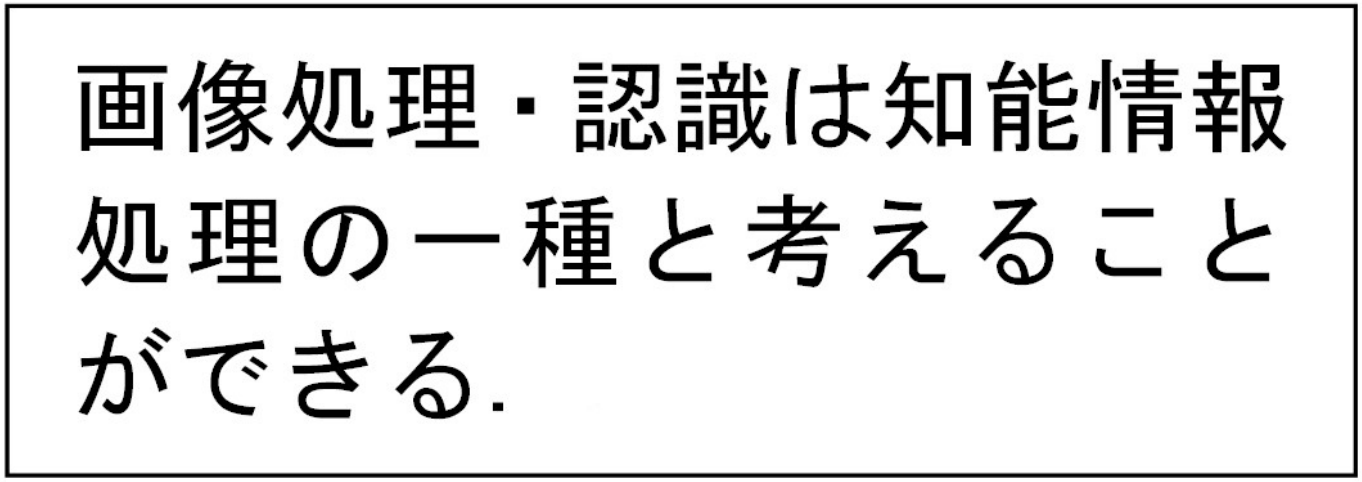
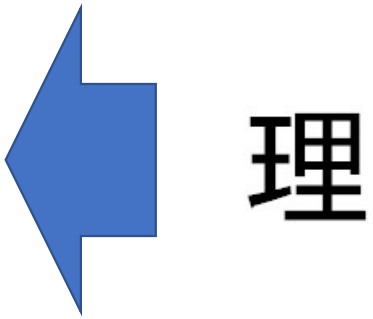
テンプレートマッチング

