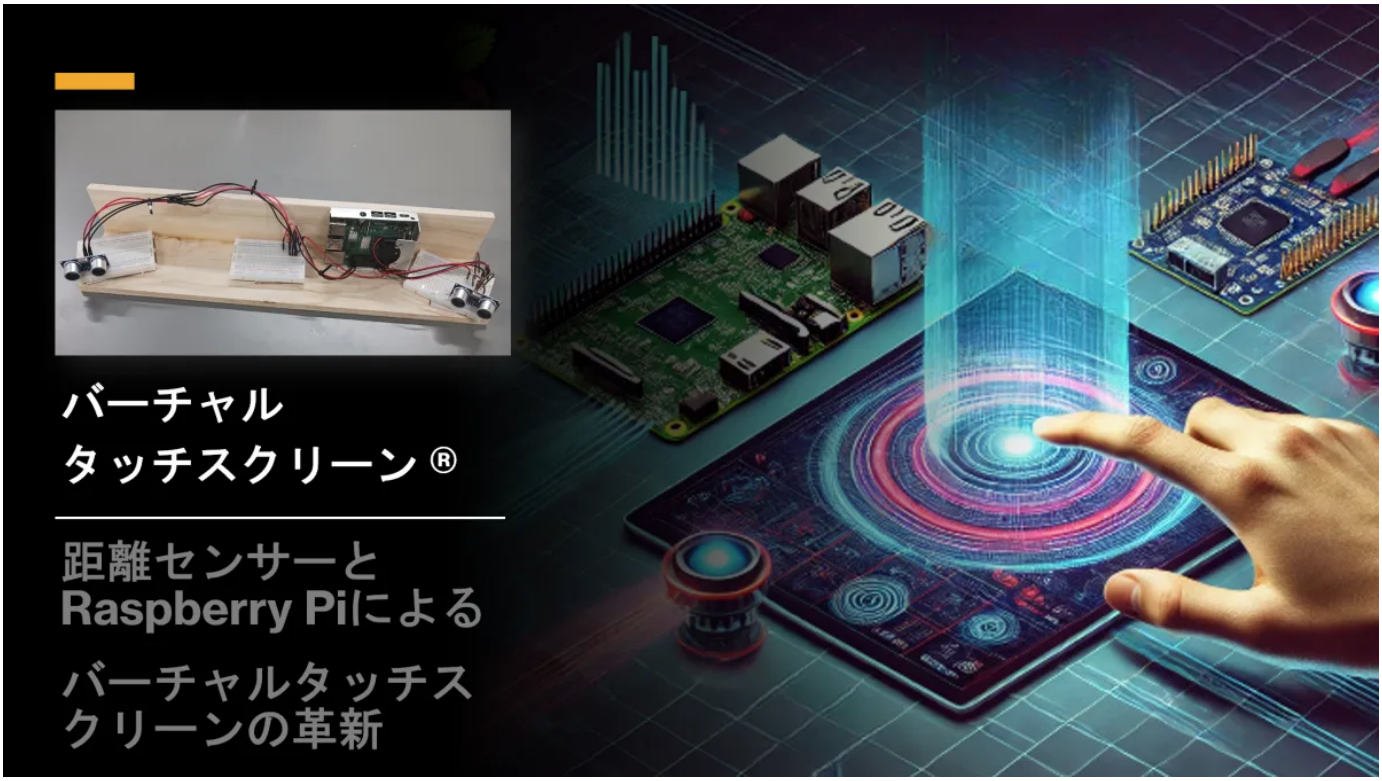
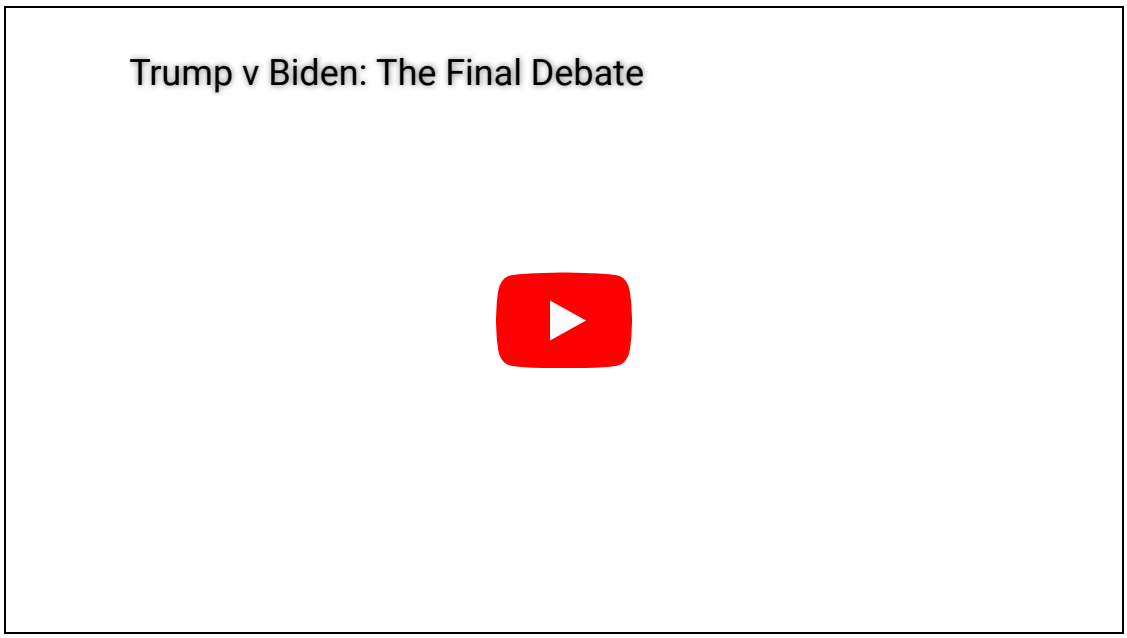
バーチャルタッチスクリーン 距離センサーによるバーチャルタッチ技術

