車両制御システム向けデータ管理手法に関する研究

山田 真大

2015年1月

概要

近年，プリクラッシュセーフティ技術など，車両の状態や周辺状況を判断し，ドライバへの警告や自動制御により運転の支援を行う車両制御システムが登場している．たとえば，車両に搭載された複数のセンサからの情報に基づき，操舵回避の支援を行い，衝突が避けられない状況では介入ブレーキを作動させることで衝突衝撃を緩和し被害を軽減するシステムがある．また，衝突回避システム，車両追従システム，レーン逸脱警告システム，自動駐車システムなどもある．これらの現状の車両制御システムの構成は，複数の異種センサによりデータを取得し，1 つあるいは複数の電子制御ユニット（ECU）においてアプリケーションプログラムがデータ処理を行い，その結果，ステアリングやブレーキなどのアクチュエータの操作を行う．このように車両制御システムでは複数の ECU のデータを利用して動作を行い，また複数の車両制御システム間には共通利用するデータが存在する．たとえば，ステアリングセンサのデータは，衝突予防システムや車両追従システム，車線逸脱防止システムから共通で利用される．現在，車載ソフトウェアの規模と複雑性の増大に対処するため，車載ソフトウェアの開発手法によって，ソフトウエアプラットフォームを明確に定義し，ソフトウェアの共通化，汎用化と再利用性を高めてきた．しかし，複数の ECU を協調動作させて実現する車両制御システムは，ECU 全体を見据えた開発が必要であり，新たな車両制御システムが増えていくに従い，多様なセンサが多数搭載され，データの量と種類の増大によって開発コストが増加する．開発コストの増加を抑えるため，車両制御システムで扱う車載データをもとに論理的なデータ空間を定義し，要求に応じてア

プリケーションプログラムにデータ提供を行うことができるようにする必要がある．本研究では，論理的なデータ空間を構築するために，データ管理システムを導入することを提案する．

これまで地図データをデータベース管理システムで管理して車両制御システムが利用したという事例はあるものの，車両制御システムが主として利用するセンサデータのためにデータ管理システムを導入したという事例がなく，検証するために適用事例が必要である．また，データ管理システムの導入によってオーバヘッドの増加が見込まれるが，許容可能な遅延時間の短い車両制御システムにおいて，データ管理システムが利用可能であるかどうかといった点が不明である．そこで，本研究ではデータ管理システムの適用と実現可能性の確認の 2 点を研究課題として定め，取り組むことにした．

研究を進めるにあたり，データベース管理システムおよびデータストリーム管理システムの 2 種類の手法を適用した．初めの手法では，車両追従システム，自動駐車システムの 2 つの車両制御システムを用意し，それぞれのシステムのアプリケーションプログラムが共通のデータベース管理システムを利用して動作することを確認し，データ管理システムの適用を行った．また，実時間性の評価によって車両制御システムが許容可能なオーバヘッドの範囲内に収まることを評価し，実現可能性の確認を行うことができた．2 つ目の手法では，前方車強調カメラ映像表示システム，衝突警告システムの 2 つの車両制御システムを用意し，それぞれのアプリケーションプログラムが共通のデータストリーム管理システムを利用して動作することを確認し，データ管理システムの適用を行った．また，衝突警告システムの遅延時間の評価によって，時間制約を満たしていることを確認し，実現可能性の確認を行うことができた．

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

序論

1.1　研究の背景

近年，プリクラッシュセーフティ技術など，車両の状態や周辺状況を判断し，ドライバへの警告や自動制御により運転の支援を行う車両制御システムが登場している [1, 2]．たとえば，車両に搭載された複数のセンサからの情報に基づき，操舵回避の支援を行い，衝突が避けられない状況では介入ブレーキを作動させることで衝突衝撃を緩和し被害を軽減するシステムがある [3, 4]．また，衝突回避システム，車両追従システム，レーン逸脱警告システム，自動駐車システムなどもある [5]．車両制御システムにおいて，周辺の物体を検知するためにミリ波レーダやレーザレーダ，カメラを始め車輪速センサ，加速度センサ，位置検出センサ，ステアリングセンサなどの多様なセンサを複数搭載し，将来的には車車間通信や路車間通信の利用も加わって多くの種類のデータが利用される．

図 1.1 に，車両制御システムの現状のシステムの構成モデルの概略を示す．車両制御システムは複数の電子制御ユニット（以下 ECU）によって構成される．ECU にはセンサやアクチュエータが接続されており，ECU 上では車両制御システムの機能を実現するアプリケーションプログラムが動作している．アプリケーションプログラム

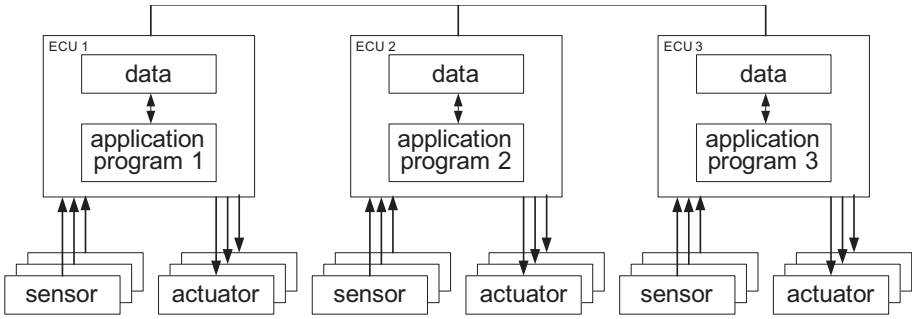

Figure 1.1: 現行システム構成モデル

は，アプリケーションプログラムが動作している ECU に接続されているセンサからのデータや他の ECU 上のアプリケーションプログラムから間接的にセンサデータを取得し，計算処理を行った上でアクチュエータの操作を行ったり，新たに外部の ECU 上のアプリケーションプログラムにデータを提供する．このように車両制御システムでは複数の ECU のデータを利用して動作を行い，また複数の車両制御システム間には共通利用するデータが存在する．たとえば，ステアリングセンサのデータは，衝突予防システムや車両追従システム，車線逸脱防止システムから共通で利用される．

1.2 研究課題

現在，車載ソフトウェアの規模と複雑性の増大に対処するため，AUTOSAR[6] に代表される車載ソフトウェアの開発手法によって，ソフトウエアプラットフォームを明確に定義し，ソフトウェアの共通化，汎用化と再利用性を高めてきた．しかし，複数の ECU を協調動作させて実現する車両制御システムは，ECU 全体を見据えた開発が必要であり，新たな車両制御システムが増えていくに従い，多様なセンサが多数搭載され，データの量と種類の増大によって開発コストが増加する [7]．このような状況において，データベース管理システムのようにデータを管理していない場合，アプリケーションとデータに強い依存関係が生まれ，その結果，データの不整合性，不統一性，冗長性が生じる [8]．同様に車両制御システムのソフトウェア開発で起きる問

題点について以下で説明する.

変更容易性

外部 ECU からどのようなデータが提供されるか明確な定義がなく，ある ECU に装着されているセンサのデータに依存して別の ECU でアプリケーションプログラムを開発した場合，ECU やセンサに変更が発生し，データの通信方法やデータフォーマットを変更し，アプリケーションプログラムがそのままデータを他の ECU へ配信した場合，他のアプリケーションプログラムが正しくデータを利用できない（不整合性）.

再利用性

以前に開発した ECU を，新規に開発している車両制御システムの一部として用いる場合，どのようなデータが必要であるかといった情報がなく，また新規に開発している車両制御システムにおいて，ECU 全体としてどのようなデータを提供できるのかといった情報がない（不統一性）と，その ECU を再利用することが難しい.

リソースの利用効率

外部の ECU が，どのようなデータが取得していて，どのような処理を施しているかという情報が分からない場合，外部の ECU にすでにあるセンサを重複して装着してしまったり，データを加工して別のデータに変換するといった処理が各 ECU が個別に行うことで，本来共通化できるような処理が ECU 間で重複してしまうといったことが発生 (冗長性）し，全体としてみると HW リソースや CPU リソースを無駄にしてしまうといった問題が発生する.

データ量と種類の増大による開発コストの増加を抑えるため，図 1.2 に示すように車両制御システムで扱う車載データをもとに論理的なデータ空間を定義し，要求に

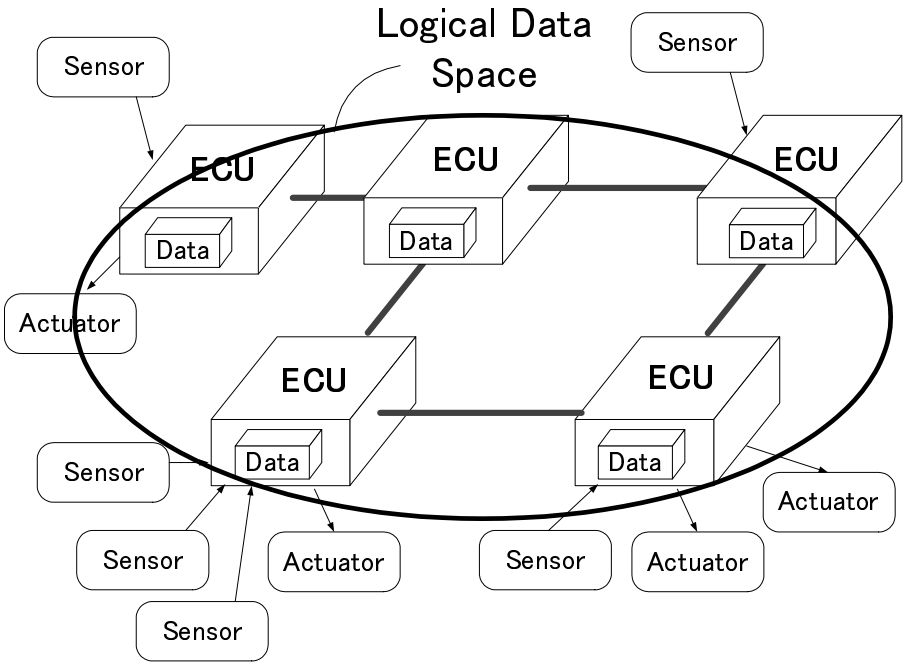

Figure 1.2: 車両制御システムの論理データ空間の構築

応じてアプリケーションプログラムにデータ提供を行うことができるようにする必要がある．ここで，論理的なデータ空間とは，各 ECU が持つデータを全体として 1 つの集合として保持していることをいい，データの定義が一意にされており，データがどの ECU に配置されているかということを意識せずに取得でき，データへアクセスするための方法が用意されていることと定義する．本研究では，論理的なデータ空間を構築するために，データ管理システムを導入することを提案する．データ管理システムとは，ネットワークを介して複数のアプリケーションプログラムが共通して利用可能であり，データの受け渡しを要求に応じて行う機能を有するものとして定義する．このデータ管理システムは，データアクセスのために統一したインタフェースを持ち，データフォーマットの定義機能，データ加工のための処理機能を提供する．統一したインタフェースとは，データの種類や ECU 間のネットワークの種類に依存することなく，定められた記述仕様に従ってデータへの問い合わせを記述することで，データの受け渡しが可能なインタフェースとする．このインタフェースによって，各

ECU のアプリケーションプログラムがそのインタフェースを通してデータの取得や提供を行うことができるようになり，またデータフォーマットをデータ管理システムに定義しておけば，アプリケーションとデータを独立させることができるので，変更容易性の低下を防ぐことができ，ECU やアプリケーションプログラムの再利用も容易になる．また，どのようなデータ加工処理が必要であるかということを，ECU 全体としてデータ管理システムが把握できるため，共通化できる処理の発見が容易となり，リソースの効率化につながる効果が期待できる．

しかし，これまで地図データをデータベース管理システムで管理して車両制御システムが利用したという事例はあるものの，車両制御システムが主として利用するセンサデータのためにデータ管理システムを導入したという事例がなく，検証するために適用事例が必要である．また，データ管理システムの導入によってオーバヘッドの増加が見込まれるが，許容可能な遅延時間の短い車両制御システムにおいて，データ管理システムが利用可能であるかどうかといった点が不明である．そこで，本研究では以下の 2 点を研究課題として定め，取り組むことにした．

- データ管理システムの適用　シミュレータ相当の環境で動作する車両制御システムに，データ管理システムを適用する
- 実現可能性の確認 シミュレータ相当の環境で車両制御システムがデータ管理システムを用いて動作できることを確認をする

1.3　アプローチ

データ管理システムの適用では，車両制御システムで取り扱うデータに着目して，車両制御システムおけるデータ管理システムの検討を行った．本研究を進めるあたり，最も普及しているデータベース管理システムの評価は，まず最初に取り組むべきであると考え，取り上げることとした．その次に，車両制御システムには，車両重量

などのほとんど更新されないデータと，車速などの高頻度に更新されるデータがあるため，データの有効期間や問い合わせの頻度に対して，データベース管理システムとは反対の前提に基づいて考えられたデータストリーム管理システムを取り上げるべきであると考えた．よって，以下の 2 種類のデータ管理システムを用いる手法 (データ管理手法）によってデータ管理システムの適用課題に取り組んだ．1 つはデータベース管理システムを用いる手法であり，データをリレーショナルデータのモデルとして扱う．この手法は，データへの問い合わせ頻度が低く，データは長期間にわたって有効に利用できるという前提で考えられたデータ管理手法である．この手法の具体的な内容，評価，考察については，3 章で述べる．もう 1 つはデータストリーム管理システムを用いる手法であり，データをデータストリームのモデルとして扱う．この手法は，データへの問い合わせ頻度が高く，データは短期間しか有効に利用できないという前提で考えられたデータ管理手法である．この手法の具体的な内容，評価，考察については，4 章で述べる．

実現可能性の確認とは，データ管理システムの適用によって車両制御システムがアプリケーションプログラム動作可能であることを確認する．具体的には，シミュレータ相当の環境で，2 つの車両制御システムのアプリケーションプログラムが同じデータ管理システムを利用して動作できることであり，また，2 つの車両制御システムのうち，一つは車両追従や衝突警告のような許容できる遅延時間の短い車両制御システムを選択し，その許容できる遅延時間以内に収まることを評価できることとする．前者は，本研究の目的にデータの共有があるため，最低限確認しなくてはならない動作環境である．後者は，データ管理システムのデメリットであるオーバヘッドが車両制御システムにとって許容できるかを確認するために必要である，

今回の研究では，シミュレータ相当における車両制御システムに対して，データ管理システムを適用することと実現可能性を確認することが目的であり，実際の車両制御システムへの導入を対象とはしていない．また，変更容易性，再利用性，リソースの利用効率の評価についても，実際の車両制御システムでの評価項目であるとして，

今回の研究の範囲外とする．

1.4 論文の構成

本論文の構成は以下の通りである．2 章において，データベース管理システムとデータストリーム管理システムについて紹介する．また車両制御システムの特徴やシステムで取り扱うデータについて述べる．3 章において，データベース管理システムを用いて車両制御システムを動作させた事例について紹介する．4 章において，データストリーム管理システムを用いて車両制御システムを動作させた事例について紹介する．5 章では，関連研究について述べ，6 章で，まとめと今後の課題について述べる．

CHAPTER 2

データ管理システムと車両制御システム

2.1　データベース管理システム

データベース管理システム（DBMS, Data Base Management System) とは，情報データの集積であるデータベースを維持管理するためのソフトウェアである [9]．データベース管理システムは，ユーザやアプリケーションに対して，データを定義，操作するためのインタフェースを提供している．ユーザやアプリケーションは，そのインタフェースを用いてデータの論理的な操作要求（クエリ）を発行する．クエリを受けつけたデータベース管理システムは，その論理的な操作要求を磁気ディスクなどの記憶媒体への物理的な操作に変換し，記憶媒体からデータの読み出しや記憶媒体への書き込みを行う．この構成により，ユーザやアプリケーションは DBMS を介してデータを共有でき，また，ユーザやアプリケーションは DBMS に対してデータへ論理的な操作要求を発行するだけで，記憶媒体の詳細を考慮する必要がないというメリットがある．

インタフェース

DBMS は論理的なデータを定義，操作するためのインタフェースを提供している．DBMS の多くはリレーショナル DBMS と呼ばれ，そのインタフェースは国際標準化機構 ISO によって Structured English Query Language(以下 SQL) として標準化されている．SQL によってデータベースの定義，登録，変更，削除，検索といった操作が可能であり，ユーザやアプリケーションによって使用することが可能となっている．

