〔非公開〕

ＴＲ－Ｍ－００１０

イメージ表現支援技術の研究

－画像群からの奥行き情報の抽出－

福　田　　浩　士　　井　上　　誠　喜
Hiroshi　FUKUDA　　Seiki　INOUE

１９９６．２．２３

ＡＴＲ知能映像通信研究所

イメージ表現支援技術の研究

- 画像群からの奥行き情報の抽出 -

The Technology for Image Expression

- Extraction of Depth Information from a Group of Images -

In this research , I extracted the depth information from a group of images , and synthesized images using the depth information.

I applied a triangulation theory to extract depth information and a block matching to get a motion of pixels.

平成 8 年 2 月 23 日

ATR 知能映像通信研究所　第 3 研究室

豊橋技術科学大学　実務訓練生

福田　浩士　[Hiroshi Fukuda]

1 背景・目的

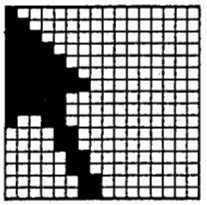
複数方向からシーンを撮影した画像群からその3次元構造を抽出する研究が幅広く行われている.今回の実習は,その画像群から奥行き情報を求め,その情報をもとに画像の合成を行なうことを目的とする.また,奥行き情報の取得方法として,各画素の移動量をブロックマッチングで求め,その移動量により奥行きを決定する方法について述べる.

2 画像処理の基礎

今回の実務訓練で初めて画像処理を行うので,まず画像のデータ表現など基礎的なことを学ぶ必要があった.そこで,既存の画像処理プログラムを用いて画像処理を行い,2値化,輪郭抽出,雑音除去などを勉強した.また,そのプログラムを参考にして,簡単な画像処理プログラムを作成した.

2.1 C言語による画像処理プログラム

コンピュータの中では,画像を画素(pixel)に分けて,各画素における濃淡の値を整数値(ディジタル)表現している.簡単な例をFig.1(a)に示す.この図をC言語で表したものが同図(b)で,ディジタル画像の左上隅の画素を(0,0)番目の画素とし,その要素から数えて横方向にi画素,縦方向にj画素進んだ位置にある(i,j)番目の画素の値を配列要素image_in[j][i]に格納している.ここでは量子化ビット数を8ビットとし,画素の濃淡を0~255の数値で表している.



(a) 入力画像

```
#define X_SIZE 16
#define Y_SIZE 16

char image_in[Y_SIZE][X_SIZE]
          ={{1,0,0,  },
            {1,1,0,  },
                |
            {0,0,0,  }}
```

(b) 画像データ表現

Fig. 1: C言語による画像処理の例

2.2 画像の雑音除去

移動平均法とメディアンフィルタによる雑音の除去を行なった.使用した画像は8mmビデオカメラで撮影したものをPPM形式の277×197画素に落した画像で,この画像をR,G,Bにわけてそれぞれをモノクロ画像として使用した. R,G,Bの画像それぞれに移動平均法とメディアンフィルタを5回行ない,再度PPM形式に戻した.その結果をFig.2に示す.

これを見ると両方ともノイズは減少したが,画像が全体的に平坦になったことがわかる. しかし,移動平均法はエッジ部分をすべてぼかしてしまっているのに対し,メディアンフィルタの方はエッジ部分をある程度残していることがわかる.これにより,メディアンフィルタの方がエッジの保存において優れていることがわかった.

2.3 簡単な画像処理プログラムの作成

入力画像の指定した位置に指定した高さのダイヤ(高さ:幅=2:1)を出力するプログラムを作成した.そのプログラムをリストA.1に実行結果をFig.3に示す.ここで使った画像は上記の画像のRのモノクロ画像である.

実行手順は次の通り.

(a) 元画像

(b) 移動平均法

(c) メディアンフィルタ

Fig. 2: 画像処理結果 1

実行手順

1. 対角線の交点 (138,98), 高さ 100, 濃度 0 のダイヤを描く
2. 対角線の交点 (138,98), 高さ 80, 濃度 255 のダイヤを描く
3. 対角線の交点 (138,98), 高さ 40, 濃度 0 のダイヤを描く
4. 対角線の交点 (138,98), 高さ 20, 濃度 255 のダイヤを描く
5. 対角線の交点 (0,0), 高さ 160, 濃度 0 のダイヤを描く
6. 対角線の交点 (0,0), 高さ 80, 濃度 255 のダイヤを描く
7. 対角線の交点 (276,0), 高さ 160, 濃度 0 のダイヤを描く
8. 対角線の交点 (276,0), 高さ 80, 濃度 255 のダイヤを描く
9. 対角線の交点 (0,196), 高さ 160, 濃度 0 のダイヤを描く
10. 対角線の交点 (0,196), 高さ 80, 濃度 255 のダイヤを描く
11. 対角線の交点 (276,196), 高さ 160, 濃度 0 のダイヤを描く
12. 対角線の交点 (276,196), 高さ 80, 濃度 255 のダイヤを描く

ここでは,2 値のダイヤを出力するようにしているが, プログラムを少し変更すれば 0~255 の 256 種類の濃度のダイヤを出力することができる.

(a) 入力画像

(b) 処理結果

Fig. 3: 画像処理結果 2

3 ステレオ画像データベース

本研究では「筑波大学多視点画像データベース」を使用した.

3.1 画像の概略

- 9 行 9 列のマトリクス状に配置した 81 視点から撮影
- 1 frames/scene(81 images/scene)
- 640×480 pixel, RGB 24bit
- PPM フォーマット

3.2 撮影方法

1 台のカメラを,X-Y 軸の 2 方向に平行移動するアームロボットに取り付け, 視点を格子点状に等間隔で移動させながらシーンを撮影した.

3.3 画像ファイル

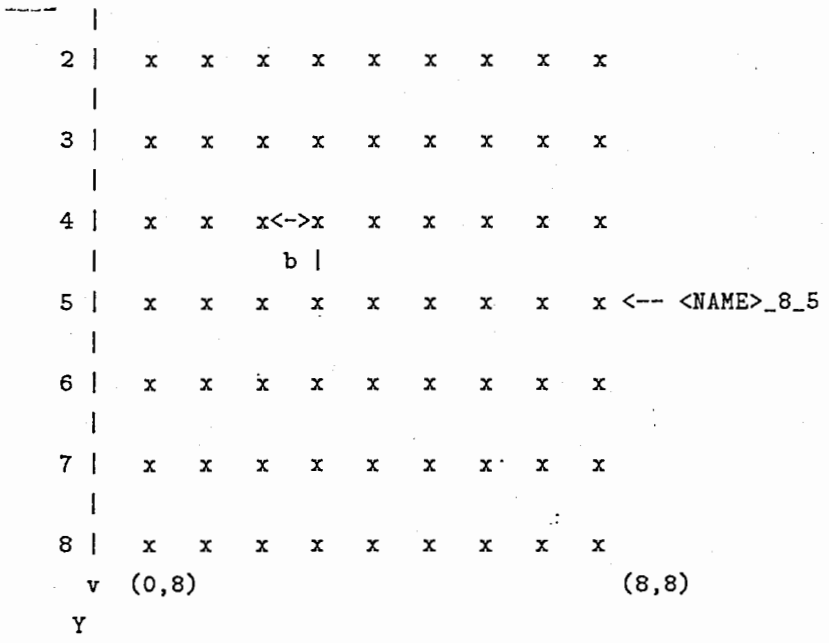
XY 平面上の格子点状に配置された,$9 \times 9 = 81$ 視点によって, 静止物体のシーンを撮影したものである. 格子の間隔は、シーンごとに一定の値 (基線長 : b) とした. すべての視点でカメラの光軸は,XY 平面と直行しており, 輻輳は無い.

各ファイル名は次の命令規則に従う.

<NAME(3~4 文字)> _ <X(1 文字)> _ <Y(1 文字)>

NAME は {SANT(ぬいぐるみ),PLNT(観葉植物),CITY(町のジオラマ),KID(マネキン)} のいずれかであり, X と Y はカメラの座標位置を示す.

```
      0   1   2   3   4   5   6   7   8
    ----------------------------------------->X
   |  (0,0)                           (8,0)
 0 |   x   x   x   x   x   x   x   x   x
   |
 1 |   x   x   x   x   x   x   x   x   x
```



(物体に向かって見た場合の図)

本研究では,CITY(街のジオラマ) を用いた.

パラメータ

- ファイル名：CITY_0_0 ～ CITY_8_8
- 基線長 (b)：X 軸 8mm / Y 軸 8mm
- 輻輳：なし
- CCD 面から物体までの距離：手前の建物　33cm, 高層ビルの右端の角　66cm
- レンズの焦点調節：40cm 弱
- レンズの絞り: 4

今回 CITY_5_5 を対象画像とした. Fig.4に CITY_5_5 を示す.

4　奥行き情報の取得

ここでは, 画像の奥行き情報の取得方法について述べる.

4.1　三角測量の原理による奥行き情報の取得

今回,3 角測量の原理から奥行きデータを求めることにした. Fig.5に,2 次元平面における 3 角測量の原理を示す. 並列に並べた一対の観測系があり, それらのレンズ中心 a,b を通る直線を x 軸, a を原点とする xy 直交座標を考え, レンズ中心の間隔を l, レンズ中心と観測面の距離を f とする. このとき, 標点 P の像が観測面上の位置 x_a, x_b で観測されたとすると, 標点 P(x,y) は,

$$x = \frac{x_a l}{x_a - x_b}, \qquad y = \frac{fl}{x_a - x_b} \tag{1}$$

で与えられる. いま正確な奥行き情報は必要なく, 各画素ごとの奥行きの比がわかれば良いので,$x_a - x_b$ を求め, その値 (つまり移動量) が大きければ y は小さくなりその画素は手前にあることがわかり, 移動量が小さければ y は大きくなりその画素は奥にあることがわかる. 今回画素の移動量を求めるためにブロックマッチングの手法を用いた.

Fig. 4: 対象画像

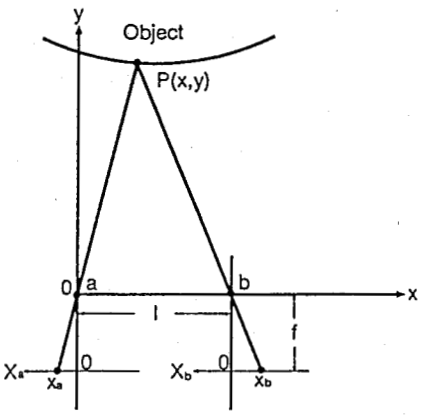


Fig. 5: 三角測量の原理

4.2 移動量の計算

ある画素 pixel(x_i, y_j) を中心とする縦 Y_l, 横 X_l(それぞれ奇数) のブロックを考える. ブロックの移動量を dx とし, 水平方向のステレオ画像をそれぞれ画像 1, 画像 2 とする. 画像 1 のブロックを水平方向に動かし, 画像 2 との差分の総和 D,

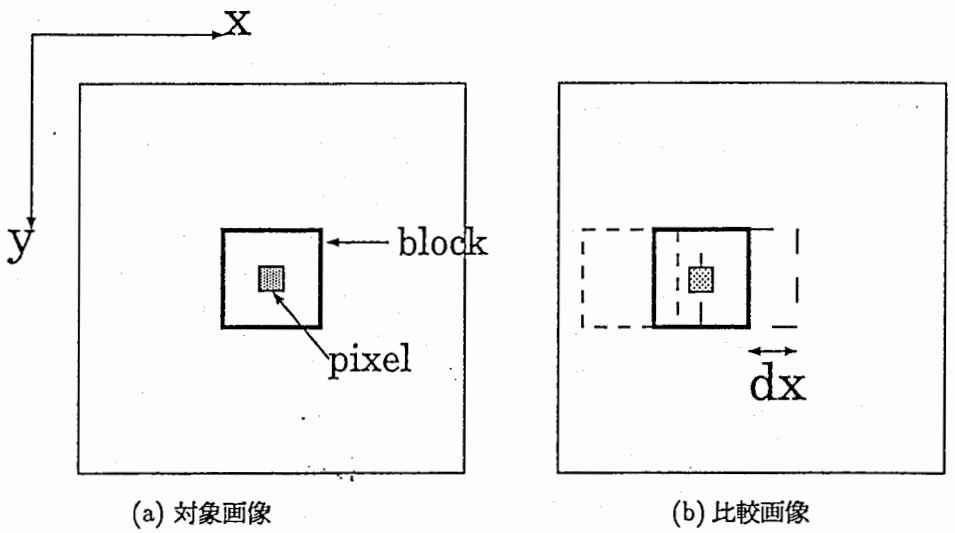
$$D = \sum_{m=-\frac{X_l}{2}}^{\frac{X_l}{2}} \sum_{n=-\frac{Y_l}{2}}^{\frac{Y_l}{2}} |pixel_1(x_{i+m}, y_{j+n}) - pixel(x_{i+m} + dx, y_{j+n})| \tag{2}$$

が最小になるような dx を求め, その大きさを画素 (x_i, y_j) の移動量とする. つまり, 最も画素情報が似ているブロックを探すのである. リスト A.2 に移動量を求めるプログラムを示す.

4.3 奥行き情報の取得結果

4.3.1 マッチングに用いたブロックの大きさによる結果の相違

今,CITY_5_5 を対象画像とし,CITY_6_5 を比較画像とする. このとき, マッチングに用いたブロックの大きさによる結果の相違を Fig.7に示す. この図は奥行き画像にエッジ画像 (輪郭抽出し, 細線化した画像) を重ねてある. 奥行き



(a) 対象画像　　(b) 比較画像

Fig. 6: ブロックマッチング

画像の見方は, 濃度が濃い画素ほど奥の方に, 淡い画素ほど手前にあると判断されている.