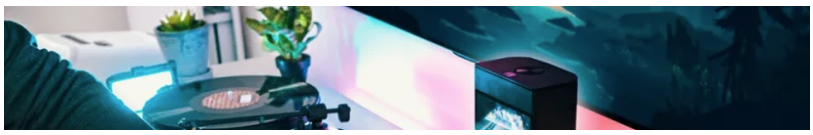
Mamanchuk Mykola(/designspark/user/Mamanchuk%20Mykola) **3 Jul 2024**

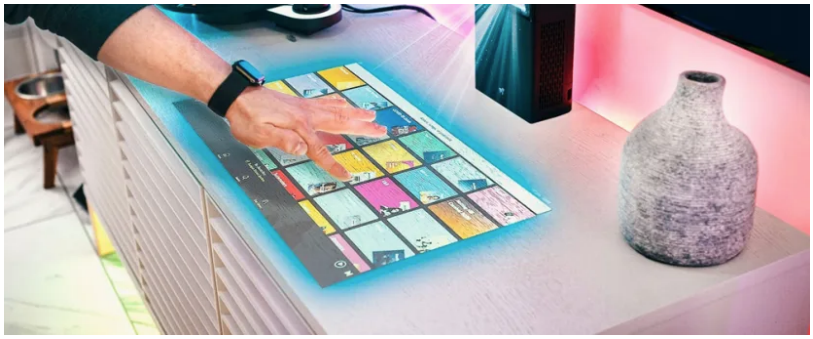


ショーケース概要:



このプロジェクトでは、**低コストの距離センサー**を使用して、物理的な接触なしにインタラクティブなタッチスクリーンをシミュレートする技術を開発しました。下記の記事では、我々のシステムがどのようにして従来のインタラクションを再定義するかを探ります。

