MACHINE VISION IMAGE TRACKING

YU-JUI, CHEN

B053021023

ABSTRACT

- Open image
- Method 1
- Method 2
- Test

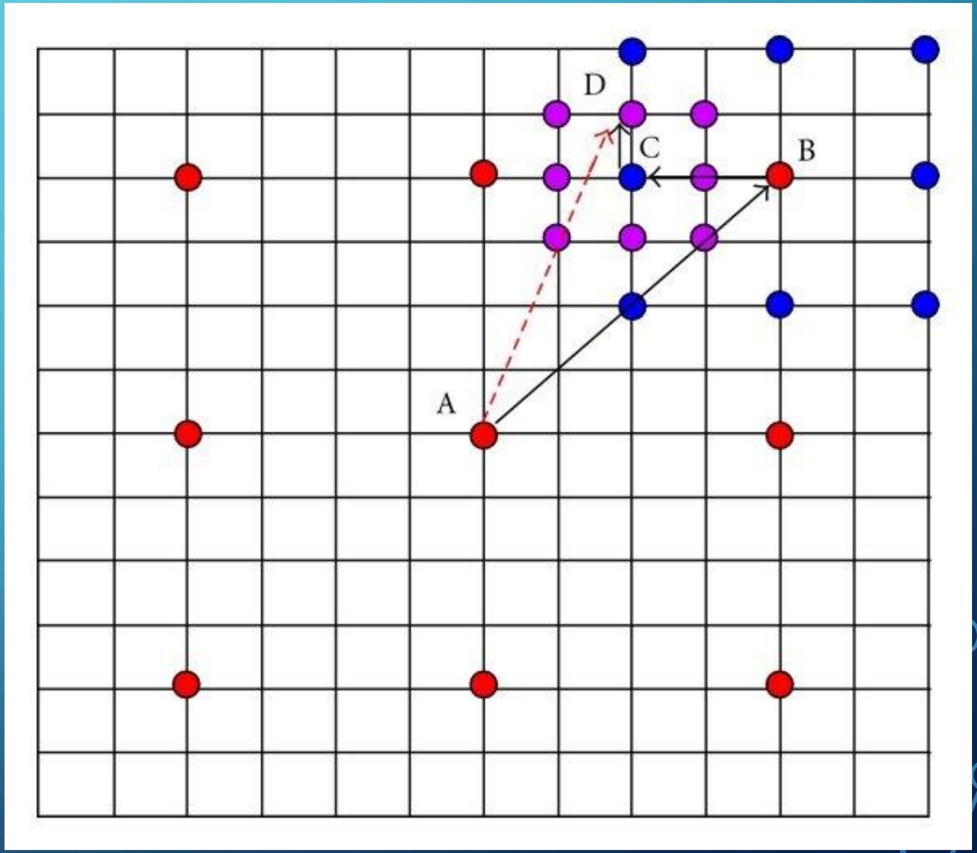
METHOD 1

- Three step search
- Sum of square difference

THREE STEP SEARCH

- Is a simple and effective algorithm to reduce the amount of calculation
- Starting from setting the center and then decide the distance of other eight positions in the figure. Comparison the calculated values of the nine positions. According to the conditions of the judgment, select the desired position among the nine points.

Reduce the distance, and then repeat the above steps twice.