Fig.7を見ると, ブロックが小さいとうまくマッチングが取れないところが多くなりかなり雑音が含まれ, 逆にブロックが大きいと雑音は減るが, エッジ部分が鈍ってしまうことがわかる. 今回は, 奥行き情報を用いた画像の切出しを目的としているため, ある程度のエッジの正確さが必要であるのでブロックを 7×7 としたときの奥行き情報を用いることにした.

4.4 4つの奥行き画像の合成

Fig.7を見ると対象画像では存在していたのに比較画像では存在しなくなる部分 (建物の陰に隠れてしまう部分) のマッチングがうまく取れていないことがわかる. また, 水平方向の画像のマッチングでは, 水平方向にほとんど変化のない部分のマッチングも取れていないことがわかる.

そこで, 対象画像の上下左右の画像 (それぞれ,CITY_5_4,CITY_5_6,CITY_4_5,CITY_6_5) を比較画像としたときに得られる 4 枚の奥行き画像を合成することを考えた (Fig.8).

Fig.9~Fig.12にその 4 枚の奥行き画像を示す.

4.4.1 画像の合成方法

• 最も信頼性の高いデータを用いる　ある画素 pixel(i,j) について 4 つの奥行きデータの中から, 最も信頼性の高い, つまりマッチングの度合いが一番良い (式 (1) において求めた最小の D が 4 つ中でも最小) のものを選んでその画素の奥行きデータとする方法である. この方法で合成を行った結果を Fig.13に示す.

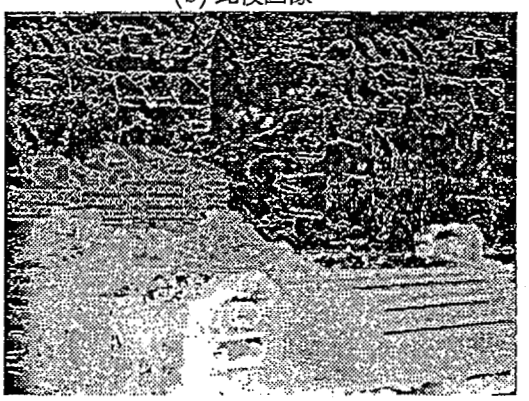
Fig.13を見ると, 建物の陰に隠れてマッチングがうまく採れないところを除去できたことがわかる. しかし, ところどころノイズがのっている. この原因を確かめるために, ノイズであると考えられるある画素について, ブロックの移動量とマッチングの度合いの関係を見ることにした. Fig.14に pixel(540,395) についての関係を示す. これを見ると, この部分は水平方向に画素の変化があまり見られない部分であるために水平方向 (left,right) のグラフにほとんど変化がないことがわかる. そして,right の移動量が小さいところで最小値をとっているためこの奥行きデータがその画素の奥行きデータになってしまっているのである.

• 分散が小さいグラフを除き, 残りの内で最も信頼性の高いデータを　上の結果より今度は,4 つのデータの内グラフの分散が小さい, つまり変化があまりないものは除いて, 残ったものの中でマッチングの度合いが最も良いところを選ぶことを考えた. その結果を Fig.15に示す.

(a) 対象画像

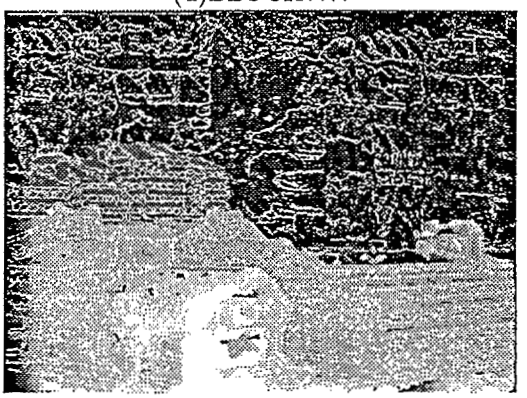
(b) 比較画像

(c)BLOCK3×3

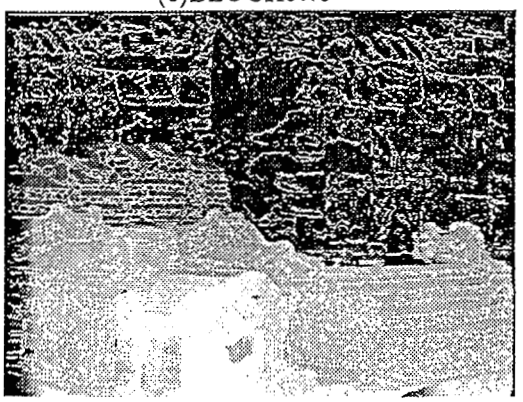


(d)BLOCK7×7

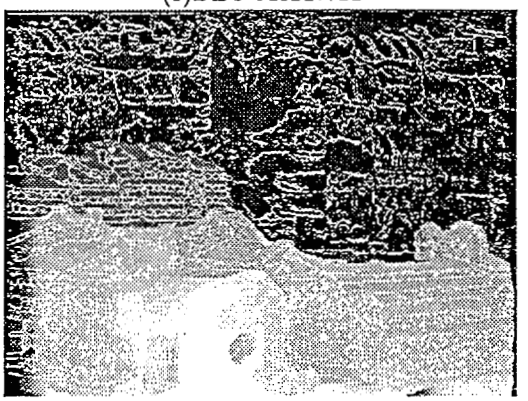
(e)BLOCK9×9



(f)BLOCK11×11

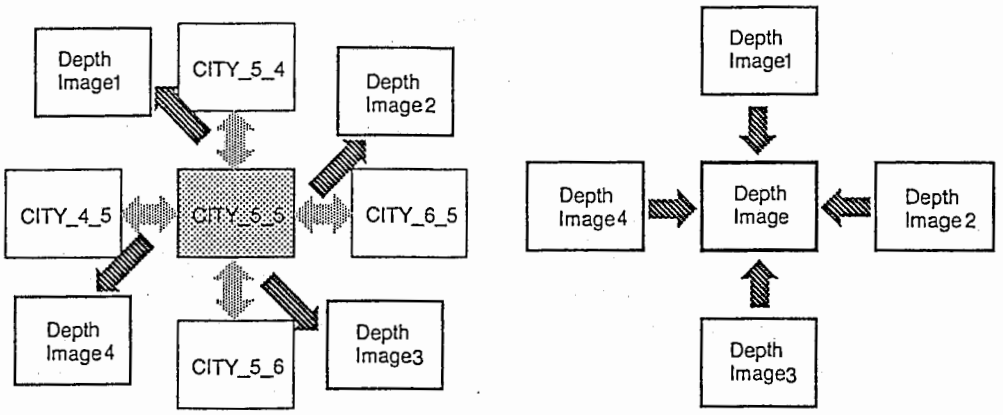


(g)BLOCK15×15



(h)BLOCK21×21

Fig. 7: ブロックの大きさによる奥行き情報の相違



(a) 奥行き情報の取得　　(b) 奥行き情報の合成

Fig. 8: 4 つの奥行き奥行き情報の合成

Fig.15と Fig.13を比較するとかなりノイズが除去されていることがわかる．しかし，まだ除去されていない部分がある．この部分は,Fig.16のような特性になっている．これによると，分散の高い upper において 30 の近辺で最小となるべきところで移動量の小さい部分に最小値が出てしまっているからである．

