大規模分散環境における 人流・交通流シミュレータの設計と課題

平野 流[1] 廣井 慧[2] 米澤 拓郎[1] 河口 信夫[1,3]

概要：人流・交通流のシミュレーションは，都市計画や避難計画，渋滞の予測・解析等で活用されている．近年，IoT デバイスやスマート社会基盤の普及により，大規模化や異種シミュレータ間連携の他に，データ同化や動的拡張などの需要が高まっている．これらの研究は個々の研究では行われているが，これら四つの機能が統合されたシミュレーション環境は，必要に応じて利用できる状況にはなっていない．本論文ではこれらの統合を考慮しつつ，主に大規模化に着目し，大規模分散環境におけるシミュレータ環境の設計と評価を行う．本シミュレータ環境では，時間と空間によって分散し，通信を用いて連携する時空間オブジェクトモデルを採用している．最後に分散環境における性能の測定・評価を行った．

A Design and Issue of Person Trip and Traffic Flow Simulator in Large-scale Distributed Environment

RUI HIRANO[1] KEI HIROI[2] TAKURO YONEZAWA[1] NOBUO KAWAGUCHI[1,3]

1. はじめに

人や車が多数行き交う社会において，人流・交通流のシミュレーションは重要な技術となっている．一般的に，人流・交通流のシミュレーションにはマルチエージェントシミュレーションが多く用いられる．マルチエージェントシミュレーションでは，現実を模倣した個々の自律したモデルを作成し，相互に影響させて起こる現象を分析する．交通や防災・社会科学などに活用されており，特に人流・交通流では，災害時の避難誘導や商業施設の混雑緩和，動線分析によるサービス向上などを目的として活用される．既存の人流・交通流シミュレータには，PTV VisWalk[1] と呼ばれる人流シミュレータや，オープンソースの交通流シミュレータである SUMO[2]，避難シミュレータである ESCAPES[3] など多数存在する．

近年，スマート社会基盤の運用や IoT デバイスの普及により，多種多様なデータが大量かつ瞬時に手に入るようになっている．それにより，今までの規模を超えた都市レベルの超大規模なシミュレーションを，安定的に長期間行うことも必要になってくると考えられる．

また，都市レベルのシミュレーションには一つの対象物だけではなく，人や車，鉄道，信号などの様々な要素を関連させる必要があることは言うまでもない．しかしながら，従来のシミュレータでは少数のエージェントを対象とするものも多く，その対象とならないものについてはシミュレーションを行えない．そのため，複数の異種シミュレータを連携させ，複数の対象物による相互作用を考慮しながら行える仕組みも必要となる．

さらに，実世界のデータを用いた，リアルタイム性のあるシミュレーションの需要も発生している．実世界データを用いる場合，センサーの設置場所により，データの取得可能領域に偏りが発生するといった課題も存在する．そのため，データ取得可能な領域のデータを参考にし，取得できていない領域の状態を推測するデータ同化という研究も行われている．

加えて，既存のシミュレータの多くは実行前に設定を完了し，実行中に変更できない静的なシミュレーションを前提にしている．リアルタイムなシミュレーションをふまえ

[1] 名古屋大学大学院工学研究科 Graduate School of Engineering, Nagoya University
[2] 京都大学防災研究所 Disaster Prevention Research Institute, Kyoto University
[3] 名古屋大学未来社会創造機構 Institutes of Innovation for Future Society, Nagoya University

© 2020 Information Processing Society of Japan

ると，エリアやエージェントの種類などシミュレーションの構成を，実行中に拡張できる必要があると考えられる．

大規模化，異種シミュレータ間連携，データ同化，動的拡張という四つの研究課題に対して，これらを統合的に組み合わせたシミュレーション環境は，必要に応じて利用できる状況にはなっていない．我々は，これら四つの機能を統合的に組み合わせたシミュレーション環境の開発と実用化を目指している．本論文では，異種シミュレータ間連携，データ同化，動的拡張という機能の統合を考慮しつつ，主に大規模化に着目し，大規模分散環境におけるアーキテクチャの設計と実験評価を行う．

