ANALOG DEVICES

AN-1021
APPLICATION NOTE

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

MMSE-Based Multipoint Calibration Algorithm for Touch Screen Applications

by Ning Jia

INTRODUCTION

Modern devices typically use LCDs with touch screen technology as user interfaces. Due to their simple construction and well-known operation, resistive-type touch screens are the most popular in cost-conscious designs. However, the mechanical misalignments and scaling factors of resistive-type touch screens affect the X and Y coordinates produced by the touch screens. Therefore, it is difficult to perfectly align the coordinates of the touch screen to the display (LCD or otherwise) behind it. When the final product that includes a touch screen comes out of the box, a calibration algorithm must first be done.

The classical calibration algorithm for a touch screen is a three-point calibration algorithm that uses three reference points. The classical three-point calibration algorithm is both efficient and effective, but its performance is low when the touch screen is comparatively large. This application note proposes the minimum mean square error (MMSE)-based multipoint calibration algorithm, which uses more than three reference points, for resistive-type touch screens. Both mathematical deduction and experimentation prove that this algorithm is more accurate than the classical three-point calibration algorithm.

Rev. 0

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

TABLE OF CONTENTS

MATHEMATICS

Figure 1 shows a circle in red where its ideal center is O (origin) and its ideal radius is R. Consider that the red circle represents an image shown by the LCD under the touch screen. The blue ellipse represents an exaggerated set of points produced by the touch screen when the user traces the red circle shown by the LCD. Therefore, the reconstructed image appears rotated, translated, and scaled by a different factor in each axis because of mechanical misalignments and scaling factors of resistive-type touch screens. The challenge for the calibration algorithm is to translate the set of coordinates reported by the touch screen into a set of coordinates that accurately represent the displayed image.

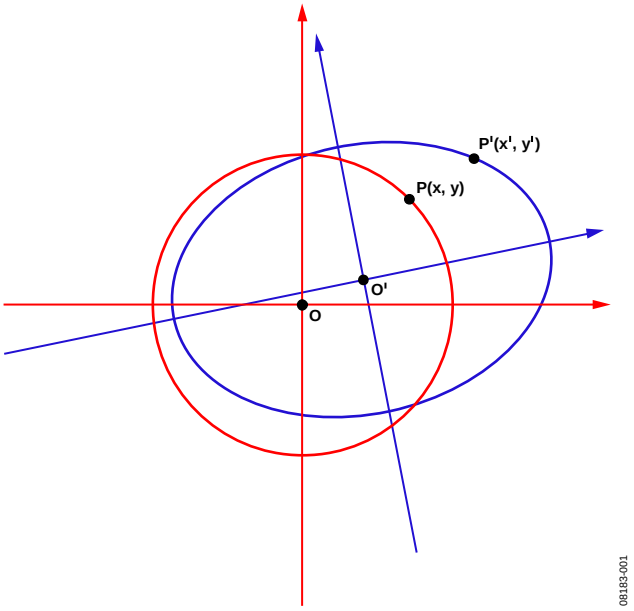

Figure 1. Errors in Touch Screen

If there is a point where its ideal coordinates are P(x, y), and the coordinates produced by touch screen are P'(x', y'). (Point P is on the red circle, and that Point P' is the corresponding point on the blue ellipse.) Suppose that P'(x', y') could return to its ideal coordinates, P(x, y), after it rotates by θ, is scaled by K_X and K_Y in each axis, and is translated by T_X and T_Y in each axis. The purpose of the calibration algorithm is to calculate the coefficients of θ, K_X, K_Y, T_X, and T_Y. The coefficients can then be used to calibrate the coordinates directly produced by the touch screen.

To make the analysis easy to understand, assume that P'(x', y') are the coordinates directly produced by the touch screen, and its equivalent representation in polar coordinates is P'(Rcos θ_0, Rsin θ_0). In addition, assume that the corresponding ideal coordinates of P'(x', y') are P(x, y). Because of the relationship between P'(x', y') and P(x, y), it is reasonable to get the following:

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

According to the trigonometric function,

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

and

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

The following equations can then be obtained:

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

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

And then,

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

where:
θ, K_X, K_Y, T_X, and T_Y are all constants.

Let:

$$\cos\theta K_X = KX_1$$

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

$$T_X = KX_3$$

$$\sin\theta K_Y = KY_1$$

$$\cos\theta K_Y = KY_2$$

$$T_Y = KY_3$$

Then the previous equations can be rewritten as

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

These equations can be used to calibrate the P'(x', y') back to P(x, y). There are three unknown coefficients in each equation for either the X axis or the Y axis.

