1.1. 可視化プログラムの前提条件

■ 本報告書では、デマンドコンセッション交通運行プログラム（以降、「本プログラム」と呼ぶ）を用いた、乗客輸送シミュレーションの可視化プログラム(以下、「可視化プログラム」という)のアルゴリズムを報告する。

■ アルゴリズムの実施前提条件は以下の通りである

番号	項目	検証目的
1	前提ソフト	OSSを利用する
2	リアルタイム性	1FPS(Frame per Second)単位の表示を条件とする
3	スケーラビリティ	オブジェクトの同時表示数の上限は1万とする
4	表示エリア	制限なし。地球上の全ての地区で表示できること
5	対応機能	(1)シミュレーンプログラムと同時連動、(2)csv,dbからの読み出し可能、(3)スマホ等のデバイスを使った仮想/現実連動機能

1.2. 前提OSSについて

- 前提とするOSS
  PruneMobileを採用
  (https://github.com/TomoichiEbata/PruneMobile)
- PruneMobleとは
  複数の人間や自動車等の移動体のリアルタイムの位置情報を、地図上に表示する、PruneCluster(https://github.com/SINTEF-9012/PruneCluster)のアプリケーションである
- PruneClusterとは
  地図上で、複数(百万個規模も対応可能)の座標マーカーを、見やすくまとめてリアルタイム表示することができる。ブラウザ上（クライアント側）で動くライブラリとして提供されている。
  デモサンプルは以下の通り
  ・デモ①（10人のリアルタイム移動） http://sintef-9012.github.io/PruneCluster/examples/moving.10.html
  ・デモ②（100万人のリアルタイム移動） http://sintef-9012.github.io/PruneCluster/examples/random.150000.html
- PruneMobileは、前ページのリアルタイム性、スケーラビリティ、表示エリアの全ての条件を見たす機能を有している

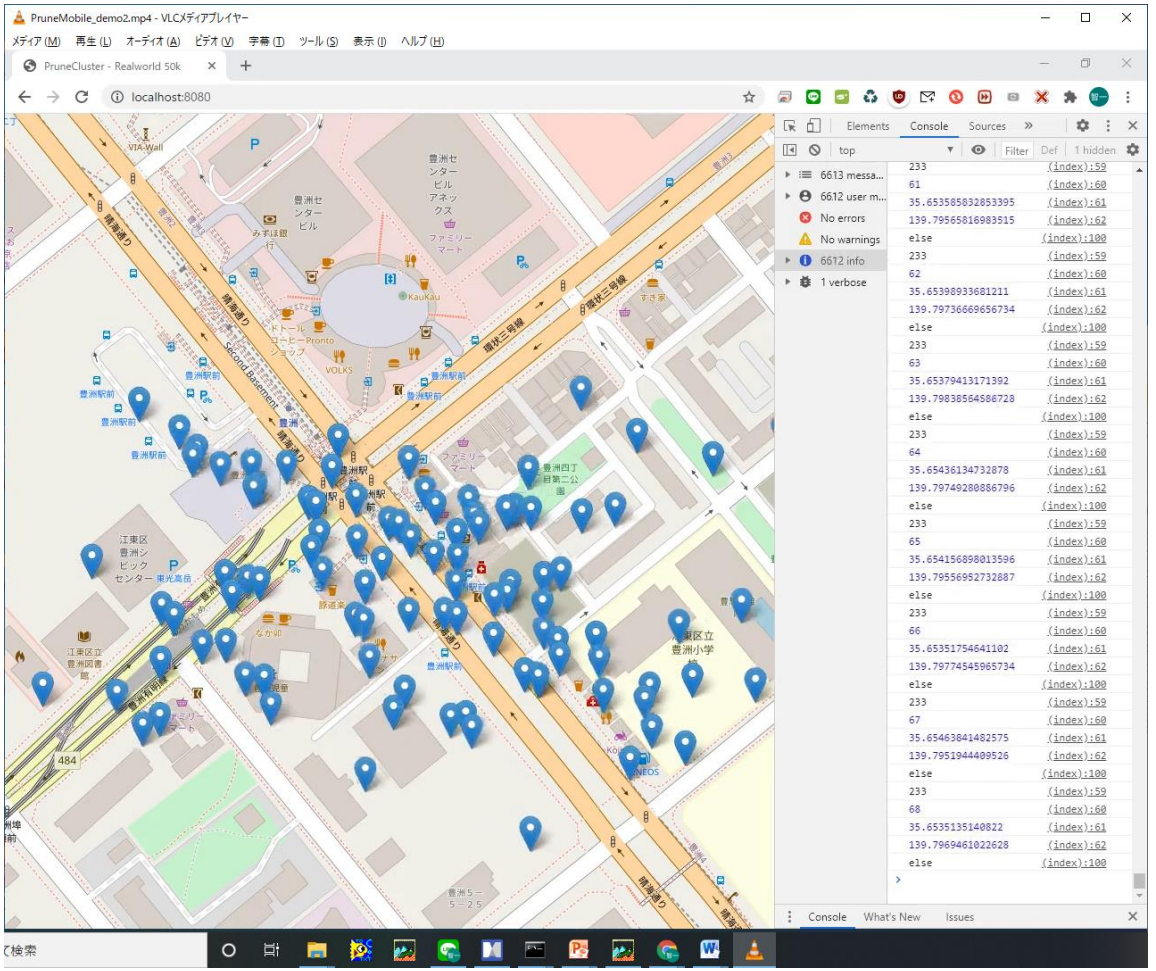
1.3. PruneMobileの概要

(1)複数の人間や自動車等の移動体のリアルタイムの位置情報を、地図上に表示する、PruneCluster (https://github.com/SINTEF-9012/PruneCluster) のアプリケーション

