動画像処理

動画像処理の目的

- 産業用画像処理：動画像を用いた欠陥検査・非破壊検査
- 監視カメラ映像の処理：不審者・不審物の検知・異常検知
- 自動車用画像処理：車載カメラで撮影した動画像の処理
- 医用画像処理：肺・心臓などの臓器の動き，関節の動きの解析
- ロボットビジョン：視覚センサをもつロボットの３次元環境認識

本講義の目的

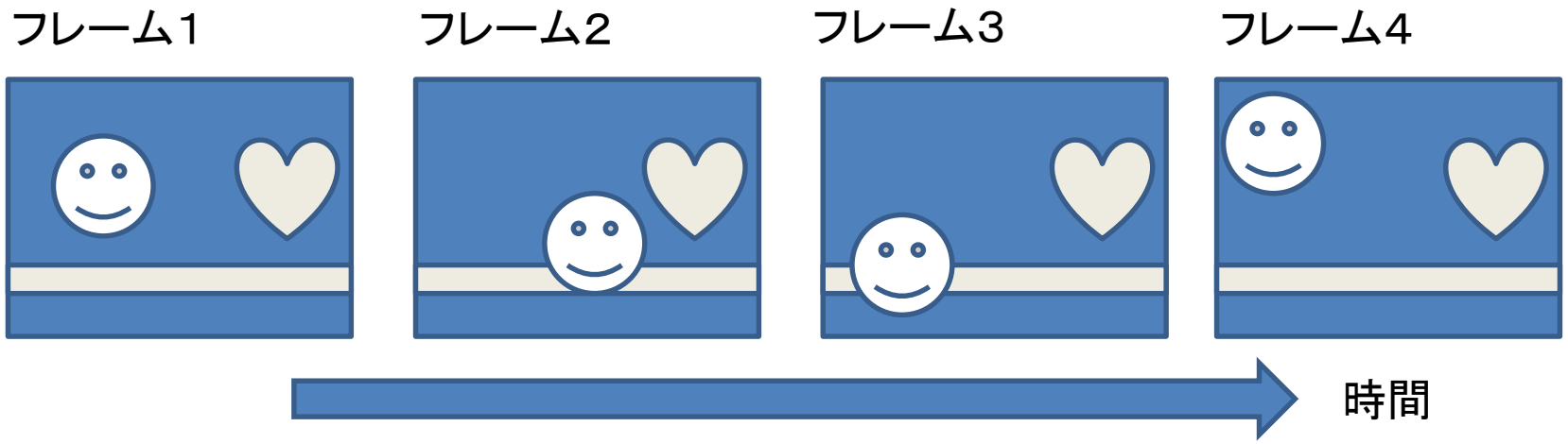
- 動画像からの背景画像の作成
- 背景画像との差分による移動物体の抽出
- 移動軌跡の算出

