Vol.2022-CDS-34 No.2
2022/6/9

Android における遅延の少ないスクリーン表示回転に関する一考察

鳥海光伽[1]　熊倉孝太[1]　神山剛[2]　小口正人[3]　山口実靖[1]

概要：スマートフォンには端末の向きに応じてスクリーン表示方向を変える機能が搭載されている．しかし、現在の実装は悲観的なポリシーを採用しており，端末の回転後にある程度の遅延時間を経てからスクリー表示の回転が行われる．これはユーザエクスペリエンスの低下にも繋がっていると我々は考える．本稿では，端末回転の予測による遅延の少ないスクリーン表示回転手法を提案する．そして，評価実験によりその有効性を示す．

キーワード：Android，画面自動回転，加速度，加速度計，機械学習，SVM

1. はじめに

スマートフォンやタブレット型 PC が普及し，これらは多くのユーザが使用する重要な情報端末の一つとなっている．スマートフォンやタブレット PC の多くは，ユーザが自由に向きを変えて使用することが可能であり，端末に搭載されている OS(Operating System)が加速度センサより得た加速度を元に端末の向きを推定し，端末におけるスクリーンの表示方向を随時修正する．

ただし, スマートフォンやタブレット PC の主要な OS のスクリーン表示方向修正機能は近年大きくは変わっておらず，旧来の方針のまま実装されており，それが使用されている．その方針は悲観的であり，ユーザが端末を回転させて端末の OS を加速度の方向の変化を検出してもすぐにはスクリーンの表示方向を修正せず，変化の検出後に一定時間その方向の維持が検出された場合にのみスクリーン表示の回転を行う．この方針およびそれに基づく実装は，楽観的な端末方向の判断による回転の誤検出(false-positive)の削減に効果があると思われるが，回転が実装されてから一定の時間は端末の回転が検出されず(false-negative)，スクリーン表示が回転されるまでに遅延を生じさせることになる．我々は，この遅延はユーザエクスペリエンスの低下を招く場合があると予想している．

本稿では，現在のスマートフォン OS のスクリーン表示の向きおよび回転の検出の機能を紹介し，その課題を述べる．そして，機械学習を用いた遅延の少ない端末回転の検出手法を提案し，性能評価によりその有効性を示す．具体的には，著名なスマートフォン OS の一つである Android OS に着目し, 同 OS における端末回転の検出およびスクリーン表示の向きの決定の方法を解説する．そして，SVM(Support Vector Machine)を用いた近い未来における端末回転の推定による遅延の少ないスクリーン表示回転手法を提案し，性能評価によりその有効性を示す．

2. Android における端末向きの回転の検出とスクリーン表示方向の回転

2.1 Android

Android OS は主としてモバイル端末で使用される OS である. 同分野にて最も高い世界シェアを有しており**エラー! 参照元が見つかりません。**，2022 年 4 月において約 72%のシェアを有している[2]．スマートフォンやタブレット型 PC が普及し，同分野にて最も多く使用されている Android OS は非常に重要なプラットフォームの一つになっている．

また，同 OS はカーネルにオープンソースである Linux カーネルを用いており，ユーザ空間部の実装もオープンソースとなっている．よって，実装内容を確認することができるとともに，それを修正して提案手法を搭載したスマートフォンを構築することができる．

本稿では，使用者数が多く重要性が高く，現在の実装内容の確認や改変による提案手法の搭載が可能である Android に着目して考察を行う．

2.2 Android における端末向きの検出とスクリーン表示回転制御

Android OS には，加速度センサにて計測された加速度より端末の向きを推定してスクリーンの表示方向を調整する機能が搭載されている．また，同 OS はオープンソースであり，その実装も公開されている．

以下，Android 12.0.0_r96 の実装の紹介を行う．前述の様に, これら機能の実装は長期間大きな改変はされておらず，当該バージョン以外のバージョンにおいても，ソースコードツリーにおける当該機能実装ソースコードのファイルの位置の変更などの機能的な修正を伴わない変更を除き，大きく変わっていない．端末の向きの回転の検出機能は，同 OS 実装の `WindowOrientationListener.java`[3]に実装されている．クラス `OrientationSensorJudge` のメンバ変数 `mProposedRotation` が端末の向きを表して

