

# データフローモデルを基盤とした ドライビングシミュレータ構築環境の評価

青野 朝日<sup>1</sup> 佐藤 健哉<sup>1</sup>

**概要：**近年、複数の情報機器やセンサデバイスを組み合わせることにより、シミュレータ上で現実に近い体験を行うことが可能である。このような体感シミュレータの一つとしてドライビングシミュレータが挙げられる。従来は、シミュレータ上の車両モデルを入力機器によって制御するのみであったが、VR(Virtual Reality)機器の技術的進歩により、現実で運転している感覚を得ることが可能になりつつある。しかし、現実と同じ運転体験を行うためには多くのモジュールをシステムに組み込む必要がある。VR 機器、手の形状を認識するセンサデバイス、音響システム、送風機など現実に近づけるほどモジュール数の増加が見込まれるため、システムの複雑化、管理の難しさが問題点として挙げられる。また、自動で 3D マップを作成するソフトウェアがあるが、こうした処理負荷の大きいソフトウェアが次々に開発されているため、柔軟にこのようなソフトウェアとの連携を行う必要がある。そこで、本研究では、前述で挙げた問題点を解消するために、システム全体が細分化されたプロセス間をデータ通信することによって構築されるデータフローモデルを利用し、ドライビングシミュレータの実装を行った。また、既存手法との比較評価を行い、提案手法の優位性を示した。

## Evaluation of Driving Simulator Environment Based on Data Flow Model

ASAHI AONO<sup>1</sup> KENYA SATO<sup>1</sup>

### 1. はじめに

#### 1.1 背景

近年、複数の情報機器やセンサデバイスを組み合わせることで、シミュレータ上で現実に近い体験を行うことが可能である。VR(Virtual Reality)機器を主軸として、あらゆる機器を用いたシミュレータが企業フォーラムにおける製品デモやアミューズメントに活用されている。主な利用例として、ドライビングシミュレータが挙げられる。従来、自動車の展示場において、視覚的な情報と静止状態の車内を実際に触れて車両の情報を得ていた。しかし、VR 機器の技術的進歩により、前述した情報に加えて車両のハンドル、アクセル・ブレーキペダルを操作し、擬似的に動作状態の車両情報を得ることが可能になった。ここで、複数の機器を組み合わせたシミュレータを構築する際には以下の問題が発生する。

- ・ システム構成の複雑化

現実に近い体験、あらゆる場面を想定した柔軟なシミュレータを構築する際、追加するモジュールが増加するほどシステム全体の構成が複雑化[1]する。例えば、モジュールを組み込むほど指数関数的にソースコードの総量が増加する。また、システム構成の変更や新たに情報機器の追加を容易におこなうことができなくなる。さらに、マルチスレッドの処理をメンテナンスすることは高い技術力が求められる。

- ・ 処理負荷の増加

処理負荷の原因として、新たな機器をシミュレータに組み込むとその分処理量が増加する。また、Google 社が自動で 3D マップ[2]を作成するソフトウェアを開発するなど、今後大きな処理負荷が予想される。処理負荷の問題を解消するためには、分散処理環境へ遷移する必要がある。

<sup>1</sup> 同志社大学大学院理工学研究科情報工学専攻

そこで本研究では、システムの複雑性と処理負荷の増加を解消し、急速に進歩する技術に対して柔軟に対応可能なシミュレータ構築環境を考察する。通信方法にデータフローモデルを利用し、その有用性を評価する。

## 1.2 目的

本研究の目的は、データフローモデルが新たな情報機器、処理負荷の大きなソフトウェアに対して、柔軟に対応できる環境と想定し、実際に構築を行い、その有用性を評価することである。

## 2. 既存のシミュレータ

### 2.1 既存シミュレータの構築環境

シミュレータを構築する際、メインプロセスを主軸とし、システム全体のプロセスが密接に関わりあっている。図1に示すように、送信するのみのモジュールであっても送信先のプロセスと相互に接続されるため、組み込むモジュールが増加するほど複雑になる。また、メインプロセスが中核を担うため、どこか一箇所のモジュールがエラーを起こすだけでシステム全体が動作しなくなる。複雑化する構成の中、全体として高い管理能力が必要である。

### 2.2 既存シミュレータの問題点

#### ・ モジュール切り替え

シミュレーション環境において、システム構成は一意に定まらない。ドライビングシミュレータといえば、シミュレーションソフトウェアに入力装置の情報をを用いて複数のディスプレイに出力するものを想定するが、出力装置をVR機器に変更、ロボカーのような実機を操作するものもある。既存手法では、環境が変わるたびにデータの送信側、受信側のプロセス内容だけでなく、システム全体を変更する必要がある。技術進歩が目まぐるしい時代において、縛りのない柔軟なモジュール切り替えが要求される。