CLASSICAL THREE-POINT CALIBRATION ALGORITHM

The previous analysis produces the calibration equation for either the X axis or the Y axis. Each equation has three unknown coefficients for either the X axis or the Y axis. Therefore, if the information for the three irrelative reference points was available, one can construct a linear system of equations and get the solution for the unknown coefficients by solving the equations.

Assuming that the ideal coordinates of the three reference points are (x_0, y_0), (x_1, y_1), and (x_2, y_2), and their corresponding sampled coordinates are (x'_0, y'_0), (x'_1, y'_1), and (x'_2, y'_2), then the equations for both the X axis and the Y axis are

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

and

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

The equations can then be rewritten as matrix type:

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

and

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

The calibration coefficients, KX_1, KX_2, KX_3, KY_1, KY_2, and KY_3, can then be calculated by solving the equations.

The results calculated by using the elimination method are as follows:

Let $k = (x'_0 - x'_2)(y'_1 - y'_2) - (x'_1 - x'_2)(y'_0 - y'_2)$, then

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

$$KX_2 = \frac{(x_1 - x_2)(x'_0 - x'_2) - (x_0 - x_2)(x'_1 - x'_2)}{k}$$

$$KX_3 = \frac{y'_0(x'_2x_1 - x'_1x_2) + y'_1(x'_0x_2 - x'_2x_0) + y'_2(x'_1x_0 - x'_0x_1)}{k}$$

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

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

$$KY_3 = \frac{y'_0(x'_2y_1 - x'_1y_2) + y'_1(x'_0y_2 - x'_2y_0) + y'_2(x'_1y_0 - x'_0y_1)}{k}$$

MMSE-BASED MULTIPOINT CALIBRATION ALGORITHM

Using the coefficients calculated by the classical three-point calibration algorithm, the three reference points can be calibrated to the exact ideal position. However, for other points that are not close to the reference points, the calibration performance is not good enough, especially when the size of the touch screen is comparatively large. The experimental result in the Examples section also proves this problem. Therefore, consider using more than three reference points to get the best calibration coefficients.

Assuming that there are N + 1 reference points (N + 1 > 3), whose ideal coordinates are (x_0, y_0), (x_1, y_1), …, (x_N, y_N), and whose corresponding sampled coordinates are (x'_0, y'_0), (x'_1, y'_1), …, (x'_N, y'_N), then the equations are as follows:

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

and

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

Note that the number of equations in each equation system (N + 1) is larger than the number of unknown coefficients (3).

The purpose is to then calculate the best calibration coefficients to fit all the (N + 1) reference points. The method to get the best coefficients is to follow the MMSE rule. This is why the algorithm is called MMSE-based multipoint calibration algorithm.

Taking the X axis as an example, define an aim function

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

where FX is the sum of the square error for the reference points.

The best coefficients of KX_1, KX_2, and KX_3 are the coefficients that can minimize the aim function, FX. Therefore, the following equations can be used:

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

that is

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

These equations can be simplified as

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

Let

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

The equations can then be rewritten in matrix type as R • KX = BX, and the best coefficients, KX = R^{-1} • BX, can then be calculated by solving the previous equations system.

Similar to the X axis, let

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

then the best coefficients for the Y axis, KY = R^{-1} • BY, can be calculated by solving the equation system, R • KY = BY.

The results calculated by using the elimination method are as follows:

Let

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

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

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

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

where $\sum \bullet$ represents $\sum_{i=0}^{N} \bullet$

The equation systems R • KX = BX and R • KY = BY can then be rewritten as

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

and

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

It is easy to see that the equation systems have the same format as the former equation systems for the classical three-point algorithm:

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

and

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

The same formula can then be used to calculate the results of coefficients.

Let

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

Then

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

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

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

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

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

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

ANALYSIS FOR MMSE-BASED MULTIPOINT CALIBRATION ALGORITHM

By the a_0, a_1, a_2; b_0, b_1, b_2; c_0, c_1, c_2; and d_0, d_1, d_2 calculation, it is clear that a_0, a_1, a_2 can be treated as the weighted averages of x'_i by three different methods, and b_0, b_1, b_2 can be treated as the weighted averages of y'_i by three different methods. Similarly, c_0, c_1, c_2 can be treated as the weighted averages of x_i by three different methods, and d_0, d_1, d_2 can be treated as the weighted averages of y_i by three different methods.

