計測自動制御学会東北支部 第 197 回研究集会 (2001.10.13)
資料番号 197-12

ドライビングシミュレータ用仮想環境の構築

Construction of Virtual Reality Environment for Driving Simulator

相原泉太郎 *, 野村亮太 **, 砂田哲也 **, ○熊谷正朗 **, 江村超 **

Sentaro Aihara*, Ryota Nomura**, Tetsuya Sunada**,
Masaaki Kumagai**, Takashi Emura**

*キヤノン株式会社,　**東北大学大学院

*CANON INC., **Tohoku University

キーワード : バーチャルリアリティ(Virtual reality), ドライビングシミュレータ (Driving simulator), 立体視 (Binocular stereopsis), ヘッドマウントディスプレイ (Head mounted display)

連絡先 : 〒 980-859　仙台市青葉区荒巻字青葉 01　東北大学大学院工学研究科機械電子工学専攻 江村研究室
熊谷正朗, Tel.: (022)217–6969, Fax.: (022)217–6967, E-mail: kumagai@emura.mech.tohoku.ac.jp

1. はじめに

近年, 海外で自動車を借りて運転する旅行者が増えている. 旅先での移動手段として自動車は便利であるが, 日本国内との運転規則 (標識など), 道路形態の特徴 (ロータリー, 立体交差), 慣習などの相違のために混乱し, 交通事故に至る場合もある. 特に, 左側通行と右側通行の違いは決定的である. しかし, このような危険は, 渡航前に訓練することで大幅に軽減することが可能であると考えられる.

そこで, 著者らはバーチャルリアリティ(以下 VR という) を用いたドライビングシミュレータの開発に着手した [1, 2]. 基本となるシステムに, 各国別の環境データを搭載することで運転練習を可能とすることを目標としている. 一般に VR の臨場感を高めるには, 構成する 3 次元コンピュータグラフィック (以下 3DCG という) を精細にし, 体験者の動作を違和感無く反映させることが重要であるが, 環境の構築に要する手間が多くなり, また処理速度が低下する. 本論文では, 実在の地図や市販の 3DCG 用データを元にした VR 用の環境生成システム, 体験者の運転する車両と同時に環境内を走行する対向車, 処理速度向上のために試みた手法について報告する.

2. シミュレータのハードウェア

本章では著者らが開発したドライビングシミュレータについて述べる.

2.1 システムの概要

シミュレータシステムの概要を **Fig. 1**に示す. 本システムは画像の生成, 自車 (体験者の運転する車両) と自律運転車 (指定された経路にそって自動的に運転される車両) の運動演算などを行うコンピュータ, 体験者にステレオ画像を呈示するステ

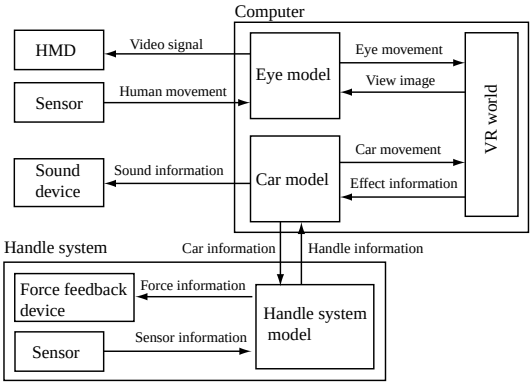


Fig. 1　Block diagram of driving simulator system.

Fig. 2　Stereo HMD and motion sensor.

レオ型ヘッドマウントディスプレイ (以下 HMD), 体験者の頭部の動きを検出する姿勢角センサ, ハンドル操作など体験者の動作を取得し, フィードバックする運転席と, ソフトウェア群からなる.

2.2　コンピュータと開発環境

コンピュータは可能な限り性能が高いことが望ましいが, コストパフォーマンスを考慮し, 市販のパーソナルコンピュータ (PC, PentiumIII 600MHz ×2) を使用した. 画像生成に用いるビデオカードには, 後述するステレオ出力をサポートする 3DLabs 社製 Oxygen GVX1 を使用した. OS は Microsoft 社製 Windows2000 であり, 同 Visual C++ 6.0 を開発に使用した.

2.3　VR 環境の呈示

2.3.1　ステレオ映像の生成と呈示

3DCG の生成には OpenGL を使用した. OpenGL は OS に対する依存性が低いため移植性に優れ, ステレオ表示も規格でサポートされている (ただし, 実装している市販ビデオカードは一部である). 生成した映像は Sony 製ステレオ対応 HMD, LDI-D100B によって呈示した (**Fig. 2**). この HMD は 1.2 [m] の位置にスクリーンが存在するように光学系が設定されているため, 映像生成パラメータである焦点距離は 1.2 [m] と設定し, 視線の方向もスクリーン中央とした.