本講義の目的

- 動画像からの背景画像の作成
- 背景画像との差分による移動物体の抽出
- 移動軌跡の算出

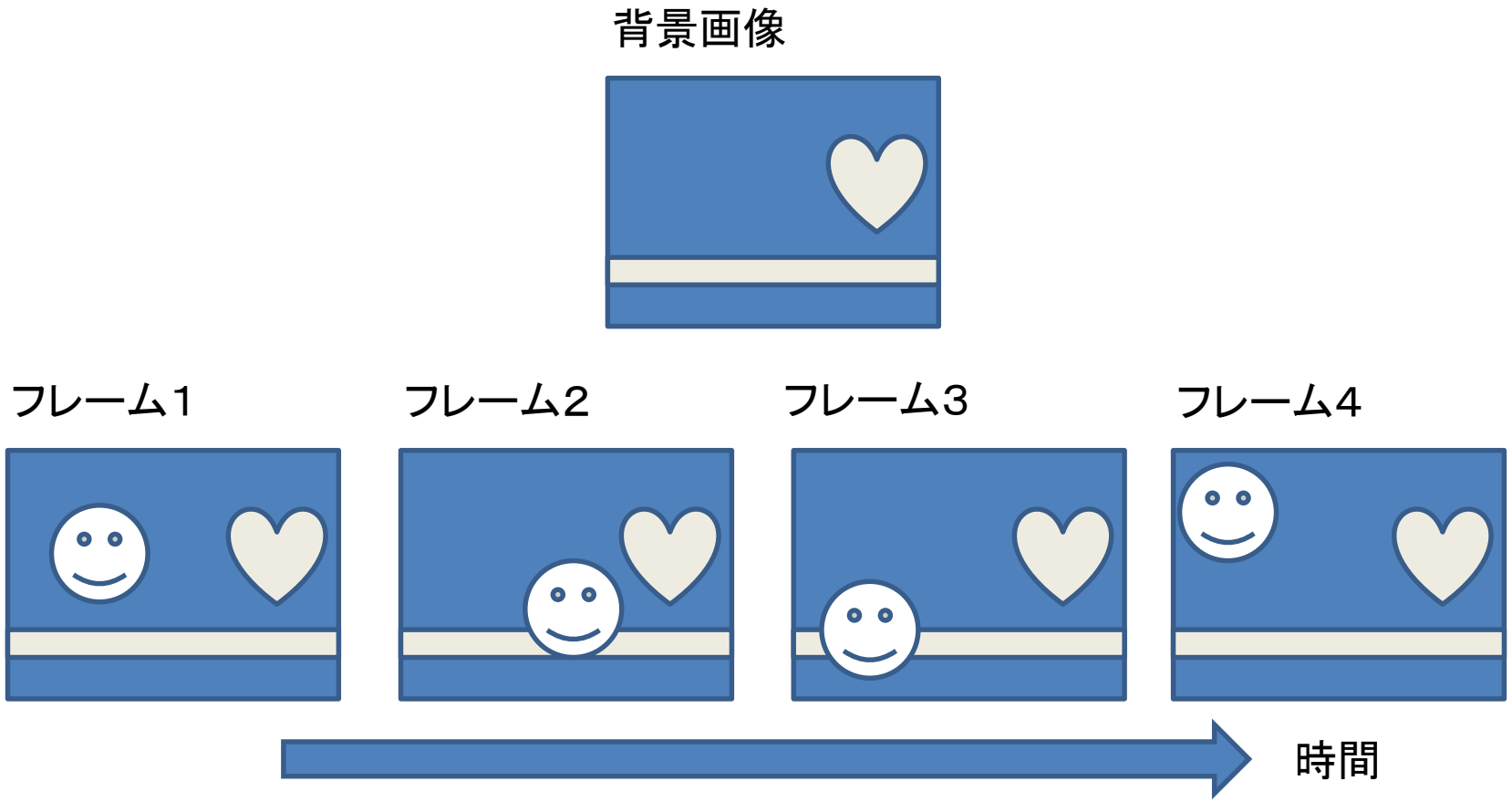
動画像

- 動画像：静止画像列の集まりと考える



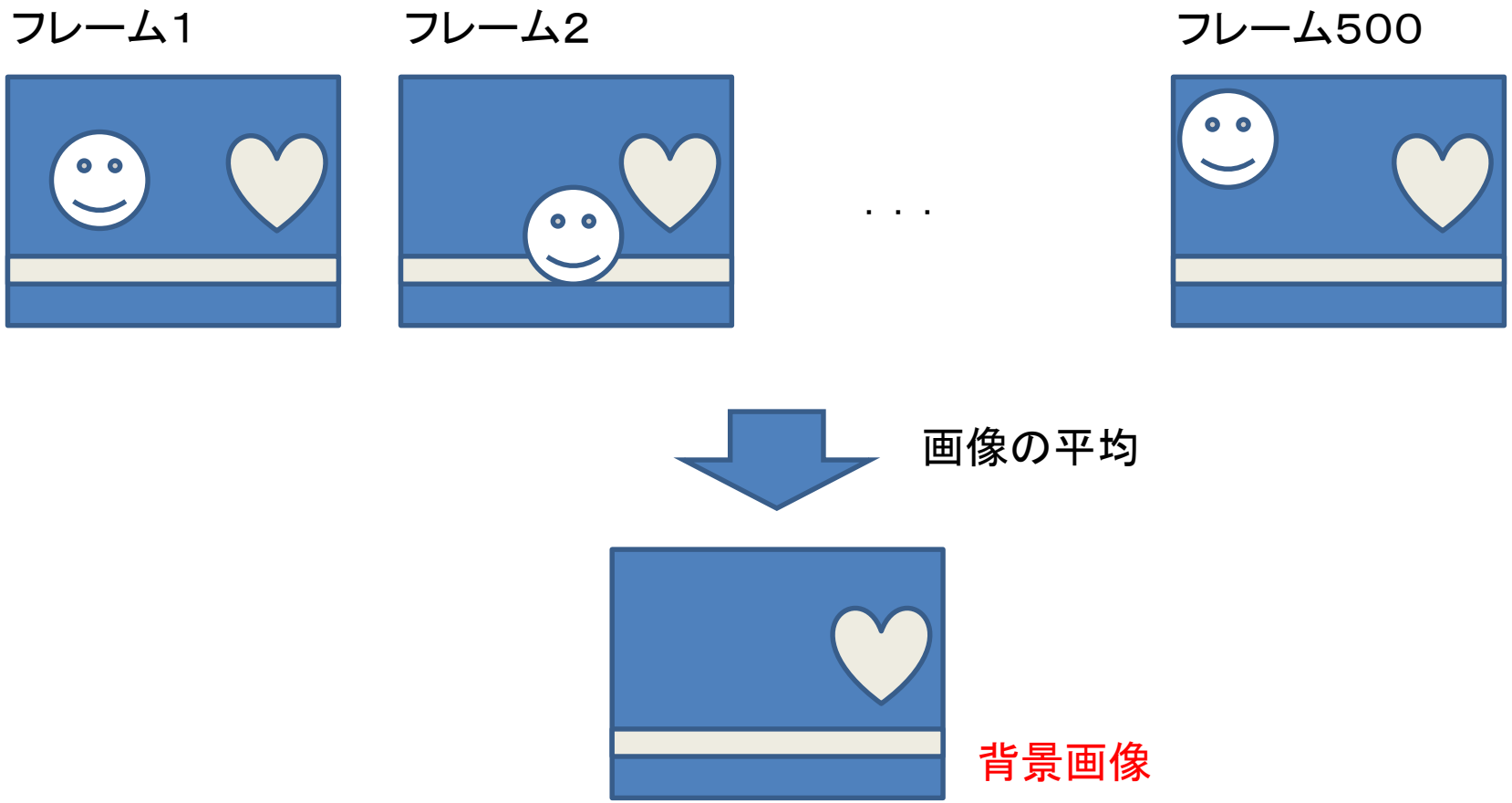
背景画像

- 背景画像：静的な画像（移動物体を含まない画像）

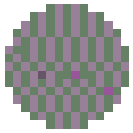


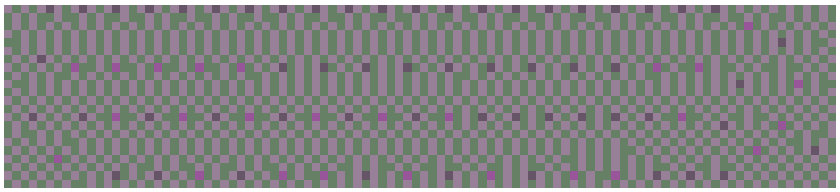
動画像から背景画像を作るには？

- 全フレームの平均を求める



サンプル画像





main関数のひな型

```
int Num = 14;       //読み込む画像枚数
load_movie_images(Num); //画像の読み込み
extract_bg(Num);   //背景の抽出
save_image(Num, “”);   //背景画像の保存
```

pgmmovielib.h

#define MAX_IMAGESIZE　　100

バッファ容量が大きくならないように画像サイズを下げた

#define MAX_NUM_OF_IMAGES　　20

動画を連続する画像の集まりとして考えるため，20枚の画像を読み込むようにした

連続する画像の読み込み

- プロトタイプ宣言 :

void load_movie_images(int num);

・int num: 読み込む画像の枚数を指定する

・読み込むファイル名は，0.pgm, 1.pgm, . . . num-1.pgmと仮定する

```
void load_movie_images(int num) {
        int i;
        char img_num[15];
        for (i = 0; i < num; i++) {
                sprintf(img_num, "%d", i);
                load_image(i, strcat(img_num, ".pgm"));
        }
}
```

```
void load_movie_images(int num) {
        int i;
        char img_num[15];
        for (i = 0; i < num; i++) {
                sprintf(img_num, "%d", i);
```

