情報処理学会論文誌　コンシューマ・デバイス & システム　Vol.7　No.2　76–86 (May 2017)

コンシューマ・システム論文

マルチストリーミングセンサデータ向けリアルタイム空間補間可視化システム

若森 和昌[1,a)]　丸島 晃明[1,†1]　峰野 博史[2,b)]

受付日 2016年9月30日, 採録日 2017年2月27日

概要：無線センサネットワークの普及にともない，様々なセンサデータが収集可能となっただけでなく，人間がセンサデータを視覚的に理解することを目的としたセンサデータ可視化システムの開発もさかんに行われている．なかでも空間補間を用いた可視化は，無線センサネットワーク設置空間の視覚的理解をさらに促進する技術として注目されている．しかし，空間補間を用いた可視化の処理量は膨大であることから，既存システムでは，加速度データ等の高レートな複数のストリーミングセンサデータを，リアルタイムに空間補間し可視化することは難しかった．本研究では，高レートに生成される複数のストリーミングセンサデータに対しても，リアルタイムに空間補間して可視化できるシステムを提案する．本システムは，空間補間を並列処理し，WebGL を用いて描画することで，可視化に要する処理時間を短縮し，リアルタイムな空間補間と可視化を実現する．プロトタイプとして 18 台の無線加速度センサノードを用いた振動可視化システムを開発し基礎評価を行ったところ，18 台から各 60 Hz で生成されるストリーミングセンサデータをフレームレート 60 fps で空間補間し可視化可能なことを確認した．また，吊橋における屋外評価において振動の伝播を空間補間して可視化できることも確認でき，手軽に建造物の振動分析ができる見通しを得た．

キーワード：リアルタイム可視化，空間補間，ストリーミングセンサデータ，Web システム

Real-time Spatial Interpolation Visualization System for Multi-streaming Sensor Data

Kazumasa Wakamori[1,a)]　Koumei Marushima[1,†1]　Hiroshi Mineno[2,b)]

Received: September 30, 2016, Accepted: February 27, 2017

Abstract: Wireless sensor networks are becoming more widespread and sophisticated. The collected data is used for visualization system that allow humans to evaluate sensor data by their sense of sight. Especially, a visualization using spatial interpolation promotes users to evaluate multi-point sensor data as a spatial image. However, since of the visualization using spatial interpolation requires high processing capacity, existing systems cannot visualize streaming sensor data in real-time. This study proposes a real-time spatial interpolation visualization system for multi-streaming sensor data using parallelization and WebGL. Experimental results revealed that our developed system using 18 nodes with 60 Hz acceleration sensors can visualize the vibration of installed area in 60 fps. It can also be easily used to analyze the vibration characteristics of structural objects.

Keywords: real-time visualization, spatial interpolation, streaming sensor data, web system

[1] 静岡大学大学院総合科学技術研究科
Graduate School of Integrated Science and Technology, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
[2] 静岡大学学術院情報学領域/JST さきがけ
College of Informatics, Academic Institute, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan / JST, PRESTO, Kawaguchi, Saitama 332–0012, Japan
[†1] 現在，ヤフー株式会社
Presently with Yahoo Japan Corporation
[a)] wakamori@minelab.jp
[b)] mineno@inf.shizuoka.ac.jp

1. はじめに

近年，センサ技術，無線通信技術の発展にともない，無線センサネットワーク（以降，WSN）の普及が進んでいる．特に，無線通信の伝送速度が向上したことで，加速度といった高レートでサンプリングされるセンサデータをリアルタイムに収集可能となった．高レートで絶え間なく伝送され続けるセンサデータをストリーミングセンサデータと呼び，将来はIoTデバイスの普及やスマートシティが実現され，生成されるストリーミングセンサデータ量は増大することが予想される．そのため，増大するストリーミングセンサデータのリアルタイム活用技術の発展は重要な課題である．

センサデータの活用例として，センサデータ可視化システム [1], [2], [3], [4], [5], [6] がある．可視化システムは，センサデータを人間が視覚的に理解可能な形式に変換し，ユーザへ提示するシステムである．その中でも，空間補間を用いた可視化はセンサを配置した空間の状態をヒートマップ状に描画するため，人間が空間の状態を直感的に理解可能となる [5]．空間補間を用いた可視化ではヒートマップの粒度を高精細にすることで滑らかな可視化となり，さらなるユーザエクスペリエンスの向上が期待できる．しかし，高精細な可視化を行う場合，システムの可視化処理である空間補間と描画の処理量は増大する．また，ストリーミングセンサデータは高レートで生成されるため，ストリーミングセンサデータのリアルタイム可視化において許容できる可視化処理時間は限られる．そのため，空間補間を用いた複数のストリーミングセンサデータの可視化において，リアルタイムかつ高精細な可視化を実現するには可視化に要する処理時間の短縮が重要となる．

本研究では，空間補間を用いた可視化システムにおける可視化処理である空間補間と描画の処理時間の短縮に重点をおき，高レートで生成されるストリーミングセンサデータをリアルタイムに空間補間し，高精細に可視化できるシステムを提案する．本システムは，一般的な液晶画面のリフレッシュレート 60 Hz を最大限利用し，Web ブラウザ上でフレームレート 60 fps でのリアルタイム空間補間可視化の実現を目指す．60 fps の可視化において許容される処理時間は 16.7 msec であり，Web Workers [7] を用いて空間補間を並列処理するだけでなく，WebGL [8] を用いて高速に描画することで，高負荷な可視化処理を 16.7 msec 未満で実行可能とする．

以降，2 章では関連研究について述べ，3 章ではシステムの提案を行う．4 章ではプロトタイプシステムの実装について，5 章で基礎評価結果，6 章で屋外評価結果を述べ，7 章では提案システムの応用例を検討する．最後に 8 章で本論文をまとめる．

2. 関連研究

WSN を用いたセンサデータ可視化システムには，消費電力や人間行動を可視化することで省電力化を促すシステム [2] や，屋外の環境データを可視化することで，一般市民の周囲の環境理解を促進し，緑地計画に利用するシステム [3] がある．これらのシステムは，可視化方法をグラフや計測点のマッピングとしており，空間補間を用いていない．特に環境データ可視化システム [3] では，空間補間を用いることで，市民の環境の理解をさらに促進できると考える．

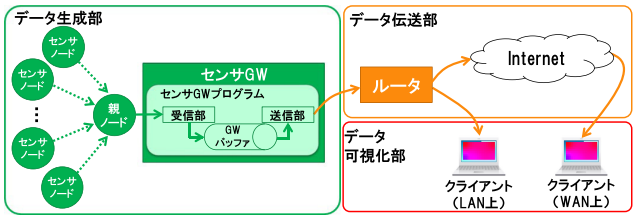
センサデータの空間補間を用いた可視化システムとして温湿度可視化システム [4], [5] がある．温湿度可視化システムは WSN を用いて温湿度を収集し，クライアントで空間の温湿度分布を可視化する．空調設備を設置した室内で，時間の経過にともなう温湿度変化をヒートマップで可視化することで，空調設備の性能や効果を手軽かつ適切に評価できることが示されている．またガラスハウス等の施設園芸環境へ応用することで，ハウス内の温湿度のムラの可視化を実現している．しかし，温湿度は単位時間あたりの変化量が比較的小さく，サンプリング周期を秒または分単位で設定することが多い．そのため，加速度といった高レートで生成されるストリーミングセンサデータのリアルタイム可視化は想定されていない．

