当てられた voxel はその位置によって順序集合として蓄えられる (図 1)．本論文では各チェインの先頭の voxel を primary voxel と，そして他の voxel を sub-voxel と呼ぶ．

各 $posID$ のための voxel のチェインの先頭が primary voxel として扱われる．言い換えると最も古い voxel を primary voxel として扱う．全ての他の voxel は sub-voxel として分類される．primary voxel は共有 3D 空間上に存在することが出来るただ一つの voxel である．図 1 の $negativeVoxel$ は delete 操作により追加される特別な voxel である．$negativeVoxel$ がチェイン上に存在する状態では $negativeVoxel$ の次の voxel が primary voxel になる．

voxel は非同期的に site によって追加される．このことより，各 site で共有 3D モデルの形が異なってしまうかもしれない．しかしながら，各位置の voxel のチェインは全順序で並び替えられる．そのため，全ての site で ChainVoxel 内の voxel は最終的に同じ順序になる．

4.3 ChainVoxel の操作

この節では ChainVoxel 上の操作について説明を行なう．ここでは 2 つの操作 *insert* と *delete* を定義する．各操作は sid と ts によって識別される．これらの操作を Algorithm 1, 2 に示す．

4.3.1 Insert operation

insert 操作は sid，ts と $posID$ を必要とする．sid は site の識別子である．本論文では，C 言語の time 関数のような現在時間を取得できる関数を想定する．タイムスタンプ ts は，関数によって取得される．$posID$ はユーザの入力によって値が取得される．これらの情報は $insertVoxel$ に設定される．$negativeVoxel$ は最初の delete 操作によって追加される．$negativeVoxel$ は各 $posID$ のための voxel の順序集合に高々一つ存在する．insert 操作は $posID$ に既に $negativeVoxel$ が存在する場合，関数 f_{ts} を用いることによって，$negativeVoxel$ の ts を確認する．$f_{ts}(negativeVoxel) \geq ts$ の場合，insert 操作は失敗することになる．

4.3.2 Delete operation

delete 操作は $posID$ に挿入された voxel を取り除くものである．しかし，実際の delete 操作は現在の voxel のチェインに特別な voxel である $negativeVoxel$ を追加する操作として定義している．$negativeVoxel$ より前の voxel は無効化される．既に $negativeVoxel$ が存在する場合は，ts の値に応じて $negativeVoxel$ の ts の更新が行なわれる (Algorithm 2 の 5 行目)．sid は基本的には正の整数であるが，$negativeVoxel$ には負の値が割り当てられている．本論文では voxel のチェインから有効ではなくなった voxel を取り除くためのガベージコレクションの仕組みがあることを前提としている．

Algorithm 1 insert operation

1: **Inputs:**
   $sid \leftarrow$ own site ID
   $ts \leftarrow$ current time
   $posID \leftarrow$ the given position from users
2: **Initialize:**
   $insertVoxel \leftarrow \perp$
3: $insertVoxel \leftarrow (sid, ts, posID)$
4: **if** $negativeVoxel \in V_{posID}$ **then**
5: **if** $f_{ts}(negativeVoxel) \geq ts$ **then**
6: return /* do nothing */
7: **end if**
8: **end if**
9: $V_{posID} \cup \{insertVoxel\}$
10: broadcast $insert(sid, ts, posID)$ to every other sites

Algorithm 2 delete operation

1: **Inputs:**
   $sid \leftarrow$ a negative integer
   $ts \leftarrow$ current time
   $posID \leftarrow$ the given position from users
2: **Initialize:**
   $negativeVoxel \leftarrow \perp$
3: **if** $negativeVoxel \in V_{posID}$ **then**
4: **if** $f_{ts}(negativeVoxel) < ts$ **then**
5: $update(negativeVoxel, ts)$ /* update the timestamp */
6: **end if**
7: **else**
8: $negativeVoxel \leftarrow (ts, posID)$
9: $V_{posID} \cup \{negativeVoxel\}$ /* add a $negativeVoxel$ */
10: **end if**
11: broadcast $delete(ts, posID)$ to every other sites

site p が実行した操作は p を除いた S 内のすべての site に送信される．このような操作を p のローカル操作と呼ぶ．一方，site $q \in S \setminus \{p\}$ においては，リモート操作と呼ばれる．