周囲の環境のほか, メータ類を含む運転席内部, 各種ミラーに映る後方の映像も描画することで, 運転の臨場感を高めた. ミラーに映る後方の映像は, 体験者の視点のミラーに対称な位置に設定した仮の視点から見た映像を, 一旦メモリ上で描画した上で, 改めてミラーのオブジェクトにテクスチャとして貼付けることで実現した.

2.3.2　頭部の動作検出

映像生成に必要な視線の方向や視野の上方のベクトルを得るため, 3 軸まわりの姿勢が得られる, TOKIN 製 3D モーションセンサ MDP-A3U7 を使用した.

本センサは姿勢角のみを検出するセンサであるため, 頭を動かして陰になっている部分の確認をするなどの動作は再現できない. そこで, 本システム用に, 回転磁界と差動磁界を利用したモーションキャプチャを開発した [3, 4]. 本モーションキャプチャは姿勢角を高精度に直接検出することができ, 0.1 [deg] の分解能と 0.5 [deg] の精度を有する. また, 運転席を検出範囲とした場合, 位置検出の分解能は 0.5 [mm], 誤差は 5 [mm] 程度の性能が得られる. 検出システムは完成しており, 今後運転席と結合し, 頭部の動作を 6 軸で検出可能とする予定である.

2.3.3　環境音の呈示

運転の臨場感を出すため, 実車からサンプリングした音データを元に, エンジン音や方向指示器などの動作音を出力するようにした.

Fig. 3　Driver's seat with force-feedback steering wheel.

2.4　運転席

運転席は実際の自動車の運転席まわりの部品を利用して構成した (**Fig. 3**).

ハンドルは角度を検出すると共に反力を加えられるように, ロータリエンコーダ付きの DC サーボモータ (山洋電気社製 L720, 200W) を接続した. これを用いてハンドルに適当なトルクを与え, 路面からの反力や, ハンドルの遊び, ハンドル切れ角の限界などの感覚を与えられるように操作した. それに加えて, 回転角速度に比例したフィードフォワード出力を加えている. これは, モータを電圧出力のアンプで駆動したことで, 人間がハンドルを速く回そうとした場合にアンプに逆起電力が吸収され, ハンドルが重くなる現象が確認されたためで, 逆起電力分を補償した.

アクセルおよびブレーキペダルには, ポテンショメータをリンクを介して接続し, その踏み込み量を検出した.

その他, ギアのシフトポジション (AT 仕様), 各種ライト類, 方向指示器などのスイッチ類の操作も検出できるようにした. これらは, 自動車の運転に関わる他, 運転評価時にも使用できる.

運転席には日立製マイクロプロセッサ H8/3048 を組み込み, ハンドルの反力の制御を行った. 操作情報などは RS232C を通じて VR 用のコンピュータと送受する.

2.5　シミュレータのソフトウェア

シミュレータは体験者の運転操作により移動する車両や, 信号のような時間変化する要素を逐次演算すると共に, 道路や地形, 建物のような動きのない要素とあわせて, VR により体験者に映像を出力する. VR 環境の呈示についてはすでに述べているので, ここでは環境内で変化のあるオブジェクトの扱いについて述べる.

2.5.1　車両の運動

車両は 2 種類ある. 一つは運転操作が可能な自車であり, もう一つは環境内を自動的に走行している自律運転車である. 後者は, 市街地における運転の臨場感を高めると共に, 合流や追い越しといった運転操作を可能とするために不可欠な要素であると考えた.

車両は物理的なシミュレーションを行うことで動きを決定している. 進行方向に対しては, エンジンによる牽引力, 空気抵抗, 斜度による重力の影響, ブレーキによる制動を考慮し, 運動方程式を立てて数値計算を行った. 必要となる車両質量などのパラメータは, 実車のものを参考に, 運転感覚に違和感がないように決定した. 操舵に関しては, 実車同様にアッカーマンステアリングモデルを採用し, 舵角から旋回半径を求め, 進行方向を決定した.

自車については, 運転席で検出したハンドル, アクセル, ブレーキの各操作をこの演算の入力とし, 車両の位置を決定した. それに対して, 自律運転車は, 操作入力を演算で求める必要がある.

