社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－CVIM－154 (12)
2006／5／18

規範フロー場と実フロー場の差異解析による道路状況認識の研究

山中 隆* 中野 広樹* 下脇 克友* 渡邊 睦* 緒方 淳**
鹿児島大学 工学部 情報工学科* 〒890-0065 鹿児島市郡元 1-21-40
株式会社デンソーアイティーラボラトリ**

あらまし オプティカルフロー場を解析することによる外界認識の研究が行われている。しかしながら、移動中の視点から静止物体の検出、認識を行う研究例はまだ少ない。本稿では、障害物が存在しない標準的な環境で想定されるオプティカルフロー場を「規範フロー」として予め計算により求め、テレビカメラから入力した画像を処理して得たオプティカルフローとの差異を検出、解析することにより静止障害物を検出する新しい手法を提案する。本提案方式を PC 上のソフトウェアとして実装し、室内に設定した模擬環境にて実験を行い、その有効性を確認した。

キーワード 障害物検出、車載カメラ、オプティカルフロー、規範フロー、KLT 法

Research of traffic condition recognition by analyzing a difference of optical flow and model flow

Takashi YAMANAKA*, Hiroki NAKANO*, Katsutomo SHIMOWAKI*,
Mutsumi WATANABE*, and Jun OGATA**
Department of Information and Computer Science, Kagoshima University*
1-21-40 Korimoto Kagoshima, 890-0065, Japan
Denso IT Laboratory**

Abstract. Many researches have been executed for world recognition by analyzing optical flow field. However, there are still few research examples of recognizing a standstill object from moving cameras. This paper reports a technique to detect a standstill obstacle using a new idea "model flow" which is the optical flow field where there is no obstacle. The model flow is calculated using knowledge as for navigation environment. Standstill objects are automatically detected by analyzing the difference between a model flow field and a real optical flow field which is acquired by using moving camera images. This system was installed as a PC software. Experiential results in simulated outdoor environment have showed the effectiveness of the proposed method.

Keyword. obstacle detection, in-vehicle camera, optical flow, model flow

1. はじめに

自動車関連技術における主要課題は快適性、走行性から安全性を求めるものへと変化してきている。具体的にはABSやエアバッグなどで搭乗者の安全を確保する技術はほぼ確立しており、現在では事前に危険を察知して回避する性能の確立や、歩行者など外界への安全性を高めるための研究が盛んに行われている。

外界の安全確保のために認識するべき道路の状況と

して、カーブなどの道路形状、歩行者等の移動障害物、停止車両などの静止障害物などが挙げられる。本研究では静止障害物を対象とし、動画像解析手法を用いて自動的に検出する手法を新たに考案した。

2. 従来技術と課題

障害物検出の方法として、距離情報を用いるものと動きの情報を用いるものが考えられる。

距離情報を用いるものには、距離センサを利用する方法[1]や、ステレオ視を利用する方法[2]がある。動きの情報を用いるものには FOE を利用する方法[3]が従来研究されている。

距離センサでは、センサから物体までの距離や位置などを知ることが出来る。物体までの距離を高い精度で得ることが出来るため、低速での衝突を警告するなどの実用化もなされている。しかし走行中は対象物によって危険度や回避行動を判断しなければならない。そこで検出した障害物が何であるかまで認識できる事が望ましい。ここで距離の情報のみよりも、映像としての情報を用いる方法が適切であると考える。

映像から距離情報を求める方法にステレオ視がある。風景中の物体認識などに用いられている。しかしこの方法はある程度の距離をとった 2 つのカメラ画像の重複部分でしか処理が出来ず、視野が狭くなってしまう。そのためにカメラに近い部分は共通視野が出来ないため処理が出来ず、衝突する危険のある接近した障害物に対して、単眼での処理が出来る方法が適切であると考える。

映像としての情報から動きの情報を用いる方法に、オプティカルフローと FOE を利用するものがある。この FOE の違いを調べることによって背景と違う運動をする移動物体の検出に使われている。しかし静止障害物は背景と同じ FOE を持つため、本研究の手法とするには不適である。

そこで、移動状態で静止障害物を検出するために、従来の手法に代わる新手法として『規範フロー』という概念を提案する。

標準的な環境、標準的な移動状態から予測される、画像中の各場所におけるオプティカルフローを計算により求め、これを規範フローとする。そして動画像処理により得られた実際の画像のオプティカルフローとの差異を検出することにより、障害物を検出する。