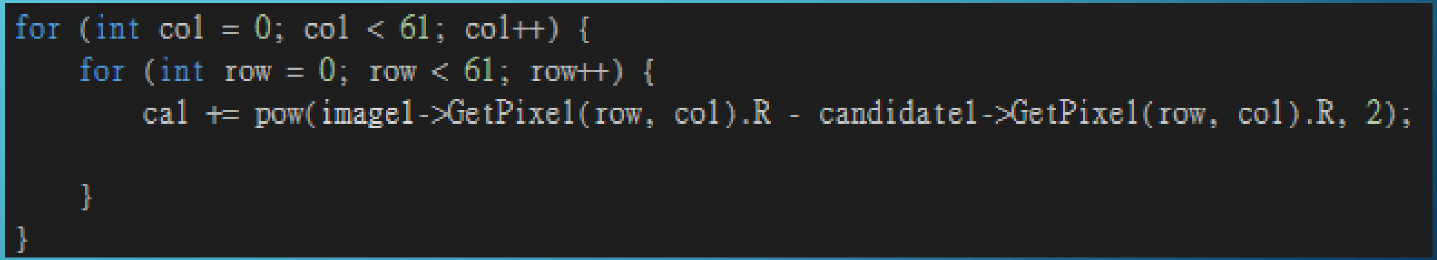


SUM OF SQUARE DIFFERENCE

- $\sum_{i=1}^{m} \quad \sum_{j=1}^{n} \quad \{f[i,j]-g[i,j]\}^2$

$f[i,j]=$ test image

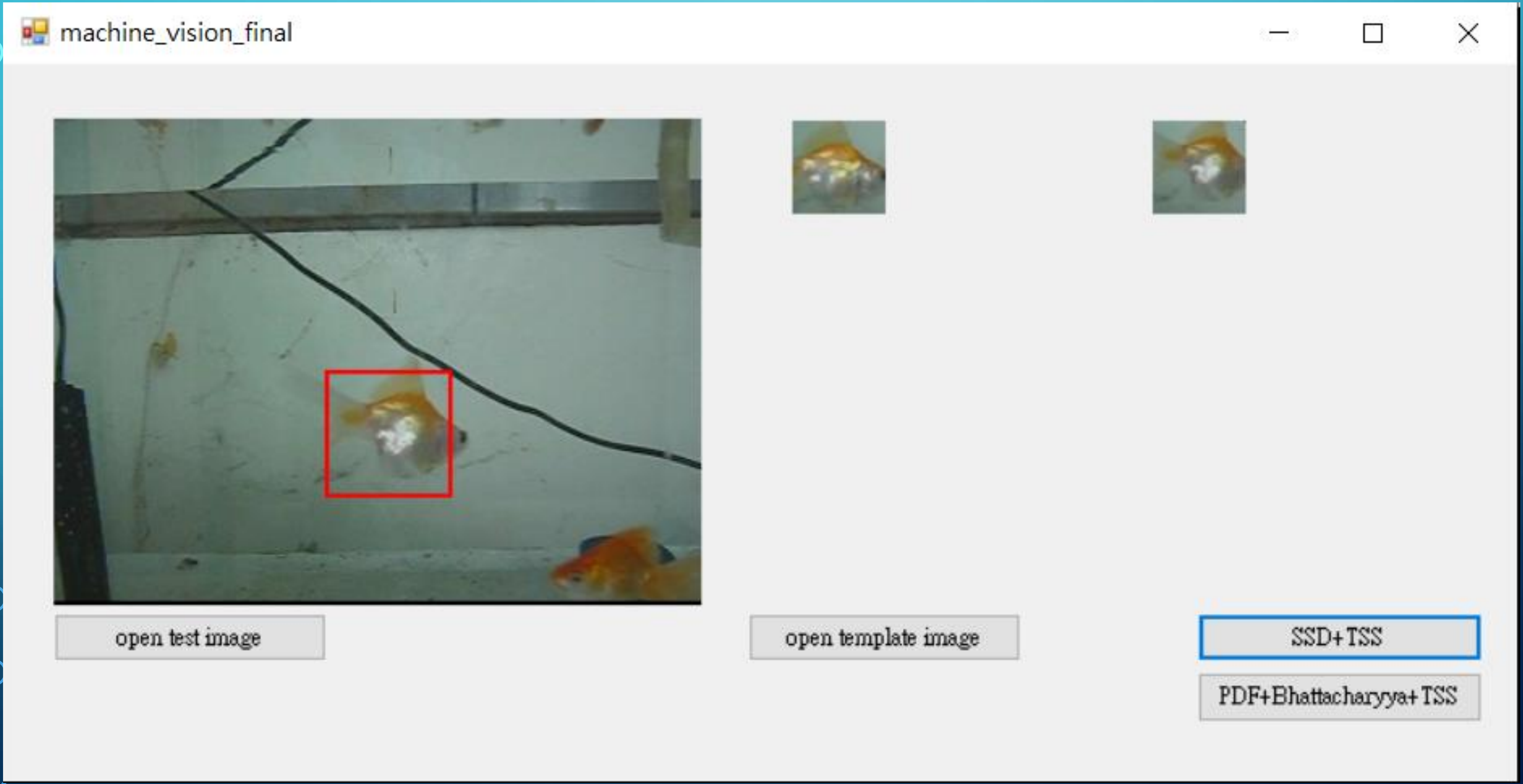
$g[i,j]=$ template image