1 工学院大学
Kogakuin University
2 長崎大学
Nagasaki University
3 お茶の水女子大学
Ochanomizu University

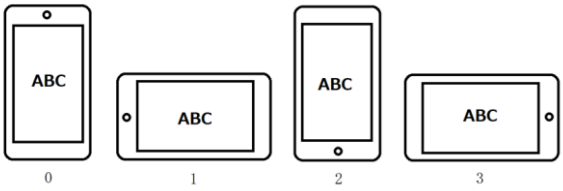

図 1　Android における変数 mProposedRotation の値と，端末の向きの関係

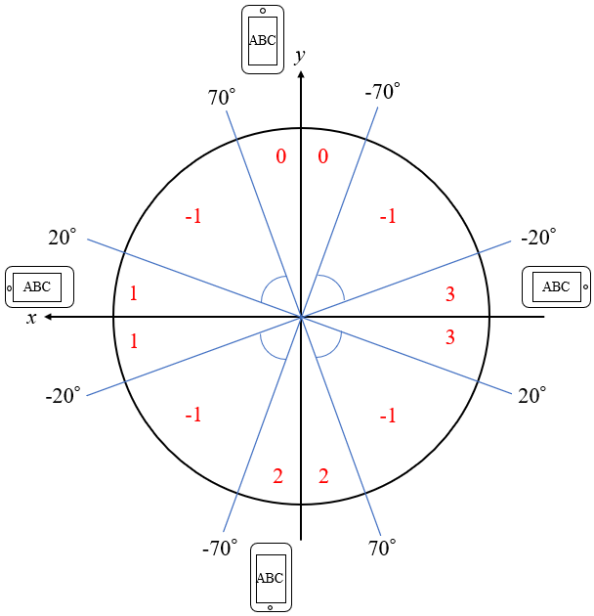

図 2　変数 mProposedRotation の値と，加速度計から推定される角度の関係

おり，この変数は，-1 以上 3 以下の 5 種類の整数値を取る．この変数に格納されている値とその値が表現する端末の向きは，図 1 の様になる．図の端末の中にある丸印は，端末の上部を意味しており，多くの端末においてこの値にカメラが搭載されている．また図における上下の方向と鉛直方向が一致している．mProposedRotation の値が 0 である状態は，portrait と呼ばれる状態であり，端末が縦長の状態で使用されていることを示している．値が 1 である状態は，landscape と呼ばれる状態であり，横長に使用されていることを示している．また，端末の値が特定できない場合は mProposedRotation の値は-1 となり，このときは端末スクリーン表示の回転は行われず現在のスクリーン表示方向を維持する．換言すると，端末の向きを高い確信度で特定できない場合は端末の回転を行わない安全性が高く悲観的な実装となっている．

Android OS は，端末の加速度計から得られた加速度から端末の向き(鉛直方向に対する角度)を推測し，この角度により mProposedRotation の値を決定する．角度と mProposedRotation の値の関係は，後述の事項を除き図 2 の通りである．すなわち，端末のスクリーンにタッチせずに，端末を十分に遅く回転させた(角度の変化が十分に小さい)場合は，図 2 の様な関係となる．同図から分かる様に，mProposedRotation=0 の状態と mProposedRotation=1 の状態の間の半分以上(20 度から 70 度)は mProposedRotation=-1 のスクリーン表示の回転を実行しない状態であり，安全かつ悲観的な設計となっている．