本論文の構成は次のようになっている．まず，第 2 節で人流・交通流シミュレータの大規模化に関する既存研究を示す．次に，第 3 節で本シミュレータ環境の設計コンセプトを説明する．第 4 節では現状の実装されている機能について述べる．そして，第 5 節でそれらの設計が実装されたシミュレータを用いてパフォーマンス測定・評価を行い，第 6 節でまとめとする．

2. 関連研究

シミュレーションの大規模化手法には，大きく分けて，計算能力の向上による手法と計算資源の拡張による手法が挙げられる．計算能力の向上による大規模化手法は，GPU(Graphic Processing Unit) や HPC(High Performance Conputing) といった高性能な計算技術を用いて大規模化を行う手法である．Richmond ら [4] は，エージェントの処理を GPU で行うことで高速化し，大規模化を図った．また，先山ら [5] は，GPU によるマルチエージェントシミュレーションライブラリの設計と実装を行い，CPU と比べて処理性能の向上を示した．

計算資源の拡張による大規模化手法は，いわゆる分散型シミュレーションとよばれる手法である．シミュレーションにおける計算をモジュール毎に分散できる設計にし，複数の計算機上で通信を行い，大規模化を実現する．小藤ら [6] は，PC の分散台数によるシミュレーション時間の変化を評価し，一定台数を超えると平均計算時間が短縮することを示した．

計算能力の向上による手法は，複数の計算資源の利用を考慮していないことも多く，エージェント数やエリアの規模に限界が生じる可能性がある．一方，分散型シミュレーションは，独立した機能をもつモジュール群を分散させることで，計算処理の分散や異なるシミュレータによる連携，エージェントやエリアなどの動的な追加削除にも対応しやすいといった特徴をもつ．そのため，本研究では分散型シミュレーションを採用する．

既存の分散型シミュレータの代表例として HLA(High Level Architecture) に従うシミュレータ基盤がある．HLA とは，1990 年代に米国防省の主導で標準化された，異種

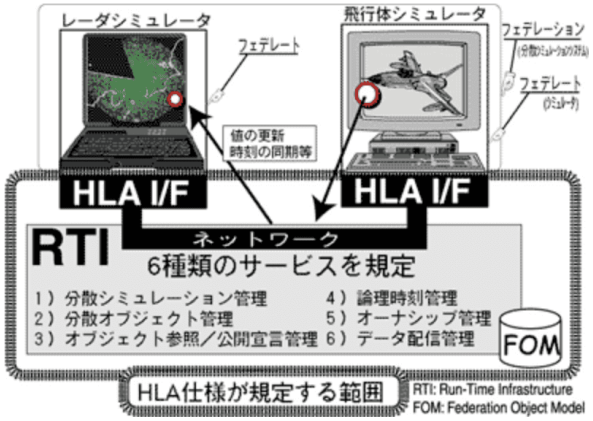

図 1　HLA と RTI の概要 [9]

シミュレータが連携するための規格 [7] である．HLA では図 1 のように，RTI(Run Time Interface) と呼ばれるミドルウェアを通して，分散されたモジュールの時刻同期やデータ配信，オブジェクト管理等が行われる．Muhammad ら [8] は，HLA に従うシミュレータ基盤のアーキテクチャを考案している．しかしながら，HLA は異種シミュレータが連携することに焦点が当てられており，大規模化に対しては焦点をあてられていない．また，RTI へのサービスが集中により，通信量などのオーバーヘッドが発生することや，RTI を提供しているベンダによって通信プロトコルが変わるため，複数の RTI を協調させるのは難しいといった問題が指摘されている [9][10].

