LRTS: LDM リアルタイムシミュレータの開発と評価

鈴木 有也[1,a)]　牧戸 知史[1]　佐藤 健哉[2]　高田 広章[3]

概要：Local Dynamic Map（以降，LDM と略す）は，複数の車両とその周辺環境から得られる様々なデータを収集し解析する ITS アプリケーションのプラットフォームとして注目されている．しかしながら，蓄積型のデータベースを用いた LDM では，車両台数が増加するとリアルタイム性を発揮できないために，LDM を用いた ITS アプリケーションのリアルタイム動作評価を行うシミュレータを構築するのが困難であるという問題点がある．本研究では，リアルタイム性能を重視したストリーム LDM を用いて LDM リアルタイムシミュレータ（以降，LRTS と呼ぶ）を構築し評価する．衝突警告アプリケーションを例としてリアルタイム性を評価したところ，運転シミュレータと LDM 間のラウンドトリップタイムが，車両台数 37 台〜45 台において，平均で 5.98[ms]，最大で 13.36[ms] となり，アプリケーション要件である 100[ms] 以内の応答が可能であることが示された．

キーワード：Local Dynamic Map(LDM)，自動車，シミュレータ，リアルタイム，ITS

LRTS: A Development and Evaluation of LDM Real-Time Simulator

Naoya SUZUKI[1,a)]　Satoshi MAKIDO[1]　Kenya SATO[2]　Hiroaki TAKADA[3]

Abstract: Local dynamic map(LDM) is known as a platform of ITS applications that gathers and analyzes various data from vehicles and their surroundings. However LDM implemented by store-based databases has poor real-time performance when it deals with many vehicles, thus it is difficult to develop a real-time simulator that evaluates ITS applications on LDM. We construct LDM real-time simulator called *LRTS* based on Stream LDM that excels in real-time performance and evaluate it by constructing a collision warning application. The result shows that the response time between a driving simulator and LDM is 5.98[ms] on average and 13.36[ms] at most from 37 to 45 vehicles, and thus it can satisfy 100[ms] deadline that is the application requirement.

Keywords: Local Dynamic Map(LDM), motor vehicle, simulator, real time, ITS

1. はじめに

ネットワーク技術が進歩し，M2M 通信のような機器同士の常時接続を前提としたサービスが広まりつつある現在，複数の車両とその周辺環境から得られる様々なデータを収集し解析したうえで車両にフィードバックする仕組みが提案されつつある [14]．このような仕組みを提供する ITS プラットフォームのひとつとして近年注目されているのが，Local Dynamic Map（以降，LDM と略す）である [3]．LDM は，交通の安全にかかわる静的および動的な要素（例：自車両情報・周辺車両情報・天候状態・交通状態・詳細地図等）を格納するデータの集合体であり，**図 1** に示すように，異なる更新頻度を持つ 4 階層のデータ構造を有する．すなわち，Type 1 の静的データ（地図），Type 2 の準静的データ（道路標識，建造物等），Type 3 の準動的データ（渋滞，道路工事等），および Type 4 の動的データ（自他車両，歩行者等）である．特に，Type 3 と 4 に関しては，リアルタイムな更新が要求される．LDM を ITS のプラットフォームとして用いることで，複数の車両，およびその周辺環境から得られる様々なデータを入力とし車両にフィードバックする ITS アプリケーション（以降，単にアプリケーションと略す）の構築を容易化できる．これらのアプリケーションの例としては，広範囲にわたる複数の他車と自車の位置・速度等から衝突を予測して運転者に

1 株式会社 豊田中央研究所
Toyota Cenral R&D Labs., Inc.
2 同志社大学モビリティ研究センター
Mobility Research Center, Doshisha Univ.
3 名古屋大学大学院情報科学研究科附属 組込みシステム研究センター
Center for Embedded Computing Systems, Nagoya Univ.
a) naoya@mosk.tytlabs.co.jp

ⓒ 2014 Information Processing Society of Japan

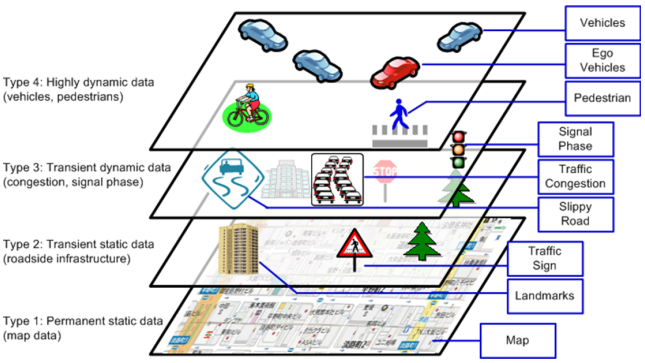

図 1　LDM(Local Dynamic Map)
Fig. 1　LDM(Local Dynamic Map)

警告を発する衝突警告アプリケーション，街全体の現在の渋滞を把握，あるいは将来の渋滞を予測して渋滞を回避するルートを提示する渋滞回避案内アプリケーション等が挙げられる．欧州の SAFESPOT プロジェクトにおいては，アプリケーションのプラットフォームとして LDM を位置付け，機能，API，ユースケース等を策定している [5]．

LDM とそれを利用したアプリケーションをリアルタイムに動作させ，ITS に応用するためには，アプリケーションの動作評価を繰り返し行って，アプリケーションが設計者の意図通りに動作するかを検証する必要がある．動作評価でもっとも説得力があるのは，実車を用いた評価である．たとえば，テストコース等で複数の実車を走行させ，それらのセンサデータを LDM にリアルタイムに入力し，アプリケーションが，その LDM からの情報をもとに計算を行って車両にフィードバックするといった仕組みが考えられる．しかしながら，実車での実験は，実車へのセンサ搭載，テストコースの準備等，経済的および時間的コストがかかることに加え，実験車両を走行させることに少なからずリスクが伴う．また，多数の車両を用いたスケーラビリティ評価，衝突警告等の安全系のアプリケーション評価等，実車評価が困難なシナリオ検証も存在する．