関係代数

関係代数とは，リレーショナル DB に対するデータ操作言語の基礎として用いられており，関係とよばれる集合に対する操作のことを言う．ある集合 $G_1, G_2, ... , G_n$ が与えられたとき，その直積集合 $S = G_1 \times G_2 \times \cdots \times G_n$ の部分集合のことを関係と呼び，各 G_i を関係 S の属性と呼ぶ．また，関係 S の各要素 $(v_1, v_2, ..., v_n), v_i \in G_i$ のことをタプルと呼ぶ．操作には，制限，射影，直積，和，差，積，結合，商の 8 つの基本操作からなる．関係に対し操作を行うことで得られた結果は，やはり関係となる．

制限

制限 (restriction, selection) は，関係から条件を満たすタプルを抜き出す操作である．

射影

射影 (projection) は，関係から指定された属性のみを抜き出す操作である．

直積

直積 (product) は，2 つ関係からタプルを取り出し，そのすべての組み合わせを求める操作である．

和

和 (union) は，属性の数と属性のドメインがすべて同じである 2 つの関係があった時に，それぞれの関係の和集合をとる操作である．

差

差 (difference) は，指定された関係から別の指定された関係に属するタプルを取り除く操作である．

積

積 (intersection) は，2 つの関係の共通となっているタプルをとる操作である．

結合

結合 (join) は，2 つの関係の直積をとり，その中から，それぞれの関係における属性や属性の間で与えられた条件を満たすタプルを抜き出す操作である．

商

積 (division) は，2 つの関係 R_1 と R_2 が与えられ，R_1 は R_2 との共通属性を持つとき，R_1 に属するタプルで，R_2 との共通属性値が R_2 のタプルとなっているものを集め，その中で固有属性値が同じものはグループ化し，各グループにおける共通属性値の集合が R_2 を含んでいる場合その固有属性値を取り出す操作である．

想定アプリケーション

データベース管理システムを利用するアプリケーションは，人事管理システム，在庫管理システム，販売管理システム，生産管理システムなどのビジネスアプリケーション，Computer aided design(CAD) や Computer Aided Software Engineering(CASE)

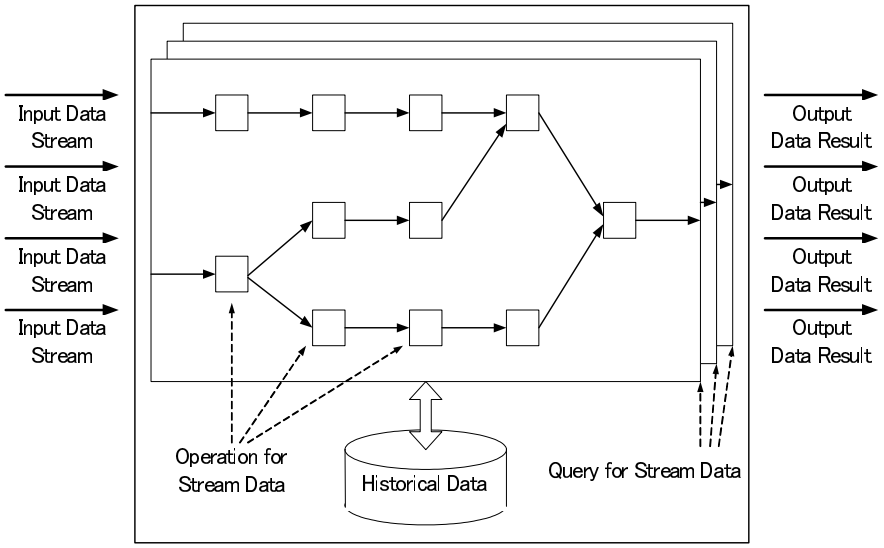

Figure 2.1: データストリーム管理システム概要

などのエンジニアリングデータの管理，化合物情報や遺伝子情報などの科学データの管理など利用方面は多岐にわたる [8].

2.2 データストリーム管理システム

データストリームとは，継続的に高頻度に到着し，その量は無限であるという想定のデータ項目のことを指し，タイムスタンプや一つ以上のデータ要素によって順序付けがされる．データ項目は要素と型によって定義付けされ，データストリームにおける要素は，リレーショナル DB におけるタプルと同等とみなされている．データストリーム管理システムとは，データストリームを入力として受け取り管理するための仕組みであり，データストリームに対する演算やクエリを提供するシステムを指す．

これらの仕組みを実現するものとして，TelegraphCQ[10]，Aurora[11] などがある．図 2.1 にデータストリーム管理システムの概要を示す．入力にストリームデータを受け，あらかじめ登録されているクエリ情報を元に逐次的に演算を行いながら，最終的に演算結果をストリームデータとして出力する．必要であれば，処理結果としての履

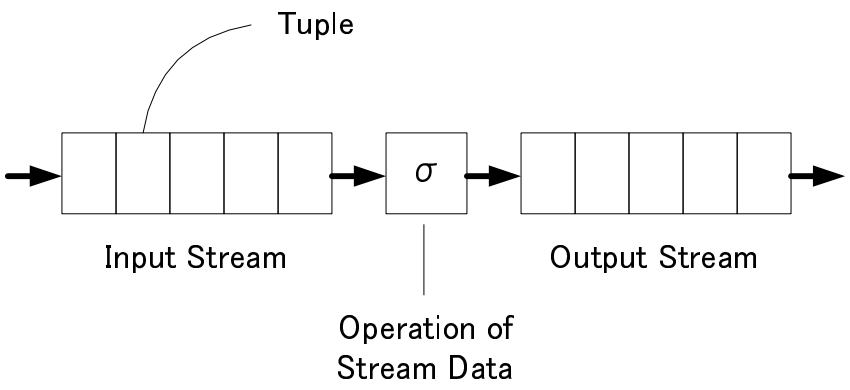

Figure 2.2: ストリーム演算

歴データを保持することも行う.

継続型クエリ

データストリーム管理システムにおけるクエリは，一般に継続型クエリと呼ばれる [12, 13]．継続型クエリは，どこからストリームデータを受け取り，どのようなストリーム演算を行い，どこへストリームデータを出力するかというクエリ情報を管理システムに一度登録することで，あとはそのクエリ情報に従いストリーム演算によりデータ処理とデータの送受信を継続する．そのためアプリケーションは要求のたびにクエリを発行する必要がなく，SQL のようにその都度クエリを出す場合と比較してオーバヘッドの削減が可能となる.

演算セット

データストリーム管理システムで扱うデータとその演算処理は，ストリームデータとストリーム演算としてモデル化される．図 2.2 にストリーム演算のモデルを示す．タプルとは同一のスキーマを持つ属性値の集合からなり，ストリームデータは時系列のタプルの集合で構成され，ストリーム演算が実施される．一般にストリーム演算は一つ以上のストリームデータを入力として受け取りストリームデータに対する演算を

行う．演算結果は再びストリームデータとして出力ストリームに流される．ストリーム演算は，Map, Aggregate，Join，Filter，Union, Drop などの要素で構成される [14].

MAP

Map は，入力ストリームのそれぞれのタプルに指定された変換を行い，その結果をストリームとして出力する演算である．変換を行うためにあらかじめ関数が用意されてり，演算のパラメータを使ってどの関数を用いるか指定することが可能である．関数には，sin, cos, tan, min, max, log, exp, sqrt, pow などが用意されている．また，これらの関数を四則演算によって組み合わせて指定することが可能である．

AGGREGATE

Aggregate は，入力ストリームに複数のタプルが到着した場合に複数のタプルを引数にとり指定された関数を実行し出力を行う演算である．たとえば，複数のタプルに対してその平均値を求めるといった際にこの演算が用いられる．

JOIN

Join は，2 つの入力ストリームを受け取り，入力ストリーム間で指定された条件を満たす場合のみ，一つのストリームに結合して出力する演算である．

FILTER

Filter は，条件式を設定し, 入力ストリームのそれぞれのタプルに対しその条件が成立する場合だけ, ストリームとして出力する演算である．

UNION

Union は，同一のスキーマを持つ 1 つ以上の入力ストリームを統合して，一つのストリームとして出力する演算である．

DROP

Drop は，計算負荷を低減するために特定の条件に従って入力ストリームのタプルを間引く演算である．

クエリ最適化

アプリケーションやユーザから登録されたクエリを解析し，演算の統合や共有化，演算の実行順序の変更を行うことで，その計算量を減らし，CPU やメモリといったリソースを節約する機能を保持する．

想定アプリケーション

データストリーム管理システムは，データの到着頻度が高く，データの到着をトリガとしてデータ処理を行い，その結果をアプリケーションやユーザに提供することを想定している．そのため，工場の異常監視を行うアプリケーションや，温度や湿度，天候などの環境測定センサのデータを読み取り反応するセンサネットワーク・アプリケーションや，株価や外国為替レートのモニタリングや売買を行う金融アプリケーションが，データストリーム管理システムを利用している．

2.3 車両制御システム

車両制御システムは，車両に複数のセンサやカメラ，通信機を搭載し，それらから得られるデータを組み合わせて解析し，車両周辺の状況を理解してドライバの補助や自動操縦を行うシステムである．現在これらのシステムは複数の ECU から構成され，ある ECU はセンサを利用し車両の情報を取得し，ある ECU はセンサから得られたデータを下に判断を行い，ある ECU はその判断情報を元にアクチュエータを通して操作を行う．このように，ECU は一つのセンサデータのみを利用するとは限らず，複数

のセンサデータが必要な場合には，そのセンサが接続された ECU とそのセンサデータを元に判断制御または操作制御を行う ECU 間を Controller Area Network(CAN)[15] などのネットワークと通して接続し，データ取得を行う必要がある．このようなシステムの間では，共通で利用されるデータもあればシステム固有で利用するデータもあるため，さまざまな種類のデータ形式を取り扱わなくてはならない．

以下に列挙するような，さまざまなシステムが登場している．

プリクラッシュ・セーフティ

レーダやカメラを用いて車両前方の障害物を監視し，障害物との距離と車両の速度や加速度を用いて衝突まで時間を算出し，衝突の危険があると判断した場合は，ブレーキやステアリングを自動制御したり，ドライバに警告を行うシステムである．

ドライバモニタ

ドライバをカメラを用いてドライバに異常がないか監視し，異常があったと判断できた場合は警告等を用いて注意喚起を促すシステムである．

レーダクルーズコントロール/アダプティブクルーズコントロール

レーダやカメラを用いて前方車両との距離を測定し，前方車両と一定の車間距離を保つよう車両を制御するシステムである．主に高速道路など直線が長い距離を走行するために用いられる．

レーンキーピング

カメラを用いて車両が走行している道路の白線を検出し，車両がその白線を逸脱しないようステアリングをコントロールするシステムである．

自動駐車支援

車両後方に取り付けたカメラを用いて，車両の駐車位置を推定し，アクセルやステアリングを制御して駐車位置まで車両を走行するシステムである．

また，実現はされていないが，以下のようなシステムが検討されている．

路車間通信を利用した車両制御システム

交差点や信号機などに取り付けた送信機から車両へ無線通信を行い，周辺情報の伝達を行う．カメラやレーダでは検出できない範囲にある情報を取得できるため，交差点付近における出会い頭の衝突を防ぐことが可能となる．

車車間通信を利用した車両制御システム

車両同士が無線を用いて相互に情報を伝達しながら，車両の制御を行うシステムである．カメラやレーダを利用して得られる周辺車両の情報は，推測によって算出された値であるが，車車間通信では，より正確な情報を伝達することができるため，より安全な車両の制御が可能となる．

取り扱うデータ

車両制御システムが取り扱うデータは大きく分類して，動的なデータと静的なデータの 2 種類が存在する．動的なデータは，車両走行中に時間に応じて変化するもので Global Positioning System(GPS) による位置情報，ミリ波レーダ，レーザレーダから得られる車間距離，CMOS カメラ，赤外線カメラ，ドライバモニタから得られる画像情報，車速，ステアリング角度，加速度センサからのデータ，車車間通信や路車間通信を通して得られる周辺車両や状況についてのデータなどがある．静的なデータは，車両走行中に変化しないもので，地図データや，サイズや重量など車両についての情報がある．動的なデータは，一般にデータサイズが小さく更新頻度が高いため，データ

ストリーム管理システムによる管理が適しており，逆に静的なデータは，データサイズが大きく更新頻度が低いため，データベース管理システムによる管理が適している．

CHAPTER 3

車両制御システムのためのセンサデータ統合管理方式の検討

3.1　概要

本章では，それぞれの車両制御システムにおけるアプリケーションプログラムの設計・開発を容易にする目的で，アプリケーションプログラムとセンサ部を分離し，センサから得られたデータを統合管理し，複数のアプリケーションプログラムから共通にデータを利用する方式の検討を行う．また，このセンサデータ統合管理方式の実現可能性を検討するために，プロトタイプシステムを構築し評価する．プロトタイプシステムには，データベース管理システムによるデータ管理手法によって，SQL インタフェースを用いたセンサデータへのアクセスを，アプリケーションプログラムが利用できるようにする．そして，センサデータ統合管理方式を実現したシステムの有効性を検証するため，二つの車両制御システムを実装し，シミュレータ上で評価を行う．

本章の構成について述べる．3.2 ではセンサデータを統合管理するための方式について述べる．3.3 では，システム設計について述べる．3.4 では実装について述べる．3.5 では評価結果を示す．3.6 では結論を述べる．

3.2 センサデータ統合管理

3.2.1 システム要求

本節では，車両制御システムのデータ統合管理に対する要求事項を実時間性，演算，トランザクションの観点から述べる．

3.2.1.1 実時間性

人が危険を検知してから行動を起こすまでの時間を空走時間 (あるいは反応時間) と呼び，その平均はおよそ 660ms と言われている [16]．今回，車両制御システムの中で時間制約が厳しい衝突の危険性に対し動作する運転支援システムにおいて，危険な事象が発生してから車両制御システムが危険だと判断するまでの間が空走時間以内であれば有効であると定義する．この時間を，危険な事象を示すデータが発生してから車両制御システムに到達するまでの遅延時間と考え，その内訳は次のようになる．

(1) センサデバイスによるデータ取得時間

(2) 取得したデータをデータベースへ渡す際にかかるデータ伝送時間

(3) データベースが受け取ったデータを挿入する処理時間

(4) 車両制御システムがデータベースへクエリを発行する際のクエリの送信時間

(5) データベースでクエリに対する必要なデータを処理する時間

(6) データベースから車両制御システムへ必要なデータを返す送信時間

(7) 受け取ったデータを使って危険かどうかの判断を行う処理

この中で，(1) は一般に 100ms 以内 [17] であることが求められ，この条件を満足させるためには (7) において同等の処理時間であることが求められる．したがって，本研究ではそれぞれの処理時間を 100ms として定義する．そのため，残りの時間 460ms 以内にデータの送受信とデータベース内でのデータ処理を完了することを実時間性の要求条件とする．

3.2.1.2 演算

一般にデータベース利用を行うアプリケーションがよく利用する選択・射影・結合・集約の演算は，車載制御システムでも必要となる．以下に車載制御システムでそれぞれの演算の利用例について述べる．

・選択，射影

基本的な問い合わせである選択，射影に関して，たとえば，Surround をセンサにより得られた周辺物体の認識データを格納したテーブルであるとした場合，そのテーブルの中から，現在時刻からさか上って t 時間前の物体 ID の位置座標のみを取得する SQL によって，車両の軌道計画の設定が可能となる．

```
Surround(物体 ID，位置座標，測定時刻)
```

```
SELECT Surround.位置座標 FROM Surround WHERE 測定時刻 ＞ ($現在時刻
-t$)
```

・結合

衝突危険性に対して動作する車両制御システムでは，センサで得られた認識物体の形状を調べるために，事前に定義された情報 (たとえば建物) との結びつけを行い，その認識物体に対して，自車の制御方法選択の判断を行う材料としてクエリ結果の利用が想定ができる．Surround テーブルはセンサにより認識した周辺物体の認識テーブルで，Buildings テーブルは地図データとして管理している建物に関するデータのテーブルとする．

```
Surround(物体 ID，位置座標，測定時刻)
```

```
Buildings(建物 ID，位置座標，形状 ID，住所)
```

```
SELECT * FROM Surround, Buildings
        WHERE Surround.位置座標 ＝ Buildings.位置座標
```

・集約

たとえば，道路の同一レーン上に存在する車両の台数を検出し，渋滞の程度を計算するために利用するということが想定できる．これらのテーブルの情報は，車両単体で得られるデータと車車間通信や路車間通信によって得られるデータを蓄積したものとする．

```
Surround(物体ID，位置座標，測定時刻，レーンID)
```

```
SELECT レーンID, SUM(物体ID) FROM Surround GROUP BY レーンID;
```

