Vol.2019-ITS-76 No.6
2019/2/28

鉄道建築限界に適した 3 次元点群データフォーマットの設計

遠山 喬[†1]　長峯 望[†1]　向嶋 宏記[†1]

概要：近年，LiDAR の高性能化・低価格化が進み，LiDAR を用いた 3 次元計測による設備管理の改善が図られている．LiDAR を鉄道の建築限界測定に適用する場合，得られる点群のデータサイズが大きいという問題がある．特に，点群データを読み書きするために必要な時間は，性能上の重要な課題である．そこで，鉄道建築限界に適した 3 次元点群のデータフォーマットを設計した．本稿では，その概要と有効性について述べる．

キーワード：LiDAR，3 次元点群処理，3 次元地図，データ圧縮，建築限界

A Data Format for Three-Dimensional Point Cloud Suitable for Railway Structure Gauging

TAKASHI TOYAMA[†1]　NOZOMI NAGAMINE[†1]
HIROKI MUKOJIMA[†1]

Abstract: In recent years, the LiDARs have been improved in performance and their prices have been reduced, and the attempts have been made to improve the facilities management using three-dimensional measurement with the LiDARs. In the case of applying the LiDARs to the railway structure gauging, there is the problem that the amount of acquired point cloud data is large. In particular, the time required to read or write the point cloud data is an important issue in their performance. Therefore, we designed a three-dimensional point cloud data format suitable for railway structure gauging. In this paper, we present its outline and its effectiveness.

Keywords: LiDAR, 3-D Point Cloud Processing, 3-D Map, Data Compression, Structure Gauge

1. はじめに

近年，自動車の自律走行技術の研究開発が活発化しており，これに伴い測域センサや LiDAR (Light Detection and Ranging) と呼ばれるレーザを用いた3次元計測センサの高性能化と低価格化が進んでいる．このため，自動車分野に限らず，様々な分野で 3 次元計測を用いた設備の保守・管理の効率化・高度化が試みられている．この流れの中で，我々は，鉄道の建築限界の測定に LiDAR を応用する研究開発を行っている．

鉄道の建築限界は，列車運行の安全を確保するために，建造物等が支障してはならない軌道周辺の領域として定められている．図 1 に JR 在来線の建築限界の例を示す．建築限界の支障は，軌道や沿線設備の位置が変化することで発生しうるため，定期的な測定・管理が必要となっている．そこで，建築限界測定の低コスト化と設備管理の高度化を目的として，我々は LiDAR を用いた車上検測を提案している[1]-[3]．図 2 に提案システムの外観を示す．提案システムでは，LiDAR を複数台設置することにより，測定範囲の拡大と，空間分解能の向上を図っている．

提案システムで得られるデータは，反射レーザ受信強度の情報を含む 3 次元点群である．図 3 に，提案システムで

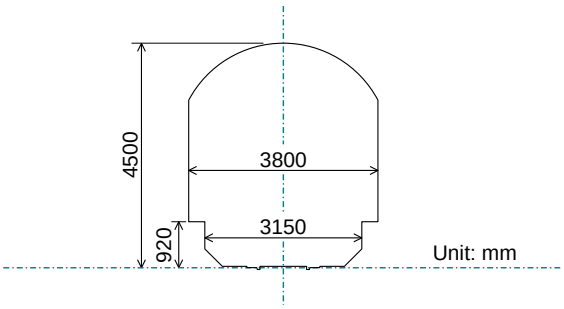


図 1　JR 在来線の建築限界の例
Figure 1　An example of the structure gauge on Japanese railways.

図 2　LiDAR を用いた建築限界車上検測システム
Figure 2　On-board structure gauging system with LiDARs.

†1 （公財）鉄道総合技術研究所
Railway Technical Research Institute

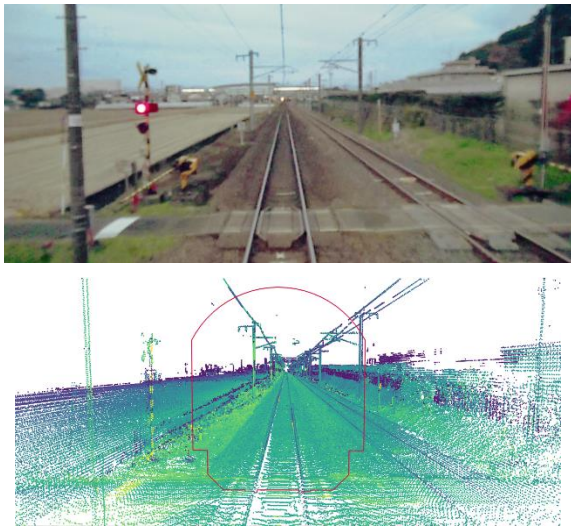


図 3　前方映像と測定 3 次元点群
Figure 3　The cab view image and the measured 3-D point cloud.

