基于 Tsai 方法的照相机标定

《计算机视觉》作业报告

何琪辰 06122477

计算机工程与科学学院

作业要求

给出如图 1所示，对应图像如图 2所示,其中，外圆直径 90mm，内圆直径 60mm，图像中心坐标为：Xc=384，Yc=288，CCD阵面上单位距离的像素点数：Nx=Ny=120 pixels/mm。

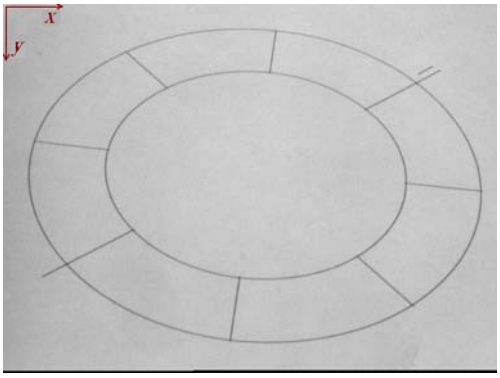


图 1 照相机获取的图片

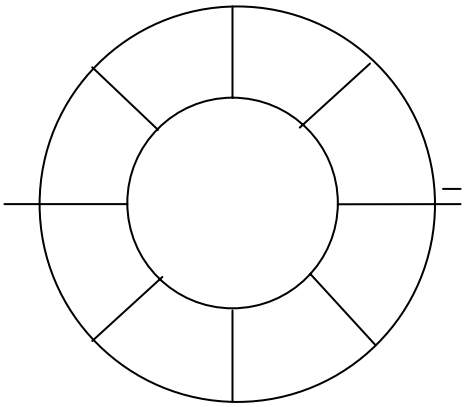


图 2 定义世界坐标系下的真实图片

实施过程

定标程序使用

图 3所示为定标程序界面，图中显示的内容为加载照相机图片后，在右上方显示世界坐标上的点，此时用户选取照片上对应的点，形成点对，供定标时使用。选取的点的顺序如图 4所示。

