令和 4 年度特別研究

高速道路での合流の支援システム

龍谷大学　理工学部情報メディア学科

T190528　平松 輝也

指導教員　　三好 力 教授

内容梗概

自動運転技術の発達により自動車の運転に対する不安やストレスは少なくなってきている。各社の運転支援技術の向上により本線を走っている際には、前方車両を追随し、自動でアクセルやブレーキの操作、ハンドル操作を行うことが可能となってきている。しかしながら、高速道路という速度の速い場所を運転する際には日常では感じえない不安やストレスがある。高速道路の運転において一番の不安要素は合流であるが、合流を自動運転で行うことは現状できていない。そこで、将来的には自動運転で合流ができることを目標としつつ、合流を支援できるシステムの開発に取り組んだ。

目次

1. はじめに.....1
1-1 研究背景.....1
1-2 自動運転について.....1
2. 既存技術.....3
2.1 運転支援システム.....3
2.2 SUBARU の運転支援システム［3］.....3
2.3 日産自動車の運転支援システム［4］.....5
2.4 メルセデスベンツの運転支援システム［5］.....6
2.5 テスラの運転支援システム［6］.....7
2.6 ETC システム［7］.....7
2.7 自動運転で合流を行う際の問題点.....8
3. 提案手法.....9
4. 実験.....10
4.1 前提条件.....10
4.2 事前実験.....10
4.3 実験手法.....12
4.3.1 シミュレーション環境.....15
4.4 実験結果.....17
4.4.1 実験 1.....17
4.4.2 実験 2.....19
5. まとめ.....21

謝辞
参考文献
付録

1. はじめに

1-1 研究背景

近年、自動運転の技術は目覚ましい発展を見せている。中でも車線変更をハンドルを握ることなく車線変更を行うことが実用レベルでできるようになった。高速道路を走行する上で苦手と感じる部分はどんなところかのアンケートを取った資料によると約 6 割のドライバーがジャンクションやインターチェンジなどの自動車が頻繁に合流する道と回答している［1]。しかし、現状自動運転で合流を行うことはできていない。そこで、自動運転によって高速道路の合流を行えるようにすることで、ドライバーの負担を軽減することができると考えた。本研究では、高速道路の合流支援システムを作成することでドライバーの負担を軽減することを目標とする。

1-2 自動運転について

自動運転とは、人が運転しなくても自動車が自動で設定された目的地に到着したり、道路上を走行したりすることを指す。

自動運転には 0～5 段階によるレベル分けがされている［2]。

レベル	名称	運転主体	走行領域
0	運転自動化なし	人	適用外
1	運転支援	人	限定的
2	部分運転自動化	人	限定的
3	条件付運転自動化	システム	限定的
4	高度運転自動化	システム	限定的
5	完全運転自動化	システム	限定なし

自動運転のレベル分け（参考：JSAE「運転自動化レベルの概要」）

図 1：自動運転レベル分け表

レベル 0 では、運転自動化されていない状態で、自動運転や運転支援の技術が何もない状態でドライバーが主体となって運転操作を行う。

レベル 1 では運転支援が搭載されており、システムがアクセル・ブレーキ操作またはハンドル操作のどちらかを部分的に行うことができるが、ドライバーが運転操作を行う。

レベル 2 ではレベル 1 よりも高度な運転支援システムが搭載され、アクセル・ブレーキ

の操作またはハンドル操作の両方を部分的に行うことができる。運転はドライバーが主体となって運転操作を行う。

レベル 3 では条件付で自動運転を行うことができる。決められた条件下で、全ての運転操作を自動化。ただし運転自動化システムが作動中も、システムからの要請でドライバーはいつでも運転に戻れなければならない。システムが作動中はシステムが主体となって運転操作を行う。

レベル 4 では自動運転を行うことができる。決められた条件下で、全ての運転操作を自動化し、決められた条件下であればドライバーは何もしなくてよい。システムが主体となって運転操作を行う。

レベル 5 では完全自動運転を行うことができる。条件なく、全ての運転操作を自動化することができ、ドライバーは何もしなくてよい。システムが運転操作を行う。

現在、自動運転はレベル 3 まで実現している。レベル 3 では条件付きで手を放し、アクセル・ブレーキを操作することなくシステムが運転操作を行ってくれる。しかし、システムが自動運転を維持できなくなるとドライバーが運転操作を行わなければいけない。現状、自車からのセンサやカメラ情報をもとに条件を決定している。しかし、高速道路の合流では風防・騒音対策の壁や本線と合流車線の並走区間が短いことで情報を取得することが難しく、高速道路の合流が実現していないと考えらえれる。

自動運転に必要な技術に、自動車の制御、周囲の状況を把握することの 2 点が考えられる。自動運転で合流を行う際に問題となるのが周囲の状況を把握する部分である。現在の運転支援システムでは自動車自身がカメラやセンサを搭載し、情報を取得している。しかし、高速道路の合流では高速道路の本線と合流車線が離れており、情報を取得することが難しいため自動運転で合流を行うことができていないと考える。

2. 既存技術

2.1 運転支援システム

運転支援システムについて自動車メーカーが各社それぞれの独自技術を用いて研究などを行っておりここ数年で大きく成長してきている。

運転支援システムには様々なシステムが存在しており、飛び出してきた人を検知したり、前方の状況からブレーキが必要な際に緊急ブレーキを作動させるプリクラッシュブレーキや高速道路上などで前方の車をアクセル操作をしなくても追従していくクルーズコントロール。道路上の白線などをカメラから得た情報をもとに画像処理を行い、白線を認識することで車線を逸脱したときには警報を鳴らしたり、ハンドル操作を行うレーンキープ機能など運転者の負担が少なくなるように様々なシステムが存在している。

自動車メーカーごとに様々な名前で呼ばれており、トヨタ自動車ではトヨタセーフティセンスと言う名前で呼ばれている。

2.2 SUBARU の運転支援システム［3］

スバルではアイサイトと呼ばれる運転支援システムを自動車に搭載している。

アイサイト X と呼ばれる SUBARU 最高峰の運転支援システムは、アイサイトの主要デバイスであるステレオカメラの刷新を筆頭に、4 つのレーダーによる 360°センシングの実現、さらに 3D 高精度地図データや GPS＋準天頂衛星を利用したことでより高度な運転支援を実現している。

