卒業論文

題目

斜交軸変換を用いる
ステレオ画像の平行化

指導教員

近藤　利夫 教授

2018 年

三重大学 工学部 情報工学科
コンピュータアーキテクチャ研究室

藤田　礼彰 (412848)

内容梗概

近年，ITS（高度道路交通システム）が発展する中，自動運転の実現に向けた研究・開発が盛んに行われている．その成果の一つとして，障害物を検出，距離測定し，自動でブレーキをかけるシステムがすでに実用化されている．この距離測定に利用されているのがステレオ画像処理である．この自動化システムのコスト削減には，ステレオ画像処理の高効率化が必要不可欠である．しかし，キャプチャ画像を射影変換で整形するステレオ画像の平行化と呼ばれる処理が，その高効率化実現のネックになっている．射影変換が不連続なメモリアクセスを必要とするために，SIMD 命令による並列処理が機能しないからである．

これに対し，当研究室では，SIMD 命令に対応した斜交軸変換を用いる平行化のための整形法が提案され，射影変換を用いる手法と同等の距離検出精度が得られることが示されている．しかし，平行化のための対応点探索を目視に頼っており，設置状態の変化等に応じて行う必要のある，キャリブレーションの自動化に対応できていない．

そこで，本研究では，特徴点座標 2 組とエッジ座標の 2 つを用いることで，エッジ部で生じる開口問題を改善することと，変換パラメータに閾値を設けることの 2 つにより，画像の伸縮による誤差を改善する，斜交軸変換に適した対応点探索法を提案する．また，それを使って斜交軸変換を用いる平行化を行うことで，従来の射影変換による平行化並みの視差検出精度を得られることを示す．

Abstract

In recent years, as ITS (Intelligent Transportation System) has been developed, the related research and development for automatic operation is conducted vigorously. As one of the achievements, detects obstacles, measures distance, and automatically brakes has been applied. In above mentioned achievements, stereo image processing is used for distance measurement. High efficiency of stereo image processing is indispensable for cost reduction of this automated system. However, the processing which is called as the parallelization of stereo images that shapes capture images by projective transformation has become a bottleneck in realizing high efficiency of stereo image processing. The parallel processing based on the SIMD instruction does not work well because the projective transformation requires noncontiguous memory access.

Therefore, our laboratory proposed a shaping method for collimation by using oblique axis transformation corresponding to the SIMD instruction. Using this method, we obtains the same distance detection accuracy as that of the method using projective transformation. Since it relies on the visual inspection of the corresponding point search for collimation, the processing is performed according to the change of setting state.

Therefore, the automation for realizing calibration is difficult to realize. Based on mentioned reason, we propose a corresponding point search method which is suitable for oblique axis transformation to reduce errors. This method is based on using two pairs of feature point coordinates and two edge coordinates for solving the aperture problem occurring at the edge portion and setting a threshold value in the transform parameter for reducing the errors from the expansion and contraction of image.

Moreover, throughout the parallelization which is realized by oblique axis transformation, the proposed method provides the detection of a precision which is similar to the parallelization by the conventional projective 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）の進展する中，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障害物検知システムは実用化されているが，高コストであり，電力消費を抑えるのが難しい．そこで，将来的に高速な汎用プロセッサ用いて，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知システムにおいては，２台のカメラを用いるほうがステレオ画像処理により，対象物体までの距離測定精度が高いため，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カメラの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，この射影変換では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欠点がある．

1.2　研究の目的

本研究では，斜交軸変換と呼ばれるステレオ画像の平行化法を実用的にすることを目的とする．具体的には，先行研究で提案されていた斜交軸変換を用いた平行化において，対応点を 2 組選択する際，目視に頼っていた．理由としては，ステレオカメラの設置状態の変化等で必要となる，自動的なキャリブレーションに対応した対応点選択アルゴリズムが確立されていなかったからである．この自動化には，従来の射影変換とは異なる，斜行軸変換向きの対応点探索法を明らかにする必要がある．

2　ステレオ画像処理の概要

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，その対象物との距離を測る方法である．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（視差）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

