平成 30 年度　特別研究報告書

近接通知を用いた接触事故防止支援システムの検討

龍谷大学　理工学部　情報メディア学科

T150523　西向　一哉

指導教員　三好　力　教授

内容梗概

近年、自動車の接触事故防止システムが発達し、ドライバーの負担が軽減されてきている。しかしながら、この便利になった接触事故防止システムは年式の古い車種には搭載されておらず、ドライバーの誰もが利用できているわけではない。また、二輪車においては、自動車ほど接触事故防止システムが研究されていない。そこで、本研究では、自動車、二輪車を問わず接触事故防止システムを利用したいという人のために、気軽に運転の負担を軽減することができるシステムを提案する。現代社会で普及率の高いスマートフォンを用いることで、この接触事故件数の減少にも期待できる。

目次

第 1 章 緒言 1

1.1 研究背景 1

第 2 章 関連研究及び既存技術 4

2.1 スバルリヤビークルディテクション 4

2.1.1 スバルリヤビークルディテクションとは 4

2.1.2 スバルリヤビークルディテクションの特徴 4

2.2 B2V コミュニケーションシステム 5

2.2.1 B2V コミュニケーションシステムとは 5

2.3 路車間通信(V2I)システム 5

2.3.1 路車間通信(V2I)システムとは 5

2.4 問題点 6

2.4.1 スバルリヤビークルディテクション 6

2.4.2 B2V コミュニケーションシステム 6

2.4.3 路車間通信(V2I)システム 6

第 3 章 提案手法 7

3.1 概要 7

3.2 提案システム 7

第 4 章 実験 12

4.1 実験目的 12

4.2 実験条件 12

4.3 実験 1 の実験内容 12

4.3.1 位置情報取得 12

4.3.2 サーバーに位置情報を送信 13

4.3.3 車車間距離の計算 13

4.4 実験 2 の実験内容 14

4.4.1 位置情報と時速の取得 14

4.4.2 サーバーに位置情報と時速情報を送信 14

4.4.3 車両の位置を予測 14
第 5 章 実験結果と考察 15
5.1 実験 1 の実験結果 15
5.2 実験 2 の実験結果 18
5.3 考察 21
第 6 章 結論 22
謝辞 23
参考文献 24
付録 25

第1章　緒言

1.1 研究背景

衝突被害軽減ブレーキ、レーンキープアシストといった自動車の運転支援技術が発展し、運転手は余裕を持ち運転できるようになってきた。しかしながら、二輪車には自動車のような運転支援技術が搭載されているケースは少なく、運転手への支援が無い。二輪車は自動車に比べ、精密機器を搭載可能な場所が少なく、雨や風といった外部からの障害を直に受けてしまうからだ。表 1.1 から、二輪車乗車中の死者数は自動車の死亡事故と比較しても4倍以上あり、自動車よりも二輪車の方がはるかに高い死亡率であることがわかる。

表 1.1　自動車と二輪車の死傷者数と死亡率

平成 28 年	負傷者数	死亡者数	死亡率
自動車	412,750	1,338	0.32%
二輪車	31,106	460	1.48%

このことから考えられることは、二輪運転者は体がむき出しになっており、転倒した場合に衝撃を直接受けてしまい、その結果が死亡率に繋がっているということである。また、二輪車の事故件数は、単独事故が一番多く、次に自動車との接触事故が多くなっている。この自動車と二輪車の接触事故の要因としては、次の 3 つが挙げられている。一つ目に、「自動車から見て、二輪車は死角に入りやすい」ということだ。対向車の陰、自動車のピラー、サイドミラーとバックミラーに映らない左右後方といった、自動車の運転者から見えない死角に二輪車が入り、急な路線変更で接触事故を起こすケースがある。二つ目に、「二輪車のすり抜け」である。自動車に比べ車幅が小さく、渋滞時に自動車の間をすり抜けて走っていく運転者が多い。自動車の運転手はすり抜けてくる運転手に気づけず、方向転換し接触してしまう。そして最後に、「自動車からは、二輪車はより遠く、遅く感じる」ということだ。二輪車は自動車に比べ車体が小さく、自動車の運転手からは距離間やスピード感がつかみにくいという特徴がある。

図 1.2 からも分かるように、自動車と二輪車の接触事故の発生しやすい時間帯は早朝 4 時～8時及び 20 時～22 時周辺と、通退勤時間であり渋滞による二輪車のすり抜け、加えて長時間運転による運転手の集中力の低下が事故を誘発させている。そして、図 1.3 から事故の種類を見てみると、二輪車は 2 本のタイヤで走行し、わずかなことでバランスを崩し倒れるため単独転倒が最も多くなっている。次いで右折・直進時、出会い頭の順に事故が多発している。この右折・直進時といった事故は単独事故よりも死傷者が多くなりやすく、更なる渋滞を生み、すり抜けによる事故の発生にも繋がると考える。

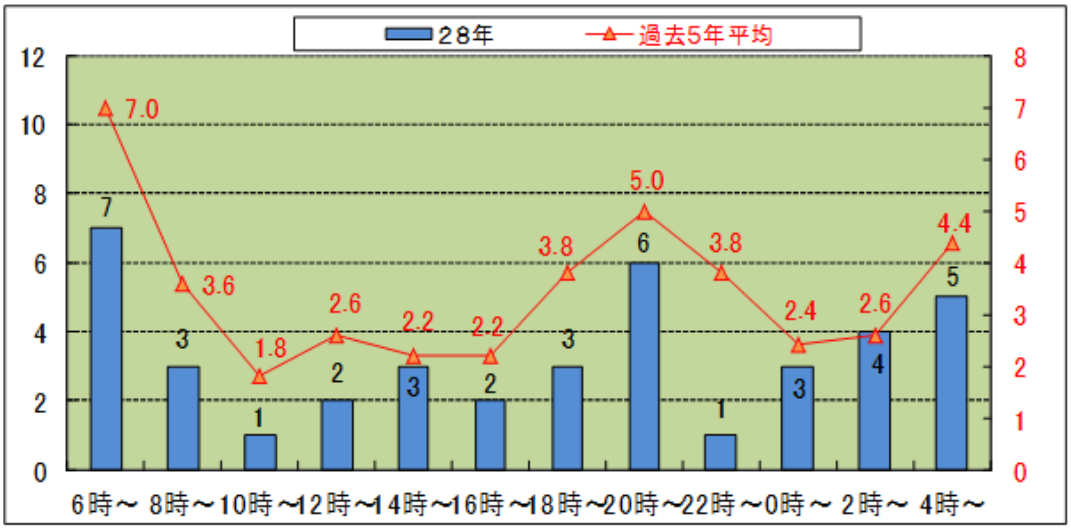

図 1.1 二輪車による事故の発生時間

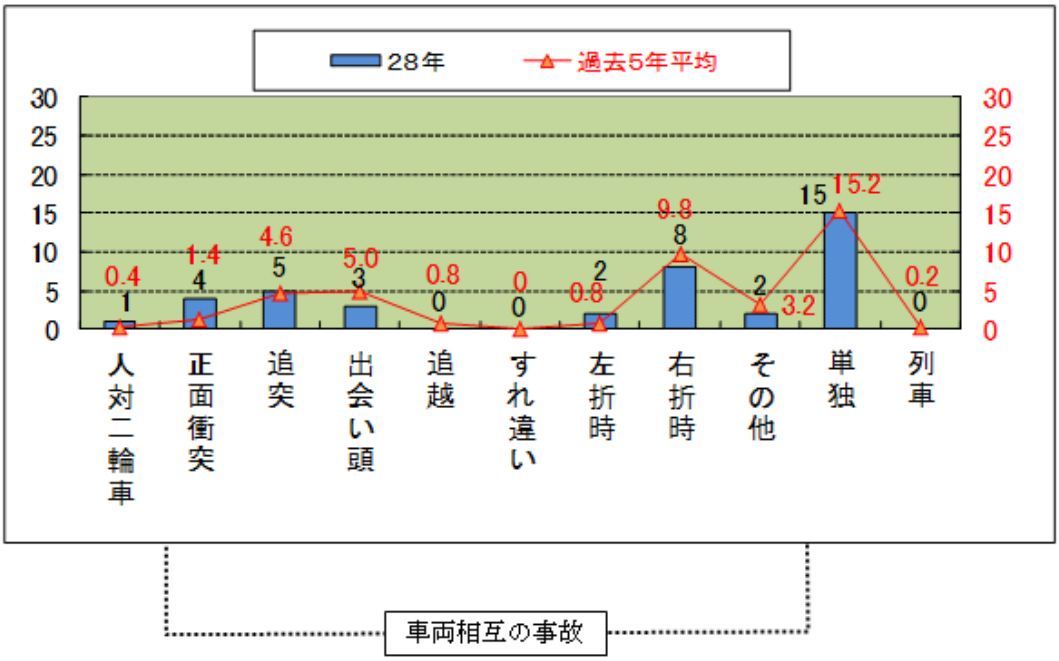

図 1.2 二輪車による事故の種類

なぜ、このような事故が発生するかというと、現状は二輪車運転者が利用できる接触事故防止ツールはなく、国は二輪車運転者への注意喚起を促しているだけだからだ。自分の周囲を注意しながら運転することは当たり前のことだが、注意喚起だけでは運転者の察知能力にも優劣があり、完璧な接触事故防止策にはなっていない。そのた