一方で，十分小さくない速度で端末を回転させた場合や，スクリーンのタッチを繰り返しながら端末を回転させた場合などは，図 2 とは異なる対応となる．実装の詳細を図 3 に示す．端末の回転(角度の変化)が検出されると，同図の isPredictedRotationAcceptableLocked() メソッドにより，スクリーン表示の回転を行うか否かが決定される．同メソッドが false を返した場合は，スクリーン表示の回転は実行されない．同図の赤字部の様に，端末の向きの回転が検出されても，その状態を所定の時間 PROPOSAL_SETTLE_TIME_NANOS 継続するまではメソッドの戻り値は false となり，スクリーン表示の回転は実施されない．また図の青色部の様に，最後にスクリーンがタッチされてから所定の時間 PROPOSAL_MIN_TIME_SINCE_TOUCH_END_NANOS が経過していない場合もメソッドの戻り値は false となり，スクリーン表示の回転は実施されない．

これらの様に，Android OS は端末の向きの回転が検出されても複数の所定の条件が満たされるまではスクリーン表示の回転は行わない悲観的な方針となっている．悲観的な方針を用いていることの利点と欠点に関する考察は，6 章にて後述する．

3. 関連研究

3.1 モバイル端末の握り方に基づくスクリーン自動回転に関する研究

Cheng らは，モバイル端末の加速度センサが提供する加速度値から推定される鉛直方向ではなく，ユーザの端末の握り方に基づきスクリーン表示方向を回転させる手法を提案している[4]．著者らは，ユーザが横たわっている状況などでは鉛直方向を元に端末スクリーンの表示方向を調整することは適切でないと主張し，静電容量センサと SVM を用いてユーザが端末をどのようにつかんでいるかを推定してユーザの向きを推定し，この推定されたユーザの向きに基づきスクリーンの表示方向を調整する手法を提案している．この研究では iPod Touch の 4 つの側面と背面に合計 44 個の静電容量センサを設置し，センサから得られた値を SVM により学習および分類している．そして，性能評価により，従来の重力加速度により推定する方法より 31.3%高い精度で推定できることを示している．

また，小川らは静電容量センサを用いず，端末の内臓セ

```
/**
 * Returns true if the predicted rotation is ready to be
advertised as a
 * proposed rotation.
 */
private boolean isPredictedRotationAcceptableLocked(long
now) {
 // The predicted rotation must have settled long enough.
 if (now < mPredictedRotationTimestampNanos +
PROPOSAL_SETTLE_TIME_NANOS) {
  return false;
 }
 // The last flat state (time since picked up) must have
been sufficiently long ago.
 if (now < mFlatTimestampNanos +
PROPOSAL_MIN_TIME_SINCE_FLAT_ENDED_NANOS) {
  return false;
 }
 // The last swing state (time since last movement to put
down) must have been
 // sufficiently long ago.
 if (now < mSwingTimestampNanos +
PROPOSAL_MIN_TIME_SINCE_SWING_ENDED_NANOS) {
  return false;
 }
 // The last acceleration state must have been sufficiently
long ago.
 if (now < mAccelerationTimestampNanos
   + PROPOSAL_MIN_TIME_SINCE_ACCELERATION_ENDED_NANOS) {
  return false;
 }
 // The last touch must have ended sufficiently long ago.
 if (mTouched || now < mTouchEndedTimestampNanos
   + PROPOSAL_MIN_TIME_SINCE_TOUCH_END_NANOS) {
  return false;
 }
 // Looks good!
 return true;
}
```

図 3　WindowOrientationListener.java におけるスクリーン表示回転の実施の判定メソッド isPredictedRotationAcceptableLocked()の実装[3]

ンサから得られたデータを用いてユーザが端末を握っている姿勢を推定する手法を提案している[5][6]. 具体的にはこれらの手法では，端末に内蔵さているジャイロセンサおよび加速度センサから得られたデータ，タッチスクリーンへの入力情報のみを用いて, SVM でユーザの姿勢を含むモバイル端末の握り方を推定している．そして，性能評価により，提案手法は 99.7%の精度でユーザに対する端末の向きを推定できることを示している．