取得した点群の 3 次元表示の例を当該箇所の前方映像とともに示す．図 3 の 3 次元点群では，受信強度が低い点を暗い色，受信強度が高い点を明るい色として示している．

提案システムでは，設計上，毎秒 17 万点以上の計測が可能であり，1 時間の計測における点群の規模は 6 億点を超える．また，計測時の車両速度は最高で 80 km/h に対応しており，停車や加減速を加味しても，1 時間の計測で得られる点群が数十 km 以上の広範に及ぶ．このような，データ量および空間的大きさの両方の観点で大規模な点群をどのように処理するかは，実用上の大きな課題と言える．特に，点群データの読み書きに要する時間は，点群処理全体の処理性能を大きく制限するため，重要な問題と言える．

しかし，3 次元点群のデータフォーマットは幾つか存在するものの，3 次元地図等の応用分野において，特定の製品に依存せず，オープンな標準として確立したものは存在しないと我々は認識している．また，既存の 3 次元点群データフォーマットは，必ずしも鉄道における建築限界管理の用途に固有の要件を満足していない．

そこで，鉄道の特性を活用し，鉄道の建築限界管理用途に固有の要件を満足する 3 次元点群データフォーマットを設計した．本稿では，データフォーマットの考え方とその有効性について述べる．

2. 鉄道建築限界における要件

2.1 鉄道の特性

鉄道の線路は，1 方向に長いという特徴を持つ．また，線路の分岐箇所は原則として駅構内に限られ，局所的である．これは 2 次元的なメッシュ構造を持つ一般の道路とは大きく異なる点である．この特徴のため，鉄道沿線設備の多くはキロ程で位置管理されており，絶対的な地理座標（緯度，経度）より，レールを基準とした相対座標の方が重要な意味を持つ．建築限界の管理は，キロ程とレールを基準とした相対位置を用いる典型例である．

鉄道車両は，一般的な乗用車と比較して車高が高い．また，鉄道車両と自動車ではサスペンションの構造が大きく異なる．これら要因のため，鉄道車両の車体に LiDAR を設置して 3 次元計測を行った場合，車体動揺，特にロール方向の回転の影響を強く受ける．したがって，3 次元計測データの高度な活用のためには，車体動揺補正が不可欠と言える．一方で，鉄道車両には自動車のようなステアリング機構はなく，車両は必ずレールに沿って走行するため，車体動揺補正にレール位置を利用することができる[2]．左右レールの間隔（軌間）はほぼ一定であり，場所によって様々な形態をとりうる道路の白線や縁石とは異なる．

2.2 座標系に関する要件

ミラー等によりレーザビームの方向を変化させる LiDAR では，その測定データはセンサ位置を中心とした極座標で表現される．一方，鉄道建築限界の評価を行う上では，点群データはレール長手方向（キロ程方向），マクラギ方向（水平方向），鉛直方向を軸とした直交座標で表現されているのが合理的である．また，複数台の LiDAR の測定データを統合する上でも，共通の座標系として直交座標が適している．

極座標から直交座標への変換は，理想的には LiDAR の設置位置と車両位置・速度に基づき，比較的少ない計算負荷で実行可能である．しかし，実際には車体動揺の影響が無視できないため，極座標から直交座標へ変換し，動揺量を算出し，LiDAR の設置位置に動揺量を加味した上で，改めて極座標から直交座標へ変換するという，少なくとも 2 パスの処理が必要となる．特に，動揺量の算出は，点群からのレール抽出といった比較的計算負荷の大きい処理を含むため，データを参照する度に座標変換を行うのは非効率である．したがって，鉄道建築限界用の 3 次元点群データフォーマットでは，点群は動揺補正後の直交座標形式で保管されていることが求められる．

なお，具体的な座標系については，キロ程増加方向を X 軸正方向，マクラギ方向（水平方向）を Y 軸，鉛直上向きを Z 軸正方向とした右手系を採用する．これは鉄道においてはキロ程を第 1 軸（X 軸）とするのが合理的であること，点群処理の代表的なソフトウェアライブラリである Point Cloud Library (PCL)[4][5]が右手系を標準としていることによる．また，提案システムでは点群を構成する各点が反射レーザ受信強度の情報を持っており，受信強度の情報は，形状だけでは識別が困難な物体の識別を補助する効果があることから[3]，データフォーマットでは，XYZ 直交座標に受信強度 (I: intensity)を加えた座標系を扱えるものとする．

2.3 座標の数値に関する要件

鉄道建築限界の管理が mm 単位で行われていること，LiDAR 単体の測距精度について数十 mm 以下が実現されていることから，特にマクラギ方向・鉛直方向の座標は 10 mm 以下の分解能で表現できることが求められる．

また，断面方向（Y-Z 平面）については，建築限界の評価を行う上で，±5 m 程度の範囲の座標が扱える必要がある．一方，現状で普及しているクラスの LiDAR の測定可能範囲は数十 m に及ぶため，鉄道部外の周辺設備や地形等を含む全測定データを保存する場合は，相応の範囲に対応する必要がある．

キロ程方向（X 軸）については，測定車両（非旅客営業車両を想定）が駅を発車し，次に駅に到着するまでの 1 行程の測定データを 1 単位として扱えることが望ましい．具体的な距離は線区により様々であるが，50 km 分に対応していれば十分と考える．

2.4 その他の要件

近年，HDD (Hard Disk Drive)より高速な大容量ストレージとして SSD (Solid State Drive)の普及が進んでいる．ただし現時点では依然として HDD の方がビット単価が安いこと，点群データは書き込み（低速）より読み込み（比較的高速）の機会の方が多いという非対称性から，点群データの保存は HDD に対して行う前提とする．

測定データの長期保管を考えた場合，データの破損検知が行えることが望ましい．また，データ破損時の影響範囲は可能な限り小さい方が良い．

3. データの実態調査

3.1 調査対象

データフォーマットを設計するにあたり，取得点群データの実態調査を実施した．鉄道沿線の設備や環境は多様であるため，提案システムで計測したデータのうち，特徴の異なる 4 つのシーンを選定した．表 1 に選定した各シーンの単線・複線区分，電化方式，駅・高架橋・トンネル有無を示す．シーン I は都市部の高架駅周辺，シーン II は田畑の多い郊外，シーン III は切土区間を含む山間部，シーン IV はトンネル区間（一部切土）である．いずれのシーンもキロ程方向に 2 km の範囲に含まれる点群を抽出した．

表 1　調査対象シーンの概要
Table 1　Summary of target scenes.