3. 設計コンセプト

3.1 Synerex による分散システム構成

分散型シミュレーション環境の課題として，データ交換基盤へのデータ集中が挙げられる．本シミュレーション環境では，データ集中への対応として Synerex[11][12] という分散システムを用いる．Synerex とは，我々が開発している，スマート社会における需給交換基盤であり，高い分散性能と柔軟性をもつ．Synerex を用いた構成例を図 2 に示す．Synerex は，データ交換基盤とプロバイダとよばれる分散モジュールによって構成され，通信によってデータを交換する．また，図 3 のようにデータ交換基盤の分散が可能である点も特徴の一つである．異なる Synerex 間における情報は，Gateway プロバイダによって交換が行われ，情報伝達の一貫性が保たれる．この Synerex の分散機能により，データ交換基盤へのデータ集中の抑制が可能になる．

3.2 ドメイン・エリアによる時空間オブジェクトモデル

Synerex を用いた本シミュレーション環境の構成を，図 4 に示す．本シミュレーション環境では，時空間オブジェクトモデル [13] の仕組みを採用している．歩行者や車などのエージェントプロバイダ，可視化を行う可視化プロバイダ，

© 2020 Information Processing Society of Japan

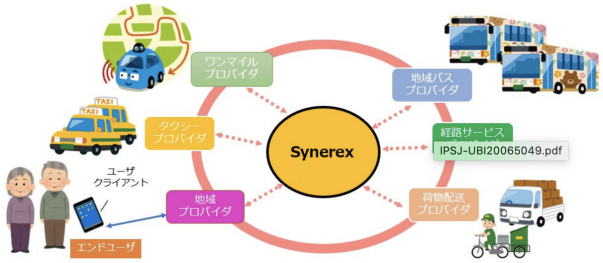

図 2　Synerex を用いたサービスシナリオ例

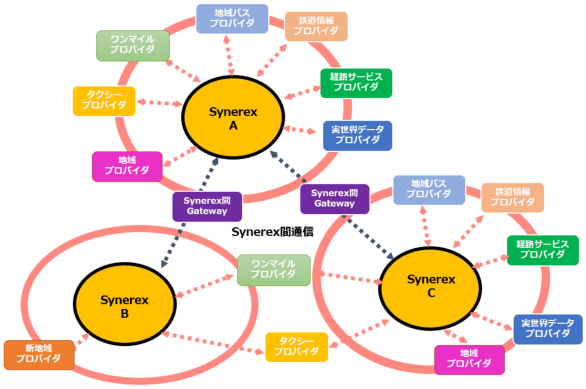

図 3　Gateway による Synerex 間接続

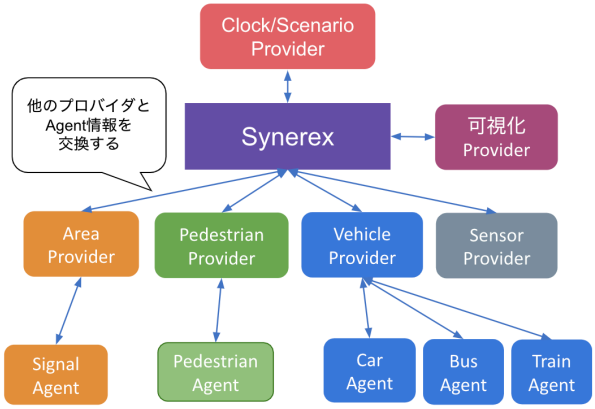

図 4　プラットフォームの概要

クロック同期，各種命令を行うプロバイダなど，単一の役割をプロバイダごとに割り当て，データ交換基盤 Synerex を通じて連携する．

また，プロバイダはドメイン毎だけでなく，エリア毎による分割が可能である，例えば，図 5 のように，対象エリアをエリア A とエリア B にわけ，それぞれのエージェントを各エリアで計算し，可視化時に統合することも可能になる．

3.3　重複エリアによる通信数抑制

上述のプロバイダ分割には，プロバイダの増加により通信回数が増えるという副次的な問題が発生する．特に，エリア分割による影響が大きく起因している．エリアを分割する場合，エージェントの計算時に，同じエリアの異種ドメインのエージェントの取得に加えて，エリア境界付近での相互影響を考慮するため，隣接しているエリアのエージェントを取得する必要がある [14]．計算後はエリア境界におけるエージェントのずれを解消するため，再度隣接エリアと情報を交換し，更新をおこなう．そのため，隣接しているエリア数を nei，ドメイン数を dom とすると，エージェントの情報交換に関わる総通信回数 com は，以下のようになる．

