卒業論文

題目

SIMDプロセッサに適した車載キャプチャ画像変換法の研究

指導教員

近藤 利夫 教授

2010年

三重大学 工学部 情報工学科
計算機アーキテクチャ研究室

水野　貴誠 (405851)

内容梗概

近年,ITS(高度道路交通システム）の進展の中で, 車に搭載したカメラで周囲の道路状況を認識する, 安全運転支援用のステレオ画像処理システムの研究が盛んに行われている. 現在距離計測のための車載ステレオ画像処理ではキャプチャ画像の変換に射影変換が精度の高さや処理の容易さという点から利用されている. しかし, 不連続なメモリアクセスが必要となるため,SIMD 型の並列演算による高速化が困難である. 本研究ではこの射影変換に代わる並列アクセスに適したキャプチャ画像変換法を明らかにし, その有用性を示すことを目指す.

今回は射影変換と提案手法である斜交軸変換での距離精度の比較を行った. 距離算出精度においては, 射影変換の誤差が平均約４％に対し, 斜交軸変換では平均約６％が確認された. ４組の画像しか評価しなかったにも関わらず, 平均で２％ほどの差をつけられている結果となった. そのため斜交軸変換で射影変換をそのまま置換できるとの結論はえられなかった.

Abstract

Recently, in the development of ITS(intelligent transport system), the research on Stereo Image Processing System to support safe driving by recognizing the road conditions is very active. The homography is used for measuring headway distances because of its high accuracy and easiness. But it needs discontinuous memory accesses, so it's difficult to speed up by parallel arithmetic like SIMD. Therefore, the objective of this research is to find out the car image transformation which adapts to the parallel access that takes the place of homography. And to verify the validity of the system.

We compared the headway distance accuracy calculated by two ways, oblique axis transformation, which we propose, and homography. In this experiment, homography had error margin of 4% on the average, on the other hand, oblique axis had 6% on the average. Though we tried it on only four images, there are margin of 2% between two ways. As a result, we could not verify that homography of image transformation part can be replaced with oblique axis transformation.

目 次

図 目 次

表 目 次

1　はじめに

1.1　背景

近年,ITS(高度道路交通システム）の進展の中で, 車に搭載したカメラで周囲の道路状況を認識する, 安全運転支援用の車載画像処理システムの研究が盛んに行われている. レーザーやミリ波レーダー等のセンサーによる前方検知システムは実用かされているが, 高コストであり, 電力消費を押さえるのが難しい. そこで将来的に高速な汎用プロセッサを用いてより安価に実現されることが予想される, 画像を用いた障害物検知システムの開発が行われている. このカメラベースの前方障害物検出システムにおいては, ２台のカメラを用いる方がステレオ画像処理により, 対象物体までの距離計測が可能になるため, カメラ１台のみを使用するシステムに比べ, 高性能なシステムの構築が可能となる. 現在距離計測のための車載ステレオ画像処理ではキャプチャ画像の変換に射影変換が精度の高さや処理の容易さという面から利用されている. しかし, 不連続なメモリアクセスが必要となるため, SIMD 型の並列演算による高速化が困難であるという弱点がある.

1.2 研究の目的

本研究ではこの射影変換に代わる並列アクセスに適したキャプチャ画像変換法を明らかにし, その有用性を示すことを目指す. 具体的には並列アクセスによって高速化の期待が持てる画像変換法として斜交軸変換の適用性を明らかにする. 斜交軸変換は同一水平線上, 垂直線上のラスタ走査によって実現可能であり, 隣接画素の一括したメモリアクセスが可能であり, 1 画素あたりのメモリ読み出し, 書き込み時間の短縮につながるからである.

2　ステレオ画像処理による距離計測

2.1　ステレオ画像

ステレオ画像処理とは同一の対象物を異なる２つの視点から観測し, 両撮影面上にある対象物の位置のずれによって, その対象物の位置を測る方法である. 距離計測の前提として, 前方障害物を検知する必要があり, それは道路平面上に存在する高さを持った物体の画像変換した際に生ずるずれを認識することにより行われる. 距離計測はその画像を元に行う. また今回利用するのは平面投影ステレオ法と呼ばれる画像変換である. これは左画像中に存在するすべての点が道路面と同じ高さを持つと仮定し, 左画像を右画像へ逆投影する. この逆投影画像と実際の右画像との間の差を求め, その差分値が大きい領域を障害物であると判断する手法である.

2.2　ステレオ画像処理による距離計測

ステレオ画像処理による前方障害物の距離検出の手順は

画像入力 → 画像変換 → 視差算出 → 距離算出

であり, この中で最も処理時間を必要とするのは差分絶対値和（SAD;Sum of Absolute Differences）を利用した視差検出である. 一般的にこの処理の演算量を削減するために前処理としてステレオ画像の補正処理を行う. ス

テレオ画像は通常, 探索ラインが走査線と一致するのが理想であるが, 実際に二つのカメラをそのように配置するのは事実上不可能である. この画像処理は入力画像の探索ラインを同一水平線上に乗るよう補正するもので, ステレオ画像の平行化と呼ばれ幾何学変換を施すことによって実現される.