ストリーミングセンサデータを用いた可視化システムとして，加速度データを用いた振動可視化システム [6] がある．この振動可視化システムは建造物に設置した WSN から加速度を収集し，建造物の振動を可視化する．開発されたシステムの動作検証では，歩道橋の床板の振動を可視化することに成功した．しかし，このシステムは，振動計測から可視化までのリアルタイム実行および Web ブラウザを用いたモニタリングや，インターネットを介した遠隔モニタリングは想定されていない．ユースケースを限定せず汎用的な可視化システムとするには，汎用端末に内蔵されている Web ブラウザを用いて，リアルタイムモニタリングや遠隔モニタリングを可能とすることが望まれる．

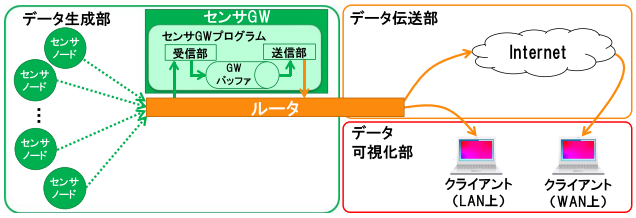
3. マルチストリーミングセンサデータ向けリアルタイム空間補間可視化システム

3.1 システムアーキテクチャ

本研究では，高レートで生成される複数のストリーミングセンサデータを，空間補間を用いてリアルタイムに可視化できるシステムを提案する．提案システムのアーキテクチャを**図 1** に示す．システムは，大別してデータ生成部とデータ伝送部，データ可視化部から構成される．データ生成部では多点配置したセンサノードからセンサデータを収集し，センサゲートウェイ（以降，センサ GW）に集約する．データ伝送部ではセンサ GW に集約されたセンサ



(a) 一般的なセンサネットワークを用いたアーキテクチャ



(b) センサノードをLAN上に配置するアーキテクチャ

図 1　システムアーキテクチャ
Fig. 1　System architecture.

データをクライアントへ伝送する．データ可視化部ではクライアントでセンサデータの空間補間と描画をリアルタイムに行う．既存の可視化システムの多くは，センサデータを，データベースサーバ（以降，DB サーバ）を介してクライアントへ提供するアーキテクチャとしているが，センサ GW から直接クライアントへ伝送することで DB へのアクセスに要する遅延を削減し，センサデータをリアルタイムにクライアントへ提供する．センサデータの蓄積は，システムのクライアントとして DB サーバを設置することで，DB サーバがセンサデータを受信でき，蓄積が可能となる．

図 1 (a) は通信規格として ZigBee 等を用いた一般的な WSN でのデータ収集を想定したアーキテクチャであり，図 1 (b) はセンサノードを，Wi-Fi を用いて LAN に接続し，データ収集を行うアーキテクチャである．本システムにおける可視化の精度は，センサノードの許容台数や時刻同期等，WSN の性能に影響を受ける．センサノードの台数が多いほど，多くのサンプルデータが得られ，補間精度の向上が見込めるが，リアルタイムにデータ収集可能な最大ノード台数は WSN の性能に依存する．センサノードにおける時刻同期は GPS を用いて絶対時刻を取得する手法 [9] や，Flooding Time Synchronization Protocol [10] 等の時刻同期プロトコルを適用した WSN を使用することで実現できる．そのため，本システムは，ユーザが収集データの特性やシステムの利用目的に応じて適切な WSN を選択することを想定したアーキテクチャとする．

また本システムは Web システムとして構築し，クライアントでは Web ブラウザを用いて可視化する．Web システムでは，Web ブラウザ搭載端末であればハードウェアや OS に非依存で，クライアント端末を選択できるだけでなく，クライアントプログラムをサーバから HTTP 経由で取得でき，クライアント端末におけるプログラムのインストールは不要である．そのため Web システムとして構築することで，ユーザは汎用端末を用いてシステムを利用でき，システムの利用開始時や更新時においてクライアントプログラムのインストールやアップグレードの手間を削減できる．

3.2　システムへの要求事項

提案システムにおいてリアルタイムに可視化可能なマルチストリーミングセンサデータの生成レートは，クライアント画面のリフレッシュレートに依存する．一般的な液晶画面のリフレッシュレートは 60 Hz であり，液晶画面の描画性能を最大限利用したシステムとするため，本システムでは多点配置したセンサノードから各 60 Hz で生成されるマルチストリーミングセンサデータを，フレームレート 60 fps でリアルタイムに空間補間可視化することを目標とする．

3.3　センサ GW-クライアント間での低遅延なデータ伝送

リアルタイムな可視化を実現するには，センサ GW からクライアントにおいて，低遅延でデータ伝送することが求められる．本システムにおいて低遅延なデータ伝送を実現するには，大きく 2 つの検討事項が存在する．

1 つ目の検討事項はデータの伝送方式である．本システムでは WebSocket を用い，低遅延なデータ伝送を実現する．リアルタイムな可視化を行うためには，ユーザが Web ブラウザで更新ボタンをクリックすることなく，非同期にデータが伝送されることが望ましい．また，ストリーミングセンサデータを効率良く伝送するために軽量な伝送方式が求められる．Web ブラウザ上で動作する非同期通信として，Ajax と WebSocket があげられる．Ajax は HTTP を用いて動作するため，データ受信のためにはクライアントサイドからのリクエスト送信やパケットへの HTTP ヘッダ付与が必要である．一方，WebSocket はデータをサーバサイドからクライアントサイドへプッシュ送信することができ，クライアントサイドからのリクエスト送信は不要である．またコネクション確立後に送信するパケットにはアプリケーション層のヘッダが不要であることから，WebSocket は Ajax に比べ軽量な伝送方式であるといえる．実際に，Ajax と WebSocket のデータ伝送性能の比較を行った研究 [11] では，WebSocket は Ajax に比べネットワークの使用帯域幅が 50%下回ることが示されている．そのため，本システムでは WebSocket を採用し，低遅延なデータ伝送を可能とする．

2 つ目の検討事項は空間補間処理の実行場所である．本システムではクライアントで空間補間処理を行うことで，伝送データ量を削減し，さらに低遅延なデータ伝送を実現する．空間補間処理の実行場所は，センサ GW とクライアントの 2 つがあげられるが，クライアントで空間補間処理を行うことで，センサ GW からクライアントへの伝送デー