整数型の変数ｉ を文字列img_numに変換する

```
void load_movie_images(int num) {
        int i;
                sprintf(img_num, "%d", i);
                load_image(i, strcat(img_num, ".pgm"));
        }
}
```

i番目の画像にファイル名img_num.pgmの画像を読み込む

背景抽出関数　extract_bg

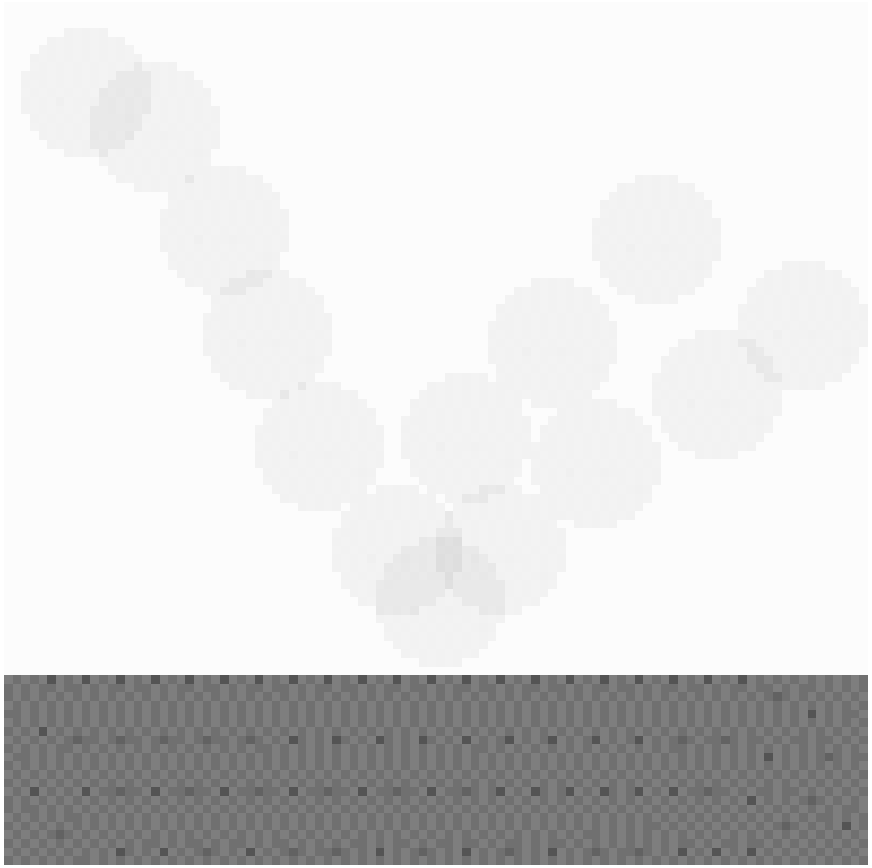
```
void extract_bg(int num) {
        int i;
        int x, y;
        int tmp_buf[100][100];
        width[num] = width[0]; //背景画像をnum番目に格納
        height[num] = height[0];
        for (y = 0; y < height[0]; y++) {
                for (x = 0; x < width[0]; x++) {
                        tmp_buf[x][y] = 0;      //初期化
                }
        }
```

```
        for (i = 0; i<num; i++) {
                for (y = 0; y < height[0]; y++) {
                        for (x = 0; x < width[0]; x++) {
                tmp_buf[x][y] += (int)image[i][x][y] / num;
                        }
                }
        }
        for (y = 0; y < height[0]; y++) {
                for (x = 0; x < width[0]; x++) {
                        image[num][x][y] = tmp_buf[x][y];
                }
        }
}
```

```
        for (i = 0; i<num; i++) {
                for (y = 0; y < height[0]; y++) {
                        for (x = 0; x < width[0]; x++) {
                tmp_buf[x][y] += (int)image[i][x][y] / num;
                        }
        }
        for (y = 0; y < height[0]; y++) {
                for (x = 0; x < width[0]; x++) {
                        image[num][x][y] = tmp_buf[x][y];
                }
        }
}
```