次に，その部分を除去しなければならない．その除去方法として以下のことが考えられる．

- 大きなブロックで大体の移動量を求めておき，その近辺のみ探索するようにする
- 残っている波形を足し合わせ，その値が最小となる移動量をとる．

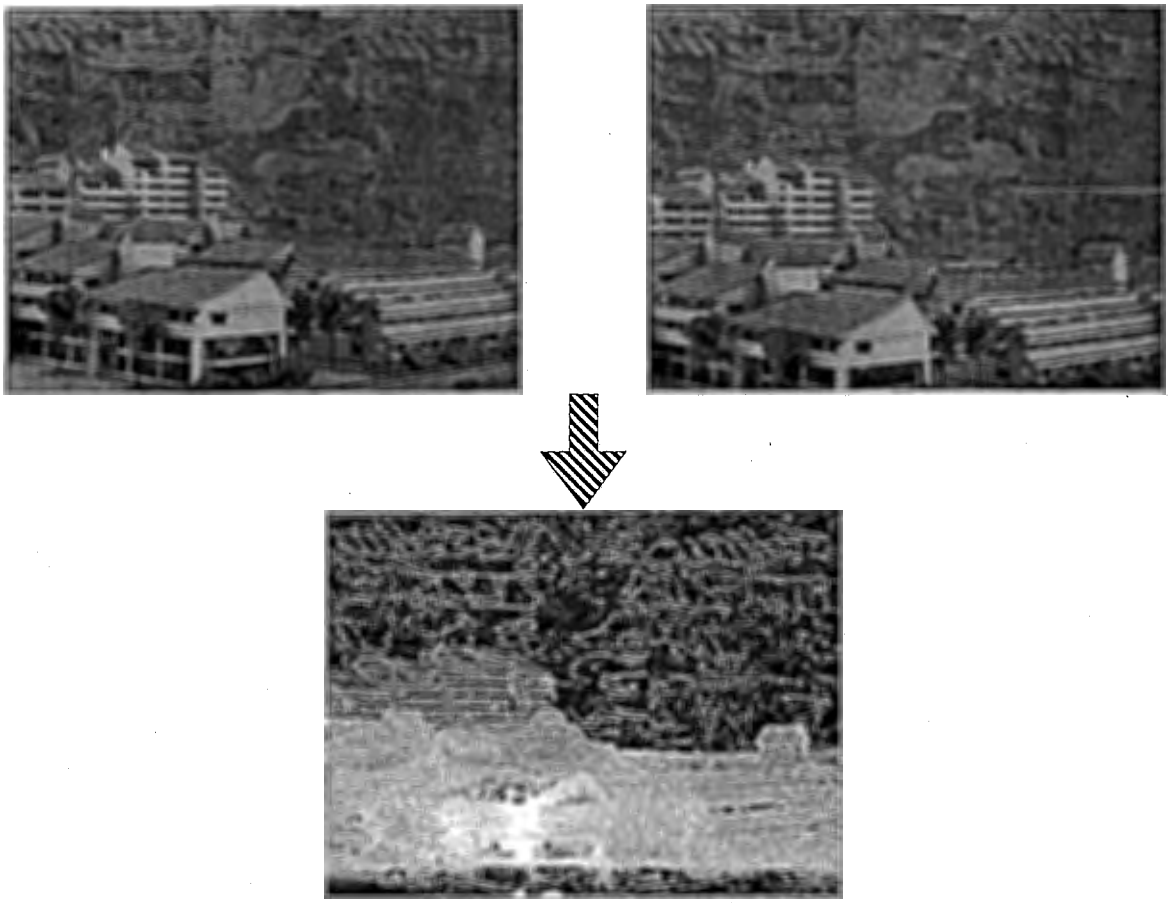


Fig. 9: 1 つ上の画像を比較画像としたときの奥行き画像

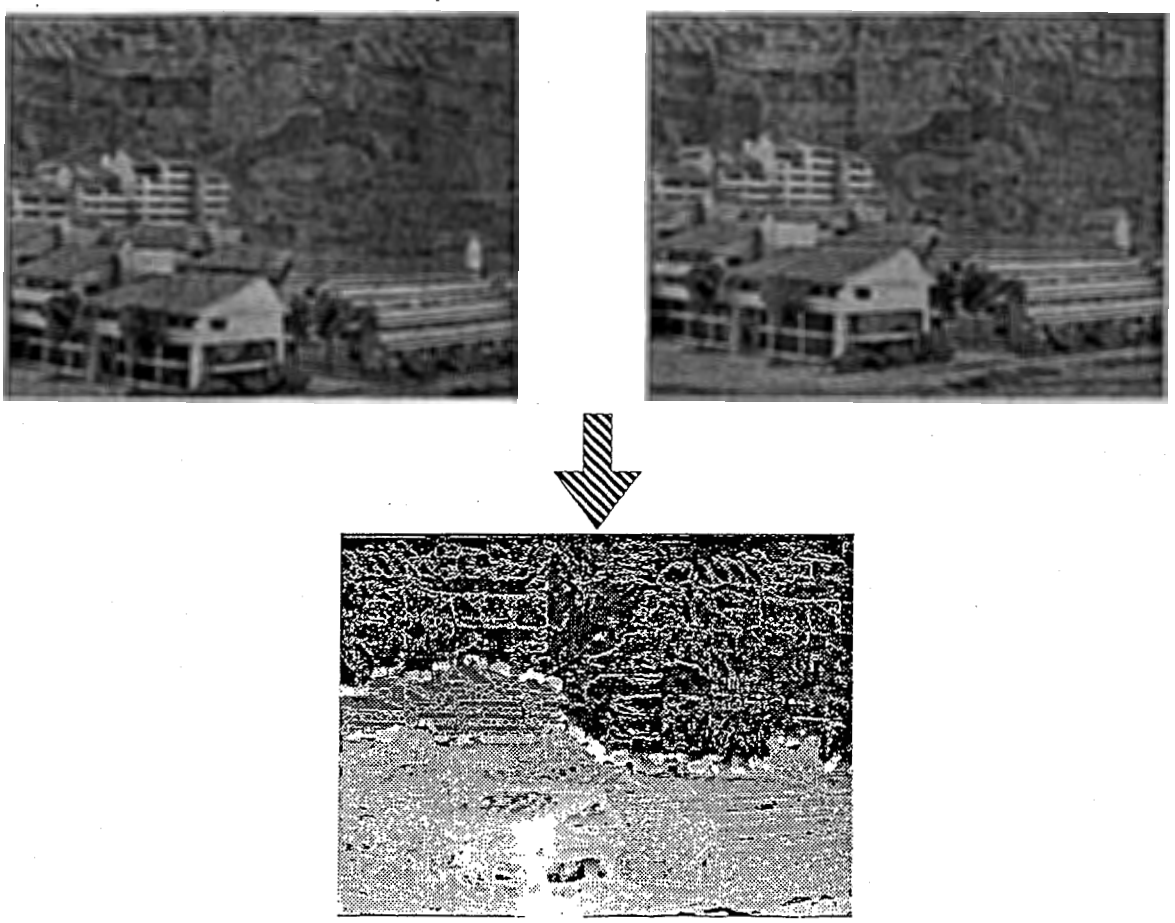


Fig. 10: 1つ下の画像を比較画像としたときの奥行き画像

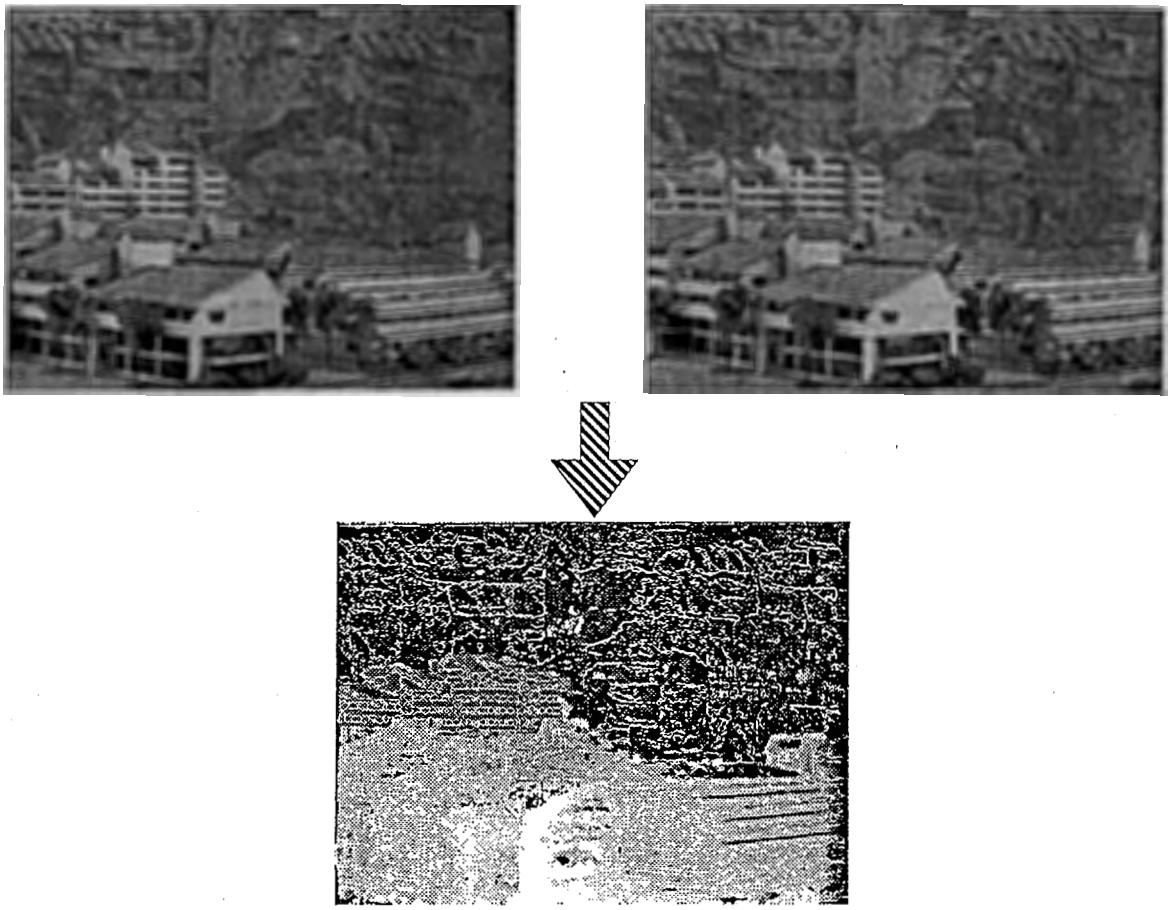


Fig. 11: 1つ左の画像を比較画像としたときの奥行き画像

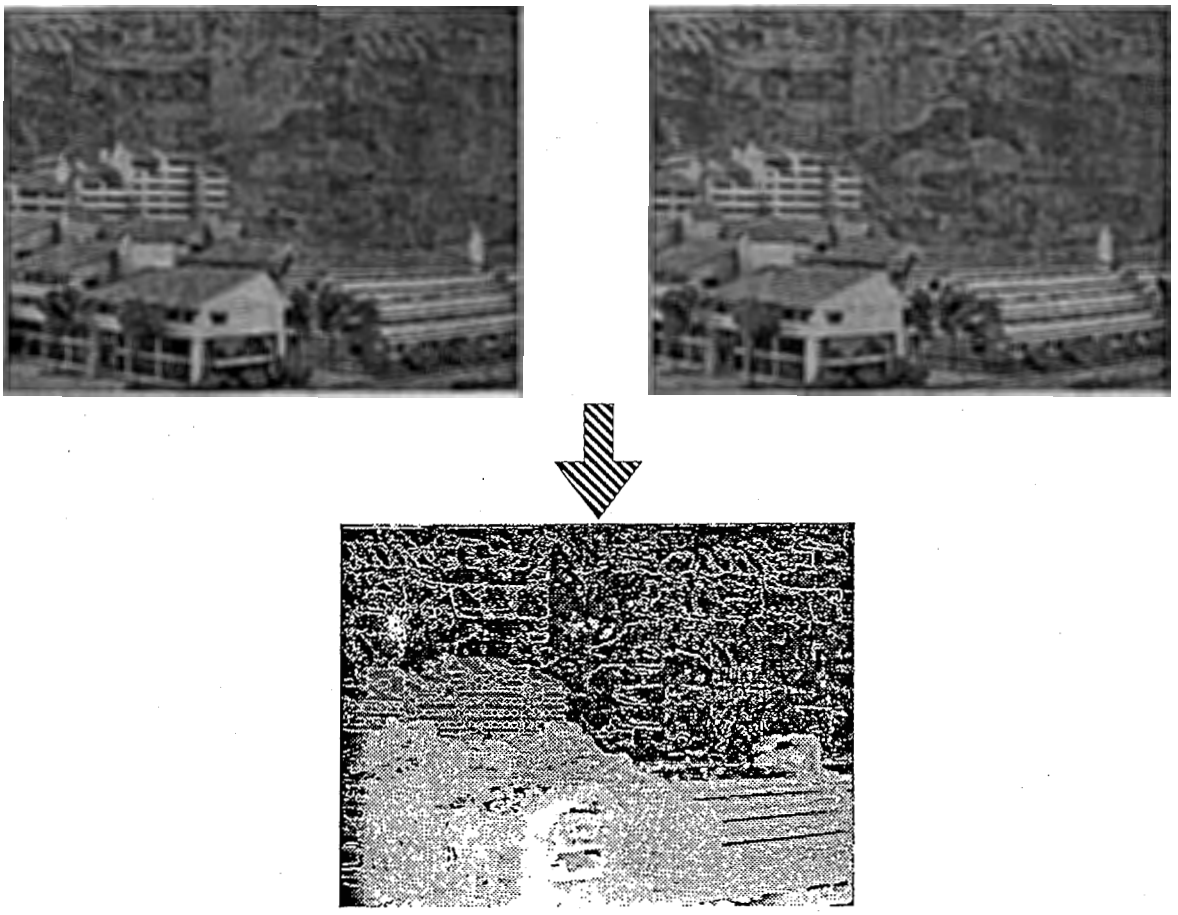


Fig. 12: 1つ右の画像を比較画像としたときの奥行き画像

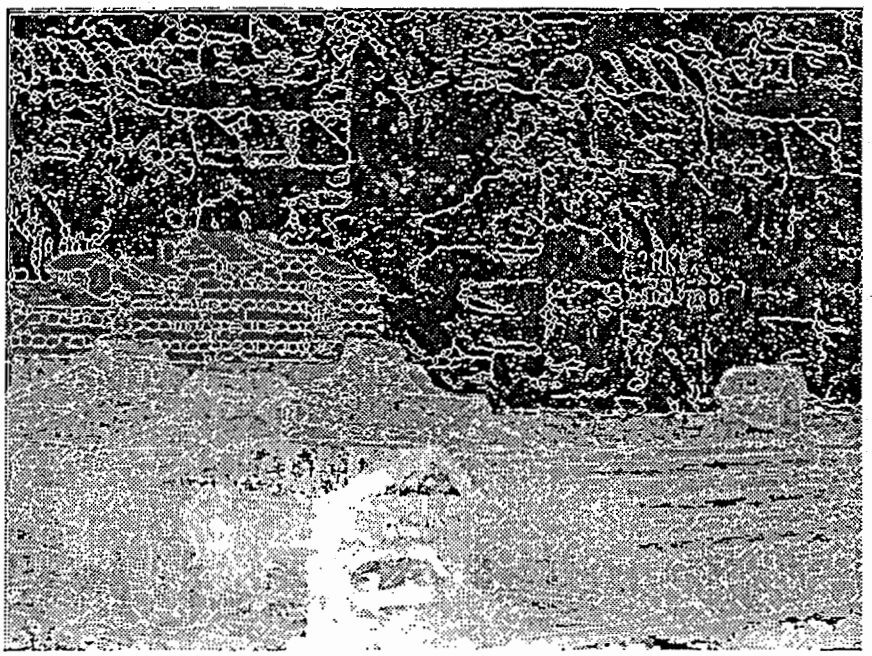


Fig. 13: 合成した奥行き画像 (マッチングの信頼性の高いところを採ったもの)

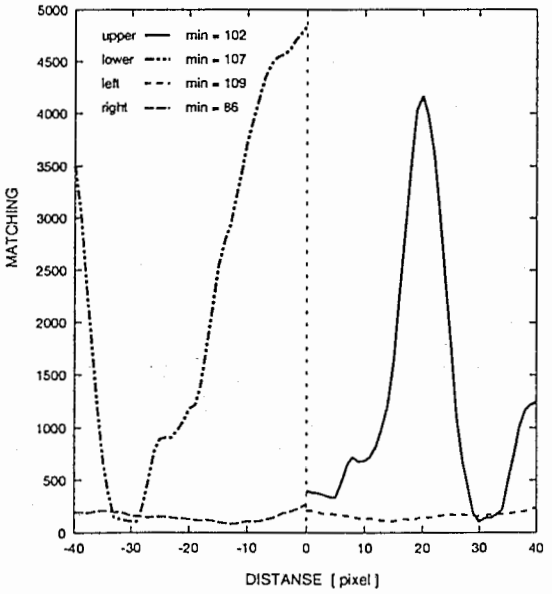


Fig. 14: ノイズ部分のマッチングのグラフ pixel(540,395)

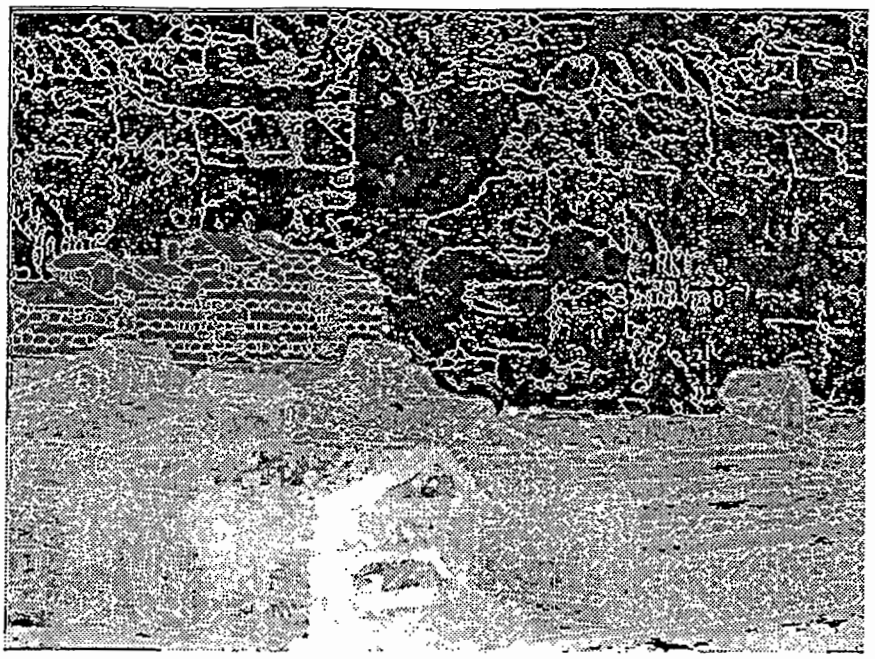


Fig. 15: 合成した奥行き画像 (分散の小さいものを除いた場合)

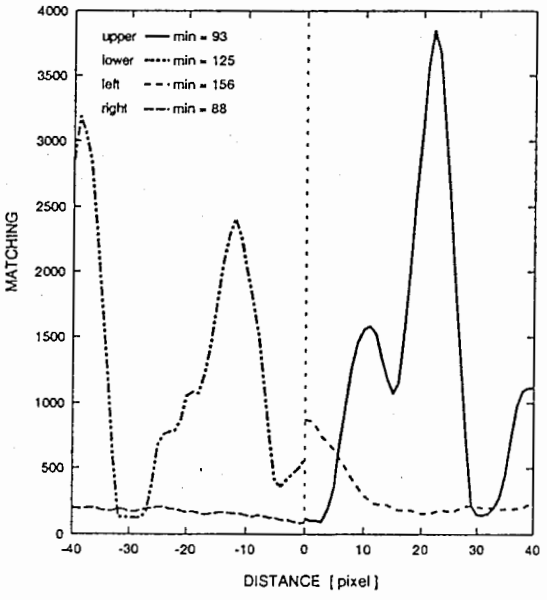


Fig. 16: 除去されていない部分のマッチングのグラフ

5 画像の合成

得られた奥行き情報を用いて,対象画像に処理を施した. その結果を Fig. 17に示す.

