タッチ・スクリーン・アプリケーションに対する MMSE に基づくマルチポイント・キャリブレーション・アルゴリズム

著者: Ning Jia

概要

一般に現代の機器では、ユーザ・インターフェースとしてタッチ・スクリーン技術を使ったLCDを採用しています。抵抗型タッチ・スクリーンの構造はシンプルで動作が良く知られているため、低価格デザインでは最も広く採用されていますが、抵抗型タッチ・スクリーンの機械的なミスアライメントとスケール・ファクタにより、タッチ・スクリーンから発生されるX座標とY座標が影響を受けます。このため、タッチ・スクリーンの座標を背後のディスプレイ (LCDなど)に完全に位置合わせすることは困難です。タッチ・スクリーンを内蔵する最終製品を出荷する際には、キャリブレーション・アルゴリズムを実行することが必要です。

タッチ・スクリーンに対する従来型キャリブレーション・アルゴリズムは、3つのリファレンス・ポイントを使う3ポイント・キャリブレーション・アルゴリズムでした。従来型の3ポイント・キャリブレーション・アルゴリズムは能率的かつ効果的ですが、タッチ・スクリーンが比較的大きい場合に性能が低下します。このアプリケーション・ノートでは、最小平均2乗誤差(MMSE)に基づくマルチポイント・キャリブレーション・アルゴリズムを提案します。このアルゴリズムでは、抵抗型タッチ・スクリーンに対して3ポイントより多いリファレンス・ポイントを使います。数学的予測と実験では、このアルゴリズムの方が従来型 3ポイント・キャリブレーション・アルゴリズムより正確であることが示されています。

Rev. 0

アナログ・デバイセズ株式会社

本　　社／〒105-6891　東京都港区海岸 1-16-1　ニューピア竹芝サウスタワービル
電話 03（5402）8200
大阪営業所／〒532-0003　大阪府大阪市淀川区宮原 3-5-36　新大阪トラストタワー
電話 06（6350）6868

目次

数学的基礎

図1 に、赤で示す円の理論中心はO (原点)で理論半径はRです。この赤の円はタッチ・スクリーンの下のLCDにより表示されるイメージを表しているものと見なします。青の楕円は、ユーザがLCDにより表示される赤の円を追跡するときに、タッチ・スクリーンにより発生される、誇張されたポイントのセットを表します。抵抗型タッチ・スクリーンには機械的ミスアライメントとスケール・ファクタの不一致があるため、再生されたイメージは各軸で異なるファクタを使って回転、移動、スケールされて現れます。キャリブレーション・アルゴリズムでの困難は、タッチ・スクリーンから出力される座標を、表示されたイメージを正確に表す座標へ変換する点にあります。

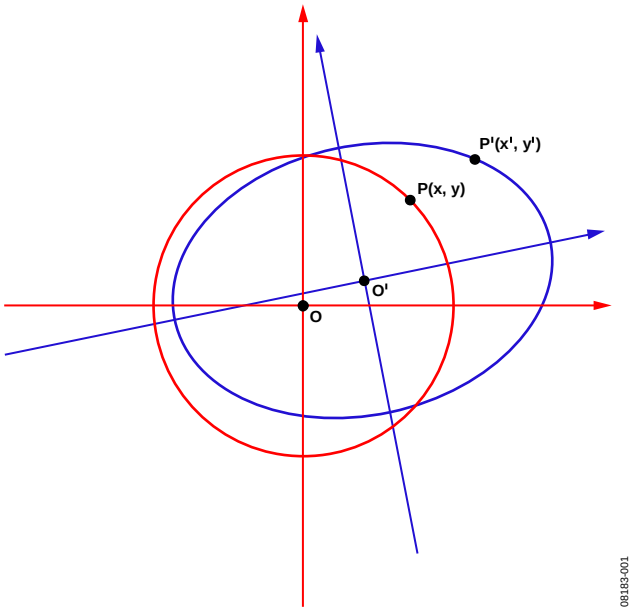

図 1.タッチ・スクリーンでの誤差

理論座標が P(x, y)であるポイントが存在し、かつタッチ・スクリーンで発生される座標が P'(x', y')であるとします(ポイント Pは赤い円上にあり、ポイント P' は青い楕円上にある対応するポイントです)。P'(x', y')はθだけ回転され、各軸でK_XとK_Yによりスケールされ、各軸でT_XとT_Yにより移動された後に理論座標P(x, y)へ戻すことができるものとします。キャリブレーション・アルゴリズムの目的は、係数 θ、K_X、K_Y、T_X、T_Yを計算することです。その後、これらの係数を使ってタッチ・スクリーンから直接発生された座標をキャリブレーションすることができます。