運転経路については, 道路情報を元に決定した. まず, 大域的にどの経路を走行するかを決定する. これは始点と終点を与えることで, Dijkstra 法によって生成することを可能とした. 局所的には, 経路内の車線に沿うようにフィードバック制御で操舵角を決定するようにした.

自律運転車の速度は, 道路に付随させた制限速度や一時停止, 信号といった情報を取得し, それらを遵守するようにした上で, 以下の式をもとに加

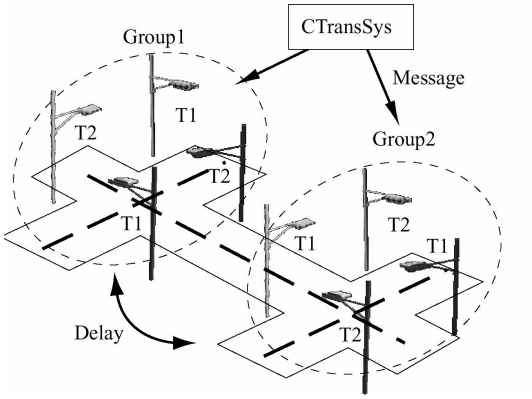


Fig. 4　Control of traffic lights.

減速を行った.

$$a(t+T) \propto \frac{v(t)^m}{[\Delta x(t)]^l}\Delta v(t) \qquad (1)$$

これは, 交通シミュレーションなどでも用いられる車追従モデル [5] であり, a と v はそれぞれ加速度と速度を示し, Δx と Δv は前方の車との距離と速度差, T は (運転手と車の両方合わせた) 応答遅れ時間であり, m と l は定数である. これらにより, 車両の増加に伴い自然に渋滞が発生するなど, 現実に近い交通状態を作り出した.

2.5.2　信号機の実装

信号機は単体では交通規則データとして, 一時停止などの標識と同列に, 道路上に配置されている. ただし, 信号機は外部からのメッセージを受け取り, 外観を変化させ, 車両の進行の可否情報を更新可能なように実装した. **Fig. 4**に示すように, このような信号機を交差点に複数配置し, 1グループとして設定し, 管理する機構を設けた. これは, 信号の切り替えタイミングを制御し, 切り替え時には該当する信号機にメッセージを送信するものである. 信号機本体と, タイミング制御を分離することで, 複数の信号を設置したり, 信号機の外観を変更することを容易とした.

3.　VR 環境の構築

実体感を高めるには, VR によって呈示する環境に十分な情報を用意することが必要である. 特に, ドライビングシミュレータの場合は移動可能な範囲が広いため多くのデータを要する. また, 本研究の目的は, 自由な市街地での走行であるため, 特定のコースのみの生成に留まらず, 広域の情報生成が必要である. そのため, 環境データの構築はシミュレータのソフトウェアの開発と並んで, 時間を要する作業となった. この作業を軽減するため, シミュレータと平行して環境構築用のシステムを開発した.

環境の構築は 3 段階からなる. まず道路網を生成し, 次いで配置する建物を製作する. 最後に, 道路網上に建物類を配置し, 標識・表示を配置して交通ルールを設定するという手順になる.

3.1　道路網の生成

道路については, 実在の都市の道路を既存の地図情報から再現可能とすることにした. 道路網を無から構築するのに比較し, 地図をもとに大まかに生成し, 手直しを加えるほうが, 構築は容易となる.

そのためには, 以下の手順で処理を行った.

1) 地図画像 (**Fig. 5**(a)) より, 道路領域を抜き出す (同 (b)).
(画像処理ソフトによる前処理, 2 値化等)

2) 細線化を行う (同 (c)).

3) 画像からピクセル間の連結状態を調べることにより, 道路の分岐点および端点を検索する.

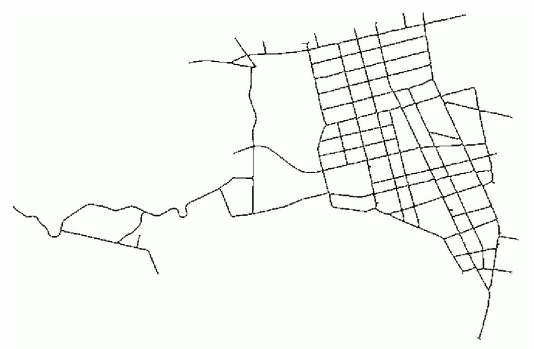
4) 分岐点を起点として追跡することで個々の道路を抽出する.

5) 最小二乗法により 6 次の Bézier 曲線に近似する (同 (d)).