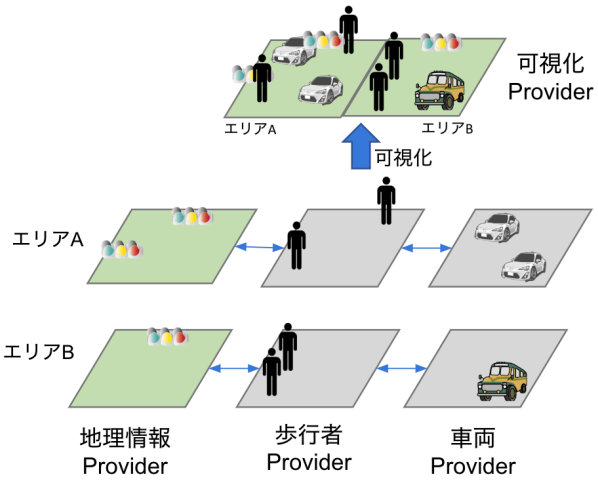

図 5　エリアとドメインによるプロバイダの分割

$$com = nei \cdot dom + 1 \cdot (dom - 1) + nei \cdot 1 \qquad (1)$$

ここで，(1) 式の第一項は隣接エリアの全ドメインのエージェントプロバイダとの通信回数，第二項は同じエリアの異種ドメインのエージェントプロバイダとの通信回数，第三項は計算後のエージェント更新に伴う通信回数を表している．よって，n 個のエージェントプロバイダ間の総通信回数 c は

$$c = \sum_{i=1}^{n} dom_i + \sum_{i=1}^{n} nei_i + \sum_{i=1}^{n} nei_i \cdot dom_i - n \qquad (2)$$

と表せる．エリア間の通信では周辺のエリアのみと通信を行うため，全プロバイダによるメッシュ通信とはならない．そのようなエリア間通信のローカリティにより，ある程度の通信回数は抑制されるものの，さらなる工夫の余地があると言える．

本アーキテクチャでは，通信回数のさらなる抑制のため，エリアの境界に重複エリアを用いる．重複エリアとは，図 6 のように，隣接しているエリアを重なるように配置したエリアである．重複エリア内のエージェントは両エリアが管理する．また，重複エリアの中心線によって分けられるエリアを管理エリアと呼ぶ．各エージェントプロバイダは，重複エリアのエージェントを考慮し，管理エリアにおけるエージェントを計算すればよい．そのため，隣接しているエリアのエージェント情報を，最初に取得する必要がなくなる．そのため，重複エリアを考慮した場合の総通信

© 2020 Information Processing Society of Japan

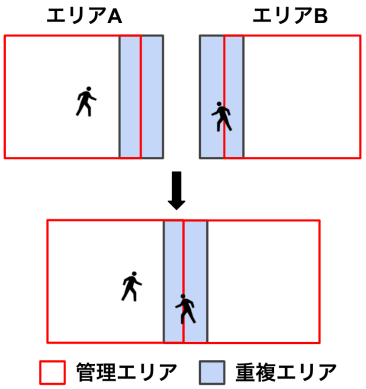

図 6　重複エリアの概要

回数 com は以下のようになる．

$$com = 1 \cdot (dom - 1) + nei \cdot 1 \tag{3}$$

ここで，(3) 式の第一項は同じエリアの異種ドメインのエージェントプロバイダとの通信回数，第二項は計算後のエージェント更新に伴う通信回数を表している．よって，n 個のプロバイダ全体の総通信回数 c は

$$c = \sum_{i=1}^{n} dom_i + \sum_{i=1}^{n} nei_i - n \tag{4}$$

となる．以上から，重複エリアの使用によって $\sum_{i=1}^{n} nei_i \cdot dom_i$ 回分の通信回数が削減できる．これは隣接エリアやドメイン数が多くなる程増大していくため，削減効果は非常に大きいと言える．