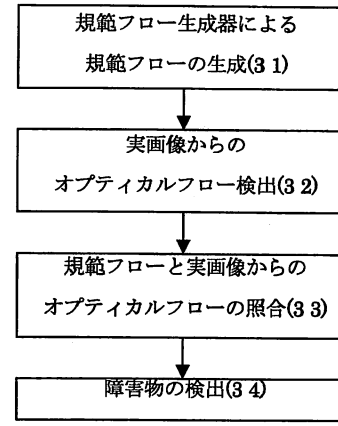

図 1　本システムのフローチャート

この手法は、FOE のようにいくつかのオプティカルフローを元に、ひとつの情報として取り出すのではなく、フロー場として考えることにより、ひとつひとつのフローの情報を利用できる。これによってノイズが出てきた場合も他のフローの情報により吸収できると考えられる。また、局所的にはベクトルを比べるだけという簡単な比較計算であるため、処理の負荷も軽く、リアルタイムで処理できると考えられる。

3. 処理の流れ

本システムのフローチャートを図 1 に示す。図中の番号は手法を説明している章番号である。

まず自動車の速度やハンドルの切り角などの情報を受け取り、仮想空間上の路面に配置した特徴点の座標を移動させて、その動きを画面に投影したオプティカルフローを算出する、規範フロー生成プログラムを作成した。この生成プログラムから算出されるオプティカルフローを規範フローとする。

次に KLT 法により実画像のフローを抽出。画面をブロックに分けて各ブロックに規範フローの長さや向きを設定。実画像のフローの長さと向きを調べて、そのフローの入っているブロックの規範との差異がないかを照合する。

最後に差異があると判断したフローがブロックの中にあった場合、そのブロックに障害物が存在するとして

検出する。

以下各処理について、規範フローの生成、入力画像中の特徴点抽出とフロー検出、規範フローと実フローの照合、障害物の検出の順に具体的に説明する。

3.1. 規範フローの生成

3.1.1. 生成プログラムの概要

本システムの基礎となる規範フローを生成するために規範フロー生成プログラムを作成した。

まず固定パラメータとして、カメラの路面からの高さ、カメラの焦点距離、水平に対するカメラの傾き、車の前輪と後輪の間隔、車の右の車輪と左の車輪の間隔の 5 つのパラメータを設定することにより、仮想空間上の路面に特徴点を配置する。

次に変数パラメータとして、スピード、タイヤの切り角を毎回受け取り、車の状態にあわせて特徴点の座標を変更。特徴点の位置を、はじめに設定した固定パラメータの情報から計算し、カメラ画像のどこに投影されるかを算出する。前状態と現状態をそれぞれ規範フローの始点と終点として出力する。

3.1.2. 速度の計算

実際のオプティカルフローは対象物の動く速さやその距離だけでなく、入力画像の取り込み時間間隔でも長さが変わってしまう。規範フローはその時間間隔にあわせて計算されなければならない。

規範フロー生成プログラムは車の速度を車の計器から取得し、システムの処理系に対応した値へと変換する。日本の自動車は一般的に速度を時速で表示しているため、表示された時速を x、時速から処理用に変換した値を y、システムが 1 秒間に処理できる回数を fps とすると、そのとき変換式は以下のようになる。

$$y = \frac{10^6 * x}{3600 * fps}$$

fps を定数にせずに、毎回算出し反映させることにより、実画像のフローとシミュレータの同期をとり、そのときの最高の処理速度で規範フローを算出することができる。

3.1.3. 画面投影の計算について

プログラム上の空間と画面の位置をピンホールカメラに近似させ、図 2 に示すように視点を原点とし、y 軸を奥行きとした空間を x－y 平面で考えたとき、カメラの焦点距離を f とすると、画面(CCD)の位置は

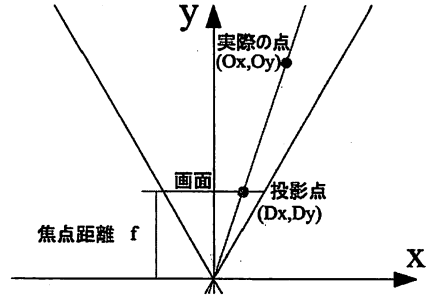

図 2　x・y 平面

$y = f$

となる直線である。また、画面の範囲は CCD の中心を y 軸とする CCD の幅になる。画面に投影される点は、この直線と、特徴点の座標と原点を結ぶ直線の交点になる。