最後に Bézier 曲線に近似を行ったのは, データ量の低減と, シミュレータ側の演算を容易にするためである. Bézier 曲線は変数 $0 < t < 1$ に対するパラメトリック曲線として与えられるが, 本システムでは, 各種標識や信号など交通規則は道路に

(a) Original map

(b) Binarized map



(c) Thinned roads

(d) Roads fitted to 6th Bézier curves

Fig. 5　Process of image of map into parametric road curves.

付随するものとして扱い, その位置を t によって表すようにした. 体験者の運転を評価する場合や自律運転車が運転規則を読み取る場合には, 空間座標から最寄りの道路上での位置を求めて照合するが, この場合にパラメトリック曲線は扱いやすく, そのなかでも Bézier 曲線は 道路の微修正などが容易であったため, 採用した. 道路のデータとしては, これに車線数を加えたものとし, シミュレータ側で道路の幅を表現するポリゴンを計算し映像を生成するものとした.

3.2　建物等の生成

建物, 標識・表示, 樹木は直方体, 平板などの低負荷のモデルに, 画像をテクスチャマッピングで貼付けることによって生成した. 標識等は幾何学的な構造のものが多く種類も少ないため市販の描画ソフトウェアによって, 樹木は実物の写真を加工することによってテクスチャを生成し, 適切な形状の平板に貼り付けることでオブジェクトとした (**Fig. 6**). 建物に関しては, 精細な写真を得ることは容易ではなく, また種類も多いほうがよいため, 市販の 3DCG 用データ集も使用することとした. これらは高精度な静止画の CG を生成することを目的としており, 非常に精密であるが, そのまま VR 環境に取り込むと非常に高負荷である. そこで, 一旦 3DCG ソフトウェアによってレンダリングし, 得られた 6 面の画像を簡素なモデルに貼付けた. この貼付け作業も, 座標を指定するなど手間がかかるため, **Fig. 7**に示すような専用のツールを開発し, モデルとテクスチャ画像を一体にして建物データファイルとした.

3.3　VR 世界編集ソフトウェア

以上の道路, 建物等を統合し, 交差点情報や交通規則を設定するための編集ソフトウェアを開発した (**Fig. 8**). このソフトウェア自体も環境の描画機能を備えており, 3 次元空間内に建物等を配置できる. また, 国土地理院発行の地形データを結合す

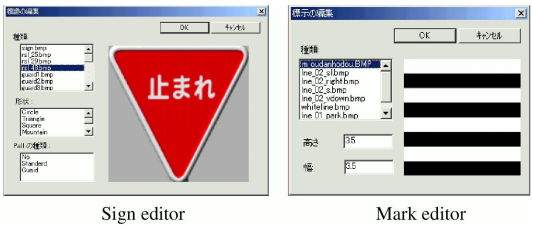


Fig. 6　Traffic signs and marks editor.

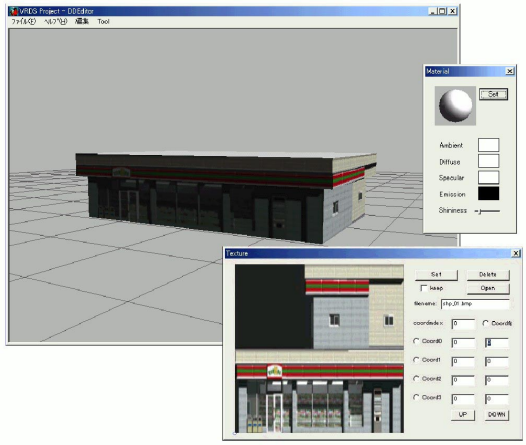


Fig. 7　Building creator.

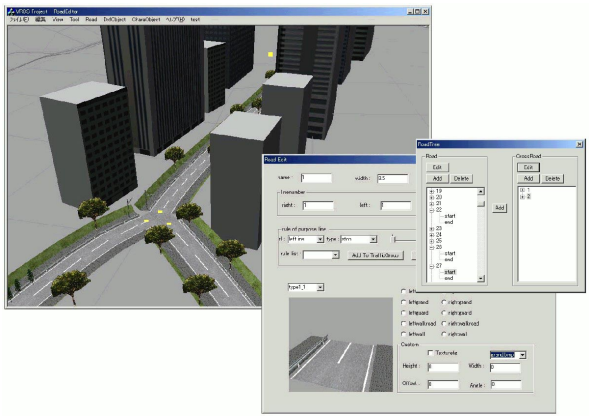


Fig. 8　VR environment developping tool.

