Vol.2017-MBL-82 No.5
Vol.2017-UBI-53 No.5
2017/3/9

カーナビアプリのためのSVMを利用した車の停止判定方法

大平 雄貴[1]　浅見 宗広[1]

概要：車載カーナビと同等な機能をもつカーナビアプリが普及しつつあるが，カーナビアプリが有する大きな弱点の一つとして，GPS が届かないトンネル内では現在位置推定が困難になる点が挙げられる．車載カーナビでは，車両側から直接車の速度情報を取得できるため，正確な現在位置推定が可能だが，スマートフォンではその手法が選択できないためである．そこで，我々はスマートフォンに搭載されているセンサーを利用して速度を推定する手法でその課題の解決を試みている．その実現のために，車が走行状態か停止状態かを判定する機構が必要となるが，本論文では，車内に置かれているスマートフォン端末から取得される加速度センサーの値を特徴量とし，機械学習アルゴリズムの一つであるサポートベクトルマシンを利用して，車が走行状態か停止状態かを判定する方法を提案する．また，本論文ではその判定方法の精度を向上させるためにいくつか行った改良点を説明し，提案手法の精度の検証結果を報告する．

キーワード：スマートフォン，加速度センサー，GPS，サポートベクトルマシン

1. はじめに

昨今では，スマートフォンアプリとしてカーナビアプリがいくつも世に出ている．そのアプリの機能は従来の車載ナビと遜色ない水準にまでなっている．

しかし，車載ナビに比べカーナビアプリの著しい欠点となっているのが，トンネル内で自車の位置の特定が困難な点である．車載ナビでは，車の車輪の回転数などから車の速度がわかるので，正確な車の位置が割り出せる．しかし，カーナビアプリでは，通常 GPS により車の位置を割り出しているため，GPS 電波が届かないトンネル内では，車の位置が割り出せないという問題がある．

その際，GPS の代わりに車の位置特定に有用となるのは，スマートフォンに搭載されている加速度センサーである．加速度センサーは大抵のスマートフォンに搭載されており，加速度を積分すれば速度が求められることから，GPS に代わる車の位置の特定手段として期待される．

しかし，加速度センサーを用いれば，難なく車の速度が求められ車の位置が特定できるというものでもない．それには，1）初期速度，2）車の進行方向の加速度，の情報が必要となる．

初期速度については，車がトンネル内に進入して GPS により計測できる速度が計測不可となる直前の速度を採用すればよいが，車の進行方向の加速度を正確に求めることは非常に困難である．

加速度センサーから計測される値は，車に任意の角度で設置されたスマートフォンの座標系の値であり，これを車の座標系に変換する必要があるが，迅速にかつ正確な変換式をどのように求めるかの課題がある．

また，端末の向きが何らかの原因で変化したり，道路の傾斜や路面状況など様々な環境変化による影響など考慮する必要も出てくる．更に，速度を求めるには，加速度を積分する処理を伴うが，これは誤差が蓄積することを意味し，更に正確な速度推定のハードルを上げることになる．

日本の首都圏では，トンネル内においても頻繁に渋滞が発生する．従って，ユーザーの体感的にはトンネル内で渋滞により停止したらアプリの位置を示すアイコンも停止することが望ましい．停止中であればドライバーはスマートフォンの画面を注視することができ，そこで停止中であるにも係わらず不自然にアイコンが動いていたらアプリの信頼性にも影響する．

また，位置特定の精度を上げる意味でも，停止の判定精度を上げることは重要である．それは，停止しているときは確実に止めて位置特定誤差を少なくすることに加え，初期速度のリセットに利用でき，加速度積分して累積した誤差も同時にリセットできるからである．

そこで，まず停止状態を正確に判定するために，走行状態と停止状態の二状態をサポートベクトルマシン（以下 SVM）により学習し，判定する手法を考案した．

この分野における従来手法は，加速度センサーから定期

[1] ヤフー株式会社
Yahoo Japan Corporation, Chiyoda, Tokyo 102-8282, Japan

的 (20ms など) に 3 軸加速度値を計測し，1 秒間での分散が，ある閾値以下であれば停止と判定する方法がある [1]．ただし，最適な閾値をどう設定するかなど課題があるといえる．