3.2.1.3　トランザクション

車両周辺の情報には，逐次変化するデータとほとんど変化が無い2つの種類のデータがある．障害物認識によってガードレールや木を認識した場合，その位置情報や特徴情報は比較的時間が経過しても変化することが無い．一方前方車を認識した場合，その情報は比較的短い時間で大きく変化する．変化の少ないデータは，保存しておくことで現在の測定値と比較したり，測定そのものを過去のデータで代用するといったことができる．このことから，車載制御システムで扱うデータを管理する場合において，永続的ストレージに保存し，その中で不要なデータは定期的に削除するといったトランザクションが必要になる．

3.2.2　システム構成

本研究で提案するセンサデータ統合管理方式を実現するためのシステム構成のモデル概略を図3.1に示す．複数のシステムにおいてそれぞれ管理されていたデータを

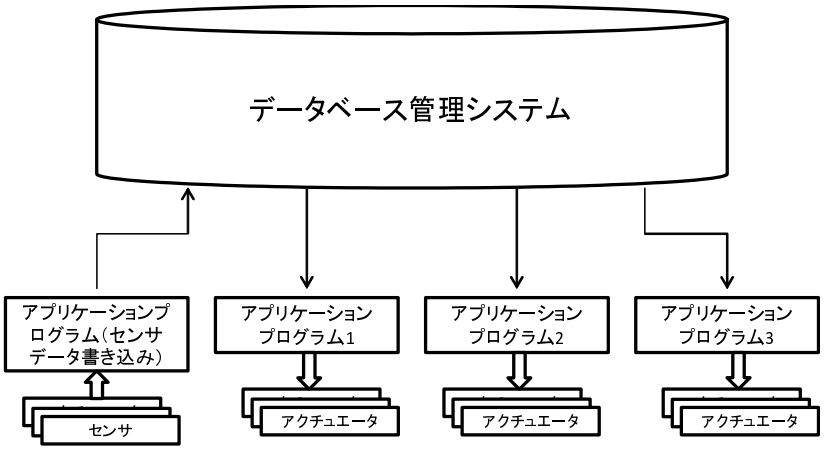

Figure 3.1: 提案システム構成のモデル概要

統合管理し，センサを切り離す構成を採る．この構成は，データの管理において一般的に取られる手法である [18]．車速/車輪速センサやステアリングセンサ，加速度/角速度センサなどから得られる車両の運動状況を示すセンサデータは，データベース管理システムにおいて定義されたスキーマに合わせてデータフォーマットを変換し，データベース管理システムへ書き込みを行うことで，複数のアプリケーションプログラムから共通に利用することが可能である．しかし，ミリ波/レーザレーダ，可視光/赤外線カメラ，超音波センサなどから得られる周辺の物体のデータは多様であり，センサ設置の有無や，センサそれぞれが検知できる範囲や精度も様々である．そこで，周辺の物体のデータを統一的に管理するための方式が必要となる．

3.2.3　データ管理方式

本研究においては，占有グリッド (occupancy grid)[19]，シーングラフ (scene graph)[20] を用い，車両の運動状況を示すセンサデータも含めたリレーショナルデータとして統合管理を行う．このデータモデルを図 3.2 に示す．

センサデータでは，車両位置や車両速度，それ以外の車両についての情報，車輪

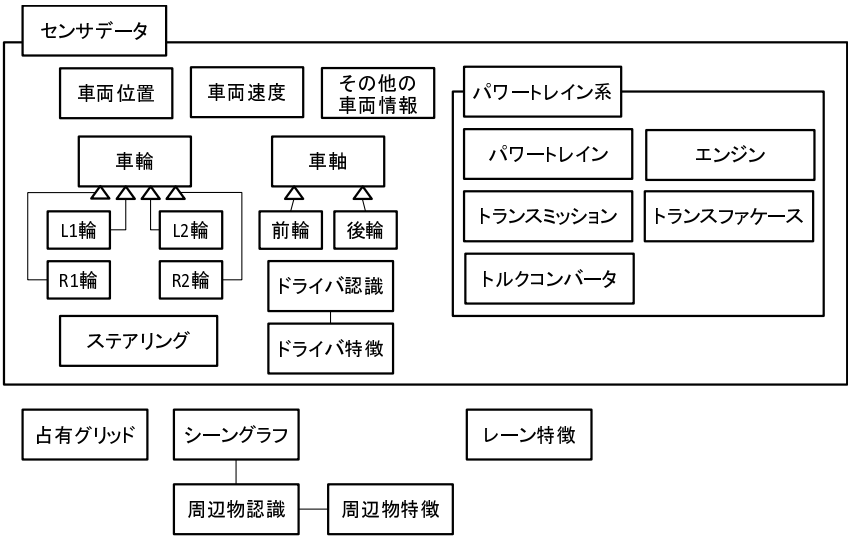

Figure 3.2: 3D360DB データモデル

や車軸，ステアリングについての情報，パワートレインについての情報，ドライバについて情報を定め，主に車両内で取得できるデータを管理するために用いる．占有グリッドは，車両周辺に存在する物体についての情報を管理するために用いる．一般的な占有グリッドでは，各グリッドに物体の存在する確率を割り当て，特定の位置に物体が存在するかどうかを判定するのみである．ここでは，ロボット分野で利用されている SLAM[21] の手法を取り入れ，センサの有無や，精度の違いを共通化しつつ，物体の形状，大きさ，向き，代表点の位置などの属性情報を付加し統合するためにシーングラフを利用する．シーングラフにより，占有グリッドの情報を補完することができ，周辺の物体間の相対位置関係によるツリー構造の構成が可能となり，自車両と他の車両や障害物との衝突判定や衝突予測を行うことできる．また，車両，建造物，歩行者などの物体に加え，レーンなどの情報も含めセンサから得られたデータの統合表現が可能となる．これらセンサデータの保持，および，保持されたデータの取得のために，SQL を利用したデータアクセスインタフェースを定義し，これらの機構をすべて含めて 3D360DB と呼ぶ.

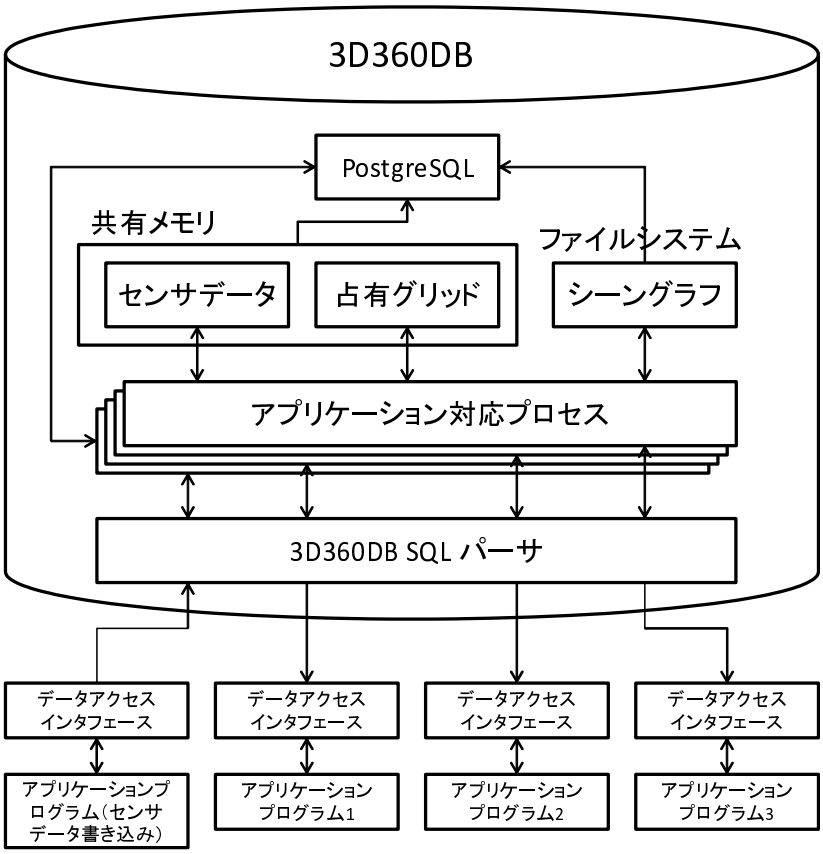

Figure 3.3: システム内部構成

3.3　システム設計

3.3.1　システム内部構成

図 3.3 に本システムの内部構成を示す．センサから取得し処理を行ったデータの書き込みは左に記したアプリケーションプログラムが実施し，その他のアプリケーションプログラムは，3D360DB からのデータの取得を行い，データ処理やアクチュエータ操作を行う．具体的には，各アプリケーションプログラムは，データアクセスインタフェースと呼ばれる 3D360DB へアクセスするための関数を利用して SQL コマンドを実行することで 3D360DB からのデータ送受信が可能になる．3D360DB は，アプリ

占有グリッド

x	-	位置 x
y	-	位置 y
z	-	位置 z
cell	-	セルの値

Figure 3.4: occupancygrid のスキーマ定義

ケーションプログラムから送信された SQL コマンドをパーサによって解析し，アプリケーションプログラムに対応するためのプロセス (アプリケーション対応プロセス) を生成して，それぞれのコマンドに応じてデータのアクセス先を変化させる．アクセス先は，SQL コマンドによるテーブルの指定に依存して，占有グリッド，シーングラフ，センサデータの 3 つに分かれる．また，履歴の保存のために，スナップショットを保存するためのプロセスが存在し，占有グリッド，シーングラフ，センサデータから定期的にデータを読み取り，PostgreSQL によってリレーショナルデータで保存するという作業を行う．

3.3.2 占有グリッド

占有グリッドを示すグリッドマップの各セルには，そのセルを占有する周辺物体の存在確率を保持する．本研究においては，マップサイズを 200m × 200m，セルサイズを 0.5m，セルの値は 0 から 255 の値として，利用する際に確率値に変換する．SQL による統一アクセス実現のため，SQL 文でテーブルに occupancygrid を指定することで，グリッドマップの値の参照・更新が可能となる．occupancygrid のデータスキーマは，図 3.4 のように定義した．左の列から順に，属性名，単位，意味を示している．

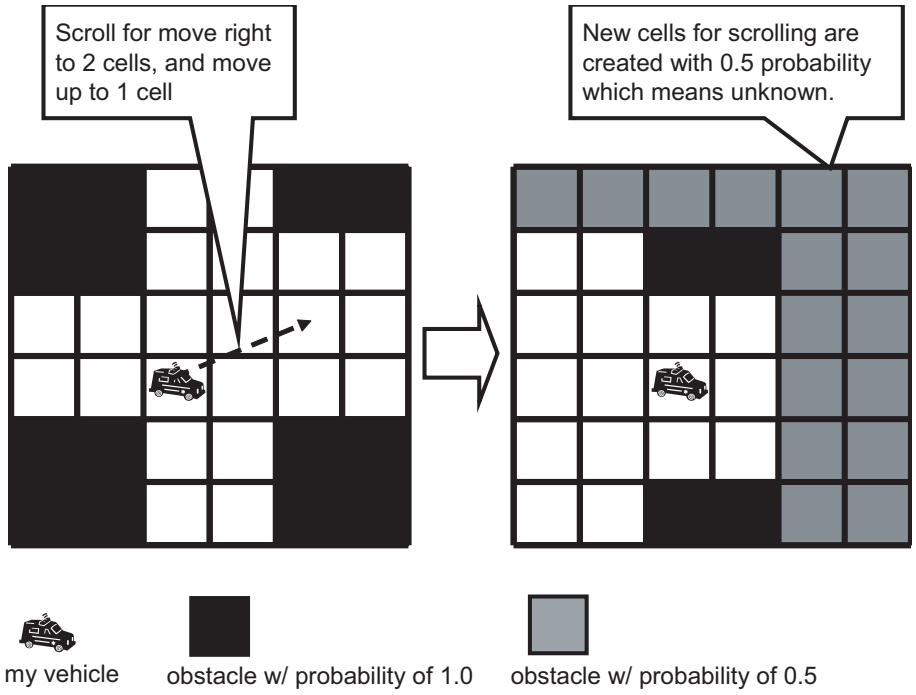

Figure 3.5: 占有グリッドのスクロール機能

・グリッドマップ上の位置 x、y での障害物の確率値の取得

select cell from occupancygrid where x=xidx and y=yidx;

xidx，yidx はグリッドマップ上の位置

グリッドマップにおける表現可能な領域には上限があり，車両周辺のすべての位置のグリッドマップを保有することができない．そこで，車両がグリッドマップ範囲外に移動した場合は，グリッドマップのスクロール機能により対応する．これは，図 3.5 のように，グリッドマップ範囲外に自車が移動した際などに利用することを想定し，スクロール命令を実行することでスクロール範囲外のセルは切り捨てを行い，新たに出現したセルは，物体があるかどうか不明なので 0.5 として値を初期化する．

また，占有グリッドは車両周辺の障害物に対して統一したデータ表現が可能であり，これを利用して自車の移動可能距離を車両周辺全体の情報から推定する機能も保持する．これは車両が特定の方向に対しどれだけ移動できるかという推定を行う機能

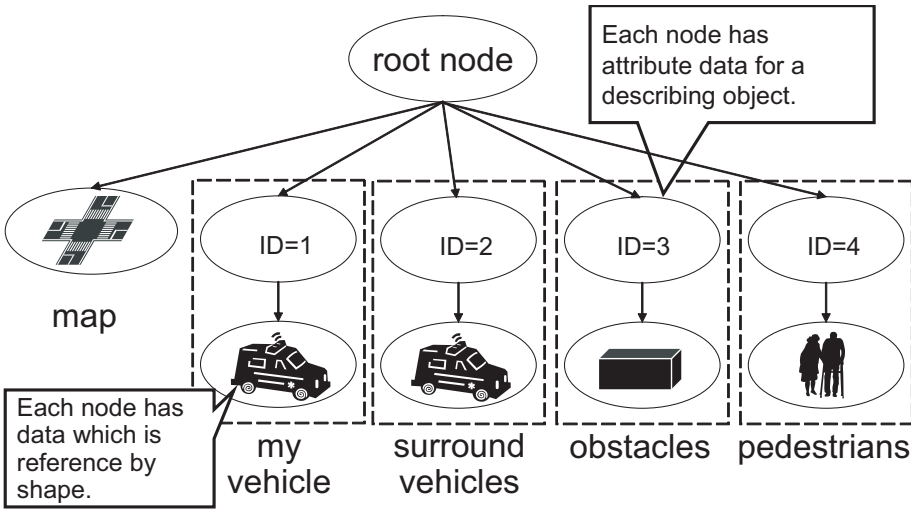

Figure 3.6: シーングラフのツリー構造

で，次の SQL 文を実行することで障害物までの距離を返すものである．

・占有グリッドによる移動可能距離推定

select shiftability(vx, vy, vz) from occupancygrid;

vx，vy，vz は自車の向き情報

3.3.3 シーングラフ

シーングラフは，図 3.6 に記述するように，root ノードを頂点としてその下に物体のデータを表現するノードが連なる構成を採る．それぞれのノードは物体の情報を持ち，その情報を 2 つのノード Transform ノードと Drawable ノードで表現する．Transform ノードは物体の位置，向き，大きさを持ち，Drawable ノードは物体の形状を持つ．

SQL による統一アクセスのため，SQL 文によりシーングラフにアクセスし，ノードに対して参照・更新を行うことができるように，ノードの作成，ノードの削除，ノードの移動，ノードの回転の 4 つの操作を実現できるように設計する．シーングラフにアクセスする際にはテーブル ID として scenegraph_node を指定する構成としている．シーングラフ（scenegraph_node）と周辺物認識，周辺物特徴（feature_surround）のス

シーングラフ

app_id	text	アプリケーション・インスタンスID
geode_id	int	オブジェクトID
nodefile_id	int	ノードファイル(オブジェクト3Dデータ)ID
width	m	横幅
height	m	高さ
position_x	-	位置x
position_y	-	位置y
position_z	-	位置z
rotate_x	deg	角度x
rotate_y	deg	角度y
rotate_z	deg	角度z
modified_datetime	datetime	データ作成・更新時刻

周辺物特徴

app_id	text	アプリケーション・インスタンスID
geode_id	int	周辺物ID
position_x	m	距離x方向
position_y	m	距離y方向
position_z	m	距離z方向
r_vx	km/h	相対速度x方向
r_vy	km/h	相対速度y方向
r_vz	km/h	相対速度z方向
width	m	幅
height	m	高さ
depth	m	奥行

周辺物認識

app_id	text	アプリケーション・インスタンスID
geode_id	int	周辺物ID
type	int	周辺物種別
is_forward_vehicle	int	前方車？
tn		車間時間
ttc	int	衝突時間
scenegraph_app_id	text	シーングラフ: アプリケーション・インスタンスID
scenegraph_geode_id	int	シーングラフ: オブジェクトID