アイサイトが優れている点はセンサとカメラ両方使っているのに加え 3D 高精度地図データや GPS＋準天頂衛星を利用しているところだ。

・プリクラッシュブレーキ

プリクラッシュブレーキとは、クルマに取り付けられたセンサが障害物などを感知して、衝突を防止する安全装置。

主に先行車への追突の回避支援を行うプリクラッシュブレーキ。追突の危険がある時は、警報や軽いブレーキで注意を促し、万一の時は 追突回避のための強い自動（被害軽減）ブレーキを作動させる。しかし、様々な条件によりシステムが十分作動しない場合もある。

・後退時ブレーキアシスト

後退時ブレーキアシスト（RAB=Reverse Automatic Braking）は、後退時に壁や障害物に衝突するおそれがあるとシステムが判断した場合、マルチファンクションディスプレイやマルチインフォメーションディスプレイの表示とブザーで知らせ、必要に応じて自動（被害軽減）ブレーキによって衝突を回避、または被害を軽減する。

・AT 誤発進抑制制御・AT 誤後進抑制制御

発進時にペダルやギヤの操作を間違えて前（後ろ）の壁に衝突！そんなまさかの急発進を抑止する AT 誤発進抑制制御（AT 誤後進抑制制御）。

・アイサイト・ツーリングアシスト

アイサイト・ツーリングアシストは、高速道路などでのすべての車速域で、アクセル、ブレーキ、ステアリング操作を自動でアシスト。しかし、完全な自動運転ではなく、あくまでも運転支援システム。

・全車速追従機能付クルーズコントロール

全車速追従機能付クルーズコントロールは、高速道路や自動車専用道路でドライバーの運転の疲れを軽減し、快適なドライブをもたらす。しかしあらゆる状況を判断し制御を行うものではない。あくまでドライバーをサポートする運転支援システム。

・アクティブレーンキープ

アクティブレーンキープは、高速道路など自動車専用道路での走行時、ステレオカメラで道路の白線を認識。操舵制御することで、車線中央の維持や車線逸脱の抑制を行う運転支援システム。

・警報＆お知らせ機能

クルマがふらついたり、車線からはみ出しそうになったり、先行車の発進に気づかなかったり…。そんなとき、注意を促してくれる警報＆お知らせ機能。

図 2：アイサイトのステレオカメラシステム

2.3 日産自動車の運転支援システム［4］

日産自動車では

「手を放すという、新しい運転のカタチ」をキャッチコピーにハンズオフドライブができるようになっており、高速道路の運転でクルマが車速や車線維持をアシストすることでハンドルから手を放したまま、 ドライブが楽しむことができる。

「前のクルマ、 追い越しましょうか?」がキャッチコピーの車線変更追い越し支援では、前方車の速度に応じて、システムが追い越しを提案。ハンドルに手を添えてスイッチを押すだけで、車線変更をアシストすることができる。

「加減速を、どこまでもなめらかに」では、高速域(最大 120km/h)でも作動し、下り坂ではスピードをキープ。 停車しても、またスタートすることができる。アクセルもブレーキも、クルマがサポートすることができる。

「カーブもお手のもの」では、走行レーンを認識しキープするから、曲がり心地がスムースに。直線だけでなく、カーブでもハンドリングをアシストしてくれる。どんな道でも、クルマが運転をサポートすることができる。

「標識の場所も、 インストール済み」では、自動車に道路情報をインストールすることで自動車が道路情報を知り尽くしたパートナーになる。ナビリンク機能で標識やカーブ、ジャンクションの位置を把握し、道路形状をもとに加減速をアシストすることができる。

先行車追従停止・停止保持とは

先行車との距離を保つように、自動でアクセルとブレーキをコントロール。渋滞のときも、先行車に合わせてブレーキをキープする。

ハンドル支援とは

車線の中央付近を走行するようにステアリングを制御し、ドライバーのハンドル操作を支援する。

ナビリンク機能とは

制限速度に応じた設定速度の切り替え カーブの大きさに合わせた減速をナビ情報と連動してシステムが支援する。

ナビ連動ルート走行とは

ナビをセットしたルートに沿って、追い越しや分岐など、システムが高速道路の出口まで走行をアシストし、3D 高精度地図データと連携することで、さらに正確なドライブを楽しむことができる。

360 度センシングとは

7 個のカメラの映像、5 個のレーダー、12 個のソナーで、道路の白線、標識、周辺車両を検知する。

図 3 : 360 度センシングのイメージ図

インテリジェントインターフェースとは

360 度センシングの情報やプロパイロット 20 の作動状態などの情報を、メ ーター部分とヘッドアップのディスプレイでわかりやすく表示する。

2.4 メルセデスベンツの運転支援システム［5］

メルセデスベンツではインテリジェントドライブと呼ばれるシステムを搭載している。メルセデスベンツではクラスと呼ばれる種類分けを行っている。その中で、E クラス ステーションワゴンは、前方約 250m、側方約 40m、後方約 80m を検知するレーダー、約 90m の 3D 視を含む約 500m 前方をカバーするステレオカメラという最先端のセンサーシステムを搭載している。

ステアリングアシストを備えたアクティブディスタンスアシスト・ディストロニックは、高速道路はもちろん街中や郊外でも、ステアリングに手を添えているだけで、前走車との車間を維持しながら、走行中の車線を走り続けることができる性能を獲得するなど、現状自動車に搭載されている運転支援システムの中ではメルセデスベンツのインテリジェントドライブとテスラのオートパイロットが一番自動運転に近いと考えている。

図 4 : インテリジェントドライブのイメージ図

2.5 テスラの運転支援システム［6］

テスラではオートパイロットと呼ばれるシステムを搭載しており、8 台のカメラと強力なビジョン処理により、360 度の視界と、最長 250m まで先を視認できるようになっている。