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（平行化）→ 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．この処理をステレオ画像の平行化という．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 2 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．そのため，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，視差検出のための対応点探索の範囲が 1 ライン中に限定できるため，演算量の削減につながる．

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．

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

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

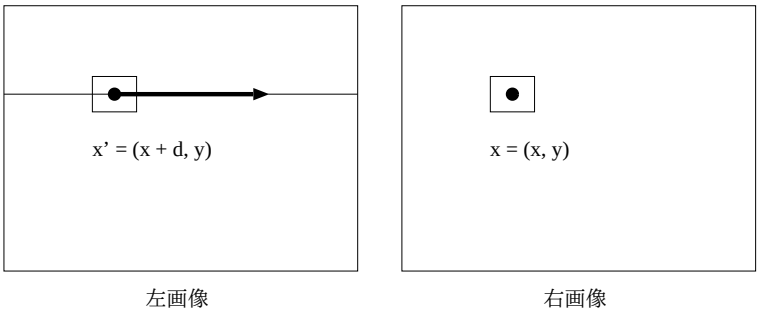
※ (i,j) は変換前のアドレス，(u,v) は変換後のアドレス

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．

2.4 視差検出

左右の画像から，視差検出を行う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，画像中の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． 右画像を基準とし，左画像で対応

図 2.1: 左右画像の視差検出



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，SAD により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．

2.5　距離算出式の導出

ステレオ画像処理の測距方法は三角測量である．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でき，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．

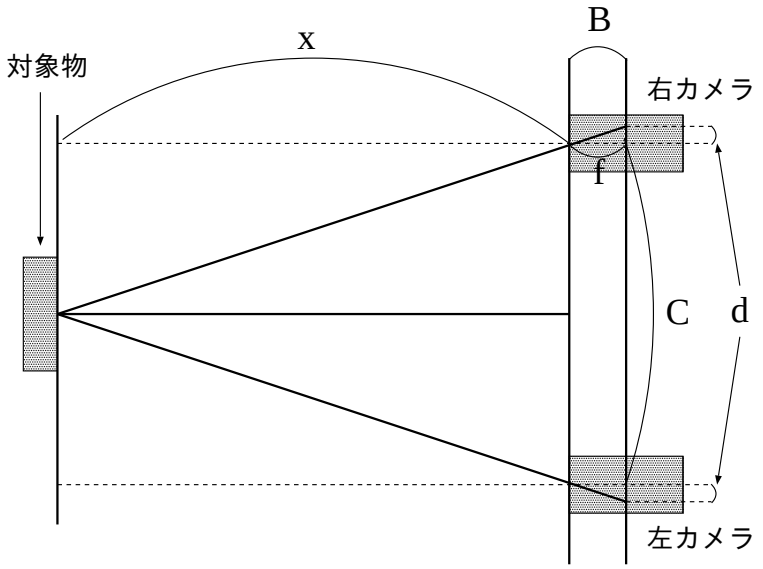


図 2.2: 三角測量

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

$$\text{図 2.2 より } C : x = d : B, \quad \text{これより} \quad d = \frac{CB}{x} \tag{3}$$

$$\text{焦点の公式より} \frac{1}{x} + \frac{1}{B} = \frac{1}{f}, \quad \text{これより } B = \frac{fx}{x - f} \tag{4}$$

$$\text{式 (3)(11) より } d = \frac{fC}{x - f}, \quad \text{これより } x - f = \frac{fC}{d} \tag{5}$$

3　斜交軸変換による高速化とその問題点

3.1　斜交軸変換

当研究室では，すでに SIMD 並列処理に適した幾何学変換処理として斜交軸変換が研究されている．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 (6)(7) によって求められる．