3.2 ユーザの顔の向きに基づくスクリーン自動回転に関する研究

Cheng らは，モバイル端末のスクリーンをユーザの顔の向きに合わせて自動的に回転させる手法 iRotate を提案している[7]．通常の手法は，重力の方向(鉛直方向)に基づいて端末のスクリーン表示の向きを調整するが，本文献の手法では，モバイル端末のフロントカメラを用いてユーザの顔を検出し，それに応じてスクリーンの表示方向を調整する．すなわち，ユーザが横たわっている場合は，鉛直方向を下とせずに，ユーザの顔の向き(重力と水平の向き)に合わせてスクリーン表示を回転させる．著者らは，iPhone と iPad 上でリアルタイムに動作する iRotate を実装し，20 人の参加者によるフィージビリティスタディを通じて iRotate の精度と限界を評価している. また著者らは, iRotate はユーザの明示的な入力を必要とせずに，さまざまなユーザの姿勢やデバイス(端末)の向きに対応することができると主張している．

3.3 端末向きの角速度を用いた近い未来の端末角度の予想に基づく手法

我々は文献[8]にて，計測された角速度から端末の角度と，角度の変化を用いて近い未来の端末の角度を予想し，予想された端末角度に基づきスクリーン回転を実施する手法を提案している．そして，性能評価により少ない遅延時間にてスクリーン表示の回転を実行できることを示している．

同手法は，定期的に端末の加速度計により加速度を取得し，端末の角度(鉛直方向に対する端末の角度)を計算する．そして，端末角度の変化より角速度を求め，この角速度より近い未来の角度を推定する．未来の角度の予測は，角度の変化の線形近似により行う．すなわち，現在の角度に最新の角速度を加算することにより行う．

以下に詳細を記す．当該手法は，端末において定期的に加速度が得られることを前提としており，その更新間隔を $intv_{acc}$ と呼ぶ．$intv_{acc}$ は端末の加速度計の実装により決定される．$intv_{acc}$ ごとに観測される加速度から，各観測時点での端末の角度を計算し，最新の n 個の角度を保持する．それら n 個の角度を，$angle[0], angle[1], \ldots, angle[n\text{-}1]$と呼ぶ．$angle[0]$が最も古い角度であり，$angle[n\text{-}1]$が最も新しい角度である．

次に，端末の角度の時間変化より，過去の角速度（単位時間あたりの角度の変化）を計算する．角速度 $v[0], v[1], \ldots, v[n\text{-}2]$は次の様に定義される．

$$v[i] = (angle[i+1]\text{-}angle[i]) / intv_{acc} \quad (1)$$

そして，現在よりも $intv_{acc}$ 先の未来における角度

angle[*n*] を以下の式(2)により推定する．*angle*[*n*-1]は最新(現在)の角度であり，*v*[*n*-2]は最新(最も近い過去)の角速度である．

$$angle[n] = angle[n\text{-}1] + v[n\text{-}2] * intv_{acc} \quad (2)$$

提案手法は，近い未来の角度である *angle*[*n*]が，回転を判定する角度の閾値を超え，かつ最新の角速度 *v*[*n*-2]のノルムが角速度の閾値を超えている場合は，スクリーン表示の回転を実行する．同文献の例では，角度の閾値は-45 度や 45 度とし，角速度の閾値は 10 度/ $intv_{acc}$ としている．$intv_{acc}$ は 65ms である．

以下に同文献で用いられている *n*=2 の例を説明する．本手法は $intv_{acc}$ ごとに加速度を取得し，最新の 2 個の角度を記録し保持する．それら角度は，古い方から *angle*[0]，*angle*[1]と呼ばれる．最新の角速度は *v*[0]であり，

$$v[0] = angle[1]\text{-}angle[0] \quad (3)$$

となる．

近い未来(現在よりも $intv_{acc}$ 先の未来)の角度 *angle*[2]は，式(4)にて推測される．

$$angle[2] = angle[1]+v[0] \quad (4)$$

そして，*angle*[2]が角度の閾値である 45 度や-45 度を超え，かつ *v*[0]のノルムが閾値である 10 度/$intv_{acc}$ を超えているときにスクリーン表示の回転を実行する．