特徴点の座標を(Ox,Oy)、投影点の座標を(Dx,Dy)とすると、Dy＝f となるから、Dx は次の式で求めることができる。

$$Dx = \frac{Ox}{Oy} * f$$

また y－z 平面で同じ計算をすることにより、投影点の z 座標を求めることができる。

この計算により画面の中心を原点とした z－x 平面に投影点をあらわすことができる。その後、CCD の大きさを、実際の入力画像の大きさに変換することにより入力画像に応じた規範フローを表示させる。

3.1.4. カメラの傾きを考えない特徴点の配置と移動

カメラの傾きが 0 度の場合、はじめに設定したカメラの高さを反映して路面の高さに配置された y 軸と平行な直線上に特徴点を配置し、その点は y 軸と平行に移動する。模式図を図 3 に示す。図中の環境点と書かれた点がこのときの特徴点である。環境点については後に述べる。この動きで相対的にカメラの移動を再現している。このときの移動量は先に紹介した 1 回の処理ごとの移動量となり、原点方向に移動する。ただしカメラの位置よりも手前にきた点は一番奥に再配置される。

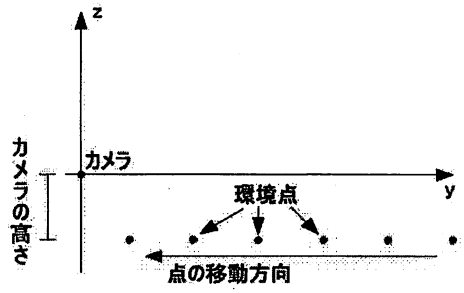

図3　環境点の設定と動作

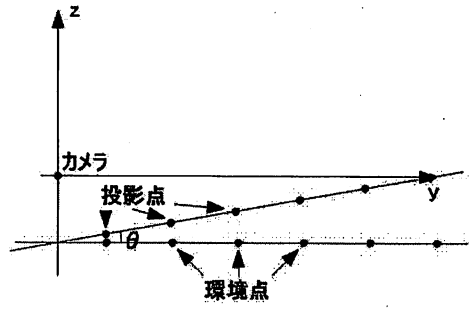

図4　カメラの傾き θ の時の投影点の配置

3.1.5. カメラの傾きを考えた特徴点の配置

カメラの傾きが θ の場合、y 軸に対し θ だけ傾いた平面上に点を配置し、その平面に沿って移動する。模式図を図 6 に示す。図中の投影点と書かれた点がこのときの特徴点である。投影点については後に述べる。

しかし傾けての移動は、カーブやカメラの傾きなどの状態変化の繰り返しによる誤差の蓄積が原因となるバグなどが生じてくる可能性がある。

そこで、カメラの傾きが 0 度で直進状態にある、誤差の生じにくい基準となる環境に配置した特徴点を「環境点」と呼ぶことにする。また、そこからカーブやカメラの傾きによって座標位置を変換した特徴点を「投影点」と呼ぶことにする。

環境点は図 5 に示すように、z 軸方向への変化が無く、傾けることによる z 軸方向へ移動に関するの誤差の蓄積を防ぐことが出来る。また、y 軸方向への移動も整数値に変換したスピードだけに依存するため、毎回の移動による誤差も防ぐことが出来ると思われる。カメラに傾きをつけても、環境点は傾き 0° の動作を続ける。

投影点は環境点の位置から算出されて、環境点の座標は保持したまま投影点を画面上に投影する。カメラの傾きを与えられると、図 4 に示すように、環境点の位置はそのまま、カメラの傾きの分だけ斜めになった状態に配置される。

3.1.6. 旋回時の投影点

ハンドルを切ったときも環境点はそのままの状態で動作する。タイヤが切れると切れ角に応じた回転半径、回転の中心位置が計算される。この計算方法は次項で説明する。この値と環境点を元に、投影点はカーブの位置に