そこで，代替手段として挙げられるのが，シミュレーションによるアプリケーションの動作評価である．すなわち，車両の運転を模擬できる運転シミュレータ上で，自車とともに複数の車両を走行させ，それらの車両の走行データをリアルタイムに LDM に入力する．LDM では，入力された複数の車両データをリアルタイムに処理し，アプリケーションを動作させ，結果を運転シミュレータ上の車両にフィードバックする．そのうえで，LDM とアプリケーションの動作を可視化し，自車に対するアプリケーションからの応答が適切なタイミングで行われるかを評価する．この動作評価では，運転シミュレータ上で操作する自車に対して，アプリケーションからの応答（通知・制御等）が適切なタイミングで出力されるかを評価することが求められるため，シミュレーションにおいてもリアルタイム性が重要となる．以上のようなことが可能なリアルタイムシミュレータが，アプリケーションの動作評価に有効といえる．

しかしながら，SAFESPOT が提案する蓄積型のデータベースを用いた LDM[5] では，Type 4（車両等）の動的データを扱う際に，車両台数が増加するとデータベースの処理時間が著しく増加してリアルタイム性を発揮できないという問題点がある [10]．すなわち，LDM に入力される各種データの中でも，特に Type 4（車両等）のように，高頻度（例．100ms ごと）にデータベースを更新する際のオーバヘッド，およびアプリケーションが発行したクエリを同じく高頻度に処理するオーバヘッドが問題となる．また，この問題点があるゆえに，LDM を用いたアプリケーション動作をリアルタイムに可視化して評価するシミュレータの構築が困難であるという問題点をも併せ持つ．

そこで本研究は，LDM を用いたアプリケーションのリアルタイム動作評価を行うことができる **LDM リアルタイムシミュレータ**（LDM Real-Time Simulator，以降，**LRTS** と呼ぶ）の構築と評価を目的とする．LRTS の最大の特徴は，ストリームデータ処理技術を用いたリアルタイム動作可能な LDM であるストリーム LDM[11] を使用することで，運転シミュレータから生成される複数の車両データをアプリケーションがリアルタイムに処理し，その計算結果を運転シミュレータにフィードバックし可視化することである．安全系のアプリケーション例として衝突警告アプリケーションを構築し，リアルタイムな動作評価が可能であるかを検証する．

本論文の構成は以下のとおりである．第 2 章で LRTS の主なシステム要件を列挙し，第 3 章で設計を述べる．第 4 章で実装について説明し，第 5 章で評価を行う．第 6 章で本論文をまとめる．

2.　システム要件

本章では，LRTS の主な要件を列挙する．LRTS は，LDM を用いたリアルタイムアプリケーションの設計に活用することを意図したシミュレータであるがゆえに，設計どおりに動作するかを，リアルタイムに可視化して評価できることが重要となる．LRTS は，ひとことで述べるならば，広範囲に及ぶ複数の車両をリアルタイムにセンシングし，LDM に入力してアプリケーションを動作させ，その結果を車両にフィードバックしたうえで可視化するシミュレータである．アプリケーションには，たとえば，街全体の車両（数千台規模）をセンシングし，それらの広範囲な情報を集合してある種の最適化を行い，車両にフィードバックするリアルタイム渋滞回避制御等のアプリケーションが挙げられる．また，従来の車両に搭載されているセンサでは接近車両が検出できないようなビルの死角，あるいは見通しの悪い交差点等で衝突を警告する衝突警告アプリケーションが挙げられる．よって，LRTS は，**図 2** に示すような 4 点の

ⓒ 2014 Information Processing Society of Japan

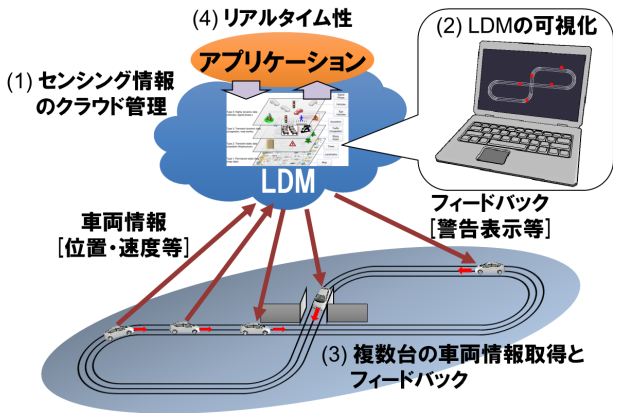

図 2　システム要件
Fig. 2　System requirements

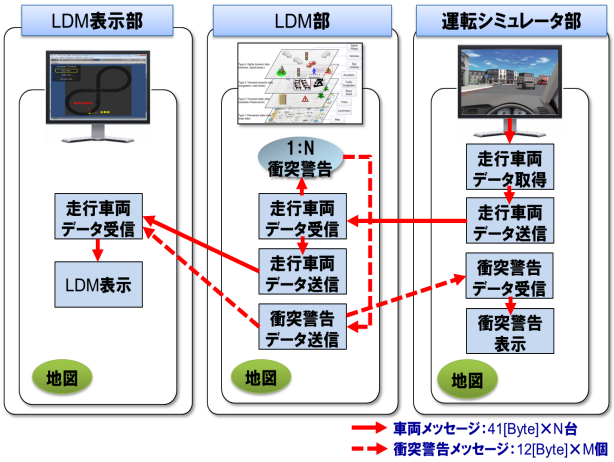

図 3　全体設計
Fig. 3　The design of LRTS

システム要件を満たすことが重要となる．すなわち，機能要件として（1）センシング情報のクラウド管理，（2）LDM の可視化，（3）複数台の車両情報取得とフィードバック，および非機能要件として（4）リアルタイム性の計 4 点である．以下，リアルタイム性の要件が厳しい衝突警告アプリケーションを例に挙げて，各要件を具体化し説明する．