図 4 に本手法の動作例を示す．過去の角度が 70 度，現在の角度が 50 度なっている．よって，最新の角速度は-20 度/$intv_{acc}$ である．この角速度より，近い未来(現在より $intv_{acc}$ 先の未来)における角度は 50 度+(-20 度)=30 度と推定できる．すなわち，近い未来において端末の回転が行われると推定できる．当該手法は，この推定が行われた時点(図における現在)でスクリーンの表示方向の回転を行う．

当該手法は，端末の角度がまだ 45 度や-45 度を超えていない時点でも，近い未来に超えることが予測された場合はスクリーン表示の回転を行う手法であり，少ない遅延時間でスクリーン表示の回転の実施を期待できるとともに，楽観的な実装であり誤判定(false-positive)によるユーザの意図しないスクリーン回転が生じるか可能性が考えられる．同文献の性能評価では提案手法を Android スマートフォンに実装してスクリーン回転の遅延時間や誤判定の有無が評価されている．そして，スクリーン回転の遅延時間が 0.38 秒削減されたことが示されている．また，端末を回転させたときや端末を振った(端末の水平面に対する角度を保ったまま左右に移動させた)ときに，従来の手法では回転の誤検出は生じなかったが，当該手法では端末回転を誤検出しスクリーン表示を回転させてしまう事例が確認されたことが示されている．特に端末を振ったときは，当該手法はほぼすべての時刻において誤ったスクリーン表示方向を採用してしまっている．

表 1　Android 端末から取得する各種データの例

時刻	角度	加速度のノルム	加速度のノルムの標準偏差	角加速度
0	99.48845	9.208233	0.14792444	2.5970612
1	94.68703	7.921529	0.6740804	6.025215
2	80.98984	7.0362425	1.0119317	8.895767
3	43.937515	8.313096	0.7802977	23.355133

```
1 1:99.48845 2:9.208233 3:0.14792444 4:2.5970612
5:94.68703 6:7.921529 7:0.6740804 8:6.025215 9:80.98984
10:7.0362425 11:1.0119317 12:8.895767 13:43.937515
14:8.313096 15:0.7802977 16:23.355133
```

図 5　学習データの例

4. 提案手法

本章にて，機械学習を用いて未来のモバイル端末の回転を予想し，予想される端末の回転に基づきモバイル端末のスクリーン表示を回転させる手法を提案する．機械学習としては，SVM を用い，モバイルとしては Android スマートフォンを用いる．

本提案手法は端末において定期的に加速度を取得することを前提としており，文献[8]と同様にこの更新間隔を $intv_{acc}$ と呼ぶ．また，対象ユーザが端末を回転させたときの加速度計の測定結果が事前に得られ，学習が可能であることを前提としている．本稿では端末スクリーンと鉛直方向が平行である状況のみを想定し，端末スクリーンと鉛直方向が垂直である状況は想定しない．

提案手法は $intv_{acc}$ 毎に 3 軸の加速度を取得し，取得した加速度から，端末の角度，加速度のノルム，加速度のノルムの標準偏差，角加速度の 4 個の値を求める．加速度のノルムの標準偏差は，過去 *nsd* 個の加速度のノルムから求める．角加速度は，角度の二階微分であり，取得した加速度の数列の第二階階差数列である．

これら 4 個の値の組を，加速度を取得するたびに求める．提案手法は，最新 *m* 組の値(4**m* 個の値)を用いて回転の学習と推定を行う．

以下に *nsd*=4，*m*=4 の例の説明をする．端末の角度は加速度の x 軸と y 軸の値から算出され，端末を回転させたときの各値の具体例を表 1 に示す．ここで時刻 0 が最も古くで，時刻 3 が最も新しい時刻である．これは，端末を回転させたとき事例であるため，正例となる．

これらの最新 4 組のデータを使い，正解ラベルを 1 とした学習用データは図 5 の様になる．本表記は，svm light[9]の表記方法に従っており，左から，正解ラベル，1 次元目から 16 次元目のデータとなっている．

