ANALOG DEVICES

AN-1021
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

基于MMSE的触摸屏应用多点校准算法

作者：Ning Jia

简介

现代设备多以液晶屏及触摸屏技术作为用户界面。因其简单的构造和众所周知的操作原理，电阻式触摸屏是讲求成本型设计的首选。然而，电阻式触摸屏存在的机械对齐误差及缩放因子，影响了触摸屏产生的X、Y坐标。因此，很难将触摸屏的坐标与其后显示屏(液晶屏等)完全对齐。含有触摸屏的最终产品出厂时，必须首先执行校准算法。

针对触摸屏的经典校准算法是一种基于三个参照点的三点校准算法。这种经典三点校准算法效率高、效果好，但当触摸屏较大时，性能较低。本应用笔记针对电阻式触摸屏，提出了一种基于最小均方误差(MMSE)的多点校准算法，采用三个以上的参照点。数学推导和实验均证明，本算法的精度优于经典三点校准算法。

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

目录

数学原理

图1所示红色圆形的理想中心为O(原点)，理想半径为R。假定该红色圆形代表的是触摸屏下方液晶屏所显示的图像。蓝色椭圆表示用户沿着液晶屏所示红色圆形进行触摸时，触摸屏产生的一组点(图中为夸张的显示)。因此，受电阻式触摸屏本身机械对齐误差和缩放因子的影响，重构后的图像在各坐标轴上均以不同的因子旋转、平移和缩放。校准算法面临的挑战是如何将触摸屏产生的坐标转换成精确代表所示图像的一组坐标。

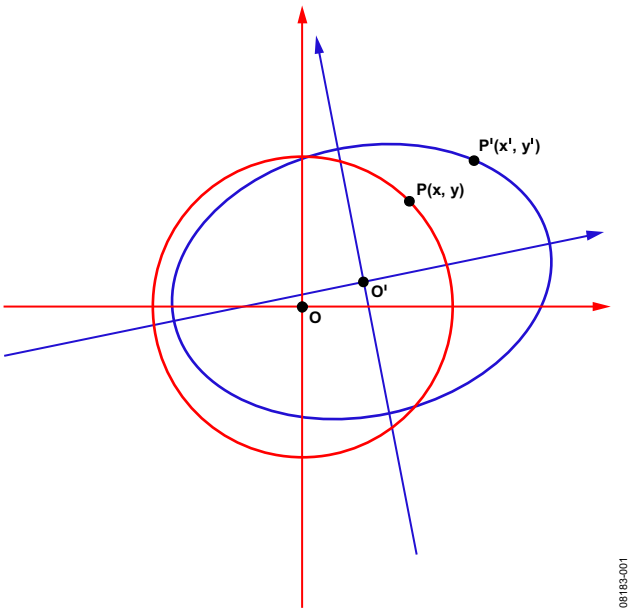

图 1. 触摸屏误差

设存在一点，其理想坐标为P(x, y)，且触摸屏产生的坐标为P'(x', y')。(P点位于红色圆形上，与其对应的P'点则位于蓝色椭圆上。)设P'(x', y') 旋转θ角度，然后在X和Y轴分别缩放KX和KY，分别平移T_X和T_Y，可以回到其理想坐标P(x, y)。校准算法的目的是计算系数θ、K_X、K_Y、T_X和T_Y。然后可利用这些系数对触摸屏直接产生的坐标进行校准。

为了便于大家理解以上分析，不妨假设P'(x', y')为触摸屏直接产生的坐标，其等价极坐标表达式P'(Rcos θ0, Rsin θ0)。另外假设，与P'(x',y')对应的理想坐标为P(x,y)。鉴于P'(x',y')与P(x,y)的关系，可得：

$$P(x, y) = P(K_X R\cos(\theta_0 + \theta) + T_X, K_Y R\sin(\theta_0 + \theta) + T_Y)$$