8 台のカメラの役割は下の図に示す。

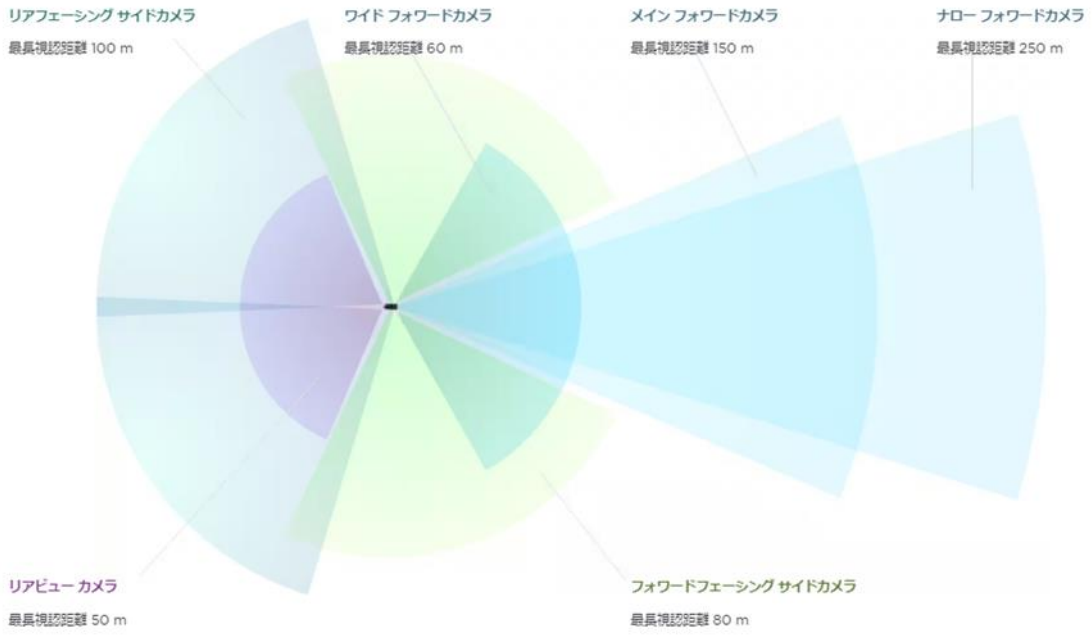

図 5：オートパイロットのセンサ・カメラシステム

このように各社独自のシステムを搭載している。その中でも共通点にカメラとセンサを用いて周囲の状況を把握する点が挙げられる。このカメラとセンサで得た情報が自動運転を行う上でとても重要になっている。

2.6　ETC システム［7］

ETC とは、英語で Electronic Toll Collection System の略で、高速道路や有料道路の料金所ゲートで、自動車や自動二輪に搭載した車載器と無線通信を行い、車種や通行区間を判別して認証や決済を行うシステム。現在日本では高速道路の利用者の約 90%が利用しているとされている。また、ETC2.0 と呼ばれる新しいシステムでは、ETC システムと、道路情報などのサービスを提供する ITS スポットサービスなど複数のサービスを、ひとつの情報基盤の上に乗せて提供するシステムの 2 つを利用することができる。

2.7　自動運転で合流を行う際の問題点

合流を行う際、自動車からのセンサやカメラ情報だけで合流するには、状況を把握するのに時間がかかってしまい、安全に合流するのには専用の車線を作らなければいけないなどの問題点がある。しかし、高速道路の本線を走っている際に、車線変更を行うことができるため自動運転の技術的にはできてもおかしくないと考えられる。また、運転支援システムについて、各社様々なシステムを使っているところが問題となっている。各社が競い合うことで技術が向上していくということを考えると問題点として言えないかもしれないが、各社の技術によって運転支援のレベルに差が生まれるところも問題点と考えられる。

3. 提案手法

問題点として挙げた、自動車にセンサやカメラを搭載し状況を取得する手法では、合流する際に必要な本線上の自動車の台数や車間距離等の情報を取得するために、本線と並走しなければならない。この問題点を解決するために、本線上にセンサやカメラを設置することで情報を取得できるようにする。この情報を自動車に伝達することで運転支援システムに情報を伝達することができるようになる。これにより、情報取得の遅れによる運転支援で合流ができない問題点は解決される。しかし、ここで各社様々な運転支援システムを開発しているところから生まれるシステムの違いによってセンサ情報等の取得方法に違いが生じることが問題となる。この問題点は既存技術でも解決することはできると考える。ETC を活用することだ。既存技術の挙げたように ETC には ETC 2.0 と呼ばれる、ETC システムと、道路情報などのサービスを提供する ITS スポットサービスなど複数のサービスを、ひとつの情報基盤の上に乗せて提供するシステムがある。ここで言う ITS スポットサービスとは IST スポットで都市圏の高速道路の渋滞データが丸ごと配信され、対応カーナビで最速ルートを逐一判断できるようになるサービスのことだ。この ETC 2.0 では様々な情報を得ることができるようになっているシステムを活用することで本線上の自動車の台数や速度などの様々な情報を受信することができるようにすることができると考える。ETC2.0 で得られる情報に付け加えた本線上の他車の速度等の情報を利用することで各社の運転支援システムの違いから生じる情報取得方法等の違いを解決することができると考えた。これにより、自動運転で合流ができていない問題点を解決することができると考える。

本研究では、本線を走行する自動車の車速や車間距離等の情報をもとに合流地点の状況を予測することで、合流を行う車両が安全に合流できるように支援を行うシステムを開発することを目標とする。支援の手法として本線の情報から合流地点での車間距離等を計算で求め、事前実験をもとに安全に合流できるかを予測し、画像を用いて警告を行うことで安全に合流を行うことができるように支援する。本システムでは本線を走行する自動車が 1 台の時と 2 台の時の 2 パターンに対応する。合流車両が決められた時間(今回は２０秒)で合流地点に到達する地点で情報の取得を行う。本線を走行する自動車の情報(車速や車間距離)から合流車両が合流地点での本線を走行する自動車の状況を計算を用いて予測する。この予測した本線を走行する自動車の位置情報を合流車両が取得する情報として渡す。センサが情報を取得してから合流車両が情報を取得する時間についても予測を立てる計算式に組み込まれている。予測データをもとに合流が安全に行えるかを検証する。本線を走行する自動車が 2 台の時、上記と同様に予測を行い、車間距離から 1 台目の前方に合流するのか 1 台目と 2 台目の間に合流するのか 2 台目の後に合流するのかを計算によって予測する。