すべての画像を足し合わせて平均化する

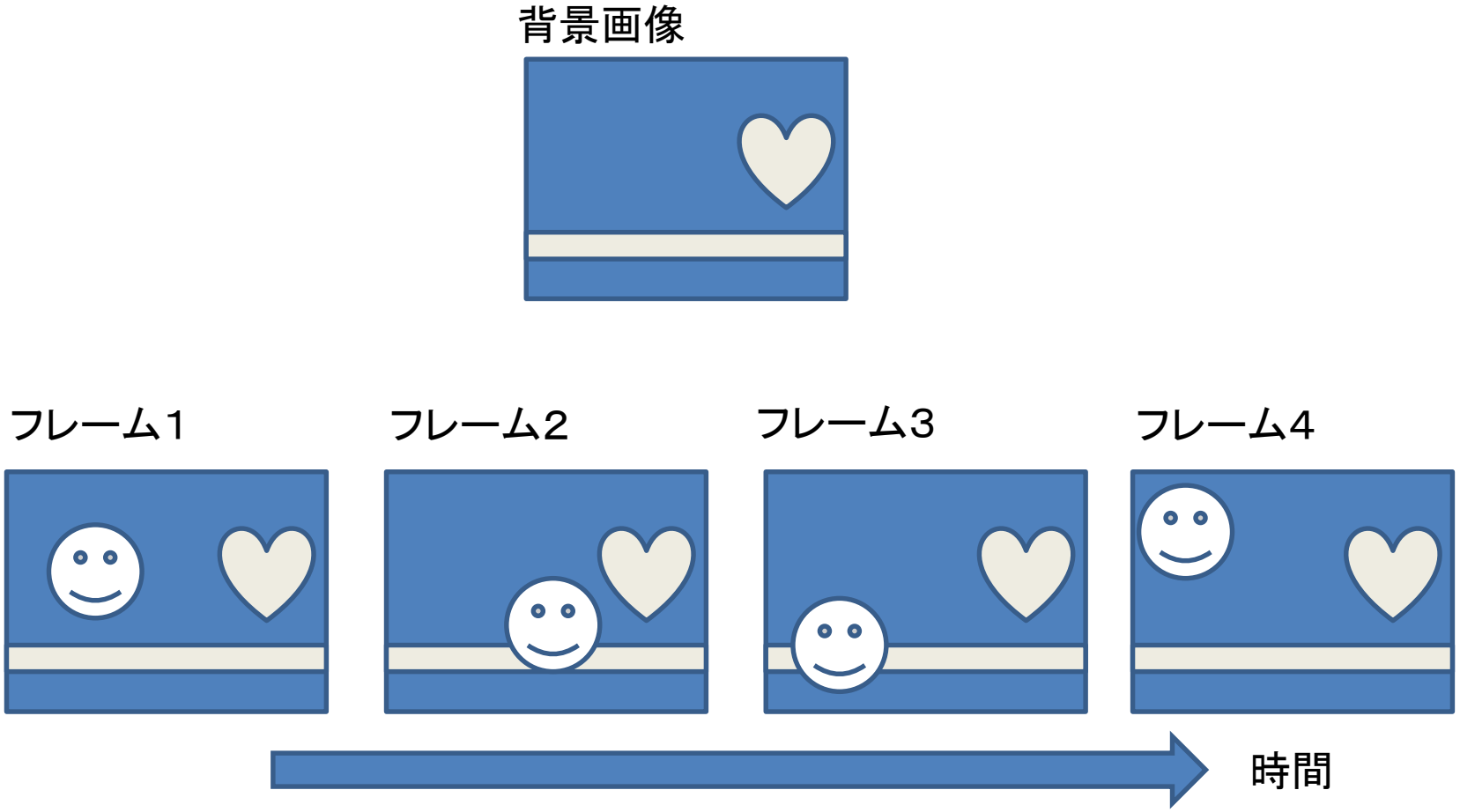
背景画像の抽出結果



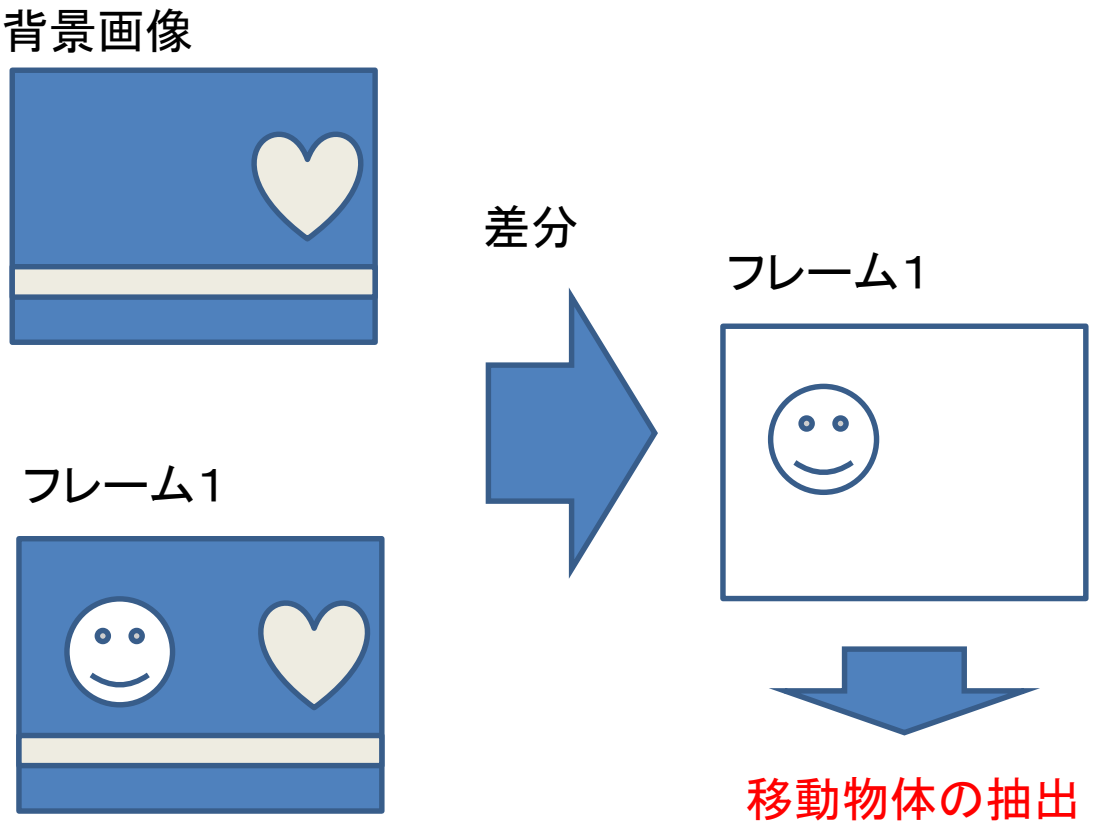
本講義の目的

- 動画像からの背景画像の作成
- 背景画像との差分による移動物体の抽出
- 移動軌跡の算出

背景画像とフレームの差分

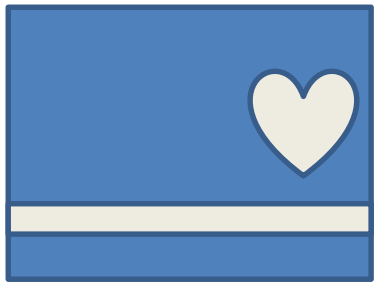


背景画像とフレームの差分



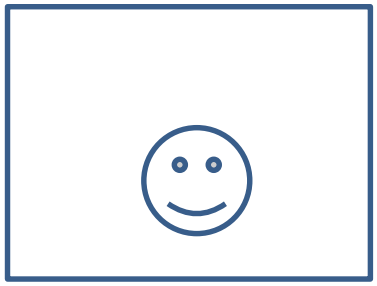
背景画像とフレームの差分

背景画像



フレーム1

フレーム2

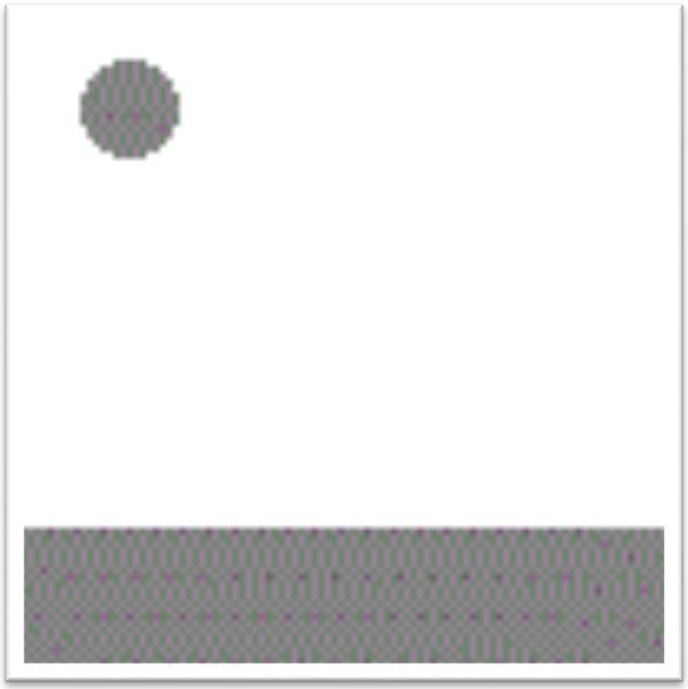


フレーム3

フレーム4

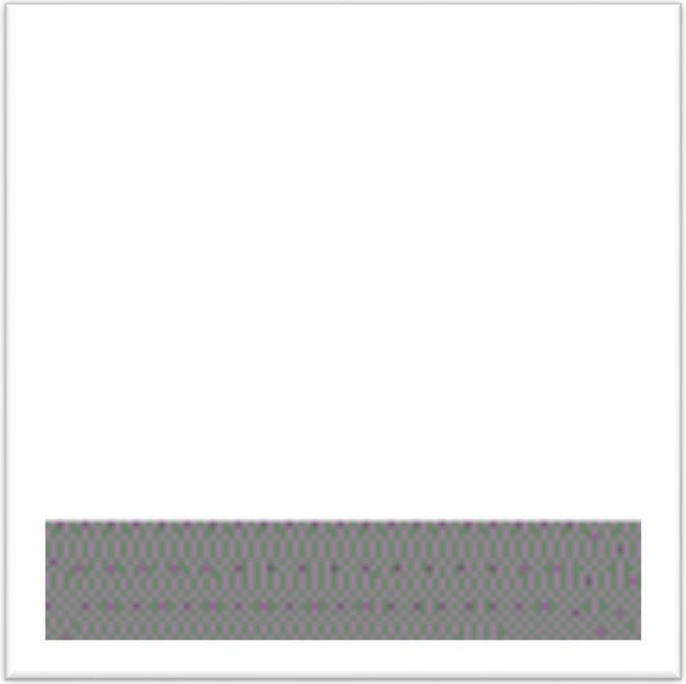
時間

背景画像とフレーム画像

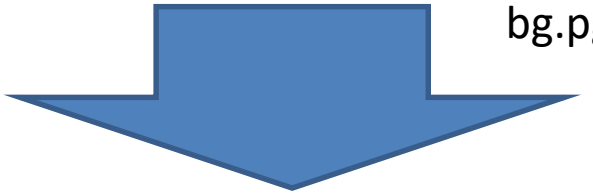


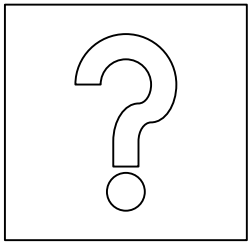
0.pgm

引き算



bg.pgm





0.pgm – bg.pgm= 0-out.pgm

関数のプロトタイプ宣言

```
void load_movie_images(int num); //作成済み
void retrieve_object(int num);     //物体抽出