- 「原画像」から「テンプレート画像」をマッチングさせる

「原画像」



「テンプレート画像」

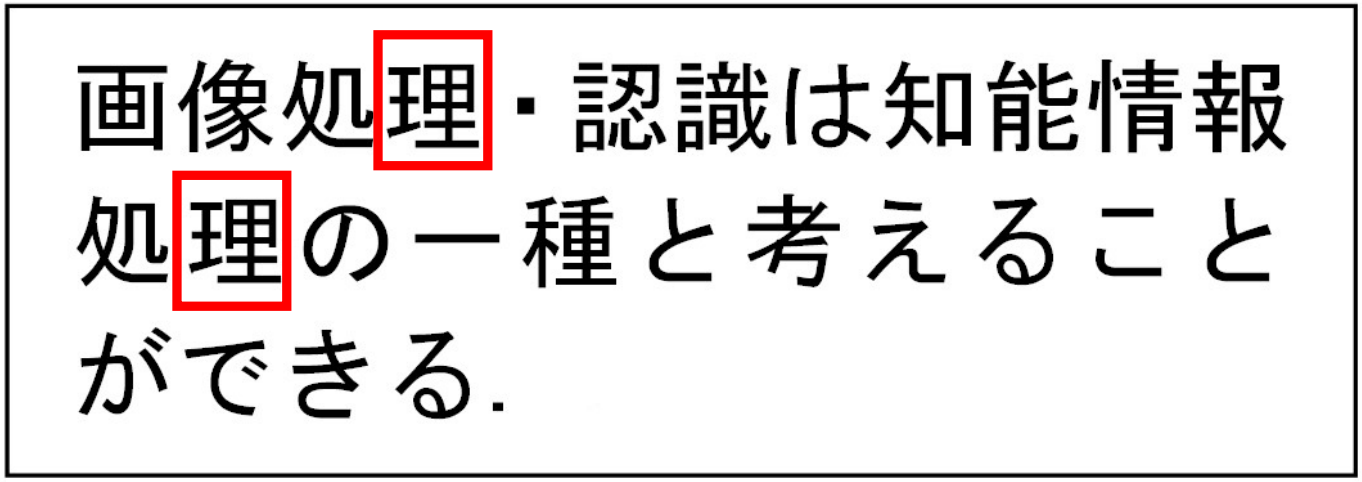


【理】を探す

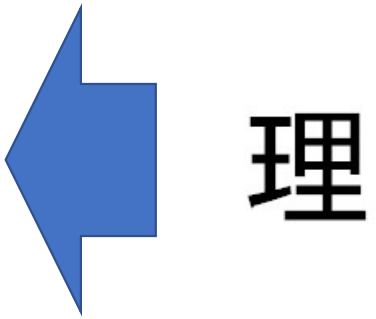
テンプレートマッチング

- 「原画像」から「テンプレート画像」をマッチングさせる

「原画像」



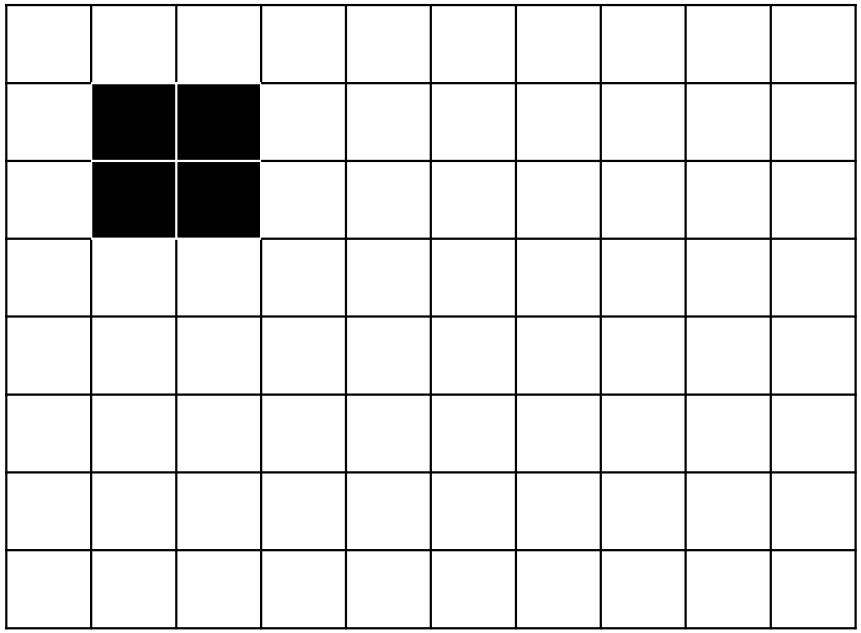
「テンプレート画像」



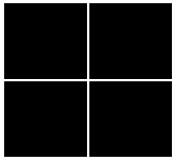
考え方

・原画像の左上からテンプレート画像と比較する

「原画像」



「テンプレート画像」