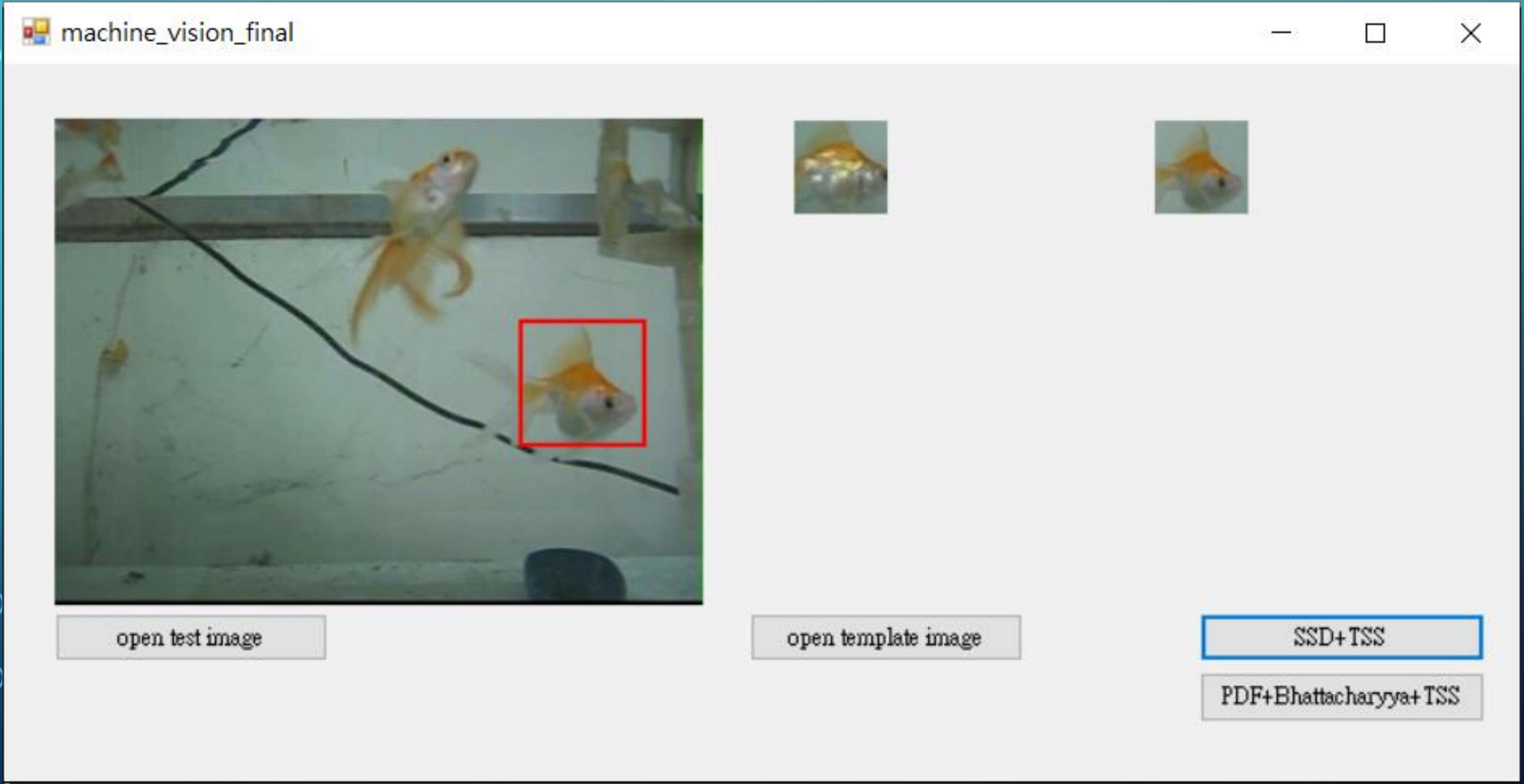
```
for (int col = 0; col < 61; col++) {
    for (int row = 0; row < 61; row++) {
        cal += pow(image1->GetPixel(row, col).R - candidate1->GetPixel(row, col).R, 2);

    }
}
```