Scene	Track	Electrification	Station	Viaduct	Tunnel
I	Double	AC 20 kV	✓	✓	–
II	Double	AC 20 kV	–	–	–
III	Single	None	✓	–	–
IV	Single	AC 20 kV	–	–	✓

調査としては，各シーンについて，点群に含まれる点の空間分布を評価する．評価方法として，大きく二つの方針が考えられる．一つはセンサで取得した全データの分布を評価するものである．この場合，実際的評価である反面，点の分布がセンサ配置等の実装に強く依存する．もう一つは，点の密度を均質化して評価するものである．ここでは，より汎用的な評価を行うため，後者の方法を採用した．均質化の方法は，空間をレール長手方向，マクラギ方向，鉛直方向それぞれ 0.125 m 単位に分割し，1 つの立方体（ボクセル）に複数の点が含まれる場合に 1 点のみを代表として抽出するものである．もっとも，この方法によっても，センシング範囲外である車両上方のデータが欠落しているように，センサ配置等への依存性は完全には排除できない．ここでは，定性的な傾向の評価を行う．

3.2 空間分布の調査結果

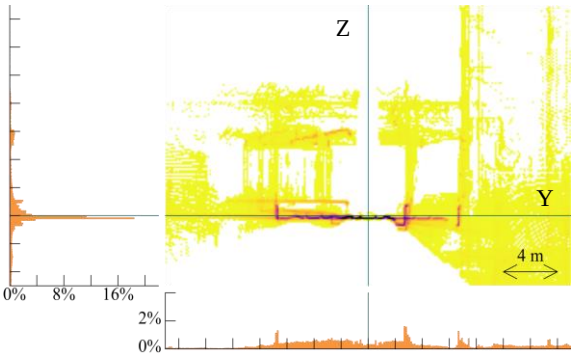
図 4 に各シーンの点の空間分布を示す．図 4(a)〜(d)の各図は 3 つのグラフから構成される．中央（右上）のグラフは，Y-Z 平面を示したものである．Y-Z 平面の各ピクセルはレールに沿った 2 km × 0.125 m × 0.125 m の領域（8,000 ボクセル）における，点を 1 つ以上含むボクセルの割合を色の濃淡で示している．左側のグラフは点の Z 座標（鉛直方向）の分布を，下側のグラフは点の Y 座標（マクラギ方向）の分布を示している．なお，中央のグラフは空間（体積）で正規化しているのに対し，座標の分布は図 4 の表示範囲外に存在する点も含めた全点数で正規化している．

図 4(a)は駅のプラットホームや高架橋の側壁が測定されていることを示している．図 4(a)の Y 座標の分布では，側壁の部分の密度が高い．図 4(b)では電化柱が測定されているものの，Z 座標の分布では，大部分がレールとバラスト道床の位置に集中しており，レールレベルより上方の点は相対的に少ない．図 4(c)では樹木が測定されており，上方の点が他のシーンと比較して多い．また，Y 座標の分布の対称性が高い．図 4(d)ではトンネル壁面が明瞭に測定されており，レールレベルより上方の Z 座標の分布は一様分布に近く，Y 座標の分布は分散が少ない．

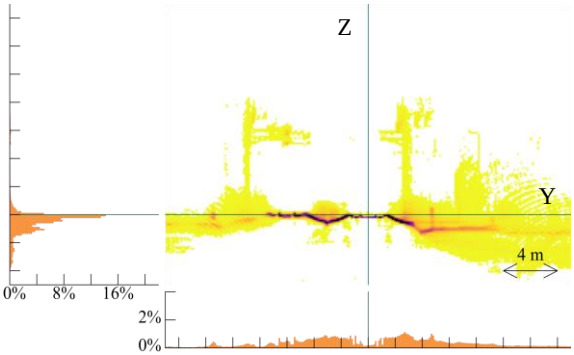
このように，シーンによって点の空間分布に違いが確認された．

3.3 空間分布に関する考察

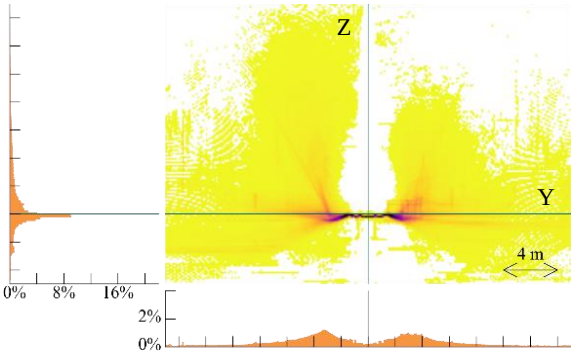
図 4 より，特に明かり区間とトンネル区間では，点の空間分布の傾向が異なることが分かる．ただし，点の大部分はレールレベルから 6 m 以下に集中していることは，明かり区間についても同様である．また，トンネル区間も含め，点の分布が道床周辺に集中している点は共通している．このため，道床の種類（バラスト，コンクリートスラブ等）によって道床周辺の設備の形状は異なるものの，全体として Y 座標の分布はおおよそレール中心に対し左右対称になると考えられる．ただし，図 4(a)の高架橋のように，長大