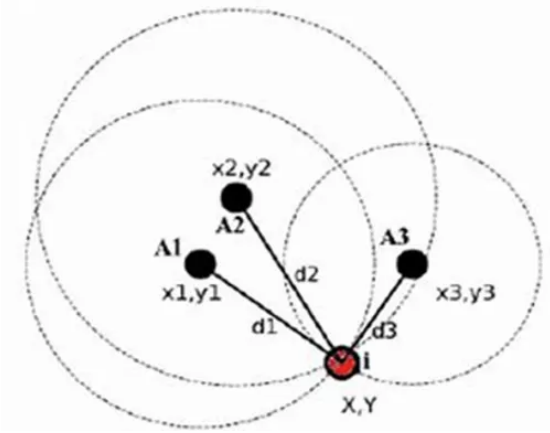




従来のタッチスクリーン技術は高価なレーザー技術を使用していますが、我々のシステムは安価で容易に入手可能なセンサーを使用して同様の結果を達成します。

理論的概要:

三辺測量法（トリラテレーション）は、3つの既知の位置からの距離を使用して物体の位置を計算する手法です。このプロジェクトでは、超音波センサーを使用して手の位置を測定し、タッチスクリーンの操作に応用しています。



上の図は、三辺測量法の概念を示しています。3つの既知の位置（A1、A2、A3）からの距離（d1、d2、d3）を使用して、未知の位置（i）を計算します。

$$x = \frac{D_1^2 - D_2^2}{2d} + \frac{d}{2}$$

$$y = \sqrt{D_1^2 - x^2}$$

このような上記の数式は、三辺測量法の計算式を示しています。x座標とy座標は、3つの距離測定値（D1、D2、D3）を使用して計算されます。

三辺測量システムコードサンプルを以下に示します。

```
import numpy

P1 = [0,0]

  #upper position


P2 = [-10,10]

  #anyway position


P3 = [10,10]

  #anyway position



def trilateration(P1, P2, P3, r1, r2, r3):

  #r1,r2,r3 is the distance of staff and sensor



  p1 = numpy.array([0, 0])
  p2 = numpy.array([P2[0] - P1[0], P2[1] - P1[1]])
  p3 = numpy.array([P3[0] - P1[0], P3[1] - P1[1]])
  v1 = p2 - p1
  v2 = p3 - p1

  Xn = (v1)/numpy.linalg.norm(v1)

  tmp = numpy.cross(v1, v2)

  Zn = (tmp)/numpy.linalg.norm(tmp)

  Yn = numpy.cross(Xn, Zn)

  i = numpy.dot(Xn, v2)
```

```
    d = numpy.dot(Xn, v1)
    j = numpy.dot(Yn, v2)

    X = ((r1**2)-(r2**2)+(d**2))/(2*d)
    Y = (((r1**2)-(r3**2)+(i**2)+(j**2))/(2*j))-((i/j)*(X))
    Z1 = numpy.sqrt(max(0, r1**2-X**2-Y**2))
    Z2 = -Z2

    K1 = P1 + X * Xn + Y * Yn + Z1 * Zn
    K2 = P1 + X * Xn + Y * Yn + Z2 * Zn
    return K1,K2
```

説明
* まず、3つの既知のポイントP1、P2、P3を定義します。これらは基準となるポイントです。

* 関数trilaterationは、これらのポイントと、それぞれのポイントからターゲットまでの距離r1、r2、r3を引数に取ります。この関数内で、まずP1を原点として設定し、P2とP3をP1からの相対座標に変換します。