タは未補間データとなる．未補間データは補間済みデータに比べデータ量が小さいことから，データ伝送時間も短縮される．高性能なセンサ GW を用いて空間補間処理を行うことで，データ量が大きい補間済みデータの伝送にともなう遅延を，センサ GW での高速な空間補間処理で解消できる場合も考えられる．しかし，一般にセンサ GW として使用される端末はノート PC やマイクロサーバ等の小型コンピュータである．そのため，センサ GW の性能はクライアントであるノートパソコンやスマートデバイスと同程度であり，空間補間の処理速度に大きな差は生じないと考える．以上より，本システムでは空間補間処理をクライアントで実行することでデータ伝送時間のさらなる低遅延化を図る．

3.4　クライアントでの低遅延な可視化処理

高レートに生成，伝送され続けるストリーミングセンサデータをリアルタイムに可視化するため，クライアントにおける低遅延な可視化処理について検討する．本システムにおいて想定するデータ生成レートは 60 Hz であるため，クライアントへのデータ入来周期は 16.7 msec である．空間補間と描画の処理が 16.7 msec 以内に完了しない場合，クライアントに未処理データが蓄積され 60 fps での可視化が困難となる．一方，詳細な空間補間を行い高精細な可視化を行う場合，空間補間と描画の処理時間は増大する．そのため，本システムでは Web Workers を用いた空間補間の並列処理と WebGL を用いた描画を採用し，可視化処理時間を低遅延で実行可能とする．

3.4.1　Web Workers を用いた空間補間の並列処理

Web Workers とは Web ブラウザで JavaScript の並列処理を可能とする標準仕様である．メインスレッドからワーカと呼ばれるスレッドを起動することで，並列処理を可能とする．Web Workers を用いた空間補間の並列処理のシーケンスを図 2 に示す．メインスレッドが各ワーカへ未補間データを送信すると，各ワーカは割り当てられた範囲の空間補間を行い，部分的な補間済みデータを生成する．各ワーカは生成した部分補間済みデータをメインスレッドへ送信すると，メインスレッドが部分補間済みデータを集約し，補間済みデータを生成する．

図 3 にメインスレッドとワーカの詳細動作を示す．図 3 (a) はメインスレッドがセンサ GW から未補間データを受信した際の動作である．受信時に過去データの処理が完了している場合，受信した未補間データをワーカへ送信する．過去データの処理が未完了である場合は，受信データを未補間データバッファへ格納する．未補間データバッファに格納されたデータは，過去データの処理が完了した後，古いデータから順にワーカへ送信される．未補間データを受信したワーカは，図 3 (b) に示すように割り当てられた範囲の空間補間を行い，部分補間済みデータを生成する．生成した部分補間済みデータはメインスレッドへ送信する．図 3 (c) はメインスレッドがワーカから部分補間済みデータを受信した際の動作である．全ワーカからのデータ受信を検知すると，受信データを用いて描画を行う．最

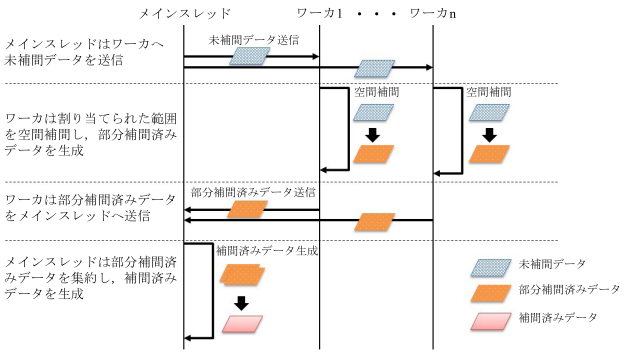
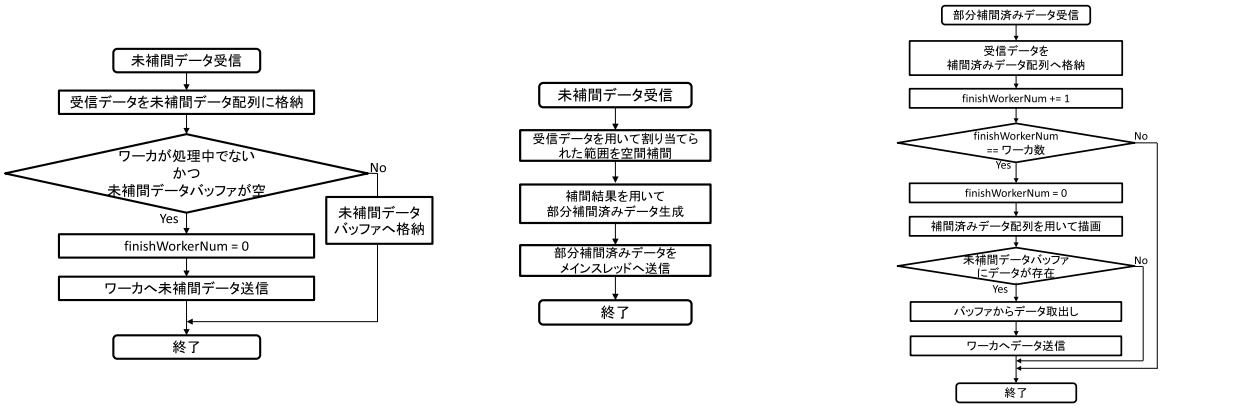


図 2　空間補間の並列処理シーケンス

Fig. 2　Parallel processing sequence of spatial interpolation.



(a)メインスレッドがセンサGWから未補間データを受信したときの動作　(b)ワーカがメインスレッドから未補間データを受信したときの動作　(c)メインスレッドがワーカから部分補間済みデータを受信したときの動作

図 3　Web Workers を用いた空間補間の並列処理におけるメインスレッドとワーカの動作

Fig. 3　Main thread and worker operation on parallel processing of spatial interpolation using Web Workers.

後に，未補間データバッファにデータが存在する場合はバッファからデータを取り出しワーカへ送信する．

メインスレッドで未補間データバッファを用意しなくても，各ワーカには受信キューが存在する．しかし，受信キューはワーカごとに独立して存在するため，あるワーカが処理を終えたとき，他のワーカが処理中であっても，受信キューに存在する次のデータを用いて処理を再開してしまう．そのため，メインスレッド側でデータをバッファリングし，すべてのワーカの処理が完了したことを検知した後，各ワーカへ次のデータを送信する．

ワーカからメインスレッドへ送信される部分補間済みデータは Transferable objects に含まれる Float32Array クラスのオブジェクトとする．Transferable objects とは，Web Workers においてデータの実体のコピーがともなわない，高速な送受信が可能なクラスの総称である．部分補間済みデータは未補間データに比べ大規模なデータであるため，Transferable objects に含まれる Float32Array クラスのオブジェクトとすることで，送受信における遅延の解消を図る．

