画像処理実験 第2回

09430509
今田将也

2020 年 10 月 6 日

1　(x,y) と (u,v) の位置に，ほぼ同じ物体が観測されていることを確認

図 1 からほぼ同じ物体が同じ位置に観測されている

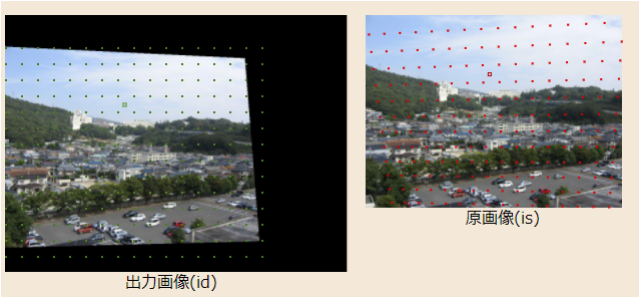

図 1: 出力画像と元画像

2　射影変換　a の要素を変化させ画像の変化を確認

変更前の配列 a

```
a=[[ .866 , -.5   ,  160],
[ .5    , .866 , -300],
[-.001 , 0     ,    1]];
```

変更後の配列 a

```
a=[[ .433 , -.5   ,  80],
[ .5    , .433 , -150],
[-.0001 , 0     ,    1]];
```

図 2: 配列変更後の出力画像

3　C 言語による実装例をもとに 上記と同様の処理を行う

実装にあたり，ImageClear 関数が定義されていなかったため，image.c に ImageClear 関数をまず定義した．参考資料から，画像のすべての要素を黒で塗りつぶせば良いため，画像の画素数分ループ処理を行い，全ての画素に黒の値である 0 を代入することで実装した．以下にそのソースを示す．また結果は図 3 に示した．

```
    void ImageClear(Image *im){
    for(int y=0;y<im->H;y++){
        for(int x=0;x<im->W;x++){
            IElem(im,x,y,0) = 0;
            IElem(im,x,y,1) = 0;
            IElem(im,x,y,2) = 0;
        }
    }
}
```

図 3: C 言語による処理結果

4　m0d を適当な行列に変更しても，img1 と img0 の位置関係が保たれることを確認

まず，図 4 に m0d 行列の変更前の画像を示す．そして，次のように画像の全体の座標位置を右にずらすよう変更を加えたところ，図 5 のような結果が得られた．この結果から，img1 と img0 の位置関係は m0d を適当な行列に変更しても保たれていることが確認できた．

```
var m0d=[[ 2.667,0,-500 ],
     [ 0,2.667,-100 ],
     [ 0,0,1 ]];
```

図 4: m0d 行列変更前

図 5: m0d 行列変更後

5　行列 m10 を修正し，ずれのない合成画像を作成することを試みる

いくつかの行列を修正し，試みたがうまくずれをなくすことができなかった．

6　m10 を算出し，合成に用いなさい

実験のページに記載されているページの特徴点を動かすことで正しい m10 を算出し，合成に用いた．合成をしたい元画像の両方で特徴点が正確に同じ物体を指すようにするということを行い算出をした．以下が算出した配列である．なお，図 6 が合成後の画像である．左下の部分をみると多少ブレが見られるものの多くの部分でブレがなくなっているためこの配列を回答とした．

```
0.930421    0.018946    118.490952
-0.043542   0.976597    22.193656
-0.000097   0.000002    1.000000
```

図 6: m10 を修正した後の合成画像

7 3x3 行列の積を計算する関数 mult33 を実装し，m1d = m10 · m0d を計算し，画像を合成

3x3 行列は，左の行列 1 行目の各要素を右側の行列 1 列目の各要素にかけて，それら 3 つの項を足しあわせ，1 行 1 列要素とする．左の行列 1 行目の各要素を右側の行列 2 列目の各要素にかけて … という手順を繰り返していき，3 行 3 列要素目までを計算すればよい．

その実装としては，2 重ループを用いて行った．左側は右側の 3 つ分が終わると次の行に進み，また，掛け合わす場所は定義より決まっているため，以下のソースコードのようにした．

```
    void mult33(double d[3][3],double a[3][3],double b[3][3]){
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            d[i][j] = a[i][0]*b[0][j] + a[i][1]*b[1][j] + a[i][2]*b[2][j];
        }
    }
}
```

そして，これを pano0.c の main 関数にて使用したのが以下である．ImageImageProjectionAlpha は，行列 m0d を使って im (0.jpg) を id に書き込む．m0d は結果を出力画像の中心寄りに表示するための平行移動（右に 100，下に 100）を行う射影変換である．行列 m1d を使った ImageImageProjectionAlpha は，id に im (1.jpg) を書き込む．m10 は img1 と img0 の関係を表し，m0d は img0 と 出力画像の関係を表す．img1 と出力画像の関係は m10 と m0d の積で表される．このときにパノラマ合成を行う必要があ

り，m10 に，先程求めた算出後の配列を使用．また，m1d は m10 と m0d の積で表されるため，mult33 を使い，積を計算した．パノラマ合成後の結果画像は図 7 に示した．

```
double m0d[][3]={
  1,0,-100,
  0,1,-100,
  0,0,1
};
im=ImageRead("0.jpg");
ImageImageProjectionAlpha(id,im,m0d,.5);
double m10[][3]={
  0.930421,   0.018946,   118.490952,
  -0.043542,  0.976597,   22.193656,
  -0.000097,  0.000002,   1.000000
}, m1d[3][3];
mult33(m1d,m10,m0d);
```

図 7: C 言語でのパノラマ合成の結果画像

8 「原画像の座標 (u,v) を出力画像の座標 (x,y) に変換する行列」を A とすると，上記の実装で用いている a は A の逆行列である．

これを確認するために，$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ の，逆行列 $\begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ をかけたときに 2 倍になるかどうかを見る．

homography.c の配列 a を先程の逆行列のように変え，実行すると図 8 のようになった．

図 8: 結果

きちんと拡大されていたことが確認できた．

9 感想

パノラマ画像の生成ができた．ヘッダーファイルからの読み込み方法や，行列の計算式など基本的な実装も危うかったのでちょっと大変でしたがなんとかできました．

最後の確認の問題が難しかったです．