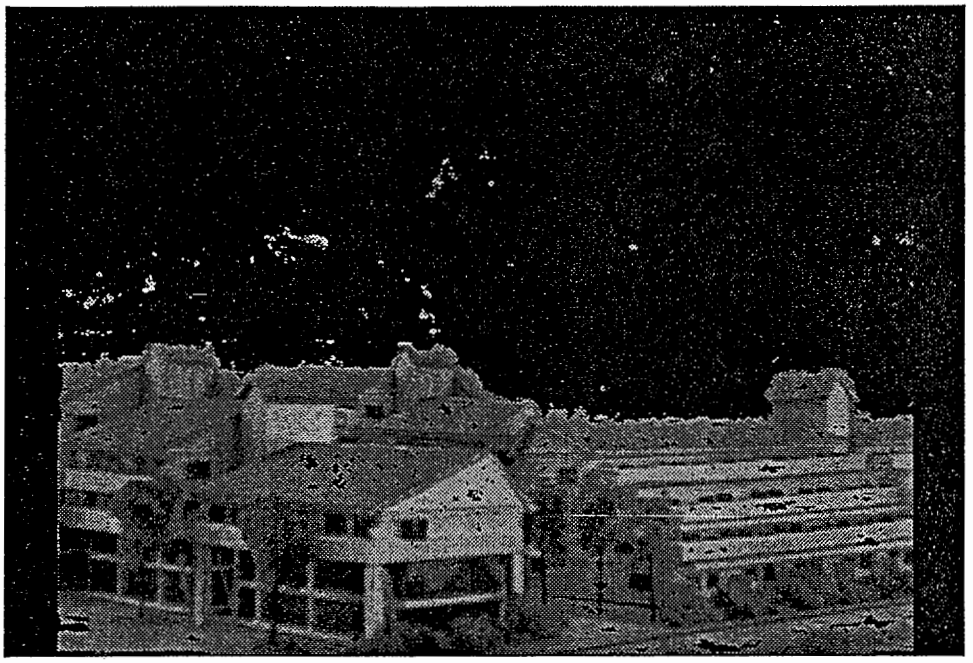


Fig. 17: 奥行き情報を用いた画像処理結果

これは,奥行きデータを用いて最も奥にあるビルディングを消した画像である.

6 まとめ

1. 画像処理の基礎勉強を行った
2. 上下左右のステレオ画像による奥行き情報抽出プログラムを作成した
   しかし,奥行き情報がうまくとれない部分がある.
3. 上下左右のステレオ画像より得られた 4 つの奥行き情報を合成することを考えた
   - 画素ごとに 4 つの中で最も信頼性の高いデータを用いる方法
   - 分散の小さいものは除いて残りの中で最も信頼性の高いデータをとる方法
     しかし,まだ奥行き情報がうまくとれていない部分がある.

謝辞

実務訓練を行うにあたって，多大な御指導をいただいた，井上誠喜室長をはじめ知能映像通信研究所第三研究室の方々に心から感謝の意を表します．また，私に充実した実務訓練の場を与えて下さった知能映像通信研究所ならびに通信システム研究所の方々にこの場を借りて厚くお礼を申し上げます．

参考文献

1. 「Ｃ言語で学ぶ実践画像処理」，八木・井上・林・中須・三谷・奥井・鈴木・金次共著，オーム社
2. 「Ｃ言語で学ぶ実践ディジタル映像処理」，八木・井上・林・奥井・合志共著，オーム社
3. 「コンピュータ画像処理入門」，田村秀行監修，総研出版
4. 「画像計測」，土屋裕著，テレビジョン学会編集，昭晃堂

Appendix

リスト A.1：ダイヤを表示する画像処理プログラム

```
#include "params.h"

put_dia(image_in,image_out,center_x,center_y,r,horl)
/* ダイヤをを描く */
unsigned char image_in[Y_SIZE][X_SIZE];
unsigned char image_out[Y_SIZE][X_SIZE];
int center_x;
int center_y;
int r;
int horl;
{
  int i,j,dia_height,hl;

  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      image_out[i][j]=image_in[i][j];
    }
  }

  if(horl == 1 || horl == 2){
    if(horl == 1) hl=HIGH;
    else  hl=LOW;

    dia_height=r/2;
    for(i = 0; i < r; i++){
      for(j = 0; j < dia_height; j++){
if(center_y-j >=0 && center_x-i >=0)
  image_out[center_y-j][center_x-i] = hl;
if(center_y-j >=0 && center_x+i <= X_SIZE-1)
  image_out[center_y-j][center_x+i] = hl;
if(center_y+j <= Y_SIZE-1&& center_x-i >=0)
  image_out[center_y+j][center_x-i] = hl;
if(center_y+j <= Y_SIZE-1 && center_x+i <= X_SIZE-1)
  image_out[center_y+j][center_x+i] = hl;
      }
      dia_height-=2;
    }
  }
}
```

画像群からの奥行き情報の取得プログラム

User's Guide

平成 8 年 2 月 23 日

ATR 知能映像通信研究所　第 3 研究室
豊橋技術科学大学　実務訓練生
福田　浩士　[Hiroshi Fukuda]

1 プログラムの概要

このプログラムは, 対象画像とその上下左右の画像を比較して対象画像の奥行き情報を得るプログラムである.
ソースプログラムは,miris37:/home/fukuda/ProImage/ に格納している.

Tab. 1: ソースプログラム

ソースプログラム	関数	概要
Video_mine.c	main	メイン (メニュー)
	image_init	画像データ領域の確保と初期化
	image_read	ディスクから画像を読み出す
	image_write	ディスクに画像を書き込む
	image_clear	画像データをクリアする
Get_Depth.c	get_depth	奥行き情報を求める
Mvec_mine.c	mvec_mine_hm	比較画像が左のときの移動量を求める
	mvec_mine_hp	比較画像が右のときの移動量を求める
Mvec_mine_v.c	mvec_mine_vm	比較画像が上のときの移動量を求める
	mvec_mine_vp	比較画像が下のときの移動量を求める
Gousei.c	gousei	2 画像の合成
Nichika	nichika	画像の 2 値化
Median.c	medfil	メディアンフィルタ
Dep_Gousei.c	depth_gousei	4 つの奥行き画像の合成 1
Dep_Gousei2.c	depth_gousei2	4 つの奥行き画像の合成 2
mat_graph.c	mat_graph_hm	ある画素についてのマッチングの度合いのグラフ (左)
	mat_graph_hm	比較画像が右
mat_graph_v.c	mat_graph_vm	比較画像が上
	mat_graph_hp	比較画像が下
Block_xy.c	block_xy	ブロック内の画素の縦横の変化を調べる
Gaisetsu.c	gaisetsu	外接長方形を得る

2 プログラムの実行

コマンドラインで,

```
>> Video_mine
```

と入力するとプログラムが起動される. メニューは次の通り.

1. ファイルからの画像の読み込み
2. ファイルへの画像の書き出し
3. 奥行き情報の取得
4. 画像の合成
5. Work 画像の 2 値化
6. Median Filter をかける
7. 奥行きデータの合成 (最小値をとる)
8. 奥行きデータの合成 (分散を考慮)

9. 移動量 - マッチングの度合いのグラフ

10. ブロックの縦と横の変化をしらべる

11. 外接長方形を求める

2.1 ファイルからの画像の読み込み

読み込む画像データが格納されているファイル名を入力し,IN,OUT,WORK,WORK2,WORK3 のいずれかのウィンドウを選ぶ.

2.2 ファイルへの画像の書き出し

IN,OUT,WORK,WORK2,WORK3 のいずれかのウィンドウを選び, 画像を書き込むファイル名を入力する.

2.3 奥行き情報の取得

ステレオ画像より, 奥行き情報を求める. これを実行するときはあらかじめ次の 3 つのファイルを用意しておく.

1. 各奥行き画像を求める過程で得られる, 画素ごとの移動量, マッチングの度合い, マッチングの度合いの標準偏差と平均値を格納するファイルの名前を格納したファイル.

<<フォーマット>>

<移動量を格納するファイル>　　<= 比較画像が上のときのデータを格納するファイル
<マッチングの度合いを格納するファイル>
<マッチングの度合いの標準偏差を格納するファイル>
<マッチングの度合いの平均値を格納するファイル>
<移動量を格納するファイル>　　<= 比較画像が下のときのデータを格納するファイル
<マッチングの度合いを格納するファイル>
<マッチングの度合いの標準偏差を格納するファイル>
<マッチングの度合いの平均値を格納するファイル>
<移動量を格納するファイル>　　<= 比較画像が左のときのデータを格納するファイル
<マッチングの度合いを格納するファイル>
<マッチングの度合いの標準偏差を格納するファイル>
<マッチングの度合いの平均値を格納するファイル>
<移動量を格納するファイル>　　<= 比較画像が右のときのデータを格納するファイル
<マッチングの度合いを格納するファイル>
<マッチングの度合いの標準偏差を格納するファイル>
<マッチングの度合いの平均値を格納するファイル>

例) ファイルを a.dat としたとき

```
>> more a.dat
vvm.dat
minvm.dat
bunvm.dat
avevm.dat
vvp.dat
minvp.dat
bunvp.dat
avevp.dat
vhm.dat
```

```
minhm.dat
bunhm.dat
avehm.dat
vhp.dat
minhp.dat
bunhp.dat
avehp.dat
```

2. 比較画像のファイル名を格納しているファイル

<<フォーマット>>
<比較画像(上)>
<比較画像(下)>
<比較画像(左)>
<比較画像(右)>

例) ファイルを b.dat, 対象画像を city_5_5.r としたとき

```
>> more b.dat
city_5_4.r
city_5_6.r
city_4_5.r
city_6_5.r
```

3. 取得した奥行き画像を格納するファイル名を格納しているファイル

<<フォーマット>>
<比較画像(上)として得られた奥行き画像を格納するファイル>
<比較画像(下)として得られた奥行き画像を格納するファイル>
<比較画像(左)として得られた奥行き画像を格納するファイル>
<比較画像(右)として得られた奥行き画像を格納するファイル>

例) ファイルを c.dat としたとき

```
>> more c.dat
cdvmno.dat
cdvpno.dat
cdhmno.dat
cdhpno.dat
```

上記の a.dat , b.dat , c.dat に相当するファイルが必要となる. 入力の順番は番号の通り .

2.4 画像の合成

これは,2 つの画像を画素ごとで比較して画素情報の大きい方を取るものである. 主に, 輪郭抽出した画像と元画像を重ね合わせるときに使う.

これを, 行う前に IN と WORK に重ね合わせる画像を読み込んでおく必要がある.

2.5 WORK 画像の 2 値化

これは, WORK に読み込まれている画像を 2 値化するものである.

2.6 奥行きデータの合成 (最小値をとる)

4つの奥行きデータを合成する．考慮するのはマッチングの度合いのみ．これを実行するときはあらかじめ次の2つのファイルを用意しておく必要がある．

1. 奥行き画像を格納しているファイルの名前を格納しているファイル (前述の c.dat に相当)
2. 画素ごとのマッチングの度合いを格納しているファイルの名前を格納しているファイル

<<フォーマット>>

<比較画像 (上) として得られたマッチングの度合いを格納しているファイル>
<比較画像 (下) として得られたマッチングの度合いを格納しているファイル>
<比較画像 (左) として得られたマッチングの度合いを格納しているファイル>
<比較画像 (右) として得られたマッチングの度合いを格納しているファイル>

例) ファイルを d.dat としたとき

```
>> more d.dat
minvm.dat
minvp.dat
minhm.dat
minhp.dat
```

2.7 奥行きデータの合成 (分散を考慮)

4つの奥行きデータを合成する．ここではマッチングの度合い (移動量) の分散も考慮する．これを実行するときはあらかじめ次の3つのファイルを用意しておく必要がある．

1. 奥行き画像を格納しているファイルの名前を格納しているファイル (前述の c.dat に相当)
2. マッチングの度合いの分散の小さいものを除くための閾値と，移動量の分散が大きいものを除くための移動量の閾値．

<<フォーマット>>

<マッチングの度合いの分散の閾値> <移動量の分散の閾値>

例) ファイルを e.dat としたとき

```
>> more e.dat
200 20
```

3. 各データ (移動量, マッチングの度合い, 分散 (標準偏差),) を格納しているファイルの名前を格納しているファイル

<<フォーマット>>

<移動量 (比較画像が上)>
<移動量 (比較画像が下)>
<移動量 (比較画像が左)>
<移動量 (比較画像が右)>
<マッチングの度合い (比較画像が上)>
<マッチングの度合い (比較画像が下)>
<マッチングの度合い (比較画像が左)>
<マッチングの度合い (比較画像が右)>
<分散 (比較画像が上)>
<分散 (比較画像が下)>
<分散 (比較画像が左)>
<分散 (比較画像が右)>

例) ファイルを f.dat としたとき

```
>> more f.dat
vvm.dat
vvp.dat
vhm.dat
vhm.dat
minvm.dat
minvp.dat
minhm.dat
minhp.dat
avevm.dat
avevp.dat
avehm.dat
avehp.dat
```

2.8 移動量 - マッチングの度合いのグラフ

これを実行する前に,
IN : 比較画像
WORK : 対象画像
をそれぞれ読み込んでおく. 実行手順は次の通り.