- Red color only

RESULT – TEST IMAGE 1



RESULT – TEST IMAGE 4



METHOD 2

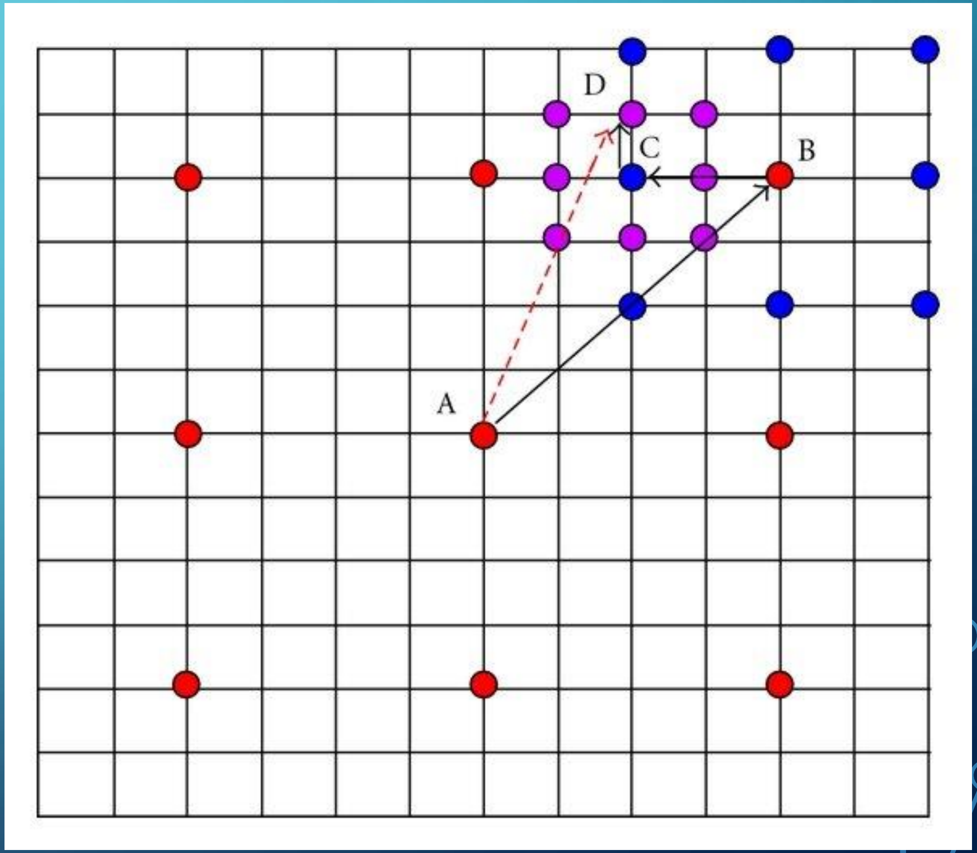
- Three step search
- Probability density function approach
- Bhattacharyya coefficient