め、二輪車も自動車と同じように利用できる接触事故防止ツールが今後必要になると考える。

本研究では、二輪車と自動車の接触事故と、誘発された渋滞による二次被害を防ぐため、GPS による位置情報アプリを用いて車両同士が接触する前に接触危機を警告するシステムを提案する。また、双方の運転者は自分が所持しているスマートフォンで気軽に利用できるため、接触事故防止システムの普及にも期待できる。

第 2 章　関連研究及び既存技術

2.1 スバルリヤビークルディテクション

2.1.1 スバルリヤビークルディテクションとは

スバルが開発した後側方警戒支援システムのこと。車体後部に内蔵されたセンサーによって、自車の後側方から接近する車両を検知。衝突の危険があるとシステムが判断した場合、ドアミラー内側の LED インジケーターや警報音でドライバーに注意を促すシステムである。

2.1.2 スバルリヤビークルディテクションの特徴

特徴は3つあり、ドアミラーからは見えにくい後側方の車両を検知し注意を促す「死角車両検知技術(図 2.1.1)」。二つ目に、隣車線の後方から高速で近づいてくる車両を検知し、警告する「車線変更支援技術(図 2.1.2)」。三つ目に、駐車場などからの後退時、自車の後側方から接近する車両を検知し運転手に知らせる「後退時支援技術(図 2.1.3)」がある。

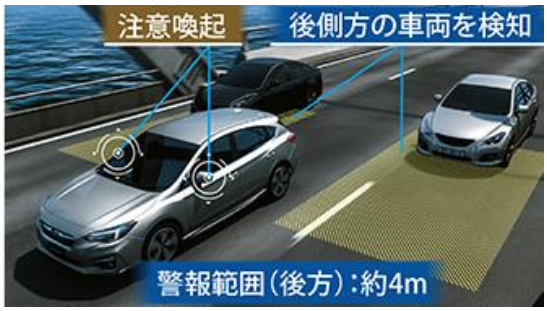

図 2.1.1 死角車両検知技術

図 2.1.2 車線変更支援技術

図 2.1.3 後退時支援技術

2.2 B2V コミュニケーションシステム

2.2.1 B2V コミュニケーションシステムとは

イスラエルの企業 Autotalks 社が、大型車両向けに開発したバイク車間の通信技術のことである。Wi-Fi を利用したこの B2V コミュニケーションシステムは、視界の内外を問わずバイク付近の車両をトラッキングし、衝突の危険が迫っている場合は運転者に知らせ、事故を防ぐための十分な時間を確保するというものだ。近距離通信技術を使い、位置や速度、進行方向、ブレーキモードなどの情報を、同種のハードウェアを装備する付近の車両とやり取りすることができる。このハードウェアは軽量でコンパクト、低電力なので、二輪車にも十分組みあわせられる。

2.3 路車間通信(V2I)システム

2.3.1 路車間通信(V2I)システムとは

道路に設置された対応機器と通信を行うことで、車車間通信同様に自車のセンサー類だけではわからない情報を入手できるシステム(図 2.3)。トヨタ クラウン初搭載の「ITS Connect」で対応している。また、センサーやカメラが搭載された路側機では、路側機自身が車や横断歩道の歩行者の有無を調べ、周辺の車に送信することでも事故防止につなげることができる。

図 2.3 路車間通信システム

2.4 問題点

2.4.1 スバルリヤビークルディテクション

一つ目に、納車以降はつけられない高額な装備であるということ。つまり、新車購入時にディーラーにオプションとして装備してもらわなければ利用することができない。中古車を購入する際にも、スバルリヤビークルディテクションが搭載されている車を選ぶと選択肢が大幅に狭まってしまう。二つ目に、二輪車は自動車に比べ車体のサイズは3~4 倍小さく、自動車に搭載されたセンサーでは感知できない場合があるということ。雨天時等でも確実に感知できることが重要である。また、このシステムは車に搭載したセンサーを利用しているため、事故などでセンサー部分が損傷してしまった場合、数十万の修理代も必要になってしまう。

2.4.2 B2V コミュニケーションシステム

ほぼすべての車両に同種の技術が備わっていなければ機能しないという点がある。つまり、普及していなければ効果が薄い。また、同種のハードウェアを搭載していなければならず、各メーカーがスムーズに採用するとは考えにくい。B2V 技術がこれらの機能を発揮するには、政府による義務化が必要となってくる。

2.4.3 路車間通信(V2I)システム

道路側に対応装置がなければならず、日本で整備されているのは東京都や愛知県など、まだまだごく一部に限られている。全国の道路に対応装置を設置することは現実的ではない。V2I 実現のためには、まずもって V2I 対応機器を搭載した車両が無いことには、車車間通信も、路車間通信もできない。まずは「ITS Connect」など V2I 対応機器搭載車を増加させる必要がある。しかし、ユーザーにとっては役に立つかどうかわからない物のために、オプション料金、つまり追加料金のコスト増を受け入れるかどうか、という問題点がある。

第 3 章　提案手法

3.1 概要

本研究では、自動車と二輪車の接近を運転者に通知するアンドロイドアプリを作成することで、車両同士の接触を防ぎ、現状は利用できる防止ツールがない二輪車も、自動車と同じように利用できるようにする。これに加え、接触事故防止システムの普及率を増加させることも目的とする。

関連研究及び既存技術は二輪車には設備できず、いずれも普及率が低く安全システムを使用する環境が非常に少ない。最も簡単な解決策として「Android Auto」や「Apple CarPlay」、カーナビなどの純正車載端末に B2V といった安全システムを組み込むことが望ましい。しかし、車自体のエンジンなどを制御するコンピューターにも関わるため、車のコントロールを Google や Apple に委ねたくないと自動車メーカーの反発がある。現状、自動車が利用できる防止ツールの問題点は、利用するのには金額面で敷居が高いことが挙げられる。また、ユーザーが支払う値段に対してのコストパフォーマンスが悪く、実用性が薄いということもある。こうした要因が普及率の低迷に繋がっており、ユーザーは普及していない安全システムを高いお金を支払ってまで利用しようとは考えない。

そこで、本研究では GPS を用いた位置情報を利用するアプリを作成することにより、自動車運転者だけでなく、双方の運転者が所持している携帯端末から気軽に接触事故防止システムを利用できるようにする。また、端末同士の通信はアプリ間だけで行わずサーバーを用いる。その理由は、アプリ間での通信では自動車と二輪車が多対多になってしまうからである。アプリであれば既存技術のように高い初期投資は要らず、全年代でのスマートフォン利用率が 2017 年時点で 71.3%となっている現代社会においては、携帯端末さえあれば誰もが利用することができると考える。また、アプリは携帯端末を用いて利用するため、設置場所のない二輪車運転者でも利用できる。そして、将来は、google map のナビゲーションの一部の機能になれる事が期待できる。