据三角函数，

$$\cos(\theta_0 + \theta) = \cos\theta_0 \cos\theta - \sin\theta_0 \sin\theta$$

及

$$\sin(\theta_0 + \theta) = \sin\theta_0 \cos\theta + \cos\theta_0 \sin\theta$$

可得以下方程式：

$$\begin{cases} x' = R\cos\theta_0 \\ y' = R\sin\theta_0 \end{cases}$$

$$\begin{cases} x = K_X R\cos(\theta_0 + \theta) + T_X = K_X R(\cos\theta_0 \cos\theta - \sin\theta_0 \sin\theta) + T_X \\ y = K_Y R\sin(\theta_0 + \theta) + T_Y = K_Y R(\sin\theta_0 \cos\theta - \cos\theta_0 \sin\theta) + T_Y \end{cases}$$

则有

$$\begin{cases} x = \cos\theta K_X x' - \sin\theta K_X y' + T_X \\ y = \cos\theta K_Y y' - \sin\theta K_Y x' + T_Y \end{cases}$$

其中：

θ、K_X、K_Y、T_X和T_Y均为常数。

设：

$$\cos\theta K_X = KX_1$$

$$-\sin\theta K_X = KX_2$$

$$T_X = KX_3$$

$$\sin\theta K_Y = KY_1$$

$$\cos\theta K_Y = KY_2$$

$$T_Y = KY_3$$

则前述方程式可改写为：

$$\begin{cases} x = KX_1 x' + KX_2 y' + KX_3 \\ y = KY_1 y' + KY_2 y' + KY_3 \end{cases}$$

可利用以上方程式来校准P'(x', y')，从而得到P(x, y)。对于X轴和Y轴而言，各方程式中均含有三个未知系数。

经典三点校准算法

通过上述分析，我们得到了X轴或Y轴的校准方程式。对于X轴或Y轴，各方程式均含有三个未知系数。因此，如果可获得三个不相关参照点的信息，则可构建出线性方程组，通过解方程来获得未知系数的解。

设三个参照点的理想坐标为(x_0, y_0)、(x_1, y_1)和(x_2, y_2)，其对应的采样坐标为(x'_0, y'_0)、(x'_1, y'_1)和(x'_2, y'_2)，则X轴和Y轴的方程式分别为

$$\begin{cases} x_0 = KX_1x'_0 + KX_2y'_0 + KX_3 \\ x_1 = KX_1x'_1 + KX_2y'_1 + KX_3 \\ x_2 = KX_1x'_2 + KX_2y'_2 + KX_3 \end{cases}$$

及

$$\begin{cases} y_0 = KY_1x'_0 + KY_2y'_0 + KY_3 \\ y_1 = KY_1x'_1 + KY_2y'_1 + KY_3 \\ y_2 = KY_1x'_2 + KY_2y'_2 + KY_3 \end{cases}$$

基于此，可将以上述方程改写为矩阵模式：

$$\begin{bmatrix} x'_0 & y'_0 & 1 \\ x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \end{bmatrix} \bullet \begin{bmatrix} KX_1 \\ KX_2 \\ KX_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

及

$$\begin{bmatrix} x'_0 & y'_0 & 1 \\ x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \end{bmatrix} \bullet \begin{bmatrix} KY_1 \\ KY_2 \\ KY_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

在此基础上，可以通过解方程算出校准系数KX_1、KX_2、KX_3、KY_1、KY_2和KY_3。

通过消元法求得的结果如下：

设 $k = (x'_0 - x'_2)(y'_1 - y'_2) - (x'_1 - x'_2)(y'_0 - y'_2)$ 则

$$KX_1 = \frac{(x_0 - x_2)(y'_1 - y'_2) - (x_1 - x_2)(y'_0 - y'_2)}{k}$$

$$KX_2 = \frac{(x_1 - x_2)(x'_1 - x'_2) - (x_0 - x_2)(x'_1 - x'_2)}{k}$$

$$KX_3 = \frac{y'_0(x'_2x_1 - x'_1x_2) + (y'_1x'_0x_2 - x'_2x_0) + (y'_2x'_1x_0 - x'_0x_1)}{k}$$