4. 実験

4.1 前提条件

本実験では以下の前提を設け、これらのパラメータをシステムの前提としてシミュレーションを行った。

本線は片側一車線の道路。

車速は常に一定。

本線を走る車の速度および車間距離をセンサを用いて情報を取得。

情報の受信には ETC を活用。

合流する自動車がセンサ情報を受け取る場所は合流地点まで 20 秒かかる距離と仮定。

4.2 事前実験

まず事前準備として時速 50 ㎞から 120 ㎞までの間で 100m 進むのに必要な時間を求めた。図 6 に示す通り時速 50 ㎞では 7.2 秒、時速 80 ㎞では 4.5 秒、時速 90 ㎞では 4 秒、時速 100 ㎞では 3.6 秒、120 ㎞では 3 秒と、とても緩やかな曲線を描く。

時速 50 ㎞からにした理由は高速道路の最低制限速度であるためだ。また、時速 100 ㎞までは規制等が無ければ出してもよいとされている。そして、時速 120 ㎞は日本の高速道路の一部の区間で制限速度に設定されている。

図 7 では時速 50 ㎞から時速 120 ㎞までの間で 1 秒・2 秒・3 秒・4 秒・5 秒のそれぞれの秒数で進む距離(m)を表す。

この事前実験で 100m 進むのに必要な時間(秒数)を求めた結果を実験でセンサ 1 とセンサ 2 の間を通過する時間を決定する際の参考資料として用いた。また、センサ 1 とセンサ 2 の間を通過する時間を時速 120 ㎞の時 3 秒かかることから、センサ 1 とセンサ 2 の間を通過する時間の最速値を 3 秒に設定することで時速 120 ㎞を上限時速とすることができる。

時速 50 ㎞から時速 120 ㎞までの間で 1 秒・2 秒・3 秒・4 秒・5 秒のそれぞれの秒数で進む距離(m)を求めることで、時速と時間の関係性を知ることができ、時間が長くなった際の予測に役立つと考えた。

