Open SDV Initiative

Open SDV API
Vehicle.Motion

バージョン：202509α
発行日：2025年9月30日

Open SDV Initiative

ドキュメントの位置付け

ドキュメントの目的

- このドキュメントは，Open SDV Initiativeで作成を進めている Open SDV API の内，車両運動に対するAPIについて説明するものである

ドキュメントの完成度

- 現時点で，車両運動に対するAPIの基本的な設計ができているが，このAPIで車両を安全・円滑に制御できることは示せておらず，POC等を通じた評価が必要である
- APIの細部の設計が完了しておらず，未決定事項が残っている

変更履歴

バージョン	発行日	備考
202503α	2025年3月31日	初版
202509α	2025年9月30日	

バージョン 202503aとの主な違い

規定の詳細化

- 名称と略称，位置付けと機能，機能クラス，リスククラス，構成情報，状態，サービスコール（一覧），イベント，データ型に関する規定を追加

縦方向制御APIの変更

- 目標速度を指定して縦方向を制御する2種類のサービスコールを統合
  - 応答プロファイルに「最速」を追加
- 緊急ブレーキには，停止制御サービスコールを使用するように変更
  - 停止プロファイルに「緊急」を追加

縦方向制御APIの変更

- 横方向制御の2種類のサービスコールを統合
- 横方向制御サービスコール(1)に対しては，パラメータで指定する目標位置の解釈を変更
- 横方向制御サービスコール(2)に対しては，パラメータ名を変更
  - 第1目標位置→目標位置
  - 第2目標位置→次目標位置
- 目標位置に到達し，次目標位置が設定されていない場合の振る舞いを，曲率を維持するという仕様に

車両運動に対するAPI

名称と略称

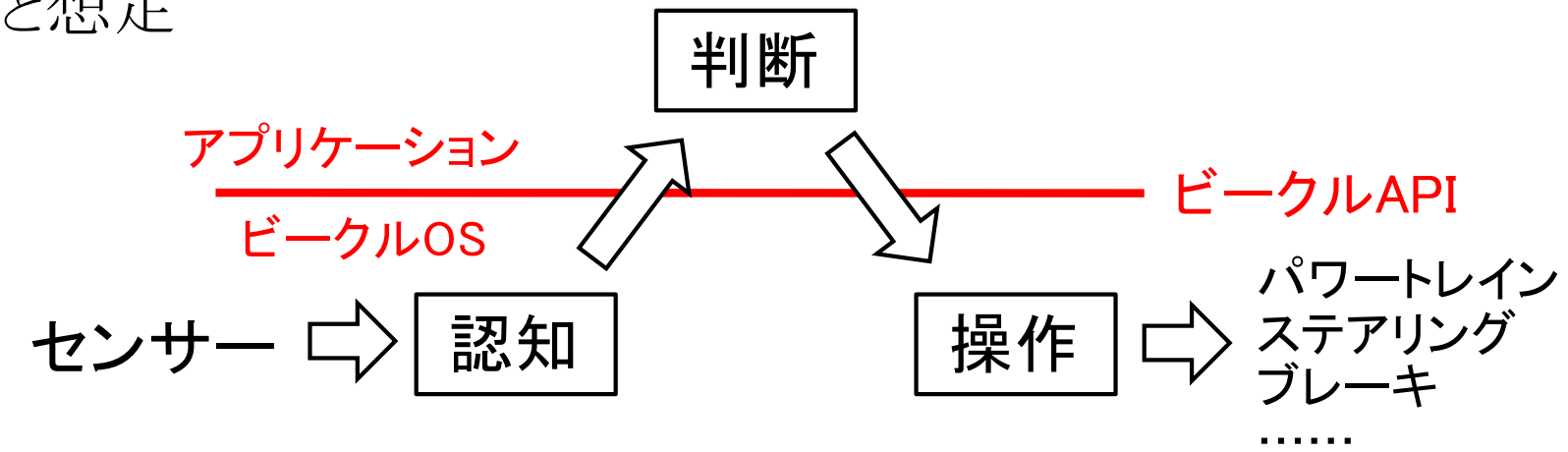
- 名称(車両の構成記述での表記):Vehicle.Motion
- 略称:Motion