$$KY_1 = \frac{(y_0 - y_2)(x'_1 - x'_2) - (y_1 - y_2)(y'_0 - y'_2)}{k}$$

$$KY_2 = \frac{(y_1 - y_2)(x'_0 - x'_2) - (y_0 - y_2)(x'_1 - x'_2)}{k}$$

$$KY_3 = \frac{y'_0(x'_2y_1 - x'_1y_2) + (y'_1x'_0y_2 - x'_2y_0) + (y'_2x'_1y_0 - x'_0y_1)}{k}$$

基于MMSE的多点校准算法

利用经典三点校准算法得到的系数，可以将三个参照点校准到理想位置。然而，对于不在参照点附近的其他点，校准性能不甚理想，当触摸屏相对较大时尤其如此。示例部分的实验结果同样证实了这个问题。因此，可考虑利用三个以上的参照点，以获得最佳校准系数。

设有N + 1个参照点(N + 1 > 3)，其理想坐标为(x_0, y_0)、(x_1, y_1)…(x_N, y_N)，且其对应的采样坐标为(x'_0, y'_0)、(x'_1, y'_1)…(x'_N, y'_N)，则相应的方程式如下：

$$\begin{cases} x_0 = KX_1x'_0 + KX_2y'_0 + KX_3 \\ x_1 = KX_1x'_1 + KX_2y'_1 + KX_3 \\ \vdots \\ x_N = KX_1x'_N + KX_2y'_N + KX_3 \end{cases}$$

及

$$\begin{cases} y_0 = KY_1x'_0 + KY_2y'_0 + KY_3 \\ y_1 = KY_1x'_1 + KY_2y'_1 + KY_3 \\ \vdots \\ y_N = KY_1x'_N + KY_2y'_N + KY_3 \end{cases}$$

注意，各方程组中的方程数(N + 1)均大于未知系数的个数(3)。

可见，我们的目的是算出适合全部(N + 1)个参照点的最佳校准系数。获得最佳系数的方法是遵循MMSE规则。这正是这种算法称为基于MMSE的多点校准算法的原因所在。

以X轴为例，定义一个目标函数

$$FX = \sum_{i=0}^{N} \left(KX_1x'_i + KX_2y'_i + KX_3 - x_i\right)^2$$

其中，FX为参照点的误差平方和。

KX_1、KX_2和KX_3的最佳系数指使目标函数FX最小的那些系数。因此，可使用以下方程式：

$$\begin{cases}\dfrac{\partial FX}{\partial KX_1}=0\\ \dfrac{\partial FX}{\partial KX_2}=0\\ \dfrac{\partial FX}{\partial KX_3}=0\end{cases}$$

即

$$\begin{cases}\dfrac{\partial FX}{\partial KX_1}=\sum_{i=0}^{N}2x_i'\left(KX_1x_i'+KX_2y_i'+KX_3-x_i\right)=0\\ \dfrac{\partial FX}{\partial KX_2}=\sum_{i=0}^{N}2y_i'\left(KX_1x_i'+KX_2y_i'+KX_3-x_i\right)=0\\ \dfrac{\partial FX}{\partial KX_3}=\sum_{i=0}^{N}2\left(KX_1x_i'+KX_2y_i'+KX_3-x_i\right)=0\end{cases}$$

以上方程式可简化为

$$\begin{cases}\left(\sum_{i=0}^{N}x_i'^2\right)KX_1+\left(\sum_{i=0}^{N}x_i'y_i'\right)KX_2+\left(\sum_{i=0}^{N}x_i'\right)KX_3=\sum_{i=0}^{N}x_i'x_i'\\ \left(\sum_{i=0}^{N}x_i'y_i'\right)KX_1+\left(\sum_{i=0}^{N}y_i'\right)KX_2+\left(\sum_{i=0}^{N}y_i'\right)KX_3=\sum_{i=0}^{N}y_i'x_i'\\ \left(\sum_{i=0}^{N}x_i'\right)KX_1+\left(\sum_{i=0}^{N}y_i'\right)KX_2+N\cdot KX_3=\sum_{i=0}^{N}x_i\end{cases}$$

