Tslib 中校准原理及其算法实现

（1）触摸屏为什么需要校正？

触摸屏与 LCD 显示屏是两个不同的物理器件。LCD 处理的像素，例如我们通常所说的分辨率是 600x800，实际就是指每行的宽度是 600 个像素，高度是 800 个像素，而触摸屏处理的数据是点的物理坐标，该坐标是通过触摸屏控制器采集到的。两者之间需要一定的转换。

其次， 在安装触摸屏时，不可避免的存在着一定的误差，如旋转，平移的，这同样需要校正解决。

再次，电阻式触摸屏的材料本身有差异而且随着时间的推移，其参数也会有所变化，因此需要经常性的校正（电容式触摸屏只需要一次校正即可，这是由两者不同的材料原理造成的，具体可参阅有关电阻式和电容式触摸屏对比的文章）

（2）如何校正？

触摸屏的校正过程一般为： 依次在屏幕的几个不同位置显示某种标记（如"+"）， 用触摸笔点击这些标记， 完成校正。如果$P_T(x, y)$表示触摸屏上的一个点， $P_L(x, y)$表示LCD上的一个点,校正的结果就是得到一个转换矩阵M, 使$P_L(x, y) = M \cdot P_T(x, y)$。

(3) 校正原理

我们知道二维几何变换包含三种平移、旋转和缩放。这三者的矩阵表示为：

平移M_T：

$$\begin{pmatrix} X_L \\ Y_L \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & T_X \\ 0 & 1 & T_Y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_T \\ Y_T \\ 1 \end{pmatrix}$$

缩放M_S：

$$\begin{pmatrix} X_L \\ Y_L \\ 1 \end{pmatrix} = \begin{pmatrix} S_X & 0 & 0 \\ 0 & S_Y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_T \\ Y_T \\ 1 \end{pmatrix}$$

旋转M_R：

$$\begin{pmatrix} X_L \\ Y_L \\ 1 \end{pmatrix} = \begin{pmatrix} Cos\theta & -Sin\theta & 0 \\ Sin\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_T \\ Y_T \\ 1 \end{pmatrix}$$

所以 $P_L = M_R \cdot M_T \cdot M_S \cdot P_T$, 将这个公式展开，其结果为：

$$X_L = X_T(S_X\,Cos\theta) + Y_T\,(-S_Y\,Sin\theta) + (T_X\,Cos\theta - T_Y Sin\theta)$$
$$Y_L = X_T(S_X\,Sin\theta) + Y_T\,(S_Y\,Cos\theta) + (T_X\,Sin\theta + T_X Cos\theta)$$

在上面的公式中，LCD上的坐标（X_L、Y_L）和触摸屏上的坐标（X_T、Y_T）是已知的，而其他的则是我们需要求的：θ，S_Y, S_X, T_Y, S_X共有 5 个变量，至少需要五个方程，因为每组点坐标（P_L, P_T）可以得到两个方程，因此我们需要采集三组点坐标。但是上面的方程涉及三角函数，运算复杂，我们可以进一步简化为：

$$X_L = X_T A + Y_T\,B + C$$
$$Y_L = X_T D + Y_T\,E + F$$

公式（1）

变量虽然多了一个，但是解题过程简单多了，更适合计算机计算，而且采集点的数量仍然为 3 组。假设LCD三个点的坐标为（X_{L1}, Y_{L1}），（X_{L2}, Y_{L2}），（X_{L2}, Y_{L2}），对应触摸屏上的三个点是（X_{T1}, Y_{T1}），（X_{T2}, Y_{T2}）。（X_{T3}, Y_{T3}），则联立两个方程组为：

$$\begin{cases} X_{L1} = X_{T1}A + Y_{T1}\,B + C \\ X_{L2} = X_{T2}A + Y_{T2}\,B + C \\ X_{L3} = X_{T3}A + Y_{T3}\,B + C \end{cases}$$

$$\begin{cases} Y_{L1} = X_{T1}D + Y_{T1}\,E + F \\ Y_{L2} = X_{T2}D + Y_{T2}\,E + F \\ Y_{L3} = X_{T3}D + Y_{T3}\,E + F \end{cases}$$

这样，触摸屏的校正实际上就是解上面的方程组，得到 6 个系数：A、B、C、D、E、F。而上面方程组按照克莱姆法则解即可。在得到 6 个系数后，以后通过触摸屏得到的所有坐标，带入公式（1）中就可以得到 LCD 上以像素表示的坐标。

附:克拉姆法则

克莱姆法则　设有方程组

$$
(1)\qquad \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n, \end{cases}
$$

如果（1）的系数行列式不等于零，即

$$
D = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} \neq 0,
$$

那么，方程组（1）有唯一解：

$$
x_1 = \frac{D_1}{D}, x_2 = \frac{D_2}{D}, \cdots, x_n = \frac{D_n}{D}. (2)
$$

其中行列式 $D_j (j = 1,2,\cdots,n)$ 是把 D 的第 j 列元素用方程组右端的常数项代替后得到的 n 阶行列式

$$
D_j = \begin{vmatrix} a_{11} & \cdots & a_{1,j-1} & b_1 & a_{1,j+1} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2,j-1} & b_2 & a_{2,j+1} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & \cdots & a_{n,j-1} & b_n & a_{n,j+1} & \cdots & a_{nn} \end{vmatrix}.
$$