Fig. 9　A sight in a VR environment.

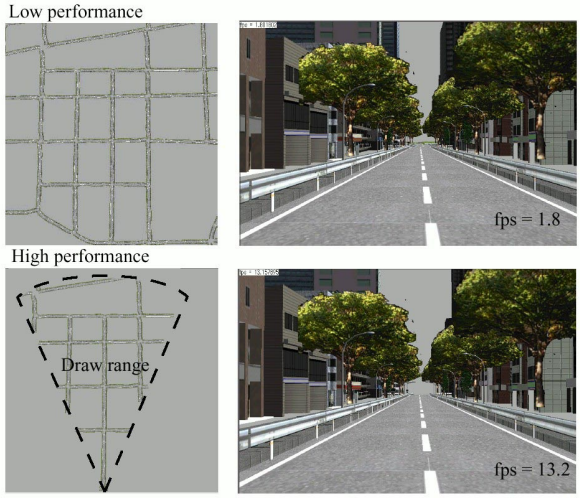


Fig. 10　Limiting the range of 3D drawing.

ることにより, 高低のある環境とすることもできる. このソフトウェアの出力がシミュレータ用のデータとなる. 構築例を **Fig. 9**に示す.

4.　高速化のための改良

環境をより緻密にするに従い, 描画速度が低下する. 代表的な性能値として, 1 秒間に映像を書き換える頻度である Frames Per Seconds(以下 FPS) 値があるが, これが 1 程度に低下することもあり得る. この場合, 動画として認識することはできず, 正常な運転操作はできない. そこで, 速度向上のための対策を検討した.

4.1　描画対象の限定

描画に用いた OpenGL のライブラリは与えられた 3 次元オブジェクトデータが描画領域に入っているか, 他のオブジェクトの陰になっていないかを判定した上で描画を行うが, 与えるオブジェクトデータが多い場合, この判定に要する処理時間は無視できない. そこで, ライブラリにデータを渡す前に, あらかじめ対象を制限することで高速化を試みた.

制限の方法としては, **Fig. 10**に示すように, 視点を扇央とし, 視線方向から左右に幅をもった扇形領域のみを描画対象とすることにした. これに

よって大幅に処理速度が向上し, 実装例では FPS 値で約 7 倍の高速化が得られた. 奥行き方向にも制限を加えてあるため, 大きな建物が視界に突然現れる場合があり, 何らかの改良が必要であると考えられる.

4.2 Dualプロセサ機とマルチスレッド化

市販のコンピュータでは, CPU を 2 個搭載した Dual プロセサ機は比較的容易に入手でき, 単純計算では 2 倍近い性能を得ることができる. ただしそのためには, 単一プログラムで構成したプログラムを複数の CPU で並列実行できる形に書き換えること (マルチスレッド化) が必要である. 本システムの場合は, 描画に使用する画像ボードは 1 枚であるため, 描画部は分割する利点がないと思われた. それに対して, 自車や自律運転車に関わる各種物理演算は, 描画と平行して進めることが可能である. そこで, プログラムをユーザインタフェースなどを担当する部分, 描画を行う部分, 物理計算を行う 3 つに分割した. 本システムの場合, 物理計算には演算量の多い衝突判定などの処理も含んでいるため, 並列化の効果を十分得ることができた.

4.3 ネットワーク分散処理

近年盛んである, ネットワーク分散処理の手法を本システムにも採用することにした. 分散は大きく二つの方法を試みた. 一つは物理計算の分離であり, もう一つは描画の分散化である.

前者の形態は, 物理計算を行うサーバと, その結果を元に描画するクライアントという形で実装した. 運転席はクライアント側のコンピュータに接続する. また, コンピュータ間は Ethernet(100BASE/TX) にて接続した. クライアントのコンピュータは, 体験者の運転操作を UDP/IP のパケットとしてサーバに送出する. サーバ側はこれを受け取り, 体験者の車や, 自律運転車の各種演算を行い, それぞれの座標, 方向を UDP/IP のパケットとして送出する. これらのパケットは, 近隣のネットワーク上の全コンピュータに到達するブロードキャストを用いて送信するものとした.