设

$$R=\begin{bmatrix}\sum_{i=0}^{N}x_i'^2 & \sum_{i=0}^{N}x_i'y_i' & \sum_{i=0}^{N}x_i'\\ \sum_{i=0}^{N}x_i'y_i' & \sum_{i=0}^{N}y_i'^2 & \sum_{i=0}^{N}y_i'\\ \sum_{i=0}^{N}x_i' & \sum_{i=0}^{N}y_i' & N\end{bmatrix},\quad KX=\begin{bmatrix}KX_1\\KX_2\\KX_3\end{bmatrix},\quad BX=\begin{bmatrix}\sum_{i=0}^{N}x_i'x_i\\ \sum_{i=0}^{N}y_i'x_i\\ \sum_{i=0}^{N}x_i\end{bmatrix}$$

基于此，可将各方程式改写为矩阵模式R • KX = BX，并可通过求解前述方程组算出最佳系数 KX = R^{-1} • BX。
与X轴类似，设

$$KY=\begin{bmatrix}KY_1\\KY_2\\KY_3\end{bmatrix},\quad BY=\begin{bmatrix}\sum_{i=0}^{N}x_i'y_i\\ \sum_{i=0}^{N}y_i'y_i\\ \sum_{i=0}^{N}y_i\end{bmatrix}$$

然后即可通过求解方程组R • KY = BY，算出Y轴的最佳系数 KY = R^{-1} • BY。

通过消元法求得的结果如下：

设

$$a_0=\frac{\sum x_i'^2}{\sum x_i'},\quad a_1=\frac{\sum x_i'y_i'}{\sum y_i'},\quad a_2=\frac{\sum x_i'}{N}$$

$$b_0=\frac{\sum x_i'y_i'}{\sum x_i'},\quad b_1=\frac{\sum y_i'^2}{\sum y_i'},\quad b_2=\frac{\sum y_i'}{N},$$

$$c_0=\frac{\sum x_i'x_i'}{\sum x_i'},\quad c_1=\frac{\sum y_i'x_i}{\sum y_i'},\quad c_2=\frac{\sum x_i'}{N},$$

$$d_0=\frac{\sum x_i'y_i}{\sum x_i'},\quad d_1=\frac{\sum y_i'x_i}{\sum y_i'},\quad d_2=\frac{\sum y_i'}{N}$$

其中，$\Sigma\bullet$表示$\sum_{i=0}^{N}\bullet$

基于此，可将方程组 R • KX = BX 和 R • KY = BY 分别改写为

$$\begin{bmatrix}a_0 & b_0 & 1\\ a_1 & b_1 & 1\\ a_2 & b_2 & 1\end{bmatrix}\bullet\begin{bmatrix}KX_1\\KX_2\\KX_3\end{bmatrix}=\begin{bmatrix}c_0\\c_1\\c_2\end{bmatrix}$$

及

$$\begin{bmatrix}a_0 & b_0 & 1\\ a_1 & b_1 & 1\\ a_2 & b_2 & 1\end{bmatrix}\bullet\begin{bmatrix}KY_1\\KY_2\\KY_3\end{bmatrix}=\begin{bmatrix}d_0\\d_1\\d_2\end{bmatrix}$$

不难发现，以上方程组的格式与前面经典三点算法的方程组相同：

$$\begin{bmatrix}x_0' & y_0' & 1\\ x_1' & y_1' & 1\\ x_2' & y_2' & 1\end{bmatrix}\bullet\begin{bmatrix}KX_1\\KX_2\\KX_3\end{bmatrix}=\begin{bmatrix}x_0\\x_1\\x_2\end{bmatrix}$$