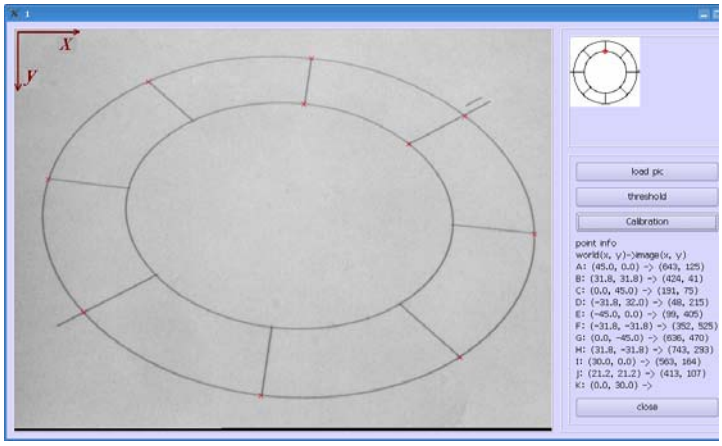


图 3 摄像机定标程序界面

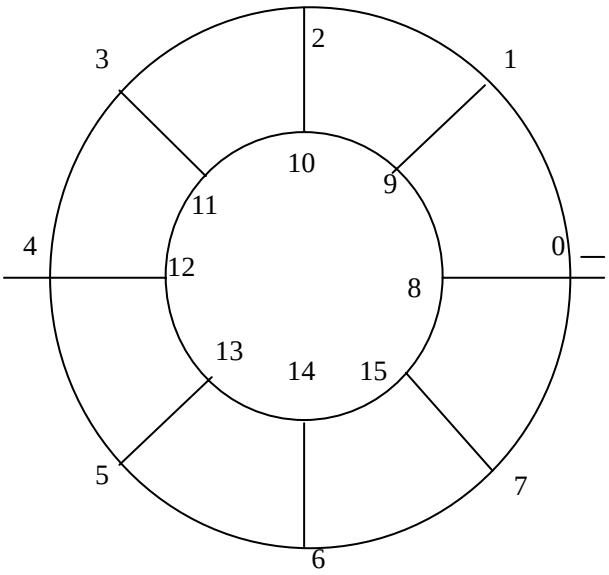


图 4 选取点的顺序

当所有点都选取完成后，程序自动开始定标运算。完成后，用户在右上角的世界坐标图像上选取点，程序会自动将该点映射到照相机坐标上，并且显示出映射点的位置，如图 5所示。

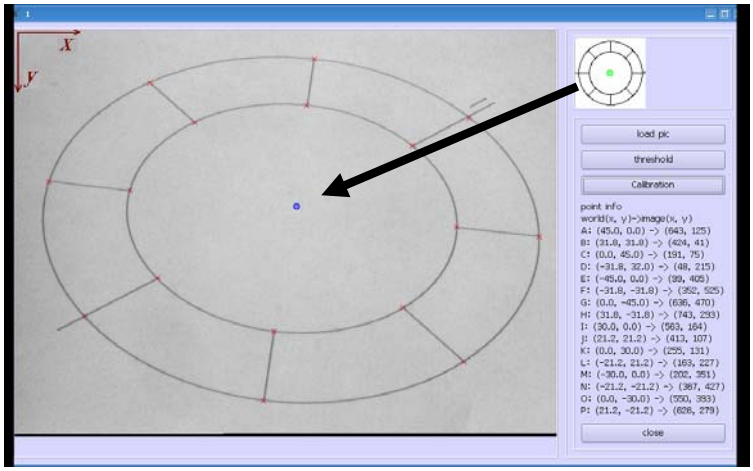


图 5 世界坐标到摄像机坐标的映射

数据示例

现在手工选取一组点进行定标运算。

照片取点如下

0: 642.000	123.000
1: 423.000	41.000
2: 191.000	74.000
3: 49.000	214.000
4: 99.000	406.000
5: 353.000	525.000
6: 637.000	469.000
7: 742.000	292.000
8: 563.000	163.000
9: 413.000	107.000
10: 254.000	131.000
11: 163.000	228.000
12: 202.000	351.000
13: 367.000	426.000
14: 551.000	393.000
15: 626.000	280.000

设 Xn Yn 为图像上的一点，通过下列运算，得到 CCD 像平面上的坐标。

$$\begin{cases} Xd = (Xn - 384)/120 \\ Yd = (Xn - 288)/120 \end{cases}$$

运算结果如下：

ideal x: 2.1500,	y: -1.3750
ideal x: 0.3250,	y: -2.0583
ideal x: -1.6083,	y: -1.7833
ideal x: -2.7917,	y: -0.6167
ideal x: -2.3750,	y: 0.9833
ideal x: -0.2583,	y: 1.9750

ideal x: 2.1083,	y: 1.5083
ideal x: 2.9833,	y: 0.0333
ideal x: 1.4917,	y: -1.0417
ideal x: 0.2417,	y: -1.5083
ideal x: -1.0833,	y: -1.3083
ideal x: -1.8417,	y: -0.5000
ideal x: -1.5167,	y: 0.5250
ideal x: -0.1417,	y: 1.1500
ideal x: 1.3917,	y: 0.8750
ideal x: 2.0167,	y: -0.0667

进行第一部分计算：3D 原点以及 X 和 Y 方向上的位置

现在可以开始对摄像机的外部系统进行定标，确定 R(旋转矩阵)、Tx 和 Ty。根据，选取 5 点来计算参数。

选择序号为 0 1 2 3 4 的 5 个点进行计算

$$ai=[xwYw \quad ywYd \quad yd \quad -xwXd \quad -yxXd]$$

$$\mathbf{A}u=s$$

求得 A 矩阵的增广

-61.875	0	-1.375	-96.75	0	2.15
-65.496	-65.496	-2.058	-10.341	-10.341	0.325
0	-80.25	-1.783	0	72.375	-1.608
19.622	-19.733	-0.617	-88.83	89.333	-2.792
-44.25	0	0.983	-106.875	0	-2.375