2.3 射影変換による画像変換とその問題点

射影変換とは幾何学変換の一つであり, ユークリッド幾何学的な線形変換と平行移動の組み合わせによる図形や形状の移動, 変換方式である. 幾何学的性質が保たれるので高精度な距離算出が可能である. 特に, 平面物体とそれを任意の位置から撮影した画像との対応に適している. あらかじめ変換元と変換後の対応点が 4 組があれば, 次の行列式よりその座標値から一意の変換パラメータを算出することが出来る.

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -X_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -X_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -X_3x_3 & -X_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -X_4x_4 & -X_4y_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -Y_1x_1 & -Y_1y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -Y_2x_2 & -Y_2y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -Y_3x_3 & -Y_3y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -Y_4x_4 & -Y_4y_4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix}$$

※ (X_i,Y_i) は変換元の座標,(x_i,y_i) は変換後の座標

この行列式によって求められた $a_1,\ldots,a_8$ のパラメータを以下の式 (1)(2)

に代入することによりアドレス計算を行う.

$$u = \frac{ia_1 + ja_2 + a_3}{ia_7 + ja_8 + 1} \tag{1}$$

$$v = \frac{ia_4 + ja_5 + a_6}{ia_7 + ja_8 + 1} \tag{2}$$

※ (i,j) は変換前のアドレス,(u,v) は変換後のアドレス

車載ステレオ画像処理の場合, カメラのキャリブレーションから変換パラメータをあらかじめ求めておくことができるが, これらの式の示す用に, 毎回のアドレス計算による不連続なメモリアクセスが必要となるため処理上のボトルネックとなっている. しかし, 画像に対しランダム走査を行ってしまうので,SIMD（Simple Introduction Multiple Data) 等の並列アクセスへの適用は難しく高速化が困難であるとされている.

3　斜交軸変換による画像変換法の提案

3.1　斜交軸変換

幾何学変換処理として SIMD 並列処理に適した斜交軸変換を提案する．斜交軸変換とは, x 軸, y 軸を求めたパラメータを元に画像を傾ける変換とする. 変換後の画像の傾きは変換元と変換後の２組によって求められる. 以下変換元の座標を (x_i,y_i), 変換後の座標 (X_i,Y_i) とする.X 座標の傾き A, y 座標の傾き B とすると A,B は以下の２式 (3)(4) によって求められる.

$$A = \frac{(x_1 - x_2) - (X_1 - X_2)}{Y_1 - Y_2} \tag{3}$$

$$B = \frac{(y_1 - y_2) - (Y_1 - Y_2)}{X_1 - X_2} \tag{4}$$

変換した入力画像と基準画像の対応点のずれを補正する必要があるため, 移動量の計算をする. 左右のずれを δx, 上下のずれを δy とする. ２つのパラメータは次の２つの式 (5)(6) によって算出される.

$$\delta x = A(height - y_i) - (X_i - x_i) \tag{5}$$

$$\delta y = (Y_i - y_i) - BX_i \tag{6}$$

上記のパラメータ（A,B,δx,δy) を適用し, ステレオ画像（左画像）の変換を試みた例を図 3.13.2 に示す.

図 3.1: 左カメラの撮影画像

図 3.2: 左カメラの変換画像

3.1.1　x 軸方向への傾き

ここでは式 (3) の説明をする.A とは, 左画像の任意の２点を結ぶ直線m と右画像の対応する２点を結ぶ直線ｎの同じ高さでのｘ座標の移動量を表す. 実際は図 (3.6) のように綺麗に２直線が並ぶことはないが, ２直線がどの位置に存在していても「ｘ軸方向への増加量」とその差は計算が可能であることを利用している.

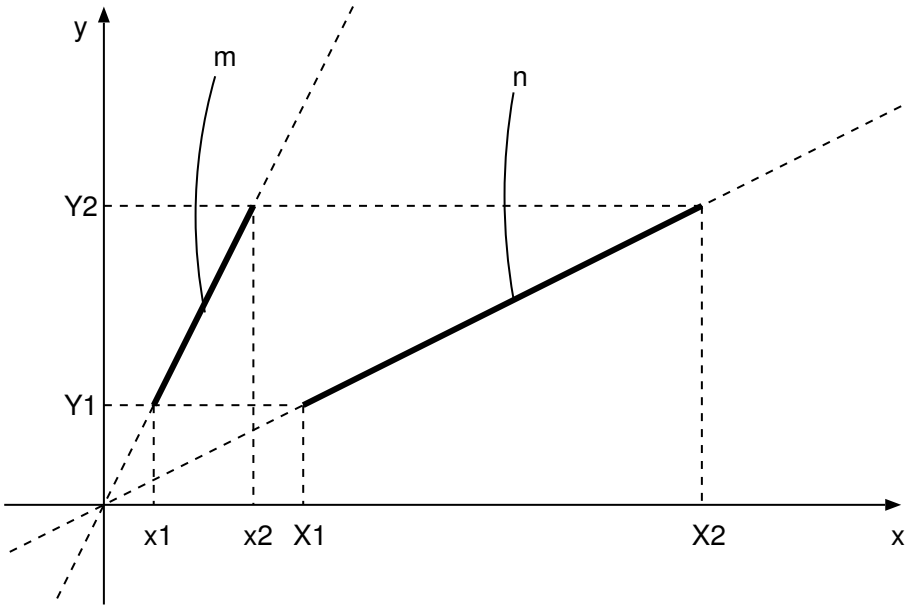

図 3.3: 水平方向への傾き

直線m上にある２点を $(x_1,Y_1)(x_2,Y_2)$, 直線ｎ上にある２点を $(X_1,Y_1)(X_2,Y_2)$ をとる. すると２直線は

$$m: \quad x = \frac{(X_1 - X_2)}{(Y_2 - Y_1)}y \qquad n: \quad x = \frac{(x_1 - x_2)}{(Y_2 - Y_1)}y \tag{7}$$