この図 6 と図 7 で求めることができた数値をもとに、実験を行う際の時間や合流地点までの距離の決定を行う。

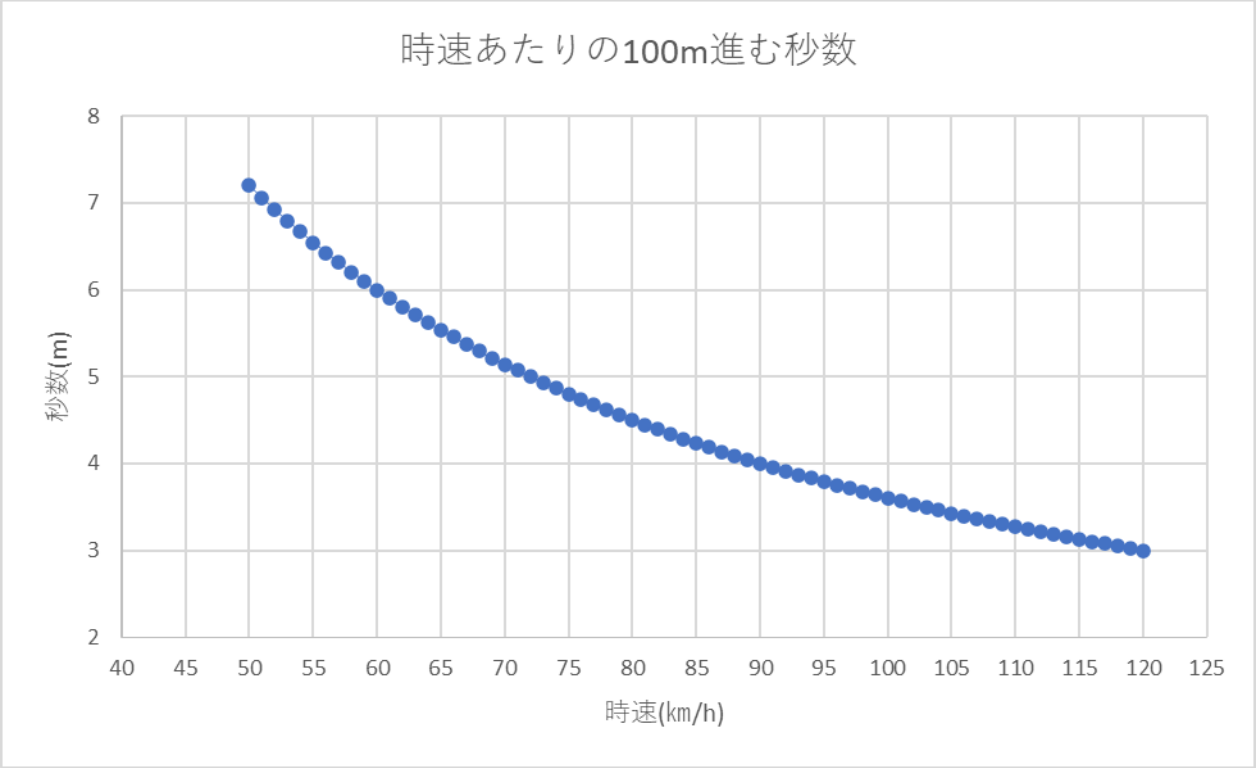

図 6 100m 進むときにかかる秒数

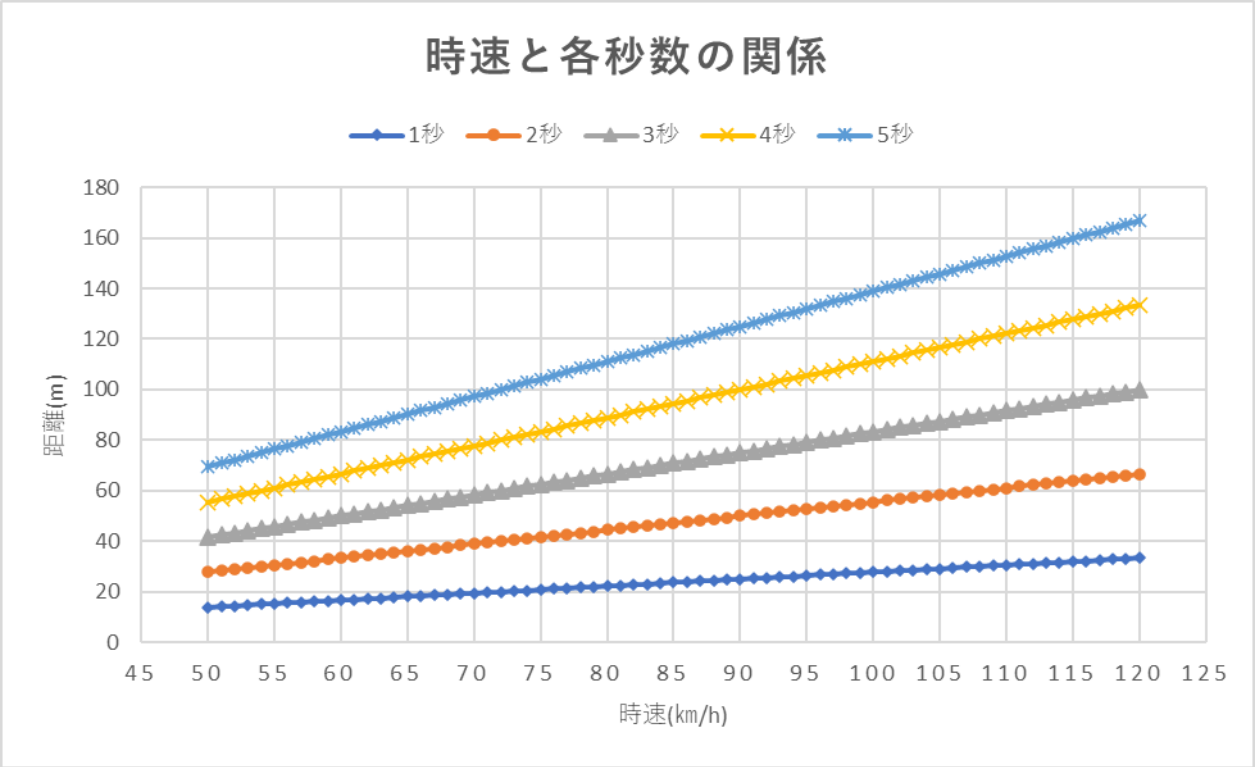

図 7 各秒あたりの時速と距離の関係

センサ 2 の位置を決める際、図 7 を参考に計算を行った結果、時速 120 ㎞の時、20 秒進むと 666.67m 進むことが分かった。これにより、センサ 1 を合流地点の手前 800m とセンサ 2 を合流地点の手前 700m の地点に設定することで、センサ情報を取得するタイミングと本線上の自動車がセンサを通過したときのタイミングが同じだった時、時速 120 ㎞の場合に合流地点で自動車同士がほぼ重なる計算になる。

次に 2 台の車間距離を測るため、2 台の自動車がセンサ 2 の上を通る間の時間を計測することで車間距離を求めることができる。ここで前提条件で本線は片側一車線の道路と仮定しているため後方車両が前方車両に追いつく場合、前方車両と同じ時速となり車間距離 2 秒のところで追従するとする。

速度	車間時間	車間距離
40km/h	2秒以上	22.2m以上
60km/h	2秒以上	33.3m以上
80km/h	3秒以上	66.7m以上
100km/h	3秒以上	83.3m以上

図 8 : 車間距離の目安［8］

図 8 の車間時間を見ると、高速道路上で安全に走行するためには前方車両から 3 秒が必要だと考えられる。また、距離が離れすぎることで割り込みの危険性も指摘されている。割り込みの言い方を変えると合流にもなるこのことから 2 台の車間距離が 6 秒あれば安全に前後に 3 秒ずつゆとりを持った合流ができると考えた。

4.3 実験手法

上記の記載の秒数は図 8 より実験者が仮定した値となっているため、検証を行った。

合流地点での車間距離等の状況を予測するために計算を行った結果から仮定した数値において画像等が正しく表示されるかの確認を行うために検証を行った。

実験では本線上を走る自動車が 100m 進む時間から時速を求める。時速を求めた本線上を走る自動車が決められた距離(本研究では 700m)を進む時間を求めることで、合流地点の交通状況を予測することができる。また、2 台の自動車がセンサ 2 を通過する時間を求めることで合流地点付近での車間距離を予測することができると考えた。

以下の値は事前実験をもとに決めた値。

まず、センサ 1 とセンサ 2 の間は 100m とする。

前提条件の合流地点まで 20 秒かかる距離から図 7 を参考にすることで、センサ 1 とセ

ンサ 2 の位置を決定する。センサ 1 は合流地点手前 800m に設定し、センサ 2 を合流地点手前 700m に設定する。

本線上を走る自動車がセンサ 1 からセンサ 2 までを通過する時間を測ることで時速を求める。

センサ 1 からセンサ 2 を通過する秒数は図 6 を参考に最速値を 3 秒としランダムで求める。

センサ 2 を本線上の自動車が通過してからセンサ情報を取得するまでの時間を計測しておくことで、本線上を走る自動車が合流地点に到達する時間を計算することができる。

本線上の自動車が通過してからセンサ情報を取得するまでの時間はランダムで求める。

取得した情報を計算式にあてはめる。

【100(m)÷1000÷走行時間(秒)×60×60】　(1.1)

（1.1）の式ではセンサ間の時速を求めることができる。

ここで求められた時速は平均時速として考えるものとする。

【700(m)÷(1.1 の結果)（時速）÷1000×60×60】　(1.2)

走行時間は 2 つのセンサ間を通る時間とする。

（1.2）の式では合流地点までにかかる時間を計算することができる。

【(1.2 の結果)－A(20 秒)－B】　(1.3)

(1.3)の式で A は合流車両が合流地点まで行く時間。B はセンサ 2 が取得した本線上の自動車が通過してからセンサ情報を取得するまでの時間。とする。

上記の式(1.1)(1.2)(1.3)は実験 2 においても使用する。

実験 2 では車間距離についての計算も行った。

この時、2 台の本線を走行する自動車が合流地点に到達する時間の差を車間距離としたため、秒数で車間距離が表示されている。

実験 1 実験 2 ともに計算結果から結果を出力する。

結果については・危険・注意・安全の三種類に分類する。

出力された結果をディスプレイ上に表示させる。

画像はそれぞれ図 9 から図 11 のものを表示させる。

図 9：合流が安全な時

図 10：合流するのに注意しなければいけない時

図 11：合流が危険な時

画像は約 20 秒表示させます。

20 秒にした理由は、情報を取得してから合流地点まで 20 秒と仮定していることから 20 秒とした。

また、それぞれの画像を表示させる条件は図 8 を参考に本線を走る自動車が合流地点から前後 2 秒以内だと危険を表示させる。2 秒から 6 秒の間は注意を表示させる。6 秒以上であれば安全を表示させる。