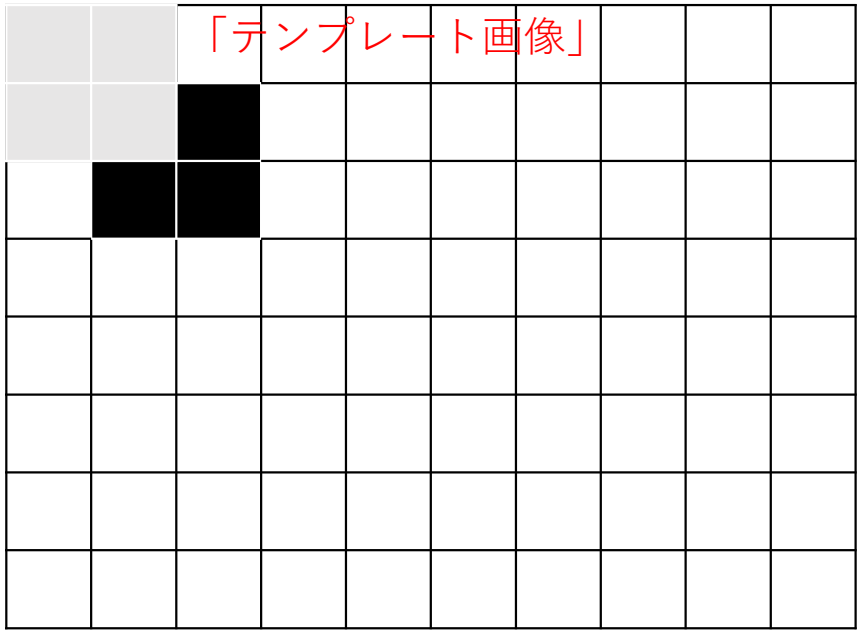


テンプレート画像を２×２ピクセルとした

考え方

・原画像の左上からテンプレート画像と比較する

「原画像」



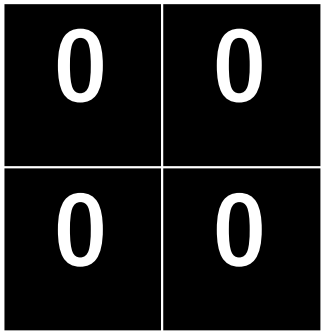
「拡大図」

0	0	255	255
0	0	0	255
255	0	0	255
255	255	255	255

考え方

- 原画像の左上からテンプレート画像と比較する

「拡大図」



評価関数

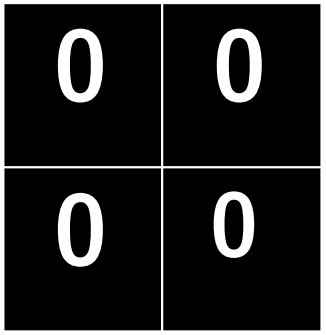
$$match = \sum_{i=0}^{1}\sum_{j=0}^{1} |image[n][x+i][y+j] - template[i][j]|$$