（1）センシング情報のクラウド管理：LRTS は，単一の交差点のような局所に存在する車両よりも，むしろ街全体のような広範囲に及ぶ車両を対象としたセンシングとフィードバックを想定する．本研究では，これを実現するもっとも簡潔な方法のひとつとして，クラウド上に LDM とアプリケーションを配置したうえで，センシング情報をクラウドに集める方法を用いる．これにより，広範囲に及ぶセンシング情報を一括管理でき，LDM とアプリケーションは，それらの情報を用いて処理を行うことができる．

（2）LDM の可視化：LDM が，現在，どのようなデータを扱っているかを可視化する必要がある．たとえば，Type 1 の地図の形状，Type 4 の車両の位置等である．なぜならば，アプリケーションは，これらの LDM のデータを入力として計算を行うため，それらのデータが妥当かを確認できることが，システム全体の動作確認に役立つからである．すなわち，LDM を可視化することにより，LDM に対してセンシング情報が正しく入力されているかを確認できる．また，車両へのアプリケーションのフィードバックが適切に行われているかを確認できる．加えて，アプリケーションの動作が誤っていると判断された場合には，その原因が LDM 側にあるのか，それともアプリケーション側にあるのかの切り分けを容易化できる．LRTS の使用例として衝突警告アプリケーションを想定すると，最低限可視化すべき LDM の要素としては，地図の形状（Type 1）と車両の位置（Type 4）が該当する．

（3）複数台の車両情報取得とフィードバック：複数の車両のセンシングデータ（位置，速度等）を取得できること，および LDM 上のアプリケーションの計算結果（警告表示等）を車両にフィードバックできることが求められる．たとえば，衝突警告アプリケーションを例とすると，各車両の ID，位置，速度等が取得でき，アプリケーションの警告表示リクエストを自車両に送信可能であることが求められる．また，車両が，アプリケーションからのフィードバックを自車に反映できることも求められる．たとえば，警告表示リクエストを受け取った車両は，その画面に警告を表示できることが求められる．

（4）リアルタイム性：リアルタイム性が特に要求されるのは，各車両のデータをセンシングして LDM に入力し，アプリケーションが LDM のデータを用いて計算を行い，車両にフィードバックされるまでの時間である（以降，ラウンドトリップタイムと呼ぶ）．どの程度のリアルタイム性が要求されるかは，LDM 上のアプリケーションに依存する．例として，衝突警告アプリケーションを考えた場合，人間の視覚における単純反応時間は 190[ms] といわれているため [8]，前方，または側方の車両に気づいてブレーキを踏むまでの時間もそれに準じると考えると，190[ms] 以下であれば，人間と同等，またはそれ以上の反応が得られると考えることができる．そこで，本研究では，100[ms] をラウンドトリップタイムのデッドラインとして，リアルタイム性の要件を定める．

3.　設計

図 3 に，第 2 章の要件を踏まえた LRTS の全体設計を示す．ここでいう全体設計とは，要件を充足するために必要な各部とその機能，および各部がやり取りするデータを定義することである．アプリケーションとして，1:N 衝突警告アプリケーション（自車周辺の N 台と自車との衝突を予測し警告を発生）を用いた場合を示す．

LRTS は，運転シミュレータ部，LDM 部，ならびに LDM 表示部の 3 部構成からなり，それぞれ共通の地図データを持つ．運転シミュレータ部は，図 2 における車両とその走

ⓒ 2014 Information Processing Society of Japan

行環境である．はじめに，走行車両データ取得機能において，N 台分の車両データを取得する．衝突警告アプリケーションが必要とする車両データは，時刻，車両 ID，位置，速度等で，1 台につき 41[Byte] からなる．シミュレータ上に N 台の車両が存在する場合は，合計で 41 × N[Byte] となる．それを 1 個の車両メッセージとしてパッキングする．次に，走行車両データ送信機能において，後述する LDM 部に対して車両メッセージをリアルタイムに送信することにより，LDM 部に衝突警告判定を行わせる．LDM 部から，時刻，警告種別等からなる衝突警告メッセージ（12[Byte] × M[台]）が送信されてきた場合は，衝突警告データ受信機能がそれを受信し，衝突警告表示機能が，警告を表示して運転者に危険を知らせる．あくまでも衝突警告判定は，運転シミュレータ部では行わず，LDM 部で行う．

LDM 部は，図 2 におけるクラウド部分である．走行車両データ受信機能は，車両メッセージを運転シミュレータ部からリアルタイムに受信し，地図データとのマップマッチング（例．車両が地図上のどの道路のどの位置を走行しているかの計算等）をリアルタイムに行った後に，1:N 衝突警告アプリケーションへ送信する．1:N 衝突警告アプリケーションは，自車と他車の位置と速度により，SDA（Stopping Distance Algorithm）[13] を用いて衝突警告を出すか否かを計算する．SDA は，停止時の車間距離が，あらかじめ定めた安全距離よりも小さくなる場合に衝突警告を出すためのアルゴリズムである．衝突の危険性があると判断され，衝突警告を出す場合，衝突警告データ送信機能により，LDM 表示部と運転シミュレータ部に衝突警告メッセージを送信する．また，運転シミュレータ部から受信した車両メッセージは，走行車両データ送信機能により，LDM 表示部にそのまま転送する．

LDM 表示部は，図 2 における LDM を可視化する部分である．LDM 表示部は，走行車両データ受信機能によって，LDM 部からリアルタイムに車両メッセージを受信し，LDM 表示機能によって，地図と重ね合わせて表示する．衝突警告メッセージを受信した場合は，どの車両に対して警告が発生したかを色分けして表示する．