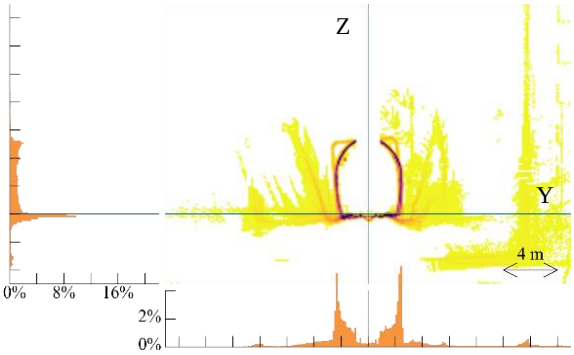
(a) Scene I



(b) Scene II



(c) Scene III



(d) Scene IV

図 4　鉄道沿線点群の空間分布
Figure 4　Spatial distributions of trackside point clouds.

構造物を含む複線区間については，Y 座標の分布はレール中心に対し左右非対称となると考えられる．

4. データフォーマットの設計

4.1 基本方針

鉄道建築限界管理で重要なキロ程方向の検索を高速化させるため，点群はキロ程（X 座標）で昇順ソートを行う方針とした．また，ランダムアクセスを高速化させるため，キロ程方向について点群を分割し，断片毎に索引（インデックス）を設けることとした．すなわち，分割した断片（部分点群）を 1 つの仮想的なファイルとし，それらをアーカイブすることにより 1 つのファイルにまとめるという形態をとる．

また，データサイズを削減するため，データ圧縮を行う．データ圧縮は，分割された個々の部分点群に対する 1 次圧縮と，アーカイブ時の 2 次圧縮の 2 段階を用意する．1 次圧縮および 2 次圧縮は，ディスク使用量の節約とディスクアクセス時間の短縮による高速化を目的としている．ただし，1 次圧縮は，メモリ上のデータサイズを小さくすることで CPU のキャッシュ利用率を向上させ，処理の高速化を実現することも目的としている．この目的の違いから，1 次圧縮に用いる技術は，展開に必要なメモリ容量が少なく，展開が高速に行えるものを選定する．

部分点群の符号化方法は，明かり区間やトンネル区間等のシーンの違いを区別せず，単一とする．シーンに符号化方法を適応させることはデータ圧縮率の向上に有効と考えられるが，仕様の複雑化を避ける方を優先した．

4.2 アーカイブ形式と分割単位

アーカイブ形式には ZIP フォーマット，特にそのサブセットにあたる ISO/IEC 21320-1:2015 に準拠することとした．ZIP フォーマットは多くの OS やフレームワークでサポートされている利点がある．

ZIP フォーマットは，アーカイブ内のファイルに対し，Deflate 圧縮(RFC1951 [6])を適用することができる．そこで Deflate 圧縮を 2 次圧縮に用いる．Deflate 圧縮は，辞書ベースの圧縮方式であり，展開時にメモリを消費するが，比較的優れた圧縮率を実現する．2 次圧縮の適用有無は部分点群単位で選択可能である．

また，ZIP フォーマットではファイル単位，すなわち部分点群単位で CRC によるデータ破損の検出が可能である．このため，データ破損の影響範囲をキロ程方向の分割単位程度に限定することができる．

キロ程方向の分割は，検索・ランダムアクセス性能や扱いやすさ，データ破損時の影響範囲等を総合的に勘案し，1 m 単位とした．旧来からの ZIP フォーマットでは，アーカイブ中のファイル数の上限が 65535 であるが，この場合であっても 65 km 分のデータが扱えるため，実用上十分と言える．それ以上の距離範囲を扱う場合は，ZIP フォーマットの拡張である ZIP64 フォーマットを用いることで対応可能である．

部分点群は，ファイル名を“線区 ID/kkkk/mmm”（kkkk:キロ程の km 単位部分を表す 4 桁の数字, mmm:キロ程の m 単位部分を表す 3 桁の数字）とすることで，検索を容易にする．また，線区 ID により区別することで，一つのアーカイブ内に複数の線区のデータを入れることが可能である．この仕組みを応用することで，同一線区であっても駅の番線を識別したり，重複キロ（同一線区の異なる地点に同一のキロ程が割り振られた状態）に対応したりもできる．

4.3 部分点群の符号化と圧縮

データフォーマットの設計において，1 m 単位で分割された部分点群を，どのように符号化し，データサイズを削減するかが，性能上の要となる．

まず，各点の座標の表現は整数列とする．これは，座標値の桁数の分散が大きくなく，浮動小数点数とすると指数部のデータが冗長になるためである．また，1 次圧縮に整数列圧縮の技術を適用し，データサイズを削減する狙いがある．なお，実際に点群処理を適用する際には，整数から浮動小数点数への変換が必要になる場合があるが，近年の CPU では十分高速に変換・スケーリングすることができる．分解能は要件に基づき 1 mm とした．