と表せる. これより n 上の点 $(X_i,\, y_j)$ と m 上の点 $(x_i,\, y_j)$ の関係は

$$n-m: \quad X_i - x_i = \frac{(X_1 - X_2) - (x_1 - x_2)}{(Y_2 - Y_1)} y_j \tag{8}$$

式 (3) を代入すると

$$X_i = A y_j + x_i \tag{9}$$

式（9) は, 直線ｎのｘの増加量は直線mのｘの増加量にA を加えることで表せることを示す.

3.1.2　y 軸方向への傾き

式 (4) も同様にして求めることができる.B とは Y 座標の移動量を表す. 式（3) の時はｙの増加量を固定したが, 今回はｘの増加量を固定することによって求める. 直線m上にある２点を $(X_1,y_1)(X_2,y_2)$, 直線ｎ上にある２点を $(X_1,Y_1)(X_2,Y_2)$ をとる. すると２直線は

$$m: \quad x = \frac{(y_1 - y_2)}{(X_2 - X_1)} y \qquad n: \quad x = \frac{(Y_1 - Y_2)}{(X_2 - X_1)} y \tag{10}$$

と表せる. これより n 上の点 $(X_i,\, Y_j)$ と m 上の点 $(X_i,\, Y_j)$ の関係は

$$n-m: \quad Y_j - y_j = \frac{(Y_1 - Y_2) - (y_1 - y_2)}{(X_2 - X_1)} x_i \tag{11}$$

式 (4) を代入すると

$$Y_i = B x_j + y_i \tag{12}$$

式（20) は, 直線ｎの y の増加量は直線mの y の増加量に B を加えることで表せることを示す.

3.1.3　y 軸方向への誤差補正

式 (6) の説明をする.δy とは左右の画像の対応点の y 座標の差である.

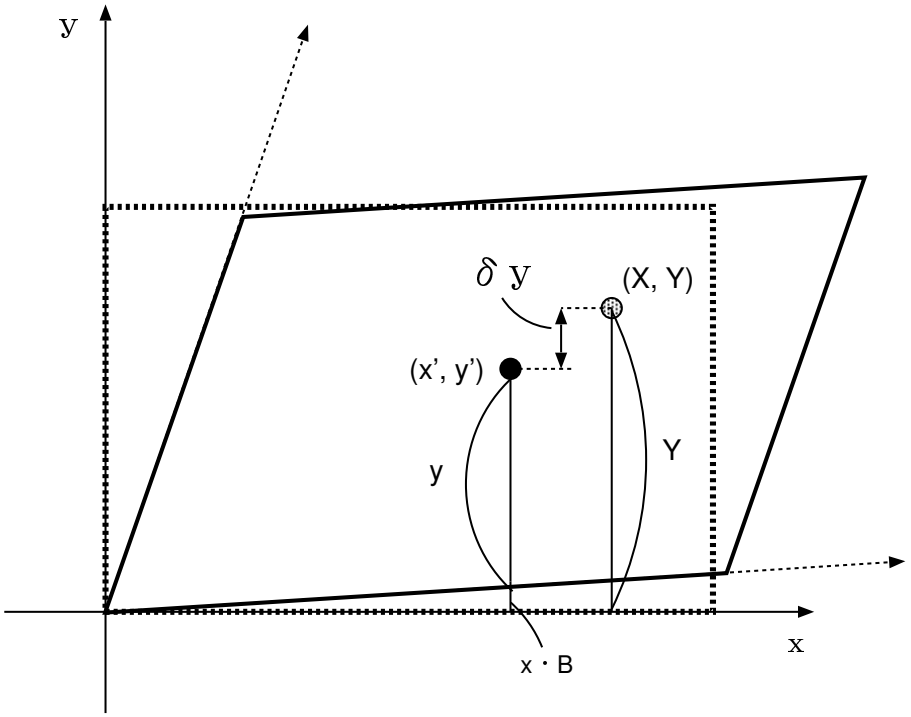

図 3.4: 垂直方向のずれ算出

図 (3.4) に示すように, 左画像の任意の点を (x,y), 画像に B の式を適応した後の左画像の点を (x', y'), 右画像の点を (X,Y) とすると

$$y' = y + Bx \tag{13}$$

$$Y = \delta y + y' \tag{14}$$

となり, 式 (13)(14) より δy が算出される.

3.1.4 x 軸方向への誤差補正

式 (5) の説明をする.δx とは左右の画像の対応点の x 座標の差である.