4. 実装

本章では，LRTS の各部の実装について説明する．各部は，それぞれ異なる PC 上に実装し，PC 間を 1Gbps のイーサネットで接続する．データの送受信には UDP を用いる．また，地図データとしては，Open Street Map[4] で使用されている OSM データ形式を共通に使用する．Open Street Map は，現在，もっとも広く普及しているフリーの地図データである．各部が持つ共通の地図データには，**図 4** で示すように，450[m] 四方の平面に，1 周 1340 [m] の 8 の字コースを配置した仮想地図を使用する．中央の交差点には信号がなく，衝突しない限り自由に車両が交差できるようにする．仮想地図は，Open Street Map のエディタである JOSM（Java Open Street Map Editor）[6] を用いて作成する．地図データの形式については，第 4.3 節で詳しく説明する．

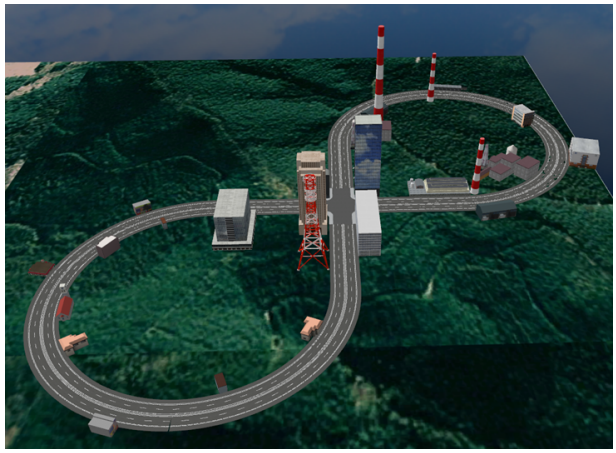

図 4 仮想地図
Fig. 4 Virtual map

4.1 運転シミュレータ部

運転シミュレータ部は，運転シミュレータと衝突警告表示アプリケーションで構成する．運転シミュレータが，走行車両データ取得機能と走行車両データ送信機能を担当し，衝突警告表示アプリケーションが，衝突警告データ受信機能と衝突警告表示機能を担当する．

運転シミュレータには，フォーラムエイト社の UC-win/Road Driving Sim v9.0[9]（以降，UC-win/Road と略す）を用いる．UC-win/Road では，UC-win/Road 上の車両に乗り込み，アクセル・ブレーキ・ステアリングを操作して運転を模擬することができる．UC-win/Road では，フレーム書き換えごとに，シミュレータ上の全車両のデータを取得することができる．そのデータには，位置，速度はもちろんのこと，たとえば，車両のヨー角，ピッチ角，ロール角，車両の方向，エンジンの RPM，ブレーキペダルの入力値，ハンドルの回転率など，約 60 種類のデータを各車両ごとに取得できる．それらのうち，ストリーム LDM と衝突警告アプリケーションが使用する，位置，速度等の 9 種類のデータ（41[Byte]）を各車両ごとに抽出し，それらのデータをまとめて 1 個の車両メッセージにした後に LDM 部に送信する．この処理は，Delphi 言語で記述する UC-win/Road のプラグインを用いて実装する．

衝突警告表示アプリケーションは，Windows アプリケーションとして C#で実装する．本アプリケーションは，LDM 部から送信されてくる衝突警告メッセージを解釈し，運転者に注意を喚起する．その様子を**図 5** と**図 6** に示す．右上のマークを表示しているアプリケーションが衝突警告表示アプリケーションである．図 5 は，時速約 60[km/h]

ⓒ 2014 Information Processing Society of Japan

図 5　運転シミュレータ部（警告なし）
Fig. 5　Driving simulator(no warning)

図 6　運転シミュレータ部（警告あり）
Fig. 6　Driving simulator(warning)

で前方に車両が存在しない道路を走行中の様子を示している．緑色の表示となり警告がないことを示す．一方，図 6 は，交差点で待機している車両の後方を，時速約 80[km/h] で接近中の様子を示している．マークが赤の警告表示に変わり，運転者に注意を喚起する．もちろん，注意を喚起されたのちに，運転者がブレーキを踏み，衝突の危険性がなくなった場合は，マークは緑の警告なしに変化する．

4.2　LDM 部

LDM 部は，LDM と衝突警告アプリケーションから構成される．LDM には，リアルタイム動作を重視したストリーム LDM[11] を使用する．ストリーム LDM は，データストリーム管理システム（DSMS: Data Stream Management System）を基盤にした LDM である．DSMS は，非同期に到着する無限個のデータ列（以降，ストリームと呼ぶ）を，あらかじめ構築しておいた問い合わせ（以降，クエリと呼ぶ）に入力することで，アプリケーションに必要なデータを，メモリ上でリアルタイムに生成し，ストリームとして出力することが可能である．ストリーム LDM の基盤となる DSMS は，eDSMS[12] である．eDSMS は，車載組込みシステム向けの DSMS として開発されており，クエリをあらかじめコンパイルしバイナリとして eDSMS に組み込むために，Aurora[2]，Borealis[1] 等の従来の DSMS と比較して，リアルタイム性に優れるという特徴を持つ．

eDSMS を用いてストリーム LDM を実装する方法を概説する．eDSMS では，基本となる演算子を組み合わせてクエリを記述する．たとえば，filter という演算子は，ストリームを構成する各データ組（以降，タプルと呼ぶ）を入力として，それらにガード条件を適用し，その条件を満たすタプルに限り出力するという機能を持つ．このような演算子を組み合わせることにより，リアルタイム性を重視した LDM を実現することができる．たとえば，複数の車両から時系列に従って入力される車両データ (位置，速度等）を解析して，自車の半径 X[m] 以内の車両をリアルタイムに抽出するといったデータ処理が可能となる．また，地図データと前述の車両データと演算することで，マップマッチングもリアルタイムに行うことができる．ストリーム LDM は，LDM の Type 1 から Type 4 までのデータ構造のうち，Type 1 の地図と Type 4 の車両を扱うことが可能である．より詳細な実装に関しては，文献 [11]，および文献 [10] を参照されたい．