1. IN に読み込んだ比較画像が対象画像に対してどこの画像なのかを選択する
2. 結果を格納するファイル名を入力する
3. どの画素かを入力する

実行例)

```
番号を入力してください ? 9
比較画像 (1: 上 2: 下 3: 左 4: 右) ? 1
結果をセーブするファイル名を入力してください (gnuplot 形式)-> mgvm540_395.dat
xi yj ? 540 395
```

結果のファイルの中身は次のようになっている .

<<フォーマット>>

<画素の移動量> <マッチングの度合い>

```
    :                   :
    :                   :
    :                   :
```

例) mgvm540_395.dat の場合

```
>> more mgvm540_395.dat
0 395.000000
1 381.000000
    :
    :
40 1245.000000
```

同じ画素について上下左右の 4 つのデータをとり,gnuplot でグラフを描くと Fig.1のようになる.

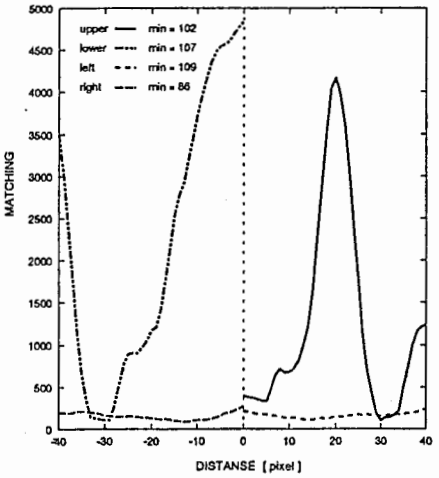


Fig. 1: gnuplot での出力

2.9 ブロックの縦と横の変化を調べる

全てのプログラムにおいてブロックの中心の画素が処理対象画素となっているのだが, ここではその画素の上下左右の画素がどういう変化をしているのかを調べる. つまり, Fig.?? において十字に色のついた領域の変化を調べるのである. 水平方向と鉛直方向の変化を別々に調べ, それぞれをコサイン変換して特性を見る.

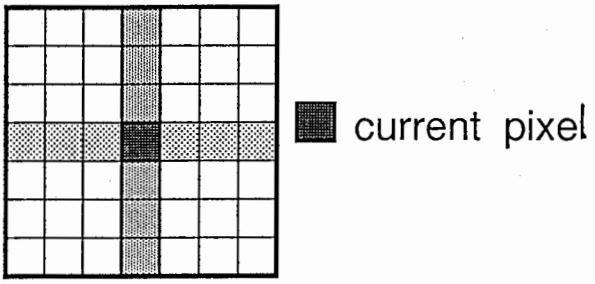


Fig. 2: 調べる領域

これを実行する前に IN に対象画像を読み込んでおく. 手順は次の通り.

1. 水平方向 (x 軸方向) との結果を格納するファイル名を入力する
2. 鉛直方向 (y 軸方向) の結果を格納するファイル名を入力する
3. 対象画素を入力する

実行例)

```
番号を入力してください ? 10
水平方向の結果を格納するファイル => dctx430_300.dat
鉛直方向の結果を格納するファイル => dcty430_300.dat
xi yj ? 430 300
```

結果のファイルの中身は次のようになっている.

```
<<フォーマット>>
    <周波数> <コサイン変換の結果>
```

```
        :           :
        :           :
        :           :
```

例）dctx430_300.dat の場合

```
>> more dctx430_300.dat
0 177.786848
1 -1.616216
2 -4.719123
      :
      :
```

3 研究データ

3.1 ディレクトリ構造

研究中に得たデータは, miris37:/home/fukuda/ProImage/ の下にある. そのディレクトリ構造を次に示す.

```
/home/fukuda/
     |
     |-- ProImage/ <= ソースプログラムなどを格納
            |
            |- Block3/ <= マッチングのブロックを 3x3 画素としたときの結果を格納
            |    |
            |    :
            |    :
            |
            |- Block7/ <= ブロック 7x7
            |    |
            |    |- Dct/ <= 10: を行ったときの結果を格納 (ブロックの変化)
            |    |
            |    |- Depth_Picture/ <= 奥行き画像を格納
            |    |
            |    |- Gousei/ <= 合成した奥行き画像を格納
            |    |
            |    |- MatchingData <= 3: を行ったときに出力されるデータ (移動量,
            |    |                        マッチングの度合い, 標準偏差, 平均値) を格納
            |    |- MatchingGraph/ <= 9: を行ったときの結果を格納
            |                             (移動量 - マッチングのグラフ)
            |
            |- Block11/ <= ブロック 11x11
            |
            |- Block21/ <= ブロック 21x21
```

3.2 データファイル名のフォーマット

- Dct/ : ある画素を中心とした縦横の画素の変化を調べた結果
  dct<x 方向 or y 方向><x>_<y>.dat
  例) 画素 (220,420) について調べた結果を格納しているファイル
  dctx220_420.dat <= x 方向
  dcty220_420.dat <= y 方向

- Depth_Picture/ : 奥行き画像
  cd<v or h><m or p>.dat
  v: 比較画像が鉛直方向
  h: 比較画像が水平方向
  m: 負の向きに移動したカメラより撮影した画像 (鉛直: 上, 水平: 左)
  p: 正の向きに移動したカメラより撮影した画像 (鉛直: 下, 水平: 右)

  したがって, vp は鉛直方向に正の向きに移動したカメラによって撮影
  された画像 (下の画像) を比較画像にしたときのものという意味となる.

・Gousei/ ： 合成した奥行き画像

gou<blocksize><min or bunt>(mf)(sai).pic

min: マッチングの良さのみを考慮

bunt: 分散を考慮

mf: 画像にメディアンフィルタをかけたもの

sai: 細線化したエッジ画像を重ねたもの

例) gou7minmf.pic

ブロックサイズ7x7画素でマッチングの良さのみを考慮したときの合成画像にメディアンフィルタをかけたもの

・MatchingData/ ： 奥行きデータを求めたときに出力される各データ

<v,min,bun,ave><v or h><m or p>.dat

v: 移動量

min: マッチングの度合い

bun: 分散(標準偏差)