及

$$\begin{bmatrix}x_0' & y_0' & 1\\ x_1' & y_1' & 1\\ x_2' & y_2' & 1\end{bmatrix}\bullet\begin{bmatrix}KY_1\\KY_2\\KY_3\end{bmatrix}=\begin{bmatrix}y_0\\y_1\\y_2\end{bmatrix}$$

在此基础上，可利用相同的公式计算系数的结果。

设

$$k=(a_0-a_2)(b_1-b_2)-(a_1-a_2)(b_0-b_2)$$

则

$$KX_1=\frac{(c_0-c_2)(b_1-b_2)-(c_1-c_2)(b_0-b_2)}{k}$$

$$KX_2=\frac{(c_1-c_2)(a_0-a_2)-(c_0-c_2)(a_1-a_2)}{k}$$

$$KX_3=\frac{b_0(a_2c_1-a_1c_2)+b_1(a_0c_2-a_2c_0)+b_2(a_1c_0-a_0c_1)}{k}$$

$$KY_1=\frac{(d_0-d_2)(b_1-b_2)-(d_1-d_2)(b_0-b_2)}{k}$$

$$KY_2=\frac{(d_1-d_2)(a_0-a_2)-(d_0-d_2)(a_1-a_2)}{k}$$

$$KY_3=\frac{b_0(a_2d_1-a_1d_2)+b_1(a_0d_2-a_2d_0)+b_2(a_1d_0-a_0d_1)}{k}$$

基于MMSE的多点校准算法的分析

通过计算a_0、a_1、a_2；b_0、b_1、b_2；c_0、c_1、c_2和d_0、d_1、d_2可以发现，可将a_0、a_1、a_2当作基于三种不同方法的x'i的加权平均值，可将b_0、b_1、b_2看作基于三种不同方法的y'i的加权平均值。类似地，可将c_0、c_1、c_2当作基于三种不同方法的xi的加权平均值，d_0、d_1、d_2则可视为基于三种不同方法的yi的加权平均。

由于最终方程组与前面经典三点算法的方程组具有相同格式，因此，可将基于MMSE规则的算法视为另一种三点算法。然而，基于MMSE的算法的区别在于，其采用的不是任何参照点的直接信息，而是参照点的加权平均信息。换言之，先要算出N + 1个直接采样点(x'_0, y'_0)、(x'_1, y'_1)…(x'_N, y'_N)的三个加权平均点(a_0, b_0)、(a_1, b_1)、(a_2, b_2)。这三个加权平均点对应的理想坐标为(c_0, d_0)、(c_1, d_1)和(c_2, d_2)。可见，基于MMSE的算法等效于对这三个加权平均点而言的经典三点算法。

基于MMSE的多点校准算法的步骤

完成基于MMSE的多点校准算法的下列步骤：

1. 选择N +1 (N + 1 > 3)个参照点(x_0, y_0)、(x_1, y_1)…(x_N, y_N)。
2. 得到触摸屏产生的参照点的采样坐标(x'_0, y'_0)、(x'_1, y'_1)…(x'_N, y'_N)。
3. 利用本应用笔记给出的公式，算出校准系数KX和KY。包括以下公式：