LDM 部の動作の詳細は，以下のとおりである．ストリーム LDM は，運転シミュレータ部から，同一のタイムスタンプを持つ車両メッセージを受信する．そのメッセージを解析し，自車データと他車データに分離した後，ストリーム LDM のキューにそれぞれを入れる．かつ地図データをもキューに入れる．自車データ，他車データ，ならびに地図データをストリーム LDM のキューに入れた後，マップマッチング処理を行う．すなわち，自車・他車がどの道路のどの部分をどの向きで走行しているかを算出する．その結果を，衝突警告アプリケーションにストリームとして入力する．

衝突警告アプリケーションは，LDM 部からマップマッチング結果をストリームで受け取り，衝突警告クエリを実行する．衝突警告クエリとは，eDSMS の演算子を組み合わせて実装した衝突警告を行うデータフロー処理であり，前方衝突と交差時衝突の 2 種類の衝突可能性を計算し衝突警告メッセージを出力する．たとえば，前方衝突警告の場合は，はじめにフィルタリング処理を行い，自車の近傍 200[m] 以内の他車を抽出する．これは，自車が時速 100[km] で走行している場合でも停止距離の目安が 112[m] といわれているため，200[m] 以上離れている他車に対して警告を出す必要性は低いといえるからである．フィルタリング処理を行った後，自車の道路と同一の道路を同一の向きで走り，かつ直近交差点からの距離が自車よりも短い他車を抽出する．そのような他車に対して前述した SDA のアルゴリズムを実行し，衝突警告を出すか否かの判定を得る．これらの処理を，車両メッセージを受信したタイミングごとに繰り返す．このように，衝突警告アプリケーションは，地図データと車両をマップマッチングしたうえで衝突警告を出すか否かを判定するため，**図 7** に示すように，自車が走行する同一道路上の車両に限り，衝突警告の対象とすることができ，対向車線の車両は衝突警告対象外にで

ⓒ 2014 Information Processing Society of Japan

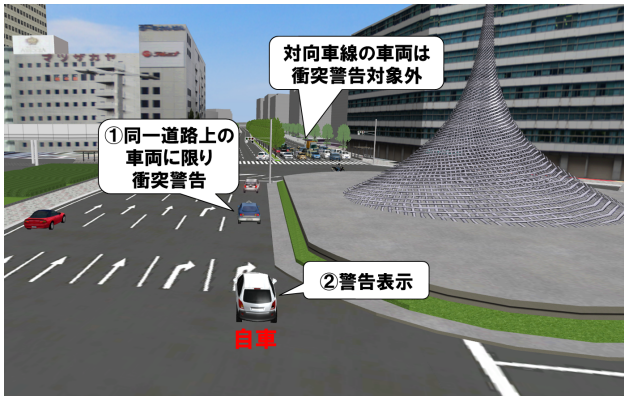

図 7　マップマッチング
Fig. 7　Map matching

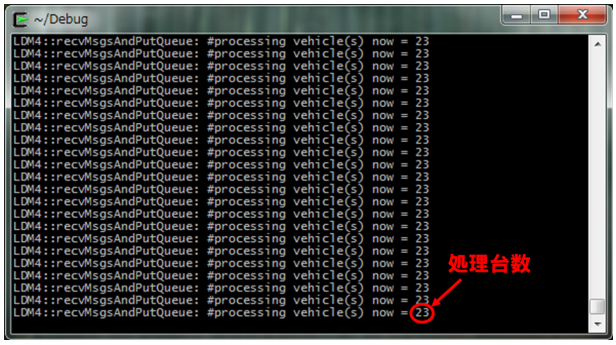

図 8　LDM 部の処理画面
Fig. 8　The screen shot of LDM processing

きる．これにより，処理負荷の軽減，ならびに誤報（False positive）の頻発を防止できる．

以上のような処理の流れで，ストリーム LDM と衝突警告アプリケーションは，運転シミュレータ部から送信される車両メッセージをリアルタイムに処理する．その実行の様子を図 8 に示す．現在，自車と他車を合計してどれくらいの台数をストリーム LDM と衝突警告アプリケーションが処理しているかがリアルタイムに表示されるようになっている．すなわち，前述した処理のうち，車両メッセージを自車データと他車データに分離した後，ストリーム LDM のキューにどれだけの車両台数が入力されたかを表示している．

4.3　LDM 表示部

LDM 表示部には，3D ゲームエンジンの Unity[7] を用いる．Unity は，ゲーム開発用に開発されたプラットフォームであり，パワフルなレンダリングエンジンを用いて少ないコーディング量でクオリティの高いリアルタイム 3D コンテンツを実現できる．また，Windows，Linux，Android 等，マルチプラットフォームに対応していることも大きな特徴である．Unity を用いると，地図（Type 1）と車両（Type 4）をリアルタイムに可視化するのが容易になる．

Unity を用いて LDM 表示機能を実装する方法は，以下のとおりである．地図（Type 1）の可視化に関しては，OSM 形式の地図データを読み込み Unity の画面に表示する．OSM 形式は，図 9 に示すように，XML のフォーマットを使用している．OSM 形式では，ノードと呼ばれる点とノードの接続を記述したウェイと呼ばれるノードの順序列により，道路線形情報が定義される．図 9 では，(緯度, 経度）がそれぞれ (0.0, 0.0)，(0.0, 0.001)，(0.0, 0.002) のノードを定義した後に，それらのノードを順に結んだウェイを定義している．