ローカル操作は各 site で直ちに実行される．一方で，リモート操作は，ローカル操作として実行した site から送信された操作を受信した後に実行される．

Vol.2016-DPS-167 No.6
Vol.2016-MBL-79 No.6
Vol.2016-ITS-65 No.6
2016/5/26

4.4 ChainVoxel の結果一貫性

全ての site は非同期的に insert と delete 操作を実行する．これらの操作は全ての site に送信される．ChainVoxel 内のデータは並行実行とネットワークの遅延が原因で，ある一定時間は各 site 内で異なっているかもしれない．この状態を *conflict* と呼ぶ．一方で，remote operation はすべての site にそのうち配信される，なぜなら，本論文では site 間では Quasi-reliable network を前提としているからである (3 節を参照)．これは全ての site で全てのローカル操作とリモート操作がいつかは実行されることを意味している．従って，Property 4 を保持しているといえる．

ChainVoxel の挿入された voxel と negative voxel は各 site の ts と sid に応じて解決される．そのため，最終的には全ての site の ChainVoxel の状態は一貫したものになる．これにより，Property 5 が保たれる．

ローカル操作は直ちに実行され，リモート操作は Quasi-reliable なネットワークとおして配信される．これは，それぞれが全ての site の ChainVoxel 内に最終的に適用されることを示している．そのため，Property 6 は保持されるといえる．

従って，ChaiVoxel は conflict 状態を解決し，3 節で定義した結果一貫性の特性を保持している．

5. プロトタイプの実装

本研究では，3D モデルのためのリアルタイム協調編集システムのプロトタイプを実装した．プロトタイプは，Java1.8 で実装されている．このプロトタイプの概要は図 2 に示している．

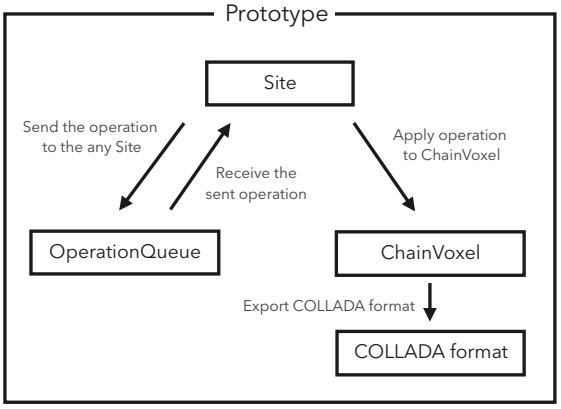

図 2 協調 3D 編集のプロトタイプの概要

それぞれの site は共有空間に ChainVoxel を持っている．また，ローカル操作の送信とリモート操作の受信のための Operation Queue を一つ持っている．ローカルおよびリモート操作は各 site で ChainVoxel に適用される．共有空間内の 3D モデルは各 site の ChainVoxel を元に表示される．

このプロトタイプはパフォーマンス評価のためだけではなく，実用的でもある．このような理由で，ChainVoxel データは COLLADA フォーマットに変換することが出来る．このフォーマットは 3D モデルの編集や閲覧のためのさまざまなアプリケーションで広く用いられている．

5.1 プロトタイプのクラス

このプロトタイプのクラス図を図 3 に示す．この図はプロトタイプの構造を表現している．各クラスの主要なメソッドを載せている．

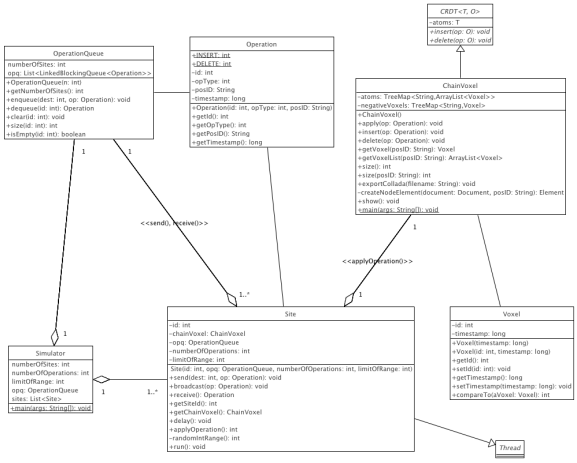

図 3 プロトタイプのクラス図

5.1.0.1 Operation

このクラスは ChainVoxel の操作を定義したクラスである．このプロトタイプでは insert と delete 操作を定義している．

5.1.0.2 OperationQueue

このクラスは各 site のための operation queue の実装である．各 site はローカル操作の送信とリモート操作の受信を queue を通して行なう．