$$k=(a_0-a_2)(b_1-b_2)-(a_1-a_2)(b_0-b_2)$$

$$KX_1=\frac{(c_0-c_2)(b_1-b_2)-(c_1-c_2)(b_0-b_2)}{k}$$

$$KX_2=\frac{(c_1-c_2)(a_0-a_2)-(c_0-c_2)(a_1-a_2)}{k}$$

$$KX_3=\frac{b_0(a_2c_1-a_1c_2)+b_1(a_0c_2-a_2c_0)+b_2(a_1c_0-a_0c_1}{k}\quad)$$

$$KY_1=\frac{(d_0-d_2)(b_1-b_2)-(d_1-d_2)(b_0-b_2)}{k}$$

$$KY_2=\frac{(d_1-d_2)(a_0-a_2)-(d_0-d_2)(a_1-a_2)}{k}$$

$$KY_3=\frac{b_0(a_2d_1-a_1d_2)+b_1(a_0d_2-a_2d_0)+b_2(a_1d_0-a_0d_1)}{k}$$

4. 在正常操作中，利用校准系数(KX, KY)和下列方程式计算P'(x', y')的校准点。

$$\begin{cases}x=KX_1x'+KX_2y'+KX_3\\ y=KY_1x'+KY_2y'+KY_3\end{cases}$$

示例

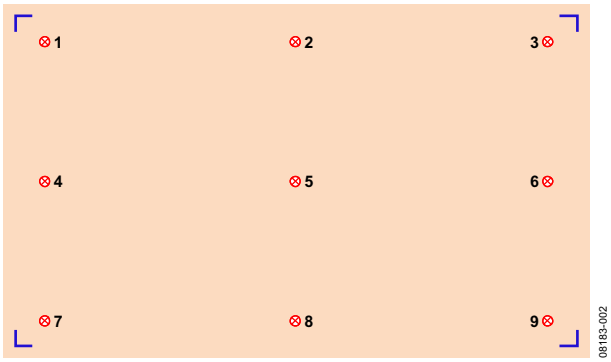

图 2. 选择参照点

如图2所示，从触摸屏上选择了九个点。其理想坐标为(3931, 3849)、(2047, 3849)、(164, 3849)、(3931, 2047)、(2047, 2047)、(164, 2047)、(3931, 246)、(2047, 246) 和 (164, 246)。触摸屏产生的对应采样坐标为(3927, 3920)、(2054, 3936)、(193, 3943)、(3911, 2119)、(2054, 2127)、(195, 2164)、(3915, 331)、(2050, 354)和(189, 371)。显然，理想坐标与对应的采样坐标之间存在极大的误差。

接下来的三个实验分别采用经典三点算法、基于MMSE的五点算法和基于MMSE的九点算法：

- 经典三点算法选择三个参照点，即图2中的点1、点6和点8。相应的校准系数为：
  KX_1=+1.011238，KX_2=−0.003952，KX_3=−24.638760
  KY_1=+0.009894，KY_2=+1.005168，KY_3=−130.112700
  利用以上系数，校准后的结果列于表1中。
- 基于MMSE的五点算法选择五个参照点，即图2中的点1、点3、点5、点7和点9。相应的校准系数为：
  KX_1= +1.009899, KX_2 = −0.002260, KX_3 = −23.715720
  KY_1=+0.008494, KY_2= +1.006247, KY_3 = −121.821000
  利用以上系数，校准后的结果列于表2中。
- 基于MMSE的九点算法选择九个参照点，即图2中的点1、点2、点3、点4、点5、点6、点7、点8和点9。相应的校准系数为：
  KX_1= +1.011161, KX_2 = −0.001887, KX_3 = −25.777180
  KY_1=+0.009718, KY_2= +1.006107, KY_3 = −126.258100
  利用以上系数，校准后的结果列于表3中。

表1. 经典三点算法的结果

Point (X, Y)	1	2	3	4	5	6	7	8	9
理想坐标	(3931, 3849)	(2047, 3849)	(164, 3849)	(3931, 2047)	(2047, 2047)	(164, 2047)	(3931, 246)	(2047, 246)	(164, 246)
采样坐标	(3927, 3920)	(2054, 3936)	(193, 3943)	(3911, 2119)	(2054, 2127)	(195, 2164)	(3915, 331)	(2050, 354)	(189, 371)
校准坐标	(3931, 3849)	(2037, 3846)	(155, 3835)	(3922, 2039)	(2044, 2028)	(164, 2047)	(3933, 242)	(2047, 246)	(165, 244)
误差	(0, 0)	(−10, −3)	(−9, −14)	(−9, −8)	(−3, −19)	(0, 0)	(+2, −4)	(0, 0)	(+1, −2)
误差平方和	(276, 650)								