$$A = \frac{(x_2 - x_1) - (X_2 - X_1)}{Y_1 - Y_2} \tag{6}$$

$$B = \frac{(y_2 - y_1) - (Y_2 - Y_1)}{X_1 - X_2} \tag{7}$$

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 (8)(9) によって算出される．

$$\delta x = A(height - y_i) - (x_i - X_i) \tag{8}$$

$$\delta y = (Y_i - y_i) - BX_i \tag{9}$$

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．

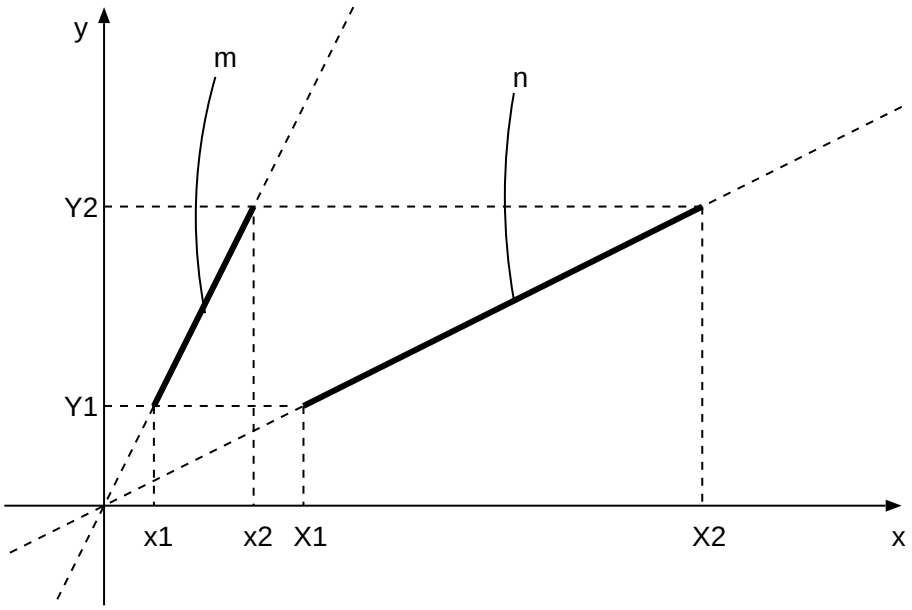


図 3.3: 水平方向への傾き

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

$$m: \quad x = \frac{(x_2 - x_1)}{(Y_2 - Y_1)}y \qquad n: \quad x = \frac{(X_2 - X_1)}{(Y_2 - Y_1)}y \tag{10}$$

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

$$n - m : \quad X_i - x_i = \frac{(X_2 - X_1) - (x_2 - x_1)}{(Y_2 - Y_1)} y_j \tag{11}$$

式 (6) を代入すると

$$X_i = A y_j + x_i \tag{12}$$

式（12) 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 (7) 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（(6) 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

$$m: \quad x = \frac{(y_2 - y_1)}{(X_2 - X_1)}y \qquad n: \quad x = \frac{(Y_2 - Y_1)}{(X_2 - X_1)}y \tag{13}$$

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

$$n - m: \quad Y_j - y_j = \frac{(Y_2 - Y_1) - (y_2 - y_1)}{(X_2 - X_1)}x_i \tag{14}$$

式 (7) を代入すると

$$Y_i = Bx_j + y_i \tag{15}$$

式（(20) は，直線 n の y の増加量が直線 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 (9) 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．