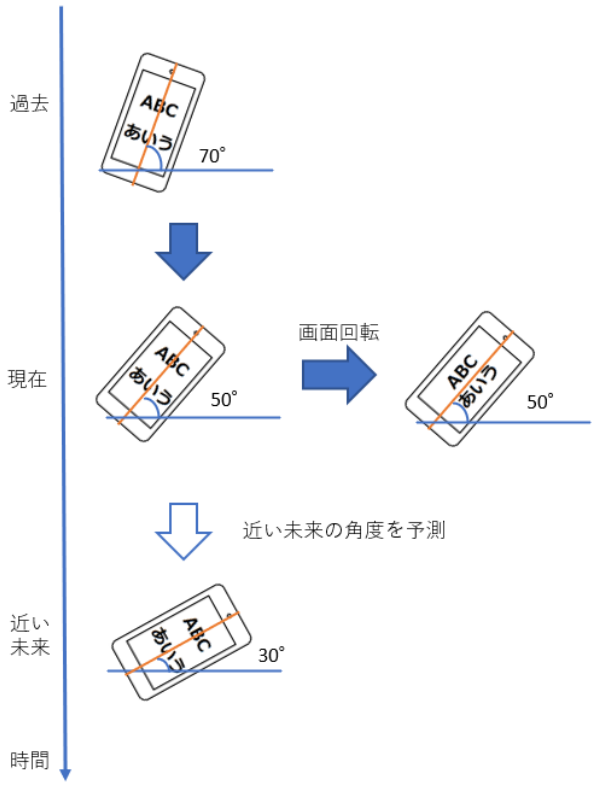

図 4　端末角度の予想[8]とスクリーン表示回転実行の動作例

前述の様に，対象ユーザが自身が端末を回転させたときの加速度の計測データを事前に提供し，これにより事前にSVM 学習モデルを作成する．そして，回転判定時は，計測された加速度を SVM 学習モデルを用いて分類し，近い未来に端末が回転されると予想された場合はスクリーン表示の回転を行う．

学習用データの正解レベルの付与は人手により行う必要があるが，本稿では人手による判断を介さずに機械的に学習用データを作成した．具体的には，m=4 として，回転実行前の 4 個の連続する角度が，75 度以上 105 度以下，75 度以上 105 度以下，0 度以上 75 度以下を満たすものを機械的に正解として抽出した．

5. 性能評価

本章にて，提案手法の性能評価を行う．

第 3.3 節で述べた様に，既存の手法[8]は端末を振ったときに誤って回転を検出してしまう課題がある．そこで，端末を回転させたときと，振ったときの加速度のデータを既存手法および提案手法に与え，これを分類させその精度を評価する．

提案手法における学習用および評価用データは回転時の正解ラベルを 1，振動時の正解ラベルを-1 として作成した．それぞれの場合について 100 個のデータ(合計 200 個のデータ)を作成し，そのうち 80 個ずつ(合計 160 個)を学習用データとし，20 個ずつ(合計 40 個)をテスト用データとした．正解率は，テストデータを SVM に入力して分類して，その分類精度により求めた．SVM は，SVM^{light} 6.02[9]を用いた．提案手法における m は 4 とし，nsd は 4 とした．

既存手法は事前学習が不要であり，正例における評価は，端末の回転を 200 回行い，その中で正しくスクリーンの向きを回転させた回数の割合を求めた．負例における評価は，端末を振り，そのときの端末スクリーンの表示方向の正しさを調査した．ただし，既存手法は加速度取得ごとに判定を行い，端末を振る場合における 1 回の定義が困難であり，負例における評価は定性的な評価にとどめる．

端末回転および端末を振ることは著者が行った．振動時のデータは，端末を縦に持ち(`mProposedRotation`=0)，水平方向に振動させる(端末を回転させずに平行移動することを繰り返す)ことにより作成した．振動の速度は毎秒 5 往復程度である．性能評価を行った環境は表 2 の通りである．既存手法[8]は実際の Android スマートフォン端末に実装されて評価が行われているが，提案手法は実際の端末から得られた加速度を計算機(PC)により学習と分類を行って評価をしている．