Figure 3.7: シーングラフのスキーマ定義

キーマを，図 3.7 のように定義した．左の列から順に，属性名，単位，意味を示している．

・ノードの作成

insert into scenegraph_node (geode_id, nodefile_id, position_x, position_y, position_z, rotate_x, rotate_y, rotate_z, width, height, depth, modified_datetime)values;

・ノードの削除

delete from scenegraph_node where geode_id=’XXX’;

XXX:物体 ID

・ノードの回転・移動

update scenegraph_node set roll=YYY . . . where geode_id=’ZZZ’;

YYY:回転を行う角度, ZZZ:物体 ID

車両位置

Latitded	deg	緯度(WGS84) -90.000000～90.000000
longtitude	deg	経度(WGS84) -1800.000000～180.000000
altitude	m	高度 -10000～35000
position_rms	m	RMS値 0～100
roll	deg	ロール -180～180
pitch	deg	ピッチ -180～180
yaw	deg	ヨー -180～180
attitude_rms	deg	RMS値 -180～180

車両速度

vx	km/h	速度 x (前後方向)
vy	km/h	速度 y (横方向)
vz	km/h	速度 z (上下方向)
ax	g's	加速度 x
ay	g's	加速度 y
az	g's	加速度 z
avx	deg/s	角速度 ロール
avy	deg/s	角速度 ピッチ
avz	deg/s	角速度 ヨー
aax	rad/s2	角加速度 ロール
aay	rad/s2	角加速度 ピッチ
aaz	rad/s2	角加速度 ヨー
beta	deg	車両サイドスリップ角
betar	deg/s	車両サイドスリップ角速度

その他車両情報

driver_name	text	運転者名
destination	text	目的地名
route	text	経由地名
occupant	int	乗員人数

Figure 3.8: センサデータのスキーマ定義 1

3.3.4 センサデータ

センサデータは，共有メモリ上に，スキーマが定義された各テーブルごとにリングバッファを用意し，アプリケーションプログラムからのデータアクセスを，直接 PostgreSQL へアクセスしなくて済むようにして高速化する．SQL の insert 命令，update 命令，where 句なしの select 命令をアプリケーションプログラムが発行した場合，このリングバッファへアクセスし，where 句が指定された select 命令をアプリケーションプログラムが発行した場合，PostgreSQL に履歴を問い合わせに行く．リングバッファの値は定期的にスナップショットを保存するためのプロセスが読み込み PostgreSQL へ書き込みを行う．センサデータとレーン特徴のスキーマは，図 3.8，図 3.9，図 3.10 のように定義した．左の列から順に，属性名，単位，意味を示している．図 3.8 では，車両位置 (platform_globalpose)，車両速度（platform_velocity），その他車両情報について定義している．図 3.9 では，パワートレイン系のデータについて定義しており，パワートレイン，エンジン（platform_powertrain_engine)，トランスミッション，トランスファケース，トルクコンバータについて定義している．図 3.10 では，車輪，車軸，ステアリング，ドライバ認識，ドライバ特徴，レーン特徴（feature_lane）について定義している．

パワートレイン

m_d1_cl	N-m	フロントデフ-クラッチトルク
m_d1_cl2	N-m	フロントデフNo.2クラッチ-クラッチトルク
m_d1_visc	N-m	フロントデフ-ビスカストルク
m_d2_cl	N-m	リヤデフ-クラッチトルク
m_d2_cl2	N-m	リヤデフNo.2クラッチ-クラッチトルク
m_d2_visc	N-m	リヤデフ-ビスカストルク
m_l1_cl	N-m	L1輪クラッチトルク
m_l2_cl	N-m	L2輪クラッチトルク
m_r1_cl	N-m	R1輪クラッチトルク
m_r2_cl	N-m	R2輪クラッチトルク
my_dr_l1	N-m	L1輪ホイール駆動トルク
my_dr_l2	N-m	L2輪ホイール駆動トルク
my_dr_r1	N-m	R1輪ホイール駆動トルク
my_dr_r2	N-m	R2輪ホイール駆動トルク
pwrwheel	kW	ホイール入力パワー
thr_eng		正規化スロットル
throttle		正規化スロットル開度

エンジン

aa_eng	rad/s2	クランク軸角加速度
av_eng	rpm	クランク軸角速度
m_eng_in	N-m	クランク軸軸入力
m_eng_out	N-m	クランク軸軸出力
mfuel	kg	燃料消費量
pwrengav	kW	利用可能トルク
pwrengin	kW	クランク軸軸入力パワー
pwrengo	kW	クランク軸軸出力パワー
qfuel	-	燃料消費率
rot_eng	rev	クランク軸回転数

トランスミッション

av_trans	rpm	出力軸回転速度
m_trans	N-m	出力軸トルク
modetran	-	トランスミッション段
pwrtrans	kW	出力軸パワー
rgear_tr	-	ギヤ比
rottrans	rev	出力軸回転数

トランスファケース

av_d3f	rpm	フロント出力軸回転速度
av_d3r	rpm	リヤ出力軸回転速度
m_d3f	N-m	フロント出力軸トルク
m_d3r	N-m	リヤ出力軸トルク
rot_d3f	rev	フロント出力軸回転数
rot_d3r	rev	リヤ出力軸回転数
m_d3_cl	N-m	クラッチトルク
m_d3_cl2	N-m	No.2クラッチ-クラッチトルク
m_d3visc	N-m	ビスカストルク

トルクコンバータ

av_tc	rpm	出力軸回転速度
k_tc	Kinv	トルコン容量
m_tc	N-m	出力軸トルク
pwr_tc	rev	出力軸パワー
r_av_tc	-	出入速度比率
r_m_tc	-	出入トルク比率
rot_tc	rev	出力軸回転数

Figure 3.9: センサデータのスキーマ定義 2

3.3.5 アプリケーション対応プロセス

3D360DB では，センサデータ，占有グリッド，シーングラフが持つそれぞれのデータに対して SQL 形式でアクセスすることができるように，SQL を発行したアプリケーションプログラムに対応するプロセスを生成してデータアクセスを行う．このアプリケーション対応プロセスとは，3D360DB を利用するアプリケーションプログラムがデータアクセスインタフェースを利用して送信した SQL 命令を解釈した後に行う最初のプロセスである．これは，SQL による統一したデータアクセスを実現するためにあり，3 つの車載データの管理方式によるアクセス手段に依存することなく，車両制御システムの開発者は SQL コマンドの利用のみでデータの送受信の要求を行うことを可能にしている．今回の方式では，SQL 形式によりアクセスを統一しており，標準化された手法に基づく操作でき，機能的で使いやすいという利点がある．

車輪

alpha	deg	(横)スリップ角
alphl	deg	遅延スリップ角
camber	deg	キャンバー角
cmpt	mm	タイヤ圧縮長
dkapl	1/s	スリップ率
fx	N	タイヤ前後力
fy	N	タイヤ横力
fz	N	タイヤ垂直荷重
gamma	deg	タイヤ対路面傾き角
kappa	-	タイヤ前後スリップ比
mx	N-m	オーバーターニングモーメント
myrr	N-m	転がり抵抗トルク
mz	N-m	タイヤアライニングトルク
pitchg	deg	タイヤ前後方向接地角
rollg	deg	タイヤ横方向接地角
vxcen	km/h	前後速度
vytc	km/h	横方向速度
aay	rad/s2	車輪回転角加速度
avy	rpm	車輪回転角速度
rot	rev	車輪回転数
vx	km/h	等価車輪速度
mz_kp	N-m	ステアリングモーメント
steer	deg	ホイールステア角
strc	deg	コンプライアンスステア角
strk	deg	車輪幾何学的ステア角
cmpad	g's	ダンパー圧縮加速度
cmpas	g's	ばね圧縮加速度
cmpd	mm	ダンパー圧縮長さ
cmprd	mm/s	ダンパー圧縮速度
cmprs	mm/s	ばね圧縮速度
cmps	mm	ばね圧縮長さ
fd	N	ダンパー圧縮減衰力
fs	N	ばね圧縮力
fsext	N	ばね外力
jnc	mm	サスペンションストローク
my_bk	N-m	ブレーキトルク
pbkch	MPa	ホイールシリンダ圧
pbkd	MPa	ライン圧

車軸

f_boost	N	ステアリングラックアシストカ
m_boost	N-m	ステアリングギアアシストトルク
strswr	deg	操舵角/固定ギヤ比
mx	N-m	ロール剛性反力
rola	deg/s2	ロール角加速度
roll	deg	ロール角
fx	N	車軸前後力
fy	N	車軸横力
fz	N	車軸上下力

ステアリング

m_sw	N-m	ステアリングホイールトルク
m_tbar	N-m	トーションバートルク
steer_sw	deg	ステアリングホイール角
strr_sw	deg/s	操舵角速度
pbk_con	MPa	マスタシリンダ圧

ドライバ認識

is_awakening	-	覚醒中？
is_lookaway	-	よそ見？
is_wamble	-	ふらつき？
is_driving_posture	-	運転姿勢？
is_alcohol	-	アルコール？

ドライバ特徴

facial_angle	deg	顔角度

レーン特徴

lat_y	m	レーン中央からの位置偏差
yaw	deg	レーンに対するヨー角
pitch	deg	レーンに対するピッチ角
curvature	-	道路曲率
width	m	レーン幅
gradient	%	レーン勾配変化

Figure 3.10: センサデータとレーン特徴のスキーマ定義 3

3.3.6 車両制御システム設計

本研究において，センサデータ統合管理方式を実現したシステムの有効性を検証するため，車両追従システム [22] と自動駐車システム [23] の二つの車両制御システムを設計する．

車両追従システムは，車の前方向に自動車が存在した場合，その自動車と一定の車間距離を保ちつつ追従を行う制御を自動で行うシステムである．一定の車間距離を保つために，前方車との車間距離や自車との速度差を必要とし，そのデータから自車の目標速度を計算し，ブレーキやアクセルを操作する．

自動駐車システムは，自車が停止している状態において，目標とする停車位置と

停車方向が与えられた場合，その停車位置に合うようにステアリングとトルクを操作するシステムである．目標位置にたどり着くと停車を行う．また駐車している最中に，3D360DB の移動推定機能を利用して車両周辺の障害物と衝突しないかどうかを常にチェックしながら動作をさせている．

2 つの制御システムを検証に用意することで，車両追従システムのように自車両と前方車両が高速に動いているときに動作するシステムと，駐車支援システムのように自車両は低速で動き周辺の物体は静止している状況で動作するシステムと双方の検証を目的とした．それぞれ動作するシステムの状況は異なり，利用するデータも共通部分と非共通部分がある．これにより，新たに車両制御システムを追加で実装する際の指針とすることが可能となる．

3.4 実装

3.4.1 実装環境

本センサデータ統合管理方式を実現するための実装は，2 つの環境を利用して実施する．3D360DB は Linux 上で実装を行い，3D360DB を利用するアプリケーションプログラムは，シミュレータである CarSim[24] と MATLAB/Simulink[25] を使い実装を行う．CarSim は，多様な車両制御システムの制御ロジックを，実車両で評価を行う前に検証することが可能なシミュレーションソフトであり，複数の走行支援システムに利用されたという実績がある [26]．CarSim で再現された環境から得られる入力データは，そのデータの変数名，意味，単位，車両のどこの部位から取得できるかを定めており，シミュレーション実行中に逐次取得することが可能である．3D360DB とアプリケーションプログラムがデータのやり取りをする際には，MATLAB/Simulink 側から C 言語相当のプログラムの実行機能が必要になるが，MATLAB/Simulink の S-Function Builder の機能を用いることで実現する．また 3D360DB が管理するデータは，CarSim

から得られるデータを元にしている．

3.4.2 3D360DB

本システム内の占有グリッド部の実装は MRPT ライブラリ [27] を利用する．占有グリッドは車両周辺情報をグリッドマップという形式で共有メモリ上に保持する構成を採る．占有グリッドは，固定サイズの 1 枚のグリッドマップを共有メモリ上に作成し，複数のアプリケーションプログラムでデータ共有を行うことができるようになっている．そのため各アプリケーションプログラム間で排他制御が必要になる．今回，書込み用セマフォ，読出し用セマフォの 2 種類を用意し，書込み時には exclusive lock，読出し時には shard lock を利用することで排他制御を実現した．ロックの対象はグリッドマップ全体としている．シーングラフは，OpenSceneGraph ライブラリ [28] を利用して実装を行う．リレーショナルデータベースは PostgreSQL を利用する．リレーショナルデータベースでは，車両制御システムがデータに対してアクセスを行うためにスキーマ定義をあらかじめ行っており，必要なデータにアクセスする際には，3.3.4 項で述べたテーブル ID を指定する必要がある．SQL によるアクセスを統一化するために，3.3.2 項と 3.3.3 項で述べたように，占有グリッドやシーングラフについても同様にテーブル ID を用意し，SQL 文の中でテーブル ID を指定することでアクセス可能とする．

3.4.3 センサデータ入力モジュール

3D360DB において管理を行うセンサデータの入力は，図 3.11 に示したセンサデータ入力モジュールが行う．入力モジュールは，CarSim を通して取得したデータを 3D360DB へ入力する役割とし，センサから取得したデータを ECU で処理した後，3D360DB へデータを提供する機能のエミュレータとして動作する．提供するデータは，大きく分類して自車，前方車，レーン，障害物の 4 つになる．自車データはテー

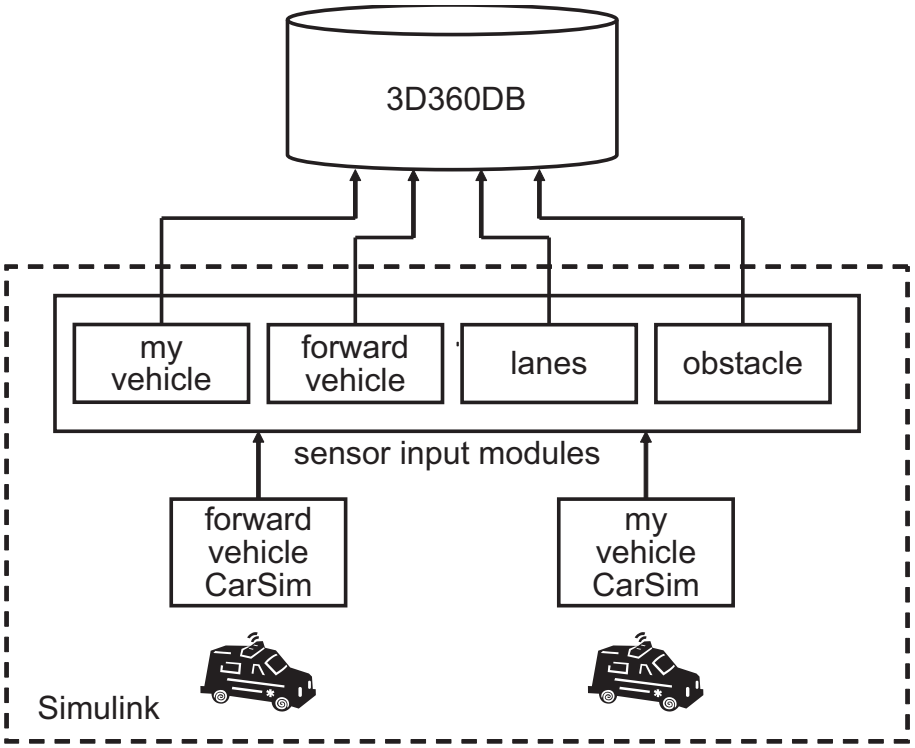

Figure 3.11: 入力モジュールのデータフロー

ブル ID として platform_globalpose，platform_velocity，platform_powertrain_engine を指定し，それぞれ自車位置，自車速度，エンジンの出力値の入力を行う．前方車データはテーブル ID として feature_surround を指定し，車間距離や速度の入力を行う．レーンはテーブル ID として feature_lane を指定し，自車との位置偏差，レーン曲率の入力を行う．

3.4.4 アプリケーション設計

車両追従システムのデータフローを図 3.12 に示す．車両追従システムを動作させるため，3D360DB は実行時，自車や周辺の情報を車載データとして保有し，提供する必要がある．今回の実装においては，CarSim をシミュレータとして利用し，CarSim から得られるデータをセンサから得られたデータと見立てて 3D360DB へ提供を行う ECU の役割を担うソフトウェアとして入力モジュールの実装する．

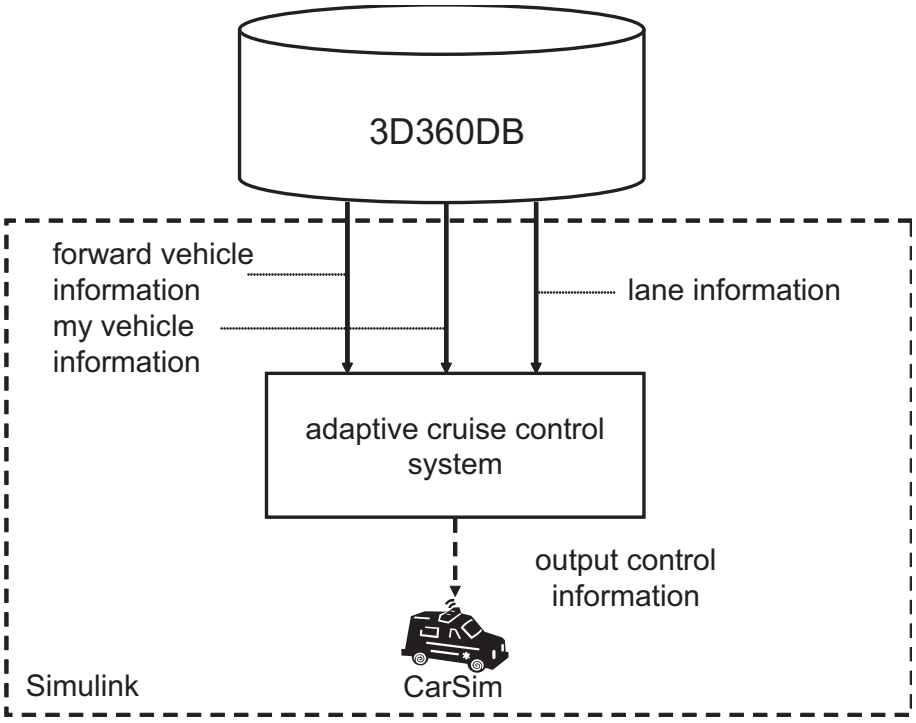

Figure 3.12: 車両追従システムのデータフロー

3D360DB より Simulink モデルで実現した車両追従システムがレーン情報，自車情報，前方車情報のデータを取得し，車間距離と自車速度を元に，制御すべき目標速度，ステアリング値の計算を行い制御情報を CarSim へ出力する．車両追従システムの表示例を図 3.13 に示す.