解此方程得

$$u = [-0.17834, 0.14328, -0.18057, 0.09440, 0.13220]^T$$

并假设 Ty=4.36413。

由于用 0 号代入检验函数

$$\begin{cases} xc = r1X0 + r2Y0 + Tx \\ yc = r4X0 + r5Y0 + Ty \end{cases}$$

得到 xc=-35.8125, yc=22.9033。由于 xc 与 x 的符号不同，则 Ty 取负值，为 -4.36413。

并计算得到旋转矩阵 R 的值，假设 r3, r6, r7, r8 为正。得

$$\begin{matrix} 0.7783 & -0.6253 & 0.0568 \\ -0.412 & -0.5769 & -0.7053 \\ 2.4033 & 0.5807 & -0.6737 \end{matrix}$$

并计算出 Tx =0.78805。

现在进行第二部分计算：焦距和 Z 轴距离。

由下式建立第二组线性方程：

$$bi = \lfloor r4Xi + r5Yi + Ty \quad yi \rfloor$$

$$vi = (r7Xi + r8Yi)yi$$

得到方程组：

-23	-2	-148.702
-36	-3	-195.436
-31	-2	-46.6
-10	-1	35.6985
14	0	-106.344

解超定方程得：

f = 6.1001　Tz=-23.71264

由于 f>0 则不需要对 Tz　r3 r6 r7 r8 取反。

则该数据的结果为

$$R=\begin{pmatrix}0.7783 & -0.6253 & 0.0568\\ -0.412 & -0.5769 & -0.7053\\ 2.4033 & 0.5807 & -0.6737\end{pmatrix}\quad T=\begin{pmatrix}0.78805\\ -4.36413\\ -23.71264\end{pmatrix}\quad f=6.1001$$

结果分析

仅从结果可以看出，是明显不正确的。通过反复试验，得知图像上取得的点只要有小小的偏差，这些偏差会带入旋转矩阵，特别是 r7 的值，与实际值有 400% 的误差。

0.7783	-0.6253	0.0568
-0.412	-0.5769	-0.7053
2.4033	0.5807	-0.6737

其他值的误差基本在 10%左右，r1, r2, r2, r5 基本稳定。

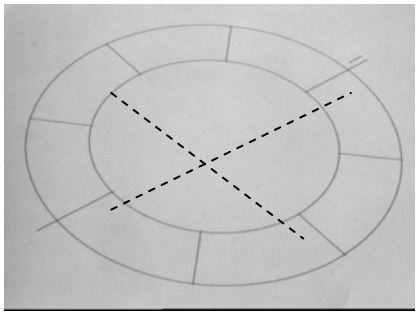
带入第二组超定方程后，误差进一步放大。Tz 基本不可信。

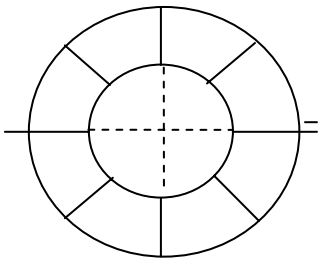
一次定标操作取点有 16 个，对这 16 个点进行排列组合，取不同的组合进行计算，并可以简单的取平均，来减少误差。最终结果共有 4074 个合法方程。

解如下

$$R=\begin{pmatrix}0.7684 & -0.6238 & -0.0263\\ -0.4103 & -0.5686 & 0.1811\\ -1.3232 & -0.1744 & -0.6544\end{pmatrix}\quad T=\begin{pmatrix}0.97616\\ -4.53651\\ 80.80404\end{pmatrix}\quad f=13.79252$$