#### ・ ネットワーク分散処理

大規模なシミュレータは処理負荷が大きいので、複数台のコンピュータで処理する必要がある。しかし、各プロセスは独立でないため、ハードウェアの構成、メインソフトの変更を行う必要がある。そのため、実行プロセスの量が多くなるほど、変更箇所の増加、場所がわからなくなるといった開発者にとって負担が大きくなる。新たなソフトウェアを組み込む際、OSが対応していないなどの問題も挙げられる。

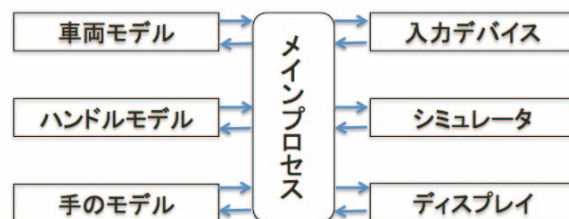


図1 既存シミュレータのシステム構成

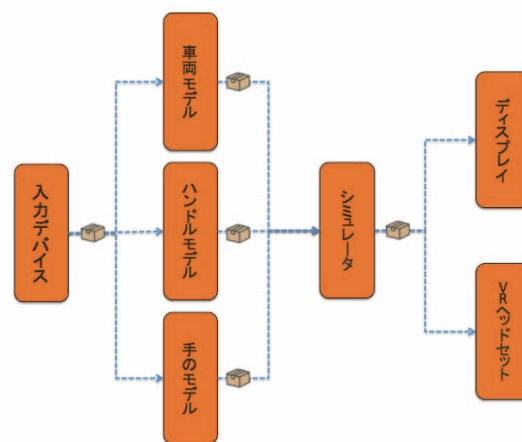


図2 データフローモデルを利用したシステム構成

## 3. 提案手法

### 3.1 概要

本研究では、ドライビングシミュレータ構築環境におけるプロセス間通信にデータフローモデルを適用し、実際にドライビングシミュレータの構築及び、その有用性を評価する。既存のシステム構築における問題点を解決するために、様々な機器を柔軟に組み込み、構成を自由に変更できる柔軟性を持った環境が必要である。データフローモデルを適用した環境で複数機器を接続したシステムの開発、テストを容易に行えるようにすることを目的とする。

### 3.2 データフローモデル

本研究におけるデータフローモデルは、システム全体を構成するプロセスを細分化し、独立したノードとする。そして、一つの計算機に複数のプロセスを立て、それぞれのプロセスがネットワーク経由でデータの受け渡しを行う。前提条件として、プロセスは全て独立し、単独で動作する。既存のシミュレータは、基本的にメインプロセスを実行することで動作する。一方、データフローモデルはメインとなるプロセスを持たない。

図2にデータフローモデルを利用したシステム構成を示す。

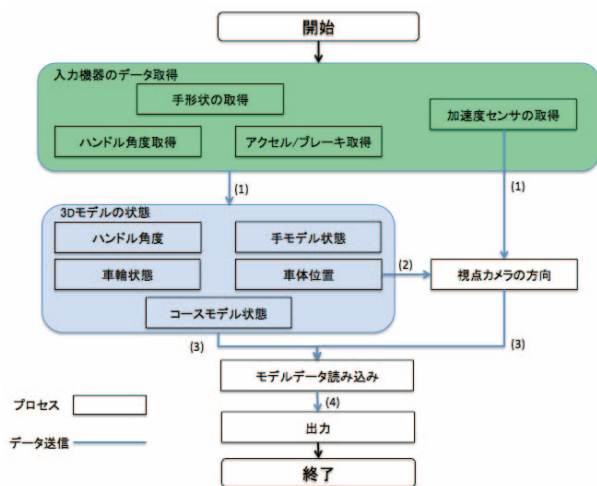


図3 システムの動作フロー

### 3.3 システム構成

本提案システムでは、以下の構成で行う。

- ・ シミュレータ

ハンドル、運転者の手形状、車両モデル、コースモデルを3Dモデルによって表現する。

- ・ 手形状、動作認識

VR デバイスを利用するため、自分の手を見ることができない。シミュレータ上で擬似的な手を作成するため、小型のセンサデバイスをVR デバイスに取り付ける。センサによって得られたデータをシミュレータとハンドルモデルを処理するプロセスへ送信する。

- ・ 入力情報取得

ハンドルとアクセル・ブレーキペダルを利用し、シミュレータ上の車両モデルを操作する。入力機器によって得られたデータは、車両モデルを処理するプロセスへ送信する。車両モデルは複数のノードで構成されているが、車輪ノードへ送信を行う。