3.4　Kubernetes による複数資源への分散

異なるサーバ間での分散機能の実装には，モダンなコンテナオーケストレーションシステムである Kubernetes を使用している．Kubernetes は，Pod と呼ばれる仮想サーバを，物理または仮想サーバ群（クラスタ）上に適切に配置する仕様のため，サーバ上の配置戦略について考える必要がなくなる．

本シミュレータの全体構成は，図 7 のように，Simulator，Master，Worker，Gateway という四つの Pod からなる．SimulatorPod はエージェントの設置やエリアの指定，拡張，クロックの制御などの命令を出す．MasterPod は各 WorkerPod への命令の受け渡しや，WorkerPod の状態監視などの役目を担う．WorkerPod は担当するエリア毎に分割される．WorkerPod 内は Provider を管理する WorkerProvider，車や歩行者を管理する AgentProvider，可視化を行う VisProvider によって構成されており，MasterPod からの命令の下にエージェントの設置，サイクルの進退などを行う．また，GatewayPod はエリア間で必要なエージェント情報の共有を行う．このような構成により，エリアを拡張したり，ドメイン追加が容易になる．

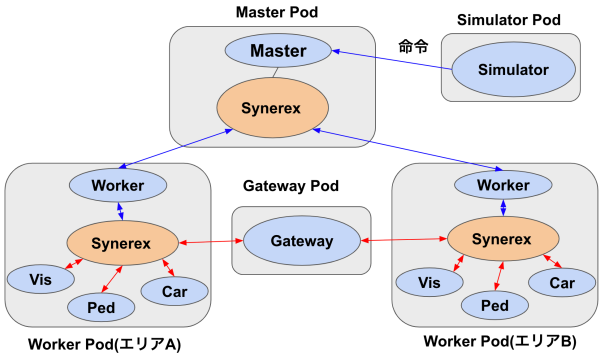

図 7　分散環境の構成図

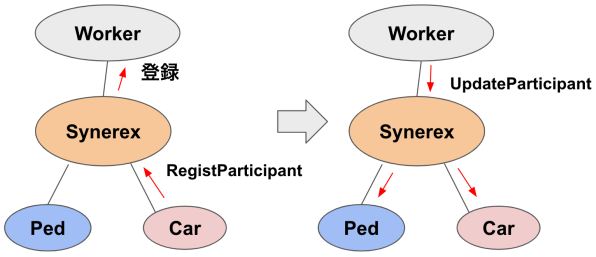

図 8　プロバイダの動的拡張における通信フロー

3.5　計算時間や通信時間に応じた自動的な処理分散機能

上述の Kubernetes は，Pod の適切な配置や死活監視は行うが，計算量や通信量の増加にともなう分割は，アプリケーション内部で判断する必要がある．本アーキテクチャでは，Worker プロバイダが WorkerPod 内の処理時間などのステータスを，定期的に MasterPod に送り，処理時間がある一定の閾値を超えた場合に，WorkerPod の分割や使用サーバの拡張を行う．この機能により，自動的な処理分散ができるため，開発者は分散を意識せずに実装できる．

3.6　参加者登録による動的拡張

エリアやエージェントの追加・削除などの動的拡張には，実行中のシミュレーションのサイクルに影響を与えないような工夫が必要である．本シミュレータでは，各プロバイダから，それらを管理する，管理 Provider(WorkerProvider や MasterProvider) にシミュレーションへの参加登録申請を行うことで実現している．具体的な処理は，図 8 のように，管理プロバイダへの登録要請をする「RegistParticipant」と，参加している全てのプロバイダへ参加者の更新を促す「UpdateParticipant」という二つの通信処理によって実現した．

3.7　複数言語への対応とオープンソース化

本シミュレータは Go 言語で記述されており，通信には gRPC を使用している．gRPC では ProtocolBuffer により API 仕様が定義されており，Javascript や Node，Java など様々な言語でサポートされている．そのため，開発者は