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version='0.6' upload='true' generator='tytlabs'>
  <node id='-1' action='modify'
   visible='true' lat='0.0' lon='0.0' />
  <node id='-2' action='modify'
   visible='true' lat='0.0' lon='0.001' />
  <node id='-3' action='modify'
   visible='true' lat='0.0' lon='0.002' />
  <way id='-1' action='modify' visible='true'>
    <nd ref='-1' />
    <nd ref='-2' />
    <nd ref='-3' />
  </way>
</osm>
```

図 9　OSM 形式の XML 例
Fig. 9　The XML example of OSM

このように OSM 形式は，XML 形式で記述されるため，各種プログラミング言語の XML パーサを用いることで，非常に少ないコーディング量で読み込みを実装することができる．Unity に関しても同様で，C#による XML パーサを用い，ノードとウェイの構造体を要素とするリスト構造に OSM 形式のデータを読み込む．その後，ノードとウェイの緯度と経度を 2 次元座標に変換し，それに沿って道路のテクスチャを配置することで，地図が表示される．

車両（Type 4）の可視化に関しては，以下のように実装する．Unity により実装した走行車両データ受信機能により，同一のタイムスタンプを持つ複数の車両メッセージ（位置・速度等）と衝突警告メッセージ（前方衝突・交差時衝突）を受信する．そのメッセージ内容を解析した後に，各車両の位置と向きを加味して車両の 3D モデルを画面上に配置する．自車は黄色，他車は青色で表示する．衝突警告メッセージを受信した場合は，その警告の対象となる車両を赤色で表示する．車両メッセージ，ならびに警告メッセージを受信したときに，これらの動作を行うことで，地図上に車両がリアルタイムに表示される．

LDM 表示部では，仮想的な 3 次元空間を再構築しているので，視点の切り替えを容易に実現できる．図 10 は，Unity によって，8 の字のマップ上を複数台の車両が走行している様子を Top View で表示したスクリーンショットである．複数の車両が走行し，かつ，自車と衝突警告の対

ⓒ 2014 Information Processing Society of Japan

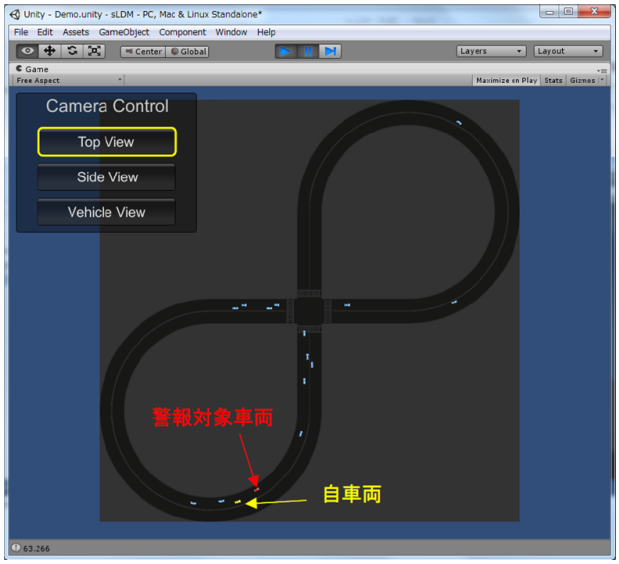

図 10　LDM 表示部 (Top View)
Fig. 10　LDM viewer(Top View)

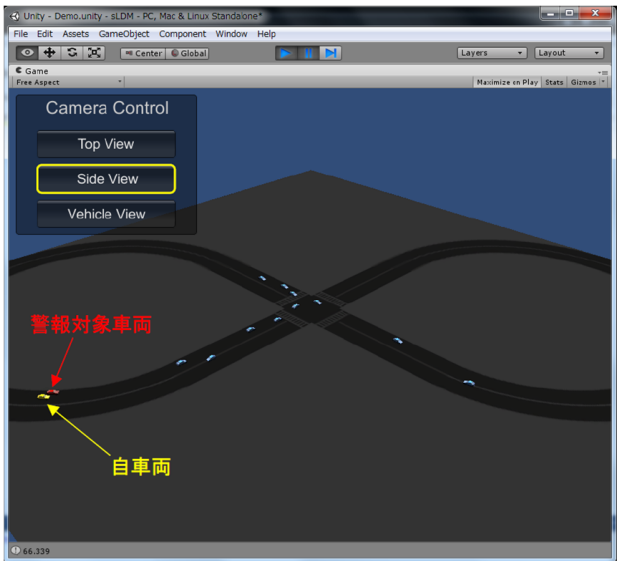

図 11　LDM 表示部 (Side View)
Fig. 11　LDM viewer(Side View)

象車両が識別されて表示されているのがわかる．視点の切り替えは，Side view（横からの俯瞰）と Vehicle view（運転者からの視点）が可能である．視点の切り替えの例として，**図 11** と**図 12** にそれぞれ，Side View，Vehicle View を示す．

5.　評価

本章では，LRTS が，第 2 章に述べたシステム要件を充足しているかどうかを評価する．はじめに（1）～（3）の機能要件について定性的に評価し，次に（4）の非機能要件について定量的に評価する．

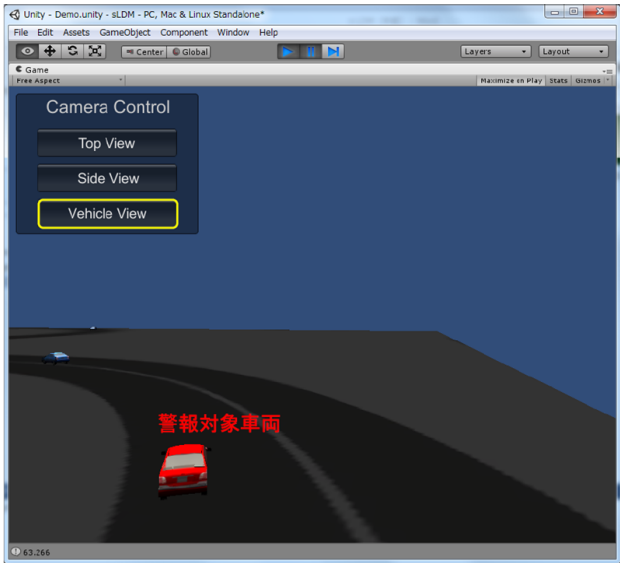

図 12　LDM 表示部 (Vehicle View)
Fig. 12　LDM viewer(Vehicle View)

5.1　機能要件の充足性

(1) センシング情報のクラウド管理：図 8 に示したように，運転シミュレータに存在するすべての車両情報が LDM 部（クラウド）に集められ，そのうえでアプリケーションが動作可能であることが確認できる．

(2) LDM の可視化：図 10，図 11，および図 12 に示したように，LDM の Type 1（地図）と Type 4 （車両）をリアルタイムに可視化できることが確認できる．また，衝突警告アプリケーションが警告を発生した場合，警告対象の車両が赤で示されることが確認できる．これにより，LDM とアプリケーションの動作がともにリアルタイムに可視化可能であることが確認できる．

(3) 複数台の車両情報取得とフィードバック：図 8，図 5，および図 6 に示したように，車両情報がシミュレータから取得できること，ならびに衝突警告が車両にフィードバック可能であることが確認できる．

5.2　非機能要件の充足性

(4) リアルタイム性：リアルタイム性の要件である，ラウンドトリップタイム 100[ms] 以下を実現できているかを評価する．評価環境を**表 1** に示す．LDM 表示部，LDM 部，ならびに運転シミュレータ部は，それぞれ別個の PC に実装し，1[Gbps] の Ethernet で接続する．ラウンドトリップタイムは，運転シミュレータ部（PC3）が車両メッセージを LDM 部（PC2）送信し，LDM 部（PC2）から運転シミュレータ部（PC3）に衝突警告メッセージが返ってくるまでの時間となる．