3.4.2　WebGL を用いた描画

本システムでは，ヒートマップの描画方法として WebGL を用い，高速な描画処理を実現する．WebGL は特別なプラグインを必要とせずに Web ブラウザ上で GPU を用いた高速な描画処理が可能な API である．WebGL と同様にプラグインなしで動作する描画 API に Canvas 2D Context があるが，WebGL は Canvas 2D Context に比べヒートマップの連続描画に適した性質を有する．WebGL または Canvas 2D Context を用いてヒートマップを描画する場合，ヒートマップ内の各点の座標・サイズ・色を指定する必要がある．一方，ヒートマップの連続描画には，各点の座標とサイズは固定であり，描画ごとに更新すべき情報は色のみという特徴がある．WebGL は点の座標・サイズ・色を独立して設定でき，設定した値は GPU 上のメモリに保持されるため，色の再指定のみで描画が可能となる．そのため，WebGL を採用することで，色情報のみを更新する高速な描画処理を実現する．

4.　プロトタイプ実装

4.1　リアルタイム振動可視化システム

提案システムのプロトタイプとして，**図 4** に示すリアルタイム振動可視化システムを開発した．本プロトタイプでは 3.4 節で述べたクライアントでの可視化処理時間の短縮に関する検証に重点をおくため，ストリーミングセンサデータを安定して生成できる図 1 (b) の Wi-Fi を用いたアーキテクチャを採用した．プロトタイプに用いた各端末のハードウェア構成は**表 1** に示す．

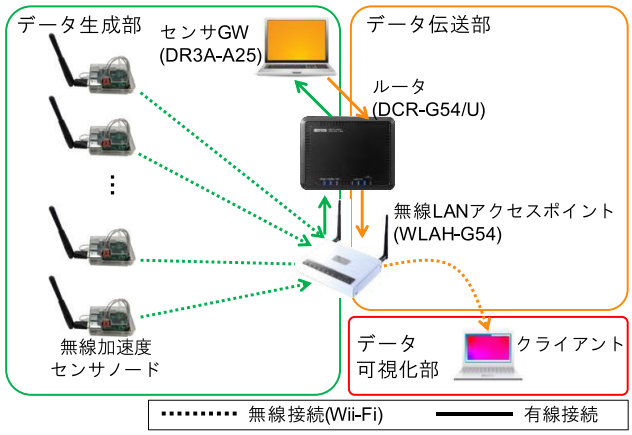


図 4　リアルタイム振動可視化システム
Fig. 4　Real-time vibration visualization system.

表 1　使用端末のハードウェア構成
Table 1　Specification of nodes used.

	無線加速度センサノード	センサ GW	クライアント
型番（製造元）	Raspberry Pi2 (RaspberryPi Foundation)	DR3A-A25 (ONKYO)	R732/E26GB (TOSHIBA)
OS	Raspbian 8.0	Ubuntu 14.04.4 LTS 64bit	Windows7 Professional 64bit
プロセッサ	ARM Cortex-A7	Intel Core i5-2430M	Intel Core i5-3230M
クロック周波数	900 MHz	2.40 GHz	2.60 GHz
コア数（論理コア数）	4 (4)	2 (4)	2 (4)
メインメモリ	1 GB	8 GB	8 GB

4.2　データ生成部の実装

データ生成部は複数の無線加速度センサノードとセンサ GW で構成され，センサノードがセンシングしたデータをセンサ GW に集約する．このとき，センサノードからセンサ GW へのデータ伝送においてデータの欠落が生じることが考えられる．TCP 通信を用いる場合は再送処理が行われ，高信頼なデータ伝送が可能となる．一方，UDP 通信を用いる場合，再送処理が行われないため，データの欠落が生じるが，つねに最新のデータを GW へ送信できる．本システムでは，リアルタイムな振動可視化を目指していることから，つねに最新のデータを GW で集約できる UDP 通信を用い，データ欠落時には直前のデータを利用することとした．実際のアプリケーションとして実装する場合，データ欠落時にはアプリケーションに応じて異常通知や適切な補間手法の適用が必要である．今回は，マルチストリーミングセンサデータをリアルタイムに空間補間して可視化できるかの検証を目的としたため，データ欠落時の適切な処理手法に関しては今後の課題とする．

4.2.1　無線加速度センサノードの実装

無線加速度センサノードの構成を**図 5** に示す．センサノードは加速度のセンシングとセンサ GW へのデータ送信を周期的に行う．センサノードへの電源供給に用いるモバイルバッテリ（cheero 製，CHE-061）は，IoT 機器に対応しており，出力電流が小さい場合でも電源供給を停止し

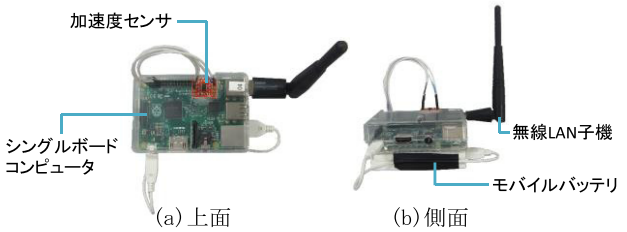


図 5　無線加速度センサノードの構成
Fig. 5　Wireless acceleration sensor node.

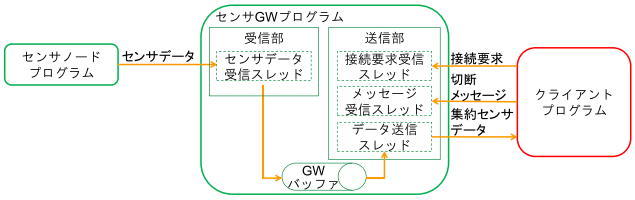


図 6　センサ GW プログラムの構成
Fig. 6　Architecture of sensor GW program.

ないため，Raspberry Pi 2 を安定して動作させることができる．3 軸加速度センサ（Analog Devices 製，ADXL345）は，比較的安価であり，I2C 通信が使用可能であるため，Raspberry Pi2 においてセンサデータ読み出し処理の実装が容易である．データシート [12] には，加速度センサは機械的な圧力に敏感であるが，センサの組立時に圧力が加わることがあると記載されており，得られる加速度データには個体差の存在が想定される．また，各軸の計測範囲は最大 ±16 g であるが，0 g における出力誤差は X，Y 軸では ±35 mg，Z 軸では ±40 mg とされており，ドリフトも存在する．そのため，より高精度な加速度測定が必要なアプリケーションではセンサごとにキャリブレーション設定やソフトウェアでのドリフト対策が必要となるが，本プロトタイプでは，クライアントにおけるリアルタイム可視化性能の検証を目的としたため，計測値のバラつきやドリフトへの対策は今後の課題とする．無線 LAN 子機にはチップアンテナに比べ長距離通信が可能なラバーダックアンテナ（ロジテック製，LAN-WH300NU2）を採用した．