定标情况如下，出现偏移的现象。虚线是在世界坐标上取得的点在摄像机坐标上的映射，出现了往左下方偏移的情况。现在未能解决该问题。





代码：

世界坐标点定义如下：

```
const double sq2 = sqrt(2) / 2;
m_worldPoints[0][0] = 45.0;
m_worldPoints[0][1] = 0.0;
m_worldPoints[1][0] = 45.0 * sq2;
m_worldPoints[1][1] = 45.0 * sq2;
m_worldPoints[2][0] = 0.0;
m_worldPoints[2][1] = 45.0;
m_worldPoints[3][0] = -45.0 * sq2;
m_worldPoints[3][1] = 32.0;
m_worldPoints[4][0] = -45.0;
m_worldPoints[4][1] = 0.0;
m_worldPoints[5][0] = -45.0 * sq2;
m_worldPoints[5][1] = -45.0 * sq2;
m_worldPoints[6][0] = 0.0;
m_worldPoints[6][1] = -45.0;
m_worldPoints[7][0] = 45.0 * sq2;
m_worldPoints[7][1] = -45.0 * sq2;
m_worldPoints[8][0] = 30.0;
m_worldPoints[8][1] = 0.0;
m_worldPoints[9][0] = 30.0 * sq2;
m_worldPoints[9][1] = 30.0 * sq2;
m_worldPoints[10][0] = 0.0;
m_worldPoints[10][1] = 30.0;
m_worldPoints[11][0] = -30.0 * sq2;
m_worldPoints[11][1] = 30.0 * sq2;
m_worldPoints[12][0] = -30.0;
m_worldPoints[12][1] = 0.0;
m_worldPoints[13][0] = -30.0 * sq2;
m_worldPoints[13][1] = -30.0 * sq2;
m_worldPoints[14][0] = 0.0;
m_worldPoints[14][1] = -30.0;
m_worldPoints[15][0] = 30.0 * sq2;
m_worldPoints[15][1] = -30.0 * sq2;
```