3.2 提案システム

自動車と二輪車の接近を検知する手法としてスマートフォン等の端末に、GPS を用いて得た位置情報を利用することを考えた。

位置による手法は以下である。

各端末にインストールしたアプリで位置情報を取得する。この位置情報をサーバーに送信し、サーバーを経由して得た、他の端末の位置情報と自分の位置情報を用いて、距離を計算する。算出された距離から、車両の接近を運転手に知らせる。ユーザーは利用する端末に、自分が運転する物が「自動車」、「二輪車」のどちらなのかを設定しておく。次に、アプリを起動後、各端末は GPS を用いて現在地の位置情報を取得

し続ける。そして、取得した位置情報をサーバーに一定間隔で送信する。自動車側の端末はサーバーから自分が送信したデータと、「二輪車」と設定されている端末のデータを受け取り2点間の距離を一定間隔で計算する。二輪車側の端末は、自分のデータと、「自動車」と設定されている端末データを受け取り2点間の距離を一定間隔で計算する。この距離が短くなっていけば車間が狭まっているため、運転者に車両同士が接近していることを知らせる。通知後、運転者が通知を認識できたら、対象としていた端末との計算を終了し、再び別の端末との距離を計算する。

図 3.2.1 にシステム構成のフローチャートを示す。

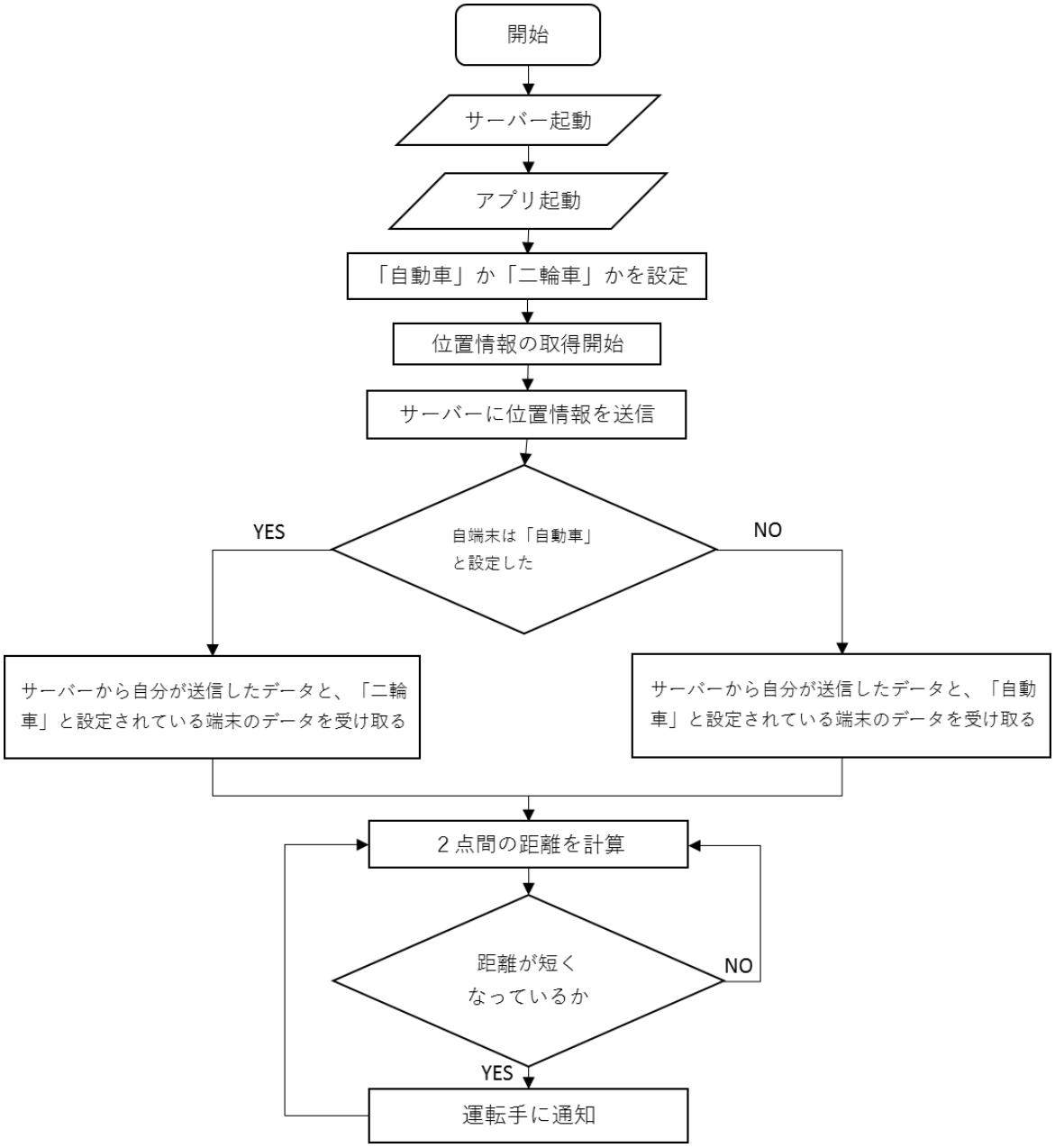

図 3.2.1 位置情報を用いた車両衝突防止システムのアルゴリズム

速度による手法は以下である
各端末にインストールしたアプリで走行中の時速を取得する。サーバーに取得した時速を送信し、サーバーから他端末の時速を得る。得た他端末の時速と自分の時速を比較することで、運転手に車両の接近を知らせる。ユーザーは利用する端末に、自分が運転する物が「自動車」、「二輪車」のどちらなのかを設定しておく。アプリを起動後、各端末は GPS を用いて現在地の位置情報を取得し続ける。そして、取得した位置情報をサーバーに一定間隔で送信する。また並行してアプリで時速を取得する。自動車側の端末は、位置情報と同じようにサーバーが受信した情報の中から、自分と距離が近く「二輪車」と設定されている端末を選択し、その端末の時速情報を受け取る。この時速情報から、二輪車と自動車のだいたいの距離が予測できるため、後方に車両が接近している可能性があると運転手に警告を送る。通知後、運転者が通知を認識できたら、対象としていた端末との通信を終了し、再び別の端末との通信を開始する。
図 3.2.2 にシステム構成のフローチャートを示す。