```

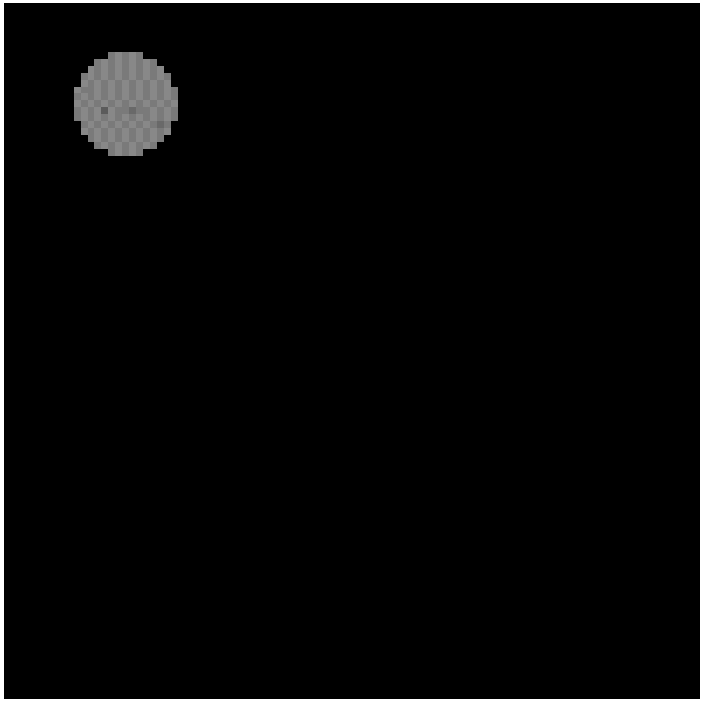
main関数のひな型

```
int Num = 14;
//背景画像の読みこみ
load_image(Num, "bg.pgm");
//連続画像の読み込み
load_movie_images(Num);
//オブジェクトの抽出
retrieve_object(Num);
```

```
void retrieve_object(int num) {
        int i;
        char img_num[15];
        int x, y;

        for (i = 0; i < num; i++) {
                for (y = 0; y < height[0]; y++) {
                        for (x = 0; x < width[0]; x++) {
                image[i][x][y] = image[i][x][y] - image[num][x][y];
                        }
                }
                sprintf(img_num, "%d", i);
                save_image(i, strcat(img_num, "-out.pgm"));
        }
}
```