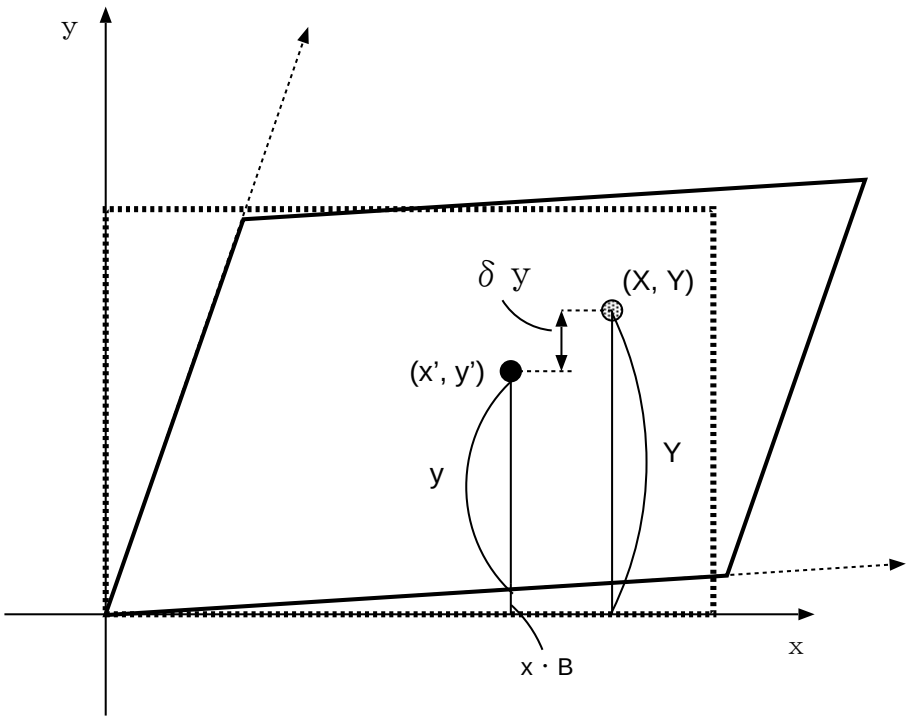


図 3.4: 垂直方向のずれ算出

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

$$y' = y + Bx \tag{16}$$

$$Y = \delta y + y' \tag{17}$$

となり，式 (16)(17) より δy が算出される．

3.1.4　x 軸方向への誤差補正

式 (8) 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．

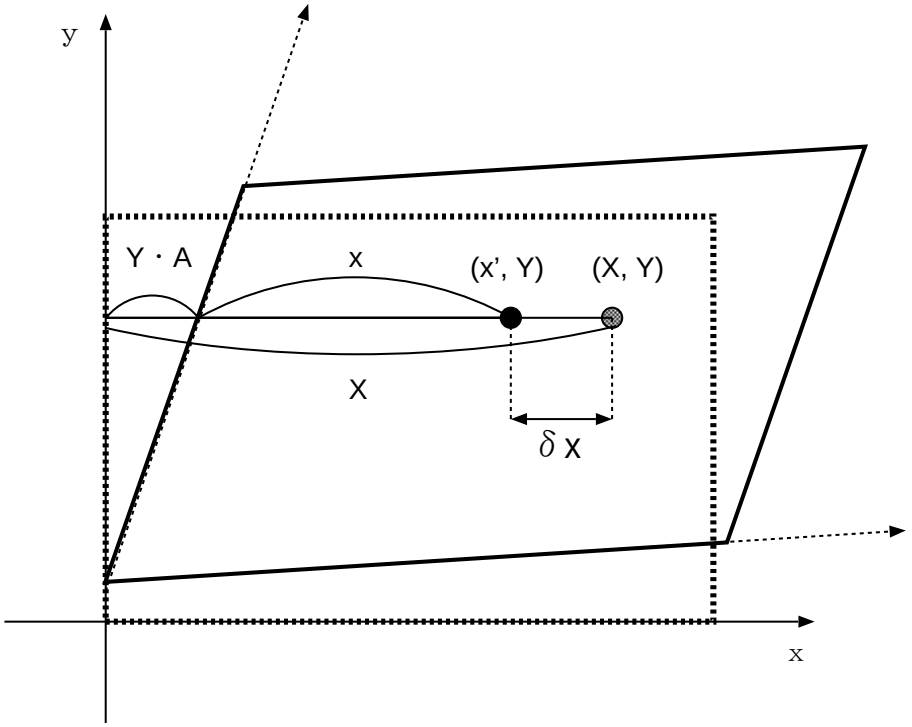


図 3.5: 水平方向のずれ算出

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

$$x' = x + yA \tag{18}$$

$$X = \delta x + x' \tag{19}$$

となり，式 (18)，(19) より δx が算出される．

3.2 SIMD 処理による高速化

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す必要があった．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．