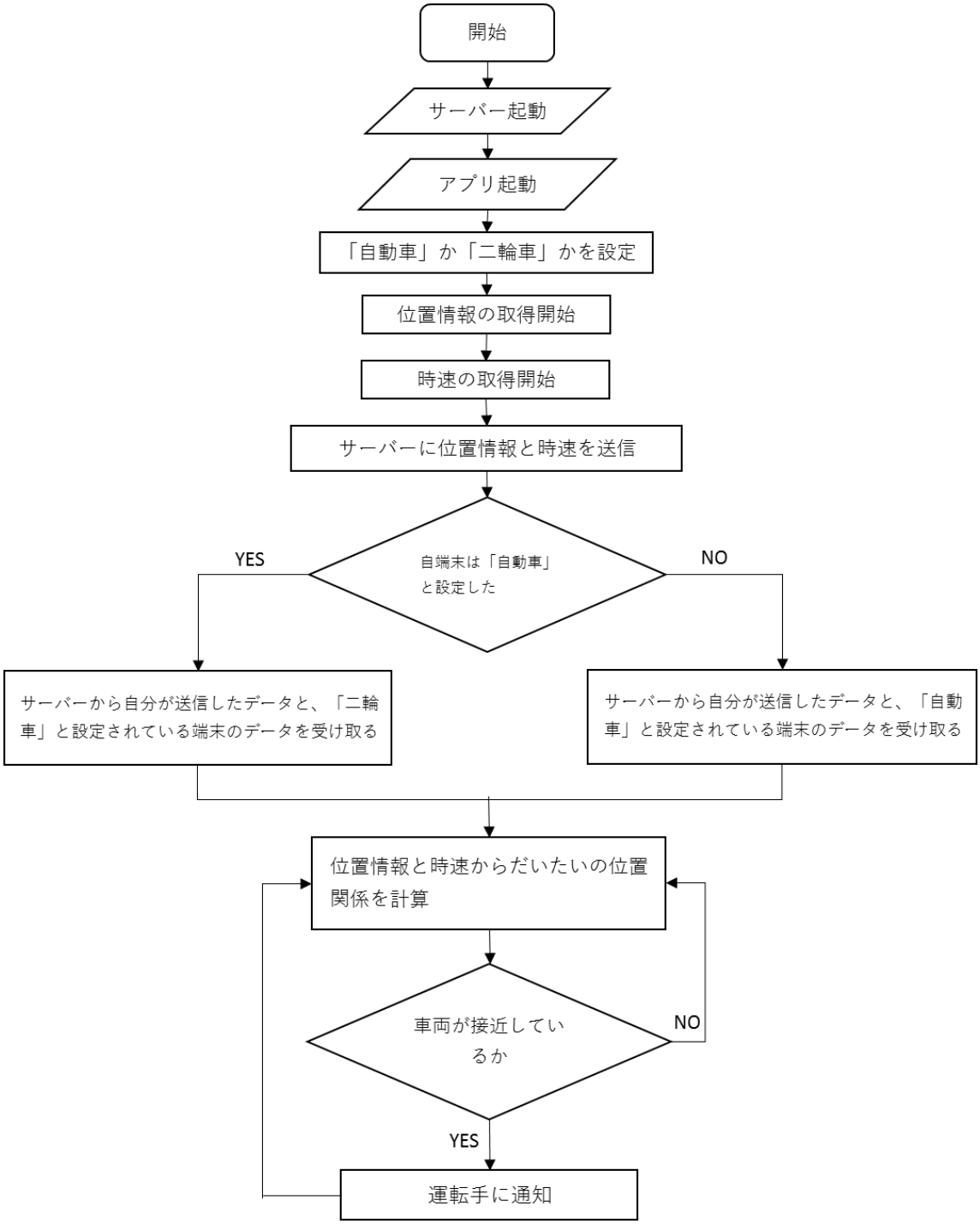

図 3.2.2 速度を用いた車両衝突防止システムのアルゴリズム

第 4 章　実験

4.1 実験目的

本実験は、第 3 章で述べたアルゴリズムの通りに行うと、位置情報や平均速度を用いて車両同士の接触を防止できるのか確認するために行った。そのため、アンドロイドタブレットを 2 台用意し、実際に各タブレットで位置情報や時速を取得して実験を行う。

4.2 実験条件

表 4.2 に実験条件を示す。

表 4.2 実験条件

実験場所	龍谷大学瀬田キャンパス
使用機器	タブレット(Android)
実験機器	PC(Windows)
使用ツール	AndroidStudio Apache PHP MySQL

4.3 実験1の実験内容

4.3.1 位置情報取得

実験 1 では位置情報を、AndroidStudio で位置情報を GPS から取得しサーバーに送信するアプリを作成して、用意した2台のタブレットにインストールし取得した。また、実験 1 は 2 車両が移動している状況ではなく、自動車側が信号待ちなどで停止していて、二輪車が後方から時速 10km で車両の隙間をすり抜けながら接近してくるという状況を仮定する。一般的に時速 10km で走行する場合の安全な車間距離は 5m とされている。そのため、全長約 12m の直線距離を用意し、0.5 秒ごとに位置情報を取得した。また、自動車後方 1m まで接近した。この実験を計 5 回行った。この間、自動車側の端末も同様に 0.5 秒間隔で同じ場所の位置情報を取得し続けた。

4.3.2 サーバーに位置情報を送信

はじめに、Apache を起動しサーバーを起動させた。次に MySQL で DATABASE「locationlog」を作成し、その中に TABLE「locationlog」を作成した。TABLE には id、username、latitude(緯度)、longitude(経度)、timestamp(時刻)を表示できるようにした。そして、この TABLE にアンドロイド端末が取得した位置情報を送信し続ける。
以下に、TABLE「locationlog」に情報を表示した時の様子を示す。

```
mysql> select * from locationlog;
+----+----------+-----------+------------+---------------------+
| id | username | latitude  | longitude  | timestamp           |
+----+----------+-----------+------------+---------------------+
|  1 | 端末２   | 34.962956 | 135.938926 | 2018-10-31 16:17:30 |
|  2 | 端末２   | 34.962957 | 135.938925 | 2018-10-31 16:17:51 |
|  3 | 端末２   | 34.962957 | 135.938925 | 2018-10-31 16:18:11 |
|  4 | 端末１   | 34.937392 | 135.927594 | 2018-10-31 16:18:15 |
|  5 | 端末２   | 34.962957 | 135.938925 | 2018-10-31 16:18:31 |
|  6 | 端末２   | 34.962938 | 135.938944 | 2018-10-31 16:18:51 |
|  7 | 端末１   | 34.937392 | 135.927594 | 2018-10-31 16:18:55 |
|  8 | 端末２   | 34.962938 | 135.938944 | 2018-10-31 16:19:11 |
|  9 | 端末２   | 34.962938 | 135.938944 | 2018-10-31 16:19:31 |
| 10 | 端末１   | 34.937392 | 135.927594 | 2018-10-31 16:19:35 |
| 11 | 端末２   | 34.962957 | 135.938925 | 2018-10-31 16:19:51 |
| 12 | 端末２   | 34.962956 | 135.938927 | 2018-10-31 16:20:11 |
| 13 | 端末１   | 34.937392 | 135.927594 | 2018-10-31 16:20:15 |
| 14 | 端末２   | 34.962956 | 135.938927 | 2018-10-31 16:20:31 |
| 15 | 端末２   | 34.962956 | 135.938927 | 2018-10-31 16:20:51 |
| 16 | 端末２   | 34.962947 | 135.938903 | 2018-10-31 16:21:11 |
| 17 | 端末２   | 34.963478 | 135.939268 | 2018-10-31 16:21:32 |
| 18 | 端末１   | 34.937392 | 135.927594 | 2018-10-31 16:21:35 |
| 19 | 端末２   | 34.963538 | 135.939314 | 2018-10-31 16:21:52 |
| 20 | 端末２   | 34.963538 | 135.939314 | 2018-10-31 16:22:12 |
+----+----------+-----------+------------+---------------------+
```

図 4.3.2　TABLE「locationlog」の情報

4.3.3 車車間距離の計算

4.3.2 で送信された各端末の位置情報をサーバーから受信し、latitude(緯度)、longitude(経度)の情報と、Location#distanceBetween メソッドを使用して 2 点間の距離を測る。

4.4 実験 2 の実験内容

4.4.1 位置情報と時速の取得

実験1と同様に、AndroidStudio で時速と位置情報を GPS から取得してサーバーに送信するアプリを作成し、用意した2台のタブレットにインストールし取得した。実験 2 も 2 車両が移動している状況ではなく、自動車側が信号待ちなどで停止していて、二輪車が時速 10 km/h ~20 km/h で自動車後方から二輪車が接近すると仮定し、実験を行う。二輪車が自動車後方 30m から 5m まで接近すると仮定した場合、時速 20 km/h では 3.6 秒、時速 15 km/h では 4.8 秒、時速 10 km/h では 7.2 秒かかる。そのため、実験 2 では自動車後方 30m 付近から 0.5 秒ごとに速度を取得し、だいたいの位置を予測する。時速 20 km/h、15 km/h、10 km/h と時速ごとに 3 回ずつ、計 9 回行った。

4.4.2 サーバーに位置情報と時速情報を送信

4.3.2 と同様に、自動車側の端末と二輪車側の端末の時速情報を各端末がサーバーに送信する。

4.4.3 車両の位置を予測

時速 1km/h で 0.5 秒間に進む距離は約 0.14m である。実験 2 では自動車側は停止していると仮定しているが、二車両が共に移動している場合は時速を差し引きし、距離の予測を行う。また、0.5 秒間で二輪車が進む距離は、時速 20 km/h で約 2.77m、時速 15 km/h で約 2.08m、時速 10 km/h で約 1.39m となる。

第 5 章　実験結果と考察

5.1 実験 1 の実験結果

自動車側の端末が同じ場所で取得し続けた位置情報と、約 12m 離れた場所から時速 20km で接近してくる二輪車の端末が取得し続けた位置情報を以下に示す。また、自動車に二輪車が接近していく中で位置情報を取得し、この実験を5回行っている。