オブジェクト抽出結果

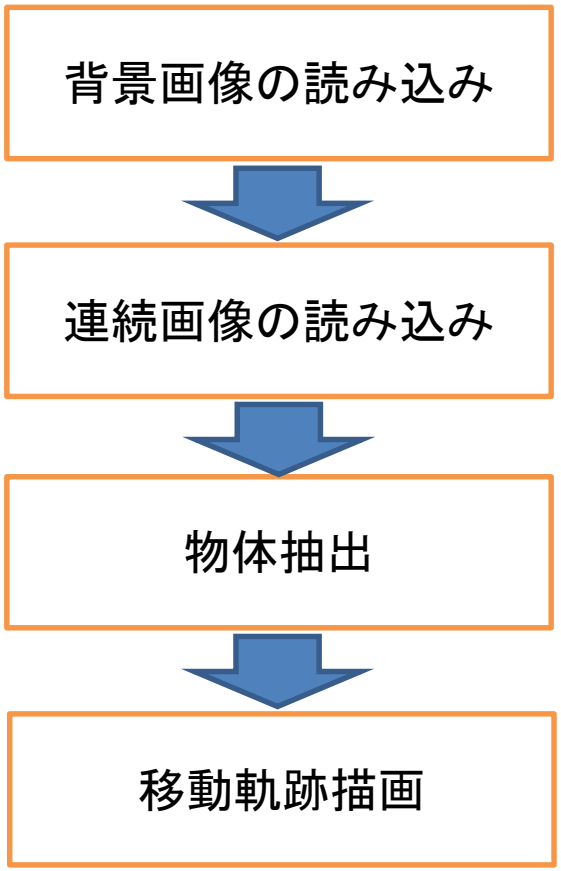


0.pgm – bg.pgm= 0-out.pgm

本講義の目的

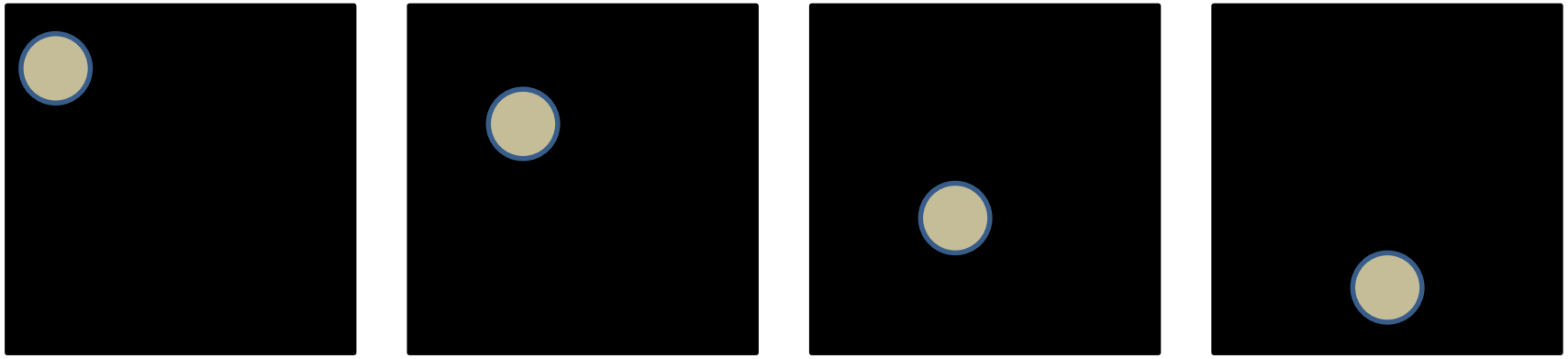
- 動画像からの背景画像の作成
- 背景画像との差分による移動物体の抽出
- 移動軌跡の算出

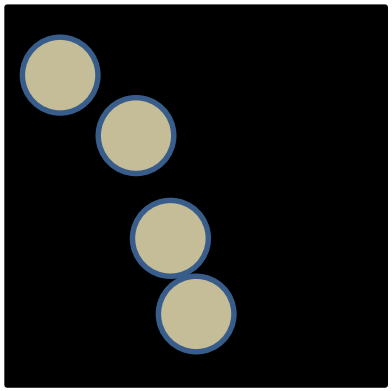
プログラムの流れ



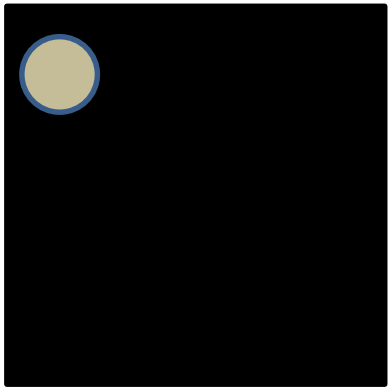
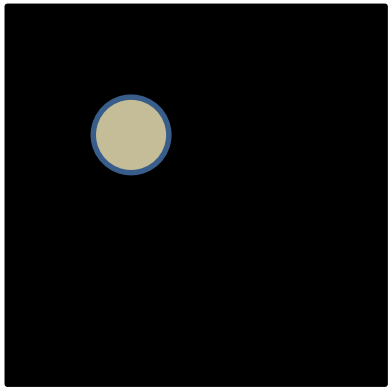
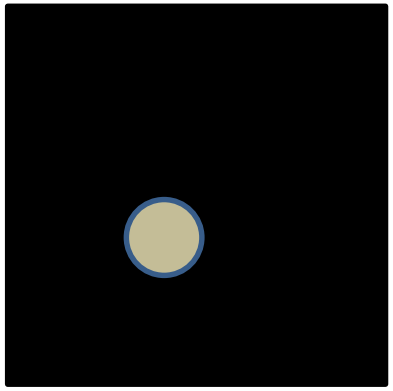
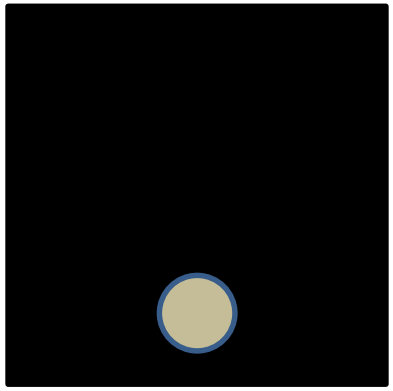

```
load_image(Num,"bg.pgm");
load_movie_images(Num);
retrieve_object(Num);
draw_trajectory(Num);
```