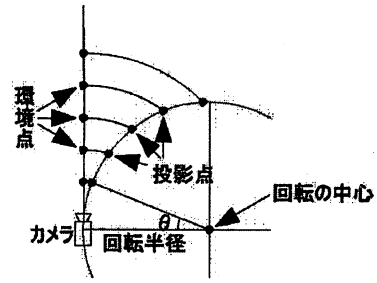

図5　カーブ時の投影点の配置

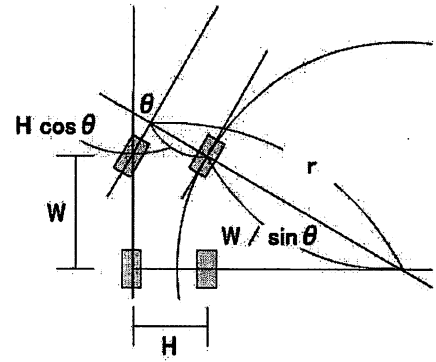

図6　回転半径の計算

配置される。投影点の配置の模式図を図 5 に示す。このときの点の配置の計算については後述する。

3.1.7. 回転半径の算出

車両の回転半径は以下の式で求めることができる。

$$r = H\cos\theta + \frac{W}{\sin\theta}$$

なお、この式で r は回転半径、H はトレッド(左右の車輪の間隔)、W はホイールベース(前輪と後輪の間隔)、そして θ はタイヤの切れ角とする。この式を示したものが図 6 である。

3.1.8. カーブ時の投影点の配置

まずカメラから環境点までのｙ軸方向の距離を弧の長さに対応させて、中心角が何度になるかを計算する。回転の中心からカメラまでの距離を半径とする円で、カメラの位置から、計算で求めた中心角だけ移動した点が、環境点に対応する投影点の座標となる。これを図７に示す。図中の記号でｓはカメラから環境点までの距離、ｒは回転半径、θは中心角、ｐは環境点の位置、p'は投影点の位置を表す。

この図より中心角の値は次の式で求めることができることがわかる。

$$\theta = \frac{s}{2\pi r} * 2\pi$$

$$= \frac{s}{r}$$

3.1.9. 規範フローの生成

投影点のカメラ画像上の座標は前状態を保存してあり、前状態をフローの始点、現在の座標位置をフローの終点として描画することでベクトルを生成する。ただし、一番奥に再配置されるときには、その特徴点に対してのベクトル描画は行われない。

3.2. 入力画像中の特徴点抽出とフロー検出

オプティカルフローの検出は、連続する２枚の入力画像の対応点を見つけることにより行われる。まず、１枚目の画像に対して Good Features to Track[4]に基づき特徴点抽出を行う。

そして２枚目の画像との対応点を見つけてオプティカルフローを検出する。この対応点の決定方法を、本システムではオープンライブラリを利用して、比較的実装の容易であったKLT法[5]を用いて入力画像のオプティカルフローを検出した。

3.3. 規範フローと実フローの照合

3.3.1. 規範フロー場の位置補正

生成された規範フローと実際の入力画像から検出したフローを照合する際に、その位置のズレを補正する必要がある。そこで実際の入力画像から検出したフローの情報から FOE を算出し、生成プログラム中の FOE の位置を合わせることによってズレを補正することにした。

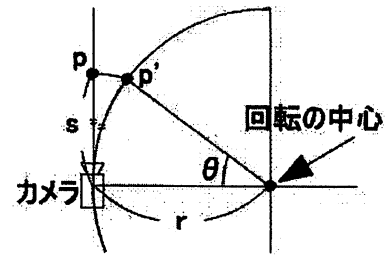

図７　カーブ時の投影点移動のための中心角計算

3.3.2. 画面の分割と規範フローの設定

本システムでは各場所における規範フローと、入力画像のオプティカルフローを比較して障害物の検出を行うため、場所と規範フローの関連付けを行う必要がある。

しかし規範フローの生成を密にしすぎると計算に時間がかかってしまいリアルタイム処理の性能を落としてしまう可能性がある。

また、規範フローは生成プログラムによって画面全体に生成されるが、実フローはテクスチャに依存して特徴の弱いところでは検出されないといった密度の違いが生まれてしまう。

そこで画面をブロックに分割し、ブロックごとに代表となる規範フローを設定する。代表となる規範フローの設定については、次項で詳しく説明する。この手法により、場所と規範フローの関連付けと、処理の効率化を両立した。また、１対１のベクトル比較ではなく、領域での基準との比較により、集中して検出されたベクトルもすべて比較対象を持つことが出来るようになる。