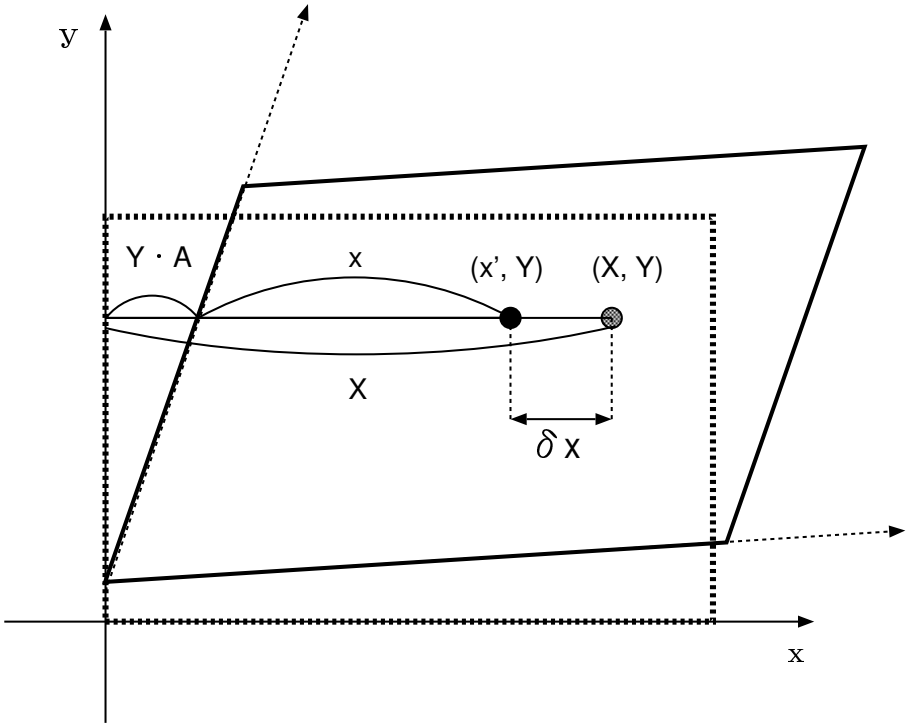

図 3.5: 水平方向のずれ算出

前章で Y 成分の位置を合わせた. 次は X 成分の位置を合わせる処理をする. 図 (3.5) に示すように. 左画像の任意の点を (x,y), 画像に B の式を適応した後の左画像の点を (x', Y), 右画像の点を (X,Y) とすると

$$x' = x + yA \tag{15}$$

$$X = \delta x + x' \tag{16}$$

となり, 式 (15)(16) より δx が算出される.

3.2 斜交軸変換による高速化

前述の通り, 射影変換では性質上, 画素単位でメモリを読み出しする必要があった. 本稿の斜交軸変換では, ラスタ走査可能な 2 つの斜交軸変換を組み合わせることで画像変換を行う.

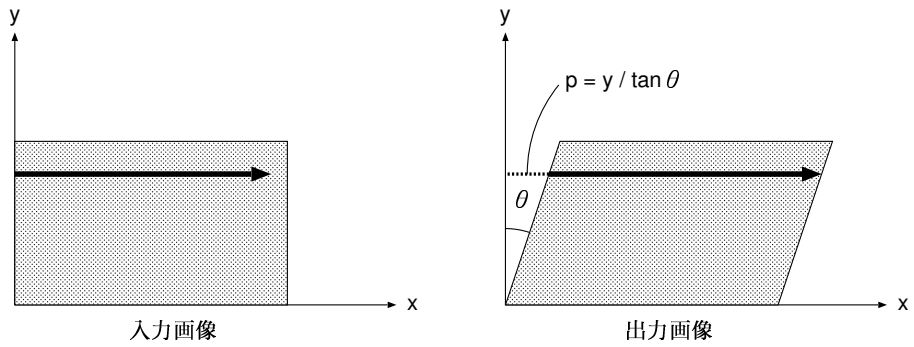

図 3.6: x 軸に平行な直線の変換

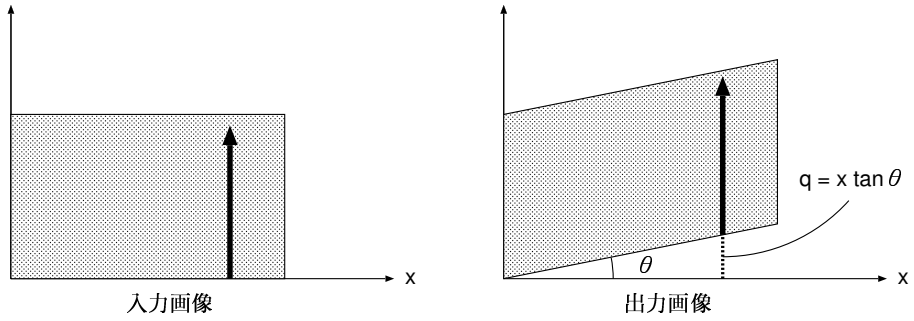

図 3.7: y 軸に平行な直線の変換

図 3.6,3.7 の様に, x 軸と平行な直線を水平方向に p 画素移動する変換

を行う. 同様に y 軸と平行な直線を垂直方向に q 画素移動する変換を行う. したがって, 入力画像も変換画像も水平方向への変換については複数の画素を一括してメモリアクセスすることが可能となる. 垂直方向に関してもメモリアクセスが容易になる. このようにデータを一括処理することが可能なことから, 並列処理を活かすことで, 画像変換において高速化が可能である.