THREE STEP SEARCH

- Is a simple and effective algorithm to reduce the amount of calculation

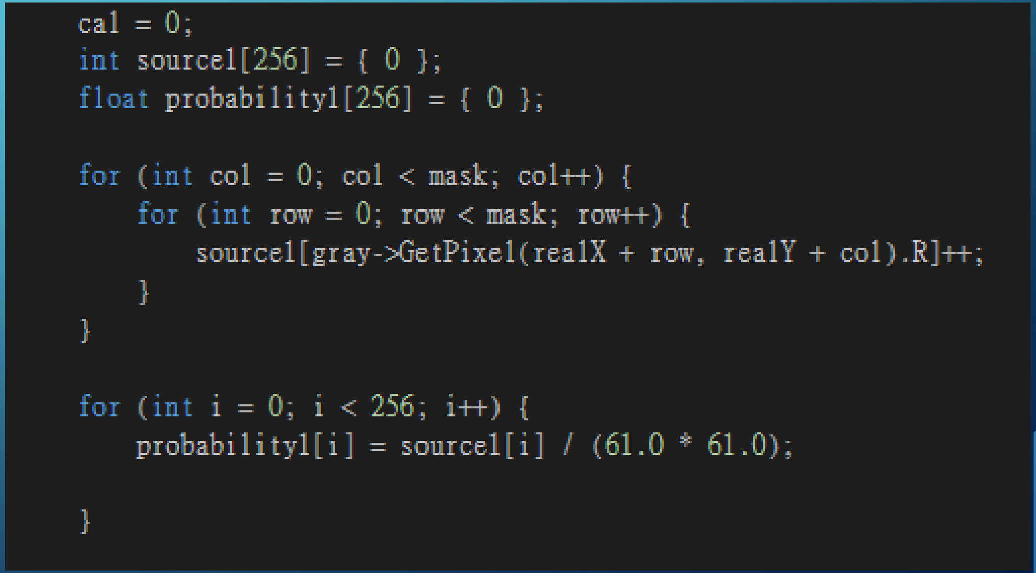
- Starting from setting the center and then decide the distance of other eight positions in the figure. Comparison the calculated values of the nine positions. According to the conditions of the judgment, select the desired position among the nine points.

  Reduce the distance, and then repeat the above steps twice.



PROBABILITY DENSITY FUNCTION APPROACH

Add all the pixels in the test image and template image with same size of mask and divided the sum with the number of pixels

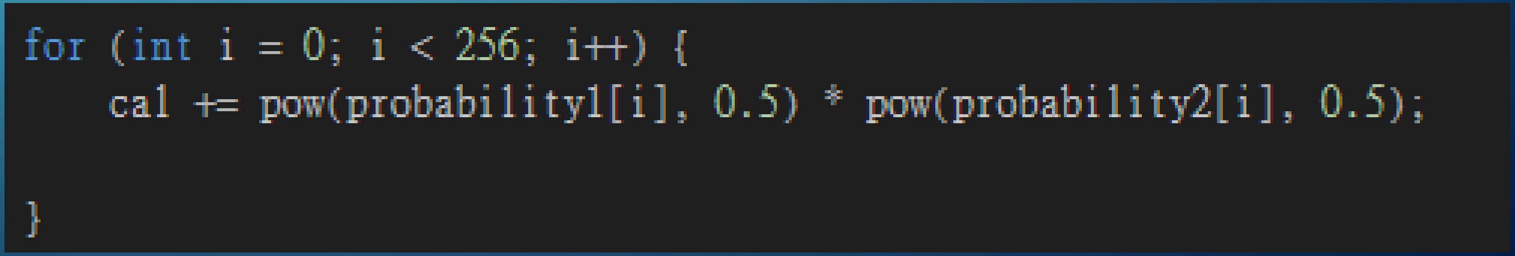


```
    cal = 0;
    int source1[256] = { 0 };
    float probability1[256] = { 0 };

    for (int col = 0; col < mask; col++) {
        for (int row = 0; row < mask; row++) {
            source1[gray->GetPixel(realX + row, realY + col).R]++;
        }
    }

    for (int i = 0; i < 256; i++) {
        probability1[i] = source1[i] / (61.0 * 61.0);

    }
```