定标程序代码清单：

```
#include <cstdio>
#include <cmath>

bool gauss(double in[5][6], double out[5]);
bool gauss2(double in[2][3], double out[2]);

void calibrate(double cameraPoints[16][2], double worldPoints[16][2], double
```

```
rotate[3][3], double translation[3], double &f)
{
	double A[5][6];
	double ideal[16][2];
	double deals[5][6];
	double s[5];
	double xc, yc;
	double B[5][3];
	double Bi[2][3];
	double ui[2];
	double u[2];
	int uCnt;

	int x, y;
	double sr;
	int t[5];
	int st[2];

	double allF;
	double allR[3][3];
	double allT[3];
	int ansCnt;

	allF = 0.0;
	for (y=0; y<3; ++y)
	{
		allT[y] = 0.0;
		for (x=0; x<3; ++x)
		{
			allR[y][x] = 0.0;
		}
	}
	ansCnt = 0;
	for (y=0; y<16; ++y)
	{
		ideal[y][0] = (cameraPoints[y][0] - 384.0) / 120.0;
		ideal[y][1] = (cameraPoints[y][1] - 288.0) / 120.0;
//		printf("ideal x: %.4lf,\ty: %.4lf\n", ideal[y][0], ideal[y][1]);
	}
	//printf("\n");
	for (t[0]=0; t[0]<12; ++t[0])
	{
		for (t[1]=t[0]+1; t[1]<13; ++t[1])
		{
```

```
            for (t[2]=t[1]+1; t[2]<14; ++t[2])
            {
                for (t[3]=t[2]+1; t[3]<15; ++t[3])
                {
                    for (t[4]=t[3]+1; t[4]<16; ++t[4])
                    {

                        for (y=0; y<5; ++y)
                        {
                            A[y][0] = worldPoints[t[y]][0] * ideal[t[y]][1];
                            A[y][1] = worldPoints[t[y]][1] * ideal[t[y]][1];
                            A[y][2] = ideal[t[y]][1];
                            A[y][3] = -worldPoints[t[y]][0] * ideal[t[y]][0];
                            A[y][4] = -worldPoints[t[y]][1] * ideal[t[y]][0];
                            A[y][5] = ideal[t[y]][0];
                        }

                        for (y=0; y<5; ++y)
                        {
                            for (x=0; x<6; ++x)
                            {
                                //printf("%.3lf\t", A[y][x]);
                            }
                            //printf("\n");
                        }
                        //printf("\n");

                        gauss(A, s);
                        sr = 0.0;
                        //printf("\ns:\n");
                        for (y=0; y<5; ++y)
                        {
                            //printf("%.5lf\n", s[y]);
                            if (y != 2)
                            {
                                sr += (s[y] * s[y]);
                            }
                        }
                        if (fabs(s[0]*s[4] - s[3]*s[1]) < 1e-4)
                        {
                            continue;
                        }
                        translation[1] = sqrt((sr - sqrt(sr*sr -
4*(s[0]*s[4]-s[3]*s[1])*(s[0]*s[4]-s[3]*s[1]))) / (2*(s[0]*s[4] - s[3]*s[1])*(s[0]*s[4] -
```

```
s[3]*s[1])));
                        //printf("Ty=%.5lf\n", translation[1]);
                        rotate[0][0] = s[0] * translation[1];
                        rotate[0][1] = s[1] * translation[1];
                        rotate[1][0] = s[3] * translation[1];
                        rotate[1][1] = s[4] * translation[1];

                        translation[0] = s[2] * translation[1];
                        //printf("tx=%.5lf\n", translation[0]);
                        xc = rotate[0][0] * worldPoints[t[0]][0] +
rotate[0][1]*worldPoints[t[0]][1] + translation[0];
                        yc = rotate[1][0] * worldPoints[t[0]][0] +
rotate[1][1]*worldPoints[t[0]][1] + translation[1];

                        //printf("check ty:       x:%.4lf\ty:%.4lf\n", xc, yc);

                        if (!(worldPoints[t[0]][0]*xc>0 &&
worldPoints[t[0]][1]*yc>0))
                        {
                            //printf("re cal the ty\n");
                            translation[1] = -translation[1];
                            //printf("Ty=%.5lf\n", translation[1]);
                            rotate[0][0] = s[0] * translation[1];
                            rotate[0][1] = s[1] * translation[1];
                            rotate[1][0] = s[3] * translation[1];
                            rotate[1][1] = s[4] * translation[1];

                            translation[0] = s[2] * translation[1];
                            //printf("tx=%.5lf\n", translation[0]);
                        }