ave: 平均値

例) bunhm.dat

左の画像を比較画像としたときに得られる分散データ

・MatchingGraph/ ： マッチングのグラフ

mg<v or h><m or p><x>_<y>.dat

詳細は上記と同じ.

```
#include        <stdio.h>
#include        <stdlib.h>
#include        "params.h"

#define IN 1 /* 入力画像用ウィンドウ */
#define OUT 2 /* 出力画像用ウィンドウ */
#define WORK 3 /* ワーク画像用ウィンドウ */
#define WORK2 4 /* ワーク画像用ウィンドウ */
#define WORK3 5 /* ワーク画像用ウィンドウ */

unsigned char *image_in; /* 入力画像配列 */
unsigned char *image_out; /* 出力画像配列 */
unsigned char *image_work; /* ワーク画像配列 */
unsigned char *image_work2;/* ワーク画像配列 */
unsigned char *image_work3;/* ワーク画像配列 */
unsigned char *code; /* 符号データ */
float *coefh; /* 水平フィルタ係数 */
float *coefv;/* 垂直フィルタ係数 */
float *coefhv; /* 2次元フィルタ係数 */

main(argc, argv)
int argc;
char *argv[];
{
  static int command = -1;
  char source[80];
  char destin[80];
  char codefile[80];
  int m, n;
  float f;
  int h, h1, h2;
  int v, v1, v2;
  float a, b;
  int i, j;
  int ctl;
  int level;
  float loop;
  float fx, fy, px, py, deg;
  long ptr;
  float qn;
  long nbyte;
  char c[128],vfile[80],minfile[80],bunfile[80],avefile[80];
  char dataname[80],outputname[80],cityname[80];
  char cityfile[80],outputfile[80];
  FILE *fpcity,*fpdata,*fpout;
  image_init();
  InitDisplay(argc, argv);

  while (command) {
    puts("### 奥行き情報取得プログラム (Mono) ###\n");
    puts("  1:  ファイルからの画像の読み込み");
    puts("  2:  ファイルへの画像の書き出し");
    puts("  3:  奥行き情報の取得");
    puts("  4:  画像の合成");
    puts("  5:  WORK画像の2値化");
    puts("  6:  Median Filterをかける");
    puts("  7:  奥行きデータの合成(最小値をとる)");
    puts("  8:  奥行きデータの合成(分散を考慮)");
    puts("  9:  移動量ーマッチングの度合いのグラフ");
    puts(" 10:  ブロックの縦と横の変化をしらべる");
    puts(" 11:  外接長方形を求める");
    puts("  0: END");

    printf("    番号を入力してください ?   ");
    scanf("%d", &command);
    switch (command){
```



```
    case 1:
      printf("    読み込む画像のファイル名を入力してください ->   ");
      scanf("%s", source);
      printf("  Which window (1:IN, 2:OUT, 3:WORK, 4:WORK2, 5:WORK3) ? ");
      scanf("%d", &n);
      switch (n) {
      case 1:
        image_read(image_in, X_SIZE, Y_SIZE, source);
        DisplayImage(image_in, IN);
        break;
      case 2:
        image_read(image_out, X_SIZE, Y_SIZE, source);
        DisplayImage(image_out, OUT);
        break;
      case 3:
        image_read(image_work, X_SIZE, Y_SIZE, source);
        DisplayImage(image_work, WORK);
        break;
      case 4:
        image_read(image_work2, X_SIZE, Y_SIZE, source);
        DisplayImage(image_work2, WORK2);
        break;
      case 5:
        image_read(image_work3, X_SIZE, Y_SIZE, source);
        DisplayImage(image_work3, WORK3);
        break;
      default:
        break;
      }
      break;
    case 2:
      printf("  Which window (1:IN, 2:OUT, 3:WORK, 4:WORK2, 5:WORK3) ? ");
      scanf("%d", &n);
      printf("  出力画像を書き込むファイル名を入力してください ->   ");
      scanf("%s", destin);
      switch (n) {
      case 1:
        image_write(image_in, X_SIZE, Y_SIZE, destin);
        break;
      case 2:
        image_write(image_out, X_SIZE, Y_SIZE, destin);
        break;
      case 3:
        image_write(image_work, X_SIZE, Y_SIZE, destin);
        break;
      case 4:
        image_write(image_work2, X_SIZE, Y_SIZE, source);
        DisplayImage(image_work2, WORK2);
        break;                                          >destin
      case 5:
        image_write(image_work3, X_SIZE, Y_SIZE, source);
        DisplayImage(image_work3, WORK3);
        break;
      default:
        break;
      }
      break;

    case 3:
      printf("各種データを保存するファイルの名前を\n 保存しているファイルを入力して
ください\n-> ");
      scanf("%s",dataname);
      if((fpdata = fopen(dataname,"r")) == NULL){
        puts("Bad File Name");
        exit(-1);
      }
```

```
        printf("比較画像のファイル名を保存している\nファイルを入力してください -> ");
        scanf("%s",cityname);
        if((fpcity = fopen(cityname,"r")) == NULL){
          puts("Bad File Name");
          exit(-1);
        }
        printf("奥行き画像をセーブするファイル名を保存している\nファイルを入力してく
ださい -> ");
        scanf("%s",outputname);
        if((fpout = fopen(outputname,"r")) == NULL){
          puts("Bad File Name");
          exit(-1);
        }
        image_read(image_work, X_SIZE, Y_SIZE, "city_5_5.r");
        DisplayImage(image_work, WORK);
        i = 1;
        while(i < 5){
          fscanf(fpdata,"%s %s %s %s",vfile,minfile,bunfile,avefile);
          printf("Read : %s, %s, %s, %s\n",vfile,minfile,bunfile,avefile);
          fscanf(fpcity,"%s",cityfile);
          printf("Read : %s\n",cityfile);
          fscanf(fpout,"%s",outputfile);
          printf("Read : %s\n",outputfile);

          image_read(image_in, X_SIZE, Y_SIZE, cityfile);
          DisplayImage(image_in, IN);
          get_depth(image_in,image_work,image_out,i,
                    vfile,minfile,bunfile,avefile);
          printf("End Cal %d\n",i);
          DisplayImage(image_out,OUT);
          image_write(image_out, X_SIZE, Y_SIZE, outputfile);
          i ++;
        }
        fclose(fpcity);
        fclose(fpdata);
        fclose(fpout);
        break;

      case 4:
        printf("INとWORKに合成する画像を取り込んでますか？(y or n) ");
        scanf("%s", c);
        if (c[0] != 'y' && c[0] != 'Y') break;
        gousei(image_in,image_work,image_out);
        DisplayImage(image_out,OUT);
        break;

      case 5:
        printf("WORKに処理対象の画像を取り込んでますか？");
        nichika(image_in,image_out);
        DisplayImage(image_out,OUT);
        break;

      case 6:
        printf("INに処理対象の画像を取り込んでますか？");
        scanf("%s", c);
        if (c[0] != 'y' && c[0] != 'Y') break;
        medfil(image_in,image_out);
        DisplayImage(image_out,OUT);
        break;

      case 7:
        printf("読み込む奥行きデータファイル名を格納している\nファイルを入力してくだ
さい");
        scanf("%s",dataname);
        if((fpdata = fopen(dataname,"r")) == NULL){
```



```
          puts("Bad File Name !!");
          break;
        }
        fscanf(fpdata,"%s",minfile);
        image_read(image_in, X_SIZE, Y_SIZE, minfile);
        DisplayImage(image_in, IN);
        fscanf(fpdata,"%s",minfile);
        image_read(image_work, X_SIZE, Y_SIZE, minfile);
        DisplayImage(image_work, WORK);
        fscanf(fpdata,"%s",minfile);
        image_read(image_work2, X_SIZE, Y_SIZE, minfile);
        DisplayImage(image_work2, WORK2);
        fscanf(fpdata,"%s",minfile);
        image_read(image_work3, X_SIZE, Y_SIZE, minfile);
        DisplayImage(image_work3, WORK3);

        fclose(fpdata);

        puts("   IN : 奥行き画像 (upper)");
        puts(" WORK : 奥行き画像 (lower)");
        puts("WORK2 : 奥行き画像 (left)");
        puts("WORK3 : 奥行き画像 (right)");

        depth_gousei(image_in,image_out,image_work,image_work2,image_work3);
        DisplayImage(image_out,OUT);

        break;

      case 8:
        printf("読み出す奥行きデータファイル名を格納している\nファイルを入力してくだ
さい");
        scanf("%s",dataname);
        if((fpdata = fopen(dataname,"r")) == NULL){
          puts("Bad File Name !!");
          break;
        }
        fscanf(fpdata,"%s",minfile);
        image_read(image_in, X_SIZE, Y_SIZE, minfile);
        DisplayImage(image_in, IN);
        fscanf(fpdata,"%s",minfile);
        image_read(image_work, X_SIZE, Y_SIZE, minfile);
        DisplayImage(image_work, WORK);
        fscanf(fpdata,"%s",minfile);
        image_read(image_work2, X_SIZE, Y_SIZE, minfile);
        DisplayImage(image_work2, WORK2);
        fscanf(fpdata,"%s",minfile);
        image_read(image_work3, X_SIZE, Y_SIZE, minfile);
        DisplayImage(image_work3, WORK3);

        fclose(fpdata);

        puts("   IN : 奥行き画像 (upper)");
        puts(" WORK : 奥行き画像 (lower)");
        puts("WORK2 : 奥行き画像 (left)");
        puts("WORK3 : 奥行き画像 (right)");
        printf("  Ready [y] ?  ");
        scanf("%s", c);
        if (c[0] != 'y' && c[0] != 'Y') break;
        depth_gousei2(image_in,image_out,image_work,image_work2,image_work3);
        DisplayImage(image_out,OUT);
        break;

      case 9:
        printf("比較画像(1:上 2:下 3:左 4:右) ?");
        scanf("%d",&m);
        printf("結果をセーブするファイル名を入力してください(gnuplot形式)->   ");
```

```
        scanf("%s", source);
        switch(m){
          case 1:
            mat_graph_vm(image_in,image_work,source);
            break;
          case 2:
            mat_graph_vp(image_in,image_work,source);
            break;
          case 3:
            mat_graph_hm(image_in,image_work,source);
            break;
          case 4:
            mat_graph_hp(image_in,image_work,source);
            break;
          default:
            break;
          }
        break;
      case 10:
        printf("INに処理対象の画像を取り込んでますか？");
        scanf("%s", c);
        if (c[0] != 'y' && c[0] != 'Y') break;
        block_xy(image_in);
        break;

      case 11:
        gaisetsu(image_out);
        DisplayImage(image_out,OUT);
        break;

      case 0:
        printf("  Bye-bye ........\n");
        break;

      default:
        printf("  Not defined number\n");
        break;
      }
  }
  EndDisplay();
}

image_init()
/* 画像データ領域の確保と初期化など */
{
  image_in = (unsigned char *)calloc((size_t)X_SIZE*Y_SIZE, sizeof(unsigned char));

  image_out = (unsigned char *)calloc((size_t)X_SIZE*Y_SIZE, sizeof(unsigned char));

  image_work = (unsigned char *)calloc((size_t)X_SIZE*Y_SIZE, sizeof(unsigned char));

  image_work2 = (unsigned char *)calloc((size_t)X_SIZE*Y_SIZE, sizeof(unsigned char))
;
  image_work3 = (unsigned char *)calloc((size_t)X_SIZE*Y_SIZE, sizeof(unsigned char))
;

  code = (unsigned char *)calloc((size_t)X_SIZE*Y_SIZE, sizeof(unsigned char));
  if ((image_in == NULL) || (image_out == NULL) ||
      (image_work == NULL) || (code == NULL) ||
      (image_work2 == NULL)|| (image_work3 == NULL)) {
    printf("Memory not enough. \n");
    exit(0);
  }
  coefh = (float *)calloc((size_t)MAX_TAP, sizeof(float));
  coefv = (float *)calloc((size_t)MAX_TAP, sizeof(float));
  coefhv = (float *)calloc((size_t)MAX_TAP*MAX_TAP, sizeof(float));
```



```
  if ((coefh == NULL) || (coefv == NULL) || (coefhv == NULL)) {
    printf("Memory not enough. \n");
    exit(0);
  }
}

image_read(image, xsize, ysize, filename)
/* ディスクから画像を読み込む */
unsigned char *image; /* 画像配列 */
int xsize; /* 画像横サイズ */
int ysize; /* 画像縦サイズ */
char *filename; /* ファイル名 */
{
  FILE *fp;

 if ((fp = fopen(filename, "rb")) == NULL) {
   printf(" open error (read) \n");
   exit(-1);
 }
  fread(image, (size_t)xsize, (size_t)ysize, fp);
  fclose(fp);
}

image_write(image, xsize, ysize, filename)
/* ディスクに画像を書き出す */
unsigned char *image; /* 画像配列 */
int xsize; /* 画像横サイズ */
int  ysize; /* 画像縦サイズ */
char *filename; /* ファイル名 */
{
  FILE *fp;

  if ((fp = fopen(filename, "wb")) == NULL) {
    printf(" open error (write)\n");
    exit(-1);
  }
  fwrite(image, (size_t)xsize, (size_t)ysize, fp);
  fclose(fp);
}

image_copy(image_in, image_out)
/* モノクロ画像データをコピーする        */
unsigned char *image_in; /* 入力画像配列 */
unsigned char *image_out; /* 出力画像配列 */
{
  long i;

  for (i = 0; i < (long)Y_SIZE*X_SIZE; i++)
    image_out[i] = image_in[i];
}

image_clear(image)
/* モノクロ画像データをクリアする        */
unsigned char *image; /* 入力画像配列 */
{
  long i;

  for(i = 0; i < (long)Y_SIZE*X_SIZE; i++)
    image[i] = 0;
}
```

```
#include <stdio.h>
#include <math.h>
#include "params.h"
#include "mc.h"

mvec_mine_hm(image_pr,image_ct,vx,minno,bunno,aveno)
/* 移動量を求める : 比較画像が左の画像 */
unsigned char image_pr[Y_SIZE][X_SIZE];
unsigned char image_ct[Y_SIZE][X_SIZE];
int vx[Y_SIZE][X_SIZE];
unsigned int minno[Y_SIZE][X_SIZE];
unsigned int bunno[Y_SIZE][X_SIZE];
unsigned int aveno[Y_SIZE][X_SIZE];
{
  int i,j,dx,ddx,dy,ddy,x,xs,xd,y,yd,ys;
  float d, min, ds, ave_d, dss, bunsan;

  ave_d = 0.0;
  for(i = 0; i < Y_SIZE; i++){      /* 画素(j,i) */
    for(j = 0; j < X_SIZE; j++){
      ds = dss = 0.0;
      for(dx = 0; dx <= DX_MAX; dx++){  /* ブロックを動かす */
        d = 0.0;
        for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){ /* ブロック内の画素を調べる */
          yd  = i + y;
          if(yd < 0)
            yd = 0;
          if(yd > Y_SIZE-1)
            yd = Y_SIZE-1;

          for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
            xd = j + x;
            if(xd < 0)
              xd = 0;
            if(xd > X_SIZE-1)
              xd = X_SIZE-1;
            xs = xd + dx;
            if(xs < 0)
              xs = 0;
            if(xs > X_SIZE-1)
              xs = X_SIZE-1;

            d += (float)abs((int)(image_ct[yd][xd] - image_pr[yd][xs]));
            /* マッチングの度合い */
          }
        }

        ds +=d; dss += d*d;

        if(dx == 0){
          min = d;
          ddx = dx;
        }
        if(d <= min){
          min = d;
          ddx = dx;
        }
      }

      ave_d = ds/(float)(DX_MAX*2+1);
      bunsan = dss/(float)(DX_MAX*2+1) - ave_d*ave_d;
      vx[i][j] = ddx;
      minno[i][j] = (unsigned int)min;
      bunno[i][j] = (unsigned int)sqrt((double)bunsan);
      aveno[i][j] = (unsigned int)ave_d;
    }
```



```
  }
}
mvec_mine_hp(image_pr,image_ct,vx,minno,bunno,aveno)
/* 移動量を求める : 比較画像が右の画像 */
unsigned char image_pr[Y_SIZE][X_SIZE];
unsigned char image_ct[Y_SIZE][X_SIZE];
int vx[Y_SIZE][X_SIZE];
unsigned int minno[Y_SIZE][X_SIZE];
unsigned int bunno[Y_SIZE][X_SIZE];
unsigned int aveno[Y_SIZE][X_SIZE];
{
  int i,j,dx,ddx,dy,ddy,x,xs,xd,y,yd,ys;
  float d, min, ds, ave_d, dss, bunsan;

  ave_d = 0.0;
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      ds = dss = 0.0;