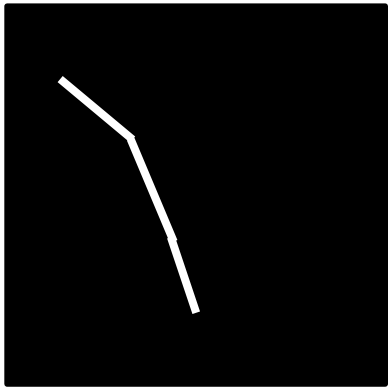
物体の移動軌跡を描く





物体の移動軌跡を描く

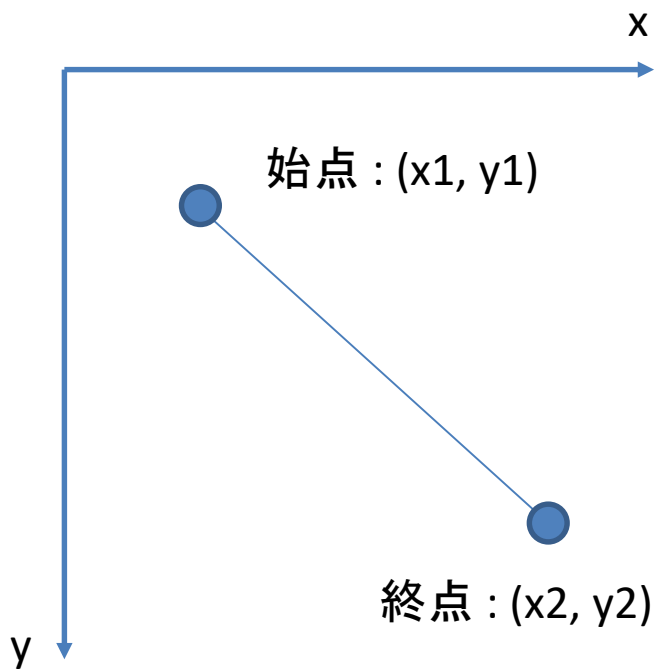







移動軌跡の様子

線分を描く関数 draw_line

- 一次関数 y = ax +bを用いて直線を描く
- 始点と終点の２点から傾きaと切片 bを求める



y1 = a * x1 + b
y2 = a * x2 + b

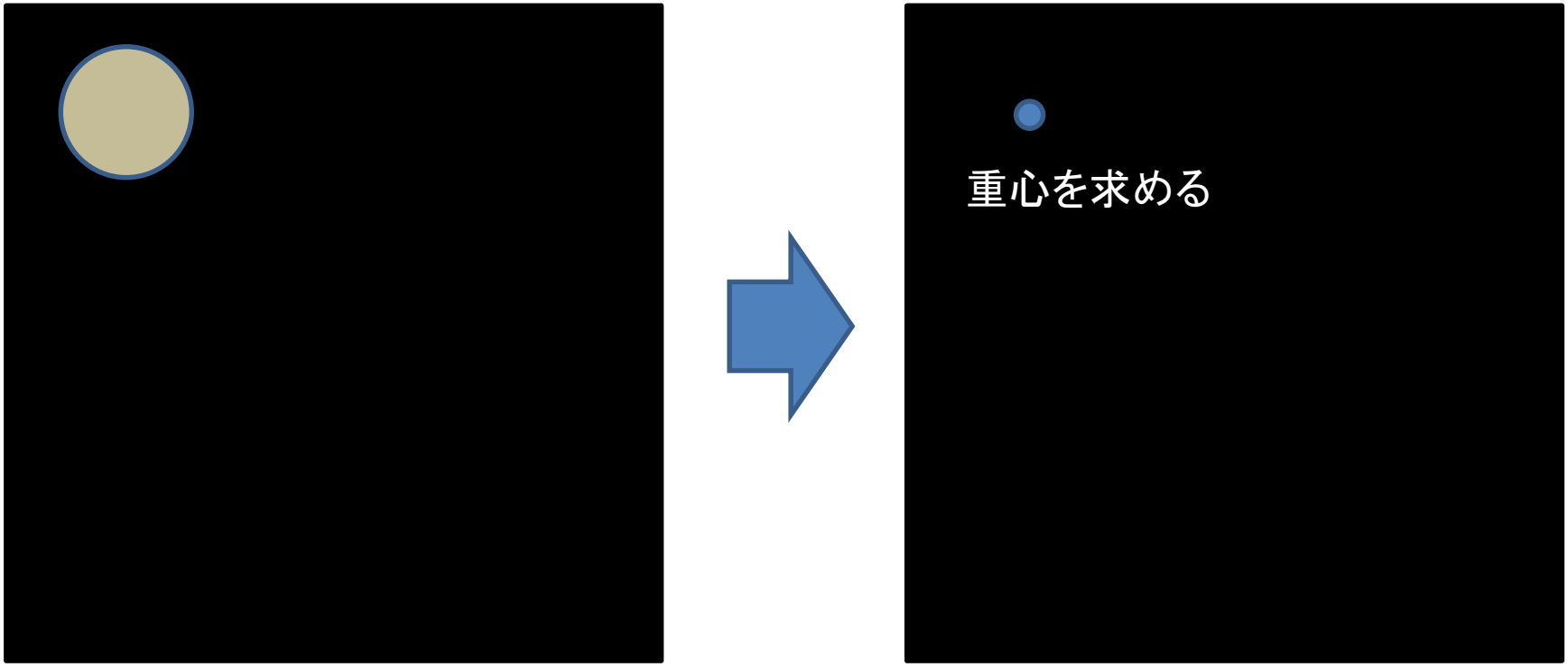
2式より・・・

a = (y2-y1)/(x2-x1)
b = ((y1+y2)-a*(x2+x1))/2

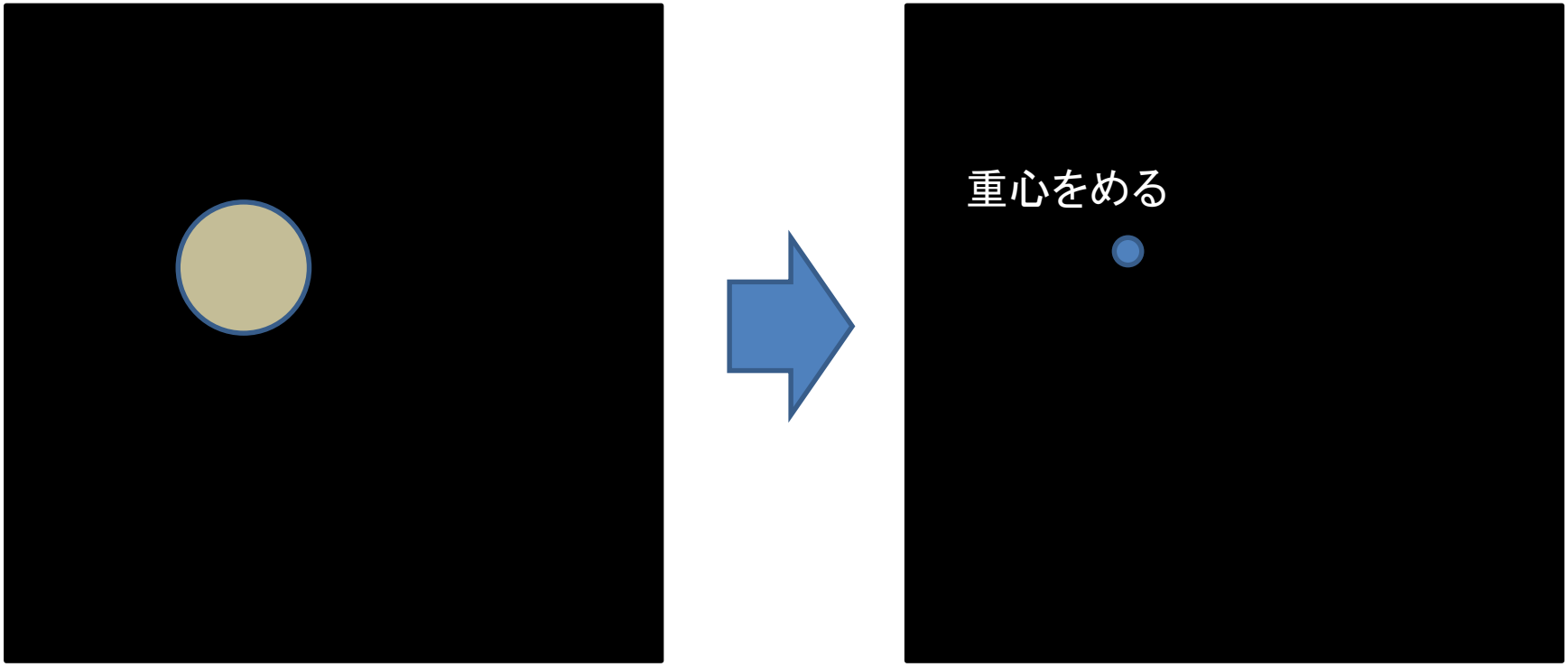
draw_line関数

```
void draw_line(int n, int x1, int y1, int x2, int y2,int col) {
        double a,b;
        int y,x;
        int x_start, x_end;
        a = (double)(y2 - y1) / (x2 - x1);
        b = (double)(((y1 + y2) - a*(x1 + x2))) / 2.0;
        if (x1 < x2) {
                x_start = x1;x_end = x2;
        }
        if (x1 > x2) {
                x_start = x2;x_end = x1;
        }
        for (x = x_start; x < x_end; x++) {
                y = (int)(a*x + b);
                image[n][x][y] = col;
        }
}
```