* 次に、v1とv2を計算します。これらはそれぞれP1からP2、P1からP3へのベクトルです。そして、v1の単位ベクトルXnを計算します。

* v1とv2の外積を計算し、これを基にZn、すなわちv1とv2に垂直な単位ベクトルを求めます。さらに、XnとZnの外積を計算し、Ynを求めます。

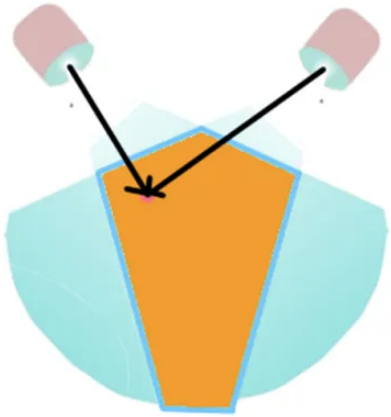
* その後、Xnとv2の内積i、Xnとv1の内積d、およびYnとv2の内積jを計算します。これらの内積は、各ベクトルの投影を表しています。

* 次に、三辺測量法の公式を使用してXとYの座標を計算します。

* 最後に、Z1とZ2を計算します。これは、2つの可能な高さの座標です。そして、最終的な2つの解であるK1とK2を計算し、これらを返します。

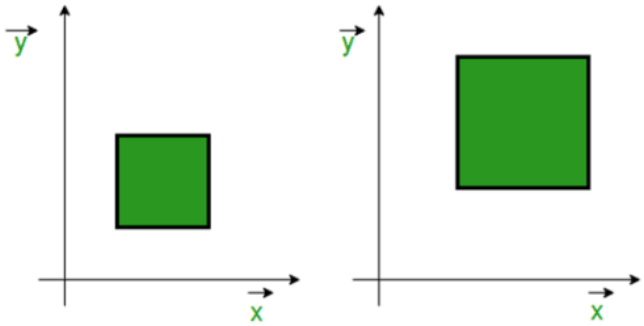
実装:

プロジェクトの理論的な基盤として、**三辺測量法**を使用して物体の位置を計算します。Raspberry Piと2つのHC-SR04超音波距離センサーを使用して、手の位置を計算し、タッチスクリーンの操作に応用します。

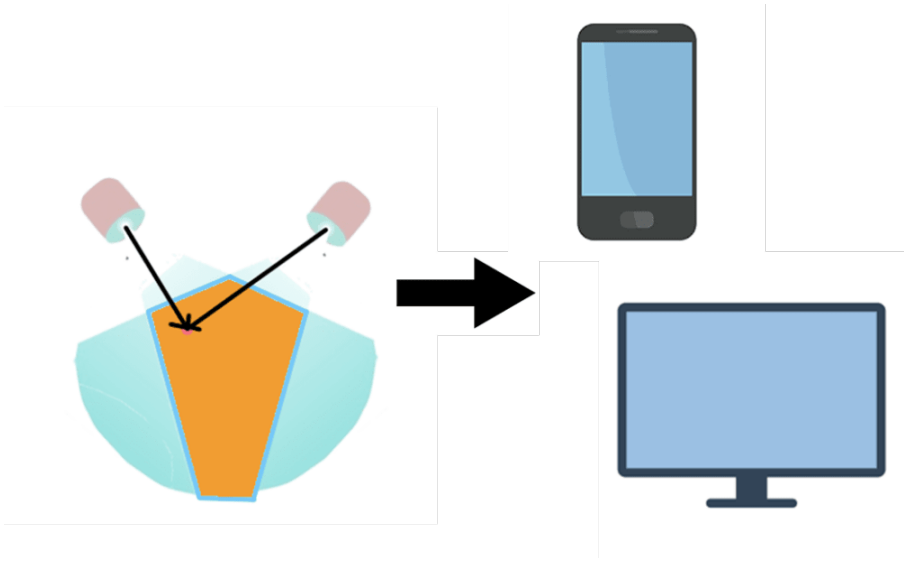


この図は、センサーの有効エリアと交差する光線のセクターを示しています。

図から分かるように、赤外線センサーの使用にはいくつかの制限があります。仮想タッチスクリーンの機能領域は、センサーから発射される光線の交差部分である有効エリアに限定されます。さたに、一つのセンサーブロックは2つのセンサーブロックをそれぞれ2つ使用するため、隣接するセンサーが干渉しないように、光を一つずつ発光する必要があります。これはノイズを引き起こすことがありますが、この設定では問題ありませんでした。