評価シナリオとしては，はじめに，図 4 で示した運転シミュレータ部の仮想地図上で前方衝突警告と交差時衝突警告ができる程度の他車を生成する．交差時衝突警告のため

ⓒ 2014 Information Processing Society of Japan

表 1　評価環境
Table 1　Evaluation environment

部名	項目	詳細
LDM 表示部 (PC1)	CPU	Intel Core i5-2540M 2.6GHz
	メモリ	8GB
	OS	Windows7(64bit)
LDM 部 (PC2)	CPU	Intel Core2 Duo E6600 3GHz
	メモリ	3GB
	OS	Cygwin on Windows7(32bit)
運転シミュレータ部 (PC3)	CPU	Intel Xeon E3-1240 v3 3.4GHz
	メモリ	16GB
	OS	Windows7(64bit)

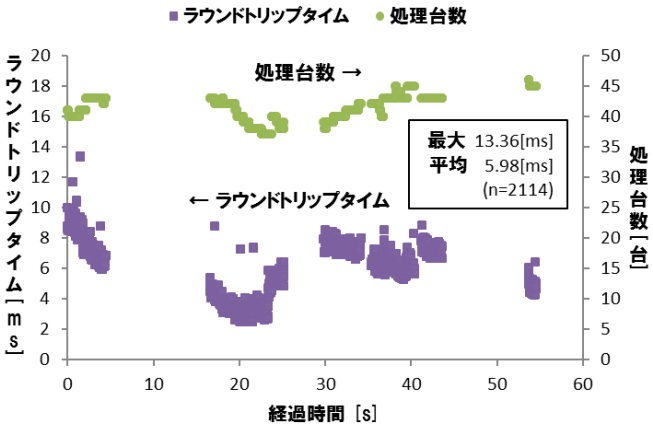

図 13　ラウンドトリップタイム（x = 経過時間）
Fig. 13　Round trip time(x = elapsed time)

に，交差点が渋滞して車両が停止しないように他車を生成する場合，本仮想地図では，1 時間当たり 2000 台のペースで車両を生成し，仮想地図上に同時に存在する車両が 50 台以下にすると，交差点での渋滞が起きずに車両が走行でき，評価に適切な環境を構築できる．次に，運転シミュレータ上で仮想地図の 8 の字コースを 1 周し，自車が前車に急接近する，あるいは交差点で左右からの他車に急接近するような運転をして警告を発生させる．そのときのラウンドトリップタイムを測定する．

評価シナリオ実行時の経過時間に対するラウンドトリップタイムの評価結果を図 **13** に示す．横軸が経過時間，左縦軸がラウンドトリップタイム，右縦軸が LDM 部の処理台数（つまり，仮想地図に存在する車両台数）をそれぞれ表す．警告が発生した部分に関してのみプロットしているため，断続的に空白部分が存在する．サンプル数 n=2114 回の警告発生に関して，ラウンドトリップタイムは，平均が 5.98[ms] で，最大が 13.36[ms] であり，100[ms] 以内のラウンドトリップタイムを実現できていることが確認できる．

また，処理台数に対するラウンドトリップタイムの評価結果を図 **14** に示す．横軸が処理台数，左縦軸がラウンドトリップタイムを表す．前述したように，渋滞しない程度

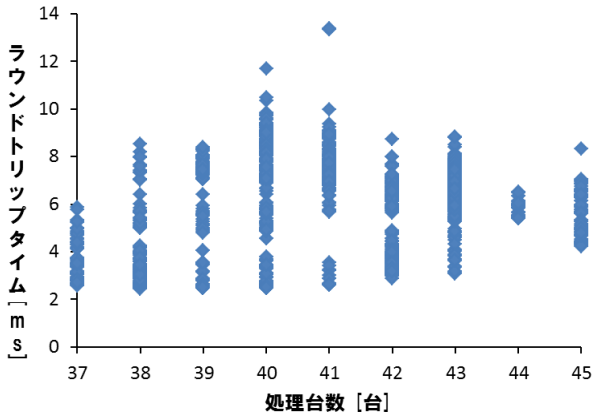

図 14　ラウンドトリップタイム（x = 処理台数）
Fig. 14　Round trip time(x = processing vehicles)

に車両を生成しているため，台数が 37 台から 45 台の評価結果となっている．ラウンドトリップタイムが処理台数に比例しないのは，第 4.2 節で述べたように，衝突警告の対象となる車両が自車の 200[m] 近傍の車両であって，運転シミュレータ上のすべての車両ではないことに起因する．すなわち，運転シミュレータ全体では車両が多いにもかかわらず，自車の周辺には車両が少ないために，マップマッチング以降の衝突警告処理（前方・交差車両の抽出，SDA アルゴリズムの実行等）が短縮されることがあるためである．

5.3　評価のまとめ

機能要件と非機能要件ともに，充足していることを確認した．すなわち，運転シミュレータ部が生成した 37 台〜45 台の車両データを，クラウドの LDM 部において，100[ms] 以内のラウンドトリップタイムで処理可能であり，運転シミュレータ部にフィードバック可能である．よって，LRTS により，アプリケーションからの衝突警告の応答が，自車の衝突間際において適切なタイミングで行われるかを評価可能であることが確認できた．

また，LDM 表示部においては，地図（Type 1) と車両（Type 4）を重ね合わせて表示でき，LDM とアプリケーションの動作がリアルタイムで把握可能である．これらのことから，LRTS は，リアルタイム性の要件を充足可能であると同時に，衝突警告アプリケーションにおいて LDM とアプリケーションの動作を可視化でき，アプリケーションの動作評価が可能であることが確認できた．

6.　おわりに

本論文では，LDM を用いたアプリケーションの動作検証をリアルタイムで行うことを目的として，ストリーム LDM を用いて LDM リアルタイムシミュレータ（LRTS）を構築し，衝突警告アプリケーションを例として評価を行った．その結果，LRTS は，衝突警告アプリケーション

ⓒ 2014 Information Processing Society of Japan

において，LDM とアプリケーションの動作をリアルタイムに可視化できることが示された．また，リアルタイム性の評価において，運転シミュレータ部と LDM 部のラウンドトリップタイムは，車両台数 37 台〜45 台において，平均が 5.98[ms] で，最大が 13.36[ms] であり，衝突警告アプリケーションの要件である 100[ms] 以内のラウンドトリップタイムを確保可能であることが示された．以上により，LRTS により，LDM を用いた ITS アプリケーションのリアルタイム動作評価が行えることが確認できた．