次に，自動駐車システムのデータフローを図 3.14 に示す．自動駐車システムにおいても，車両追従システムと同様に 3D360DB は実行時，自車や周辺の情報を車載データとして保有し，提供する必要がある．同じく，CarSim をシミュレータとして扱っており，CarSim から得られるデータをセンサから得られたデータと見立てて 3D360DB へ提供を行う ECU の役割を担うソフトウェアとして入力モジュールの実装を行う.

車両追従システムと同様，Simulink モデルで実現した自動駐車システムが，3D360DB から自車情報と障害物情報を取得し動作する．自動駐車システムは起動時，ドライバから駐車位置を指定され，自車位置と駐車位置を元に軌道を計算し，軌道に従い制御を行う．軌道上を移動する過程で，障害物情報を定期的に取得し，障害物と衝突の危

Figure 3.13: 車両追従システム

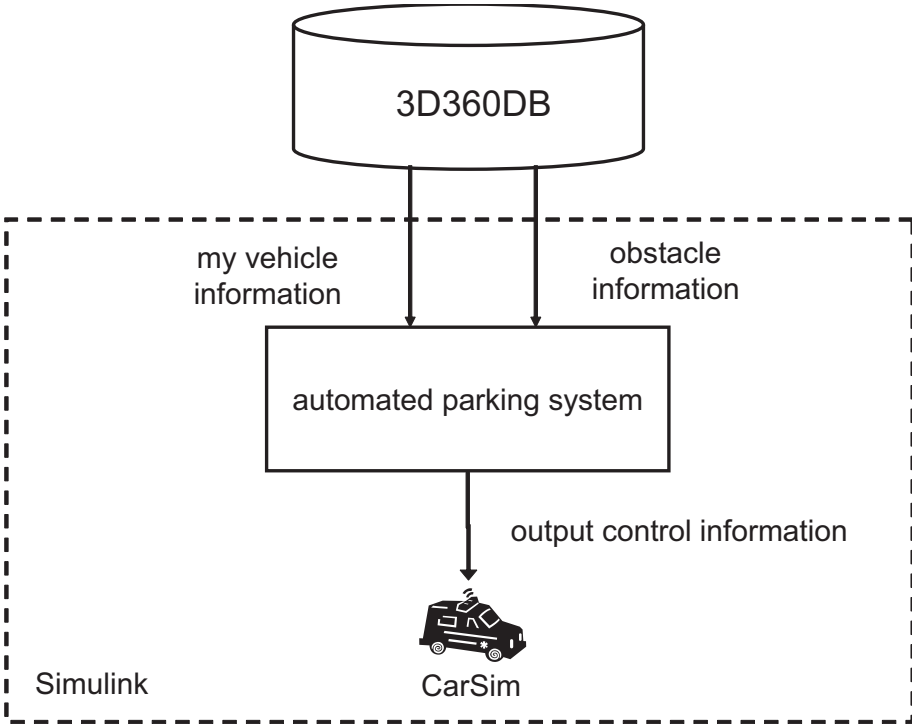

Figure 3.14: 自動駐車システムのデータフロー

険がある場合には停止を行う．自動駐車システムの表示例を図 3.15 に示す．

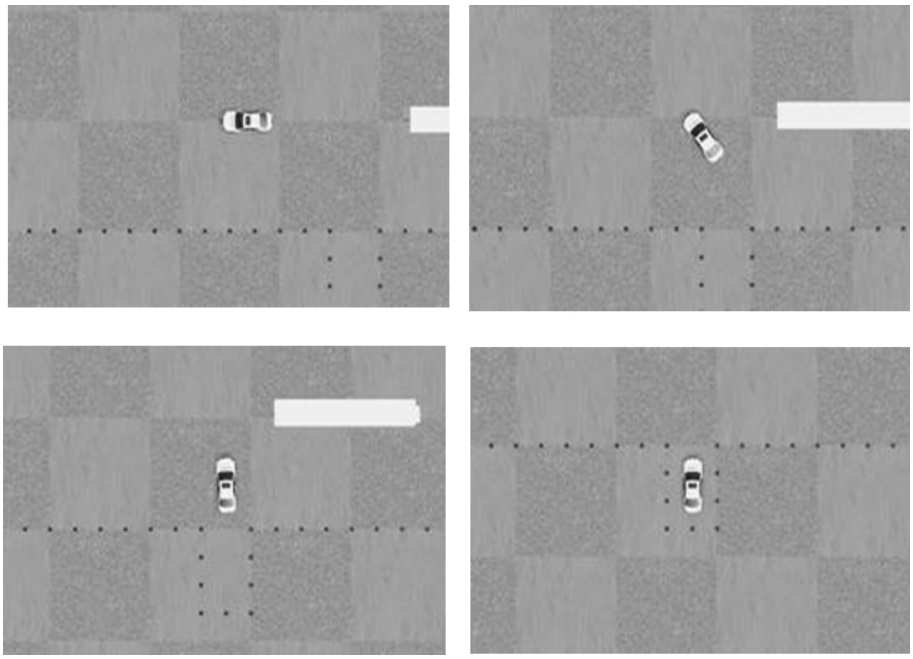

Figure 3.15: 自動駐車システム

3.5 評価

3.5.1 評価環境

性能評価は PC(CPU:PentiumD CoreDuo 2.8GHz, RAM:2GB, OS:Ubuntu 8.04) の環境で実施した．3D360DB と CarSim を利用して作成した車両制御システムの間で，データの送受信にかかったデータ量とクエリ実行の処理時間の測定を行った．データ量の測定は，車両制御システムが 3D360DB を利用する際のデータ量の見積もりを目的とし，クエリ実行の処理時間は，3D360DB を利用することで生じるオーバヘッドを調べることで実現性の検証を行うことが目的である．

3.5.2 車両追従システム

車両追従システムからのデータ取得要求により，3D360DB が SQL コマンドを実行する際の入出力のバイト数と実行時間の計測を行った．また，そのときの入力モ

Table 3.1: 車両追従システム実行時のデータサイズ

状況	アプリ	データ	実行数	平均入力バイト / 平均出力バイト
1	車両追従システム	前方車	122	92.50 bytes / 94.37 bytes
2	車両追従システム	自車	183	51.33 bytes / 87. 64 bytes
3	車両追従システム	レーン	61	46.00 bytes / 99.21 bytes
4	入力モジュール	前方車	61	127.77 bytes / 13. 00 bytes
5	入力モジュール	自車	183	208.81 bytes / 13.00 bytes
6	入力モジュール	レーン	61	75.21 bytes / 13.00 bytes

ジュールからのデータ更新要求で 3D360DB が SQL コマンドを実行した際の入出力のバイト数と実行時間の計測も実施した．それぞれの評価結果を表 3.1，図 3.16 に示す．

表 3.1 において，状況 1 は車両追従システムが前方車情報をデータとして取得する場合であり，この際の SQL の実行数は 122 回，平均入力バイトは 92. 50 バイトであり，平均出力バイトは 94.37 バイトである．状況 2 から状況 6 も同様である．図 3.16 は，表 3.1 の状況 1 から状況 6 に対応した実行時間の平均値，標準偏差，最大値を示す．

3.5.3 自動駐車システム

自動駐車システムからのデータ取得要求で，3D360DB が SQL コマンドを実行した際の入出力のバイト数と実行時間の計測を行った．また，そのときの入力モジュールからの書き込み要求で 3D360DB が SQL コマンドを実行した際の入出力のバイト数と実行時間の計測も行った．それぞれの評価結果を表 3.2 と図 3.17 に示す．

表 3.2 において，状況 1 は自動駐車システムが自車情報をデータとして取得する場合であり，この際の SQL の実行数は 7218 回，平均入力バイトは 50.00 バイトであり，平均出力バイトは 12.96 バイトである．状況 2 から状況 4 も同様である．図 3.17 は，表 3.2 の状況 1 から状況 4 に対応したクエリの実行時間の平均値，標準偏差，最

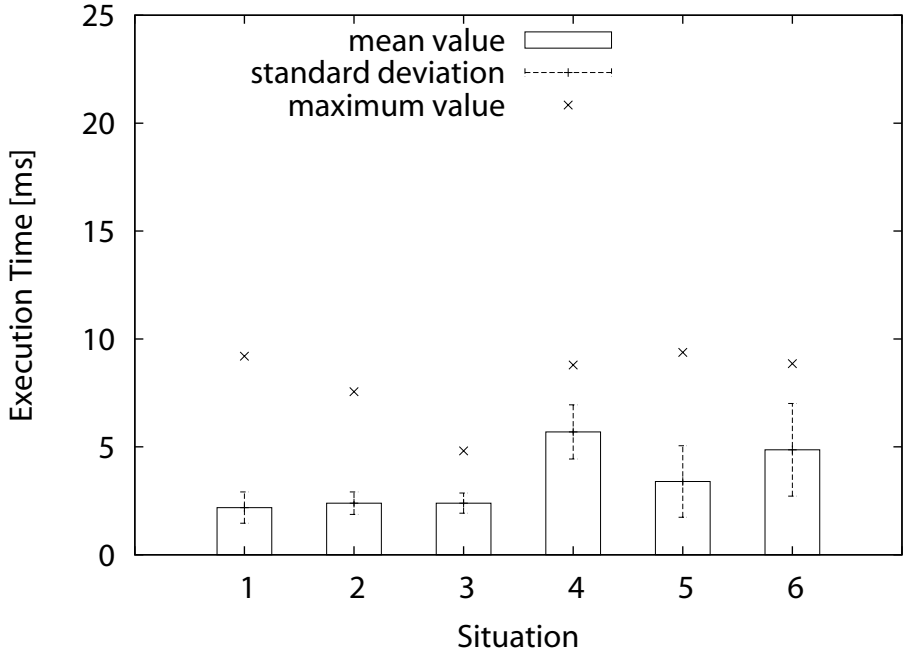

Figure 3.16: 車両追従システム実行時のクエリ実行時間

Table 3.2: 自動駐車システム実行時のデータサイズ

状況	アプリ	データ	実行数	平均入力バイト / 平均出力バイト
1	自動駐車システム	自車	7218	50.00 bytes / 112.96 bytes
2	自動駐車システム	障害物	2407	74.00 bytes / 64.00 bytes
3	入力モジュール	自車	7221	219.82 bytes / 13.00 bytes
4	入力モジュール	障害物	722	90.36 bytes / 36. 37 bytes

大値を示す.

3.5.4 3D360DB

3D360DB の占有グリッドの機能について，セル値読出し，セル値更新，セル値初期化，グリッドマップスクロール機能のそれぞれの実行時間と入出力のバイト数の測定を行った評価結果を表 3.3，図 3.18 に示す.

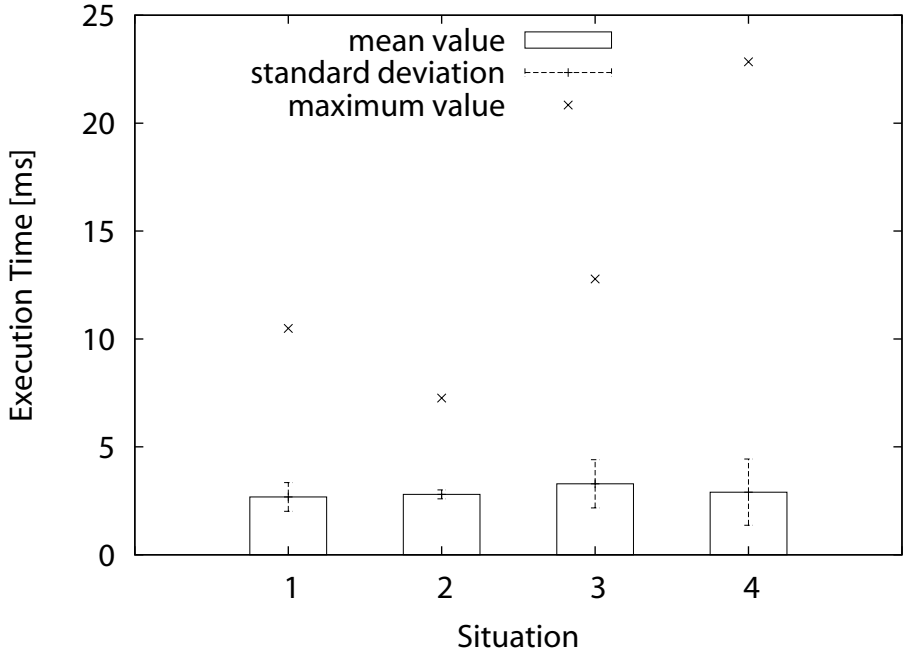

Figure 3.17: 自動駐車システム実行時のクエリ実行時間

Table 3.3: 占有グリッドのデータサイズ

状況	操作	実行数	平均入力バイト / 平均出力バイト
1	セル値読み出し	2500	57.00 bytes / 56.00 bytes
2	セル値更新	2500	60.00 bytes / 13.00 bytes
3	セル値初期化	2500	61.00 bytes / 13.00 bytes
4	スクロール機能実行	800	44.52 bytes / 5.00 bytes

表 3.3 において，状況 1 は占有グリッドマップからある特定のセルの値を読み出す場合であり，この際の SQL の実行数は 2500 回，平均入力バイトは 57.00 バイトであり，平均出力バイトは 56.00 バイトである．状況 2 から状況 4 も同様である．図 3.18 は，表 3.3 の状況 1 から状況 4 に対応したクエリの実行時間の平均値，標準偏差，最大値を示す.

3D360DB のシーングラフ機能について，ノード属性読出し，ノード移動・回転，ノード作成，ノード削除のそれぞれの実行時間と入出力のバイト数の測定を行った評

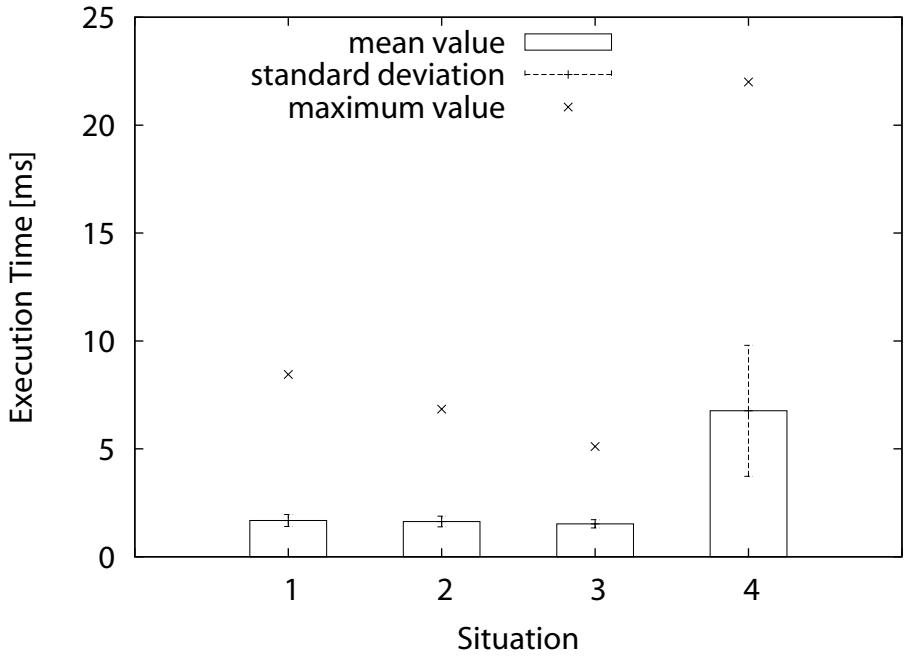

Figure 3.18: 占有グリッドの実行時間

Table 3.4: シーングラフのデータサイズ

状況	操作	実行数	平均入力バイト / 平均出力バイト
1	ノード属性値読み出し	1000	72.89 bytes / 138.00 bytes
2	ノード移動・回転	1000	73.89 bytes / 13.00 bytes
3	ノード作成	1000	182.89 bytes / 13.00 bytes
4	ノード削除	1000	50.89 bytes / 13.00 bytes

価結果を表 3.4，図 3.19 に示す.