5.1.0.3 Site

Site クラスのインスタンスはシステム内の site を表現している．インスタンス化された時に，ユニークな site 識別子 sid が割り当てられる．

5.1.0.4 Voxel

Voxel クラスは ChainVoxel 内の voxel を表現している．クラスのインスタンスは insert と delete 操作の ts，sid と $posID$ によって作成される．

5.1.0.5 ChainVoxel

このクラスは ChainVoxel データ構造の実装であり，最も重要なクラスである．site が同時に voxel を挿入したり，それらの操作を ChainVoxel に適用する．また，それを $posID$ によって自動的に分類し，それらを適切な場所内に追加する．このクラスはまた，単純なガベージコレクションの仕組みを含んでいる．これは定期的に $negativeVoxel$ の ts

より小さい値を持つ voxel をチェイン内から追い出し，それらの voxel を取り除く．

5.1.0.6 Simulator

このクラスはシナリオとパラメータを与えることでシミュレーションを動作させる．

図 4 にプロトタイプ内のクラス間の関係を表現するシーケンス図を示す．Simulator クラスはシミュレーション実行中のユーザの役割を果たしている，そしてそれは操作 *op* (i.e., insert か delete) を Site クラスに要求する．ローカル操作は各 site で直ぐに ChainVoxel に適用され，そして，他の全ての site に送信される．リモート操作は各 site に queue OperationQueue で配信される．各 site は queue からそれらを受け取り，そして自身の ChainVoxel にそれらを適用する．

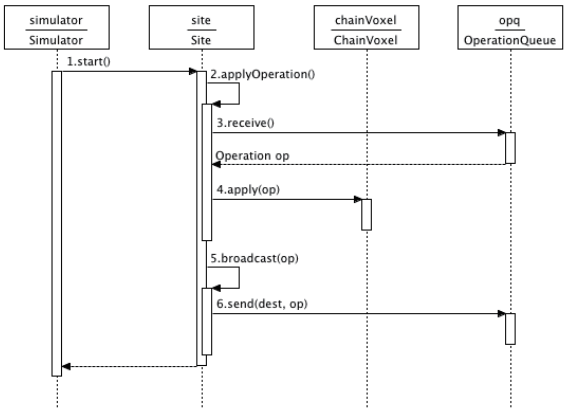

図 4 ChainVoxel に操作を適用するシーケンス図

5.2 既存の 3D ファイル形式への変換

本論文のプロトタイプは ChainVoxel 構造から COLLADA (COLLAborative Design Activity) フォーマットへのフォーマット変換器を持っている [13]．このフォーマットは多数のインタラクティブ 3D アプリケーション (e.g., Blender, SketchUp, Unreal Engine, etc.) でそれらのデータ交換として広く用いられている．

COLLADA フォーマットは以下に示される XML フォーマットのタグで 3D オブジェクトをテキストで表現しているものを含んでいる．

A voxel in COLLADA format

```
<node id="posID" name="posID" type="NODE">
  <matrix sid="transform">0.5 0 0 x 0 0.5 0 y 0 0 0.5 z 0 0 0 1</matrix>
  <instance_geometry url="#Cube-mesh"/>
</node>
```

この例は ChainVoxel 構造から voxel の COLLADA フォーマットへの変換である．このファイル変換器は COLLADA - Digital Asset Schema Release 1.4.1 [13] に基づいて実装されている．

6. 性能評価

この章では，3D モデルのための協調編集システムのプロトタイプのふるまいとパフォーマンスを議論する．初めに，実行による ChainVoxel のサイズの影響を分析する．次に，２層コミットプロトコルを用いたロックベースの実装と本論文のプロトタイプの一貫した更新にかかるまでのコストを議論する．