4　距離算出

4.1　算出式の導出

ステレオ画像処理の簡単な原理は三角測量である. 対象物がカメラ間の中心線上にあるとする. 対象物と２台のカメラを線で結ぶと三角形が出来, これを利用して計測をする. 距離算出式の導出を以下に示す.

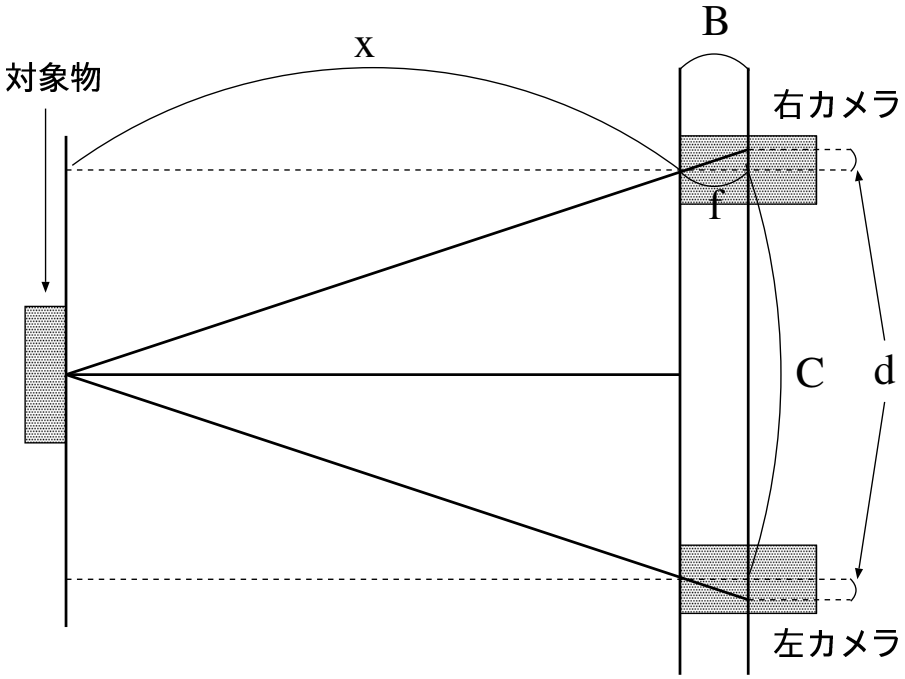

図 4.8: 三角測量

x：レンズから対象物までの距離　f：焦点距離　d：視差
C：カメラ間の距離　B：レンズから撮影面までの距離

$$\text{図 4.8 より } C : x = d : B, \quad \text{これより} \quad d = \frac{CB}{x} \tag{17}$$

$$\text{焦点の公式より} \frac{1}{x} + \frac{1}{B} = \frac{1}{f}, \quad \text{これより } B = \frac{fx}{x - f} \tag{18}$$

式 (17)(18) より $d = \dfrac{fC}{x-f}$,　これより $x - f = \dfrac{fC}{d}$　(19)

4.2　視差検出

左右の画像を比較するために対応点探索を行う. 平行化されたステレオ画像では, 対応点が同一水平線上に存在しているため画像すべての画素に対し探索を行う必要はない. 右画像の任意の点と同走査線に対し, 対応点を探索する.SAD による視差検出を行う予定であったが動作が不安定であり, あくまで画像変換の精度比較ということで実装はしていない. 今回は視差の検出は画像ソフトによる目視での確認となった.

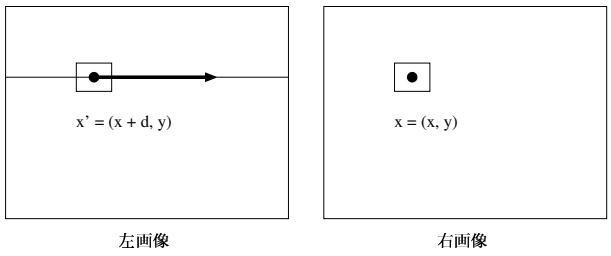

図 4.9: 左右画像の対応点探索

右画像を基準とし, 左画像で対応する点を探索する. 右画像上の点 $x = (x, y)$ の基準点に対し左画像の同一走査線上を探索し, 輝度パターンがもっとも類似した点を求める. 求めた点を x', 視差を d とすると,$x' = (x + d, y)$ で表すことが出来る.

5　評価

5.1　画像変換処理速度評価

今回は画像１枚に対しての処理であるため, 画像変換以外にかかる処理時間の割合が大きいと判断し, 画像変換の部分のみでの処理時間を比較した. 画像１０枚に対し１０回の処理時間の平均を求めた.

5.1.1　評価結果

斜交軸変換では射影変換の約７３％の処理時間で画像変換が可能であり, 高速化が可能であることが示せた.SIMD 命令を適用すれば, より処理時間は減少すると考えられる.

表 5.1: 処理時間

	射影変換	斜交軸変換	高速化倍率
処理時間	0.0450 sec	0.0325 sec	1.38 倍

※以下の環境で評価を行った

- Host: HP ProLiant ML150 G6
- CPU: Intel(R) Xeon(R) E5520 2.27GHz x 2
- Memory: 12GB
- OS: CentOS release 4.7 (64bit)
- Compiler: gcc 3.4.6

5.2 距離検出精度評価

カメラから対象物までの実距離を基準に射影変換, 斜交軸変換で求めた算出距離との誤差を求めた. 実距離で算出距離との差を割った数値を誤差率とする.