解析を容易かつ分かり安くするため、P'(x', y')をタッチ・スクリーンから直接出力される座標とし、その等価な極座標表現をP'(Rcos θ_0、Rsin θ_0)とします。さらに、P'(x', y') の対応する理論座標を P(x, y)とします。P'(x', y')とP(x, y)との間の関係を使うと、次のように表すことができます。

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

三角関数の公式から、

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

かつ

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

これらから、次式が得られます。

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

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

さらに、

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

ここで、 θ、K_X、K_Y、T_X、T_Y はすべて定数。

次のように表すと、

$\cos\theta K_X = KX_1$

$-\sin\theta K_X = KX_2$

$T_X = KX_3$

$\sin\theta K_Y = KY_1$

$\cos\theta K_Y = KY_2$

$T_Y = KY_3$

前式は次のように表されます。

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

これらの式は、P'(x', y')を P(x, y)へ戻すときのキャリブレーションに使うことができます。X 軸または Y 軸の各式で、未知係数が 3 個あります。

従来型 3 ポイント・キャリブレーション・アルゴリズム

前の解析では、X 軸または Y 軸のキャリブレーション式を求めました。各式には、 X 軸または Y 軸に対して 3 個の未知係数があります。したがって、独立な 3 個のリファレンス・ポイントについて、連立方程式を立てて、未知係数を求めることができます。

3 つのリファレンス・ポイントの理論座標を (x_0, y_0)、(x_1, y_1)、(x_2, y_2)とし、サンプルされた対応する座標を (x'_0, y'_0)、(x'_1, y'_1)、(x'_2, y'_2)とすると、X 軸と Y 軸に対する式は、次のようになります。

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

かつ

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

これらの式を行列で表すと、

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

かつ

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

この式を解くと、キャリブレーション係数 KX_1、KX_2、KX_3、KY_1、KY_2、KY_3を求めることができます。

消去法による計算結果は次のようになります。

$k = (x'_0 - x'_2)(y'_1 - y'_2) - (x'_1 - x'_2)(y'_0 - y'_2)$ とすると、

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

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

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

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

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

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

MMSEに基づくマルチポイント・キャリブレーション・アルゴリズム

従来型 3 ポイント・キャリブレーション・アルゴリズムで計算した係数を使うと、3 個のリファレンス・ポイントを正確な理論位置にキャリブレーションすることができますが、リファレンス・ポイントに近くない他のポイントに対して、特にタッチ・スクリーンのサイズが比較的大きい場合に、キャリブレーション性能は不十分です。例 のセクションの実験結果も、この問題を示しています。このため、3 個より多いリファレンス・ポイントを使って最適なキャリブレーション係数を求めます。

リファレンス・ポイント (N + 1 > 3)を N + 1 個とし、その理論座標を (x_0, y_0)、(x_1, y_1)、…、(x_N, y_N)して、サンプルされた対応する座標を (x'_0, y'_0)、(x'_1, y'_1)、…、(x'_N, y'_N)とすると、式は次のようになります。

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

かつ

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

連立方程式の式数 (N + 1)は、未知係数の数 (3)より大きいことに注意してください。

目的は、(N + 1) 個の全リファレンス・ポイントに合う最適なキャリブレーション係数を計算することです。最適係数を得る方法は、MMSE の規則に従うことです。これは、このアルゴリズムが MMSE に基づくマルチポイント・キャリブレーション・アルゴリズムと呼ばれる理由です。