有効エリアが定義された後、仮想タッチスクリーンの長方形エリアを定義することが目標です。これは、接続されたデバイスの画面に対応するエリアに対応します。



デバイス（スマートフォンまたはPC）は、**UDP**を介してスタンドのRaspberry Piに接続され、センサーをリスニングノードとして使用します。

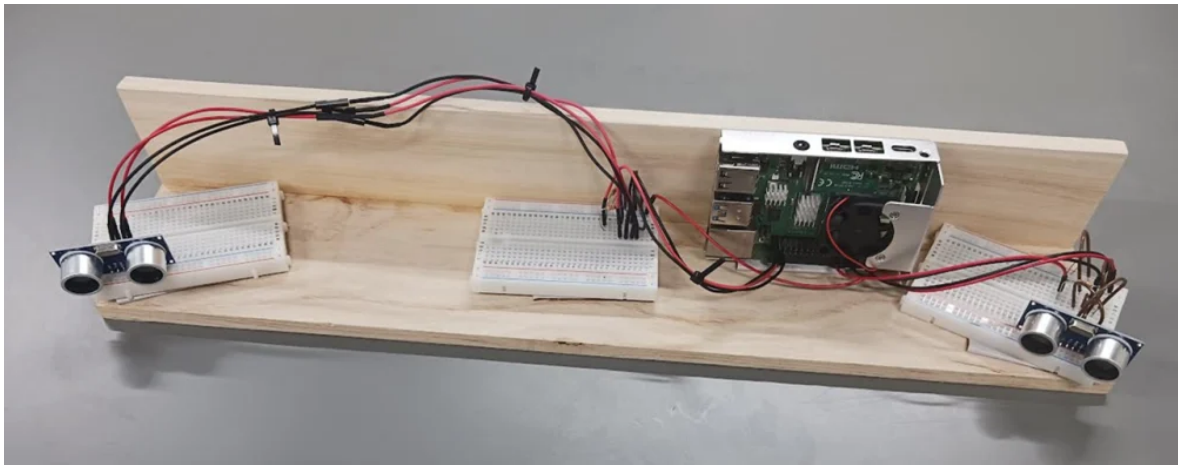
テストスタンドとコンポーネント:

ハードウェアとテストスタンドを説明します。

このプロジェクトで使用されている主要なコンポーネントは以下の通りです：

- **Raspberry Pi 4 Type B**：この小型コンピュータは、センサーからのデータを処理し、システム全体を制御します。
- **ブレッドボード**：コンポーネントを接続するためのプロトタイピング用ボードです。
- **HC-SR04超音波距離センサー**：安価で高精度の距離測定センサーです。このプロジェクトでは2つのセンサーブロックを使用しています。