5.2.1 評価結果

結果として実距離に対して射影変換では平均約 4.3%, 斜交軸変換では約 6.2% の誤差が見られた.

表 5.2: 誤差

実距離 (m)	算出距離（誤差率）	
	射影変換	斜交軸変換
23.745	24.356(+2.57%)	24.356(+2.57%)
23.874	22.482(-5.83%)	22.482(-5.83%)
23.562	21.920(-6.97%)	21.266(-9.77%)
22.893	22.482(-1.80%)	21.385(-6.59%)

6　関連研究

6.1　差分絶対値和

差分絶対値和（SAD;Sum of Absolute Differences) とは同じ大きさの二枚の画像を比較するとき, 対応する各画素について二枚の画像間の差分を取り, その絶対値を合計したもの. この数値が小さいほど, 二つの画像は類似していることを示す.

二枚の画像間の m×n 画素探索ウィンドウの SAD は以下の式 (20) によって求められる.

$$SAD = \sum_{i=-m/2}^{m/2} \sum_{j=-n/2}^{n/2} (|A(x_i, y_j) - B(x_i, y_j)|) \tag{20}$$

6.2　SIMD 演算

SIMD(Single Instruction Mlutiple Data) とは一つの命令で複数のデータに対して処理を行う命令セットである. 汎用命令では加算等の命令を実行する際, 6.3 に示すようにソースおよびデスティネーションの２つのオペランドを取り, 命令を実行しその結果をデスティネーションオペランドに格納する. ２つのオペランドを必要とするが, 解は１つしか得られない. それに対し,SIMD 命令ではソース及びデスティネーションに複数のデータを与え, それを一つの命令でまとめて実行する.6.4 の場合一つの命令で

汎用レジスタ（Destination）
D
+
汎用レジスタ（Source）
S
↓
汎用レジスタ（Destination）
D + S

表 6.3: 汎用命令

８つの演算を実行し, デスティネーション格納している. 独立した複数のデータから求めた演算の解であり, 一つの命令で８つの解が求められたことになる.SIMD は単位時間あたりのデータ処理量を増加させるというアプローチで並列化を実現している.

SIMD レジスタ（Destination）							
D7	D6	D5	D4	D3	D2	D1	D0
+							
SIMD レジスタ（Source）							
S7	S6	S5	S4	S3	S2	S1	S0
↓							
SIMD レジスタ（Destination）							
D7+S7	D6+S6	D5+S5	D4+S4	D3+S3	D2+S2	D1+S1	D0+S0

表 6.4: SIMD 命令

7　まとめと今後の課題

7.1　まとめ

画像変換速度に関してはSIMD 演算等の適用により, 並列アクセスを活かすことで高速化を図ることが今後の課題となる.

距離算出精度では, 今回はカメラの中心からのｘ座標のずれの影響を無視したため視差に射影変換と斜交軸変換の両方にある程度誤差が出た. また斜交軸変換においては, 画像の拡大, 縮小に関して考慮しておらず, 適正な補正を加えていないため, 今後適用していく必要がある. シミュレーションでは, 斜交軸変換で射影変換をそのまま置換できるとの結論はえられなかった. ４組の画像しか評価しなかったにも関わらず, 平均で２％ほどの差をつけられているからである. 今後は, 処理量の少ない適切な補正により, 射影変換との間の視差検出語差を実用的に無視できる程度に抑えられるかどうかを明らかにしていく必要がある.

その他の課題としてあげられるのが, 視差算出部分である. この動作が不安定であったため, 自動での視差検出とまで到達することが出来なかった. ブロックマッチングの比較領域の変形等を行うことで精度の向上につながるのではないかと考える.

7.2　今後の課題

7.3　両画像に対する斜交軸変換

今回の研究で, 斜交軸変換と射影変換の横方向への伸縮による誤差がみられた. これにより基準となる画像との道路平面にもずれが生じていることを意味する. これを改善するために, 左右の両画像に対し斜交軸変換を試みることを提案する. 両画像を対照的に同じだけ傾けることにより, 横方向のずれが軽減されるのではないかと考える.

パラメータをそれぞれ半分にし, 右画像には負のパラメータを与えてやることで同じ変換を行う. この変換を施し, 左右を重ね合わせた画像を図 (7.10) に示す.

図 (??) と比べてみると, 道路平面において横幅のぶれ幅が減っていることが見て取れる. このように左画像のみに斜交軸変換を施した画像よりも, 横幅の誤差を低減させることが可能である. 視差検出, 演算量が倍になることの影響等の問題はあるものの, 距離算出精度の向上や SIMD 演算により高速化が見込めるという利点もあるため今後研究によってその有用性を明らかにしていく必要があると考える.

図 7.10: 両画像の斜交軸変換

7.4 横方向への視差補正

現在の視差計算では, すべての対象物が右カメラの直線上にあると仮定して計算を行っている. しかし, 実際にその直線上に対象物が存在することは少なく, そこで横方向への視差補正を考えて比較評価する必要がでてくる.