以下の Python プログラムで計算を行う。
変数 s にはセンサ 1 を通過してからセンサ 2 を通過するまでの時間を代入する。
変数 sa にはセンサ 2 を通過してから合流車両が情報を取得するまでの時間を代入する。
変数 ts は本線を走行する自動車が合流地点に到達する時間を示す。

```
jisoku = (100/1000/s*60*60)
print(jisoku)

jikan = (700/jisoku/1000*60*60)
print(jikan)

ts = (jikan-20-sa)
ts = abs(ts)
print(ts)
```

また、以下の Python プログラムは画像を表示させるために使用する。

```
if ts < 3:
        print("危険")

        start_time = time.time()
        while True:
            img = cv2.imread("sample1.jpg")
            img = cv2.resize(img,(1200,800))
            cv2.imshow("sample1", img)
            cv2.waitKey(100)



            end_time = time.time()

            if end_time - start_time >= 20:
                cv2.destroyAllWindows()

                break
```

4.3.1 シミュレーション環境

シミュレーションは Raspberry pi 400 を使用し以下の環境で行った。

表 1： 計算機の環境構成

CPU	Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
OS	Linux 11 (bullseye)
RAM	4GB LPDDR4-3200
Raspberry pi	400

また、画像を表示させるため OpenCV をインストールした。

4.4 実験結果

4.4.1 実験 1

実験 1 では本線上の自動車が 1 台の時、センサ 1 からセンサ 2 までの時間で式(1.1)を用いて時速を求める。求めた時速を式(1.2)に代入し、合流地点までの時間を求める。式(1.2)で求めた時間を式(1.3)に代入する。20 秒は合流車両が合流地点まで到達する時間である。今回、センサ 1 からセンサ 2 までの時間を 4.2 秒、センサ 2 を本線上の自動車が通過してから合流車両が情報取得するまでの時間を 3 秒として実験を行った。

結果----------
センサ 1 からセンサ 2 までの時間
4.2
センサ 2 を通過してから情報取得までの時間
3
85.71428571428571
29.400000000000002
6.400000000000002
安全

結果より図 9 が表示された。

このように、100m 進む時間を計測することで時速を求め、その時速から合流地点までの時間を求めることができるようになった。よって、本線を走る自動車が一台だけだった場合ではこの結果で合流地点の状況をある程度予測することができる。しかし、本線を走る自動車が 1 台しかない状況はほとんどない。よって、車間距離情報が必要となってくる。

本線を走行する自動車がセンサを通過してから合流車両が情報を取得する時間が 2 秒の時に本線を走行する自動車が合流地点まで到達する時間を表 2 に示す。

表 2：取得時間が 2 秒の時

時速 90 ㎞	6 秒
時速 93 ㎞	5.1 秒
時速 100 ㎞	3.2 秒
時速 105 ㎞	2 秒

表 3：安全を表示させる回数

6 秒の時	10 回中 2 回
5 秒の時	10 回中 3 回

表 4：危険を表示させる回数

2 秒以内の時	10 回中 1 回
3 秒未満の時	10 回中 3 回

ここで実験手法では安全を表示させるのは 6 秒以上としたが、表 3 より 5 秒以上であっても安全と表示させる回数に大きな違いがなかったため 5 秒以上に変更が妥当だと分かった。安全を表示させる秒数を 6 秒で設定したとき、10 回中 2 回表示された。5 秒の時には 10 回中 3 回表示された。よって今回の実験では、安全を表示させる秒数において 6 秒と 5 秒の時に大きな差はないと判断した。また、表 2 を見ても時速に大きな差がないことが分かる。危険と注意を表示させる値についても 2 秒以内で危険を表示させることになっていたが、要 4 から 2 秒以内では危険と表示させる回数が少なくなってしまい、注意と表示させる回数が増えてしまうことから 3 秒未満の変更が妥当だと分かった。これについても同様に 2 秒の時と 3 秒の時それぞれ 10 回検証を行った。結果、2 秒の時には 10 回中 1 回危険が表示され、5 回注意が表示された。3 秒の時は 10 回中 3 回危険が表示され、注意も 3 回表示された。表 2 からわかるように今回の前提条件下で本線を走る自動車が時速 100 ㎞で、ギリギリ注意が表示され、時速 100 ㎞を上回ると危険が表示されることになった。時速 100 ㎞は高速道路の一般的な最高速度であることから危険を表示させる値は 3 秒未満の方が妥当だと考えた。

ここで、すべての数値を半分にして実験を行ってみた。センサ 2 から合流地点までの距離を 350m、合流車両が合流地点まで 10 秒で到達することができるとした。合流車両が情報取得するまでに本線を走る自動車がセンサを通過してから 1 秒経過している時に時速 90 ㎞以上でも危険が表示されてしまうことが分かったため、実験状況によって安全や危険を表示させている合流地点までの時間設定を見直す必要があると感じた。

4.4.2 実験 2

実験 2 では車間距離も求められるようにした。まず、実験 1 で行った手法で 1 台目と 2 台目の本線上を走る自動車の合流地点までの時間等を求めた。そして、車間距離を求めるために 2 台目の合流地点までの時間から 1 台目の合流地点までの時間を引くことで車間距離を求めることができ、式を(1.4)のように表す。

【2 台目－1 台目】　　(1.4)

今回の実験では、1 台目のセンサ 1 からセンサ 2 までの時間を 3.9 秒、センサ 2 を本線上の自動車が通過してから合流車両が情報取得するまでの時間を 8 秒。2 台目はセンサ 1 からセンサ 2 までの時間を 4.5 秒、センサ 2 を本線上の自動車が通過してから合流車両が情報取得するまでの時間を 2 秒として実験を行った。

結果----------
センサ 1 からセンサ 2 までの時間
3.9
センサ 2 を通過してから情報取得までの時間
8

センサ 1 からセンサ 2 までの時間
4.5
センサ 2 を通過してから情報取得までの時間
2
92.3076923076923
27.3
-0.6999999999999993
80.00000000000001
31.49999999999999
9.49999999999999
10.199999999999989
危険