Because the final equation systems have the same format as the previous equation systems for a classical three-point algorithm, the algorithm-based MMSE rule can be consider as another three-point algorithm. However, the difference in the MMSE-based algorithm is that, instead of direct information of any reference point being used, the weighted average information of the reference points is used. That is, first calculate three weighted average points (a_0, b_0), (a_1, b_1), (a_2, b_2) of N + 1 direct sampled points (x'_0, y'_0), (x'_1, y'_1), …, (x'_N, y'_N). The corresponding ideal coordinates of the three weighted average points are considered (c_0, d_0), (c_1, d_1), and (c_2, d_2). The MMSE-based algorithm is then equivalent to the classical three-point algorithm for these three weighted average points.

STEPS FOR MMSE-BASED MULTIPOINT CALIBRATION ALGORITHM

Complete the following steps for the MMSE-based multipoint calibration algorithm:

1. Select N +1 (N + 1 > 3) reference points (x_0, y_0), (x_1, y_1), … , (x_N, y_N).
2. Get the sampled coordinates of the reference points that were produced by the touch screen, (x'_0, y'_0), (x'_1, y'_1), …, (x'_N, y'_N).
3. Use the formulas given in this application note to calculate the calibration coefficients, KX and KY. These include the following:

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

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

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

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

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

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

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

4. In normal operation, use the calibration coefficients (KX, KY) and the following equations to calculate the calibration for point P'(x', y').

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

EXAMPLES

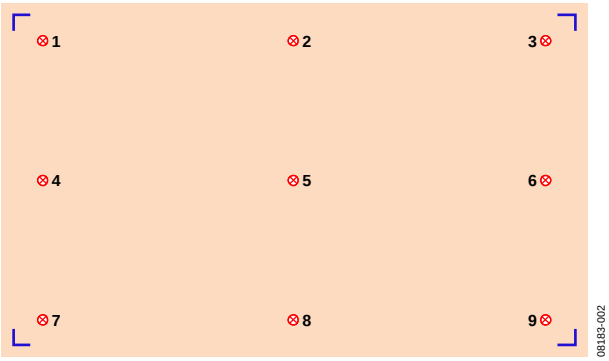

Figure 2. Selection of Reference Points

As shown in Figure 2, nine points were selected on the touch-screen. Their ideal coordinates are (3931, 3849), (2047, 3849), (164, 3849), (3931, 2047), (2047, 2047), (164, 2047), (3931, 246), (2047, 246), and (164, 246). The corresponding sampled coordinates produced by the touch screen are (3927, 3920), (2054, 3936), (193, 3943), (3911, 2119), (2054, 2127), (195, 2164), (3915, 331), (2050, 354), and (189, 371). Obviously, there are very large errors between the ideal coordinates and the corresponding sampled coordinates.

Three experiments follow that used the classical three-point algorithm, MMSE-based five-point algorithm, and MMSE-based nine-point algorithm:

- The classical three-point algorithm selects the three references as Point 1, Point 6, and Point 8 in Figure 2. The calibration coefficients are:
  $KX_1 = +1.011238$, $KX_2 = -0.003952$, and $KX_3 = -24.638760$
  $KY_1 = +0.009894$, $KY_2 = +1.005168$, and $KY_3 = -130.112700$
  By using the previous coefficients, the results after calibration are listed in Table 1.
- The MMSE-based five-point algorithm selects the five references as Point 1, Point 3, Point 5, Point 7, and Point 9 in Figure 2. The calibration coefficients are:
  $KX_1 = +1.009899$, $KX_2 = -0.002260$, $KX_3 = -23.715720$
  $KY_1 = +0.008494$, $KY_2 = +1.006247$, $KY_3 = -121.821000$
  By using the previous coefficients, the results after calibration are listed in Table 2.
- The MMSE-based nine-point algorithm, selects the nine references as Point 1, Point 2, Point 3, Point 4, Point 5, Point 6, Point 7, Point 8, and Point 9 in Figure 2. The calibration coefficients are:
  $KX_1 = +1.011161$, $KX_2 = -0.001887$, $KX_3 = -25.777180$
  $KY_1 = +0.009718$, $KY_2 = +1.006107$, $KY_3 = -126.258100$
  By using the previous coefficients, the results after calibration are listed in Table 3.