誤差は図 (7.11) に示す通りである. 実線部分ある対象物と利用する三角形

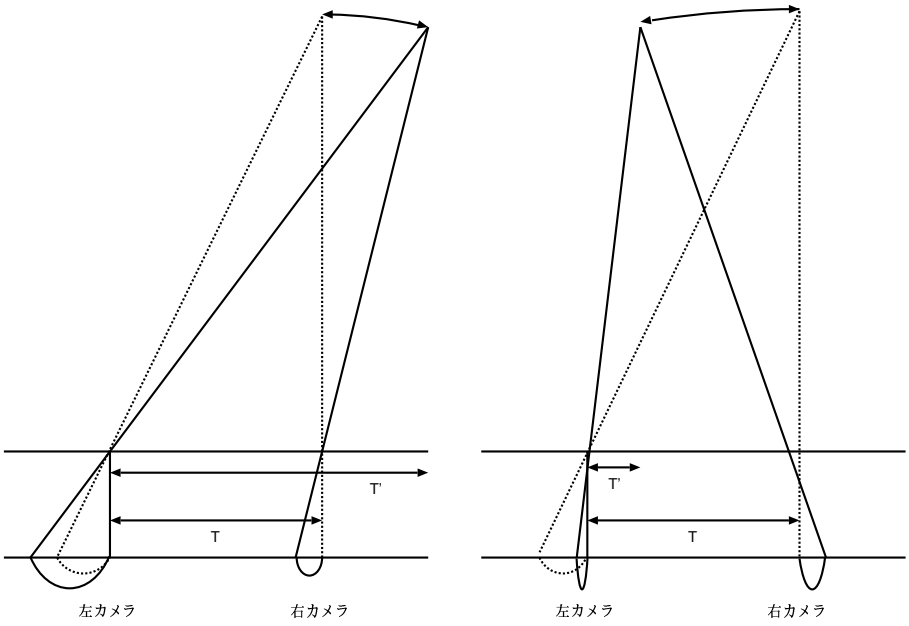

図 7.11: 横方向の視差補正

とその視差. 点線部分は距離をそのままに基準点に移動した対象物と利用する角形とその視差である. 視差を求める式 (19) からわかる用に, 視差が大きく異なることがわかる. この違いは右のカメラに生じている視差が影響している事が理由であると考えられる. そこで左カメラに生じる視差と, 右カメラに生じる視差の両方を考慮する必要があると考える. 実際の視差をｄ, 左カメラに写る視差をｄ’とすると

$$d' : d = T + T' : T \quad \text{これより} \quad d = \frac{T \times d'}{T'} \tag{21}$$

と求める事が出来る. しかし, 今回の研究ではこの式をそのまま利用してしまうと矛盾が生じてしまうという結果になった. 今後この理由を解明していく必要があると考える.

謝辞

本研究を行うにあたり, 御指導, 御助言頂きました近藤利夫教授, 並びに多くの助言を頂きました大野和彦講師, 佐々木敬泰助教授, に深く感謝いたします. また, 様々な局面にてお世話になりました研究室の皆様にも心より感謝致します.

参考文献

[1] 野口卓, 奥富正敏：“ステレオ画像からの道路平面に対する射影変換行列の導出 “, コンピュータビジョンとイメージメディア,108-4,1997 年 11 月 27 日.

[2] 田畑邦晃, 武田春夫, 町田哲夫 : “ラスタ走査とテーブル参照による画像回転の高速処理 “, 電子通信学会論文誌,Vol.J69-D,NO.1, pp.80-90, 1986 年.

[3] 服部寛 : “車載向けステレオ画像処理技術 “, 東芝レビュー,Vol.63,No.5,2008 年.

[4] 配島総一, 岩澤智也, 小島秀夫, 五ノ井武史：“ステレオ画像処理における距離測定 “.

[5] 中井宏章, 前田賢一 : “危険を察知する車載画像処理技術 “,IPSJ Magazine ,Vol.48,No.1,2007 年 1 月.

A　プログラム使用方法

A.1　siftDemoV4

siftDemoV4 は PGM 画像に対応した sift 特徴量に基づいた対応点探索用のソフトウェアである. プログラム使用の流れとしては sift→match となっている.

A.1.1　sift

この実行ファイルは PGM 画像ファイルの特徴量を求め,key ファイルとして出力する.

使用方法：`sift <filename.pgm >filename.key`

(filename) には sift 特徴量を求めたい画像の名前を入力, 実行が成功すると key ファイルが生成される. 次に生成される key ファイルの内容は以下に記す.

特徴点の数　`keypoint` の記述子ベクトルの長さ

特徴点 (1) のｘ座標　ｙ座標　-pi〜pi までのオリエンテーション

記述子ベクトルが１２８個

特徴点 (2) のｘ座標　ｙ座標　-pi〜pi までのオリエンテーション

記述子ベクトルが１２８個

特徴点 (3) のｘ座標　ｙ座標 …

の用に記述されている. 必要な情報がある場合は参考にするとよい. 以下

例を記す.

```
----------------------filename.key-------------------------------

3492 128

435.87 1088.03 105.17 1.768

 13 1 0 0 0 0 0 17 143 0 0 0 0 0 0 115 147 16 5 0

 0 1 43 147 55 14 3 0 0 5 53 55 44 28 3 0 0 0 0 8

 147 38 1 0 0 0 0 147 147 7 0 0 0 0 41 147 11 0 0 0

 0 0 27 55 18 32 15 8 1 0 0 0 147 147 5 0 0 0 0 15

 147 79 0 0 0 0 0 17 9 9 0 0 0 0 0 2 0 0 0 0

 0 0 0 0 32 8 0 0 0 0 0 0 30 8 0 0 0 0 0 0

 1 2 0 0 0 0 0 0

362.77 375.09 93.23 1.239

 9 3 0 0 0 0 0 0 0 36 2 0 0 0 3 12 29 2 0 0 0

  …

------------------------------------------------------------------
```

A.1.2 match

２枚のPGM画像と,siftで生成した２つのkeyファイルを利用して対応点をマッチングさせ, 対応点同士を線で結ぶ形で視覚化するソフト.