3.3.3. 代表となる規範フローの設定

直進状態において、通常、路面よりも高さのある静止障害物は、車両に設置されたカメラからの距離が、路面に比べて近くなるため、画面内における移動量は大きくなる。規範フローは路面上に出るフローを計算により求めたものであるから、障害物のフローは規範フローよりもフロー長が大きくなるはずである。よって規範フローよりも大きなフローを障害物のフローであると考えることにする。

そうしたとき、各ブロック内で複数の規範フローが含まれる場合、一番大きなベクトルを持つ規範フローを代表にするのが好ましいと考えられる。なぜなら、もしも

小さいベクトルを選択した場合、それよりも大きな規範フローがブロック内に存在するということになり、障害物のない路面のフローでも、代表のベクトルを超えてしまうことがあるからである。

道路平面上に現れる規範フローは、局所領域では、ほぼ同じベクトルになると考えられ、近隣領域では滑らかな変化をしていると考えられる。このことから、ブロック内に規範フローが含まれない場合、近傍のブロックからベクトルの値をとることにした。

3.4. 障害物の検出

3.4.1. 障害物フローの判断

入力画像のオプティカルフローすべてについて、そのフローの存在するブロックの代表規範フローの値と比較する。このとき規範フローと比べて、縦成分と横成分の向きが同じで、ベクトル長の長いものを障害物のフローとして検出する。

3.4.2. 障害物領域の検出と告知

このまま障害物のフローを含むブロックを障害物領域だと判断すると、ノイズも障害物のフローだと判断してしまう場合が考えられる。

特徴点はテクスチャのある障害物領域に集中して抽出される。すなわち障害物を含むブロック内に多数の障害物のフローが含まれると考えられる。これに対してノイズの場合は、ある領域に集中して出るわけではないからブロック内には障害物のフローだと判断されたノイズが少数含まれると考えられる。

このことからブロック内に含まれる障害物のフローの数に対して閾値処理をほどこし、ブロック内の障害物フローの数が閾値を越えた場合のみ、障害物が含まれると判断する。障害物が含まれると判断されたブロックは、障害物領域としてブロックの色を変えて運転者に注意を促す。

4. 実験結果

4.1. 実験条件

前章で述べた方式に基づき移動視点からの静止障害物検出システムを汎用パソコン上のソフトウェアとして試作し、提案手法の妥当性を確認する実験を行った。実験に使用したパソコンの CPU は Intel Xeon processor 3.6GHz、Memory は 2.0GB、OS は Microsoft Windows XP Professional、プログラミング言語は Microsoft 社製 Visual C++ .NET である。パソコンへの動画像の取り込みには Matrox 社製 Meteor II キャプチャーボードを用い、処理画像の解像度は VGA(640×480) で行い、カメラは SONY 社製の DCR-TRV30 を使用した。

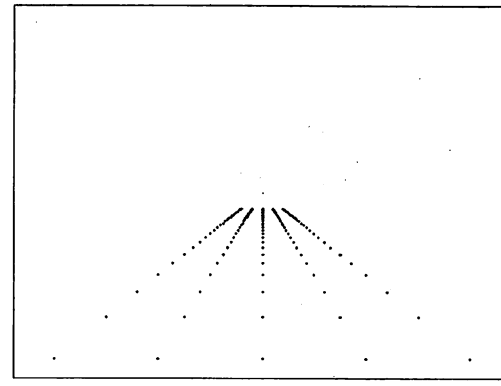

図 8　傾きのない状態の画面例

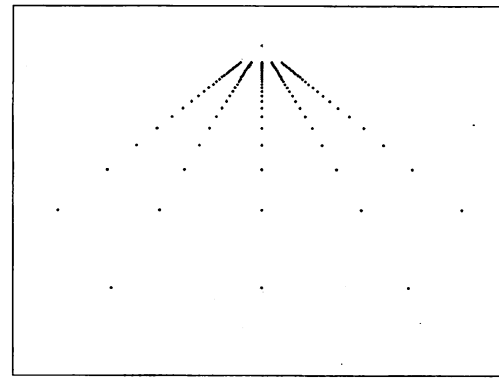

図 9　傾きのある状態の画面例

4.2. 規範フロー生成実験

規範フロー生成プログラムに、初期パラメータとして、カメラの高さを 120cm、道路の幅を 3m、特徴点の数を 150、特徴点の間隔を 2m、特徴点を配置する列の数を 5 列、1/3 インチ CCD カメラの焦点距離を 8mm と設定し、実験を行った。