X 軸を例にして、目的関数を次のように定義します。

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

ここで、FX はリファレンス・ポイントの 2 乗誤差の和です。

KX_1、KX_2、KX_3 の最適係数とは、目的関数 FX を最小にできる係数のことです。したがって、次の式を使うことができます。

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

すなわち、

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

これらの式は次のように簡素化できます。

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

次のように表すと、

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

各式は行列で R ● KX = BX と表すことができ、最適係数 $KX = R^{-1}$ ● BX は前の連立方程式を解くことにより求めることができます。

X 軸と同様に次のように表すと、

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

Y 軸に対する最適係数 $KY = R^{-1}$ ● BY は、連立方程式 R ● KY = BY を解くことにより求めることができます。

消去法による計算結果は次のようになります。

次のように表すと、

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

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

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

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

ここで、$\sum\bullet$ は $\sum_{i=0}^{N}\bullet$ を表します。

連立方程式 R ● KX = BX と R ● KY = BY は、次のように表すことができます。

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

かつ

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

この連立方程式は、従来型 3 ポイント・アルゴリズムに対する前の連立方程式と同じ形式であることが容易に分かります。

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

かつ

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

したがって、同じ式を使って係数の結果を計算することができます。

次のように表すと、

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

次式が得られます。

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

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

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

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

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

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

MMSEに基づくマルチポイント・キャリブレーション・アルゴリズムの解析

a_0、a_1、a_2; b_0、b_1、b_2; c_0、c_1、c_2; d_0、d_1、d_2 の計算から、a_0、a_1、a_2 を 3 つの方法で x'_i の重み付け平均として扱うことができ、b_0、b_1、b_2 を 3 つの方法で y'_iの重み付け平均として扱うことができることが明らかです。同様に、c_0、c_1、c_2を 3 つの方法で x_i の重み付け平均として、d_0、d_1、d_2を 3 つの方法で y_iの重み付け平均として、それぞれ扱うことができます。

最後の連立方程式は、従来型 3 ポイント・アルゴリズムに対する前の連立方程式と同じ形式であるため、MMSE 規則に基づくアルゴリズムは、もう 1 つの 3 ポイント・アルゴリズムと見なすことができます。ただし、MMSE に基づくアルゴリズムでの違いは、使用しているリファレンス・ポイントからの直接の情報ではなく、リファレンス・ポイントの重み付け平均情報を使用していることです。すなわち、最初に、N + 1 個の直接サンプルされるポイント (x'_0, y'_0)、(x'_1, y'_1)、…、$(x'_N、y'_N)$から 3 個の重み付け平均ポイント (a_0, b_0)、(a_1, b_1)、(a_2, b_2)を計算します。3 個の重み付け平均ポイントの対応する理論座標は、 (c_0, d_0)、(c_1, d_1)、(c_2, d_2)と見なされます。したがって、MMSE に基づくアルゴリズムは、これらの 3 個の重み付け平均ポイントに対する従来型 3 ポイント・アルゴリズムと等価になることが分かります。

MMSEに基づくマルチポイント・キャリブレーション・アルゴリズムの手順

MMSE に基づくマルチポイント・キャリブレーション・アルゴリズムについては、次のステップを実行します。

1. N +1 (N + 1 > 3) 個のリファレンス・ポイント (x_0, y_0)、(x_1, y_1)、…、$(x_N、y_N)$を選択します。
2. タッチ・スクリーンで発生されるリファレンス・ポイントのサンプルされた座標(x'_0, y'_0)、(x'_1, y'_1)、…、$(x'_N、y'_N)$を取得します。
3. このアプリケーション・ノートで示した式を使って、キャリブレーション係数 KX と KY を計算します。これらは次のようになります。

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

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

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

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

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

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

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

4. 通常動作では、キャリブレーション係数 (KX、KY)と次の式を使って、ポイント P'(x', y')に対するキャリブレーションを計算します。

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

例

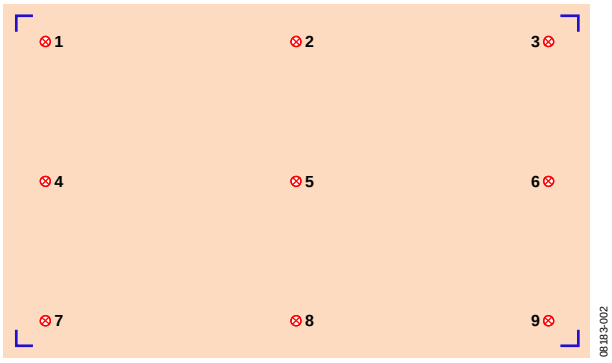

図 2. リファレンス・ポイントの選択

図 2 に示すように、タッチ・スクリーン上で 9 個のポイントを選択します。これらの理論座標は、(3931, 3849)、(2047, 3849)、(164, 3849)、(3931, 2047)、(2047, 2047)、(164, 2047)、(3931, 246)、(2047, 246)、(164, 246)です。タッチ・スクリーンから出力されるサンプルされた対応する座標は、(3927, 3920)、(2054, 3936)、(193, 3943)、(3911, 2119)、(2054, 2127)、(195, 2164)、(3915, 331)、(2050, 354)、(189, 371)です。明らかに、理論座標と対応するサンプルされた座標との間には非常に大きな誤差があります。

従来型 3 ポイント・アルゴリズム、MMSE に基づく 5 ポイント・アルゴリズム、MMSE に基づく 9 ポイント・アルゴリズムを使った 3 つの実験を次に示します。