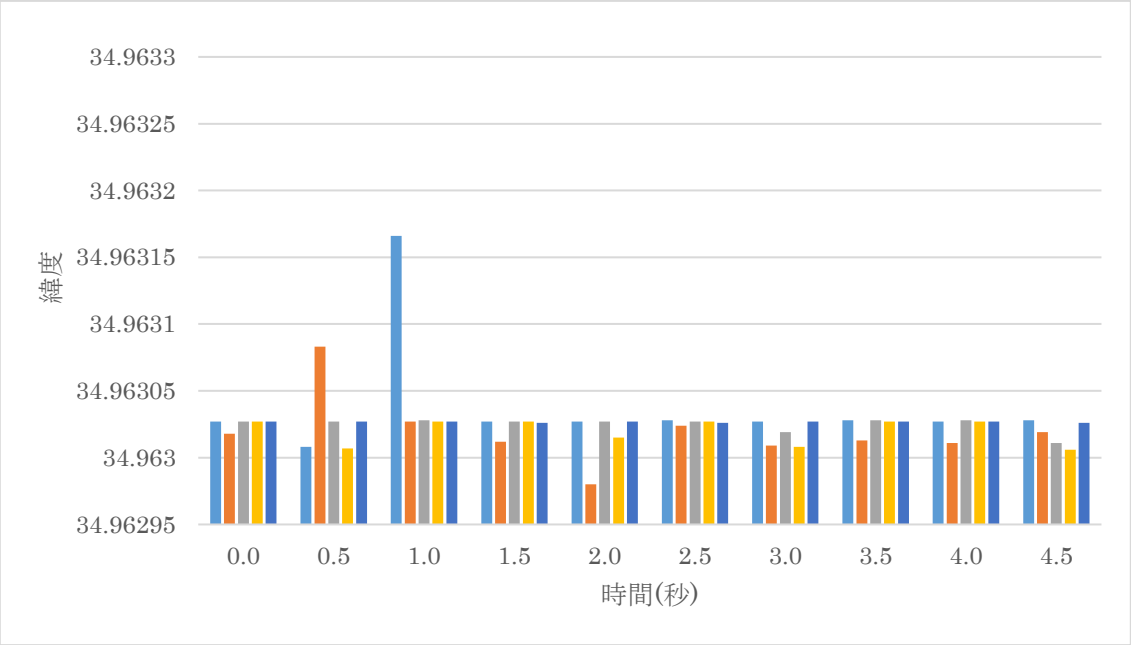

図 5.1.1　自動車端末の位置情報(緯度)のグラフ

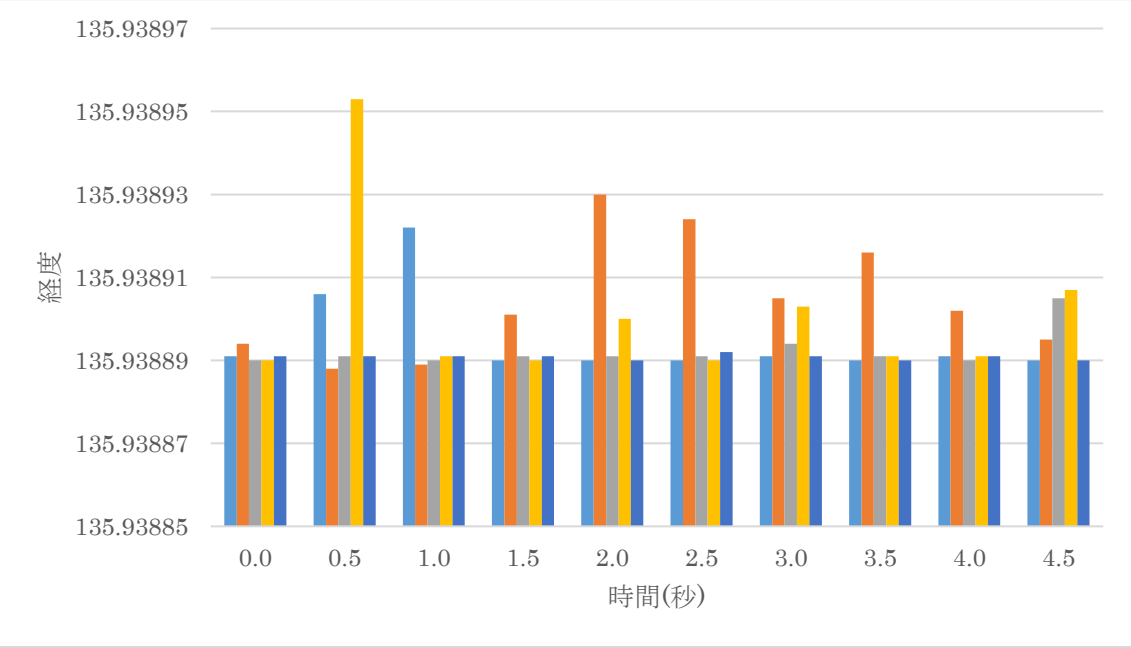

図 5.1.2　自動車端末の位置情報(経度)のグラフ

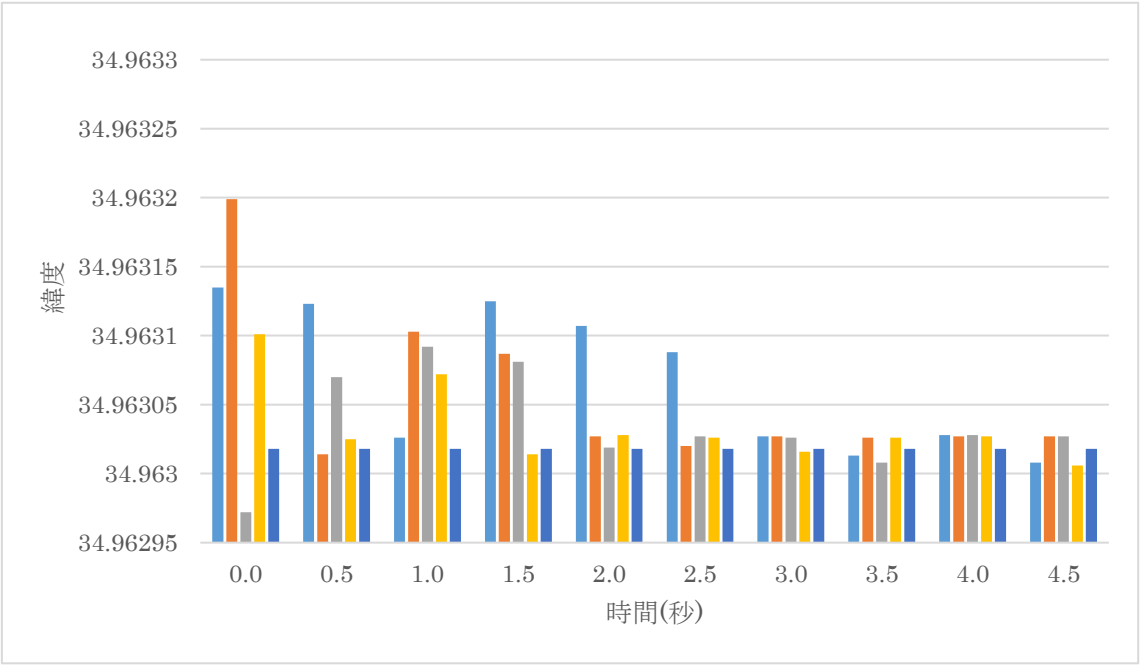

図 5.1.3 二輪車端末の位置情報(緯度)のグラフ

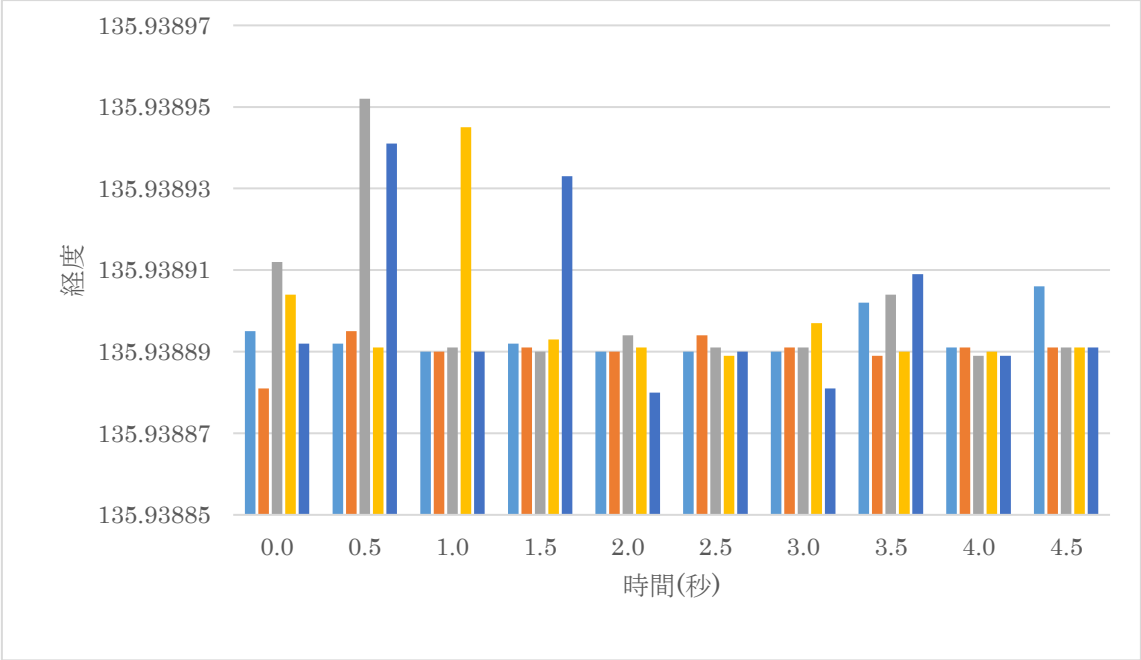

図 5.1.4 二輪車端末の位置情報(経度)のグラフ