表 3.4 において，状況 1 はシーングラフに存在しているノードからその属性の値を読み出す場合であり，この際の SQL の実行数は 1000 回，平均入力バイトは 72.89 バイトであり，平均出力バイトは 138.00 バイトである．状況 2 から状況 4 も同様である．図 3.19 は，表 3.4 の状況 1 から状況 4 に対応したクエリの実行時間の平均値，標準偏差，最大値を示す..

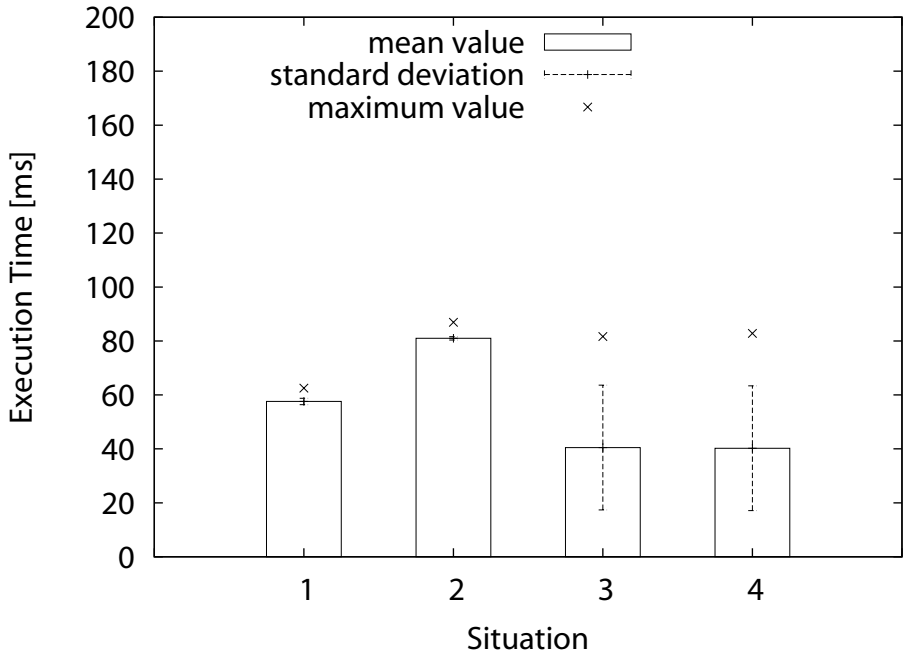

Figure 3.19: シーングラフの実行時間

3.5.5 ソースコード行数

MRPT ライブラリ，OpenSceneGraph ライブラリ，PostgreSQL を除く，3D360DB 本体のソースコードの行数は 9624 行，データアクセスインタフェースは 1466 行となった．

3.5.6 考察

車両追従システム実行時のクエリ実行時間の評価結果 (図 3.16) と自動駐車システム実行時のクエリ実行時間の評価結果 (図 3.17) では，実行時間の平均と最大の差が大きくなっている．この原因は 3D360DB において実行される SQL 命令を PostgreSQL へ処理依頼する際の，書き込み側と読み込み側のロック等によるデータベースに起因するものである．

占有グリッドの実行時間の結果 (図 3.18) では，全体的に実行時間の平均と最大の差が大きくなっている．これは占有グリッドに対する SQL 命令を受け取った 3D360DB

がその命令文をパースする際にまれに発生するメモリ不足によって発生する遅延である．またスクロール命令では，標準偏差が大きく，実行時間の平均と最大の差も大きくなっている．スクロール命令は現在のグリッドマップのデータの一部と新たにスクロールした領域とを重ね合わせて，新たなグリッドマップのデータを生成する．そのため，X 座標方面にスクロールするか Y 座標方面にスクロールするかで，現在のグリッドマップのデータ（配列）のうち残しておくデータの配置が異なるため，処理時間にばらつきが発生する．

シーングラフの実行時間の結果 (図 3.19) では，insert 処理と delete 処理の標準偏差が大きく，実行時間の平均と最大の差も大きくなっている．これは，プロセス間のシーングラフのデータ共有をファイルを介して行っており，insert 処理，delete 処理はともに最初にそのファイルをインポートし，変更が完了したらエクスポートを行っている．そのため insert や delete によりデータの書き込みや削除を行うことでファイルサイズが増減し，それに比例してファイルのインポート処理，エクスポート処理の時間の増減が発生している．

3.5.7 実時間性

今回の車両追従システムおよび自動駐車システムの評価結果が 3.2.1.1 節で定義した実時間性の要求条件に従うかの検討を行う．車両追従システムと自動駐車システムで扱っているそれぞれのデータ項目において，取得までの遅延時間を計算することで評価を行う．

データ取得の遅延時間は，3.2.1.1 節で定義した (2)〜(6) までの値の総和によって求める．ここで車両追従システムが前方車の情報を取得する場合を例にして，データの取得遅延時間を計算する．(2) から (6) までのそれぞれの値は，表 3.1 ，図 3.16 を使い，次のように計算できる．

(2)=状況 4 の平均入力バイト数の伝送時間

(3)=状況 4 のクエリ実行時間の最大値

(4)=状況 1 の平均入力バイト数の伝送時間

(5)=状況 1 のクエリ実行時間の最大値

(6)=状況 1 の平均出力バイト数の伝送時間

ここで (2)，(4)，(6) はネットワーク上でのデータ伝送時間の計算が必要になる．車載ネットワークとして利用される CAN はイベント駆動型の媒体アクセス制御方式のため，厳密に遅延時間を予測することができない．そのため，ここでは帯域の最大値 (1Mbps) の 10%を割り当てるとして遅延に対する許容時間の見積りを行う．また CAN でデータを伝送する場合，1 フレームあたり 8 バイトまでのデータしか送ることができず，8 バイトを越えるデータに対してはフレームに分割して送ることになるため，フレームごとの送信時間の総和が，送受信を行うデータの通信時間となる．また帯域幅 10%の見積もりで 1 フレームあたりの送信時間は 1.35ms となる [30].

(2) は，状況 4 の平均入力バイト数が 127.77 バイトであるため CAN では 16 フレーム送る必要があるので，その伝送時間は 16 × 1.35=21.6ms となる．(3) は図 3.16 を参照すると 8.80ms かかることがわかる．(4) は状況 1 の平均入力バイト数が 92.50 バイトであり，12 フレーム送る必要があるので，その伝送時間は 12 × 1.35=16.2ms となる．(5) は，は図 3.16 を参照すると 9.20ms かかることがわかる．(6) は状況 1 の平均出力バイト数が 94.37 バイトであり，12 フレーム送る必要があるので，その伝送時間は 12 × 1.35=16.2ms となる．よって (2) から (6) の総和が 72.00ms であり，前方車情報の取得遅延時間は 72.00ms かかることがわかる．

以下同様に，それぞれのデータ項目に対し，データ取得の遅延時間の計算を行いその結果を 3.5 に記述した．この結果により，それぞれのデータの取得による遅延時間は 460ms 以内に抑えられているため，本システムの検証において実時間性の要求条件を満たしている．

Table 3.5: データ取得遅延時間の評価

アプリ	データ項目	遅延時間
車両追従システム	前方車	72.00ms
車両追従システム	レーン	52.82ms
車両追従システム	自車	77.69ms
自動駐車システム	障害物	83.28ms
自動駐車システム	自車	90.77ms

3.5.8 応用性

今回の検証に使った車両追従システムと自動駐車システムについて考察し，3D360DBがどのようなアプリケーションプログラムに応用できるかについて検討する．車両追従システムの実行時に取得を行っているデータは，前方車，レーン，自車の 3 種類になる．自車を除くと 2 つの車両周辺の情報を扱っている．自動駐車システムは自車と障害物の 2 つのデータを利用しており，自車を除くと 1 つのみ車両周辺の情報を扱っていることになる．扱う車両周辺のデータが増えた場合，それぞれのデータに対し時間制約が必要であると考えると，データベースへの実時間性の要求もその数に比例して厳しくなると考えられる．たとえば，周辺車両，歩行者，障害物に対し自車が衝突するかどうかを判断し，危険であるなら警告を行うシステムであるならば，3 種類のデータそれぞれに時間制約が必要となる．このように考えた場合，今回の検証によって，2 種類の車両周辺のデータを扱う車両制御システムであれば応用ができると考える．

3.6 むすび

現在，複数の種類の車両制御システムが登場してきている．これらのシステムにおいては，センサデータを個別に管理し処理を行っている．そのため，異なるセンサを利用したり，新規のセンサを追加したりする場合は，アプリケーションプログラムを変更する必要が生じ，また，多種多様な複数システムにおいて，それぞれの設計・

開発を統合することが困難となる．本研究では，システムからセンサ部を切り離し，確率を利用した占有グリッドにおいてセンサから得られたデータをシーングラフ併用で統合管理し，複数のアプリケーションプログラムから共通に統合管理されたデータを利用する方式の検討を行った．この方式を利用したシステムの設計，実装を行い，シミュレータを利用し，車両追従システム，自動駐車システムの２種類の車両制御システムを構築し，評価を行い，許容可能なオーバヘッドで本システムの実現可能性を示すことができた．

今回は，センサデータ統合管理方式の検討が目的であるため，汎用 PC 上を利用し SQL データベースによりセンサデータを集中管理するデータ管理手法を採った．しかし，実際の車載における実装環境においては，各 ECU が分散してデータを管理する方式が現実的である．今後の課題として，分散データ管理およびデータ処理も含め，実際の組込み環境での設計検討を行っていく予定である．また，占有グリッドにおける複数の確率値の計算方法による精度検討も必要となると考えている．加えて今回の検討範囲では，3 種類以上の車両周辺のデータを扱う車両制御システム，たとえば 3 台以上の周辺車両を常に監視し衝突危険の警告を行うことができるシステムについては未検討であり，今後の課題としたい．

CHAPTER 4

データストリーム管理システムを利用した車載データ統合モデルの検討と評価

4.1 概要

本章では，車両制御システム間で共通に利用する論理的なデータ空間を表現するために，車載データ統合モデルを定義する．車載データ統合モデルとは，車両制御システムが利用するデータのデータ管理システムにおける表現を定義したものである．そして，車載データ統合モデルによってデータを管理しデータの提供を行う仕組みとしてデータストリーム管理システムを適用し，このデータ管理手法の実現可能性を評価する．実現可能性の評価には，車載データ統合モデルのデータを利用する 2 つの車両制御システムのアプリケーションプログラムを実装し，その要求に応じてデータ提供を行う際のオーバヘッドの計測を行った．

本章の構成について述べる．4.2 節では，対象とする車載データと車両制御システムについて述べ，車載データ統合モデルを定義する．4.3 節では，データストリーム管理システムを適用することを述べる．4.4 節では，プロトタイプの実装について述べる．4.5 節では，評価結果について述べる．4.6 節で結論と今後の課題について述べる．

4.2 車載データ統合モデル

4.2.1 取得情報

車両制御システムで利用するデータは大きく三つに分類される．自車両の状態をセンサによって測定することで得られる自車データ，車外周辺の前方車や歩行者などをカメラやレーダなどを利用して認識することで得られる車両周辺データ，またカーナビなどで利用される地図データやナンバープレートのような車両に関する情報などの静的なデータがある．自車データには，車速センサによる車速データ，ステアリングセンサによるステアリング角データ，ブレーキセンサによるブレーキ圧データ，GPS による自車の位置データなどが存在する．車両周辺データは，カメラ映像に対し認識処理を施すことで得られる前方車や歩行者，障害物など，自車に対する相対的な方向や位置データ，また，ミリ波やレーダを利用することで得られる自車との相対位置データや相対速度データが挙げられる．地図データのような静的なデータは GPS による自車位置データと組み合わせることで車両周辺の建物や道路の形状，標識などの推定に利用することができる．自車データと車両周辺データのセンサデータは時系列に生成されるため更新頻度が高く，逆に静的なデータはデータが変化することは少なく更新頻度は低いといったようなデータ特性に違いがある．また近年，複数のセンサデータを元に高精度に対象物を認識する技術が登場するなど，複数のデータからあるデータを導出するといったようなデータ間の関係が存在する [31].

4.2.2 車載データ統合モデルの設計

車載で管理するデータには，センサから直接得られる生データ，一つまたは複数のセンサから導き出すことによって得られる車外の認識物のデータや自車に関するデータ，さらに，統合的に衝突の危険性を判断する衝突予防システムにおいては，車両周辺全体のデータを必要とする場合もある．そこで，図 4.1 に示すようにデータを階層

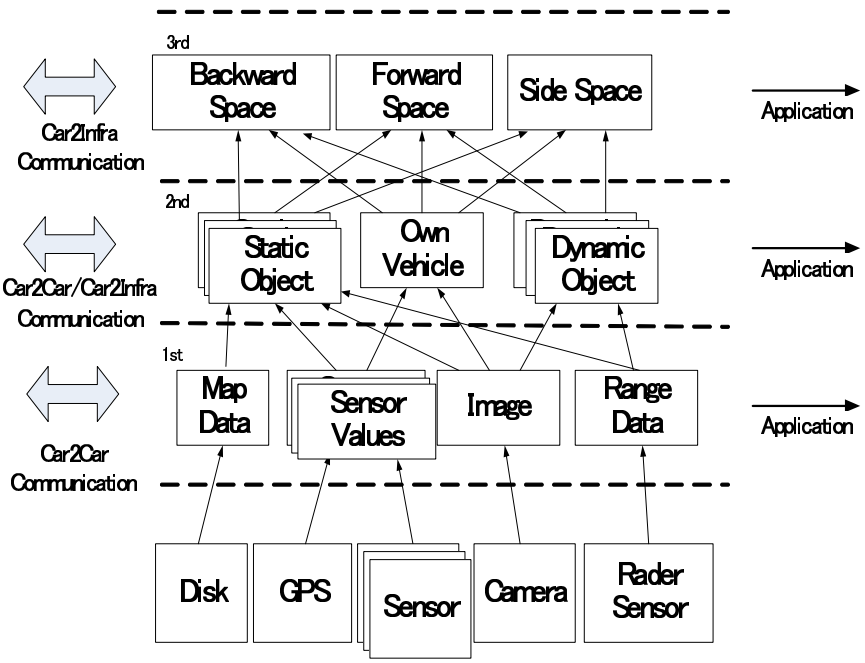

Figure 4.1: 車両制御システムのためのデータ統合モデル

的に管理し，利用する車両制御システムのアプリケーションプログラムに応じて異なる階層のデータを提供することができるように階層型の車載データ統合モデルを提案する．

車載データ統合モデルでは，データの階層は三つに分けて管理する．階層の下には，三階層に対してデータの提供を行うデバイスを配置する．デバイスには，カメラ，センサ，レーダ，GPS，ディスク装置がある．

これらのデバイスから取得した生データを第一階層 (1st) に配置する．たとえば，車速センサからは車速値，ステアリングセンサからはセンサ角，カメラからは映像データ，ディスク装置からは地図データなどが該当する．現在の車両統合制御システムは主にこの階層のデータを複数取得して車両制御を実行している．生データには少なくともセンサによってデータが得られた時刻を含めることとする．

第二階層 (2nd) では第一階層で得られたデータを元に導出できるオブジェクトを定義する．オブジェクトとは，前方車や障害物，歩行者，自車や標識やガードレール，建築物など認識可能な単位でデータを表現したもので，第一階層のデータから構築し，