本評価で用いた加速度データから端末角度(鉛直方向に対する角度)を計算したものを図 6，図 7 に示す．

評価結果を表 3 に示す．端末回転(正例)における正解率は提案手法が既存手法を大きく上回り，提案手法の有効性が確認された．端末を振った場合(負例)における正解率に着目すると，既存研究[8]における評価結果と同様に誤判定が生じてしまうことや，誤判定の確率が回転時より高いことが分かる．一方で，既存手法の正解の回数や割合は前述の様に定義が困難であるが，振動中はほぼ全ての瞬間において誤った方向にスクリーンが表示され，正解率は非常に低い状況であると言える．このことから，振動時の推定に関しても大きな改善を達成できたと予想される．

6. 考察

第 2.2 節で示した様に，Android OS のスクリーン表示の回転の実施には悲観的な方針が採用されており，同 OS は端末の向きの回転を検出しても，即座にはスクリーン表示の向きの回転を実施しない．楽観的な方針としては，図 2 における-1 となる範囲が狭い方針や，回転の検出後に即時にあるいは短い時間の維持の後にスクリーン表示の回転を実施する方針などが考えられる．悲観的な実装になっている利点と欠点について考察する．

悲観的な方針のユーザエクスペリエンス上の利点としては，ユーザの意図しない端末回転の検出とそれによる誤ったスクリーン表示回転の実施の回避が考えられる．揺れている状況などで使用した場合などに意図しないスクリーン表示の回転が実施されてしまうことなどが予想され，そ

の様な回転の実施はユーザにとってストレスになると予想される．悲観的な方針のユーザエクスペリエンス上の欠点としては，ユーザが意図して端末を回転した場合に，実際にスクリーン表示の回転が実施されるまでに遅延があり，これがユーザにとってストレスになると予想される．このように誤検出の削減と，回転実施の遅延時間の削減はトレードオフの関係にあると考えられる．

積極性(悲観的や楽観的である程度)の調整も重要な課題の一つであるといえ，アンケートなどに基づく主観的な評価も有益な取り組みになると期待できる．ただし，これはユーザの嗜好に依存する問題であり，多くのユーザに適用できるような一般性を持つ優れた調整を実現することは困難であると予想される．一方で本稿では提案した手法は，積極性を調整するのではなく，機械学習により短い遅延時間で高い精度の実現を目指すものである．よって，本稿の提案手法を採用し，かつ悲観的な方針も採用することにより，誤検出の削減と，スクリーン回転実施の遅延時間の削減の両方を実現するできる可能性も考えられる．

本稿ではスクリーン表示の回転の遅延がユーザエクスペリエンスの低下を招くという前提に基づいて考察を行っており，同様に回転の誤検出による誤った表示の回転もユーザエクスペリエンスの低下を招くと予想できる．これれら低下の程度を主観評価などにより定量的に評価し，低下の大きさを考慮して調整を行うなどにより，さらなるユーザエクスペリエンスの向上に繋げることができると予想される．

7. おわりに

本稿では，Android 端末における遅延時間の少ないスクリーン表示の回転の実現に取り組み，端末から取得した加速度の値を SVM を使って解析し，近い未来の端末の回転を予想することにより遅延時間の少ないスクリーン表示の回転手法を提案した．また性能評価により，提案手法は回転操作に対して 95%の正解率で，振動操作に対して 85%の正解率で回転か否かの分類を行うことができ，既存手法よりも高い精度で予測できることが確認された．

今後の課題として，振動時の正解率の向上方法の考察，回転と振動以外の端末の状態のときの判別，より多くの被験者による評価，既存手法との比較方法の確立などが挙げられる．