      for(dx = -DX_MAX; dx <= 0; dx++){
        d = 0.0;
        for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){
          yd  = i + y;
          if(yd < 0)
            yd = 0;
          if(yd > Y_SIZE-1)
            yd = Y_SIZE-1;

          for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
            xd = j + x;
            if(xd < 0)
              xd = 0;
            if(xd > X_SIZE-1)
              xd = X_SIZE-1;
            xs = xd + dx;
            if(xs < 0)
              xs = 0;
            if(xs > X_SIZE-1)
              xs = X_SIZE-1;

            d += (float)abs((int)(image_ct[yd][xd] - image_pr[yd][xs]));
          }
        }

        ds +=d; dss += d*d;

        if(dx == -DX_MAX){
          min = d;
          ddx = dx;
        }
        if(d <= min){
          min = d;
          ddx = dx;
        }
      }

      ave_d = ds/(float)(DX_MAX*2+1);
      bunsan = dss/(float)(DX_MAX*2+1) - ave_d*ave_d;
      vx[i][j] = ddx;
      minno[i][j] = (unsigned int)min;
      bunno[i][j] = (unsigned int)sqrt((double)bunsan);
      aveno[i][j] = (unsigned int)ave_d;
    }
  }
}
```

```
#include <math.h>
#include "params.h"
#include "mc.h"

mvec_mine_vm(image_pr,image_ct,vy,minno,bunno,aveno)
/* 移動量を求める : 比較画像が上の画像 */
unsigned char image_pr[Y_SIZE][X_SIZE];
unsigned char image_ct[Y_SIZE][X_SIZE];
int vy[Y_SIZE][X_SIZE];
unsigned int minno[Y_SIZE][X_SIZE];
unsigned int bunno[Y_SIZE][X_SIZE];
unsigned int aveno[Y_SIZE][X_SIZE];
{
  int i, j, dx, ddx, dy, ddy;
  int x, xs, xd, y, yd, ys;
  float d, min, ds, ave_d, dss, bunsan;

  ave_d = 0.0;
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      ds = dss = 0.0;
      for(dy = 0; dy <= DY_MAX; dy++){
        d = 0.0;
        for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){
          yd  = i + y;
          if(yd < 0)
            yd = 0;
          if(yd > Y_SIZE-1)
            yd = Y_SIZE-1;
          ys = yd + dy;
          if(ys < 0)
            ys = 0;
          if(ys > Y_SIZE-1)
            ys = Y_SIZE-1;

          for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
            xd = j + x;
            if(xd < 0)
              xd = 0;
            if(xd > X_SIZE-1)
              xd = X_SIZE-1;

            d += (float)abs((int)(image_ct[yd][xd] - image_pr[ys][xd]));
          }
        }
        ds += d; dss += d*d;

        if(dy == 0){
          min = d;
          ddy = dy;
        }
        if(d <= min){
          min = d;
          ddy = dy;
        }
      }
      ave_d = ds/(float)(DY_MAX*2+1);
      bunsan = dss/(float)(DY_MAX*2+1) - ave_d*ave_d;
      vy[i][j] = ddy;
      minno[i][j] = (unsigned int)min;
      bunno[i][j] = (unsigned int)sqrt((double)bunsan);
      aveno[i][j] = (unsigned int)ave_d;
    }
  }
}

mvec_mine_vp(image_pr,image_ct,vy,minno,bunno,aveno)
```



```
/* 移動量を求める : 比較画像が上の画像 */
unsigned char image_pr[Y_SIZE][X_SIZE];
unsigned char image_ct[Y_SIZE][X_SIZE];
int vy[Y_SIZE][X_SIZE];
unsigned int minno[Y_SIZE][X_SIZE];
unsigned int bunno[Y_SIZE][X_SIZE];
unsigned int aveno[Y_SIZE][X_SIZE];
{
  int i, j, dx, ddx, dy, ddy;
  int x, xs, xd, y, yd, ys;
  float d, min, ds, ave_d, dss, bunsan;

  ave_d = 0.0;
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      ds = dss = 0.0;
      for(dy = -DY_MAX; dy <= 0; dy++){
        d = 0.0;
        for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){
          yd  = i + y;
          if(yd < 0)
            yd = 0;
          if(yd > Y_SIZE-1)
            yd = Y_SIZE-1;
          ys = yd + dy;
          if(ys < 0)
            ys = 0;
          if(ys > Y_SIZE-1)
            ys = Y_SIZE-1;

          for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
            xd = j + x;
            if(xd < 0)
              xd = 0;
            if(xd > X_SIZE-1)
              xd = X_SIZE-1;

            d += (float)abs((int)(image_ct[yd][xd] - image_pr[ys][xd]));
          }
        }
        ds += d; dss += d*d;

        if(dy == -DY_MAX){
          min = d;
          ddy = dy;
        }
        if(d <= min){
          min = d;
          ddy = dy;
        }
      }
      ave_d = ds/(float)(DY_MAX*2+1);
      bunsan = dss/(float)(DY_MAX*2+1) - ave_d*ave_d;
      vy[i][j] = ddy;
      minno[i][j] = (unsigned int)min;
      bunno[i][j] = (unsigned int)sqrt((double)bunsan);
      aveno[i][j] = (unsigned int)ave_d;
    }
  }
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "params.h"
#include "mc.h"

depth_gousei(image_up,image_rt,image_lw,image_lf,image_rg)
/* 4つの奥行き画像を合成する(マッチングの良さのみを考慮) */
unsigned char image_up[Y_SIZE][X_SIZE];
unsigned char image_rt[Y_SIZE][X_SIZE];
unsigned char image_lw[Y_SIZE][X_SIZE];
unsigned char image_lf[Y_SIZE][X_SIZE];
unsigned char image_rg[Y_SIZE][X_SIZE];
{
  int i,j,min_data;
  unsigned int up_data[Y_SIZE][X_SIZE];
  unsigned int lw_data[Y_SIZE][X_SIZE];
  unsigned int lf_data[Y_SIZE][X_SIZE];
  unsigned int rg_data[Y_SIZE][X_SIZE];
  unsigned int out_data;
  FILE *fp,*fpdata;
  char bel_dat_file[80],name[80];

  puts("マッチングの度合いが格納されている");
  printf("ファイルを入力してください ->");
  scanf("%s",name);
  if((fpdata = fopen(name,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }

  fscanf(fpdata,"%s",bel_dat_file);
  printf("Input Upper Data File ->%s\n",bel_dat_file);

  if((fp = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fp,"%d",&up_data[i][j]);
    }
  }
  fclose(fp);

  fscanf(fpdata,"%s",bel_dat_file);
  printf("Input Lower Data File ->%s\n",bel_dat_file);
  if((fp = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fp,"%d",&lw_data[i][j]);
    }
  }
  fclose(fp);

  fscanf(fpdata,"%s",bel_dat_file);
  printf("Input Left Data File ->%s\n",bel_dat_file);
  if((fp = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fp,"%d",&lf_data[i][j]);
```



```
    }
  }
  fclose(fp);

  fscanf(fpdata,"%s",bel_dat_file);
  printf("Input Right Data File ->%s\n",bel_dat_file);
  if((fp = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fp,"%d",&rg_data[i][j]);
    }
  }
  fclose(fp);

  fp = fopen("min.dat","w");

  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      min_data = up_data[i][j];
      out_data = image_up[i][j];
      if(lw_data[i][j] < min_data){
        min_data = lw_data[i][j];
        out_data = image_lw[i][j];
      }
      if(lf_data[i][j] < min_data){
        min_data = lf_data[i][j];
        out_data = image_lf[i][j];
      }
      if(rg_data[i][j] < min_data){
        min_data = rg_data[i][j];
        out_data = image_rg[i][j];
      }
      fprintf(fp,"%d\n",min_data);
      image_rt[i][j] = out_data;
    }
  }
  fclose(fp);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "math.h"
#include "params.h"
#include "mc.h"

depth_gousei2(image_up,image_rt,image_lw,image_lf,image_rg)
/* 4つの奥行き画像を合成する(分散を考慮) */
unsigned char image_up[Y_SIZE][X_SIZE];
unsigned char image_rt[Y_SIZE][X_SIZE];
unsigned char image_lw[Y_SIZE][X_SIZE];
unsigned char image_lf[Y_SIZE][X_SIZE];
unsigned char image_rg[Y_SIZE][X_SIZE];
{
  int i,j,k,min_data,min_kouho[4],image_kouho[4],min_flg,max_data,docchi;
  unsigned int v_up_data[Y_SIZE][X_SIZE];
  unsigned int v_lw_data[Y_SIZE][X_SIZE];
  unsigned int v_lf_data[Y_SIZE][X_SIZE];
  unsigned int v_rg_data[Y_SIZE][X_SIZE];
  unsigned int min_up_data[Y_SIZE][X_SIZE];
  unsigned int min_lw_data[Y_SIZE][X_SIZE];
  unsigned int min_lf_data[Y_SIZE][X_SIZE];
  unsigned int min_rg_data[Y_SIZE][X_SIZE];
  unsigned int bun_up_data[Y_SIZE][X_SIZE];
  unsigned int bun_lw_data[Y_SIZE][X_SIZE];
  unsigned int bun_lf_data[Y_SIZE][X_SIZE];
  unsigned int bun_rg_data[Y_SIZE][X_SIZE];
  unsigned int out_data;
  int bunthre,vthre;
  double bun_v,ave_v;
  FILE *fpmin,*fpbun,*fpdata;
  char bel_dat_file[80],name[80];
/*  char *bel_dat_file; */

  puts("マッチングの度合いの標準偏差の閾値と");
  puts("移動量の標準偏差の閾値を格納している");
  printf("ファイルを入力してください ->");
  scanf("%s",name);
  if((fpdata = fopen(name,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  fscanf(fpdata,"%d %d",&bunthre,&vthre);
  fclose(fpdata);

  puts("各データが格納されている");
  printf("ファイルを入力してください ->");
  scanf("%s",name);
  if((fpdata = fopen(name,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }

  puts("移動量データを読み出しています ......");
  printf("Input Upper Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpmin = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
      exit(-1);
    }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpmin,"%d",&v_up_data[i][j]);
    }
  }
```



```
  fclose(fpmin);
  printf("Input Lower Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpmin = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpmin,"%d",&v_lw_data[i][j]);
    }
  }
  fclose(fpmin);

  printf("Input Left Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpmin = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpmin,"%d",&v_lf_data[i][j]);
    }
  }
  fclose(fpmin);

  printf("Input Right Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpmin = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpmin,"%d",&v_rg_data[i][j]);
    }
  }
  fclose(fpmin);

  puts("マッチングの度合いを読み出しています ......");
  printf("Input Upper Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpmin = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpmin,"%d",&min_up_data[i][j]);
    }
  }
  fclose(fpmin);
  printf("Input Lower Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpmin = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
```

```
      fscanf(fpmin,"%d",&min_lw_data[i][j]);
    }
  }
  fclose(fpmin);
  printf("Input Left Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpmin = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpmin,"%d",&min_lf_data[i][j]);
    }
  }
  fclose(fpmin);
  printf("Input Right Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpmin = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpmin,"%d",&min_rg_data[i][j]);
    }
  }
  fclose(fpmin);

  puts("標準偏差のデータを読み出しています .....");
  printf("Input Upper Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpbun = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpbun,"%d",&bun_up_data[i][j]);
    }
  }
  fclose(fpbun);
  printf("Input Lower Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpbun = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpbun,"%d",&bun_lw_data[i][j]);
    }
  }
  fclose(fpbun);
  printf("Input Left Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpbun = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
```



```
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpbun,"%d",&bun_lf_data[i][j]);
    }
  }
  fclose(fpbun);
  printf("Input Right Data File ->");
  fscanf(fpdata,"%s",bel_dat_file);
  printf("%s\n",bel_dat_file);
  if((fpbun = fopen(bel_dat_file,"r")) == NULL){
    puts("Bad File Name!");
    exit(-1);
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fpbun,"%d",&bun_rg_data[i][j]);
    }
  }
  fclose(fpbun);
  fclose(fpdata);
  fpmin = fopen("min.dat","w");

  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      min_flg = 0;
      if(bun_up_data[i][j] > bunthre){
        min_kouho[min_flg] = min_up_data[i][j];
        image_kouho[min_flg] = image_up[i][j];
        min_flg ++;
      }
      if(bun_lw_data[i][j] > bunthre){
        min_kouho[min_flg] = min_lw_data[i][j];
        image_kouho[min_flg] = image_lw[i][j];
        min_flg++;
      }
      if(bun_lf_data[i][j] > bunthre){
        min_kouho[min_flg] = min_lf_data[i][j];
        image_kouho[min_flg] = image_lf[i][j];
        min_flg++;
      }
      if(bun_rg_data[i][j] > bunthre){
        min_kouho[min_flg] = min_rg_data[i][j];
        image_kouho[min_flg] = image_rg[i][j];
        min_flg++;
      }

      if(min_flg == 0){
        ave_v = (v_up_data[i][j] - v_lw_data[i][j] + v_lf_data[i][j]
                - v_rg_data[i][j])/4;
        bun_v = sqrt(((double)(v_up_data[i][j]*v_up_data[i][j]
                              + v_lw_data[i][j]*v_lw_data[i][j]
                              + v_lf_data[i][j]*v_lf_data[i][j]
                              + v_rg_data[i][j]*v_rg_data[i][j])/4)
                     - ave_v*ave_v);

        if(bun_v < vthre){
          min_data = min_up_data[i][j];
          out_data = image_up[i][j];
          if(min_lw_data[i][j] < min_data){
            min_data = min_lw_data[i][j];
            out_data = image_lw[i][j];
          }
          if(min_lf_data[i][j] < min_data){
            min_data = min_lf_data[i][j];
            out_data = image_lf[i][j];
          }
          if(min_rg_data[i][j] < min_data){
```

```
            min_data = min_rg_data[i][j];
            out_data = image_rg[i][j];
          }
          image_rt[i][j] = out_data;
        }
        else
          image_rt[i][j] = LOW;
      }
      else{
        min_data = min_kouho[0];
        image_rt[i][j] = image_kouho[0];
        for(k = 1; k < min_flg; k++){
          if(min_kouho[k] < min_data){
            min_data = min_kouho[k];
            image_rt[i][j] = image_kouho[k];
          }
        }
      }
    }
  }
}
```

```
#include <stdio.h>
#include "math.h"
#include "params.h"
#include "mc.h"

#define PAI 3.1415936535898

block_xy(image)
/* ある画素を中心とした縦方向と横方向の変化を求める */
unsigned char  image[Y_SIZE][X_SIZE];
{
  int i,j,y,x,mx,my,u;
  unsigned char f[X_BLOCK];
  char x_data_file[80],y_data_file[80];
  double dd,cu,F[X_BLOCK];
  FILE *fpx,*fpy,*fp;

  printf("水平方向の結果を格納するファイル => ");