6.1 ChainVoxel 上の操作に関する評価

Letia らは，CRDT [11] を元にした Treedoc システムの操作数に関連する共有ドキュメントのサイズを測定した．本論文では同じ方法で本論文のプロトタイプを評価した．この章ではシミュレータ内の操作数に関連した ChainVoxel のサイズの測定を行なった．

3D 協調編集システムは都市計画，化学反応についての遠隔教育や芸術の共同作業などのような特定の目的のために基本的には用いられる．それらは明確なユースケースを持っている．そういった背景があるが，今回は任意のユースケースを想定せずに，各 site はシミュレーション上では insert と delete 操作をランダムに実行することにする．

そのうえで，本論文のシミュレーション上では各操作は他の影響を受けないことを仮定する．そして，各 site はランダムな位置に voxel のための操作 (i.e., insert と delete) を実行する．

ユーザの振る舞いは図 2 で示されたプロトタイプの *Site* クラスによってシミュレートされる．Site クラスはシミュレーション実行とシミュレーションのための設定とシナリオを読み込む．評価を行うにあたって，共有 3D 空間のサイズに関連した 3 つの異なるシナリオを扱う．$\mathbb{Z}_1^3$，$\mathbb{Z}_2^3$ と $\mathbb{Z}_3^3$ によって示され，それぞれ，$\mathbb{Z}_1 = \{x_1 | -10 \leq x_1 \leq 10\}$，$\mathbb{Z}_2 = \{x_2 | -15 \leq x_2 \leq 15\}$ と $\mathbb{Z}_3 = \{x_3 | -20 \leq x_3 \leq 20\}$ である．当然ながら，$\mathbb{Z}_1$，$\mathbb{Z}_2$ と $\mathbb{Z}_3$ は整数 $\mathbb{Z}$ の集合の部分集合である．

この章では，delete と insert 操作数による影響の解析のために 3 つのシナリオで ChainVoxel 構造の voxel 数を測定した．

図 5 に操作数に関連した ChainVoxel 内のサイズを示す．ChainVoxel のサイズ数を y 軸に示す．ChainVoxel のサイズは voxel 数で表現している．各 voxel を 1byte 相当と定義した場合，1,000 voxel は 1Kbyte に相当する．

全てのシミュレーション実行の始まりでは，ほとんどの delete 操作が失敗しているが，insert 操作は成功し続けているので voxel 数は増加を続けている．続いて，delete 操作は特定のポイントで徐々に影響を強めていく．それぞれ，$\mathbb{Z}_1^3$ では 3,000 operations，$\mathbb{Z}_2^3$ では 5,000 operations，そして，$\mathbb{Z}_3^3$ では 7,000 operations である. 図 6 はシナリオ $\mathbb{Z}_1^3$ の