4.2.2　センサ GW の実装

センサ GW はセンサデータの集約とクライアントへの送信を行う．センサ GW プログラムの構成を図 **6** に示す．センサ GW プログラムは受信部と送信部に大別される．受信部のセンサデータ受信スレッドでは，センサノードから送信された加速度データを受信し，GW バッファへ格納する．送信部の接続要求受信スレッドでは，クライアントからの WebSocket の接続要求の受信を待機し，要求を受信した場合はクライアント追加処理を行う．メッセージ受信スレッドではクライアントからの切断メッセージの受信を待機し，切断メッセージを受信した場合，クライアント削除処理を行う．データ送信スレッドでは GW バッファに格納されたデータを，クライアントプログラムへ送信する．

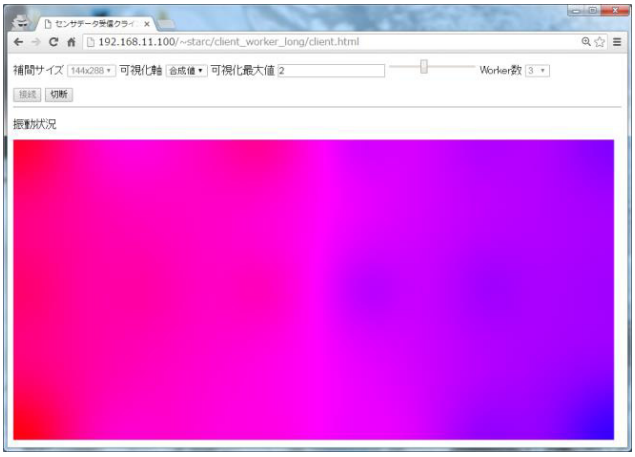


図 7　可視化例
Fig. 7　Example of visualization.

受信部と送信部を明確に分離することで，データ収集方法の変更にともない，データの形式や受信方法に変更が生じる場合でも，受信部の修正のみで変更内容を吸収できる．そのため，プロトタイプの運用時に環境や目的に応じてデータの収集方法を変更する場合にも，センサ GW プログラムの修正は受信部に限定される．

4.3　データ伝送部の実装

データ伝送部ではセンサ GW に集約されたデータをクライアントへ送信する．センサ GW とクライアントの LAN への接続は無線接続と有線接続のいずれかを選択する必要があり，両者には，通信の信頼性と端末の可搬性のトレードオフ関係がある．センサ GW は，一般に固定設置されるため，通信の信頼性を重視し有線接続とした．一方で，クライアントは，ユーザが端末を持ち歩きながらシステムを利用可能とするため，可搬性を重視し無線接続とした．

4.4　データ可視化部の実装

データ可視化部ではクライアントの Web ブラウザ上でセンサデータの可視化を行う．クライアントでの可視化例を図 **7** に示す．空間補間の結果から加速度データの値が高い位置は赤色で，低い位置は青色で表現する．

クライアントで実行する空間補間アルゴリズムには，逆距離加重法（IDW）やクリギング，最近傍補間等が存在する．その中でも，IDW はクリギングに比べ計算量が小さく，最近傍補間に比べ補間精度が高いという特徴を有する．またパラメータチューニングが容易であり実用性の高い補間アルゴリズムであると考え，本プロトタイプでは，IDW を採用した．IDW のアルゴリズムは式 (1)，(2) で表される．

$$u(x) = \frac{\sum_{i=1}^{N} w_i(x) \cdot (x_i)}{\sum_{j=1}^{N} w_j(x)} \tag{1}$$

$$w_i(x) = \frac{1}{d(x, x_i)^p} \quad (2)$$

N へデータ数，$u(x)$ は点 x の補間値，$d(x, x_i)$ は点 x と点 x_i のユークリッド距離，p は補間データの算出における近傍点の影響度を調整するパラメータである．本プロトタイプにおいて p は一般的な 2 とした．

5. 基礎評価

5.1 実験内容

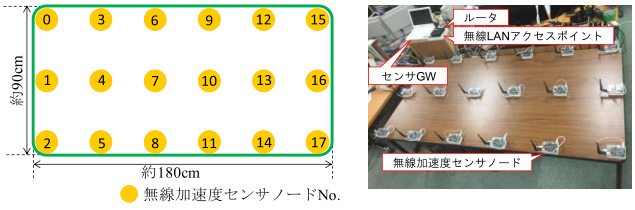
実装したプロトタイプシステムを用いて提案システムの基礎評価実験を行った．本実験では，3.4 節で述べた可視化処理時間の短縮に関する検証を行った．

まず，WebGL を用いた描画の有効性を検証するため，システムの描画方法として WebGL を用いた場合と Canvas 2D Context を用いた場合の描画処理時間を比較した．評価に用いた可視化の粒度は 9 × 18 点，18 × 36 点，36 × 72 点，72 × 144 点，144 × 288 点の 5 段階を用いた．空間補間の並列処理の評価の前に描画方法の評価を先に行う理由は，空間補間の並列処理の評価で用いる可視化の粒度を決定するためである．本システムのデータ生成レートは 60 Hz であるため，クライアントには 16.7 msec 周期でデータが入来する．描画処理に 16.7 msec 以上要する可視化の粒度では，空間補間の並列処理を組み合わせたとしてもシステムとして遅延のない可視化が困難となる．そのため，描画方法の評価を先に行うことで，空間補間の並列化の評価に用いる可視化の粒度を決定することとした．

次に，Web Workers を用いた空間補間の並列処理の有効性を検証するために，並列数（ワーカ数）を 1 から 8 まで変化させ，空間補間の処理時間を比較した．並列化の効果はクライアントの論理コア数に依存すると考える．並列数を 3 とした場合，クライアントプログラムはメインスレッドが 1，ワーカが 3 の合計 4 スレッドが動作する．クライアントの論理コア数は 4 であることから並列数を 4 以上（5 スレッド以上での動作）とした場合，動作スレッド数が論理コア数を上回る．そのため，複数のスレッドが 1 つの論理コアを時間割で使用することとなり，さらなる処理時間の短縮は見込めないと考える．並列数は本仮説を検証するために十分である 1 から 8 とした．

クライアントプログラムの可視化性能を評価するため，空間補間処理時間の計測と同時に，可視化のフレームレートを計測した．クライアントへのデータ入来周期はセンサデータの生成周期に等しい．そのため，可視化のフレームレートがセンサデータ生成レートの 60 Hz に一致する場合，クライアントでは入来するデータを遅延なく可視化できているといえる．

実験環境を**図 8** に示す．本実験は研究室内の机（幅約 180 cm，奥行約 90 cm，高さ約 70 cm）に 18 台の無線加速度センサノードを設置し，システムを動作させる形で実験



(a)センサノード配置図　(b)現場環境

図 8　基礎評価実験環境

Fig. 8　State of fundamental experiment.