- ・ ディスプレイ出力

シミュレータによって生成された3Dモデルを0.1秒間隔でディスプレイとVR デバイスに出力する。また、VR デバイ스에搭載された加速度センサのデータをシミュレータへ送信し、運転者の頭動作を表現する。

### 3.4 実行手順

図3にドライビングシミュレータの動作フローを示す。また、実行手順を以下に示す。

- (1) 入力機器から得られたデータをそれぞれのプロセスから対応したデータを送信
- (2) 視点カメラは車体に追従するため、車体位置のデータを受け取る。
- (3) 3Dモデルの状態を制御するプロセスは、必要なデータを受け取り、そのデータを適用したモデルデータ送信し、シミュレータが受信する。

表1 実装環境

| 構成要素    | 機種               | OS/Software             | Version                 |
|---------|------------------|-------------------------|-------------------------|
| サーバ     | ASUS X550V       | Ubuntu<br>ROS<br>Gazebo | 14.04<br>Indigo<br>6.00 |
| 入力装置    | Driving Force GT | -                       | -                       |
| 出力装置    | Oculus Rift      | -                       | Developer Kit 2         |
| センサデバイス | Leap Motion      | -                       | -                       |

表2 プログラム言語

| 構成要素       | プログラミング言語   |
|------------|-------------|
| サーバ(ROS)   | C++, python |
| シミュレータ     | C++, python |
| 3D モデル     | C++, python |
| 入力装置, 出力装置 | C++         |
| センサデバイス    | C++         |

- (4) 受信したモデルデータを元にドライビングシミュレータを生成し、VR 機器及びディスプレイへ出力する。

## 4. 実装

### 4.1 実装環境

本研究では、表1に示す環境でシステム構築を行う。データフローモデルを実現するため、サーバにミドルウェアであるROS(Robot Operating System) [3]を導入する。今回システム内で動作する全てのプロセスは、ROSで動作することを前提としている。また、実装に使用した言語は表2のようになり、複数プロセスが動作するROSとシミュレータ[4]に関しては、C++を基礎とし、車両モデルなどの3Dモデルを制御するプロセスはpythonで記述している。入出力機器といったモジュールのプロセスに関しては、C++で実装を行った。

### 4.2 利用技術

本研究が示すデータフローモデルを実現するためにROSのtopic通信方式を利用する。ROSは、OSRF(Open Source Robotics Foundation)が開発・メンテナンスしているロボット用OSである。ROS通信には返答を待たないtopicと返答を待つserviceの通信方法がある。この2つの通信方法は併用することが可能である。今回利用するtopic通信は、ノード間をトピックと呼ばれるデータを分類し、系統ごとに作成される論理チャネルのデータベースで接続し、発信側ノードがトピック上にメッセージを

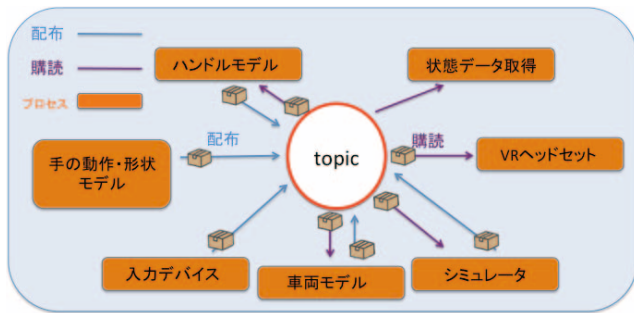


図4 topic 通信方式を利用したシステム構成

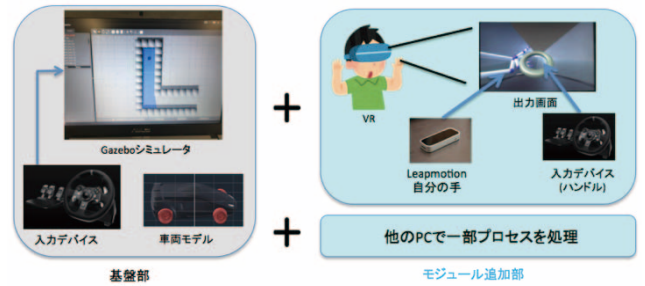


図5 実装システムのモジュール構成

発行し、受信側ノードが、このメッセージを受信することにより通信を行う。

利点として、それぞれのノードが独立しているため、ソフトウェアの再利用性が高く、異なる計算機上のノードとデータ通信が可能である。また、ノードごとに言語が違っていても、プログラムの内容が合っていれば通信を行うことが可能であり、ROS に搭載されているデータロギングツールを用いると、複数センサの時間の同期が可能である。さらに、運用時は部分ごとのソフトウェアのエラーがシステム全体の停止につながりにくい。イーサネットや無線LANなどのプライベートなネットワークやインターネットとの親和性が高く、ハードウェア的にも分散したシステムを構築するのに適している。図4にROSのtopic通信方式を利用したシステム構成を示す。