255	255	255	255
255	0	0	255
255	0	0	255
255	255	255	255

考え方

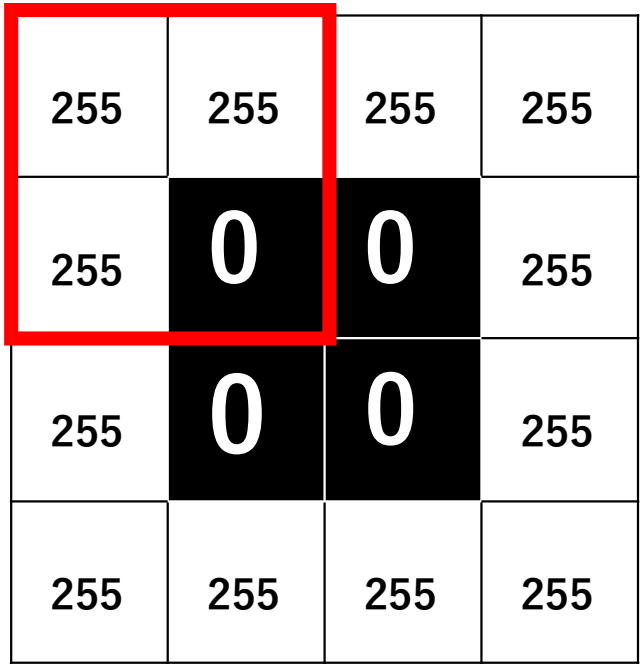
・原画像の左上からテンプレート画像と比較する

「拡大図」



評価関数：

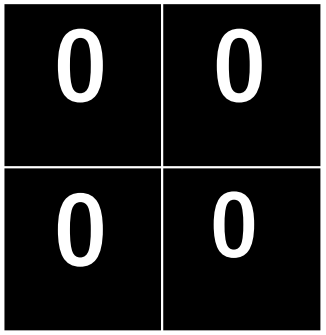
match= |255-0|+|255-0|+|255-0|+|0-0| ＝765



考え方

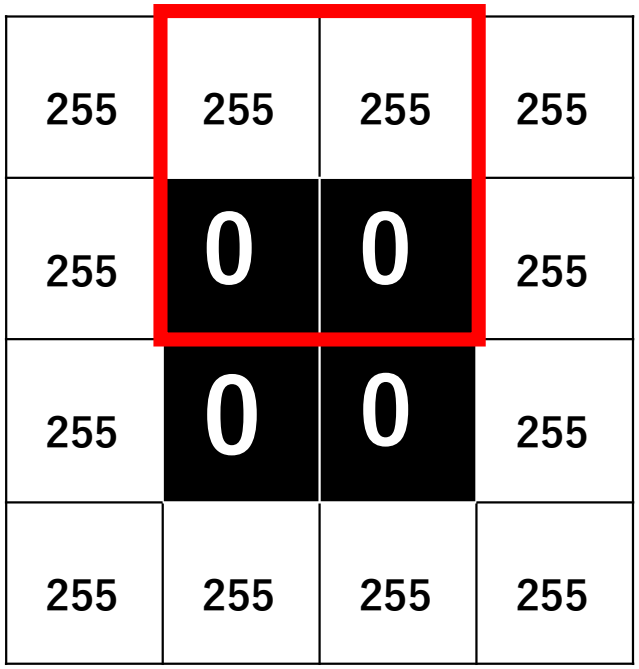
・探索対象を左へ移動させて同様に評価する

「拡大図」



評価関数：

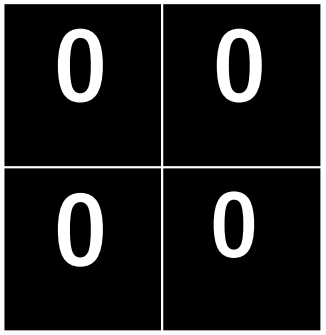
match= |255-0|+|255-0|+|0-0|+|0-0| ＝510



考え方

・探索対象を左へ移動させて同様に評価する

「拡大図」



評価関数：

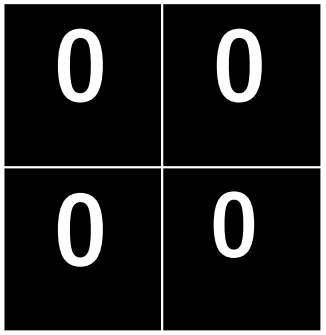
match= |255-0|+|255-0|+|0-0|+|255-0| ＝765