一方，提案手法では，車の現在の状態を加速度センサーのデータとして収集し，そのデータをもとに SVM により学習を行う．そうすることで，その時々に応じたモデルが設定できるため，車種の違い，スマートフォン端末の違い，その他走行時の状況により判定精度が悪化しないようにすることができる．

2. 提案手法

本節では，提案手法について説明を行う．この手法はスマートフォン端末が車の走行・停止状態を推定するためのものであるため，想定する状況としては運転されている車内にスマートフォンがあることが前提である．

また，ある程度の振動，使用者が端末の向きを変化した際への対策は後述する方法により施されているが，基本的にはスマートフォンが車内の同じ場所・向きにあることを想定している．

以下では，はじめに、SVM の学習データを収集しつつ学習する段階について，次に、学習する際に利用する特徴量について，最後に、その他精度を向上させるための改良点について説明する．

2.1 学習段階

まず，学習段階についての説明を行う．

機械学習アルゴリズムとしては SVM を採用し，実処理には LIBSVM[2] を用いた．LIBSVM は C++により実装され，さらに jar 形式も提供されており Android・iOS 両方の OS で利用できるためである．

SVM のカーネルとしては，一般的には RBF カーネルが用いられることが多いが，今回は，Linear カーネルを用いた．RBF カーネルは適切なパラメータを設定しないと良い精度で学習せず，そのパラメータを探索するのに時間がかかるためである．一方，Linear カーネルは仕組みが単純だが，学習が速いという点と，本利用条件で要求される精度が確保されるようなパラメータの設定は容易である点から Linear カーネルを用いることにした．

ここで，走行状態とは車が時速 3km 以上で走行している状態とし，停止状態は車が時速 1km 以下である状態と定義し，学習に用いるデータは，車の走行状態時の加速度センサーの値とその際の正解状態を知るための GPS から求められる速度（以下，GPS 速度），及び車の停止状態時の同様の値を用いた．

また，GPS 速度としては，Android, iOS で提供されている GPS 速度を直接取得できるインターフェースを用いた．具体的には，Android では Location クラスの getSpeed メソッド，iOS では，CLLocation クラスの speed 属性を利用した．これらの値は，単純に緯度経度の点列の差分から割り出す方法により求めた速度 (以下，緯度経度速度) よりも正確であることが調査により分かった．

図 1 は，ある走行区間の GPS 速度の値，緯度経度速度の値，また車から直接速度情報が取得できる OBD2 コネクターを用いて取得した速度（OBD2 速度）を図示したものである．OBD2 の値が最も正確であると考えられるが，OBD2 速度の点列と GPS 速度の点列とが近い形状を示していることがわかる．一方，緯度経度速度の点列は地点によっては大きく値を跳ねあげたりしているのがわかる．

また図 2 は，GPS 速度・緯度経度速度と OBD2 速度との差分をヒストグラムとして図示したものである．GPS 速度の方が，誤差が少ないといえる．

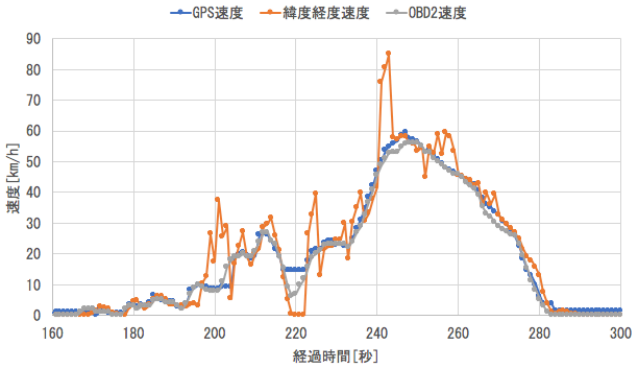

図 1 速度比較

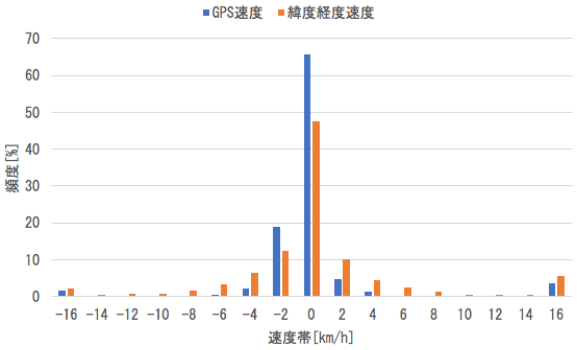

図 2 OBD2 速度との差分の分布

こうなる結果の理由として，GPS 電波は高層ビルなどの様々な障害物により乱れて，誤差の大きい緯度経度が出力されることがあげられる．一方，GPS 速度は，OS のインターフェースの説明資料に記載はないものの，GPS 電波からドップラー効果を利用し算出された速度を用いているものと推測される．結果，障害物の影響を受けにくく比較的正確な速度を出すことができるようである．