次に、自動車側の端末と、0.5 秒ごと(約 1.1m ごと)に自動車側に接近していく二輪車側の端末が取得した位置情報(緯度、経度)から 2 点間の直線距離を計算した結果を以下に示す。

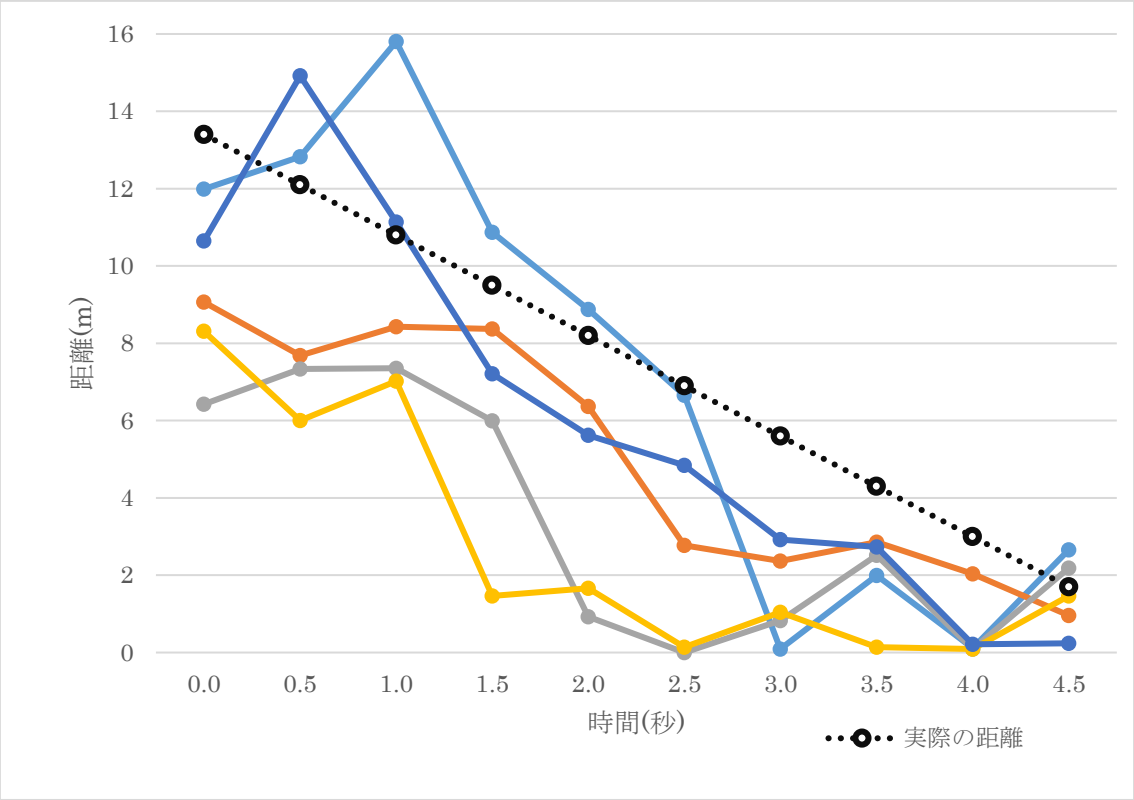

図 5.1.5 緯度、経度から求めた 2 点間の距離

5.2 実験 2 の実験結果

約 30m 離れた場所から接近してくる二輪車の端末が取得し続けた時速情報を以下に示す。また、時速 20km/h、15km/h、10km/h と時速ごとに 3 回ずつ測定し実験を行った。