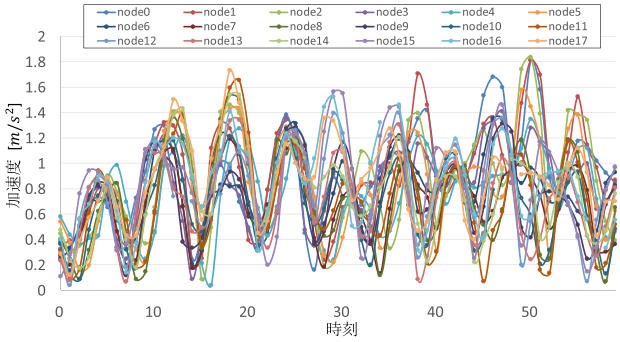


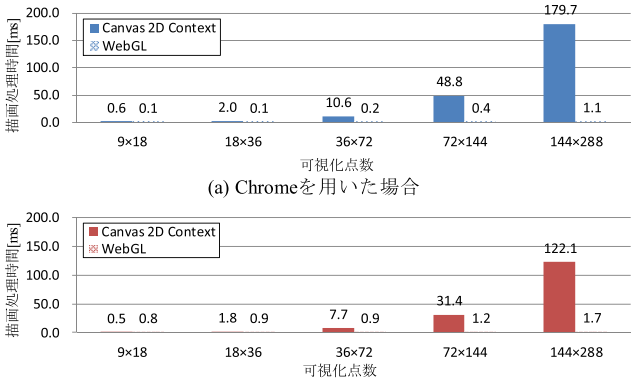
図 9　基礎評価実験時の加速度データの一部

Fig. 9　A part of acceleration data in fundamental experiment.

を行った．空間補間と描画の処理時間とフレームレートの計測値は，1,000 回処理を行った際の平均値とした．描画に用いる HTML の canvas 要素のサイズは，可視化対象である机の縦横比 1 : 2 に合わせ 450 × 900 とし，Web ブラウザは Google Chrome と Mozilla Firefox を用いて評価した．本プロトタイプシステムを用いて振動状況のリアルタイム空間補間可視化を検証するため，センサノードを設置した机を手動で揺らし続けることで，つねに振動が存在する環境で実験を行った．本システムを用いて可視化した，加速度センサから得られる 3 軸加速度合成値の絶対値を**図 9** に示す．図 9 の各凡例ラベルは，図 8 (a) 中のセンサノード No. に対応している．各センサノードから得られた加速度データの変動状況を確認できることから，本実験中は継続した振動が存在する環境であったことが分かる．

5.2 描画処理時間

描画方法として WebGL を用いた場合と Canvas 2D Context を用いた場合の各描画処理時間を**図 10** に示す．図 10 から WebGL は Firefox での 9 × 18 点の描画を除く 9 項目で Canvas 2D Context に比べ高速に描画処理を行えることが分かる．WebGL と Canvas 2D Context どちらも可視化の粒度が精細になるにつれて処理時間が増大する．しかし，WebGL では最も精細な可視化の粒度 144 × 288 点を描画した場合でも，Canvas 2D Context に比べ Chrome では約 163 倍高速な 1.1 msec，Firefox では約 72 倍高速な 1.7 msec で処理を終えることを示した．WebGL を用いた描画では Chrome が Firefox に比べ高速に処理を終えたことが分か



(a) Chromeを用いた場合

(b) Firefoxを用いた場合

図 10　描画の処理時間

Fig. 10　Processing time for drawing.

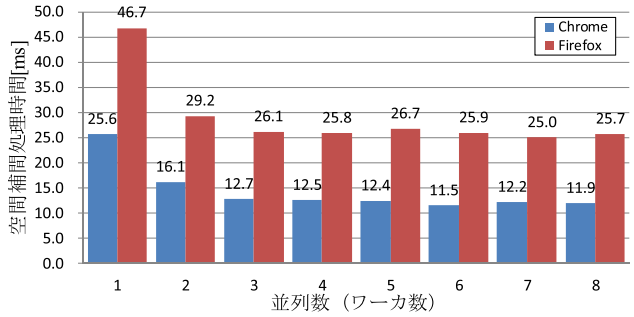


図 11　空間補間の処理時間

Fig. 11　Processing time for spatial interpolation.

る．その 1 つの要因として JavaScript エンジンの違いがあげられる．JavaScript エンジンとして，Chrome は V8 を，Firefox は SpiderMonkey を採用しており JavaScript エンジンの違いから処理時間の差が生じていると考えられる．

5.3　空間補間処理時間

空間補間処理の並列数を 1 から 8 に変化させた場合の処理時間を**図 11** に示す．描画方法の評価において，WebGL を用いた描画では最も精細な可視化の粒度 144 × 288 点においても Chrome では 1.1 msec，Firefox では 1.7 msec で描画処理が実行できることが図 10 で示された．そのため，本評価では可視化の粒度として 144 × 288 点を用いた．図 11 から空間補間は並列処理することで処理時間が短縮されることが分かる．並列数を増加させると処理時間の短縮が並列数 3 で頭打ちとなり，並列数 4 以上の場合は大きな性能向上が見られない．5.1 節で述べた，空間補間の並列処理化における処理時間の短縮はクライアントの論理コア数に依存するという仮説を裏付ける結果が得られた．また Chrome は Firefox に比べ高速に空間補間処理を行えることが示された．WebGL を用いた描画処理における処理時間の差と同様に，JavaScript エンジンの違いが性能差の要因の 1 つであると考えられる．

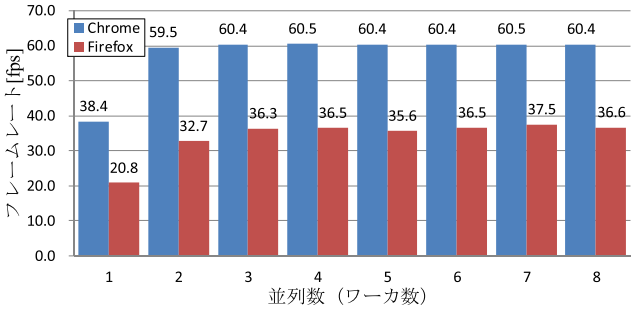


図 12　可視化フレームレート

Fig. 12　Visualization frame rate.