学習データの収集は GPS 電波が取得可能な場所で行う．つまり学習処理はトンネル以外の道路上で行われる．

学習データは車を運転している間 1 秒おきに計算され，

1 秒間に取得された GPS 速度と加速度センサーの値を用いる．加速度センサーの値をどう計算して学習データとするかは 2.2 節で説明する．

学習データは走行状態時のデータと停止状態時のデータをそれぞれ 60 個収集する．各データ 60 個が集まった時点で SVM の学習を行う．また，新たに 60 個ずつデータが収集された時点で各 120 個のデータで学習を行う．また同様の手順で各 180 個のデータで学習を行う．180 個のデータが集まると，収集した学習データは一度リセットされ，新たに 60 個ずつデータを収集することになる．この一連の処理が繰り返し行われる．

2.2 特徴量

この節では，SVM の学習に用いる特徴量について説明する．

加速度センサーの値から算出する，学習データの特徴量としては，平均，標準偏差，最大値，最小値の 4 項目を用いる．また，加速度センサーの値そのままで特徴量を出すのではなく，後述する方法により，加速度の平均方向成分と，それに垂直な方向の成分の大きさ（以下，垂直成分）に分解してから特徴量を算出する．

よって，特徴量としては平均方向成分の平均，標準偏差，最大値，最小値，垂直成分の平均，標準偏差，最大値，最小値の計 8 項目となる．加速度センサーの計測間隔は 20ms とした．

2.2.1 加速度の平均方向とそれに垂直な方向の成分の算出

この節では，加速度の平均方向とそれに垂直な方向の成分の算出方法について説明を行う．

一般に，スマートフォンの加速度センサーの値は，スマートフォンの縦方向，横方向，画面に垂直な方向の 3 軸（スマートフォンの中心部を基準とした座標系）方向における加速度の値が取得できるが，このまま特徴量として利用すると，端末の向きが変わった途端に，収集した加速度のデータや学習により構築したモデルが意味をなさなくなってしまう．

この問題を解決するために，スマートフォンの座標系に後述する座標変換を施し，本論文で採用した座標系に置き換える手法をとった．これにより，端末が何らかの原因で向きを変えても，同じモデルが利用できるようにした．

この座標変換の具体的な式は下記の (1)〜(4) 式となる．このとき，$(x, y, z)^T$ がスマートフォンの座標系の 3 軸の加速度センサーの値を表し，$(X, Y, Z)^T$ が採用した座標系の 3 値である．そして，$R_z(\gamma(t)), R_y(\beta(t)), R_x(\alpha(t))$ が，各軸に対する変換に必要な回転を表している．

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R_z(\gamma)R_y(\beta)R_x(\alpha) \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{1}$$

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{2}$$

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{3}$$

$$R_z(\gamma) = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

次に座標変換の回転角 $(\alpha(t), \beta(t), \gamma(t))$ の求め方を説明する．まず，1 秒ごとの加速度センサーの x,y,z 軸の計測値の平均を求める．その平均値の成分 $(\bar{a}_x(t), \bar{a}_y(t), \bar{a}_z(t))^T$ の方向を加速度平均方向と定義する．これは，車が停止していれば重力方向そのものなので鉛直下方向を示す．なお，車が加減速中であれば加速度平均方向は鉛直下方からずれる．

次に，加速度平均方向が，X 軸の負方向に一致するような座標変換式を求める．すなわち，上記の座標変換の式が下記のようになる．

$$\begin{aligned} \begin{pmatrix} \bar{a}_x(t) \\ \bar{a}_y(t) \\ \bar{a}_z(t) \end{pmatrix} &= R_z(\gamma(t))R_y(\beta(t))R_x(\alpha(t)) \begin{pmatrix} -l \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} -l\cos\beta(t)\cos\gamma(t) \\ -l\cos\beta(t)\sin\gamma(t) \\ l\sin\beta(t) \end{pmatrix} \end{aligned} \tag{5}$$