255	255	255	255
255	0	0	255
255	0	0	255
255	255	255	255

考え方

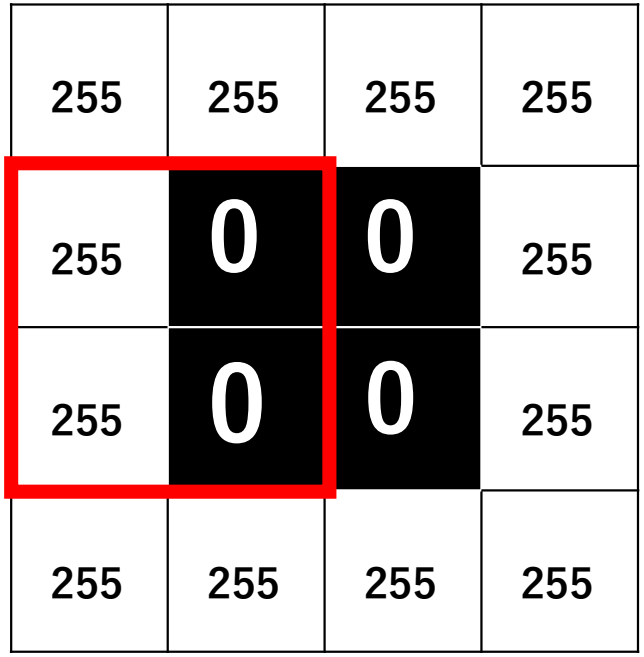
・探索対象を左へ移動させて同様に評価する

「拡大図」



評価関数：

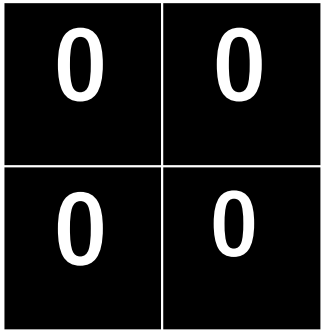
match= |255-0|+|0-0|+|255-0|+|0-0| ＝510

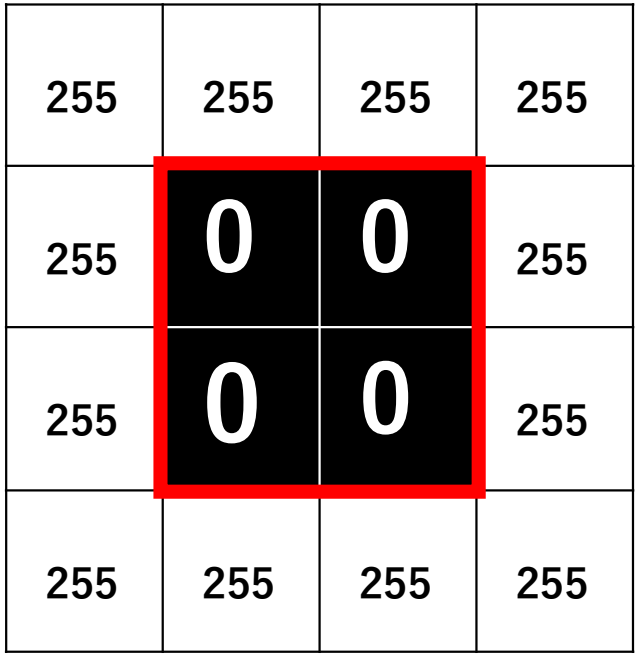


考え方

・探索対象を左へ移動させて同様に評価する

「拡大図」





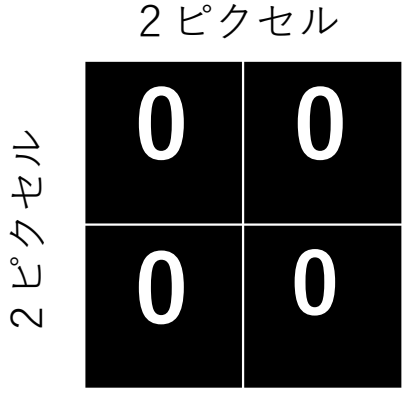
評価関数：

match= |0-0|+|0-0|+|0-0|+|0-0| ＝0

テンプレートにマッチすると評価関数の値が低下する

マッチング判定方法

・実際にはグレースケールでも扱えるように「閾値」を設定して，match < threshold　として判定する



ここではテンプレート画像の面積
（２×２）＝４を閾値 thresholdとする

main関数

```
char file[256]=“”;
printf("●原画像を読み込みます¥n");
load_image( 0, file );   /* ファイル → 画像No.0(原画像) */
printf("●文字のテンプレート画像を読み込みます¥n");
load_image( 1, file );   /* ファイル → 画像No.1(テンプレート) */
/* 画像No.0のマッチング部分を反転．No.2：出力画像 */
matching( 0, 1, 2 );
save_image( 2, file );   /* 画像No.1 → ファイル */
```

matching関数

```
void matching( int n1, int n2, int n3 )
/* 原画像(n1)上をテンプレート(n2)で走査し一致部分を反転 */
/* (n3)は出力画像 */
{
    int x,y,s,t,sum,threshold;

    /* 画像No.n3 の準備 */  copy_image(n1,n3);
    /* マッチング処理 */
    threshold = width[n2]*height[n2];
```