設計目標

- 達成できる制御品質の低下を最小限にしつつ，車両の違いに依存せず，サービスコールの呼び出し周期を長くできるAPIとすることを目標とする

想定するアプリケーション

- 本API仕様を検討するにあたり，以下の機能をアプリケーションとして実現することを想定
  - ACC（Adaptive Cruise Control）
  - LKAS（Lane Keeping Assist Systems）
  - 自動運転
  - 自動駐車
- AD/ADASの「認知」「判断」「操作」の3要素の内，「認知」と「操作」をビークルOS，「判断」をアプリケーションの役割と想定



位置付けと機能

- Motionは，車両の移動を操作するためのシングルトンオブジェクトである
  - 車両運動の状態を参照することにより，車両の現在の速度や加速度を取得することができる
  - 車両運動を制御することにより，車両のパワートレイン，ブレーキ，ステアリングなどを制御することができる
- 縦方向制御と横方向制御のそれぞれに対して，標準的なロック機能（LockableObject と同等）を持つ
  - 例えば，緊急ブレーキは，発動時に縦方向制御をロックする
  - 縦方向制御と横方向制御のサービスコールは，オプションで，操作優先度を指定することができる

機能クラス

- Motionは，以下の機能クラスを持つ
  - CONTROL_LONGITUDINAL … APIにより縦方向制御ができる
  - CONTROL_LATERAL … APIにより横方向制御ができる

リスククラス

- Motionは，以下のリスククラスを持つ
  - RISK_COLLISION … 自車の責任で何らかの物体（他の車両，歩行者，障害物，落下物など）に衝突するリスク
  - RISK_GROUNDLOSS … 脱輪や落下のリスク
  - RISK_VIOLATION … 交通規則を違反するリスク
- RISK_COLLISIONをさらに分類するかは今後の検討課題
  - ODDや走行速度によって分類する案がある
- この他にも，横転のリスクなどが考えられるが，車両側（ビークルOS）で対策できると思われる

構成情報

- capabilities … 車両が備えている機能クラスの集合
- riskClasses … 車両で対策できていないリスククラスの集合
- TODO : 車両の運動性能に関する情報を含める。今の時点でわかっている構成情報は以下の通り
  - 速度制御サービスコールの「最速」以外の各応答プロファイルにおける最大加速度・最大減速度
  - 停止制御サービスコールの「緊急」以外の各停止プロファイルにおける最大減速度
  - 停止制御サービスコールの「緊急」以外の各停止プロファイルにおける10km/h（10km/hが良いか要検討）から停止できる距離（最悪ケース）

状態

- speed … 現在速度(km/h)
- acceleration …現在加速度
  - longitudinal … 縦方向加速度(単位:m/s^2)
  - lateral … 横方向加速度(単位: m/s^2)
  - vertical … 鉛直方向加速度(単位: m/s^2)
- angularVelocity …現在回転速度
  - roll … ロール(前後軸まわりの回転速度, 単位:度/s)
  - pitch … ピッチ(左右軸まわりの回転速度, 単位:度/s)
  - yaw … ヨー(鉛直軸まわりの回転速度, 単位:度/s)

※ 以上の名称と単位はVSSに合わせたが, それで良いか要検討

状態 – 続き

- longitudinalLockStatus … 縦方向制御のロック状態
- lateralLockStatus … 横方向制御のロック状態
- 縦方向制御の目標値
  - longitudinalControlStatus … 縦方向制御の状態(速度制御中, 停止制御中, 速度維持中, 故障中)
  - targetSpeed … 目標速度(km/h)
  - responseProfile … 応答プロファイル
  - stopProfile … 停止プロファイル
- 横方向制御の目標値
  - lateralControlStatus … 横方向制御の状態(サービスコール(1～2)による制御中, 曲率維持中, 故障中)
  - targetLocation … 目標位置
  - nextTargetLocation … 目標位置

サービスコール(一覧)