  scanf("%s",x_data_file);
  printf("鉛直方向の結果を格納するファイル => ");
  scanf("%s",y_data_file);

  fpx = fopen(x_data_file,"w");
  fpy = fopen(y_data_file,"w");
  fp = fopen("data_file","a");

  printf("xi yi ?");
  scanf("%d %d",&mx,&my);
  fprintf(fp,"(%d,%d)\n",mx,my);
  fprintf(fp,"y\n");
  for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){
    f[y + Y_BLOCK/2] = image[my+y][mx];
    fprintf(fp,"%d ",f[y + Y_BLOCK/2]);
  }
  fprintf(fp,"\n");

  for(u = 0; u < Y_BLOCK; u++){
    if(u == 0)
      cu = 1.0/sqrt(2.0);
    else
      cu = 1.0;
    dd = 0.0;
    for(i = 0; i < Y_BLOCK; i++){
      dd += f[i]*cos((2*i+1)*u*PAI/(2*Y_BLOCK));
    }
    F[u] = 2.0*dd*cu/Y_BLOCK;
    fprintf(fpy,"%d %lf\n",u,F[u]);
    fprintf(fp,"%d %lf\n",u,F[u]);
  }
  fprintf(fp,"x\n");
  for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
    f[x + X_BLOCK/2] = image[my][mx + x];
    fprintf(fp,"%d ",f[x + X_BLOCK/2]);
  }
  fprintf(fp,"\n");

  for(u = 0; u < X_BLOCK; u++){
    if(u == 0)
      cu = 1.0/sqrt(2.0);
    else
      cu = 1.0;
    dd = 0.0;
    for(i = 0; i < X_BLOCK; i++){
      dd += f[i]*cos((2*i+1)*u*PAI/(2*X_BLOCK));
    }
    F[u] = 2.0*dd*cu/X_BLOCK;
```



```
    fprintf(fpx,"%d %lf\n",u,F[u]);
    fprintf(fp,"%d %lf\n",u,F[u]);
  }
  fprintf(fp,"\n");
  fclose(fpx);
  fclose(fp);
  fclose(fpy);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "params.h"
#include "mc.h"

get_depth(image_pr,image_ct,image_rt,com,vfile,minfile,bunfile,avefile)
/* 画素の奥行き情報を求め,表示する */
unsigned char image_pr[Y_SIZE][X_SIZE]; /* 前フレームの画像 */
unsigned char image_ct[Y_SIZE][X_SIZE]; /* 現フレームの画像 */
unsigned char image_rt[Y_SIZE][X_SIZE]; /* 出力画像 */
int com;
char *vfile;   /* 移動量を保存するファイル */
char *minfile; /* マッチングの度合いを保存するファイル */
char *bunfile; /* 分散(標準偏差)を保存するファイル */
char *avefile; /* 平均値を保存するファイル */

{
  int *vno; /* 移動量のデータ */
  int porm,docchi;
  unsigned int *minno,*bunno,*aveno;

  vno = (int *)malloc(sizeof(int)*X_SIZE*Y_SIZE);
  minno = (unsigned int *)malloc(sizeof(unsigned int)*X_SIZE*Y_SIZE);
  bunno = (unsigned int *)malloc(sizeof(unsigned int)*X_SIZE*Y_SIZE);
  aveno = (unsigned int *)malloc(sizeof(unsigned int)*X_SIZE*Y_SIZE);

  if(vno == NULL || minno == NULL || bunno == NULL || aveno == NULL){
    puts("Memory not enough.");
    exit(-1);
  }
  if(com == 1){ /* 比較画像が上の画像 */
    mvec_mine_vm(image_pr,image_ct,vno,minno,bunno,aveno);
    porm = 2;
  }
  if(com == 2){ /* 比較画像が下の画像 */
    mvec_mine_vp(image_pr,image_ct,vno,minno,bunno,aveno);
    porm = 1;
  }
  if(com == 3){ /* 比較画像が左の画像 */
    mvec_mine_hm(image_pr,image_ct,vno,minno,bunno,aveno);
    porm = 2;
  }
  if(com == 4){ /* 比較画像が右の画像 */
    mvec_mine_hp(image_pr,image_ct,vno,minno,bunno,aveno);
    porm = 1;
  }

  cal_depth(vno,minno,bunno,aveno,image_rt,porm,vfile,minfile,bunfile,avefile);

  free(vno);
  free(minno);
  free(bunno);
  free(aveno);
}

cal_depth(vno,minno,bunno,aveno,depth_data,porm,vfile,minfile,bunfile,avefile)
/* 奥行き情報の計算 */
int vno[Y_SIZE][X_SIZE];
unsigned int minno[Y_SIZE][X_SIZE];
unsigned int bunno[Y_SIZE][X_SIZE];
unsigned int aveno[Y_SIZE][X_SIZE];
unsigned char depth_data[Y_SIZE][X_SIZE];
int porm;
char vfile[80];
char minfile[80];
char bunfile[80];
```



```
char avefile[80];
{
  int i,j;
  FILE *fpv,*fpmin,*fpbun,*fpave;

  fpv = fopen(vfile,"w");
  fpmin = fopen(minfile,"w");
  fpbun = fopen(bunfile,"w");
  fpave = fopen(avefile,"w");

  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      if(porm == 1){
        if(vno[i][j] > 0)
          depth_data[i][j] = 0;
        else
          depth_data[i][j] = (unsigned char)(255 - (vno[i][j] + DX_MAX) * 255 / DX_MA
X);
      }
      if(porm == 2){
        if(vno[i][j] < 0)
          depth_data[i][j] = 0;
        else
          depth_data[i][j] = (unsigned char)(vno[i][j]* 255 / DX_MAX);
      }
      fprintf(fpv,"%d\n",vno[i][j]);
      fprintf(fpmin,"%d\n",minno[i][j]);
      fprintf(fpbun,"%d\n",bunno[i][j]);
      fprintf(fpave,"%d\n",aveno[i][j]);
    }
  }
  fclose(fpv);
  fclose(fpmin);
  fclose(fpbun);
  fclose(fpave);
}
```

```
#include <stdio.h>
#include "params.h"
#include "mc.h"

gaisetsu(image_rt)
/* 外接長方形を求める */
unsigned char image_rt[Y_SIZE][X_SIZE];
{
  int max_y[LEVEL];
  int min_y[LEVEL];
  int max_x[LEVEL];
  int min_x[LEVEL];
  int max_label,min_label,i,j,k,l;
  int label[Y_SIZE][X_SIZE],crlabel,dum[Y_SIZE][X_SIZE];
  FILE *fp;

  fp = fopen("labelcut.dat","r");
  min_label = 255; max_label = 0;

  for(i = 0; i < LEVEL; i++){
    max_y[i] = max_x[i] = -1;
    min_y[i] = Y_SIZE+1;
    min_x[i] = X_SIZE+1;
  }
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fscanf(fp,"%d",&label[i][j]);
      dum[i][j] = label[i][j];
      if(label[i][j] > max_label)
        max_label = label[i][j];
      if(label[i][j] < min_label && label[i][j] != 0)
        min_label = label[i][j];
      image_rt[i][j] = LOW;
    }
  }
  fclose(fp);

  printf("%d %d\n",min_label,max_label);

  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      crlabel = label[i][j];
      if(crlabel >= min_label && crlabel <= max_label){
        if(i > max_y[crlabel]){
          max_y[crlabel] = i;
        }
        if(i < min_y[crlabel]){
          min_y[crlabel] = i;
        }
        if(j > max_x[crlabel]){
          max_x[crlabel] = j;
        }

        if(j < min_x[crlabel]){
          min_x[crlabel] = j;
        }

      }
    }
  }
  for(k = min_label; k <= max_label; k++){
    max_y[k]++;
    if(max_y[k] > Y_SIZE)
      max_y[k] = Y_SIZE;
    max_x[k]++;
    if(max_x[k] > X_SIZE)
```



```
      max_x[k] = X_SIZE;
    min_y[k]++;
    if(min_y[k] < 0)
      min_y[k] = 0;
    min_x[k]++;
    if(min_x[k] < 0)
      min_x[k] = 0;
  }


  for(k = min_label; k <= max_label; k++){
    for(i = min_y[k]; i <= max_y[k]; i++){
      for(j = min_x[k]; j <= max_x[k]; j++){
        image_rt[i][j] = HIGH;
        dum[i][j] = k;
      }
    }
  }
  puts("tuuka");
  fp = fopen("labelsq.dat","w");
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      fprintf(fp,"%d\n",dum[i][j]);
    }
  }

  fclose(fp);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "params.h"
#include "mc.h"

nichika(image,image_rt)
/* 画像を2値化する */
unsigned char image[Y_SIZE][X_SIZE];
unsigned char image_rt[Y_SIZE][X_SIZE];
{
  int i,j,thre;

  printf("Threshold ?");
  scanf("%d",&thre);
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      if(i < 50)
        image_rt[i][j] = 0;
      else{
        if(image[i][j] < thre)
          image_rt[i][j] = 255;
        else
          image_rt[i][j] = 0;
      }
    }
  }
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "params.h"
#include "mc.h"

mat_graph_hm(image_pr,image_ct,datafile)
/* マッチングのグラフを求める(比較画像が左の画像) */
unsigned char image_pr[Y_SIZE][X_SIZE];
unsigned char image_ct[Y_SIZE][X_SIZE];
char *datafile;
{
  int i, j, dx, ddx, dy, ddy;
  int x, xs, xd, y, yd, ys, cnt_mat,mx,my;
  float d, min, ds, ave_d;
  FILE *fp;

  printf("xi yj ? ");
  scanf("%d %d",&mx,&my);

  fp = fopen(datafile,"w");
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      if(j == mx && i == my){
        for(dx = 0; dx <= DX_MAX; dx++){
          d = 0.0;
          for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){
            yd  = i + y;
            if(yd < 0)
              yd = 0;
            if(yd > Y_SIZE-1)
              yd = Y_SIZE-1;

            for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
              xd = j + x;
              if(xd < 0)
                xd = 0;
              if(xd > X_SIZE-1)
                xd = X_SIZE-1;
              xs = xd + dx;
              if(xs < 0)
                xs = 0;
              if(xs > X_SIZE-1)
                xs = X_SIZE-1;

              d += (float)abs((int)(image_ct[yd][xd] - image_pr[yd][xs]));
            }
          }
          fprintf(fp,"%d %f\n",dx,d);
        }
      }
    }
  }
  fclose(fp);
}

mat_graph_hp(image_pr,image_ct,datafile)
/* マッチングのグラフを求める(比較画像が右の画像) */
unsigned char image_pr[Y_SIZE][X_SIZE];
unsigned char image_ct[Y_SIZE][X_SIZE];
char *datafile;
{
  int i, j, dx, ddx, dy, ddy;
  int x, xs, xd, y, yd, ys, cnt_mat,mx,my;
  float d, min, ds, ave_d;
  FILE *fp;
```



```
  printf("xi yj ? ");
  scanf("%d %d",&mx,&my);

  fp = fopen(datafile,"w");
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      if(j == mx && i == my){
        for(dx = -DX_MAX; dx <= 0; dx++){
          d = 0.0;
          for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){
            yd  = i + y;
            if(yd < 0)
              yd = 0;
            if(yd > Y_SIZE-1)
              yd = Y_SIZE-1;

            for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
              xd = j + x;
              if(xd < 0)
                xd = 0;
              if(xd > X_SIZE-1)
                xd = X_SIZE-1;
              xs = xd + dx;
              if(xs < 0)
                xs = 0;
              if(xs > X_SIZE-1)
                xs = X_SIZE-1;

              d += (float)abs((int)(image_ct[yd][xd] - image_pr[yd][xs]));
            }
          }
          fprintf(fp,"%d %f\n",dx,d);
        }
      }
    }
  }
  fclose(fp);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "params.h"
#include "mc.h"

mat_graph_vm(image_pr,image_ct,datafile)
/* マッチングのグラフを求める(比較画像が上の画像) */
unsigned char image_pr[Y_SIZE][X_SIZE];
unsigned char image_ct[Y_SIZE][X_SIZE];
char *datafile;
{
  int i, j, dx, ddx, dy, ddy;
  int x, xs, xd, y, yd, ys, cnt_mat,mx,my;
  float d, min, ds, ave_d;
  FILE *fp;

  printf("xi yj ? ");
  scanf("%d %d",&mx,&my);

  fp = fopen(datafile,"w");
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      if(j == mx && i == my){
        for(dy = 0; dy <= DY_MAX; dy++){
          d = 0.0;
          for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){
            yd  = i + y;
            if(yd < 0)
              yd = 0;
            if(yd > Y_SIZE-1)
              yd = Y_SIZE-1;
              ys = yd + dy;
              if(ys < 0)
                ys = 0;
              if(ys > Y_SIZE-1)
                ys = Y_SIZE-1;

            for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
              xd = j + x;
              if(xd < 0)
                xd = 0;
              if(xd > X_SIZE-1)
                xd = X_SIZE-1;

              d += (float)abs((int)(image_ct[yd][xd] - image_pr[ys][xd]));
            }
          }
          fprintf(fp,"%d %f\n",dy,d);
        }
      }
    }
  }
  fclose(fp);

}
mat_graph_vp(image_pr,image_ct,datafile)
/* マッチングのグラフを求める(比較画像が下の画像) */
unsigned char image_pr[Y_SIZE][X_SIZE];
unsigned char image_ct[Y_SIZE][X_SIZE];
char *datafile;
{
  int i, j, dx, ddx, dy, ddy;
  int x, xs, xd, y, yd, ys, cnt_mat,mx,my;
  float d, min, ds, ave_d;
  FILE *fp;
```



```
  printf("xi yj ? ");
  scanf("%d %d",&mx,&my);

  fp = fopen(datafile,"w");
  for(i = 0; i < Y_SIZE; i++){
    for(j = 0; j < X_SIZE; j++){
      if(j == mx && i == my){
        for(dy = -DY_MAX; dy <= 0; dy++){
          d = 0.0;
          for(y = -Y_BLOCK/2; y <= Y_BLOCK/2; y++){
            yd  = i + y;
            if(yd < 0)
              yd = 0;
            if(yd > Y_SIZE-1)
              yd = Y_SIZE-1;
              ys = yd + dy;
              if(ys < 0)
                ys = 0;
              if(ys > Y_SIZE-1)
                ys = Y_SIZE-1;

            for(x = -X_BLOCK/2; x <= X_BLOCK/2; x++){
              xd = j + x;
              if(xd < 0)
                xd = 0;
              if(xd > X_SIZE-1)
                xd = X_SIZE-1;

              d += (float)abs((int)(image_ct[yd][xd] - image_pr[ys][xd]));
            }
          }
          fprintf(fp,"%d %f\n",dy,d);
        }
      }
    }
  }
  fclose(fp);

}
```

```
#include "params.h"
#include "mc.h"

medfil(image_in, image_out)
/*  ノイズ除去  */
unsigned char image_in[Y_SIZE][X_SIZE]; /* 入力画像配列 */
unsigned char image_out[Y_SIZE][X_SIZE]; /* 出力画像配列 */
{
        int i, j, k,tiMes;
        unsigned char c[9],dummy[Y_SIZE][X_SIZE];

        printf("times ? ");
        scanf("%d",&tiMes);

        for(i = 0; i < Y_SIZE; i++){
                for(j = 0; j < X_SIZE; j++){
                        dummy[i][j] = image_in[i][j];
                }
        }

        for(k = 0; k < tiMes; k++){
                for (i = 1; i < Y_SIZE-1; i++) {
                        for (j = 1; j < X_SIZE-1; j++) {
                                c[0] = dummy[i-1][j-1];
                                c[1] = dummy[i-1][j];
                                c[2] = dummy[i-1][j+1];
                                c[3] = dummy[i][j-1];
                                c[4] = dummy[i][j];
                                c[5] = dummy[i][j+1];
                                c[6] = dummy[i+1][j-1];
                                c[7] = dummy[i+1][j];
                                c[8] = dummy[i+1][j+1];

                                image_out[i][j] = median(c); /* メディアンフィルタ */
                                dummy[i][j] = image_out[i][j];
                        }
                }
        }
}
median(c)
/*  9つの画素の中央値（メディアン）を求める */
unsigned char   c[];
{
        int     i, j, buf;

        for (j = 0; j < 8; j++) {
          for (i = 0; i < 8; i++) {
            if (c[i+1] < c[i]) {
                buf = c[i+1];
                c[i+1] = c[i];
                c[i] = buf;
            }
          }
        }
        return(c[4]);
}
```