BHATTACHARYYA COEFFICIENT

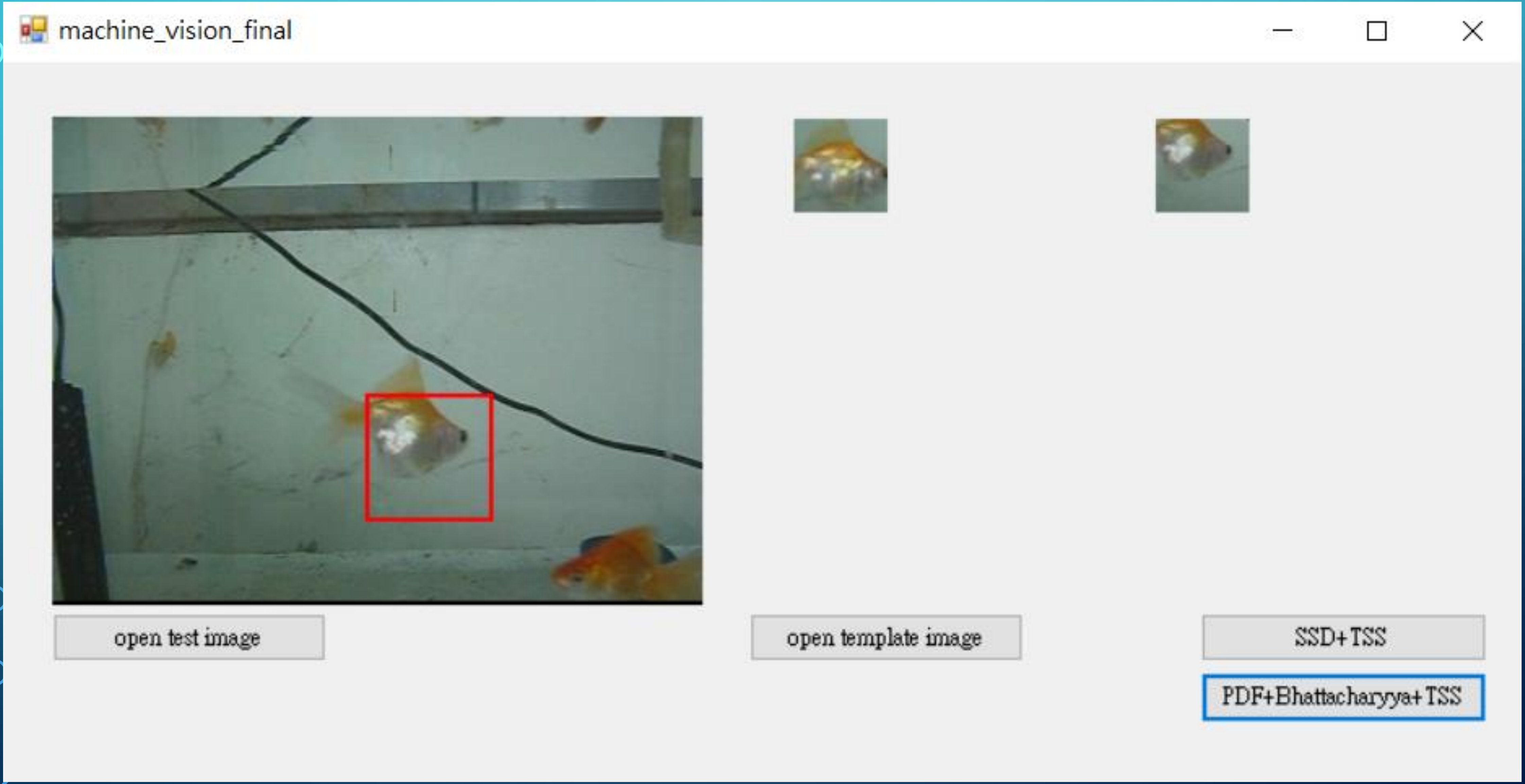
As we know the probability density in both test image and template image with same size of mask, then we multiple two probability density and square it.