・n1：原画像，n2：テンプレート画像，n3：出力画像
・threshold = width[n2]*height[n2];　⇒　テンプレートの画素数を閾値に設定している

matching関数

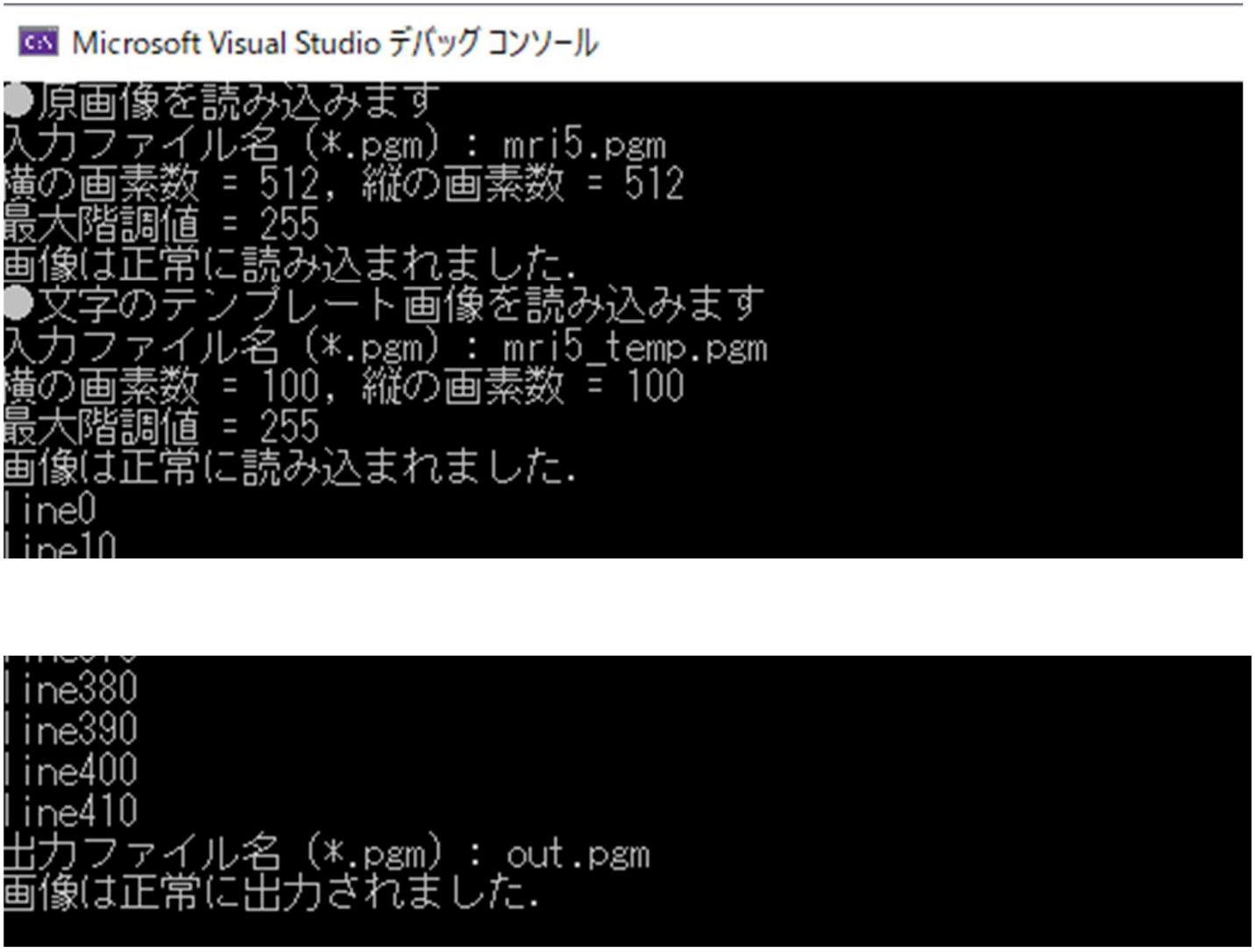
```
for(y=0;y<height[n1]-height[n2];y++){
        if (y % 10 == 0) printf("line%d¥n",y);
        for(x=0;x<width[n1]-width[n2];x++){
            sum=0;
            for(t=0;t<height[n2];t++)
                for(s=0;s<width[n2];s++)
                    sum += abs( image[n1][x+s][y+t] - image[n2][s][t] );
```