少なくとも時刻と位置座標の情報を属性として持つこととする．オブジェクトは走行時に位置座標が動的に変化するオブジェクトと変化しない静的なオブジェクト，自車の三つに分類する．この階層からデータ取得を行うことで，第一階層の中から必要なデータを取得するよりも，より意味のあるまとまりでデータが集約されているため，取得する側にとっては要求するデータの数が第一階層と比べて少なくてすむ．車両追従システムのような前方車の情報の利用が必要な場合にこの階層のデータを利用することが想定している．

第三階層 (3rd) では第二階層のオブジェクトごとのデータの位置関係を元に，自車に対する周辺データを構築する．データは前方，後方，側方の単位で分類し，その位置に該当する第二階層のデータの時刻を元に集約する．集約とは，時刻の値 t が等しいデータ (t,A) とデータ (t,B) が第二階層にあった場合、(t,A, B) としてデータを生成することをいう．たとえば，衝突の危険を検知するようなシステムの場合，車両が前方向に走行中は，第三階層の前方空間のデータを取得することで，前方向に存在する車両や障害物の情報を一括して取得でき，衝突の危険性がないかどうか判断することに利用できる．この場合も第二階層の前方車や障害物など個々のオブジェクトを表すデータを取得するよりも，関心のあるデータが位置関係により集約されているため，取得するデータ数を減らし効率よくデータ取得が可能になる．

第二階層や第三階層のデータ取得には，オーバヘッドの増加という課題が発生する．オーバヘッドとは，データが入力されてからアプリケーションプログラムがデータを受け取る間に，各階層においてデータの通信にかかる時間，他のデータと同期を取るために待ち合わせる時間，データの処理にかかる時間を全て加算したものとする．

4.3 データストリーム管理システム

車載データ統合モデルの実現を行うには，2.3 節で述べたように，車両制御システムからの動的データの取得方法が課題になる．たとえば，リレーショナルデータベー

スでは，データをスキーマ定義し SQL というインタフェースを通して，データの更新や取得を行う．この場合，一つ一つの SQL の実行毎にデータを取得するため，センサのような時系列に生成されるデータをすべて取得するには効率が悪い．そこで本研究では，2.2 節で紹介した継続型クエリを持ったデータストリーム管理システムによるデータ管理手法を提案する．

4.4 実装

4.4.1 環境

本論文では，階層化によるオーバヘッドの計測を目的として実装を行った．シミュレーション環境として PC/UNIX を使い，入力に使うセンサデータはあらかじめ測定済みでディスクに保存されているデータを用いた．

4.4.2 データストリーム管理システム BOREALIS の適用

本研究では，ブランダイス大学，ブラウン大学，マサチューセッツ工科大学で開発された分散型データストリーム管理システムを実現した Borealis[14] を利用して実装を行った．Borealis はソースコードが公開されており，Linux 上で動作し，アプリケーションプログラムは C++言語で記述する．Borealis の利用により，ノードと呼ばれる実行単位を複数の PC 上に配置することで，分散環境のデータの統合管理を行うことができ，現在複数の ECU に分散しているセンサデータの取り扱いが容易になる．

4.4.3 アプリケーションプログラム実装

階層化したデータを利用するアプリケーションプログラムとして次の 2 つを設計・実装した．

(1) Warning System(衝突警告システム)　前方空間のデータを取得し，衝突の危険

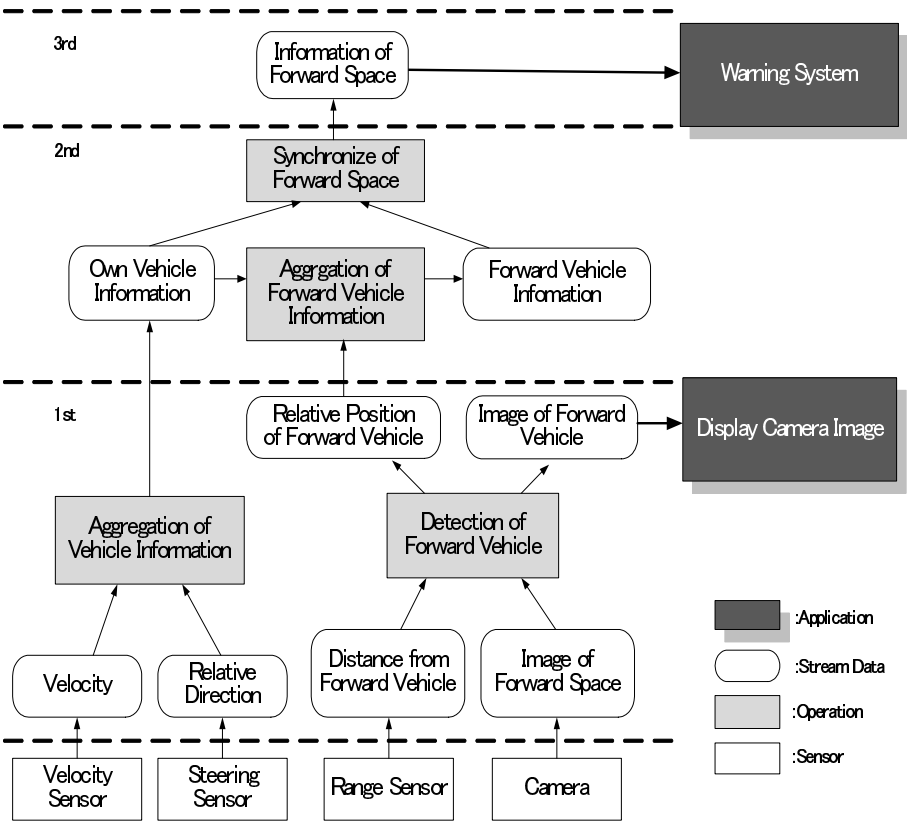

Figure 4.2: 実装システムのためのデータストリーム

性を統合的に判断し危険度に応じて警告するシステムである．危険度は衝突までの時間 1〜3 秒の範囲で赤色，3〜10 秒の範囲で黄色の文字で警告を行う．

(2) Display Camera Image(前方車両協調カメラ映像表示システム)　前方車検出を行いカメラ映像により強調表示するシステムとする．

実装システムの構成を図 4.2 に示す．実車両に複数のセンサを搭載し，それらのセンサから得られたデータを解析する研究があり [32] ，今回入力として利用するデータは，その研究で利用されているデータ収集車に取り付けられた車載カメラ，車間距離センサ，ステアリング角センサ，車速センサから得られたデータである．車載カメラから得られた前方画像，および車間距離センサより得られた前方車との距離情報を元に，前方画像の中から前方車部分を検出する．一方，車速センサから得られた車速データ，ステアリング角センサから得られた操舵角から自車データを集約する．

Display Camera Image は，前方車検出された画像を取得し表示を行う．また，自車データと前方車データの相対座標から前方車情報を計算し，自車データと同期を取り前方情報とする．Warning System はこの情報を取得し衝突までの時間を計算し状況に応じて警告を行う．ここで設計した階層型データの実現にはデータ間での変換処理とデータ提供を行う仕組みが必要であり，Borealis を利用しストリーム演算を実装することで実現した．

4.4.4 ストリーム演算の実装

階層化を行ううえで，各データ間の変換処理が必要になる．本研究ではこの変換処理を Borealis のストリーム演算として実現した． Borealis は元来，環境のモニタリングを行うアプリケーションのための仕組みとして開発されたものであり，車載データの用途としては作られていない．そのため Borealis に標準で用意されているストリーム演算に加えて，前方車認識などストリームデータに対する固有の処理を別途追加実装した．今回，前方車検出も含め, 以下のストリーム演算を追加することで車載データ統合モデルの実現を行った．

(1) Detection of Forward Vehicle (DFV)　前方画像と前方車との距離を取得し，前方車検出を行い，前方車と自車の位置偏差と前方車検出画像を出力するストリーム演算である．

(2) Aggregation of Vehicle Information (AVI)　車速，操舵角を入力として受け取り，自車の座標，車速，方向を計算するストリーム演算である．

(3) Aggregation of Forward Vehicle Information (AFVI)　(1)，(2) の出力ストリームを受け取り，前方車の座標，車速，方向を導出するストリーム演算である．

(4) Synchronize Forward Space (SFS)　(2)，(3) の出力ストリームを受け取り，前方空間に存在する情報を同一のタイムスタンプで集約するストリーム演算である．

Table 4.1: 評価時間

項目	仕様
CPU	PentiumD CoreDuo 2.8GHz
Memory	2GB
OS	Ubuntu(Linux) 8.04
通信プロトコル	TCP/IP

4.5 評価

本研究では，車載データの階層構造を構築し，そのデータを利用するアプリケーションプログラムの要求に応じてデータ提供を行うシステムを構築した．このとき，上位階層のデータ取得の際にはオーバヘッドが生まれるため，データを利用するシステムごとにオーバヘッドを許容できるかどうかの問題が生じる．そこで Warning System と Display Camera Image を実行する際，それぞれの階層からデータの取得を行うとき，センサデータの入力から目的とするデータを取得するまでのオーバヘッドの測定を行った．

4.5.1 評価環境

評価環境を表 4.1 に示す．評価に利用したデータは，あらかじめ車両にセンサを取り付けた上で路上を走行して得られた過去のデータを利用する．データの周期は Camera Image と Steering Sensor，Velocity Sensor が 33ms，Range Sensor が 11ms となっている．Range Sensor だけ周期が短いのは前方車認識を行う際，前方車の有無の判断に利用する重要度の高いデータであることが理由である．通信時間として以下の項目を測定した．

- Range Sensor → DFV
- DFV → Display Camera Image
- DFV → AVFI

Figure 4.3: スクリーンショット

・AVFI → SFS

・SFS → Warning System

通信時間とは，データの送信から受信先がデータを取得しストリーム演算の実行を開始するまでの時間とする．

データストリーム管理システムの評価環境実行時の画面を図 4.3 に示す．画面左下側の 4 つのウィンドウがそれぞれセンサデータを入力するプロセスのコンソール画面を表しており，画面左上側の 4 つのウィンドウはそれぞれストリーム演算を実行しているコンソール画面となっている．また，データ取得を行い衝突の危険を警告する Warning System と前方車検出画面を表示する Display Camera Image がそれぞれ画面右側に表示されている．

4.5.2 評価結果

通信時間の評価結果を表 4.2 に示す．Range Sensor と DFV の間で通信時間が 13.0ms かかっているのは画像データ待ち合わせのためである．またストリーム演算の実行

Table 4.2: 評価結果

送信	受信	通信時間
RangeSensor	DFV	13.0ms
DFV	Display Camera Image	7.6ms
DFV	AVFI	2.3ms
AVFI	SFS	1.0ms
SFS	Warning System	0.9ms

時間として DFV の計測を行い，結果が 3.6ms となった．他のストリーム演算の実行時間はごく小さいため今回の計測では無視をする． Range Sensor のデータ入力から Warning System にデータが到着するまでのオーバヘッドが 20.8ms，Display Camera Image までのオーバヘッドが 20. 6ms であった．一般に衝突警告システムは総合遅延時間が 100〜800ms の範囲であれば有効であるという評価が出ているため [33]，今回のプロトタイプ実装では時間制約を満たしている．

4.6 まとめと今後の課題

本研究では，車両制御システムのための車載データ統合モデルを提案し，データストリーム管理システムである Borealis を利用しプロトタイプ実装を行い，その評価を行った．車載データ統合モデルを利用することで，アプリケーションプログラムが選択する階層によって取得するデータ数を減らすことができ，効率よくデータ取得が可能になるという利点がある．車速，ステアリング角，車間距離，車載カメラの映像といったセンサデータを入力とし，前方車強調カメラ映像表示システム，および，衝突警告システムのアプリケーションプログラムを実装し，システムからデータ取得を行う際の時間を計測することで，階層化によるオーバヘッドの検証を行った．PC を利用した実装環境においては，本システムのオーバヘッドは許容範囲であり，実現可能性を確認できた．

今後の課題として，実環境を想定した実装および評価と，そのオーバヘッドの低

減化があげられる．さらに，多くの車両制御システムによる車載データ統合モデルの検証を行うことも必要である．

CHAPTER 5

関連研究

5.1 LDM

European Commission Information Society Technologies によって行われている SAFE SPOT プロジェクト [34] では，Local Dynamic Map(以後 LDM) が提案されている [35]. LDM はドライバに危険を警告する車両安全システムのためのデータ統合プラットフォームである．

LDM はデータベース管理システムを用いてすべてのデータをデータベースのテーブルとして管理している．そして 2 つのアプリケーションプログラムインタフェース (以後 API) をデータアクセスのために用意しいる．それらは Level1 API と Level2 API と呼ばれる．Level1 API は SQL ベースのクエリであり，Level2 API は Level1 API よりも高度なクエリである．たとえば，データベースにある車両と道路のリレーショナルデータを用いることで，自車両が走行している道路上にいる車両のリストを提供することができる．LDM によって管理されるデータは，道路標識のような静止している物体や，車や道路上にいる人など移動している物体といった，周辺認識によって得られる関連するすべての静的，一時的，動的な情報を表現している．

LDM はデータベース管理システムによるデータ管理手法である．しかし，動的な

情報のように更新頻度の多いデータにとって，このデータ管理手法は不適切である．なぜならアプリケーションプログラムは，データが更新される毎に，最新の情報を取得するために SQL クエリを呼ぶ必要があるためである．それゆえ，本研究はデータベースとデータストリームという 2 つのアプローチを用いて実装し，それらのパフォーマンスを評価を実施した．

5.2 PREVENT

PReVENT[36] は車両の衝突防止や道路上の安全を確保するするため European Commission により設立された研究プロジェクトである．このプロジェクトのサブプロジェクトである ProFusion2[37] は，複数のセンサの融合により障害物や外部の環境の認識の精度を上げることを目的としたプラットフォームの研究を行っている．センサフュージョンを実現するための一手法として，一般的な占有グリッド技術を用いており，センサの追加や削除，センサの種類の変更に柔軟に対応できるものではない．また，本研究で実施しているようなデータに対しての統一的なアクセス方法に関して，具体的な実現方法を示しているものではない．

5.3 ITS-SAFETY2010

国土交通省自動車交通局が進める安全運転支援システムを実現するためのプロジェクト [38] であり，衝突被害軽減ブレーキ，前方車両追従などのアプリケーション技術の確立を目指すものである．本プロジェクトにおいては，それぞれのシステムは独立して開発され，機能，性能を確認することが目的であり，それぞれのシステムを融合し効率的な開発を目指すものではない．

5.4 ロボット用サービス提供のための共通プラットフォーム

ロボットが利用者に対してサービスを提供するため，利用者の位置や状況，行動の意味に関する環境情報を構造化し理解するための研究が行われている [39]．この研究は，室内外におけるロボットサービスに必要な人や物の位置を計測・蓄積・構造化すると共に，様々なロボットサービスにおいて汎用的に利用可能とする共通プラットフォームである．そのプラットフォームでは，位置情報の計測・蓄積・構造化技術をシームレスに扱う四層構造化モデルを提案している．一方，本研究においては，利用するアプリケーションプログラムが必要なデータ階層を選択してクエリを行うため，アプリケーションプログラムは最上位の階層以外に，下位の階層のデータも利用可能である．このことは，車両統合制御システムに応じて要求するデータの数や種類に違いがあり, かつ許容できるオーバヘッドも違いがあるためである．

5.5 RT-MIDDLEWARE

ロボットを機能単位で分割し，それらを集積することで一つのロボットシステムを構築するためのミドルウェア (RT-middleware) がある [40]．本研究では，機能単位からではなくデータ単位で統合管理を行い，データ取得の効率化を実現するということを目標としている．

5.6 プローブカー

また，プローブカーで車内の情報をセンターへ送信し，センターで多数の車両情報を集計，または統計を取ることで，車両に対し交通情報や天候情報通知のサービスを行うシステムがある．そのようなシステムにおいても，車内のデータに対し統一表

現を定め，異なるベンダ間でも車両とセンター間でデータの送受信ができるようにするための研究がある [41]．本研究との違いは，自車データだけではなく前方車や歩行者などの車両周辺データも対象にしている．さらにデータの管理に階層構造を設け，利用するアプリケーションプログラムによって提供するデータの形式を変化させることでデータ取得の効率化を行っている．

CHAPTER 6

結論

6.1　まとめ

本研究では，センサの数や種類の増加によっておきる車両制御システムの開発コストの増加に対応するため，センサから得られたデータを管理し，複数のアプリケーションプログラムから共通に利用可能なデータ管理システムの提案し，シミュレータ相当の環境で動作する車両制御システムのアプリケーションプログラムへの適用と実現可能性の検証を行った．3 章では，車両追従システム，自動駐車システムの 2 つの車両制御システムを用意し，それぞれのシステムのアプリケーションプログラムが共通のデータベース管理システムを利用して動作することを確認し，データ管理システムの適用を行うことができた．また，実時間性の評価によって車両制御システムが許容可能なオーバヘッドの範囲内に収まることを評価し，実現可能性の確認を行うことができた．4 章では，前方車強調カメラ映像表示システム，衝突警告システムの 2 つの車両制御システムを用意し，それぞれのアプリケーションプログラムが共通のデータストリーム管理システムを利用して動作することを確認し，データ管理システムの適用を行うことができた．また，衝突警告システムの遅延時間の評価によって，時間制約を満たしていることを確認し，実現可能性の確認を行うことができた．