前述のとおり，1 次圧縮として整数列圧縮の技術を用いる．整数列圧縮は，小さな数に短い符号語を，大きな数に長い符号語を割り当てることで，データサイズを削減することを基本とする．ゴロム・ライス符号やガンマ符号等，1960～70 年代から提案・利用されている技術であるが，近年では CPU のキャッシュ利用効率が処理性能に多大な影響を与えるようになったことから，データベースエンジンの高速化技術等で再注目されている．

整数列圧縮の手法は多数提案されており，文献[7]に多くの手法とそれらの性能が整理されている．広く利用されている手法に Variable Byte が挙げられるが，Variable Byte に代表される 1 数値をバイト単位で符号化するバイト指向の手法は，本用途においては非効率である．これは，水平方向(Y) 座標，鉛直方向(Z)座標の大部分が 1 バイトでは表現できず，2 バイト必要となるためである．1 数値あたり 2 バイト使用するのであれば，無圧縮の整数（±32,767 mm が表現可能）で十分である．このため，ビット幅等の符号化方法を 1 数値単位ではなく，多数の数値をまとめて指定する Frame-Of-Reference (FOR)の考え方が有効と考え，ここではその 1 手法である SIMD-FastPFOR [7]を選定した．

SIMD-FastPFOR は，PFOR (Patched FOR)の派生手法である．PFOR の基本概念は，1 ブロック（例えば 128 個）の数値を共通の n ビット幅で符号化し，例外的に n ビットに収まらなかった数値の位置（インデックス）と超過量を別途符号化するというものである．PFOR は，本用途の座標値のように，数値の桁数（ビット幅）の分散が少ない数列においては，比較的高い圧縮率が期待できる．圧縮率と展開速度の間には概してトレードオフの関係があるが，SIMD-FastPFOR では，CPU の SIMD (Single Instruction Multiple Data)命令を用いたベクトル化（並列化）により，高い展開速度を実現している．ここでは 1 ブロックの数値の個数を 128 とした．図 5 に，128 個の 10 ビット幅の数値を SIMD 最適に配置した例を示す. なお, 数値の個数が 128 の倍数でない場合, 端数個の数値は SIMD-FastPFOR では効率的に符号化できない．このため，端数個の数値については別の整数圧縮手法である Simple-16[8]を用いて符号化することとした．

Simple-16 は，Simple-9, Simple-8e 等の Simple ファミリと呼ばれるビットパッキング手法の 1 つである．32 ビット整数を 1 単位として上位 4 ビットに格納方法（16 通り）,下位 28 ビットに実際の値を格納する．例えば 0~3 の範囲の数値であれば 1 単位に 14 個格納できる．

SIMD-FastPFOR, Simple-16 を含め，多くの整数列圧縮手法においては，負数を効率的に符号化できない．そこで，データを Y-Z 平面の 4 象限で区分する方法を採用した．符号の情報を各象限に含まれる点の数（これも非負数である）に集約し, Y 座標, Z 座標それぞれの絶対値をとることで，整数列圧縮が可能な非負数の列を得ることができる．ただし, この弊害として, X 座標の昇順ソートが不完全となる．これについては，象限単位ではソート済みであることを利用し，展開時にマージソートにより高速にソートすることで対応する．もっとも，応用上 1 m の範囲内でソートが不完全であっても問題ない場合は，マージソートは省略できる．また，X 座標の列は単調増加となるので，その差分（増加量）も非負数となる．このため差分符号化により圧縮率

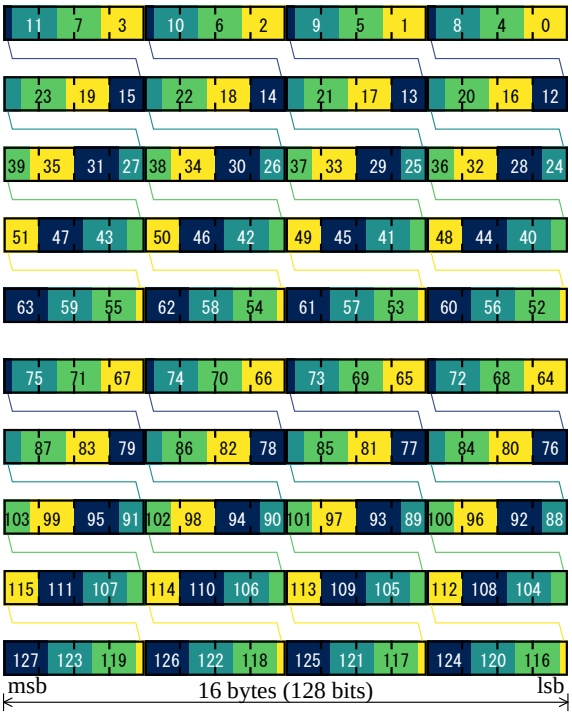


図 5　SIMD 最適な 128 個の数値配置

Figure 5　Packed 128 integers in a SIMD suitable form.