4.2.1. 傾きを変化させたときの表示実験

カメラの傾きをつけない状態を設定して特徴点を配置した画面を図 8 に示す。正面の遠方が画面の中心に来ていてカメラが水平に向いている状態であることがわ

かる。

カメラを水平から 10 度下向きに傾けた状態を設定して特徴点を配置した画像を図 9 に示す。図 8 と比べて正面遠方が画面の中心よりも上にずれていて、カメラが地面のほうを向いて傾いていることがわかる。

4.2.2. カーブの投影実験

初期パラメータとして実際の車のカタログを参照し、トレッドを 1470mm、ホイールベースを 2805mm に設定し、カメラを 10 度下向きに傾けて、ハンドルを右に 27 度切った場合の特徴点の配置を図 10 に示す。自分の進行方向が画面奥で右に曲がっていることが確認できる。

4.2.3. 規範フロー表示実験

スピードの値を変化させてフローベクトルの表示実験を行った。スピードを上げるにつれベクトル長は長くなり、ベクトルの方向に変化は無いことが確認できた。図 11 はカメラの傾きを水平から 10 度下向きにし、時速 20km のときの規範フローの様子を示している。

4.3. 実フロー検出実験

処理速度を考慮して特徴点の数を最大 400 とした。実際の車両への搭載が出来なかったため、屋外で考えられる要素を屋内に模擬環境として設定し、実験を行った。ここで屋外において考えられる要素として、テクスチャの無い路面、道路標示のような路面のテクスチャとしてビニールテープを貼る事で再現、テクスチャのある背景として机の脚で再現した。カメラは床面からの高さ 33cm の位置に、水平から約 9 度下に向けて設置し、約秒速 80cm で直進させた。

このときのフロー検出例を図 12 に示す。画面内に表示されている細い線が検出されたフローで、背景や路面のテクスチャ部分に検出されていることがわかる。

4.4. 障害物検出実験

室内に構築した模擬環境用にパラメータの値を、カメラの高さ 33cm、道路の幅を 1m、特徴点の数を 60、特徴点の間隔を 60cm、特徴点を配置する列の数を 3 列、カメラの傾きを水平から下向きに 9 度に設定した。ブロックはあまり小さく分けるとフローがブロックをまたいでしまい、大きくしすぎると障害物の領域を余分に表

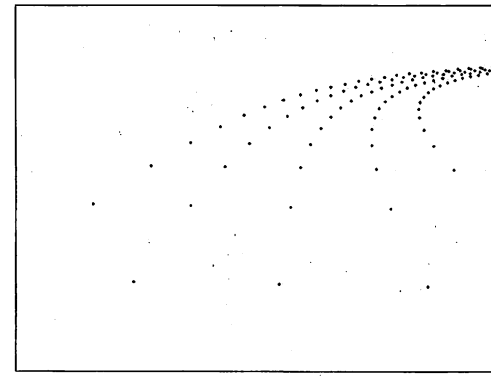

図 10　ハンドルを切った場合の画面の例

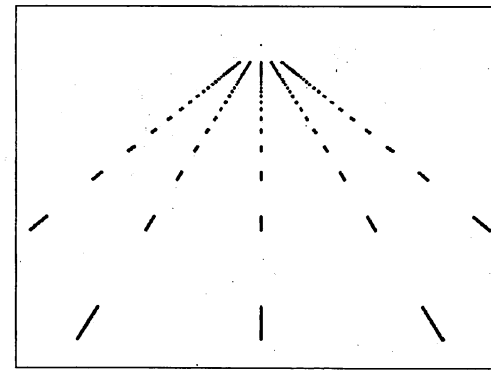

図 11　規範フローベクトルの表示例

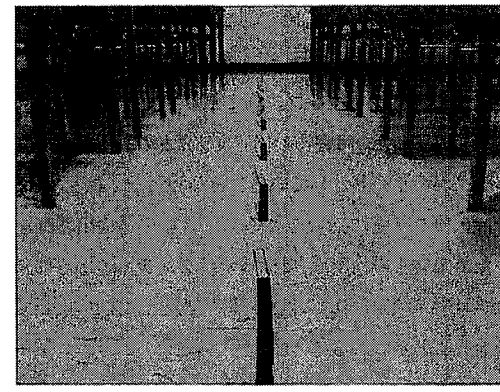

図 12　フロー検出例

示してしまう。そこで今回の実験における移動状態をもとに、生成プログラムで生成される規範フローのフロー長最大のものが入る大きさとして、縦に 6 等分、横に 8 等分して実験を行った。模擬環境上に障害物に見立てた箱を設置し、カメラを移動させたときの障害物検出実験を行った。