使用される1つのHC-SR04センサーを示します。画像から、ブロックが2つのセンサーを含んでいます。

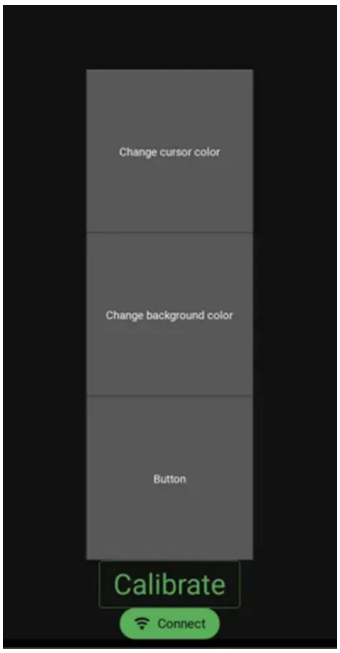


テストスタンドのセットアップと構成を示します。

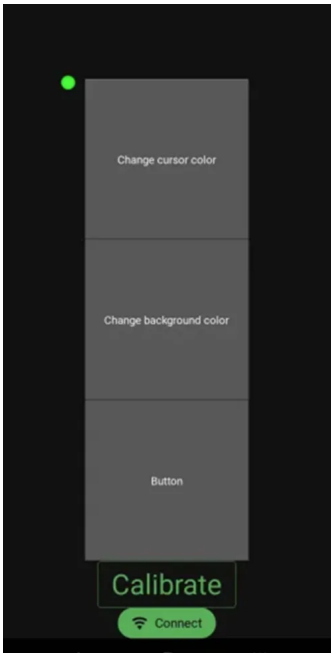
テストスタンドは、上記のコンポーネントを組み合わせたもので、約30度の角度で向き合う2つのセンサーブロックを含んでいます。これにより、より便利な交差エリアが形成されます。すべてのコンポーネントは中央のブレッドボードに接続されています。Raspberry Piはソケットから電源を供給されています。データは0.5秒ごとに記録および更新されます。最新バージョンのPythonを使用しています。

使用方法:

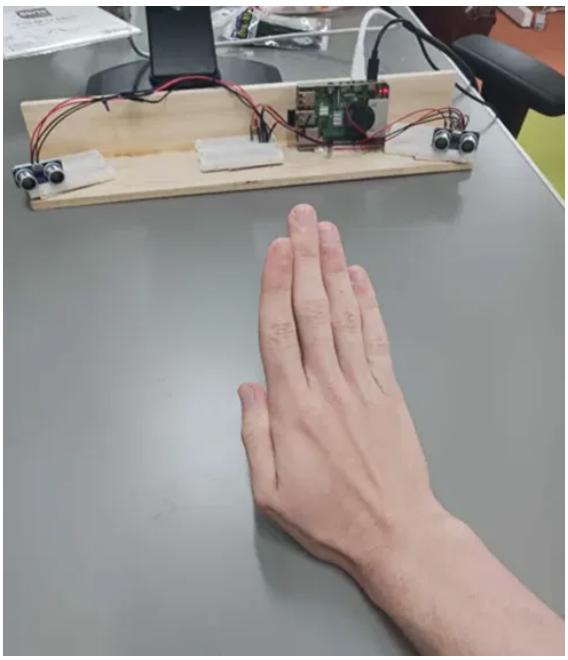
アプリケーションの使用方法について、ステップバイステップの手順を提供します。



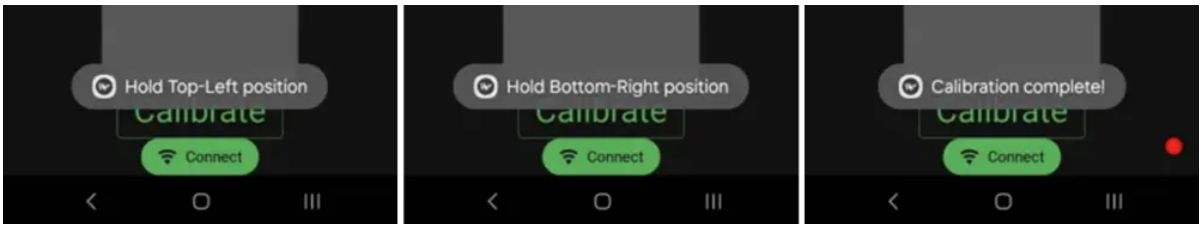
アプリケーションの起動直後には5つのボタンが表示されます: カーソル色変更、背景色変更、ボタン、キャリブレーション、接続。



最初のステップは、デバイスをRaspberry Piと同じネットワークに接続することです。次に接続ボタンを押すと、接続メッセージが表示され、カーソルが表示されます。