在上面说过，只需要三组点坐标，我们就可以完成触摸屏的校正，其基本公式为：

$$X_L = X_T A + Y_T B + C$$
$$Y_L = X_T D + Y_T E + F \qquad \text{公式（1）}$$

实际上，在校正时，采集的触摸屏的点坐标有一定的误差，也就是说采集几个三组点坐标，分别计算 A、B、C、D、E、F,其结果不尽相同。在 tslib 的 ts_calibrate 中，采集了五组点坐标，具体代码参见 ts_calibrate.c 中的 perform_calibration()。一般来说，采集的点越多，校正的精确性就越高。

为了在计算过程中兼顾 5 个点的坐标， ts_calibrate 将公式（1）变换如下：

$$X_L = X_T A + Y_T B + C$$
$$X_L \cdot X_T = (X_T)^2 A + Y_T X_T B + X_T C$$
$$X_L \cdot Y_T = Y_T X_T A + (Y_T)^2 B + Y_T C \qquad \text{公式（2）}$$

$$Y_L = X_T D + Y_T E + F$$
$$Y_L \cdot X_T = (X_T)^2 D + Y_T X_T E + X_T F$$
$$Y_L \cdot Y_T = Y_T X_T D + (Y_T)^2 E + Y_T F$$

以第一组（A、B、C）为例， 进一步变换为：

$$\sum X_L = (\sum X_T)A + (\sum Y_T)B + nC$$
$$\sum (X_L \cdot X_T) = \sum ((X_T)^2) A + \sum (Y_T X_T) B + \sum X_T C$$
$$\sum (X_L \cdot Y_T) = \sum (Y_T X_T) A + \sum (Y_T)^2 B + \sum Y_T C \qquad \text{公式（3）}$$

n表示坐标的数量，ts_calibrate中就是 5, 分别对X_T, Y_T, X_L, $X_L X_T$, $X_L Y_T$, $(X_T)^2$, $(Y_T)^2$, Y_T 求和，带入公式（3）中，就可以求出A、B、C，同理可求D、E、F。

解的时候用的是逆矩阵的方法，即：

$P_0 = M \cdot P_1$ ======> $(M)^{-1} P_0 = P_1$

我们可以看出，运用上述方法可以处理任意多的采集点，而不局限于 5 个，只是采集点过多就会冗余，对校正精确性的提高作用很少，反而增加了计算时间。

采用上述算法的程序源代码如下:

```
int do_calibration(void)
{
    int j;
    float n, x, y, x2, y2, xy, z, zx, zy;
    float det, det1, det2, det3;
    float scaling = 65536.0;

    n = x = y = x2 = y2 = xy = 0;
    for (j = 0; j < 5; j++)
    {
```

```
        n += 1.0;
        x += (float)cal.x[j];
        y += (float)cal.y[j];
        x2 += (float)(cal.x[j] * cal.x[j]);
        y2 += (float)(cal.y[j] * cal.y[j]);
        xy += (float)(cal.x[j] * cal.y[j]);
    }

    det = n * (x2*y2 - xy*xy) + x * (xy*y - x*y2) + y * (x*xy - y*x2);
    if (det < 0.1 && det > -0.1)
    {
        printf("Determinant is too small!\n");
        return 1;
    }

    z = zx = zy = 0;
    for (j = 0; j < 5; j++)
    {
        z += (float)cal.xfb[j];
        zx += (float)(cal.xfb[j] * cal.x[j]);
        zy += (float)(cal.xfb[j] * cal.y[j]);
    }

    det1 = n * (zx*y2 - xy*zy) + z * (xy*y - x*y2) + y * (x*zy - y*zx);
    det2 = n * (x2*zy - zx*xy) + x * (zx*y - x*zy) + z * (x*xy - y*x2);
    det3 = z * (x2*y2 - xy*xy) + x * (xy*zy - zx*y2) + y * (zx*xy - zy*x2);

    cal.a[0] = (int)((det1 / det) * scaling);
    cal.a[1] = (int)((det2 / det) * scaling);
    cal.a[2] = (int)((det3 / det) * scaling);

    printf("%10d %10d %10d\n", cal.a[0], cal.a[1], cal.a[2]);

    z = zx = zy = 0;
    for (j = 0; j < 5; j++)
    {
        z += (float)cal.yfb[j];
        zx += (float)(cal.yfb[j] * cal.x[j]);
        zy += (float)(cal.yfb[j] * cal.y[j]);
    }

    det1 = n * (zx*y2 - xy*zy) + z * (xy*y - x*y2) + y * (x*zy - y*zx);
    det2 = n * (x2*zy - zx*xy) + x * (zx*y - x*zy) + z * (x*xy - y*x2);
    det3 = z * (x2*y2 - xy*xy) + x * (xy*zy - zx*y2) + y * (zx*xy - zy*x2);
```

```
    cal.a[3] = (int)((det1 / det) * scaling);
    cal.a[4] = (int)((det2 / det) * scaling);
    cal.a[5] = (int)((det3 / det) * scaling);

    cal.a[6] = (int)scaling;

    return 0;
}
```

相关的数据结构如下:

```
typedef struct {
    int x[5], xfb[5];
    int y[5], yfb[5];
    unsigned int a[7];
} calibration;
```

其中,x 和 y 分别表示五个点在触摸板上的坐标,xfb 和 yfb 分别表示 5 个点在 lcd 屏幕上的坐标,a 数组从 a[0]到 a[5]分别为 A、B、C、D、E、F 和一个除数，用于模拟浮点运算。

刘言强
2009-11-3 于上海