ただし，$l = \sqrt{\bar{a}_x(t)^2 + \bar{a}_y(t)^2 + \bar{a}_z(t)^2}$

上記の式からは，$\alpha(t)$ を持つ変数が消去されることに注意する．これは，α が任意であることを示している．これにより，$\sin\beta(t), \cos\beta(t), \sin\gamma(t), \cos\gamma(t)$ の値を求めると次のようになる．

$$\sin\beta(t) = \bar{a}_z(t)/l \tag{6}$$

$$\cos\beta(t) = \sqrt{1.0 - \sin\beta(t)\cdot\sin\beta(t)} \tag{7}$$

$$\sin\gamma(t) = -\bar{a}_y(t)/l_{xy} \tag{8}$$

$$\cos\gamma(t) = -\bar{a}_x(t)/l_{xy} \tag{9}$$

ただし，$l_{xy} = \sqrt{\bar{a}_x(t)^2 + \bar{a}_y(t)^2}$

次に，(1) 式の左から $R_z(\gamma(t))R_y(\beta(t))R_x(\alpha(t)$ の逆行列をかけると次式が得られる．

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R_x(-\alpha)R_y(-\beta)R_z(-\gamma)\begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (10)$$

従って，加速度平均方向が X 軸の負方向となるような変換式は，(11) 式のようになる．

$$\begin{pmatrix} a_{tx} \\ a_{ty} \\ a_{tz} \end{pmatrix} = R_y(-\beta(t))R_z(-\gamma(t))\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \quad (11)$$

ただし，YZ 平面の成分に関する特徴量としては，YZ 平面での大きさを採用した。したがって、X 軸の回転は任意になるので，$\alpha = 0$ とした．その結果、a_{tx}, a_{ty}, a_{tz} は下記の式で求められる．

$$\begin{aligned} a_{tx} &= a_x \cdot \cos\beta(t) \cdot \cos\gamma(t) \\ &\quad + a_y \cdot \cos\beta(t) \cdot \sin\gamma(t) - a_z \cdot \sin\beta(t) \end{aligned} \quad (12)$$

$$a_{ty} = -a_x \cdot \sin\gamma(t) + a_y \cdot \cos\gamma(t) \quad (13)$$

$$\begin{aligned} a_{tz} &= a_x \cdot \sin\beta(t) \cdot \cos\gamma(t) + a_y \cdot \sin\beta(t) \\ &\quad \cdot \sin\gamma(t) + a_z \cdot \cos\beta(t) \end{aligned} \quad (14)$$

この変換式により，1 秒間に計測した加速度をそれぞれ変換する．

平均方向成分は，この

$$a_{tx}(= a_h)$$

を用いる．垂直成分は次のように求める．変換後の (Y, Z) 面は加速度平均方向と垂直な面を構成している．この (Y, Z) 面でのベクトルの大きさを垂直成分と定義する．つまり

$$\sqrt{a_{ty}^2 + a_{tz}^2}(= a_v)$$

が垂直成分となる．このようにして求めた，

$$(a_h, a_v)$$

により各特徴量を算出する．

2.2.2 各特徴量の算出

1 秒間で取得された各加速度に対して前節で求めた変換式により，導出された a_h, a_n の平均，標準偏差，最大値，最小値を求める．

一秒間に N 個の加速度センサーの値が取得されたとする．

平均方向の平均，標準偏差，最大値，最小値はそれぞれ以下のようになる．

$$\bar{a}_h = \frac{\sum_{i=1}^{N} a_{hi}}{N}(= a_h) \quad (15)$$

$$s_h = \sqrt{\frac{(\sum_{i=1}^{N} a_{hi} - \bar{a}_h)^2}{N}} \quad (16)$$

$$a_{h.max} = \max\{a_{hi}, i = 1, 2, \cdots, N\} \quad (17)$$

$$a_{h.min} = \min\{a_{hi}, i = 1, 2, \cdots, N\} \quad (18)$$

垂直方向成分についても同様であるため省略する。これらの式により算出された値を学習データの特徴量とした．

2.3 改良点

前節では，基本的な学習の流れを説明した．しかし，それ以外にも精度を向上させるために改良を施した点があるので説明する．

2.3.1 最適なモデル選択

2.2 節で説明した学習は継続され SVM のモデルがその都度生成されるが，学習している際の状況によっては，あまり精度の良くないモデルが生成される可能性がある．そこで，精度をある程度安定させるために学習してできたモデルを一定数保持し続け，収集した直近の検証データで予測を行い，その正解率が最も高いモデルをそれ以降の停止判定に採用するという機構を追加した．