## 4.2 動作概要

図5は、実装システムにおけるモジュール構成である。また、図6は実際にドライビングシミュレータの動作風景を撮影した画像である。



図6 ドライビングシミュレータの動作風景

表3 システムの変更箇所比較

|                         | 既存手法            | 提案手法  |
|-------------------------|-----------------|-------|
| モジュール切り替え<br>(ディスプレイ出力) | メインプロセス<br>&出力先 | 出力先のみ |
| ネットワーク分散処理              | システム全体を<br>変更   | 変更なし  |

から発信されているデータを受け取るように指定するだけで自由に変更することが可能である。そのため、複数機器への同時出力も容易であった。

### ・ ネットワーク分散処理環境の構築

既存手法で一つの計算機上で複数の処理を行う場合、マルチスレッドのコードをメンテナンスする必要がある。また、複数の計算機での処理を実現しようとする、システム全体の構成変更を行わなければならない。一方、提案手法の場合、プロセスが独立で動作し、そのプロセス同士がインターネット通信をすることによってシステムを構成している。そのため、マルチスレッドのメンテナンスを行う必要がない。プロセスの一部を他の計算機で処理させる時は、その計算機同士をインターネット接続するだけでコードを一切変更することなく行うことができた。

## 6. 考察

評価結果より、プロセスを細分化してインターネットを経由したデータ通信を行うことで、システム構築がしやす

## 5. 評価

### 5.1 評価内容

本研究における評価は、モジュール切り替え、ネットワーク分散処理環境の構築、の2項目において既存手法と比較し、データフローモデルを用いることの優位性を調べる。表3に上記条件におけるシステム変更箇所の比較を示す。

### 5.1 既存手法との比較

#### ・ モジュール切り替え

モジュール切り替えに関して、ディスプレイ及びVR機器への出力処理を例に挙げる。既存手法では、出力先を異なる機器にする場合、メインプロセスを変更し、相互に接続する必要がある。現在、シミュレータの機能として、出力先の変更をUIによって行うことは可能であるが、同時に複数機器へ出力することは困難である。提案手法の場合、プロセス間をデータベースで繋いでいる。出力側のプロセスにおいて、シミュレータ

くなり、モジュール切り替えやネットワーク分散処理環境への遷移を既存手法に比べ、容易に行えることを示した。柔軟な適応能力を必要とするドライビングシミュレータにおいてはデータフローモデルを用いるべきである。結論として、本研究において、システム構築面からみた場合、提案手法の方に優位性がある。

しかし、今回調査できなかった点がある。それは、インターネット通信における遅延時間である。本研究においては、実装を行ったプロセスの数が少なく、他の計算機に遷移してもほとんど遅延時間は発生しなかった。大規模なシステムになるほど、プロセス数は増加し、通信帯域が圧迫されてしまい遅延時間の増大が懸念される。今後、構築面だけではなく、あらゆる面からデータフローモデルの評価を行う必要がある。

## 7. まとめ

本稿では、システムの複雑性と処理負荷の増加を解消し、急速に進歩する技術に柔軟に対応可能なシミュレータ構築環境を考察した。通信方法にデータフローモデルを利用し、その優位性の評価を行った。実装では、複数の機器を利用し VR 機器を主軸としたドライビングシミュレータを構築した。その過程において、モジュール切り替えとネットワーク分散環境開発に大きな優位性を得ることができた。

本研究において不十分であった、提案手法を用いた大規模システムにおけるデータの遅延時間に関して調査を行い、構築したドライビングシミュレータにロボカーなどの他のシミュレータとしての要素を組み込んでいく。

## 参考文献

- [1] 金刺宏樹, 鈴木豊太郎, 松岡聡. "並列分散システムにおける大規模交通シミュレーションの性能最適化." 研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2015-HPC-148 (28), pp.1-6, 2015.
- [2] 3D マップ自動生成. 入手先<<http://little-beans.net/review/google-3dmap/>>(参照 2017-4-29)
- [3] ROS. 入手先<<http://www.ros.org/>> (参照 2017-4-29)
- [4] Codd-Downey, R., et al. "From ROS to unity: Leveraging robot and virtual environment middleware for immersive teleoperation." Information and Automation (ICIA), 2014 IEEE International Conference on. IEEE, 2014.