5.4　可視化フレームレート

図 12 に可視化フレームレートの計測結果を示す．図 12 から Chrome を用いた可視化では並列数 3 以上の場合，フレームレートが 60 fps に達しており，60 Hz で生成されるデータを遅延なく可視化できることが示された．一方で，Firefox を用いた可視化ではフレームレートが 60 fps に満たないことが分かる．この要因として空間補間処理における遅延があげられる．描画方法の検証から WebGL を用いることで，Firefox でも 1.7 msec で描画処理を完了できることが示された．しかし，空間補間は並列処理を用いても最小処理時間が 25.0 msec とデータ入来周期を上回った．そのため，空間補間処理における遅延が生じ，フレームレートが 60 fps に達しなかったといえる．

また，クライアントの処理性能は Web ブラウザだけでなく，クライアントのハードウェア性能にも依存すると考えられる．この対策として，可視化の粒度をクライアントの処理性能に適した値に設定することで解決できると考える．可視化の粒度を粗くすると空間補間の処理時間も短縮される．そのためクライアントの処理性能に適した可視化の粒度を設定することで，遅延のない可視化が実現できると考える．可視化開始時に処理性能のベンチマークを行い，ベンチマーク結果から可視化の粒度を決定することで，使用しているクライアントの性能に適した可視化の粒度を動的に決定できると考える．

6.　屋外評価

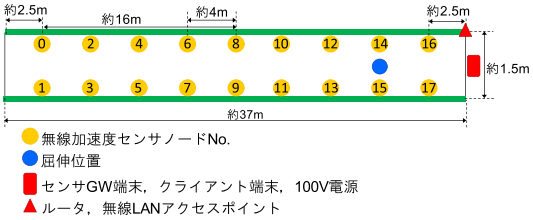
6.1　実験内容

本研究で実装したリアルタイム振動可視化システムの有用性を評価するため，静岡県浜松市内の吊橋（橋長約 37 m，幅員約 1.5 m）にシステムを設置し吊橋の振動可視化実験を行った．吊橋上に 18 台のセンサノードを配置し，吊橋の上で屈伸を行うことで振動を与え，クライアントでの可視化を観測した．**図 13** に実験環境を示す．観測においてはクライアントの画面を iPhone6 のカメラを用いて 60 fps で動画撮影し，記録した．

6.2　実験結果・考察

本実験において得られた加速度データを図 14 に示す．図 14 の各凡例ラベルは図 13 (a) 中のセンサノード No. に対応しており，屈伸位置に最も近いセンサノード 14，15（図 14 (b)）において大きな加速度が得られた後，他のセンサノードから得られる加速度が順に変動していることから，振動伝播の様子を確認できる．また，図 14 の加速度が得られたときの可視化結果を撮影した動画からフレーム画像を生成し，時系列に並べた一連の画像を図 15 に示す．図 15 から屈伸を行った地点から振動が左右に伝播していることが分かる．振動の伝播は吊橋の構造に関係するが，Web ブラウザ上で振動の伝播をリアルタイムに可視化できたことから，本システムを用いることで，吊橋の振動状況を容易かつリアルタイムに分析可能と考える．

図 13 に示したように，無線 LAN のアクセスポイントは吊橋の柱上に設置した．これは当初センサ GW 付近に設置していたが，一部のノードとの通信が不通であったためである．一般に無線アンテナは地面に近い位置に設置した場合，電波が減衰し通信距離が短くなるため，高い位置



(a) 使用機器配置と屈伸位置

(b) 現場環境

図 13　屋外評価実験環境

Fig. 13　State of outdoor experiment.

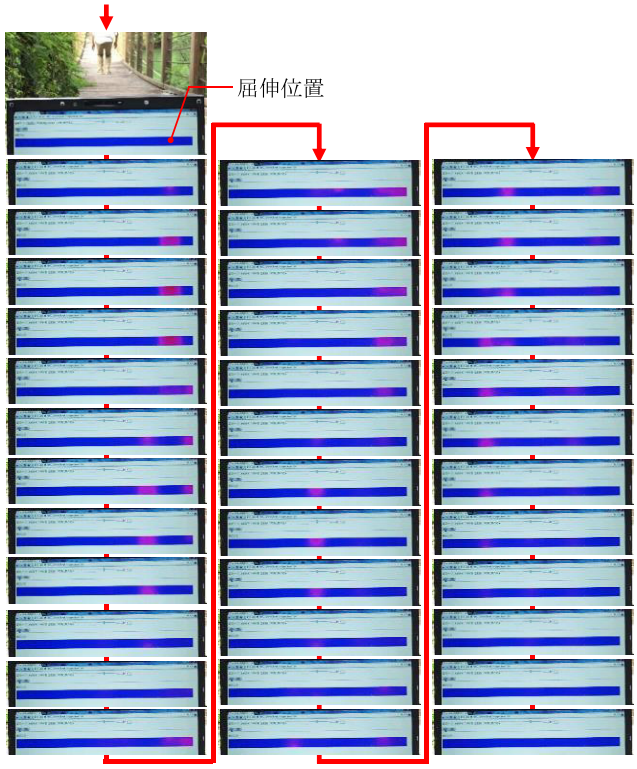
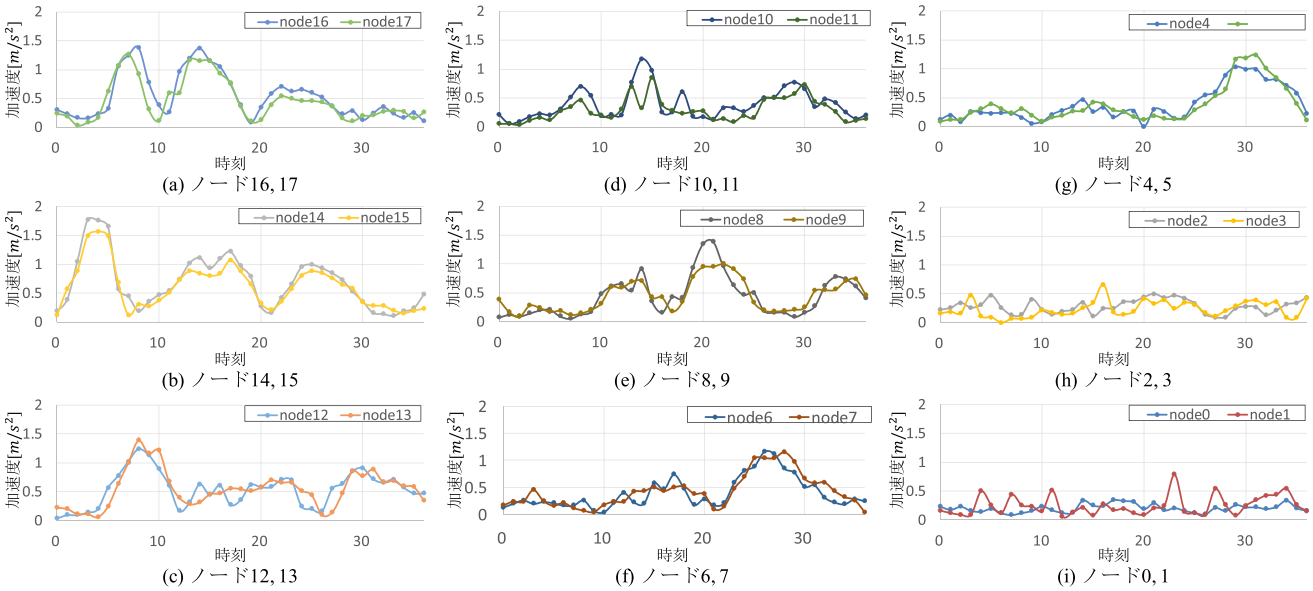


図 15　人間が屈伸を行ったときの可視化結果

Fig. 15　Visualization result when a human performed a deep-knee bend.