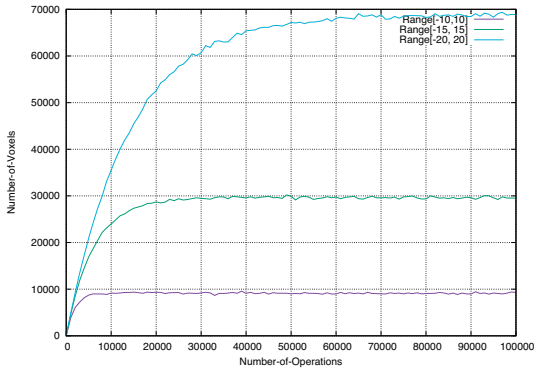

図 5 ChainVoxel のサイズ v.s. 操作数

空間での voxel の平均削除数を示している．

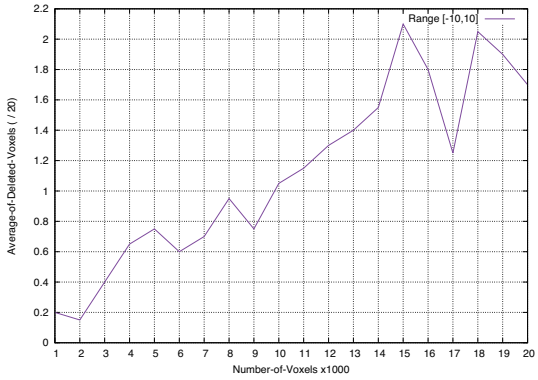

図 6 $\mathbb{Z}_1^3$ 空間での平均削除 voxel 数

voxel の増加を応じて，delete 操作の影響は大きくなっている．その結果，増加は穏やかになることで voxel 数は安定しかかっている．

6.2 二相ロックを用いたシステムとの比較

この章では，全ての site での ChainVoxel の収束までにかかるステップ数とメッセージ数において，2 相コミットプロトコルを用いたロックベースの協調編集システムと本論文のプロトタイプを比較した．ロックベースの協調編集システムはクライアント・サーバ構造を元に実装した．本論文では比較のためにそれぞれ違った内容のイベントシミュレータを用いる．

図 7 と図 8 は収束のためのステップ数とメッセージ数を示している．本論文のプロトタイプは収束のためのステップ数に関して二相ロックのシステムより 30 倍の性能差を出している．それはメッセージの計算量の点でも効率的であることを意味している．

7. まとめと今後の展望

本論文では，3D モデルのスケーラブルな協調編集システムのための ChainVoxel と呼ばれる Conflict-free なデータ構造を提案した．それは CRDT を元にし，チェインハッシュテーブル内の共有 3D モデルの操作の系列を持っている．これは複数の site による同時にデータの更新が起きた時に，排他制御を行なわずに共有データの結果一貫性特性を保証することが出来る．

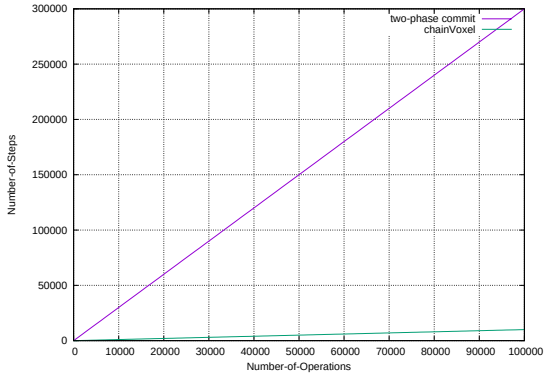

図 7 収束のためのステップ数

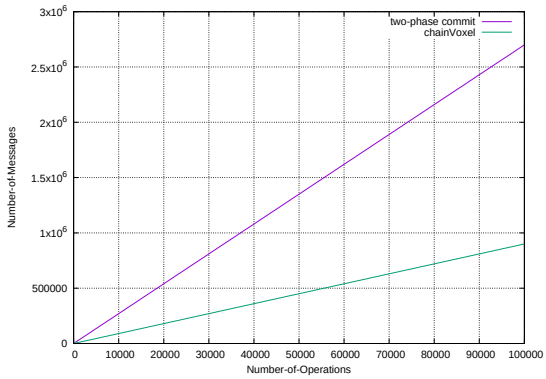

図 8 収束のためのメッセージ数

ChainVoxel は順序キューで各位置で voxel の conflict を解決することができる．本論文では 2 つのプリミティブである ChainVoxel 構造の insert と delete 操作も提案した．本論文では ChainVoxel 構造で表現された共有 3D モデルが全ての site で最終的に同じ状態に収束することを確認した．その後，本論文では，これらの操作の影響を明確にするために，シミュレーションで ChainVoxel のサイズを分析し，プロトタイプの効率性を確認するために，ロックベースの実装とプロトタイプを比較しました．

今後は，2 つの voxel との union 操作のような高レベルな操作を考えている．この操作のタイプはシステムが複雑になることが考えられる．また，ユーザの意図を保存するための，いくつかの ChainVoxel での操作の順序付けのための Operational Transformation アルゴリズムを採用する予定である．

参考文献

[1] Vidot, N., Cart, M., Ferrié, J. and Suleiman, M.: Copies Convergence in a Distributed Real-time Collaborative Environment, *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, New York, NY, USA, ACM, pp. 171–180 (online), DOI: 10.1145/358916.358988 (2000).