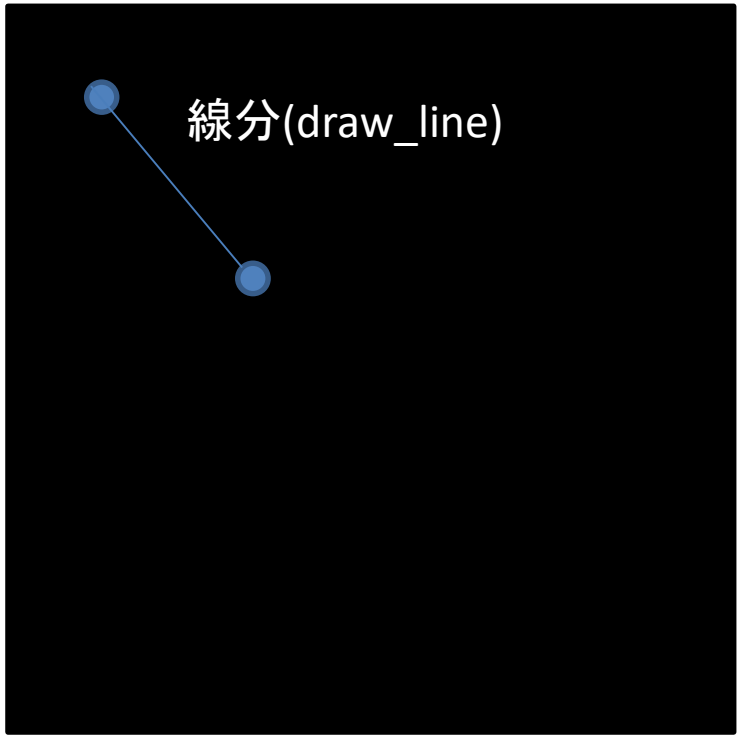
物体の重心 (Center of Gravity)算出 : calcCoG



物体の重心 (Center of Gravity)算出 : calcCoG



物体の重心 (Center of Gravity)算出 : calcCoG



重心 calcCog

重心のx座標：

$$cog\_x = \frac{\sum image[n][x][y] * x}{sum}$$

重心のy座標：

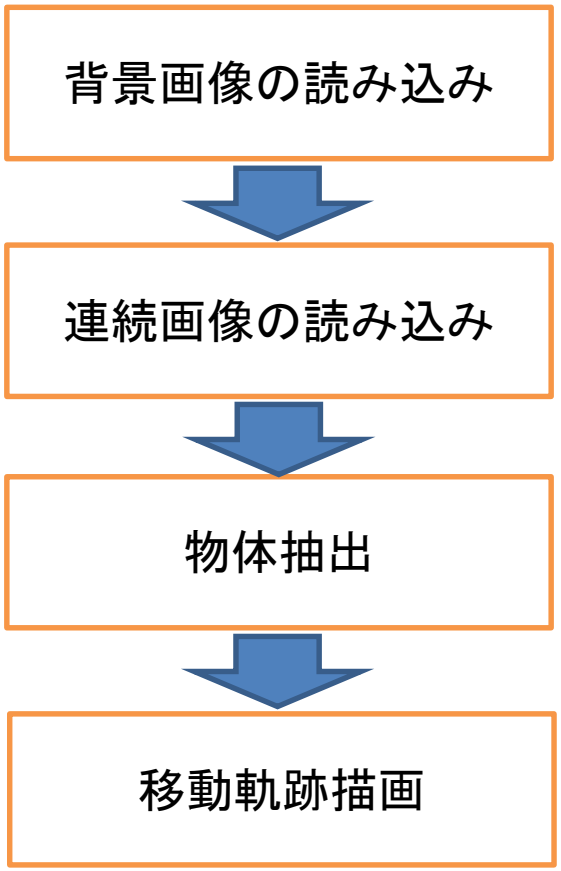
$$cog\_y = \frac{\sum image[n][x][y] * y}{sum}$$

ただし，　$sum = \sum image[n][x][y]$

関数：calcCoG

```
void calcCoG(int n,int *cog_x, int *cog_y) {
        int x, y, sum;
        *cog_x = 0;*cog_y = 0;sum = 0;
        for (y = 0; y < height[n]; y++) {
                for (x = 0; x < width[n]; x++) {
                        *cog_x += image[n][x][y] * x;
                        *cog_y += image[n][x][y] * y;
                        sum += image[n][x][y];
                }
        }
        *cog_x /= sum;*cog_y /= sum;
}
```

プログラムの流れ




```
load_image(Num,"bg.pgm");
load_movie_images(Num);
retrieve_object(Num);
draw_trajectory(Num);
```

```
void draw_trajectory(int num) {
        int i;
        int cog_x, cog_y;
        int sx, sy;
        calcCoG(0, &cog_x, &cog_y);
        sx = cog_x;
        sy = cog_y;
```

```
void draw_trajectory(int num) {
        int i;
        int cog_x, cog_y;
        int sx, sy;
        calcCoG(0, &cog_x, &cog_y);
        sx = cog_x;
        sy = cog_y;
```

画像番号

参照渡しで値を戻す

つづき

```
        for (i = 1; i < num; i++) {
                calcCoG(i,&cog_x,&cog_y);
                draw_line(num, sx, sy, cog_x, cog_y, 0);
                sx = cog_x;
                sy = cog_y;
                printf("%d,%d¥n", sx, sy);
        }
        save_image(num, "trajectory.pgm");
}
```

つづき

```
        for (i = 1; i < num; i++) {
                calcCoG(i,&cog_x,&cog_y);
                draw_line(num, sx, sy, cog_x, cog_y, 0);
                sx = cog_x;
                sy = cog_y;
                printf("%d,%d¥n", sx, sy);
        }
        save_image(num, "trajectory.pgm");
}
```

始点　　終点

つづき

```
        for (i = 1; i < num; i++) {
                calcCoG(i,&cog_x,&cog_y);
                draw_line(num, sx, sy, cog_x, cog_y, 0);
                sx = cog_x;
                sy = cog_y;
                printf("%d,%d¥n", sx, sy);
        }
        save_image(num, "trajectory.pgm");
}
```

画像番号　num:背景画像 (bg.pgm)　　黒線

移動軌跡の抽出結果