結果より図 11 が表示された。

10.1999 は 1 台目と 2 台目の車間距離を表している。車間距離が安全であっても危険と表示されているのは、1 台目の自動車と合流車両が合流地点で重なってしまっているからで、安全と表示させるには 1 台目も 2 台目も合流地点から±5 秒以上かつ車間距離も 5 秒以上必要といったとても厳しい条件になってしまった。しかし、安定した結果を出すためには条件を厳しくしなければいけなかった。

この問題点を解決するために、図 9 から図 11 までの画像に車間距離を表示させることにした。これにより、車間距離を把握することができるようになった。

また、高速道路には凍結や濃霧等で制限速度が低くならない限り、時速 50 ㎞の最低制限速度が設けられていることから、時速 50 ㎞を下回って走行している時には渋滞等の高速道路の混雑が予想されるため、時速 50 ㎞を下回っている際には図 12 のような画像を表示させることで速度を保った状態で混雑しているか、渋滞等の止まってしまうような道路状況かを区別できるようにした。

図 12：渋滞等の時

5. まとめ

今回、提案したシステムでは本線を走っている自動車の時速をセンサ情報から計算し、合流地点の状況を予測することで安全に合流できるかを画像で表した。また、車間距離を計算によって予測することで、ドライバーの負担を軽減することができたと考えられる。しかし、本線を走る自動車の速度が一定であると仮定していたため実際に使用するには難しいと考えられ、予防的な役割しか持たせることができなかった。よって、センサをさらに増やすことでより正確に予測することができる考えられる。また、本システムに加え、合流車両が存在していることを本線を走る自動車に通知することで合流車両と本線を走行する自動車の両方ともの安全性を向上させることができると考えた。

本研究の発展として合流車両の情報取得を一回ではなく複数回取得できるようにすることで精度の高い情報を得ることができると考えられる。また、車間距離等についても本線を走る自動車の時速が早い場合、合流車両が本線を走る自動車のすぐ後ろに合流したとしても事故が起こりにくいと予想できる、精度の高い情報を得ることができるようになれば車間距離等の値についても再考すべきだと考える。また、本線を走行する自動車の台数が増えた時に値や計算方法などを再考しなければならないと考えた。

表示方法の改善にも取り組むことができると考えられる。ヘッドマウントディスプレイのような AR 表示を行うことでスマートフォンやカーナビゲーションの小さなディスプレイだけではなくサイドウィンドウやフロントウィンドウの下部に表示することができるようになり、視覚情報を増やせるようになると共に、視線移動から生まれる事故を減らすことができると考えた。

謝辞

本研究を進めるにあたりご指導頂いた卒業論文指導教員の三好 力教授に感謝いたします。また、日常の議論を通じご協力いただいた三好研究室の皆様や学友に感謝いたします。

参考資料

［1］ドライバーの 4 人に 1 人が高速道路は「苦手」と感じている
https://kurukura.jp/safety/191206-80.html

［2］自動運転のレベル分けとは？レベル 0〜5 までを一挙解説
https://www.macnica.co.jp/business/maas/columns/135343/

［3］アイサイト X｜ 運転支援システム アイサイトオーナーズサポート
https://www.subaru.jp/eyesightowner/eyesight-X/

［4］【日産自動車】 ProPILOT - プロパイロット公式ページ
https://www2.nissan.co.jp/BRAND/PROPILOT/?sclisid=LS_TIJ_99_GO_GLIS_PC_00494885&gclid=CjwKCAjwve2TBhByEiwAaktM1B5P5T27ZJLuzloCAALCT2VIFMWc-MuOuGreSAMk4EJHv44JdXk0YBoCpsMQAvD_BwE

［5］セーフティ ｜The E-Class Stationwagon - メルセデス・ベンツ(メルセデスベンツ公式サイト)
https://www.mercedes-benz.co.jp/passengercars/mercedes-benz-cars/models/e-class/e-class-estate/safety/model-year-update-s213.module.html

［6］オートパイロット ｜ テスラジャパン - Tesla
https://www.tesla.com/jp/autopilot

［7］ETC の意味とは。登録・利用方法や高速の ETC 割引の使い方
https://www.zurich.co.jp/car/useful/guide/cc-what-is-etc/

［8］走行中の適切な車間距離は？ ｜JAF クルマ何でも質問箱
https://jaf.or.jp/common/kuruma-qa/category-drive/subcategory-technique/faq138

［9］【徹底比較】運転支援システム比較！どのメーカーのシステムが一番いい？
https://www.ancar.jp/channel/15566/

付録

実験 1 のプログラム

```
import cv2
import time

def judge( s,sa,):
    jisoku = (100/1000/s*60*60)
    print(jisoku)

    jikan = (700/jisoku/1000*60*60)
    print(jikan)

    ts = (jikan-20-sa)
    ts = abs(ts)
    print(ts)

    if jisoku <= 50:
        print("渋滞")

        start_time = time.time()
        while True:
            img = cv2.imread("sample4.jpg")
            img = cv2.resize(img,(1200,800))
            cv2.imshow("sample4", img)
            cv2.waitKey(100)

            end_time = time.time()

            if end_time - start_time >= 20:
                cv2.destroyAllWindows()

                break

    else:

        if ts < 3:
            print("危険")

            start_time = time.time()
            while True:
                img = cv2.imread("sample1.jpg")
                img = cv2.resize(img,(1200,800))
                cv2.imshow("sample1", img)
                cv2.waitKey(100)


                end_time = time.time()

                if end_time - start_time >= 20:
                    cv2.destroyAllWindows()

                    break

        elif 3 <= ts and ts <= 5:
            print("注意")

            start_time = time.time()
            while True:
                img = cv2.imread("sample2.jpg")
                img = cv2.resize(img,(1200,800))
                cv2.imshow("sample2", img)
                cv2.waitKey(100)

                end_time = time.time()

                if end_time - start_time >= 20:
                    cv2.destroyAllWindows()

                    break

        elif ts > 5:
            print("安全")

            start_time = time.time()
            while True:
                img = cv2.imread("sample3.jpg")
                img = cv2.resize(img,(1200,800))
                cv2.imshow("sample3", img)
                cv2.waitKey(100)


                end_time = time.time()

                if end_time - start_time >= 20:
                    cv2.destroyAllWindows()

                    break

s = input("mae センサ 1 からセンサ 2 までの時間¥n¥n")
s = float(s)

sa = input("mae センサ 2 を通過してから情報取得までの時間¥n¥n")
sa = float(sa)

judge( s,sa)
```