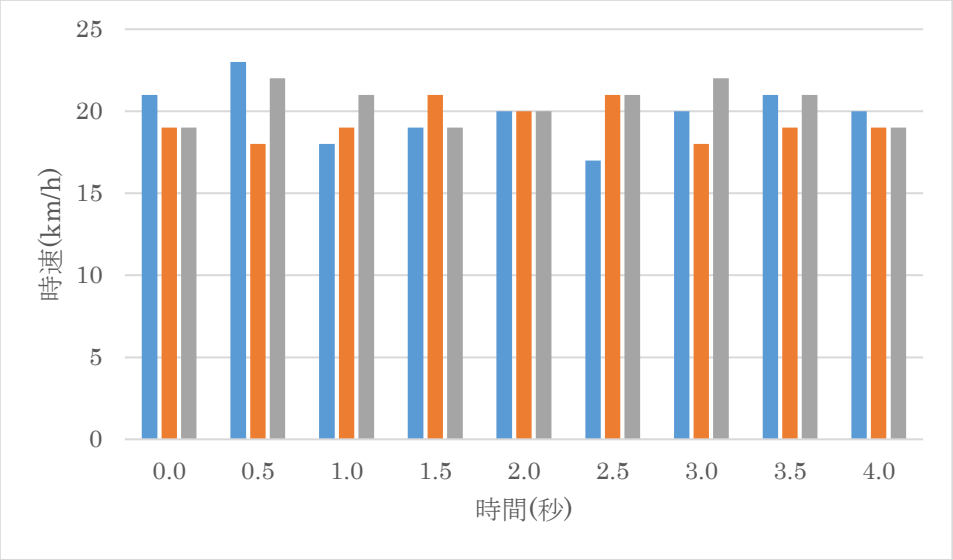

図 5.2.1 車を時速 20km/h で走行した時の測定時速

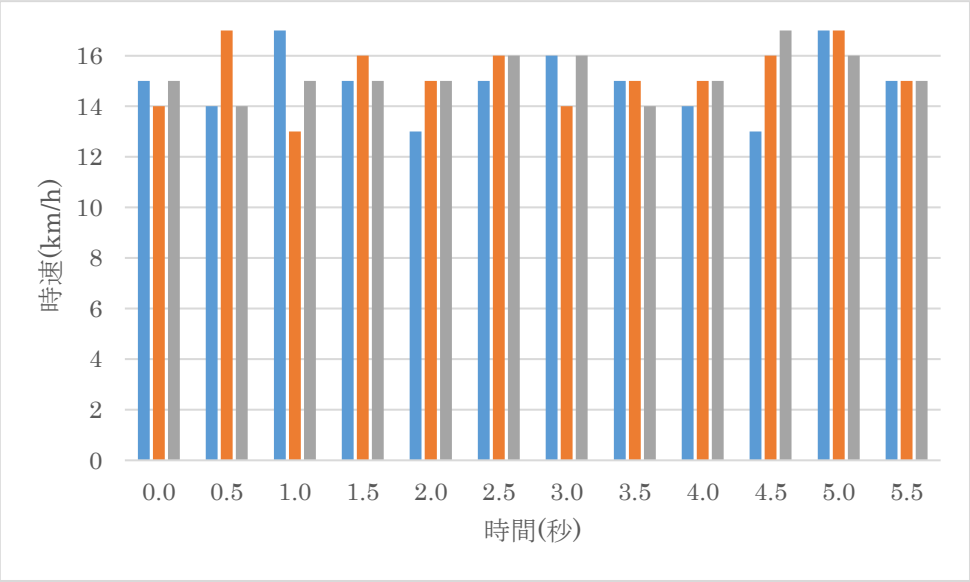

図 5.2.2 車を時速 15km/h で走行した時の測定時速

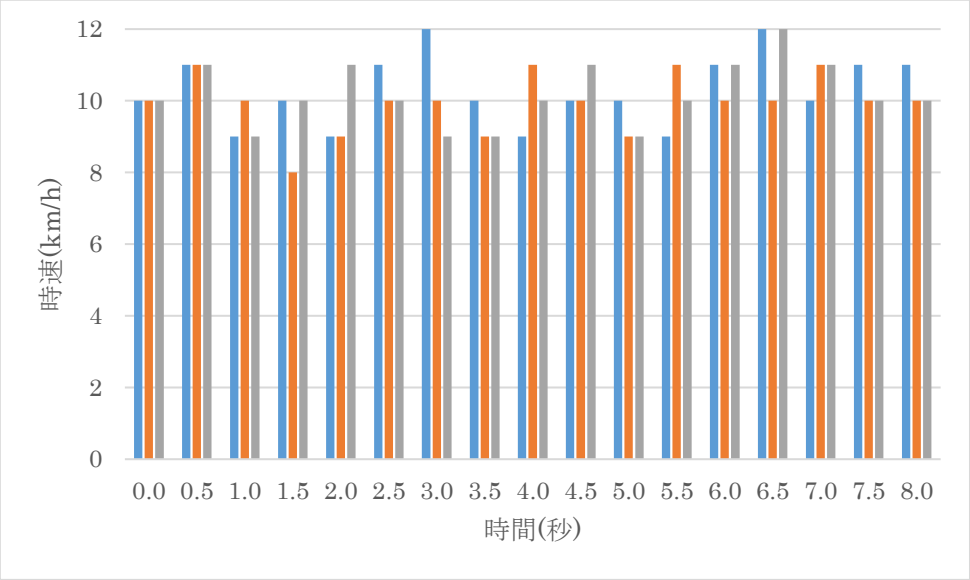

図 5.2.3 車を時速 10km/h で走行した時の測定時速

次に、二輪車側の端末が取得した時速情報から車両間のだいたいの距離を、時速 20km/h、15km/h、10km/h と時速ごとに計算した結果を示す。また、二輪車は自動車の約 30m 後方から接近しているとする。

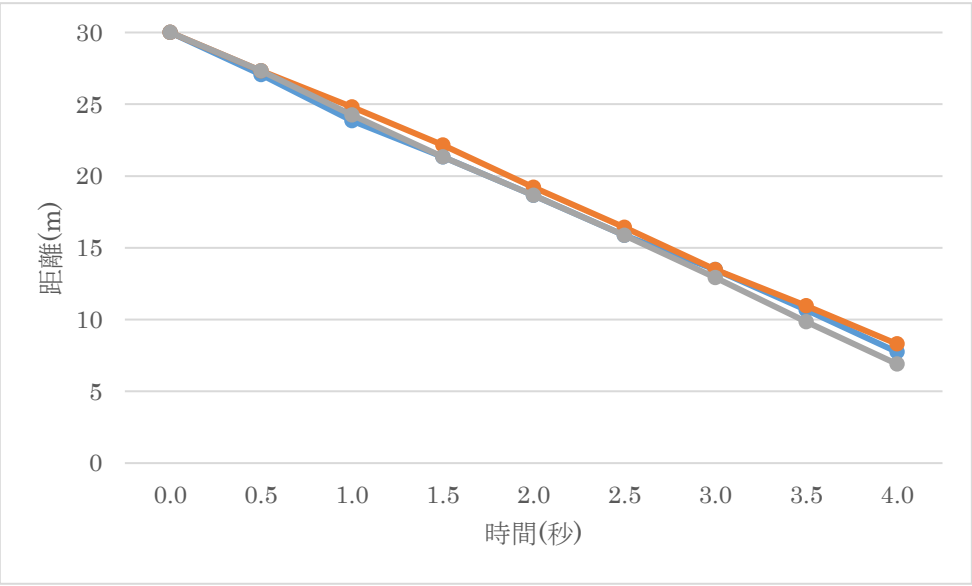

図 5.2.4 時速 20km/h 時の時速情報から計算した 2 点間の距離

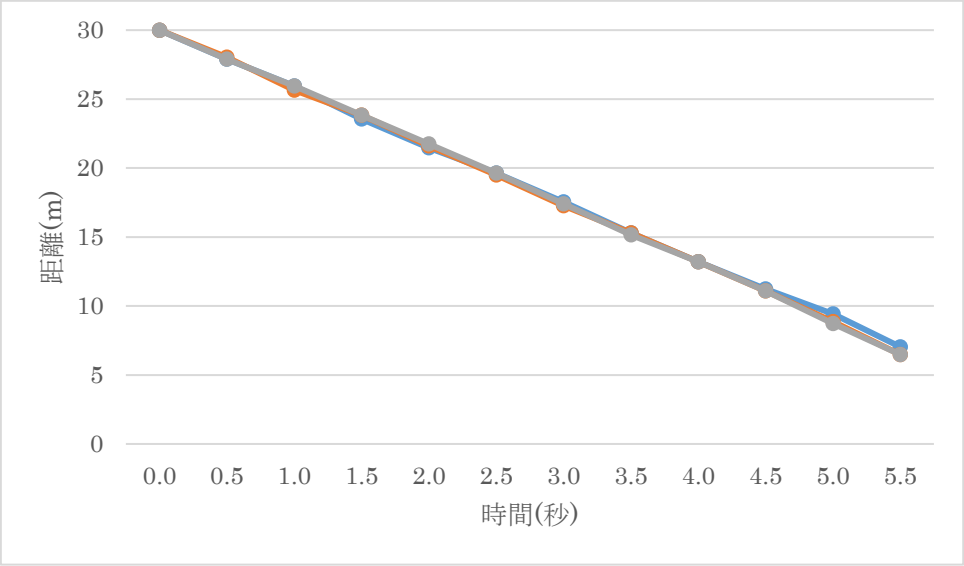

図 5.2.5 時速 15km/h 時の時速情報から計算した 2 点間の距離

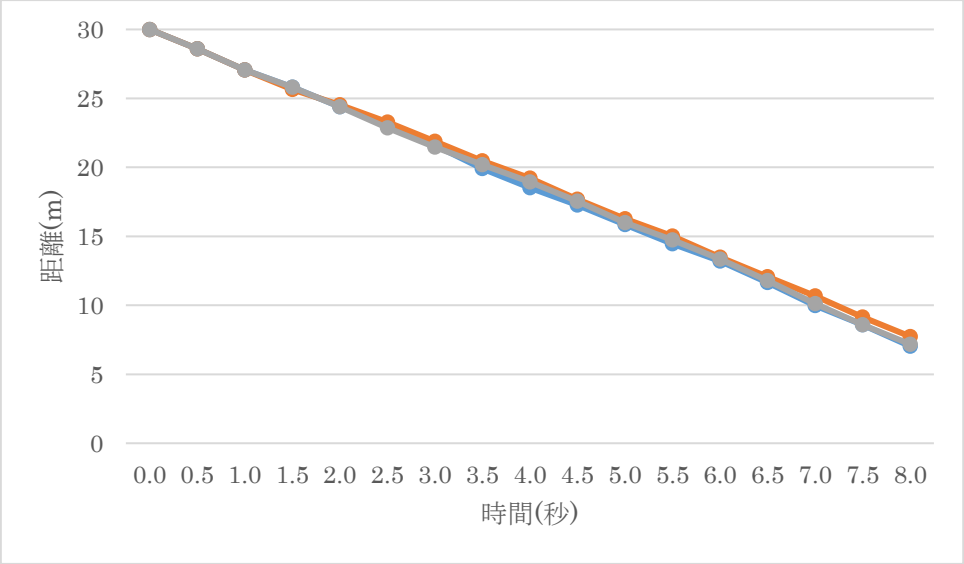

図 5.2.6 時速 10km/h 時の時速情報から計算した 2 点間の距離

5.3 考察

実験 1 の図 5.1.1 と図 5.1.2 から、自動車側の端末は、移動していないこともあり位置情報にバラつきが少ないことがわかる。これに比べて、図 5.1.3、図 5.1.4 の二輪車側の端末は移動しながらの測定であり、数値にバラつきが出た。これは、5 回全てにおいて同じ速度で接近することが難しいことや、アンドロイド端末の処理能力が移動速度に追いついていないことが考えられる。