表2. 基于MMSE的五点算法的结果

Point (X, Y)	1	2	3	4	5	6	7	8	9
理想坐标	(3931, 3849)	(2047, 3849)	(164, 3849)	(3931, 2047)	(2047, 2047)	(164, 2047)	(3931, 246)	(2047, 246)	(164, 246)
采样坐标	(3927, 3920)	(2054, 3936)	(193, 3943)	(3911, 2119)	(2054, 2127)	(195, 2164)	(3915, 331)	(2050, 354)	(189, 371)
校准坐标	(3933, 3856)	(2042, 3856)	(162, 3847)	(3921, 2044)	(2046, 2036)	(168, 2057)	(3929, 245)	(2046, 252)	(166, 253)
误差	(2, 7)	(−5, +7)	(−2, −2)	(−10, −3)	(−1, −11)	(4, 10)	(−2, −1)	(−1, +6)	(+2, +7)
误差平方和	(159, 418)								

表3. 基于MMSE的九点算法的结果

Point (X, Y)	1	2	3	4	5	6	7	8	9
理想坐标	(3931, 3849)	(2047, 3849)	(164, 3849)	(3931, 2047)	(2047, 2047)	(164, 2047)	(3931, 246)	(2047, 246)	(164, 246)
采样坐标	(3927, 3920)	(2054, 3936)	(193, 3943)	(3911, 2119)	(2054, 2127)	(195, 2164)	(3915, 331)	(2050, 354)	(189, 371)
校准坐标	(3938, 3856)	(2044, 3854)	(162, 3842)	(3925, 2044)	(2047, 2034)	(167, 2053)	(3932, 245)	(2046, 250)	(165, 249)
误差	(7, 7)	(−3, +5)	(−2, −7)	(−6, −3)	(0, −13)	(3, 6)	(+1, −1)	(−1, +4)	(+1,+3)
误差平方和	(110, 363)								

结论

如前述实验结果所示，无论采用哪种校准算法，校准坐标均远远优于直接采样坐标。另外，通过比较三种实验，可得出以下结论

- 经典三点校准算法有助于将三个参照点校准到理想位置。另外，对于靠近三个参照点的点，其性能表现非常好。然而，对于不在参照点附近的点，经典三点校准算法的表现不甚理想。这种算法的误差平方和是测试所用三种算法中最大的。因此，对于触摸屏尺寸相对较大的应用，经典三种校准算法并非好的选择。
- 对于某些点(接近参照点的点)，基于MMSE的多点校准算法的表现不如经典三点校准算法。然而，就整个触摸屏来看，基于MMSE的多点校准算法的误差平方和小于经典三点算法，因为它利用了三个以上参照点的信息。因而，总体而言，其性能优于经典三点算法。
- 对于基于MMSE的多点校准算法，使用的参照点越多，性能越佳。

实验结果与数学推导相符合。

代码执行

以C语言编写的校准算法代码见“编码”部分。有三个参照点时，代码执行经典三点校准算法。有三个以上的参照点时，代码执行基于MMSE的多点校准算法。代码已通过ADuC7026测试(ADuC7026是ADI公司出品的一款MCU产品)。三个示例实验的结果均采用该代码计算得到。