© 2020 Information Processing Society of Japan

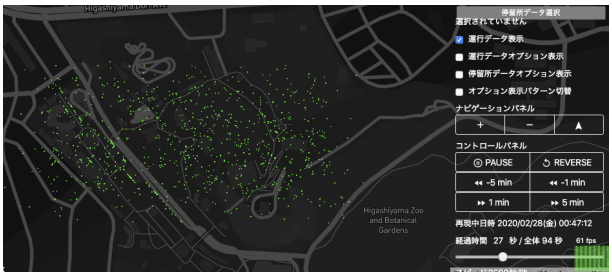

図 9　Harmoware-VIS よる人流シミュレーションの可視化

様々な言語でシミュレータの記述に対応することができる．また，本シミュレータはオープンソース化されており，GitHub 上で公開しているため誰でも利用可能である．

4.　現時点で実装されている機能

我々は，2019 年から本シミュレーション環境を開発しており，GitHub でオープンソースとして公開している．現時点では，第 3 章で述べたコンセプトの内，自動的な計算量分散とプロバイダの動的拡張以外の機能は実装済みである．これら機能は今後の実装課題となる．

また，人の回避行動アルゴリズムが実装された RVO2 ライブラリを用いてシミュレータを実装し，人流シミュレーションを模したサンプルプログラムを作成した．そのプログラムを用いれば，簡易的な人流シミュレーションを実行できる．また，Harmoware-VIS という可視化ライブラリを使用した GUI もあり，図 9 のようにシミュレーションの様子を確認できる．さらに，現状はエリアの動的な拡張機能は実装されてないものの，エージェントを動的に追加することは可能である．

本シミュレータは Kubernetes 上で起動でき，Docker，Kubernetes 環境が揃っていれば，理論上は，Linux，Windows などのサーバ構成を問わず動作可能である．Kubernetes はモダンなアーキテクチャであり，Google や Amazon，IBM など主要なベンダーによってサポートされているため，開発者の利用障壁は低くなると思われる．

5.　実験と評価

5.1　実験内容

第 3 章で考案したコンセプトを実装し，本シミュレータ環境における分散台数と性能の評価実験を行った．シミュレーションプログラムとして，人の回避行動のアルゴリズムである Optimal Reciprocal Collision Avoidance[15] を用いて，簡単な人流シミュレータを作成した．1 台から 4 台の仮想サーバに Kubernetes クラスタ環境を構築し，各サーバ数における設置可能なエージェント数の最大値を測定した．ここで，エージェントの密度は 1 人/m^2 とし，エージェントの増加に伴いエリアを拡大して，エージェント密度を一定にした．エージェント数の最大値の判定は，使用しているサーバの CPU 使用率が 100%となった時とした．また，使用する仮想サーバ 1 台につき，図 7 で説明した WorkerPod を 1 つ設置した．つまり，サーバ数 2 台目から Gateway プロバイダを介したエージェント情報の交換が行われる．今回の実験で使用した仮想サーバのスペックを表 1 に示す．

表 1　使用した仮想サーバの性能

使用した機材等	詳細
CPU 性能	Intel Xeon Silver 4214，2.20GHz
使用 CPU コア数	1Core
使用メモリ	2GB

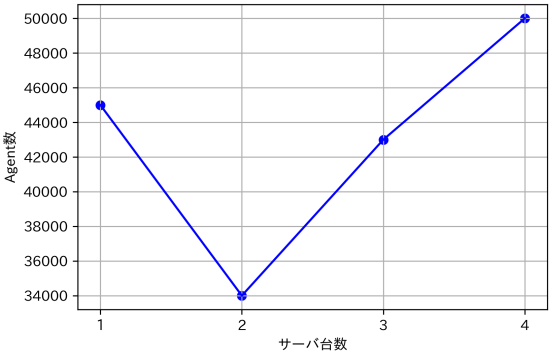

図 10　サーバ台数に対する Agent 数の推移

5.2　結果と考察

図 10 に，サーバ台数に対するエージェント数の推移の結果を示す．この結果から，サーバ台数が 2 台と 3 台のエージェント数を見ると，サーバ数 1 台の場合と比べて低くなっていることがわかる．これは，サーバ 1 台でシミュレーションを行う場合，Gateway を介した通信や，重複エリアにおける計算などのオーバーヘッドがないためだと考えられる．サーバ 2 台から 4 台までは台数が増えるにつれて，ほぼ線形にエージェント数が増えていることがわかる．また，サーバ台数 4 台使用時に，1 台の時のエージェント数を超えていることがわかる．今回の実験では 4 台のサーバを使用したが，今後，サーバー台数をさらに増やし，配置可能なエージェント数の推移を追加で実験する必要があると考える．