謝辞 本研究は JSPS 科研費 21K11854, 21K11874 の助成を受けたものである．

参考文献

[1] 山口 実靖, "スマートフォン OS ―多彩なアプリケーション環境を現出させた立役者―", 電子情報通信学会 通信ソサイエティマガジン, 2017-2018, 11 巻, 3 号, p. 179-185, 2017. DOI: 10.1587/bplus.11.179

[2] “Mobile Operating System Market Share Worldwide | Statcounter Global Stats," https://gs.statcounter.com/os-market-share/mobile/worldwide (access 2022-05-18)

表 2 測定環境

端末名	Nexus5 (2013)
OS	Android 6.0.1
加速度の更新間隔 ($intv_{acc}$)	65 [ms]

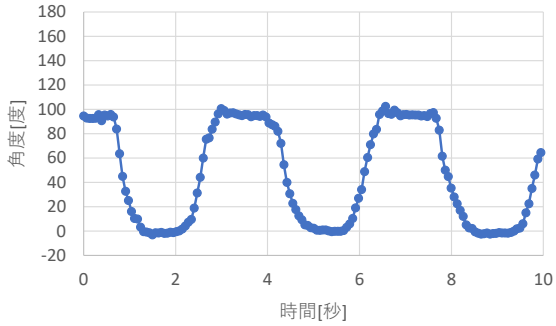

図 6 実験データ(加速度より計算した端末角度．端末回転)

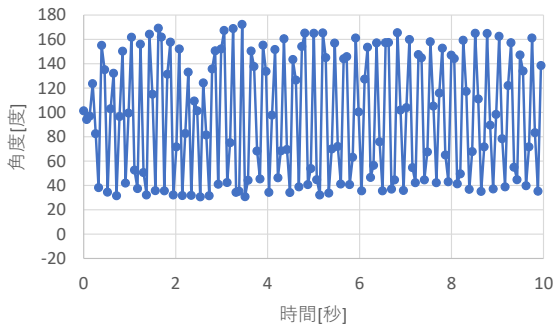

図 7 実験データ(加速度より計算した端末角度．端末振動)

表 3 性能評価結果

テストデータ	既存手法[8]	提案手法
回転と振動	-	90%
回転	73%	95%
振動	-	85%

[3] The Android Open Source Project, "WindowOrientationListener.java," https://android.googlesource.com/platform/frameworks/base/+/refs/tags/android-mainline-12.0.0_r96/services/core/java/com/android/server/wm/WindowOrientationListener.java

[4] Lung-Pan Cheng, Meng Han Lee, Che-Yang Wu, Fang-I Hsiao, Yen-Ting Liu, Hsiang-Sheng Liang, Yi-Ching Chiu, Ming-Sui Lee, and Mike Y. Chen. 2013. IrotateGrasp: automatic screen rotation based on grasp of mobile devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13). Association for Computing Machinery, New York, NY, USA, 3051-3054. https://doi.org/10.1145/2470654.2481424

[5] 小川 剛史，朴 燦鎬，"携帯端末の利用環境に依存しない端末把持姿勢認識手法"，情報処理学会論文誌デジタルコンテンツ(DCON)，Vol.4，No.1，pp.10-18 (2016)
[6] 小川 剛史，朴 燦鎬，"ユーザの姿勢を考慮したモバイル端末の把持姿勢認識"，情報処理学会論文誌デジタルコンテンツ(DCON)，Vol.5，No.1，pp.31-37 (2017)
[7] Lung-Pan Cheng, Fang-I Hsiao, Yen-Ting Liu, and Mike Y. Chen. 2012. IRotate: automatic screen rotation based on face orientation. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). Association for Computing Machinery, New York, NY, USA, 2203–2210. https://doi.org/10.1145/2207676.2208374
[8] Koga Toriumi, Takeshi Kamiyama, Masato Oguchi, Saneyasu Yamaguchi, "Device Rotation Prediction based on Acceleration Prediction in Android," 2022 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), 2022.
[9] Thorsten Joachims, “SVMlight Support Vector Machine,” https://www.cs.cornell.edu/people/tj/svm_light/ (access 2022-05-18)