今後の展望としては，衝突警告以外のアプリケーション（例．渋滞回避制御）の動作評価を行って LRTS の適用範囲を拡大すること，車両とクラウドの無線通信の遅延とそのばらつきを考慮してアプリケーションの動作を評価可能にすること，多数の車両が必要なアプリケーション（例．街全体の車両を対象とする燃費最適化）に対応するためにスケーラビリティを追求すること等が挙げられる．

謝辞 本研究の一部は，SCOPE(12180615) と科研費 (25240007) の助成を受けている．

参考文献

[1] Abadi, D., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y. and Zdonik, S.: The Design of the Borealis Stream Processing Engine, *In proc. of Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, pp. 277–289 (2005).

[2] Abadi, D., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N. and Zdonik, S.: Aurora: a new model and architecture for data stream management, *The VLDB Journal*, Vol. 12, No. 2, pp. 120–139 (2003).

[3] ETSI: Intelligent transport systems(ITS); Vehicular communications; Basic set of applications; Local dynamic map(LDM); Rationale for and guidance on standardization, TR 102 863, European Telecommunications Standards Institute (2010).

[4] Open Street Map project: Open Street Map, Open Street Map project (online), available from ⟨http://www.openstreetmap.org/⟩ (accessed 2014-06-03).

[5] SAFESPOT: SAFESPOT core architecture - LDM API and usage reference (2010).

[6] Scholz, I.: JOSM: Java Open Street Map Editor, JOSM developers (online), available from ⟨https://josm.openstreetmap.de/⟩ (accessed 2014-06-03).

[7] Unity Technologies: Unity, Unity Technologies (online), available from ⟨http://unity3d.com/⟩ (accessed 2014-06-03).

[8] Welford, A.: *Reaction Times*, pp. 371–381, Academic Press (1980).

[9] フォーラムエイト：UC-win/Road Ver.9 Driving Sim, フォーラムエイト (online), available from ⟨http://www.forum8.co.jp/product/ucwin/road/ucwin-road-1.htm⟩ (accessed 2014-06-03).

[10] 伊藤信一，山口晃広，佐藤健哉，本田晋也，高田広章：ストリーム LDM における地図データのストリーム化機構の設計と評価，情報処理学会研究報告 Vol.2013-ITS-53, pp. 1–8 (2013).

[11] 山口晃広，島田秀輝，伊藤信一，石原直樹，本田晋也，佐藤健哉，高田広章：ストリーム LDM：データストリーム管理システムによる LDM の高速化，第 11 回 ITS シンポジウム 2012，pp. 127–130 (2012).

[12] 勝沼聡，山口晃広，熊谷康太，本田晋也，佐藤健哉，高田広章：車載組込みシステム向けデータストリーム管理システムの開発，電子情報通信学会論文誌，Vol. J95-D, No. 12, pp. 2031–2047 (2012).

[13] 平岡敏洋，田中雅樹，熊本博光， 泉達也，畑中健一：衝突回避減速度を用いたリスク評価指標の提案，自動車技術会論文集，Vol. 40, No. 2, pp. 545–551 (2009).

[14] 北山浩透：クルマからのデータ活用による新サービスとプラットフォーム，情報処理，Vol. 54, No. 4, pp. 337–343 (2013).

ⓒ 2014 Information Processing Society of Japan