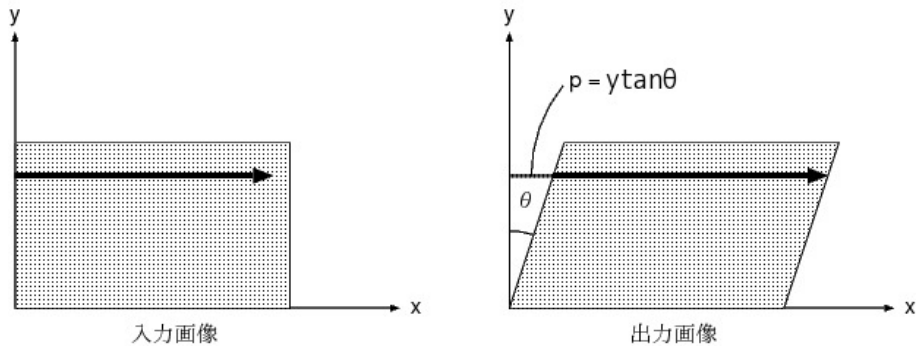


図 3.6: x 軸に平行な直線の変換

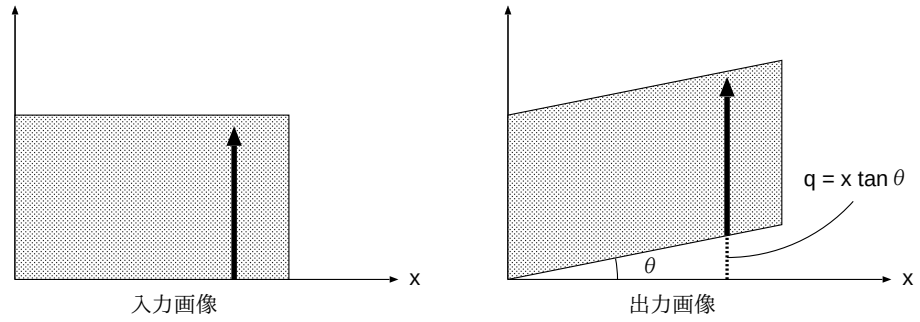


図 3.7: y 軸に平行な直線の変換

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，画像変換が高速化がされる．

3.3 斜交軸変換の問題点

この斜交軸変換は，x 座標の傾き A，y 座標の傾き B が大きくなると画像の伸縮が大きくなり，実距離との誤差が大きくなってしまう．そこで A，B の値を半分にし，左右の両画像に斜交軸変換を行うことで伸縮による誤差を低減している．しかし目視での対応点決定であったため，設置状態の変化等で必要になる自動的なキャリブレーションには対応できていなかった．この自動化には 4 組の対応点を用いる射影変換とは異なる，斜交軸変換向きの対応点探索法を明らかにする必要がある．この際，対応点検出ミスの発生を抑えるためのエッジ付近で生じる開口問題の改善や，最良の傾きとなる A，B を探索し，見つけ出すことが課題となる．

4　提案手法

ここでは，斜交軸変換に適する平行化に対応した開口問題の解消法と，最良の画像の傾きを求める手法の２つを提案する．まず開口問題については，特徴点座標，エッジ座標を出力させ，特徴点座標からエッジ座標を取り除くことで改善する．左右の特徴点を対応付けた画像を対応付けた特徴点座標の組，およびエッジ座標を一部抜粋したものを以下に示す．