(a) ノード16, 17　(d) ノード10, 11　(g) ノード4, 5

(b) ノード14, 15　(e) ノード8, 9　(h) ノード2, 3

(c) ノード12, 13　(f) ノード6, 7　(i) ノード0, 1

図 14　屋外評価実験時の加速度データ

Fig. 14　Acceleration data in outdoor experiment.

にアクセスポイントを設置することですべてのノードと通信が可能となったと考える．

7. 応用例

提案システムは多点配置したセンサノードから生成されるマルチストリーミングセンサデータをリアルタイムに空間補間可視化できる．マルチストリーミングセンサデータをリアルタイムに空間補間して可視化可能にすることで，建築，土木分野への貢献が期待できる．たとえば，構造物の振動状況を簡易的にでもリアルタイムに簡単に把握できるようになれば，建造物の点検が効率的にできるようになり，気になる建造物のみ専門家と専門機器を用いて再評価するという新たな点検手法や構造評価手法の実現に役立てられると考える．今後，建造物の振動具合と評価基準の紐付け等，専門家との議論が必要であるが，本研究で開発したプロトタイプシステムを用いて，実際にシステムを動作させながら議論を進めることで，本システムの応用に向けた議論の具体化が期待できる．

8. おわりに

本研究では，マルチストリーミングセンサデータ向けリアルタイム空間補間可視化システムを提案し，プロトタイプシステムを開発した．クライアントにおける低遅延な可視化処理の実現のため，空間補間には Web Workers を用いた並列処理を，描画方法には WebGL を採用した．基礎評価の結果，空間補間は並列処理を用いて処理時間が短縮可能であり，WebGL は Canvas 2D Context に比べ高速な描画処理が行えることを確認した．可視化のフレームレートを計測したところ，空間補間を並列処理し，WebGL を用いて描画することで，60 Hz で生成されるセンサデータを 60 fps でリアルタイムに可視化できることを示した．また，屋外評価の結果，吊橋の振動伝播をリアルタイムに可視化でき，本システムを用いて建造物の振動をリアルタイムに分析できる見通しを得た．現在，橋梁等の建造物の点検において，可視化システムを用いたモニタリングは活用されていないが，本システムを用いることで，建造物の振動といった物理量の変化と健全性の関係性をリアルタイムに分析可能となり，新たな建造物点検手法の開発に有用な基盤システムを実現できたと考える．

今後の課題としては，インターネットを介したデータ伝送やクライアント数を増加させた場合の可視化性能の評価があげられる．また，本システムで可視化可能なデータの生成周期はクライアント画面のリフレッシュレートに依存する．そのため，一般的な液晶ディスプレイのリフレッシュレートである 60 Hz よりも高いサンプリングレートが必要な振動や音声の可視化は困難である．その対策として 60 Hz よりも高い周波数でサンプリングしたセンサデータを，可視化用に 60 Hz に圧縮する手法の検討が必要である．

謝辞　本研究は，半導体理工学研究センター（STARC）共同研究 IS プログラム No.1529 により実施したものである．

参考文献

[1] Villanueva, F.J., Aguirre, C., Rubio, A.B. et al.: Data stream visualization framework for smart cities, *Soft Computing*, Vol.20, Issue 5, pp.1671–1681 (2016).
[2] 中根　傑，江田政聡，横山昌平ほか：センサネットワークにおける大規模な可視化システムの開発，第 2 回データ工学と情報マネジメントに関するフォーラム（DEIM2010），D5-3 (2010).
[3] 伊藤昌毅，片桐由希子，石川幹子，徳田英幸：Airy Notes：緑地計画のための無線センサネットワークによる環境モニタリング，情報処理学会論文誌，Vol.49, No.1, pp.69–82 (2008).
[4] 柴田　瞬，丸島晃明，峰野博史：3 次元可視化システムと WSN を用いた視覚的評価手法の提案，第 77 回全国大会講演論文集，Vol.2015, No.1, pp.235–236 (2015).
[5] 松野智明，串岡　聡，今原淳吾ほか：観測データの空間補間を利用した施設園芸環境の可視化・制御システムの提案，マルチメディア，分散，協調とモバイル（DICOMO2012）シンポジウム，pp.2129–2136 (2012).
[6] 川原正人，中畑和之，大賀水田生：多点同時計測による橋梁床板の動的挙動の 3 次元可視化と歩道橋における実験的検証，構造工学論文集，Vol.59A, pp.1170–1178 (2013).
[7] Hickson, I.: Web Workers, W3C (online), available from ⟨https://www.w3.org/TR/2015/WD-workers-20150924/⟩ (accessed 2016-07-20).
[8] Jackson, D. and Gibert J.: WebGL Specification, Khronos (Online), available from ⟨https://www.khronos.org/registry/webgl/specs/latest/1.0/⟩ (accessed 2016-05-20).
[9] 早速　勉，相馬　充，下代博之，橋口　隆：GPS による汎用時刻保持装置の開発，国立天文台報，Vol.5, pp.73–79 (2001).
[10] Maroti, M., Kusy, B., Simon, G. and Ledeczi, A.: The Flooding Time Synchronization Protocol, *Proc. 2nd ACM Conference on Embedded Networked Sensor Systems* (*SenSys '04*), pp.39–49 (2004).
[11] Puranik, D., Feiock, D., and Hill, J.: Real-time Monitoring using AJAX and WebSockets, *Proc. 20th Annual IEEE International Conference and Workshops on the Engineering of Computer Based Systems* (*ECBS 2013*), pp.110–118 (2013).
[12] アナログ・デバイセズ株式会社：ADXL345，アナログ・デバイセズ株式会社（オンライン），入手先 ⟨http://www.analog.com/media/jp/technical-documentation/data-sheets/ADXL345_jp.pdf⟩（参照 2016-12-18）．

若森 和昌　（学生会員）

2017 年静岡大学情報学部卒業．同年同大学大学院総合科学技術研究科進学．センサネットワーク応用，特にマルチストリーミングセンサデータ向けリアルタイム空間補間可視化の研究に従事．

丸島 晃明　（学生会員）

2015 年静岡大学情報学部卒業．2017 年同大学大学院総合科学技術研究科修了．同年ヤフー株式会社入社．無線センサネットワーク，可視化システム，ストリーミングセンサデータの記録手法に関する研究に従事．

峰野 博史　（正会員）

1999 年静岡大学大学院理工学研究科修士課程修了．同年日本電信電話（株）入社．NTT サービスインテグレーション基盤研究所を経て，2002 年 10 月より静岡大学情報学部助手．2006 年九州大学大学院システム情報科学府博士（工学）．2011 年 4 月より静岡大学情報学部准教授．2015 年 12 月より JST さきがけ研究者兼務．知的 IoT システムに関する研究に従事．2012 年度本学会長尾真記念特別賞受賞．電子情報通信学会，IEEE，ACM 各会員．