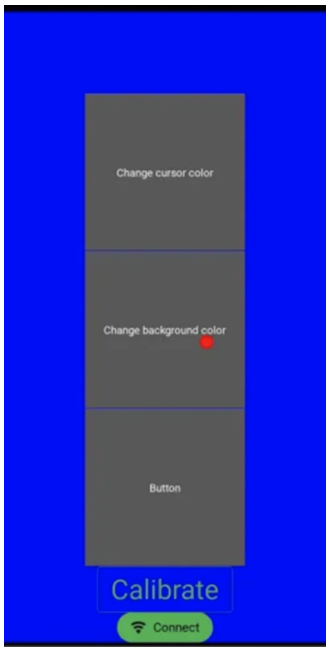


次に、センサーの前に手を動かしてカーソルが動くか確認します。動きが検出されたら、キャリブレーションを行います。



キャリブレーション手順は3つの簡単なステップに分かれています。

4. 交差する光線の有効エリア内で、仮想タッチスクリーンの長方形エリアを定義します。
5. キャリブレーションボタンをタップし、手を上部左の位置に置きます。キャリブレーションが完了するまで手を保持します。
6. 同様に、手を右下の位置に移動して保持し、キャリブレーション完了のメッセージが表示されるまで待ちます。



定義された有効エリアを使用して、画面上のカーソルを動かします。このエリア内の手の位置が画面上の位置に対応します。

手の位置を固定するとクリックが発生します。

改善点:

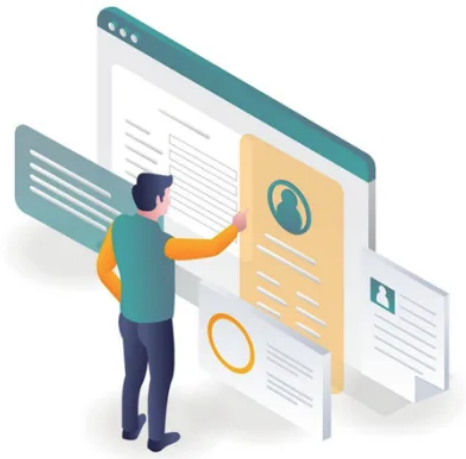
* 赤外線エミッターの場合、指示対象（例えば、手）が丸い場合に計算が最も正確です。これはキャリブレーションの過程で特に重要で、正確にキャリブレーションされたエリアはより正確なマッピングをもたらします。より良いセンサーを使用することが潜在的な改善点です。

* 現在の実装は、設計上および赤外線放射の特性により、2つの赤外線センサーブロックのみを使用して記録されたデータのノイズに制約されています。より多くのセンサーを使用することで、位置の計算がより正確になり、計算の平均化に関連する遅延が減少します。

* ノイズを減少させるために、このようなシステムの設計にMLアルゴリズムを組み込んでデータを前処理することが考えられます。これにより、指示対象が完全に丸くないために記録された位置の不一致にシステムがよりうまく対処できるようになります。

この技術は、教育、医療、エンターテイメントなど、多岐にわたる分野での応用が期待されています。未来のインタラクティブ技術としての可能性を秘めています。

発展者連絡詳細:



GitHubリポジトリのリンク: **https://github.com/RIFLE/iot-team-project (https://github.com/RIFLE/iot-team-project)**

開発チーム (GitHubのリンク):

- **Luiz Fernando Santos (https://github.com/LFRusso)** (Idea and development)
- **Sasaki Nagi (https://github.com/shakeitup108)** (Assistance and development contributions)
- **Mamanchuk Mykola (https://github.com/RIFLE)** (Assistance and management)

筑波大学 組込みプログラム開発 2024

Your next article »

Edit content (/designspark/content-creation/article/30823?lang=jp)

()

Mamanchuk Mykola(/designspark/user/Mamanchuk%20Mykola)

Edit profile (/designspark/my-account)

Enthusiastic spirit engaged in various exciting projects in computer science.

Recommended Articles

0 comments

Join the discussion