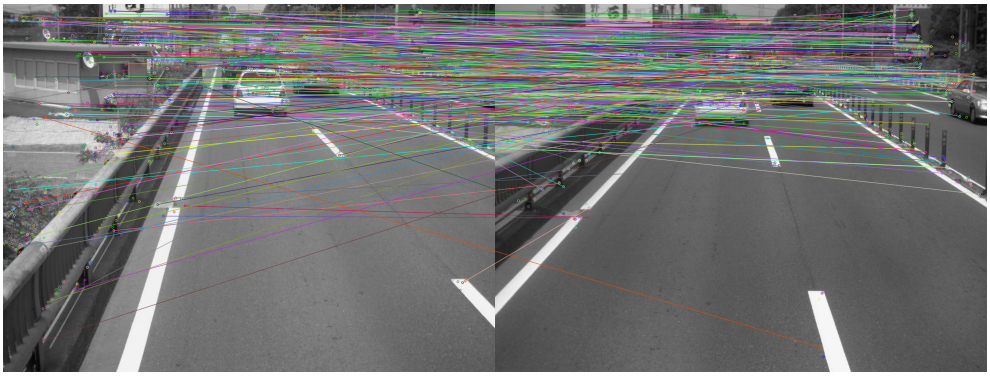


図 4.8: 特徴点マッチング

表 4.1: 出力座標

特徴点座標	エッジ座標
(112,691)(912,256)	(52,320)
(349,63)(324,88)	(53,320)
(744,145)(752,171)	(67,320)
(153,309)(111,334)	(68,320)
(410,140)(390,164)	(69,320)
(114,320)(923,971)	(114,320)
(1379,184)(1319,206)	(328,320)
(488,167)(501,159)	(657,782)
(651,75)(783,134)	(702,782)
(544,135)(918,159)	(703,782)
(691,243)(600,270)	(1107,782)
(127,101)(128,126)	(1186,782)
(1306,782)(266,575)	(1306,782)

マッチングした特徴点座標から，エッジ座標を取り除くことで開口問題を解消し，正しい対応点のみの出力が得られる．ソースコードから処理部分の一部を抜粋したコードを以下に示す．

```
/*smatch.cpp 画像から特徴点を検出*/

    std::vector<cv::KeyPoint> keypoints1;

    detector.detect(grayImg1, keypoints1);

    std::vector<cv::KeyPoint> keypoints2;

    detector.detect(grayImg2, keypoints2);
```

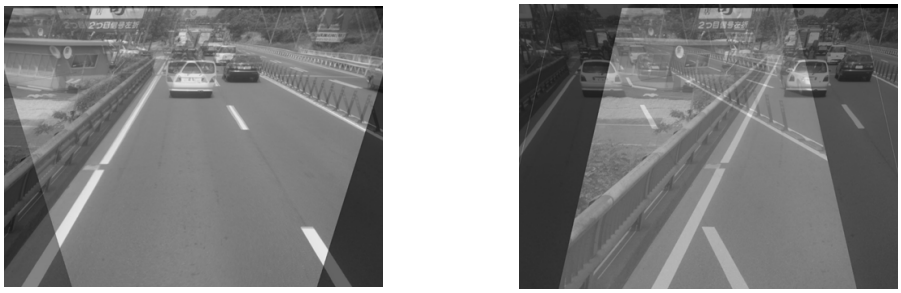
```
/*smatch.cpp 画像の特徴点における特徴量を抽出*/
    cv::Mat descriptors1;
    extractor.compute(grayImg1, keypoints1, descriptors1);
    cv::Mat descriptors2;
    extractor.compute(grayImg2, keypoints2, descriptors2);

/*smatch.cpp 特徴点マッチング*/
    cv::Ptr<cv::DescriptorMatcher>
    matcher = cv::DescriptorMatcher::create("BruteForce");
    std::vector<cv::DMatch> match, match12, match21;
    matcher->match(descriptors1, descriptors2, match12);
    matcher->match(descriptors2, descriptors1, match21);

/*smatch.cpp マッチした特徴点座標の組を csv ファイルに出力
  if 文で座標値が大きく異なる場合は出力させない条件付け*/
   for (size_t i = 0; i < match.size(); i++)
     {
      cv::DMatch forward = match[i];
```

```
      int query = forward.queryIdx;//左画像のマッチしたインデッ
クス番号
      int train = forward.trainIdx;//右画像のマッチしたインデッ
クス番号
      cv::KeyPoint kp1 = keypoints1[query];
      cv::KeyPoint kp2 = keypoints2[train];
      if(-200<(kp1.pt.y-kp2.pt.y) &&
        (kp1.pt.y-kp2.pt.y)<200 && -200<(kp1.pt.x-kp2.pt.x) &&
        (kp1.pt.x-kp2.pt.x)<200){
  fprintf(fp, "%d,%d,%d,%d\n",
                  (int)kp1.pt.x, (int)kp1.pt.y,
                  (int)kp2.pt.x, (int)kp2.pt.y);//lx,ly,rx,ry
}
      }
```

```
/*bout.c   マッチした特徴点座標からエッジ座標を取り除く*/

   while ( fgets(ereadline, N, efp) != NULL ) {

    len = strlen(ereadline);

     while ( fgets(sreadline, N, sfp) != NULL ) {

       ans = strncmp(ereadline, sreadline, len);

       if((ans > 0) || (ans < 0)){

         fprintf(bfp, "%s", sreadline);

       }

     }

  }
```