正解率を算出した際，新規に作成したモデルおよび今まで保持していたモデルの中で最も正解率の低いモデルが捨てられる．また，アプリ利用時に最後に採用されたモデルは端末に保存され，次回のアプリの起動時に読み込ませることで，停止判定ができない状態を極力なくすようにした．

正解率については以下のように計算する．まず，表 1 のような区分けで検証データを収集する．各速度帯の学習データを配列サイズの個数だけ保持する．また，速度帯によって走行状態のデータとして扱うか，停止状態のデータとして扱うかを区別する．配列サイズを超えた場合，古いデータから新しいデータに置換されていく．走行データを速度により細分化しているのは，偏りを防ぐためである．

表 1 正解率とその算出用の検証データ

速度帯 (km/h)	00 - 05	05 - 15	15 - 25	25 - 35
配列サイズ	256	24	24	24
停止/走行	停止	走行	走行	走行
速度帯 (km/h)	35 - 45	45 - 55	55 - 65	65 - 75
配列サイズ	24	24	24	24
停止/走行	走行	走行	走行	走行
速度帯 (km/h)	75 - 85	85 - 95	95-	
配列サイズ	24	24	24	
停止/走行	走行	走行	走行	

保持している各速度帯全ての検証データに対して，保持しているモデルで予測を行い，停止正解率 Rs，走行正解率 Rm を計算する．

$$Rs = \frac{Ncs}{Ns \cdot 100} \quad (19)$$

$$Rm = \frac{Ncm}{Nm \cdot 100} \quad (20)$$

ただし，各パラメータの意味は表 2 の通り．

表 2 各パラメータ

パラメータ	説明
Ncs	停止判定正解数
Ncm	走行判定正解数
Ns	停止状態総数
Nm	走行状態総数

これらから，正解率 Rt を計算する．

$$Rt = Rs \cdot \beta + Rm \cdot (1.0 - \beta) \qquad (21)$$

ただし，β は停止判定のウェイトを示し，今回は 0.5 とした．

2.3.2 姿勢変化の管理

使用者が端末を利用するために持ち上げたりすると，学習に用いるには不適切なデータが取得されることがある．それを防ぐために，端末の姿勢変化を管理する機構を追加した．

学習が行われている間，2 つの期間内で取得された加速度平均を計算し，その差異を確認する．2 つの期間内で取得された加速度平均とは，アプリを起動してから（及び後述のクリアを行なってから）現在までの加速度平均（これを長期加速度平均とする）と，直近 1 秒間の加速度平均の２つの加速度平均のことで，これらが成す角度が閾値以上になった場合，姿勢変化があったと判定するようにした．

なお，姿勢変化ありと判定された場合，長期加速度平均はクリアされ，新たに計算し直される．その後は次の条件を満たさない限り学習データの取得は行わないようにした．

直前の長期加速度平均と現在の長期加速度平均の成す角が，

a) 設定閾値以下の時，
端末が元の姿勢に戻されたと判断し，現在の長期加速度平均と短期加速度平均の成す角がある閾値未満である状態が一定期間継続すること．

b) 設定閾値を超える時，
姿勢変化ありの状態が継続されていると判断するが，一定期間新たな姿勢で端末が固定されている，かつ，その間に GPS 速度が一定の閾値を超えること．

これにより，ユーザが停車時にスマホを手に取り操作したり，休憩のため車から持ち出し歩くなどしたデータは無視される．

3. 検証

この節では，提案手法の精度について検証した結果を報告する．

3.1 停止判定の精度検証

まず，停止判定の精度についての検証結果を報告する．検証方法としては下記の方法で行った．

カーシェアリングサービス等により車を手配し，東京，名古屋，岐阜で実際に車を走行させ，独自に開発したアプリにより加速度，GPS 速度等をログに保存した．また，GPS 速度では評価精度に問題があるため，速度の正解データとして OBD2 を利用し専用アプリ [3] からログ保存した．