実験 2 のプログラム

```
import cv2
import time

def judge( s,sa,s1,sa1):
    jisoku = (100/1000/s*60*60)
    print(jisoku)

    jikan = (700/jisoku/1000*60*60)
    print(jikan)

    ts1 = (jikan-20-sa)
    print(ts1)

    jisoku1 = (100/1000/s1*60*60)
    print(jisoku1)

    jikan1 = (700/jisoku1/1000*60*60)
    print(jikan1)

    ts2 = (jikan1-20-sa1)
    print(ts2)

    ts = (ts2-ts1)

    if ts <= 1:
        ts = 2

    ts1 = abs(ts1)
    print(ts1)

    ts2 = abs(ts2)
    print(ts2)

    ts = abs(ts)
    print(ts)

    if jisoku <= 50 or jisoku1 <= 50:
        print("渋滞")

        start_time = time.time()
        while True:
            img = cv2.imread("sample4.jpg")
            img = cv2.resize(img,(1200,800))
            cv2.imshow("sample4", img)
            cv2.waitKey(100)

            end_time = time.time()

            if end_time - start_time >= 20:
                cv2.destroyAllWindows()

                break

    else:

        if ts1 < 3:
            print("jisoku:{:.2f}".format(jisoku))

            print("危険")

            start_time = time.time()
            while True:
                img = cv2.imread("sample1.jpg")
                img = cv2.resize(img,(1200,800))
cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                cv2.imshow("sample1", img)
                cv2.waitKey(100)


                end_time = time.time()

                if end_time - start_time >= 20:
                    cv2.destroyAllWindows()

                    break



        elif 3 <= ts1 and ts1 <= 5:
            print("jisoku:{:.2f}".format(jisoku))

            if ts2 < 3:
                print("jisoku2:{:.2f}".format(jisoku1))

                print("危険")

                start_time = time.time()
                while True:
                    img = cv2.imread("sample1.jpg")
                    img = cv2.resize(img,(1200,800))
cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                    cv2.imshow("sample1", img)
                    cv2.waitKey(100)

                    end_time = time.time()

                    if end_time - start_time >= 20:
                        cv2.destroyAllWindows()

                        break


            elif 3 <= ts2 and ts2 <= 5:
                print("jisoku2:{:.2f}".format(jisoku1))

                if ts < 3:
                    print("危険")
```

```
                    start_time = time.time()
                    while True:
                        img = cv2.imread("sample1.jpg")
                        img = cv2.resize(img,(1200,800))

cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                        cv2.imshow("sample1", img)
                        cv2.waitKey(100)


                        end_time = time.time()

                        if end_time - start_time >= 20:
                            cv2.destroyAllWindows()

                            break



        elif   ts1 > 5:
            print("jisoku:{:.2f}".format(jisoku))

            if ts2 < 3:
                print("jisoku2:{:.2f}".format(jisoku1))

                print("危険")

                start_time = time.time()
                while True:
                    img = cv2.imread("sample1.jpg")
                    img = cv2.resize(img,(1200,800))

cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                    cv2.imshow("sample1", img)
                    cv2.waitKey(100)

                    end_time = time.time()

                    if end_time - start_time >= 20:
                        cv2.destroyAllWindows()

                        break


            elif 3 <= ts2 and ts2 <= 5:
                print("jisoku2:{:.2f}".format(jisoku1))

                if ts < 3:
                    print("危険")

                    start_time = time.time()
                    while True:
                        img = cv2.imread("sample1.jpg")
                        img = cv2.resize(img,(1200,800))

cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                        cv2.imshow("sample1", img)
                        cv2.waitKey(100)


                        end_time = time.time()

                        if end_time - start_time >= 20:
                            cv2.destroyAllWindows()

                            break

                elif 3 <= ts and ts <= 5:
                    print("注意")

                    start_time = time.time()
                    while True:
                        img = cv2.imread("sample2.jpg")
                        img = cv2.resize(img,(1200,800))

cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                        cv2.imshow("sample2", img)
                        cv2.waitKey(100)

                        end_time = time.time()

                        if end_time - start_time >= 20:
                            cv2.destroyAllWindows()

                            break


            elif   ts2 > 5:
                print("jisoku2:{:.2f}".format(jisoku1))

                if ts < 3:
                    print("危険")

                    start_time = time.time()
                    while True:
                        img = cv2.imread("sample1.jpg")
                        img = cv2.resize(img,(1200,800))

cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                        cv2.imshow("sample1", img)
                        cv2.waitKey(100)


                        end_time = time.time()

                        if end_time - start_time >= 20:
                            cv2.destroyAllWindows()

                            break

                elif 3 <= ts and ts <= 5:
                    print("注意")

                    start_time = time.time()
                    while True:
                        img = cv2.imread("sample2.jpg")
                        img = cv2.resize(img,(1200,800))

cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                        cv2.imshow("sample2", img)
                        cv2.waitKey(100)

                        end_time = time.time()

                        if end_time - start_time >= 20:
                            cv2.destroyAllWindows()

                            break

                elif ts > 5:
                    print("安全")

                    start_time = time.time()
                    while True:
                        img = cv2.imread("sample3.jpg")
                        img = cv2.resize(img,(1200,800))

cv2.putText(img,"{:.2f}".format(ts),(400,650),cv2.FONT_HERSHEY_SIMP
LEX,5,(0,0,0),8)
                        cv2.imshow("sample3", img)
                        cv2.waitKey(100)

                        end_time = time.time()

                        if end_time - start_time >= 20:
                            cv2.destroyAllWindows()

                            break

s = input("mae センサ 1 からセンサ 2 までの時間¥n¥n")
s = float(s)

sa = input("mae センサ 2 を通過してから情報取得までの時間¥n¥n")
sa = float(sa)

s1 = input("ato センサ 1 からセンサ 2 までの時間¥n¥n")
s1 = float(s1)

sa1 = input("ato センサ 2 を通過してから情報取得までの時間¥n¥n")
sa1 = float(sa1)

judge( s,sa,s1,sa1)
```