6.2 今後の課題

本研究では，2 種類のデータ管理手法を使い，汎用 PC 上で 2 つの車両制御システムのアプリケーションプログラムから利用することで評価を行った．今後，汎用 PC と実際の車両制御システムにおける組込み環境の間の差異について問題とならない環境における再評価が必要である．また，車両制御システムを動作させた際に用いたシナリオを拡充し，より実際の状況に即したシナリオで評価を行う必要がある．実際の開発においては，搭載される車両制御システムを列挙し，それらのシステムが利用するデータを参考にデータモデルを再定義し，データ管理システムを導入した場合とそうでない場合のアプリケーションプログラムの開発コストの比較評価が必要である．また，実際の車両制御システムにおける組込み環境は，ハードウェア，ソフトウェアの両面で汎用 PC とは特徴が異なる．ハードウェア面では，CPU，メモリ，ディスク，通信などの HW リソースは小さい傾向にあり，ソフトウェア面では，OS やネットワークプロトコルに違いがある．このような違いに対処するため，実際の車両制御システムにおける組込み環境において，次に上げるような項目が，データ管理システムを構築するための課題となると考える．

最適化

データ管理システムを用いる目的の一つは，最適化にある．最適化には，計算量，通信量削減，メモリ使用量の削減といったような種類がある．また，最適化を実現するためには，問い合わせ処理の重複部分の統合やバッチ処理，データの圧縮や間引きといった方法や分散環境におけるデータと処理の配置の工夫といった手法をとる．組込み環境における HW リソースの制約は厳しいため，各 ECU の制約を満たすために必要な最適化手法の取捨選択や，いかにして処理とデータを分散された各 ECU に配置するかという手法を確立する必要がある．

リアルタイム性

車両制御システムでは，人身の安全にかかわる機能を満たすため，リアルタイム性が要求される．そのため，車両制御システムを実現するアプリケーションプログラムは，データの問い合わせに対して，一定範囲まで許容できる遅延時間を持ち，それ以上の遅延時間が経過した場合は，そのアプリケーションプログラムは不適合とされる．このような性能を満たすため，OS はスループットに重点を置く汎用 PC とは異なりリアルタイム OS が多く適用され，データ通信には Ethernet ではなく CAN や Flexray が用いられる．一方で，汎用 PC 向けに開発されたデータ管理システムでは，データへの問い合わせを一定の処理単位として区切り，内部で有しているスケジューラを用いて処理単位の実行を進める．このような方法では，スケジューラが 2 重化され，また，汎用 PC のスケジューラに依存するためリアルタイム性を満たすことが難しい．そのため，データへの問い合わせを要求するアプリケーションプログラムが，その問い合わせに対してどれくらいの遅延を許容できるかを考慮しつつ，データ管理システムが内部で行うべきデータ処理をタスク化して，リアルタイム OS 上にマッピングし，アプリケーションプログラムが要求するデータ問い合わせに対するリアルタイム性を保証する必要がある．

開発環境

ECU に接続されたセンサからデータを取得し，何らかの処理を行い，アクチュエータの操作を行うソフトウェアを開発する場合，データの流れに着目して処理を行うブロックを必要に応じて組み合わせるデータフロープログラミングが用いられる場合が多い．これはデータストリーム管理システムにおける問い合わせ処理の記述方法と共通する点が多いが，その問い合わせ処理は仕様を把握して記述する必要があり，不慣れな開発者には敷居が高い．そのため，データフロープログラミングで用いられるグラフィカルインタフェースのような形で問い合わせの記述が容易にできる開発環境が

必要になる．また，本来はECU毎に独立して開発している状況であるが，データがどのECUから取得できるかをソフトウェア開発者が意識しなくてもよくなるように，最終的に各ECUでどのようなデータへの問い合わせがありどのようなデータを提供すべきかをECU上で動作するソフトウェア全体を見て，自動的に構成するソフトウェア開発環境が必要である．

謝辞

本論文に関する研究の機会を与えていただき，多くのご指導，ご鞭撻を賜りました，名古屋大学大学院情報科学研究科の高田広章教授と本田晋也准教授に心から感謝致します．研究生活を送る上で，日頃からご支援いただき，知的刺激や活力を与えいただきました，同志社大学理工学部情報システムデザイン学科の佐藤健哉教授をはじめとする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のみなさま，高田研究室のみなさまにも深く感謝致します．また，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へ出向し研究生活を通して成長する機会を与えていただいた東芝ソフトウェア技術センターのみなさまに深く感謝致します．最後に，長い研究生活の間，私生活の面から支えてくれた家族に心から感謝致します．

参考文献

[1] 浅沼 信吉, 加世山 秀樹, ”安全運転支援のための車両予知・予測技術のとりまく状況, ”国際交通安全学会誌,Vol.31, No.1, pp.56-61, 2006.

[2] 西垣戸 貴臣, 大塚 裕史, 坂本 博史, 大辻 信也, ”予防安全の高度化を実現するセンサーフュージョン技術 ”, 日立評論,Vol.89, No.08, pp.72-75, 2007.

[3] W.D.Jones, ”Keeping Cars from Crashing”, IEEE Spectrum, Vol.38, Issue 9, pp.40-45, 2001.

[4] 藤田 浩一, 宇佐見 祐之, 山田 幸則, 所 節夫, ”衝突危険性のセンシング技術, ”自動車技術,Vol.61, No.2, pp.62-67, 2007.

[5] 国土交通省 ASV 推進検討会, 第 4 期 ASV 推進計画パンフレット: ”ASV，それは交通事故のない社会への架け橋”, 2006.

[6] S. Fürst, J. Mössinger, S. Bunzel, T. Weber, F. Kirschke-Biller, P. Heitkämper, G. Kinkelin, K. Nishikawa, and K. Lange, “Autosar–a worldwide standard is on the road,” in *14th International VDI Congress Electronic Systems for Motor Vehicles, Baden-Baden, Germany (October 2009)*.

[7] 伊東維年, ”カーエレクトロニクス化の進展とその課題 ”，産業経営研究，No. 29, pp.65-88（2010）

[8] 北川博之, ”データベースシステム ”，昭晃堂，2009.

[9] 速水治夫，宮崎収兄，山崎晴明，”データベース ”，オーム社，2009.

[10] Sirish Chandrasekaran, et al：TelegraphCQ: Continuous Dataflow Processing, Proceedings of International Conference on Management of Data, pp.668-668 (2003)

[11] Daniel J.Abadi, et al：Aurora: a new model and architecture for data stream management, The VLDB Journal, Vol.12, No.2, pp.120-139 (2003)

[12] Abadi, D.J., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack,M., Hwang, J.H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., Zdonik, ”The design of the borealis stream processing engine, ” Proc. CIDR, 2005.

[13] 渡辺 陽介, 北川 博之, ”連続的問合せに対する複数問合せ最適化手法 ”, 電子情報通信学会論文誌 Vol.J87-D-I, No.10, pp.873-886, 2004.

[14] Yanif Ahmad, et al：Distributed Operation in the Borealis Stream Processing Engine, ACM SIGMOD International Conference on Management of Data, pp.882-884 (2005)

[15] International Organization for Standardization, ”Controller Area Network (CAN) - Part 1: Data Link Layer and Physical Signaling, ISO 11898-1:2003, ”International Organization for Standardization, 2003.

[16] 自動車技術専門委員会, ”JISD0802 自動車－前方車両衝突警報装置－性能要求事項及び試験手順, ” 日本工業標準化委員会, 2002.

[17] 大杉 啓治, 宮内 邦宏, 古居 信之, 宮越 博規, ”ACC システム用スキャン式レーザレーダの開発, ” デンソーテクニカルレビュー, Vol.6, No. 1, pp. 43-47, 2001. A. Elfes, ”Sonar-Based Real-World Mapping and Navigation, ”Int. J. of Robotics and Automat, Vol. 3, No. 3, pp.249-265, 1987.

[18] R.Ramakrishnan,J.Gehrke, ”Database Management Systems, ”, McGraw-Hill, Singapore, 2002.

[19] A. Elfes, ”Sonar-Based Real-World Mapping and Navigation, ”Int. J. of Robotics and Automat, Vol. 3, No. 3, pp.249-265, 1987.

[20] Avi, Scenegraphs:Past,Present,and Future, RealityPrime, 入手先<http://www.realityprime.com/arti cles/scenegraphs-past-present-and-future> (参照 2009-09-15).

[21] H.Choset and K.Nagatani, “ Topological simultaneous localization and mapping (slam) toward exact localization without explicit localization, ” IEEE Trans. on Robotics and Automation, Vol. 17, No. 2, pp.125-137, 2001.

[22] 飯島 徹也, 東又 章, 奥田 次郎, 浅田 哲也, 橋詰 武徳, 溝口 和貴, ”車間自動制御システムの開発, ”自動車技術,Vol.53, No.11, pp.98-103, 1999.

[23] 大前 学, 橋本 尚久, 清水 浩, 藤岡 健彦, ”駐車場を有する構内における自動車の自動運転の運動制御に関する研究, ”自動車技術,Vol.35, No.3, pp.235-240, 2004.

[24] Virtual Mechanics Corporation, 車両運動シミュレーションソフト, CarSim:車両運動モデル, 入手先<http://carsim.jp/category/1275944.html>, (参照 2009-09-15).

[25] CYBERNET SYSTEMS CO, SimulinkR 7, サイバネットシステム, 入手先<http://www.cybernet.co.jp/matlab/ products/product_listing/simulink/index.shtml>, (参照 2009-09-15).

[26] Virtual Mechanics Corporation, 車両運動シミュレーションソフト, トピックス:バーチャルメカトロニクス, 入手先<http://carsim.jp/category/1286799.htmll>, (参照 2010-02-22).

[27] The Machine Perception and Intelligent Robotics Lab University of Malaga, The Mobile Robot Programming Toolkit (MRPT), Main P age - MRPT, 入手先<http://babel.isa.uma.es/mrpt/index.php/Main_P
age>, (参照 2009-09-15).

[28] OSG Community, OpenSceneGraph, osg, 入手先<http://www.openscenegraph.org/projects/osg>, (参照 2009-09-15).

[29] VMWare, <http://www.vmware.com>, (参照 2009-09-15).

[30] 飯山 真一，高田 広章, ”システム構成を考慮した CAN の最大遅れ時間解析手法, ”情報処理学会論文誌,Vol.45, No.SIG1(ACS 4), pp.66-76, 2004.

[31] 西垣戸貴臣，大塚裕史，坂本博史，大辻信也：予防安全の高度化を実現するセンサーフュージョン技術，日立評論，Vol.89，No.08，pp.72-75(2007)

[32] Nobuo Kawaguchi, et al：Multimedia Corpus of In-Car Speech Communication, Journal of VLSI Signal Processing Systems, Vol.36, Issue 2-3, pp.153-159 (2004)

[33] 高取祐介，長谷川孝明：衝突予測による警告型安全運転支援システムにおける予測手法に関する一検討, 情報処理学会研究報告. ITS，Vol.2003，No.89，pp.13-18(2003)

[34] U. of Stuttgart Institute for Human Factors and T. Managemen. (2011,June) Safespot. [Online]. Available: `http://www.safespot-eu.org/`

[35] P. Lytrivis, G. Thomaidis, I. Karaseitanidis, and A. Amditis, “Situation refinement for in-vehicle platforms in vehicular networks,” in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, pp. 204–209.

[36] ERTICO, PReVENT, PReVENT :: Home, 入手先<http://www.prevent-ip.org/download/Publications /Profusion_E-Journal_volume_2_Final.pdf>, (参照 2009-09-15).

[37] PReVENT :: ProFusion2, 入手先<http://www.prevent.
-ip.org/en/prevent_subprojects/horizontal_activities
/profusion2/>, (参照 2009-09-15).

[38] 国土交通省報道発表資料, ”ＩＴＳによる安全運転支援システムに係る公開デモンストレーション等の実施について”, 入手先<http://www.mlit.go.jp/report/press/jidosha07
_hh_000019.html>, (参照 2009-09-15).

[39] Shuichi Nishio, Norihiro Hagita, Takahiro Miyashita, et al：Robotic Platforms Structuring Information on People and Environment, Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.2637-2642 (2008)

[40] Noriaki ANDO, et al：Composite Component Framework for RT-Middleware (Robot Technology Middleware), 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp.1330-1335 (2005)

[41] 佐藤雅明：インターネットにおける自動車情報の抽象化およびデータ辞書モデルの設計，Master’s thesis，慶應義塾大学政策・メディア研究科 (2001)

研究業績

主論文に関連する研究業績

査読付きの学術雑誌論文

1. 山田真大, 鎌田浩典, 佐藤健哉, 手嶋茂晴, 高田広章, “ データストリーム管理機構を利用した車載データ統合モデルの提案と評価 ”, 自動車技術会論文集 Vol.41, No.2, March 2010.

2. 山田真大, 鎌田浩典, 佐藤健哉, 手嶋茂晴, 高田広章, “ 車両制御システムのためのセンサデータ統合管理方式の検討 ”, 電子情報通信学会論文誌　VOL.J93-D NO.7 pp1189-1201.

査読付きの国際会議論文

1. Masahiro Yamada, et al. “ Implementation and Evaluation of Data Management Methods for Vehicle Control Systems ”, VTC2011-Fall, 2011.

学会での口頭発表

1. 山田真大, 鎌田浩典, 佐藤健哉, 手嶋茂晴, 高田広章, “ ストリームプロセッシングによる車載統合制御システムのための分散型センサデータ処理機構の構築 ”, 情報処理学会研究報告, Vol.2009-ITS, No.24, Feb 2009.

2. 山田真大, 青木優, 日高隆博, 山崎二三雄, 佐藤健哉, 高田広章, “ 交通事故シナリオに基づく予防安全システムのシミュレーション分析 ”, 情報処理学会研究報告, Vol.2010-ITS-41 No.2, Jun 2010.

その他の研究業績

査読付きの学術雑誌論文

1. 山田真大, 小林良岳, 本田晋也, 高田広章, “ マルチコアにおけるコア隔離機構 Timer Shield による Linux のリアルタイム性向上 ”, コンピュータソフトウェア VOL.13 NO.4 pp77-96.

査読付きの国際会議論文

なし

学会での口頭発表

1. 中嶋健一郎, 山田真大, 長尾卓哉, 山崎二三雄, 武井千春, 本田晋也, 高田広章, “ ARMv6 アーキテクチャを用いたメモリ保護 RTOS のタスクスタック保護の設計と評価 ”, 情報処理学会研究報告, Vol.2009-EMB-14, No.8, Jul 2009.

2. 石谷健, 山崎二三雄, 長尾卓哉, 山田真大, 松原豊, 本田晋也, 高田広章, ”車載 ECU 統合向け異種 OS 間通信ミドルウェア”, 情報処理学会研究報告, Vol.2010-EMB-17, No.3, Jun 2010.

3. 長尾卓哉, 山崎二三雄, 山田真大, 石谷健, 松原豊, 本田晋也, 高田広章, ”DUOS: 車載 ECU 統合向け RTOS フレームワーク”, 情報処理学会研究報告, Vol.2010-EMB-17, No.2, Jun 2010.

4. 勝沼聡, 山田真大, 本田 晋也, 佐藤 健哉, 高田 広章, “ ストリーム処理を用いた

車々間通信データのフィルタリング方式 ”, 情報処理学会研究報告, Vol.2011-ITS-45 No.4, Jun 2011.

5. 勝沼聡, 杉本明加, 山口晃広, 山田真大, 金榮柱, 本田 晋也, 佐藤 健哉, 高田 広章,“ 車載システム向けストリームデータ処理の提案と評価 ”, 情報処理学会研究報告, Vol.2011-EMB-23 No.1, Nov 2011.

6. 島田秀輝, 山田真大, 佐藤健哉, “ 行動履歴を用いた詳細道路情報収集システムの提案 ”, 情報科学技術フォーラム講演論文集, Vol10, No.4, Sep 2011.

7. 山田真大, 林和宏, 鈴木章浩, 岡本幸大, 小林良岳, 本田晋也, 高田広章, “ CPU affinity による汎用 OS のリアルタイム性向上手法 ”, 情報処理学会研究報告, Vol.2013-OS-126 No.18, Aug 2013.