```
if (m_bKeyPressed['S']) {
    int temp[3] = { 0, 0, 0 };

    do {
        temp[0] = rand() % 10;
        temp[1] = rand() % 10;
        temp[2] = rand() % 10;
    } while (temp[0] == temp[1] || temp[0] == temp[2] || temp[1] == temp[2]);

    for (int row = 0; row < 6; row++)
        if (!(row % 2)) {
            A[row][0] = m_point1[temp[row / 2]].X;
            A[row][1] = m_point1[temp[row / 2]].Y;
            A[row][2] = 1;

            for (int col = 3; col < 6; col++) A[row][col] = 0;

            y[row] = m_point2[temp[row / 2]].X;
            y[row + 1] = m_point2[temp[row / 2]].Y;
        }
        else {
            for (int col = 0; col < 3; col++) A[row][col] = 0;

            A[row][3] = m_point1[temp[row / 2]].X;
            A[row][4] = m_point1[temp[row / 2]].Y;
            A[row][5] = 1;
        }

    InverseMatrix(A, InverseA, 6);

    for (int i = 0; i < 6; i++) {
        w[i] = 0;
        for (int j = 0; j < 6; j++) {
            w[i] += InverseA[i][j] * y[j];
        }
    }
} else {
    for (int row = 0; row < 20; row++)
        if (!(row % 2)) {
            B[row][0] = m_point1[row / 2].X;
            B[row][1] = m_point1[row / 2].Y;
            B[row][2] = 1;

            for (int col = 3; col < 6; col++) B[row][col] = 0;

            z[row] = m_point2[row / 2].X;
            z[row + 1] = m_point2[row / 2].Y;
        }
        else {
            for (int col = 0; col < 3; col++) B[row][col] = 0;

            B[row][3] = m_point1[row / 2].X;
            B[row][4] = m_point1[row / 2].Y;
            B[row][5] = 1;
        }

    LeastSquared(B, w, z, 20, 6, true, 0.9);
}
```

1. RANSAC

do while 문을 이용해서 중복되지 않는 3개의 int를 temp에 할당.

$$\begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 \\ x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} x_0^* \\ y_0^* \\ x_1^* \\ y_1^* \\ x_2^* \\ y_2^* \end{pmatrix}$$ 다음의 식을 통해 w를 구함.

행렬식을 A * w = y라고 했을 때 w = A^{-1} * y이므로 w를 구할 수 있음.

먼저 행렬 A에 값을 할당함.

이후 InverseMatrix 함수를 이용해 A의 역행렬 InverseA를 구하고 y를 곱해 w를 구함.

LeastSquared 함수를 이용해 B에 할당.

```
int newX = w[0] * m_point1[i].X + w[1] * m_point1[i].Y + w[2];
int newY = w[3] * m_point1[i].X + w[4] * m_point1[i].Y + w[5];
int temp = std::sqrt(std::pow((newX - m_point2[i].X), 2) + std::pow((newY - m_point2[i].Y), 2));

if (temp < 8) inliersCnt++;
```

x* = ax + by + c, y* = dx + ey + f이므로 newX, newY를 구하고 거리를 비교한다.

2. Least Squares

```
for (int row = 0; row < 20; row++)
    if (!(row % 2)) {
        B[row][0] = m_point1[row / 2].X;
        B[row][1] = m_point1[row / 2].Y;
        B[row][2] = 1;

        for (int col = 3; col < 6; col++) B[row][col] = 0;

        z[row] = m_point2[row / 2].X;
        z[row + 1] = m_point2[row / 2].Y;
    }
    else {
        for (int col = 0; col < 3; col++) B[row][col] = 0;

        B[row][3] = m_point1[row / 2].X;
        B[row][4] = m_point1[row / 2].Y;
        B[row][5] = 1;
    }

LeastSquared(B, w, z, 20, 6, true, 0.9);
```

행렬 B에 m_point1의 모든 10개의 점에 대해 할당을 해주고 벡터 z에 m_point2의 모든 10개의 점에 대해 할당을 해준다.
이후 LeastSquared 함수를 이용한다.

3. 결과

1에서 inliner의 개수는 어떠한 경우에도 6을 초과하지 못했다.
2에서 inliner의 개수는 0으로 stitching이 제대로 이루어지지 않았다.