使用方法:`match -im1 a.pgm -k1 a.key -im2 b.pgm -k2 b.key > out.pgm`

a,bは入力画像, 入力keyファイルの名前.out.pgmにはaとｂを上下に並べ, 対応点を結んだ画像が描写される. 出力画像は以下の用になる.

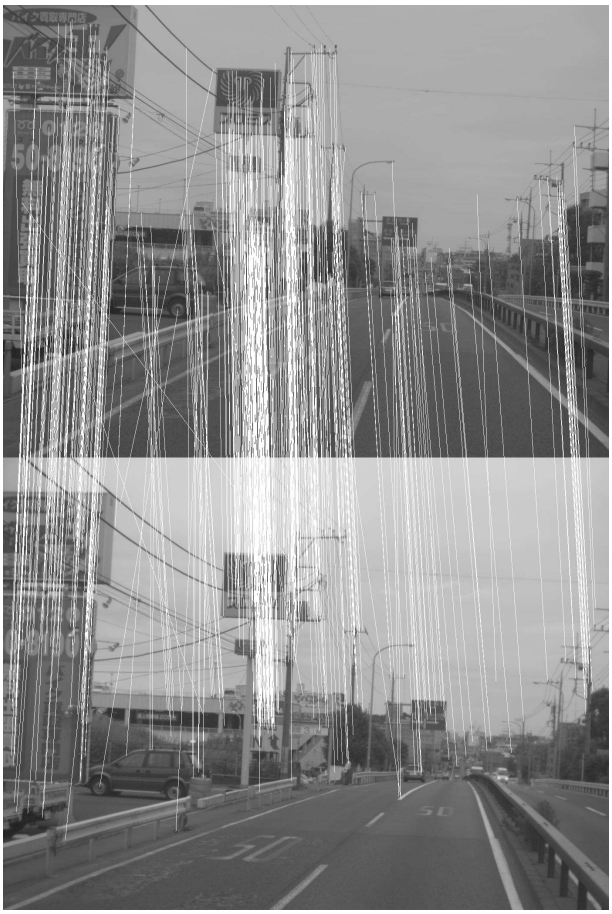

図 1.12: matchの使用例

A.2 octave

近藤研の (moule) にインストールされている算術ソフトウェア. 今回射影変換のパラメータ算出に必要な８連立方程式を octave によって解いた. moule1.arch.info.mie-u.ac.jp にログインして使用. 使用方法等は次ののHP を参考にした.

http://www.mlb.co.jp/linux/science/octave/

A.3 自作のプログラム説明

今回自作したプログラムが多いため, それぞれの使い方を簡単に説明する. プログラム本体と今回使用した左右の画像は $/home/mizuno/programs$ に存在する. ディレクトリ内で Makefile によりコンパイルすれば実行形式ファイルが生成される.

A.3.1 compare

siftDemoV4 の match に手を入れて作った対応点抽出プログラム. ２つの key ファイルを入力すると, それぞれ対応するｘ座標とｙ座標のみを抽出して key ファイルとして出力する.a.key と b.key を入力として与えた場合,match.key として対応点のデータを以下のように出力する.

----------match.key----------

(a の x1 座標)(a の y1 座標)(b の X1 座標)(b の Y1 座標)

(a の x2 座標)(a の y2 座標)(b の X2 座標)(b の Y2 座標)

...

A.3.2 axes

斜交軸変換プログラム. 対応画像形式は PGM である. 入力に左画像と compare で出力した対応点 key ファイルを元に斜交軸変換を行う. 出力ファイルとして入力ファイルに斜交軸変換を施した画像 re_axes.pgm が生成される.

A.3.3 projection

射影変換用プログラム.. 対応画像形式は PGM である. 入力に左画像と compare で出力した対応点 key ファイルを元に射影変換を行う. 出力ファイルとして入力ファイルに斜交軸変換を施した画像 re_projection.pgm が生成される.

A.3.4 sad

視差算出用プログラム. 左右の画像と右画像の基準となる点を入力し, 左画像でそれに対応する点を探す. 基準となる点の座標 (x,y) の地点から同一水平線上に右方向に探索し,SAD が最も低い点を対応点と決定する. 決定した後, 距離算出式に視差を代入し, 距離算出を行い出力する.

B 評価用データ

評価用データのステレオ画像は企業より購入したもので,CD に保存されている. データには左画像と右画像それぞれ JPG 形式で１５枚ずつ, 左右の画像を並べて対応点の位置を印した JPG 画像, 対象物までの算出距離が記される xls ファイルがある. 画像データのみプログラムと同様の $home/mizuno/programs$ のディレクトリの中の l と r にそれぞれ左画像, 右画像を置いてある.