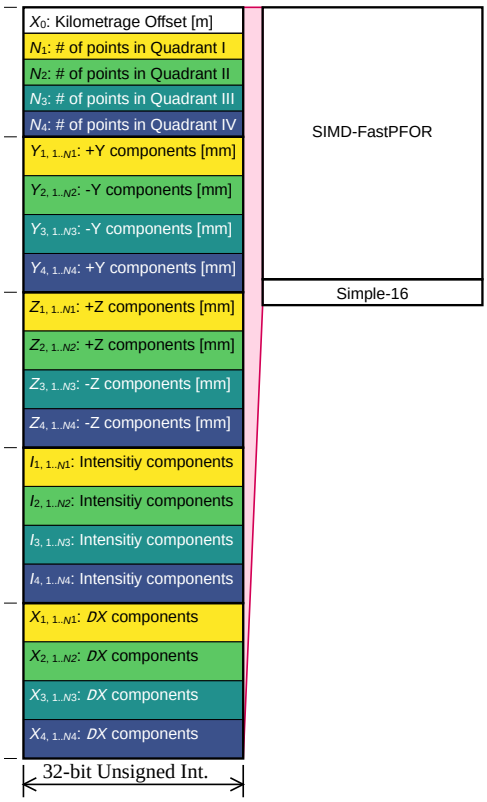


図 6　部分点群の構造
Figure 6　Structure of a fragmented point cloud.

の向上が可能である．

図 6 に部分点群の構成を示す．各構成要素の順序は，実態調査で得られた座標の分布を踏まえ，整数列圧縮の効率が高くなるよう制定した．特に X 座標の差分は 2 ビット以下の数値が連続しやすく，Simple-16 との相性が良いことから最後に配置した．また，各象限の格納順序については，Y 座標の分布は対称性が高いと考えられることから，第 1 象限，第 2 象限，第 3 象限，第 4 象限の順とした．これにより，相対的に対称性の低い Z 軸について反転を 1 回（第 2 象限から第 3 象限）とし，Y 軸と Z 軸の両方が反転すること（例えば第 2 象限から第 4 象限）を回避している．

4.4　メタデータ

アーカイブの中にどのようなデータが保存されているか，個々のファイル（部分点群）を展開する前に把握可能であれば，不要な展開処理を行わずに済む．また，どのように点群を取得したか等の情報は点群自体には含まれていない．そこで，それらメタデータをアーカイブの先頭に記録することとした．ただし，メタデータとして何が有用であるかは，実際に提案フォーマットを運用していく中で発見されていくと考えられるため，メタデータの符号化方法は，柔軟に拡張可能な JSON(RFC 8259 [9])とした．

現時点ではメタデータとして，測定日時，ヒューマンリーダブルな線区名，アーカイブ内限定の線区 ID について規定している．

5.　提案フォーマットの評価

5.1　評価方法

実際の鉄道沿線点群データを，提案フォーマットを含む後述の複数のデータフォーマットに変換し，データサイズ，シーケンシャルアクセス時間，ランダムアクセス時間を比較し，提案フォーマットの有効性を評価する．

対象データは，営業線で取得したキロ程 10k000m～15k000m の 5 km 分のデータ，全 27,573,518 点である．当該区間は複線電化であり，データはそのうちの片側の線路（上り線）のものである．また，当該区間は明かり区間であり，一部高架区間と地上駅 1 つを含む．

シーケンシャルアクセス時間については，キロ程 10～15 km の全データを読み込み，軌道中心線上の高さ 2 m の位置（おおよそ車体の中心が通る位置）に最も近い点の座標を探索する時間を評価する．

ランダムアクセス時間については，キロ程 10k000m～10k010m，12k000m～12k010m，14k000m～14k010m の 3 区間（各 10 m）のデータを読み込み，各区間内で軌道中心線上の高さ 2 m の位置に最も近い点の座標を探索する時間を評価する．

5.2　対象データフォーマットと実装

提案フォーマットと PCL の標準データフォーマットである PCD (Point Cloud Data)フォーマット[11]の大きく 2 種類について比較を行う．

提案フォーマットについては，部分点群の 2 次圧縮に Deflate 圧縮を適用した場合と，無圧縮の場合の 2 条件を用意した．Deflate 圧縮のアルゴリズムおよびその実装には，Google 社が開発した Zopfli[10]を用いた．Zopfli は反復的に符号化戦略を模索することで，高い圧縮率を実現する．圧縮に時間を要する一方，展開に要する時間は一般的な Deflate 圧縮の場合と変わらないため，書き込みより読み込みの頻度が高い用途に適する．本評価では，反復回数は 10 回とした．

PCD フォーマットは，ファイルに含まれる点の形式や点数を記した 10 行程度のテキストヘッダの後に，点群データを羅列したシンプルなフォーマットである．点群データの符号化方法として，ASCII テキスト，無圧縮バイナリ，圧縮バイナリ（LZF 圧縮）の 3 つが存在する．

PCD フォーマットにおいては，点群中の各点の順序に関して規定や制限は無いが，提案フォーマットと条件を揃えるため，いずれもキロ程（X 座標）について昇順ソートを行った．また，ASCII テキストの場合，同一の数値であっても表現方法は複数存在する．PCL 1.8.1（フォーマット設計当時の最新版．現時点での最新版は 1.9.1）の標準実装では，冗長な表現が採用されるため，本評価では小数点以下，すなわち，1 m 未満の端数については可能な限り短い表現（例：-3140.0 mm→-3.14）となるように独自の保存プログ

ラムを実装した.