スマートフォンは，特に設置姿勢を意識せず見やすいようにクレードル等により車に固定した．これらのログに対して，従来方法と提案方法による処理を実行し比較検証した．

OBD2 から取得した速度を正しい速度とし，今回は 2km/h 未満を停止状態，2km/h 以上を走行状態と定義した．

加速度等のログと OBD2 のログ取得は別端末を利用したため，同じ UNIXtime によるタイムスタンプでもずれが発生する．このずれは OBD2 速度と GPS 速度に対して最小 2 乗法を適用して補正した．

従来方法では，3 軸の加速度成分に対して 1 秒毎に分散を求め，3 つの分散値が全て 0.001 未満の場合，停止と判定，それ以外は走行と判定した．ただし，加速度は重力加速度を単位に計測した．

表 3，表 4 には実走行条件と LogID の定義，スマホ端末と DeviceID の定義をそれぞれ示した．

表 3 実走行の条件と LogID 定義

*a	場所	*b	車
T2	東京	84	TOYOTA COROLLA FIELDER
T4	東京	107	Mazda DEMIO

*a)LogID, *b) 概算走行距離 [km]

表 4 スマートフォン端末と DeviceID 定義

DeviceID	端末	DeviceID	端末
4K	404KC	P5	iPhone5
N5	Nexus5	P6	iPhone6

3.1.1 結果とまとめ

表５，図３に各正解率（東京の走行時の一部）を示した．Rs については，SVM 利用時が優位な傾向にあった．Rm，Rt については従来手法が優位な場合もあった．

注目すべきは，LogID が T2，DeviceID が N5 の結果である．従来手法での Rs が 42%と極端に低いケースにおいても SVM 利用の提案手法では 94%と高い正解率となった．

これは従来方法では対応できない特殊な環境下でも SVM の学習処理によって高い判定精度を維持できたと推測できる．ユーザーがどんな環境下でアプリを利用するか予測がつかないため，提案手法が信頼性の観点では優位であると考えられる．

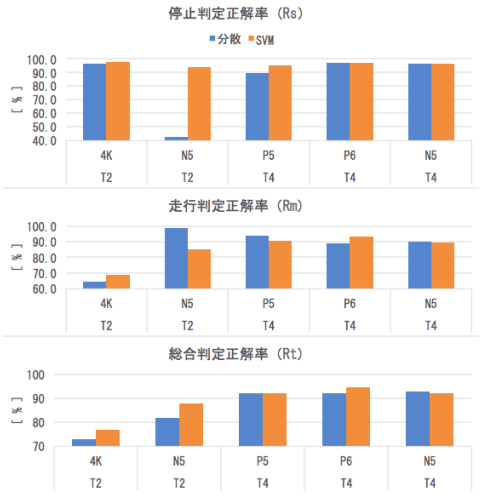

図 3 各正解率の結果

3.2 停止判定の有無によるトンネル走行の違いの検証

この節では，停止判定の機能により，どれほど実際の走行に近くなるのかということを検証した結果の報告である．

簡単に，トンネル進入時の GPS 速度のままトンネル内を走行させ，停止判定で「停止」と判定された場合のみ速度をゼロにすることを考える．なお，速度をゼロに減速する，あるいは，速度をゼロから一定速度に戻す際は緩やかに変化させるものとする．

このような処理で計算した推定速度と OBD2 の速度を比較したものが図 4 である．山手トンネルの中野長者橋出入口付近から高松入口付近まで約 5km 区間を実際に走行した時の結果である．車両は TOYOTA のアクア，端末はシャープ AQUOS SHL24 を用いた．なお，OBD2 速度の計測は別端末を利用している．OBD2 速度の結果が示すように，実際の走行時は渋滞が発生しており数回停止する状況であった．この時のトンネル進入速度は時速 50.4km であり，停止判定が「走行」と判定されたらこの速度に，「停止」と判定時はゼロに落ち込んでいることが確認できる．