次に最良の傾きについては，開口問題を改善した座標の組から乱数でランダムに 2 組の対応点を選択し，斜交軸変換を行う．そして対象物までの距離を算出し，射影変換との誤差を比較する．これを繰り返し，データ化することで，閾値を設定し，最良の傾きを決定する．ランダムに対応点を 2 組選択し，斜交軸変換を行い，左右の画像を重ねた結果を以下に示す．

図 4.9: 変換結果



(a) 例 1　　(b) 例 2

例 1 での x 軸の傾き A は 0.313084，y 軸の傾き B は 0.000885 で，例 2 では A は-0.226827，B は-0.000604 となっている．ステレオ画像の性質上，高さのない道路面上は左右の画像を重ねたとき，ちょうど重なるのが理想である．つまり，例 1 が理想に近い変換結果である．

5　性能評価

5.1　変換結果

実際の射影変換，目視での対応点決定結果に基づき斜交軸変換を行う従来法，および提案手法による左右の変換画像を重ねた結果を以下に示す.

図 5.10: 変換結果

(a) 射影変換

(b) 斜交軸変換

(c) 提案手法

道路面はちょうど重なり合っており，従来法同様，少々の誤差はあるものの，ほぼ理想の変換結果が得られている.

5.2 距離算出結果

射影変換に対し，従来の斜交軸変換と，提案手法の距離精度を比較したものを表 1 に示す．射影変換による算出距離で従来の斜交軸変換，および提案手法による算出距離との差を割った数値を誤差率とする．

表 5.2: 距離精度比較

算出距離 (誤差率)		
射影変換	斜交軸変換	提案手法
24.356	24.356(±0.00%)	24.356(±0.00%)
22.387	22.387(±0.00%)	22.389(+0.01%)
23.821	24.858(+4.36%)	24.803(+4.12%)
22.632	22.498(-0.59%)	22.731(+0.44%)

距離精度は従来の斜交軸変換と比較し，わずかながら向上し，平均誤差率は従来の 1.54%から 1.24%にまで改善されている．

6　あとがき

実験結果から，距離精度において適した対応点探索を行い，平行化することで誤差率を低減させることが可能であるとわかった．そして，同一のカメラ，設置状況において，一度キャリブレーションを行えば，そのパラメータを用いて異なる画像でも精度を下げずに平行化することが可能であることがわかった．

今後の課題としては，実際に本研究の提案手法に SIMD 命令を実装し，射影変換との処理速度を比較し，射影変換を提案手法で置き換えられることを示す必要がある．

謝辞

本研究を行うにあたり，御指導，御助言を頂きました近藤利夫教授，佐々木敬泰助教授，深澤研究員に深く感謝致します．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いたします．

参考文献

[1] 水野貴誠，:“ 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 ”，三重大学卒業論文，2010 年

[2] 河合恵奈，:“ 斜交軸変換によるステレオ画像処理の高速化の研究 ”，三重大学卒業論文，2011 年

[3] 嶋好博，柏岡誠治，東野純一 :“ ランに対する座標演算に基づく 2 値画像の高速回転のための一手法”，子情報通信学会論文誌 D，Vol.J71-D，No.7，pp.1296-1305，1988 年 7 月

[4] 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

[5] 林武史，関口眞吾，小野口一則 :“ 日本道路公団向け後尾警戒システム ”，東芝レビュー，Vol.57，No.7，2002 年

[6] 佐々木一人，河本新二 :“ 安全運転を支援する車載画像システム ”，東芝レビュー，Vol.59，No.4，2004 年

[7] 実吉敬二，塙圭二，十川能之，荒井一真 :“ステレオ画像を用いた運転支援のための前方状況認識システム ”，信学技報，1997 年 5 月

A　付録

A.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.

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

A.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，1.3 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．1.4 の場合一

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 1.3: 汎用命令

つの命令で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 1.4: SIMD 命令