- 従来型 3 ポイント・アルゴリズムでは、3 個のリファレンスを選択しました。図 2 のポイント 1、ポイント 6、ポイント 8 です。キャリブレーション係数は、
  $KX_1 = +1.011238$、$KX_2 = -0.003952$、$KX_3 = -24.638760$
  $KY_1 = +0.009894$、$KY_2 = +1.005168$、$KY_3 = -130.112700$
  この係数を使ったキャリブレーション後の結果を 表 1 に示します。
- MMSEに基づく 5 ポイント・アルゴリズムでは 5 個のリファレンスを選択しました。図 2 のポイント 1、ポイント 3、ポイント 5、ポイント 7、ポイント 9 です。キャリブレーション係数は、
  $KX_1 = +1.009899$、$KX_2 = -0.002260$、$KX_3 = -23.715720$
  $KY_1 = +0.008494$、$KY_2 = +1.006247$、$KY_3 = -121.821000$
  この係数を使ったキャリブレーション後の結果を 表 2 に示します。
- MMSEに基づく 9 ポイント・アルゴリズムでは 9 個のリファレンスを選択しました。図 2 のポイント 1、ポイント 2、ポイント 3、ポイント 4、ポイント 5、ポイント 6、ポイント 7、ポイント 8、ポイント 9 です。キャリブレーション係数は、
  $KX_1 = +1.011161$, $KX_2 = -0.001887$、$KX_3 = -25.777180$
  $KY_1 = +0.009718$、$KY_2 = +1.006107$、$KY_3 = -126.258100$
  この係数を使ったキャリブレーション後の結果を 表 3 に示します。

表 1.従来型 3 ポイント・アルゴリズムの結果

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)								

表 2.MMSE に基づく 5 ポイント・アルゴリズムの結果

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)								

表 3.MMSE に基づく 9 ポイント・アルゴリズムの結果

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)								

結論

実験結果に示すように、いずれのキャリブレーション・アルゴリズムを使っても、キャリブレーションした座標は、直接 サンプルした座標より優れています。さらに、これら 3 つの実験を比較すると、次の結論が得られます。

- 従来型 3 ポイント・キャリブレーション・アルゴリズムは、3 個のリファレンス・ポイントを理論位置へキャリブレーションするのに役立ちます。さらに、3 個のリファレンス・ポイントに近いポイントに対しては性能が非常に優れています。ただし、3 個のリファレンス・ポイントに近くないポイントに対して、従来型 3 ポイント・キャリブレーションの性能は不十分です。このアルゴリズム の 2 乗誤差の和は、テストした 3 つのアルゴリズムで最大でした。したがって、従来型 3 ポイント・キャリブレーション・アルゴリズムは、タッチ・スクリーン・サイズが比較的大きいアプリケーションには適していません。
- あるポイント (リファレンス・ポイントに近いポイント)に対して、MMSE に基づくマルチポイント・キャリブレーション・アルゴリズムの性能は、従来型 3 ポイント・キャリブレーション・アルゴリズムより劣ります。ただし、タッチ・スクリーン全体を見たとき、MMSE に基づくマルチポイント・キャリブレーション・アルゴリズムの 2 乗誤差の和は、従来型 3 ポイント・アルゴリズムより小さくなります。これは、3 個より多いリファレンス・ポイントの情報を使っているためです。このため、性能は、全体的な面で従来型 3 ポイント・アルゴリズムより優れています。
- MMSE に基づくマルチポイント・キャリブレーション・アルゴリズムの場合、使用するリファレンス・ポイント が多いほど、性能が向上します。

これらの実験結果は、数学的にも敵っています。

コードの実装

C言語によるキャリブレーション・アルゴリズムのコードを コーディング のセクションに示します。3 個のリファレンス・ポイントの場合、このコードでは従来型 3 ポイント・キャリブレーション・アルゴリズムを採用しています。.リファレンス・ポイントが 3 個より多い場合は、このコードではMMSEに基づくマルチポイント・キャリブレーション・アルゴリズムを採用しています。このコードは、ADuC7026 を使ってテストされました (ADuC7026 はアナログ・デバイセズのMCU製品)。 3 つの実験例の結果は、このコードを使って計算しました。

コーディング

```
#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;
    }
}
```

参考資料

Vidales, Carlos E. “How to Calibrate Touch screens, Embedded Systems Design.”*Embedded.com*, May 31, 2002. Embedded Systems Design.May 27, 2009.