[2] Pacull, F., Sandoz, A. and Schiper, A.: Duplex: A Distributed

Collaborative Editing Environment in Large Scale, *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, New York, NY, USA, ACM, pp. 165–173 (online), DOI: 10.1145/192844.192900 (1994).

[3] Sun, C., Jia, X., Zhang, Y., Yang, Y. and Chen, D.: Achieving Convergence, Causality Preservation, and Intention Preservation in Real-time Cooperative Editing Systems, *ACM Trans. Comput.-Hum. Interact.*, Vol. 5, No. 1, pp. 63–108 (online), DOI: 10.1145/274444.274447 (1998).

[4] Ignat, C.-L. and Norrie, M. C.: Customizable Collaborative Editor Relying on treeOPT Algorithm, *Proc. of the Eighth European Conference on Computer Supported Cooperative Work (ECSCW'03)*, pp. 315–334 (2003).

[5] Le Nam Nguyen Hoai, X. D.: Collaborative Editing Application in Mobile Ad-hoc Networks (2012-09).

[6] Oster, G., Urso, P., Molli, P. and Imine, A.: Data Consistency for P2P Collaborative Editing, *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, CSCW '06, New York, NY, USA, ACM, pp. 259–268 (online), DOI: 10.1145/1180875.1180916 (2006).

[7] Preguiça, N., Marquès, J. M., Shapiro, M. and Leţia, M.: A commutative replicated data type for cooperative editing, *ICDCS'09*, Montréal, Canada, pp. 395–403 (online), DOI: 10.1109/ICDCS.2009.20 (2009).

[8] Ellis, C. A. and Gibbs, S. J.: Concurrency Control in Groupware Systems, *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, SIGMOD '89, New York, NY, USA, ACM, pp. 399–407 (online), DOI: 10.1145/67544.66963 (1989).

[9] Sun, C. and Ellis, C.: Operational Transformation in Real-time Group Editors: Issues, Algorithms, and Achievements, *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, CSCW '98, New York, NY, USA, ACM, pp. 59–68 (online), DOI: 10.1145/289444.289469 (1998).

[10] Li, R. and Li, D.: Commutativity-based concurrency control in groupware, *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, San Jose, CA (2005).

[11] Letia, M., Preguça, N. and Shapiro, M.: Consistency without Concurrency Control in Large, Dynamic Systems, *ACM SIGOPS Operating System Review*, Vol. 44, No. 2, pp. 22–34 (2010).

[12] Shapiro, M., Preguiça, N., Baquero, C. and Zawirski, M.: Conflict-Free Replicated Data Types, *the 13th International Symposium on Stabilization, Safety and Security of Distributed Systems (SSS 2011)*, Springer, pp. 386–400 (2011).

[13] Barnes, M. and Ellen Levy Finch, S. C. E. I.: *COLLADA - Digital Asset Schema Release 1.4.1* (2008-3).

[14] inc., G.: SketchUp.

[15] project, O.: Open3D. `https://open3dproject.wordpress.com/` accessed at January 27, 2016.

[16] Li, D. and Li, R.: An Approach to Ensuring Consistency in Peer-to-Peer Real-Time Group Editors, *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, Vol. 17, No. 5-6, pp. 553–611 (2008).

[17] Ha, Y.-U., Jin, J.-H. and Lee, M.-J.: Lets3D: A Collaborative 3D Editing Tool Based On Cloud Storage, *International Journal of Multimedia and Ubiquitous Engineering*, Vol. 10, No. 9, pp. 189–198 (2015).

[18] Kang, Y., Kim, H., Suzuki, H. and Han, S.: Editing 3D models on smart devices, *Computr-Aided Design*, Vol. 59, pp. 229–238 (2015).

[19] Aguilera, M. K., Chen, W. and Toueg, S.: Using the Heartbeat Failure Detector for Quiescent Reliable Communication and Consensus in Partitionable Networks, *Theor. Comput. Sci.*, Vol. 220, No. 1, pp. 3–30 (online), DOI: 10.1016/S0304-3975(98)00235-7 (1999).