6.　おわりに

本論文では，大規模分散環境における人流・交通流シミュレータ環境の設計と評価を行った．Synerex を用いてプロバイダ毎にモジュールを分割し，データ交換基盤を通して，情報を交換・連携によりシミュレーションを行う，分散型シミュレータを実装した．このプロバイダ分割により，エージェントの計算処理にかかる負荷の分散や，計算時間の抑制が期待される．また，分散による通信回数の増加に対しては，重複エリアの実装で通信時間の大幅な抑制

© 2020 Information Processing Society of Japan

が可能となった．最後に，本シミュレータ環境における分散台数と性能の評価実験を行い，サーバ台数による性能変化を示した．

今後，エージェントの増加に伴う可視化時のデータ集中の問題の解決や，計算資源の自動的な分散機能などの実装を行っていく予定である．また，将来的には大規模化や動的拡張だけでなく，実世界のデータを用いたリアルタイムシミュレーションや異種シミュレータ間連携における課題にも取り組んでいく．

謝辞 本研究は，JST CREST JPMJCR1882，NICT 委託研究，総務省 SCOPE，JST OPERA(JPMJOP1612)，NEDO SIP2 期の支援を受けたものです

参考文献

[1] Ptv Viswalk - Pedestrian Simulation Software — PTV Group.

[2] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[3] Jason Tsai, Natalie Fridman, Emma Bowring, Matthew Brown, Shira Epstein, Gal Kaminka, Stacy Marsella, Andrew Ogden, Inbal Rika, Ankur Sheel, Matthew Taylor, Xuezhi Wang, Avishay Zilka, and Milind Tambe. ESCAPES - Evacuation Simulation with Children, Authorities, Parents, Emotions, and Social comparison. Vol. 1, pp. 457–464, 01 2011.

[4] Paul Richmond, Simon Coakley, and Daniela Romano. A High Performance Agent Based Modelling Framework on Graphics Card Hardware with CUDA. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 2, , 2009.

[5] 先山賢一, 芳賀博英. GPU によるマルチエージェント・シミュレーション用ライブラリ MasCL の設計と実装. 第 7 回社会システム部会研究会, 2014.

[6] 小藤哲彦, 竹内郁雄. 複数のシミュレータを統合するシミュレーションカーネル. マルチメディア通信と分散処理, 2001.

[7] 光行恵司. IT を活用した生産システム開発の効率化・迅速化 : 生産システムシミュレーションを用いたリスクアセスメントと分散型開発のための新たなシミュレーション環境. デンソーテクニカルレビュー, Vol. 9, No. 1, 2004.

[8] Muhammad Usman Awais and Peter Palensky et al. The High Level Architecture RTI as a master to the Functional Mock-up Interface components. *International Conference on Computing, Networking and Communications, Workshops Cyber Physical System*, 2013.

[9] 古市昌一, 和泉秀幸. 分散シミュレーションのための統合基盤アーキテクチャ HLA の紹介. 情報処理学会, Vol. 41, No. 12, 2003.

[10] 小堀壮彦, 山本一二三. 分散シミュレーション技術の紹介. MSS 技報, Vol. 21, , 2010.

[11] 河口信夫, 米澤拓郎, 廣井慧. Synerex: 超スマート社会を支える需給交換プラットフォームの設計コンセプトと機能. pp. 1–6, 03 2020.

[12] Synerex. https://github.com/synerex.

[13] Andras Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. pp. 1–10, 2008.

[14] 松倉龍之介, 村田嘉利, 鈴木彰真, 佐藤永欣. 自動車のリアルタイム位置管理方法の提案. 情報処理学会第 79 回全国大会, 2017.

[15] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body Collision Avoidance. *14th International Symposium on Robotics Research*, 2009.

© 2020 Information Processing Society of Japan