また, PCL 1.8.1 の PCD フォーマットの読み込み機能は, シーケンシャルアクセスを前提としており，部分読み込みに対応していない．一方，本評価では PCD フォーマットにおいても点群はソート済みのため，ASCII テキストおよび無圧縮バイナリでは部分読み込みを行うよう独自のプログラムを実装した．なお，点群はソート済みであっても等間隔ではないため，二分探索で読み込み範囲を特定するようにした. PCD の LZF 圧縮についてはランダムアクセスに対応していないため，これについては，PCL 1.8.1 の標準機能で全体を読み込んだ後に, 部分点群を抽出する方式とした.

5.3 実行・コンパイル環境

評価の実行環境には Windows 10 Pro (バージョン 1803)の PC を用いた．CPU は Intel Core i7 8700 で，L1 データキャッシュが 6×32 KiB，L2 キャッシュが 6×256 KiB，L3 キャッシュが 12 MiB である．RAM は 8 GiB (DDR4)である．点群データを保存したストレージは，回転数 7200 rpm の 3.5 インチ HDD で，転送モードは SATA/600 である．本評価とは別に実施した HDD のベンチマークでは，シーケンシャルリード 186 MiB/s，ランダムリード 1.5 MiB/s 程度であった.

評価に用いるソフトウェアは Microsoft Visual C++ 2017 でコンパイルを行った．コンパイルオプションとして Intel 製 CPU の SIMD 命令セットである SSE2 使用を有効とした．実行環境は 64 ビット OS であるが，互換性の観点から実行ファイルは 32 ビットバイナリ(x86)とし, より新しい SIMD 命令セットである AVX/AVX2 は不使用とした.

5.4 評価結果

以下の 5 フォーマットについて，データサイズ，シーケンシャルアクセス時間，ランダムアクセス時間の性能を評価した結果を図 7 に示す.

- Proposed (w/ Deflate): Deflate 圧縮の提案フォーマット
- Proposed (w/o Deflate) : 2 次無圧縮の提案フォーマット
- PCD (Compressed) :圧縮バイナリ PCD フォーマット
- PCD (Raw Binary) :無圧縮バイナリ PCD フォーマット
- PCD (ASCII) : ASCII テキスト PCD フォーマット

図 7(a)は, 5 km 分の全データのファイルサイズを示している．図 7(b)，図 7(c)は，読み込みに要した時間と，最接近点の探索に要した時間を示す．なお，提案フォーマットについては，読み込みに要した時間を，ディスクから符号化された部分点群に相当する領域を読み込み，（必要により）Deflate 圧縮を展開する時間(Unzip)と，部分点群の整数列圧縮を展開し点群を復元する時間(Decode)，部分点群を結合する時間(Concat)に分けて示している.

図 7 が示す通り，提案フォーマットは，無圧縮バイナリ PCD フォーマットと比較して，データサイズについて 27%以下，読み込み時間について 55%以下であった．なお，2 次無圧縮の提案フォーマットについては，1 m あたり（部分点群 1 個あたり）で平均を取ると，点数約 5,500 点に対し，データサイズ 23 KiB 程度となる．実際，5,000 個の部分点群のうち 4,991 個 (99.8%)のデータサイズは 32 KiB 未満であった.

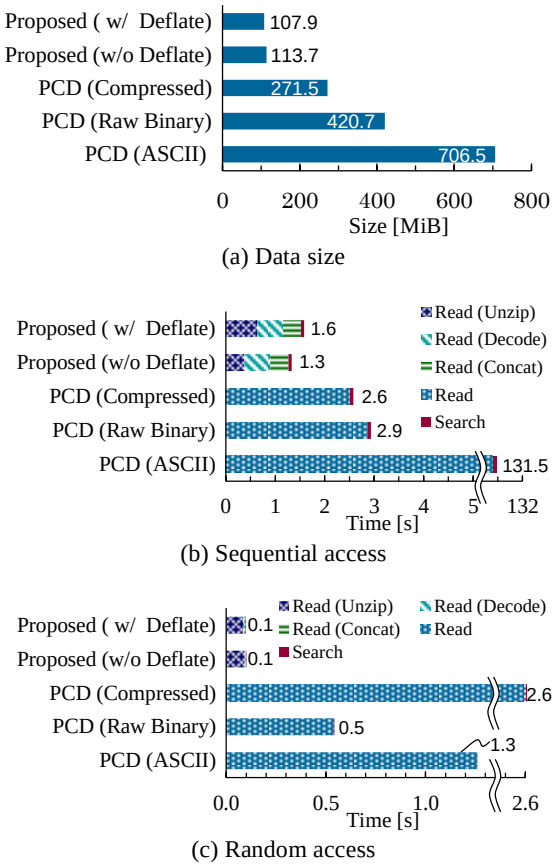


図 7　各フォーマットの性能比較

Figure 7　Performance comparison of data formats.

5.5 考察

図 7 より，データサイズ，読み込み時間の両面で，提案フォーマットは PCD フォーマットより優れていると言える. 本評価では PCD 以外のフォーマットに対する優位性については不明であるが，PCD の 3 つの形態のいずれか，あるいは組み合わせの派生形のフォーマットに対しては，優位性を有すると考える.