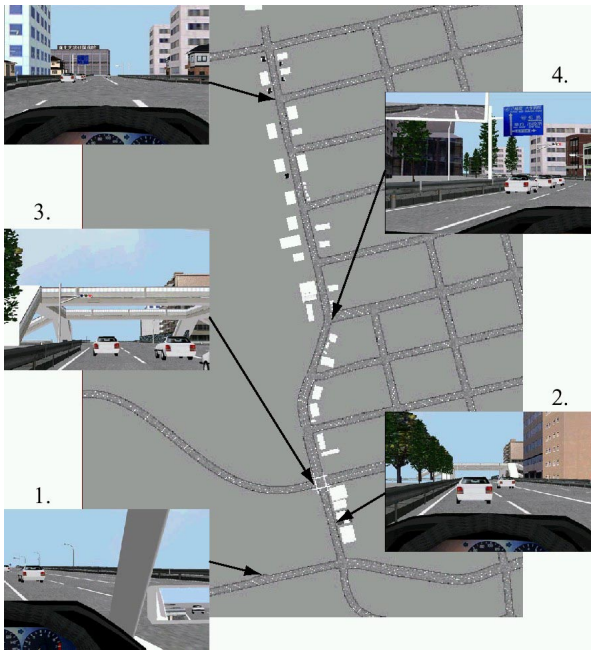


Fig. 11 Driving practice in VR environment.

この分散形態には, 負荷の分散に加えて, マルチユーザ環境やより高負荷なシステムを構築することができるという利点がある. 実際に, 複数台のクライアントコンピュータを用意し, 複数名が同時に各々の自動車を操作できることを検証した. また, 地図上で区域を分割し, 複数のサーバで分担するシステムも試作した.

後者の形態は, 具体的にはミラー内の画像描画を分担させた. 前述のように, 仮想的な視点から見える画像をあらかじめ計算し, ミラーにテクスチャとして貼り付けた上で呈示映像の描画を行う. そこで, ミラー用テクスチャを別のコンピュータで描画する方法を試みた. 視点の設定情報や, 生成した画像はネットワークにより送受する. ミラー内画像の精細さにも依存するが, 試作した例では, 全体の描画速度は 1.3 倍程度高速化することが確認された.

5. 環境の構築例と評価

以上のシステムおよびソフトウェアを用いることで, 簡単な教習コースデータを試作した. **Fig. 11** にその例を示す. FPS 値を高めるために想定経路上にのみ建物を配置しているが, 道路に関してはすべての路線を走行可能である. この路線上で, 合流, 追い越しなどが行えるようになっている. このほか, 標識・信号等をアメリカの制度に設定したデータも試作した.

システムの評価は, 複数名の被験者 (運転免許保持者) による試乗によった. 本システムの特徴の一つはステレオ映像による距離感の呈示であり, この評価のために運転データを記録し, その変動などの評価を行った. しかし, 視覚的な差異よりも慣れに起因する要素の方が大きく, 明確な数値上の差は得られなかった. 主観的には, 前方の車両などにある程度近づいたところでの安心感が違うなど, 差があることが確認されている. また, 運転の操作は実車とほぼ同じ感覚で行うことができたが, 唯一, 速度感覚だけは差異があった. 原因としては, 使用した HMD の視野の狭さが考えられ, 今後の検討課題である.

6. おわりに

本論文では, 著者らが開発しているバーチャルリアリティを用いたドライビングシミュレータについて, そのシステムと環境構築手法について述べた. 現在は, 環境呈示の表現力向上や運転評価の手法について研究を進めている. 将来的には, システム一式を公開し, オープンに環境構築を図っていきたいと考えている.

最後に, 運転席の製作には東北大学技官の鈴木正俊氏の, 環境構築においては, 東北大学大学院学生の森谷慎司君の協力を得た. 両氏と被験者の方々に感謝の意を表したい.

参考文献

1) 相原泉太郎, 森谷慎司, 熊谷正朗, 江村超: バーチャルリアリティを用いたドライビングシミュレータの開発, 計測自動制御学会東北支部 35 周年記念学術講演会, 講演番号 A12 , 59–60(1999)
2) 相原泉太郎: バーチャルリアリティを用いたドライビングシミュレータの開発, 2000 年度東北大学大学院工学研究科修士学位論文 (2001)
3) 江村超, 熊谷正朗, 野村亮太: 回転磁界と差動磁界を併用した 6 軸姿勢センサの開発, SICE 東北支部第 190 回研究集会 (2000)
4) Takashi Emura, Masaaki Kumagai, Ryota Nomura: Development of Motion Capture System using Rotating Magnetic Field and Differential Magnetic Field, SICE2001 国際セッション (2001)
5) 加藤泰義: セルオートマン法により道路交通シミュレーション, 人工知能学会誌, 15 巻 2 号 (2000)