                        rotate[0][2] = sqrt(1 - rotate[0][0]*rotate[0][0] -
rotate[0][1]*rotate[0][1]);
                        rotate[1][2] = sqrt(1 - rotate[1][0]*rotate[1][0] -
rotate[1][1]*rotate[1][1]);
                        rotate[2][0] = (1 - rotate[0][0]*rotate[0][0] -
rotate[0][1]*rotate[1][0]) / rotate[0][2];
                        rotate[2][1] = (1 - rotate[0][1]*rotate[1][0] -
rotate[1][1]*rotate[1][1]) / rotate[1][2];
                        rotate[2][2] = -sqrt(1 - rotate[0][2]*rotate[2][0] -
rotate[1][2]*rotate[2][1]);
                        if ((1 - rotate[0][2]*rotate[2][0] - rotate[1][2]*rotate[2][1])
< 0)
                        {
                            printf("r9:%.5lf\t", rotate[2][2]);
```

```
                    break;
                }
                if (rotate[0][0]*rotate[1][0] + rotate[0][1]*rotate[1][1] > 0)
                {
                    rotate[1][2] = -rotate[1][2];
                }
                //printf("rotate:\n");
                for (y=0; y<3; ++y)
                {
                    for (x=0; x<3; ++x)
                    {
                        //printf("%.4lf\t", rotate[y][x]);
                    }
                    //printf("\n");
                }
                //printf("B\n");
                for (y=0; y<5; ++y)
                {
                    B[y][0] = floor(rotate[1][0]*worldPoints[t[y]][0] +
rotate[1][1]*worldPoints[t[y]][1] + translation[1]);
                    B[y][1] = floor(ideal[t[y]][1]);
                    B[y][2] = (rotate[2][0]*worldPoints[t[y]][0] +
rotate[2][1]*worldPoints[t[y]][1]) * ideal[t[y]][1];
                    //printf("%.4lf\t%.4lf\t%.4lf\n", B[y][0], B[y][1],
B[y][2]);
                }
                uCnt = 0;
                u[0] = 0.0;
                u[1] = 0.0;


                /*
                B[0][0] = -20.7641;
                B[1][0] = -7.4782;
                B[2][0] = 7.9075;
                B[3][0] = 16.3802;
                B[4][0] = 12.9768;

                B[0][1] = 1.2917;
                B[1][1] = 0.4833;
                B[2][1] = -0.5417;
                B[3][1] = -1.1833;
                B[4][1] = -0.90;
```

```
                    B[0][2] = -21.0834;
                    B[1][2] = -0.5921;
                    B[2][2] = -7.9030;
                    B[3][2] = -25.8661;
                    B[4][2] = -14.6904;

                    for (y=0; y<5; ++y)
                    {
                        //printf("%.4lf\t%.4lf\t%.4lf\n",      B[y][0],      B[y][1],
B[y][2]);
                    }
                    */

                    for (st[0]=0; st[0]<4; ++st[0])
                    {
                        for (st[1]=st[0]+1; st[1]<5; ++st[1])
                        {
                            for (x=0; x<3; ++x)
                            {
                                Bi[0][x] = B[st[0]][x];
                                Bi[1][x] = B[st[1]][x];
                            }
                            if (gauss2(Bi, ui))
                            {
                                u[0] += ui[0];
                                u[1] += ui[1];
                                ++uCnt;
                            }
                        }
                    }

                    if (0 == uCnt)
                    {
                        continue;
                    }
                    u[0] = u[0] / static_cast<double>(uCnt);
                    u[1] = u[1] / static_cast<double>(uCnt);
                    //printf("f:%.4lf\t\tTz:%.5lf\n", u[0], u[1]);
                    if (u[0] < 0)
                    {
                        u[0] = -u[0];
                        //u[1] = -u[1];
                        rotate[0][2] = -rotate[0][2];
                        rotate[1][2] = -rotate[1][2];
```

```
                                rotate[2][0] = -rotate[2][0];
                                rotate[2][1] = -rotate[2][1];
                        }
                        f = u[0];
                        translation[2] = u[1];
                        printf("r9:%.5lf\t", rotate[2][2]);
                        allF += f;
                        for (y=0; y<3; ++y)
                        {
                                allT[y] += translation[y];
                                for (x=0; x<3; ++x)
                                {