次に、図 5.1.5 より、実際の距離と位置情報から算出した距離とでは、最大で約 7m の誤差が出てくることが分かった。また、2点間の距離が短いと誤差があまりなく、実際の距離に近しくなる。これから分かったことは、取得した位置情報において、緯度、経度が 1/10000 ずれるだけで、実際の距離と測定結果との間に数 m の誤差が生まれるということだ。普段、現在地を調べるたり、目的地を探したりする場合などであれば、この数 m の誤差はそれほど問題ではないが、車道にいる車両同士の接触を防止するという面においては、この数の誤差が致命的になってしまう。

実験 2 においては、図 5.2.1、図 5.2.2、図 5.2.3 から分かるように、実際の時速と測定した時速に大きな差は生まれなかった。そして、測定した時速と、車両は時速 1km/h で 0.5 秒間に約 0.14m 進むことを利用し、2 点間のだいたいの距離を予測した結果、図 5.2.4、図 5.2.5、図 5.2.6 のようになった。このことから、実験 2 の方法では自動車側から二輪車側の正確な位置を知ることはできないが、後方から二輪車が接近しているということは認識できるため、運転手への注意喚起として利用することができると考える。

第 6 章　結論

本研究で提案した実験 1 の、アンドロイド端末で位置情報を取得し、server を経由することで位置情報の緯度、経度から 2 点間の直線距離を求め、車両同士の接触事故を防止するという手法は、実際の距離と数 m の誤差は生まれたが、運転手に車両が接近していると認識することはできる、という結果となった。しかし、この数 m の誤差が生まれる可能性があることで、運転手は衝突防止システムを信頼することはできない。また、実験 2 では位置情報ではなく、時速情報を利用することで、運転手に正確な距離は分からないが、車両が接近していることを知らせるには十分に利用できる。という結果になった。

今後は、より端末に負担が掛からなアプリの開発、何 m 後方に車両が接近しているかなど、更に便利に利用できるようにすること、そして位置情報を取得する GPS アプリの精度を高め、実際の距離との誤差を 1m 未満にすることが、今後の課題である。

謝辞

本論文を作成するにあたり、ご指導頂きました三好力教授に深謝いたします。
また、多忙の中、日頃の議論にご協力くださった同三好研究室の皆様や、学友の皆様に心より感謝いたします。

参考文献

[1]車車間・路車間の通信システム普及の鍵はコンテンツ
https://bizgate.nikkei.co.jp/article/DGXMZO28438270220320180000000?page=2
[2][Android] GPS で位置情報を取得するアプリを作る
https://akira-watson.com/android/gps.html#3
[3]Apache2.4 系で Web サーバーを構築する
https://sanuki-tech.net/windows-server-2016/web-server-apache-2-4/install-php-7-2/
[4]交通事故統計（平成 30 年 3 月末）
https://www.npa.go.jp/news/release/2018/20180412001jiko03.html
[5]バイクは危険な乗り物なのか？事故率・死亡率を自動車と比較
http://bikenosusume.com/bike-failure-rate
[6]二輪車の死亡事故統計
http://www.keishicho.metro.tokyo.jp/kotsu/jikoboshi/nirinsha/2rin_jiko.html
[7]SUBARU
https://www.subaru.jp/impreza/impreza/safety/safety2.html
[8]クルマとバイクが“会話”すれば、事故は「3 分の 1」になる
https://wired.jp/2017/08/04/prevent-motorcycle-crashes-b2v/
[9]地図　2点間距離を測定する方法
https://qiita.com/a_nishimura/items/6c2642343c0af832acd4
[10]【Android】2 点間の距離を求める Location#distanceBetween
https://shirokai.hatenablog.com/entry/android-location-distancebetween
[11]2 点間の距離計算（C, Clojure, Go, Haskell, Java, LOGO, OCaml, Ruby）
https://qiita.com/niwasawa/items/5128101ef93a56e8a6af
[12]【MySQL, SQL】データベースを扱う基本 SQL 一覧
https://qiita.com/knife0125/items/bb095a85d1a5d3c8f706
[13] 2 地点の緯度と経度から距離を計算する
https://gist.github.com/nissuk/2360016#file-gistfile1-java
[14]20～30 代のスマホ利用率は 9 割を超え、全世代でも 7 割超え。
https://webtan.impress.co.jp/e/2017/08/02/26474
[15]安全な車間距離の目安と計算方法
https://www.kuruma-sateim.com/drive-technique/safe-distance/

付録

付録 A

Androidstudio ソース

GPS　manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bykzy_000.gpstest1">
    <uses-permission
android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATI
ON"/>
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOC
ATION"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".GPSTest1">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
            android:name=".LocationSenderService"
            android:enabled="true"
            android:exported="false">
        </service>
    </application>
</manifest>
```

付録 B

Androidstudio ソース

GPS　gpstest1

```
package com.example.bykzy_000.gpstest1;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class GPSTest1 extends Activity implements
OnClickListener {
	private Button btnSendStart;
	private Button btnSendStop;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_gpstest1);
		btnSendStop =
(Button)findViewById(R.id.btnSendStop);
		btnSendStart =
(Button)findViewById(R.id.btnSendStart);
		btnSendStop.setOnClickListener(this);
		btnSendStart.setOnClickListener(this);
	}
	@Override
	protected void onDestroy() {
		Intent locationSenderService = new
Intent(GPSTest1.this, LocationSenderService.class);
		stopService(locationSenderService);
		super.onDestroy();
	}
	@Override
	public void onClick(View view) {
		// TODO Auto-generated method stub
		if (view == btnSendStop) {
			Intent locationSenderService = new
Intent(GPSTest1.this, LocationSenderService.class);
			stopService(locationSenderService);
```

付録 C

Androidstudio ソース

距離　mainactivity.java

```
package com.example.bykzy_000.gpskyori;
import android.location.Location;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import static java.lang.Math.*;
public class MainActivity extends    AppCompatActivity {
        double lat1 =;
        double lng1 =;
        double lat2 =;
        double lng2 =;
        double r = 6378.137;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        float[] distance =
                getDistance(lat1, lng1, lat2, lng2);
        //distance[0]
        System.out.println(distance[0]);
    }
    /*
     * 2 点間の距離（メートル）、方位角（始点、終点）を取得
     * ※配列で返す[距離、始点から見た方位角、終点から見た方
位角]
     */
    public float[] getDistance(double x, double y, double x2,
double y2) {
        // 結果を格納するための配列を生成
        float[] results = new float[3];
        // 距離計算
        Location.distanceBetween(x, y, x2, y2, results);
        return results;
    }
}
                btnSendStop.setEnabled(false);
                btnSendStart.setEnabled(true);
            } else if (view == btnSendStart) {
                Intent locationSenderService = new
Intent(GPSTest1.this, LocationSenderService.class);
                startService(locationSenderService);
                btnSendStop.setEnabled(true);
                btnSendStart.setEnabled(false);
            }
    }
}
```

付録 D

Androidstudio ソース

距離　manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.bykzy_000.gpskyori">
      <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
            android:label="@string/app_name"
            android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
            android:theme="@style/AppTheme">
 <activity android:name=".MainActivity">
                  <intent-filter>
                        <action
android:name="android.intent.action.MAIN" />
                        <category
android:name="android.intent.category.LAUNCHER" />
                  </intent-filter>
            </activity>
      </application>
</manifest>
```

付録 E

Androidstudio ソース

距離　activity_main

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="274dp"
        android:layout_height="46dp"
        android:text="TextView"
        android:textSize="18sp"
        tools:layout_editor_absoluteX="16dp"
        tools:layout_editor_absoluteY="53dp" />
</android.support.constraint.ConstraintLayout>
```