Table 1. Results of the Classical Three-Point Algorithm

Point (X, Y)	1	2	3	4	5	6	7	8	9
Ideal Coordinates	(3931, 3849)	(2047, 3849)	(164, 3849)	(3931, 2047)	(2047, 2047)	(164, 2047)	(3931, 246)	(2047, 246)	(164, 246)
Sampled Coordinates	(3927, 3920)	(2054, 3936)	(193, 3943)	(3911, 2119)	(2054, 2127)	(195, 2164)	(3915, 331)	(2050, 354)	(189, 371)
Calibrated Coordinates	(3931, 3849)	(2037, 3846)	(155, 3835)	(3922, 2039)	(2044, 2028)	(164, 2047)	(3933, 242)	(2047, 246)	(165, 244)
Error	(0, 0)	(−10, −3)	(−9, −14)	(−9, −8)	(−3, −19)	(0, 0)	(+2, −4)	(0, 0)	(+1, −2)
Sum of Square Error	(276, 650)								

Table 2. Results of the MMSE-Based Five-Point Algorithm

Point (X, Y)	1	2	3	4	5	6	7	8	9
Ideal Coordinates	(3931, 3849)	(2047, 3849)	(164, 3849)	(3931, 2047)	(2047, 2047)	(164, 2047)	(3931, 246)	(2047, 246)	(164, 246)
Sampled Coordinates	(3927, 3920)	(2054, 3936)	(193, 3943)	(3911, 2119)	(2054, 2127)	(195, 2164)	(3915, 331)	(2050, 354)	(189, 371)
Calibrated Coordinates	(3933, 3856)	(2042, 3856)	(162, 3847)	(3921, 2044)	(2046, 2036)	(168, 2057)	(3929, 245)	(2046, 252)	(166, 253)
Error	(2, 7)	(−5, +7)	(−2, −2)	(−10, −3)	(−1, −11)	(4, 10)	(−2, −1)	(−1, +6)	(+2, +7)
Sum of Square Error	(159, 418)								

Table 3. Results of the MMSE-Based Nine-Point Algorithm

Point (X, Y)	1	2	3	4	5	6	7	8	9
Ideal Coordinates	(3931, 3849)	(2047, 3849)	(164, 3849)	(3931, 2047)	(2047, 2047)	(164, 2047)	(3931, 246)	(2047, 246)	(164, 246)
Sampled Coordinates	(3927, 3920)	(2054, 3936)	(193, 3943)	(3911, 2119)	(2054, 2127)	(195, 2164)	(3915, 331)	(2050, 354)	(189, 371)
Calibrated Coordinates	(3938, 3856)	(2044, 3854)	(162, 3842)	(3925, 2044)	(2047, 2034)	(167, 2053)	(3932, 245)	(2046, 250)	(165, 249)
Error	(7, 7)	(−3, +5)	(−2, −7)	(−6, −3)	(0, −13)	(3, 6)	(+1, −1)	(−1, +4)	(+1,+3)
Sum of Square Error	(110, 363)								

CONCLUSION

As shown in the results of the previous experiments, no matter which calibration algorithm was used, the calibrated coordinates are much better than the direct sampled coordinates. In addition, by comparing these three experiments, it was concluded that

- The classical three-point calibration algorithm helps calibrate the three reference points to the ideal position. In addition, its performance is very good for the points close to the three reference points. However, for the points not close to the three reference points, the performance of the classical three-point calibration is not good enough. The sum of the square error for this algorithm was the largest of the three tested algorithms. Therefore, the classical three-point calibration algorithm is not a good choice for applications where touch screen sizes are comparatively large.
- For certain points (the points close to reference points), the MMSE-based multipoint calibration algorithm has worse performance than the classical three-point calibration algorithm. However, when looking at the whole touch screen, the MMSE-based multipoint calibration algorithm has a smaller sum of square error than the classical three-point algorithm because it uses the information of more than three reference points. Therefore, its performance is better than the classical three-point algorithm from a holistic view.
- For the MMSE-based multipoint calibration algorithm, the more reference points used, the better the performance is.

The results of these experiments are consentaneous with the mathematics.

CODE IMPLEMENTATION

The code implementation of the calibration algorithm in C language is shown in the Coding section. When there are three reference points, the code implements the classical three-point calibration algorithm. When there are more than three reference points, the code implements the MMSE-based multi-point calibration algorithm. The code was tested with the ADuC7026 (the ADuC7026 is an MCU product from Analog Devices, Inc.). The results of the three example experiments were calculated by this code.

CODING

```
#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;
     }
}
```

REFERENCES

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

NOTES

AN08183-0-7/09(0)