・ if (y % 10 == 0) printf(“line%d¥n”,y); ⇒10ピクセル毎に行数を表示している
・for(x=0;x<width[n1]-width[n2];x++){　⇒　マッチングがはみ出さないようにしている
・sumが上述の評価関数に相当している

matching関数

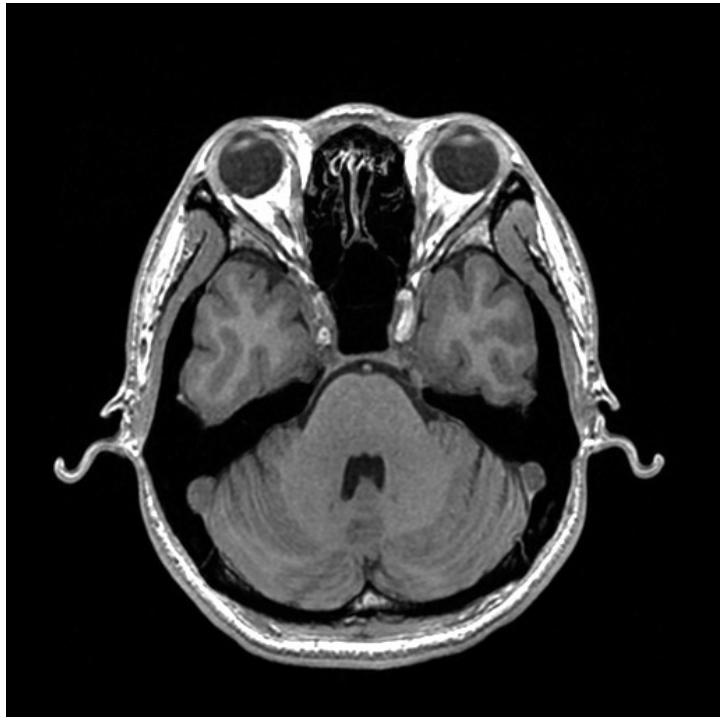
```
            if ( sum < threshold )
               for(t=0;t<height[n2];t++)
                  for(s=0;s<width[n2];s++)
                     image[n3][x+s][y+t] = 255 - image[n1][x+s][y+t];
         }
      }
   }
```

・ image[n3][x+s][y+t] = 255 - image[n1][x+s][y+t];　⇒　マッチングしたところを色調反転している

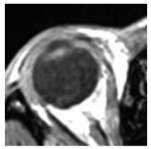

Microsoft Visual Studio デバッグ コンソール
●原画像を読み込みます
入力ファイル名（*.pgm）： mri5.pgm
横の画素数 = 512, 縦の画素数 = 512
最大階調値 = 255
画像は正常に読み込まれました.
●文字のテンプレート画像を読み込みます
入力ファイル名（*.pgm）： mri5_temp.pgm
横の画素数 = 100, 縦の画素数 = 100
最大階調値 = 255
画像は正常に読み込まれました.
line0
line10
line380
line390
line400
line410
出力ファイル名（*.pgm）： out.pgm
画像は正常に出力されました.

結果確認

原画像



テンプレート画像



マッチング画像