                                        allR[y][x] += rotate[y][x];
                                }
                        }
                        ++ansCnt;
                }
        }
    }
}
printf("\n%d solutions\n", ansCnt);
f = allF / static_cast<double>(ansCnt);
for(y=0; y<3; ++y)
{
    translation[y] = allT[y] / static_cast<double>(ansCnt);
    for (x=0; x<3; ++x)
    {
        rotate[y][x] = allR[y][x] / static_cast<double>(ansCnt);
    }
}

for (y=0; y<3; ++y)
{
    for (x=0; x<3; ++x)
    {
        printf("%.4lf\t", rotate[y][x]);
    }
    printf("\n");
}
printf("\n");
for (y=0; y<3; ++y)
{
    printf("%.5lf\n", translation[y]);
```

```
    }
    printf("f=%.5lf\n", f);
}

bool gauss2(double in[2][3], double out[2])
{
    double B[2][3];
    int x, y;
    double tmp;
    for (y=0; y<2; ++y)
    {
        for (x=0; x<3; ++x)
        {
            B[y][x] = in[y][x];
        }
    }

    if (fabs(B[0][0]) < fabs(B[1][0]))
    {
        for (x=0; x<3; ++x)
        {
            tmp = B[0][x];
            B[0][x] = B[1][x];
            B[1][x] = tmp;
        }
    }
    if (fabs(B[0][0]) < 1e-4)
    {
        return false;
    }
    tmp = B[1][0] / B[0][0];
    B[1][0] = 0.0;
    B[1][1] -= (tmp * B[0][1]);
    B[1][2] -= (tmp * B[0][2]);
    if (fabs(B[1][1]) < 1e-4)
    {
        return false;
    }
    tmp = B[0][1] / B[1][1];
    B[0][1] = 0.0;
    B[0][0] -= (tmp * B[1][0]);
    B[0][2] -= (tmp * B[1][2]);
    if (fabs(B[0][0]) < 1e-4)
    {
```

```
            return false;
        }
        out[0] = B[0][2] / B[0][0];
        out[1] = B[1][2] / B[1][1];
        return true;
}

bool gauss(double in[5][6], double out[5])
{
        //printf("gauss\n");
        int x, y;
        int t;
        int maxx;
        double tmp;
        double A[5][6];
        int sr;
        for (y=0; y<5; ++y)
        {
                for (x=0; x<6; ++x)
                {
                        A[y][x] = in[y][x];
                }
        }
        for (t=0; t<5; ++t)
        {
                for (y=0; y<5; ++y)
                {
                        for (x=0; x<6; ++x)
                        {
                                //printf("%.3lf\t", A[y][x]);
                        }
                        //printf("\n");
                }

                //find the max mian elem
                maxx = t;
                for (y=t+1; y<5; ++y)
                {
                        if (fabs(A[y][t]) > fabs(A[maxx][t]))
                        {
                                maxx = y;
                        }
                }
                //printf("%.5lf\n", A[maxx][t]);
```

```
        if (fabs(A[maxx][t]) < 1e-5)
        {
            return false;
        }

        //change the row
        for (x=t; x<6; ++x)
        {
            tmp = A[maxx][x];
            A[maxx][x] = A[t][x];
            A[t][x] = tmp;
        }

        //down
        for (y=t+1; y<5; ++y)
        {
            tmp = A[y][t] / A[t][t];
            A[y][t] = 0;
            for (x=t+1; x<6; ++x)
            {
                A[y][x] -= A[t][x]*tmp;
            }
        }
    }
    for (t=4; t>=0; --t)
    {
        if (fabs(A[t][t]) < 1e-4)
        {
            return false;
        }
        for (y=t-1; y>=0; --y)
        {
            tmp = A[y][t] / A[t][t];
            A[y][t] = 0;
            A[y][5] -= A[t][5]*tmp;
        }
    }

    for (y=0; y<5; ++y)
    {
        for (x=0; x<6; ++x)
        {
            //printf("%.3lf\t", A[y][x]);
        }
```

```
            //printf("\n");
        }

        for (t=0; t<5; ++t)
        {
            if (fabs(A[t][t]) < 1e-4)
            {
                return false;
            }
            out[t] = A[t][5] / A[t][t];
        }
        return true;
}
```

参考文献：

[1]. Tasi R Y. 1987. A versatile camera calibration technique for high-accuracy 3D Machine vision metrology using Off-theShelf TV Camera and Lenses. Journal of Robotics and Automation, 3(4): 323~344

[2]. David A. Forsyth, Jean Ponce, 2003, Computer Vistion a modern approach. Pearson Education Asia Limited.

[3]. Berthold K.P. Horn. 2000. Tsai's camera calibration method revisited.

[4]. Fabio Remondino, Clive Fraser. 2006. Digital camera calibration methods: considerations and comparisions. ISPRS Commission V symposium 'Image Engineering and Vision Metrology'.