速度と経過時間から走行距離を算出してグラフ化したものが図 5 である．また，OBD2 の結果との差分距離をグラフ化したのは図 6 である．停止判定の処理を行わず，トンネル進入速度でそのまま走行させた結果を「停止判定なし」で示した．グラフでは，対象とするトンネル区間長まで達したところまで表示している．「停止判定なし」でトンネル出口に達するのは約 360 秒後であるが，その時 OBD2 の結果から実際は 3km 手前を走行していることになる．一方，その時「停止判定あり」では実際より約 400m 先行しているに過ぎない．今回のような車が停止してしまうような状況では，停止判定のあり，なしでは大きく結果が異なり，停止判定の有用性は明らかである．

表 5 各正解率の結果

*1	*2	Rm 分散	Rm SVM	Rs 分散	Rs SVM	Rt 分散	Rt SVM
T2	4K	64.6	69.1	96.4	97.6	73.1	76.8
	N5	98.4	85.3	42.4	94.0	81.7	87.9
T4	P5	93.7	90.3	89.6	95.4	92.1	92.2
	P6	89.2	93.1	97.2	96.9	92.2	94.5
	N5	90.2	89.3	96.7	96.8	92.6	92.1

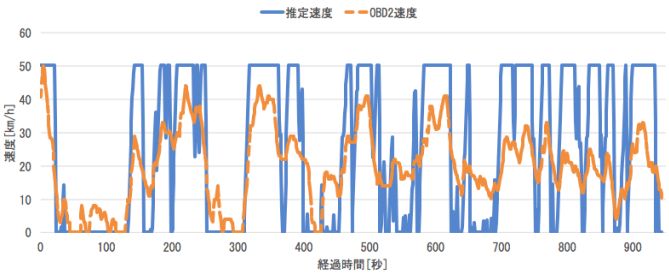

図 4 トンネル内の推定速度と OBD2 速度の比較

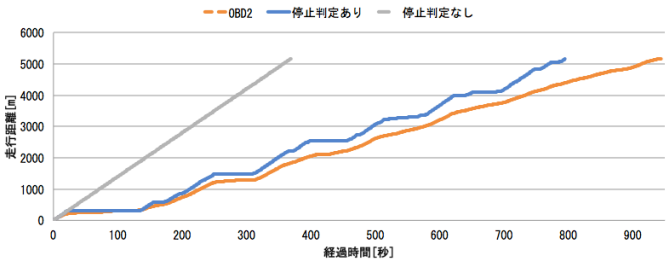

図 5 各方法によるトンネル内の累積距離の比較

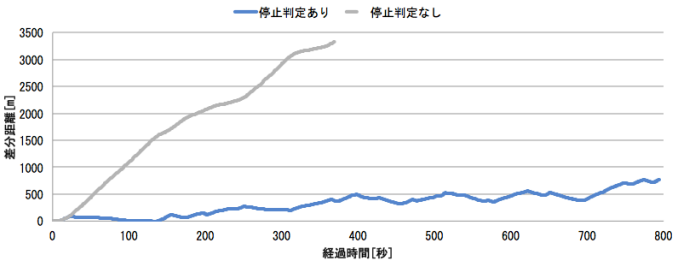

図 6 OBD2 速度との累積距離の差分

4. おわりに

本論文では，車内に置かれているスマートフォン端末から取得される加速度センサーの値を特徴量とし，車が走行状態か停止状態かを判定する方法を提案した．

また，その判定方法の精度を向上させるためにいくつか行った改良点の説明，および提案手法の精度の検証を報告した．

今後は，車の進行方向の加速度成分を抽出し，それを積

分することで速度を精度良く求められるかどうか検証を進めて行く予定である．

本提案手法はすでに「Yahoo!カーナビ」アプリ (図 7) に実装されている．是非ご利用になり本提案手法を実際に体験していただきたい．

図 7 提案手法を実装したアプリ「Yahoo!カーナビ」

参考文献

[1] 木山，高橋，祖父江，相川:「傾斜したスマ ートフォンによる自動車の 3 軸加速度算出 手法」, 情報処理学会マルチメディア，分散 協調とモバイルシンポジウム 2014 論文集,pp.16-23.
[2] LIBSVM:https://www.csie.ntu.edu.tw/ cjlin/LIBSVM/index.html
[3] OBD Info-san! MAXWIN:http://www.maxwin.jp/content/obd/m-obd-v01.html