実際の検出結果を図 13 に示す。右側に、入力画像に

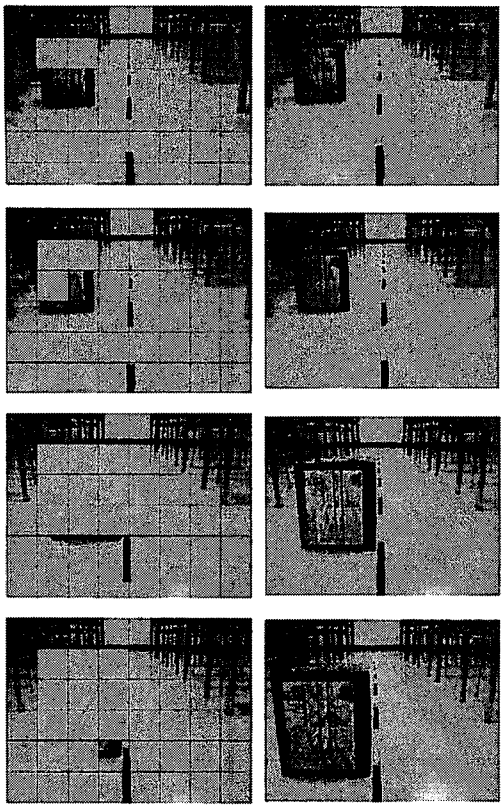

図 13　障害物検出結果

対してオプティカルフローを抽出した画面を表示し、左側にそのオプティカルフローと規範フローを照合した結果を表示した。障害物の位置で、高さのある部分のブロックが塗りつぶされていることがわかる。また、路面ではテクスチャの無いところにはフローが検出されないため障害物の検出はされない。テクスチャのある部分はフローが検出されているが、障害物として誤検出していないことがわかる。

5. 考察

障害物の検出実験の結果では、規範フローを用いた検出手法により、移動視点から静止障害物を検出することが確認できた。模擬環境ではあったがテクスチャのはっきり出る複雑背景や、道路標示などの路面テクスチャのある環境として構築しているため、車に搭載することが出来れば実シーンでの使用も可能であると考えられる。

障害物の存在しない路面のブロックでも、障害物を検出した表示が出ることがあった。これはブロックの切り方や、ブロック中の代表ベクトルの設定によるものだと考えられ、場所によってブロックのきり方を変えるなどの対策により改善が見込まれる。

6. おわりに

規範となるフローを設定し、実際に移動視点からの画像処理によって得られたオプティカルフローと比較することで、静止障害物を検出するという新たな手法を提案し、模擬環境での実験によりその有効性を示した。

今後は規範フローの設定の方法や、照合の仕方などを改良し、シミュレーションできる状態や環境を増やすことにより、さらなる改良を進める予定である。

謝辞　本研究の一部は株式会社デンソーアイティーラボラトリの援助による。ここに深謝します。

参考文献

[1]荒蒔勲,佐生徳実,梅田和昇, "小型距離画像センサを用いた移動ロボットの障害物回避手法の提案", 日本機械学会ロボティクス・メカトロニクス講演会'01 講演論文集,1P1-N4, 2001.6.

[2]岡田慧,加賀美聡,稲葉雅幸,井上博允, "色領域分割と両眼ステレオの統合による脚型ロボットの三次元障害物回避" 第 16 回ロボット学会学術講演会予稿集, pp 1505-1506, 1998.

[3] Mutsumi Watanabe, Nobuyuki Takeda, Kazunori Onoguchi, "Moving obstacle detection and recognition by optical flow pattern analysis for mobile robots", Advanced Robotics, Vol.12, No.7,8, pp.791-816, 1999.

[4]Jianbo Shi, Carlo Tomasi: "Good Features to Track", 1994 IEEE Conference on Computer Vision and Pattern Recognition(CVPR'94), pp.593-600, 1994.

[5]Jean-Yves Bouguet:"Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm",OpenCV Documentation, Microprocessor Research Labs, Intel Corporation, 1999.