- 構成参照機能
- 状態参照機能
- 縦方向制御機能(2種類)
  - 速度制御サービスコール
  - 停止制御サービスコール
- 横方向制御機能
  - 横方向制御サービスコール
- 縦方向制御のロック機能(2種類)
- 横方向制御のロック機能(2種類)
- イベント機能(3種類)

縦方向制御のサービスコール(概要)

- コアビークルAPIには，目標速度を指定して縦方向を制御する速度制御サービスコールと，目標停止位置を指定して縦方向を制御する停止制御サービスコールを用意する
  - 目標速度や停止位置の指定は，車両の違いに依存せず，最大でも数十Hz程度で設定すれば十分と考えられる
  - それより先の制御(例えば，加速度・駆動力・制動力の算出・適用)はビークルOS内で行う

縦方向制御のサービスコール(概要) – 続き

- 対応できていないリスククラスがある場合の振る舞いは以下の通り
  - 私有地を走行していることが判定できる車両では，RISK_VIOLATION に対応できていなくても，私有地を走行している場合には，操作を受け付ける
  - 上記以外の場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる
- 前車との目標車間距離を指定して縦方向を制御するサービスコールは，必要であれば拡張ビークルAPIで用意する

速度制御サービスコール

- 目標速度 v と応答プロファイル p を指定して，縦方向を制御する
  - 応答プロファイルは，「最速」，「高速」，「標準」，「低速」の4段階のいずれかで指定する
  - 応答プロファイルによって定まる加減速度制限とジャーク制限をかける
  - ただし，「最速」は加減速度制限をかけない（言い換えると，車両の最大性能で加減速する）。ジャーク制限はかける
  - TODO：応答プロファイルを拡張・カスタマイズする方法を用意するかは，今後の課題とする

速度制御サービスコール – 続き

- ビークルOSは，応答プロファイルに従って，目標速度になるように制御する
  - ただし，ビークルOSは，アプリケーションから指定される目標速度の更新周期（不定期更新を含む）や変化幅によらず制御の連続性や安定性が確保できるよう，ビークルOS内の制御器の特性を考慮したフィルタをかける
  - 車両の性能の制約によりそれができない場合には，可能な限りそれに近づくように制御する
- 目標速度に達した後は，その速度を維持するように制御を継続する

停止制御サービスコール

- 目標停止位置 x と停止プロファイル p を指定して，縦方向を制御する
  - 目標停止位置は，現在位置からの走行距離で指定する
  - 停止プロファイルは，「緊急」，「早さ重視」，「バランス」，「精度重視」の4段階のいずれかで指定する
  - 停止プロファイルによって定まる減速度制限とジャーク制限をかける
  - ただし，「緊急」は，減速度制限，ジャーク制限をかけない
  - TODO：停止プロファイルを拡張・カスタマイズする方法を用意するかは，今後の課題とする

停止制御サービスコール－続き

- ビークルOSは，停止プロファイルに従って，目標停止位置で停止するように制御する
  - サービスコールが呼び出されるとすぐに減速を開始するとは限らず，減速が必要になるまでは現在の速度を維持する
- 目標停止位置で停止できない場合の振る舞いは以下の通り
  - 停止プロファイルが「緊急」の場合には，最も短い距離で停止するように制御する
  - それ以外の場合には，エラーとする
- 緊急ブレーキは，停止プロファイルを「緊急」として，このサービスコールを利用する

横方向制御のサービスコール（概要）

- コアビークルAPIには，目標位置を指定して横方向を制御するサービスコールを用意する
  - 目標位置の指定は，車両の違いに依存せず，最大でも数十Hz程度で設定すれば十分と考えられる
  - それより先の制御（例えば，実舵角の算出・適用）はビークルOS内で行う
- 対応できていないリスククラスがある場合の振る舞いは以下の通り
  - 私有地を走行していることが判定できる車両では，RISK_VIOLATION に対応できていなくても，私有地を走行している場合には，操作を受け付ける
  - 上記以外の場合には，E_RISK_APPLICATIONエラーまたはE_RISK_USERエラーとなる