図 7(b), 図 7(c)が示すとおり, シーケンシャルアクセス，ランダムアクセスともに処理時間に占める探索の時間の割合は極めて小さい．本評価の探索処理は，実際の建築限界支障判定処理より単純化されているものの，メモリ上の数値演算処理とディスクからの読み込みでは，後者の方が処理速度上制約になっていることを示唆している．実際，ASCII テキスト PCD フォーマットについては，5 km 分の点群を読み込むのに 2 分以上を要しており，データサイズだけでなく処理速度の観点からも，大規模点群を扱う場合に ASCII テキストが適していないことを端的に示している.

圧縮バイナリ PCD フォーマットについては, 無圧縮バイ

ナリ PCD フォーマットに対してデータサイズが 65%となっており，圧縮の効果が確認できる．データサイズが小さいため，ディスク読み込みの時間が短縮されており，展開処理の時間が追加で発生しているにもかかわらず，無圧縮バイナリ PCD フォーマットよりシーケンシャルアクセスでの読み込み時間が少ない．一方，ランダムアクセスについては，ASCII テキスト PCD フォーマットを超える時間を要しており，ランダムアクセスに対応すること（部分点群に分割すること）の重要性を示している．

提案フォーマットにおける Deflate 圧縮（2 次圧縮）の有効性については，データサイズと速度のトレードオフが見られる．データサイズについては Deflate 圧縮により 5%の削減効果があるが，無圧縮バイナリ PCD フォーマットに対する圧縮バイナリ PCD フォーマットほどの効果は無い．視点を変えれば，一般に圧縮率では不利な整数列圧縮が，辞書ベースの圧縮に匹敵する高い効率を示しているとも言える．図 7(b)が示すように大規模点群を扱う場合は，Deflate の展開処理の時間の分だけ読み込み時間が増大するものの，10 万点規模の点群であればその差は無視できる．

なお，提案フォーマットの性能が優れている理由としては，SIMD-FastPFOR が高効率であるのに加え，キャッシュ利用効率が高いことが背景にあると考えられる．整数列圧縮後の部分点群のサイズは，数値上はおおよそ L1 データキャッシュに収まる．整数列圧縮の展開後は 5,500 点×4 次元（XYZI）×32 ビットで 86 KiB 程度となるため，L1 データキャッシュには収まらないものの，L2 キャッシュには収まる．

一方で，提案フォーマットのシーケンシャルアクセス時は，結合処理(Concat)の時間が無視できない結果となった．これはランダムアクセス時の使い勝手を考慮し，部分点群の展開を終えてから結合する実装としたことに起因する．展開済み部分点群を全体点群に逐次追記する方式にすることで，結合時間は，実質的にゼロに低減可能と考えられる．

6. おわりに

鉄道建築限界管理における大規模点群の処理性能を改善するため，鉄道の特性を考慮した 3 次元点群フォーマットの設計を行った．提案フォーマットを実際の測定データに適用したところ，無圧縮バイナリ PCD フォーマットと比較して，データサイズについては 27%以下，読み込み時間については 55%以下に低減可能であることが示された．提案フォーマットを活用することで，建築限界管理をはじめとする鉄道への LiDAR 応用がより容易かつ高効率に実現できると期待する．

一方で，独自規格，競合規格の乱立は望ましくないと考える．本稿で示した鉄道の特性の一部は，自動車道路等，他分野とも共通すると考えられる．部分点群に分割する考え方や，整数列圧縮の適用については，他分野にも応用できる可能性がある．鉄道に限定せず，分野横断的に 3 次元点群を効率的に扱えるようになることを期待している．

参考文献

[1] 遠山喬ほか. 測域センサを用いた建築限界判定装置と管理システムの開発. 鉄道総研報告. 2018, vol. 32, no. 5, p.11-16
[2] 大森達也ほか. 建築限界判定装置の開発に向けた車体動揺補正の検討. 平成 30 年電気学会産業応用部門大会講演論文集. 2018 , no. 5-34, p. 285-288
[3] 遠山喬ほか. 鉄道沿線 3 次元点群データからの設備抽出支援手法. 電気学会研究会資料 交通・電気鉄道 フィジカルセンサ合同研究会. 2018, TER-18-32, p. 39-44
[4] “PCL - Point Cloud Library (PCL)”. http://pointclouds.org/, (参照 2019-02-01).
[5] “Point Cloud Library (PCL),” https://github.com/PointCloudLibrary/pcl, (参照 2019-02-01).
[6] “DEFLATE Compressed Data Format Specification version 1.3”. https://tools.ietf.org/html/rfc1951, (参照 2019-02-01).
[7] Lemire, D. and Boytsov, L.. Decoding billions of integers per second through vectorization. Software Practice & Experience 45 (1), 2015
[8] Zhang, J., Long, X. and Suel , T.. Performance of Compressed Inverted List Caching in Search Engines. Proc. 17th International Conference on World Wide Web, 2008, p. 387-396.
[9] “The JavaScript Object Notation (JSON) Data Interchange Format”. https://tools.ietf.org/html/rfc8259, (参照 2019-02-01).
[10] “Zopfli Compression Algorithm is a compression library programmed in C to perform very good, but slow, deflate or zlib compression”. https://github.com/google/zopfli, (参照 2019-02-01).
[11] “The PCD (Point Cloud Data) file format”. http://pointclouds.org/documentation/tutorials/pcd_file_format.php, (参照 2019-02-01).