$$BC(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{n} \sqrt{p_i q_i}$$

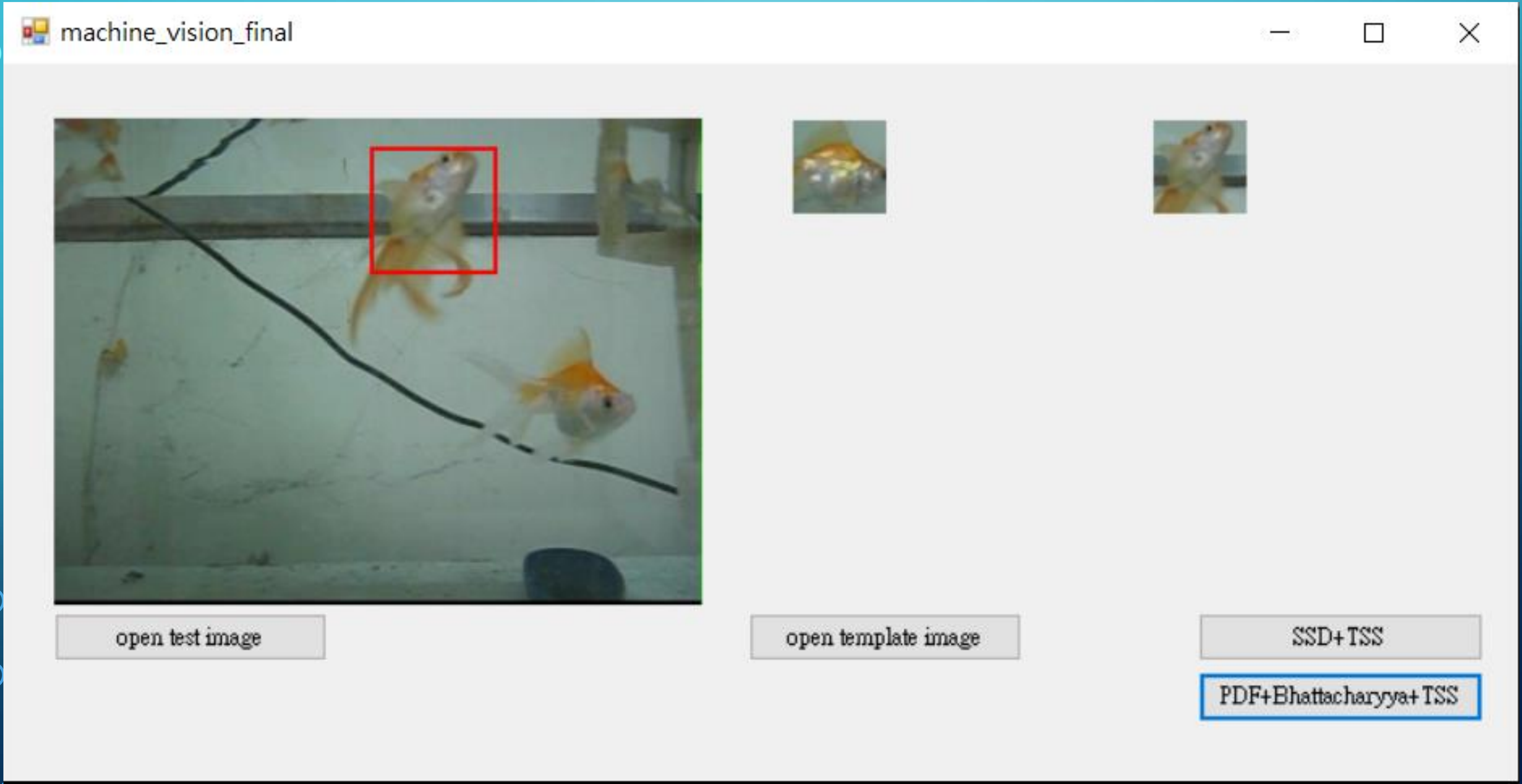


```
for (int i = 0; i < 256; i++) {
    cal += pow(probability1[i], 0.5) * pow(probability2[i], 0.5);

}
```

RESULT – TEST IMAGE 1



RESULT – TEST IMAGE 4



TEST