横方向制御のサービスコール(概要) – 続き

- 実舵角を指定して横方向を制御するサービスコールは設けない
  - 実舵角による指定では，車両の特性を知らないと制御できないため
- 自動駐車で使用するためのより高レベルなAPIは，必要であれば拡張ビークルAPIで用意する
  - 例えば，目標停止位置と目標停止位置における車両の向きや実舵角，最大速度，障害物の位置を指定して，縦方向と横方向の両方を制御するサービスコール

横方向制御サービスコール（詳細）

- 目標位置（x, y）と次目標位置（xn, yn）を指定して，横方向を制御する
  - 目標位置と次目標位置は，車両との相対座標で指定する
  - ビークルOSは，サービスコールが呼び出された時点で目標位置を対地座標に変換して保持し，車両が移動しても目標位置と次目標位置は移動しない
  - 次目標位置の指定は省略できる

横方向制御サービスコール(詳細) ― 続き

- ビークルOSは，目標位置に到達するように制御する
  - ただし，ビークルOSは，アプリケーションから指定される目標位置の更新周期(不定期更新を含む)や変化幅によらず制御の連続性や安定性が確保できるよう，ビークルOS内の制御器の特性を考慮したフィルタをかける
  - 低速走行時には，例えば，1つのクロソイド曲線(曲率の変化率が一定の曲線)に沿って走行するように制御する
  - 次目標位置は，サービスコールの呼び出しの遅延や，目標位置での操舵の連続性に配慮するための情報であり，その使用方法はビークルOSに委ねられる

横方向制御サービスコール(詳細) ― 続き

- 目標位置に到達すると，次目標位置が目標位置になり，次目標位置は設定されていない状態になる
- 目標位置に到達し，次目標位置が設定されていない場合，目標位置に到達した時点での曲率を維持する
- 到達できない目標位置を指定した場合には，エラーとする
  - 回転速度(回転半径)や回転加速度(曲率の変化率)

横方向制御サービスコール（詳細） ― 続き

- アプリケーションは，このサービスコールを，以下のいずれかの方法で使うことを想定している

(1) このサービスコールを周期的に呼び出して，目標位置に到達する前に目標位置を更新する

- LKASなど，主に高速走行時の使用方法（低速走行時に使用できないわけではない）

(2) 必ず次目標位置を指定してサービスコールを呼び出す。目標位置に到達し，次目標位置が設定されていない状態になった場合には，サービスコールを呼び出して，次目標位置を指定する

- 自動駐車や交差点での右左折など，主に低速走行時の使用方法（高速走行時に使用できないわけではない）

イベント（検討途中）

- 縦方向制御の状態の変化
  - 目標速度に到達して速度維持に遷移
  - 他のアプリケーションが制御
  - 故障検出
  - 故障復帰
  - リスクを検出／安全機能が動作？
- 横方向制御の状態の変化
  - 上と同様
- 縦方向制御のロックに関するイベント（2種類）
- 横方向制御のロックに関するイベント（2種類）

データ型の定義

- 速度，加速度，回転速度

```
type SpeedType = Float32;

type AccelerationType = struct {
    longitudinal : Float32,
    lateral : Float32,
    vertical : Float32
};

type AngularVelocityTupe = struct {
    roll : Float32,
    pitch : Float32,
    yaw : Float32
}
```

  - フィールドの名称，データ型，単位は，VSSに合わせたが，それで良いか要検討

データ型の定義 – 続き

- 構成情報

```
type MotionCapabilityType = enum {
    CONTROL_LONGITUDINAL,
    CONTROL_LATERAL
};

type MotionRiskType = enum {
    RISK_COLLISION,
    RISK_GROUNDLOSS,
    RISK_VIOLATION
};

type MotionConfigType = struct {
    capabilities : enumSet(MotionCapabilityType),
    riskClasses: enumSet(MotionRiskType),
    .........
}
```

データ型の定義 – 続き

- 状態

```
type MotionStatusType = struct {
    speed : SpeedType,
    acceleration : AccelerationType,
    angularVelocity : AngularVelocityType,
    targetSpeed : SpeedType,
    .........
};
```