编码

```
#define N    9                                  // number of reference points for calibration
algorithm
signed short int ReferencePoint[N][2];       // ideal position of reference points
signed short int SamplePoint[N][2];          // sampling position of reference points
double KX1, KX2, KX3, KY1, KY2, KY3;         // coefficients for calibration algorithm


void Do_Calibration(signed short int *Px, signed short int *Py)// do calibration for point
(Px, Py) using the calculated coefficients
{
      *Px=(signed short int)(KX1*(*Px)+KX2*(*Py)+KX3+0.5);
      *Py=(signed short int)(KY1*(*Px)+KY2*(*Py)+KY3+0.5);
}


int Get_Calibration_Coefficient()            // calculate the coefficients for calibration
algorithm: KX1, KX2, KX3, KY1, KY2, KY3
{
      int i;
      int Points=N;
      double a[3],b[3],c[3],d[3],k;
      if(Points<3)
      {
            return 0;
      }
      else
      {
            if(Points==3)
            {
                  for(i=0; i<Points; i++)
                  {
                        a[i]=(double)(SamplePoint[i][0]);
                        b[i]=(double)(SamplePoint[i][1]);
                        c[i]=(double)(ReferencePoint[i][0]);
                        d[i]=(double)(ReferencePoint[i][1]);
                  }
            }
            else if(Points>3)
            {
                  for(i=0; i<3; i++)
                  {
                        a[i]=0;
                        b[i]=0;
                        c[i]=0;
                        d[i]=0;
                  }
                  for(i=0; i<Points; i++)
```

```
                    {
                        a[2]=a[2]+(double)(SamplePoint[i][0]);
                        b[2]=b[2]+(double)(SamplePoint[i][1]);
                        c[2]=c[2]+(double)(ReferencePoint[i][0]);
                        d[2]=d[2]+(double)(ReferencePoint[i][1]);
                        a[0]=a[0]+(double)(SamplePoint[i][0])*(double)(SamplePoint[i][0]);
                        a[1]=a[1]+(double)(SamplePoint[i][0])*(double)(SamplePoint[i][1]);
                        b[0]=a[1];
                        b[1]=b[1]+(double)(SamplePoint[i][1])*(double)(SamplePoint[i][1]);
                        c[0]=c[0]+(double)(SamplePoint[i][0])*(double)(ReferencePoint[i][0]);
                        c[1]=c[1]+(double)(SamplePoint[i][1])*(double)(ReferencePoint[i][0]);
                        d[0]=d[0]+(double)(SamplePoint[i][0])*(double)(ReferencePoint[i][1]);
                        d[1]=d[1]+(double)(SamplePoint[i][1])*(double)(ReferencePoint[i][1]);
                    }
                    a[0]=a[0]/a[2];
                    a[1]=a[1]/b[2];
                    b[0]=b[0]/a[2];
                    b[1]=b[1]/b[2];
                    c[0]=c[0]/a[2];
                    c[1]=c[1]/b[2];
                    d[0]=d[0]/a[2];
                    d[1]=d[1]/b[2];
                    a[2]=a[2]/Points;
                    b[2]=b[2]/Points;
                    c[2]=c[2]/Points;
                    d[2]=d[2]/Points;
                }
                k=(a[0]-a[2])*(b[1]-b[2])-(a[1]-a[2])*(b[0]-b[2]);
                KX1=((c[0]-c[2])*(b[1]-b[2])-(c[1]-c[2])*(b[0]-b[2]))/k;
                KX2=((c[1]-c[2])*(a[0]-a[2])-(c[0]-c[2])*(a[1]-a[2]))/k;
                KX3=(b[0]*(a[2]*c[1]-a[1]*c[2])+b[1]*(a[0]*c[2]-a[2]*c[0])+b[2]*(a[1]*c[0]-
a[0]*c[1]))/k;
                KY1=((d[0]-d[2])*(b[1]-b[2])-(d[1]-d[2])*(b[0]-b[2]))/k;
                KY2=((d[1]-d[2])*(a[0]-a[2])-(d[0]-d[2])*(a[1]-a[2]))/k;
                KY3=(b[0]*(a[2]*d[1]-a[1]*d[2])+b[1]*(a[0]*d[2]-a[2]*d[0])+b[2]*(a[1]*d[0]-
a[0]*d[1]))/k;
                return Points;
        }
}
```

参考文献

Vidales, Carlos E. "How to Calibrate Touch Screens, Embedded Systems Design."Embedded.com, May 31, 2002. Embedded Systems Design.May 27, 2009.

注释

AN08183-0-7/09(0)