(2)PruneMobileに対して、任意のタイミングで位置情報(JSON形式)を送り込むだけで、地図上にマーカーが表示される

(3)シミュレーションと実世界のスマホ等を同時に表示することができる

(4)Go言語(Golang)で実装されている。数万以上の並列表示を実現する為である



1.4. 可視化プログラムの要求仕様

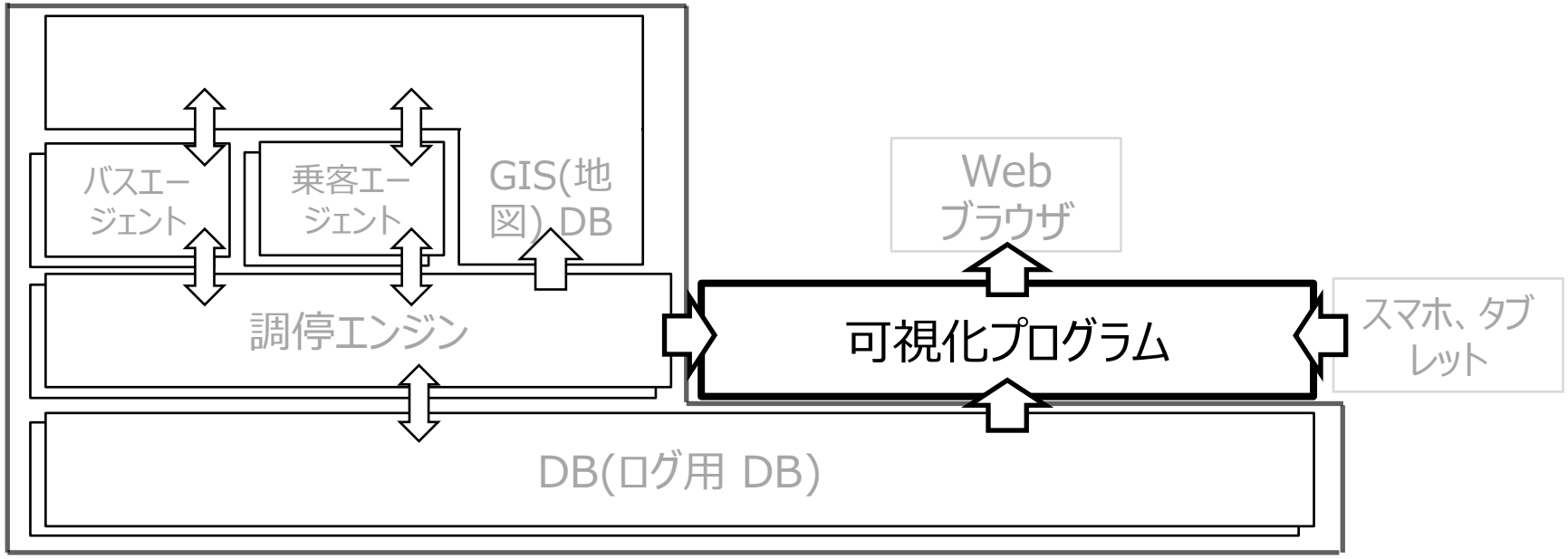
大量のオブジェクトをリアルタイムで遅延なく表示できること

項目	内容	考慮
(1)エリアサイズ	最大30～100km^{2}程度の地図が表示できること	柏の葉～豊洲案件の網羅範囲
	地図のズームイン、ズームアウトがストレスなく行えること	
(2)同時表示移動オブジェクト	同時300～1000程度のオブジェクトを表示可能であること	柏の葉案件を参考
(3)移動オブジェクトの種類	最大10程度(乗客、バス、電車)が、色違いの点で表示されれば良い	オブジェクトのアイコン化までは必要ない
(4)最小ステップ時間	1秒	1秒以下の表示は必要ない
(5)利用できる地図データ形式	Open Street Map(OSM)データが使えることが必須	国土地理院データ等(鉄道等)が使えればさらに良い
(6)表示方法	オブジェクト指定子に対して、データを書き込むタイミングで、オブジェクトの位置が変化すること	新規の書き込みがないデータは、再描画しない(オーバヘッドを回避)
	DBやメモリの存在を前提としない。	
(7)オブジェクトの発生と消滅	明示的にオブジェクトを生成した時から表示され、明示的にオブジェクトを消滅した時に表示がなくなる	表示のタイミングは、プログラム等で行う。ビューアは愚直に表示に徹する
(8)その他	オブジェクトの表示中に、色の変更やブリンク等ができれば望ましい	これは必須ではない。

2.1. 可視化プログラムの位置付け

1.目的 デマンドコンセッション(需給調停)の開発物を前提とした、可視化プログラムのアルゴリズムと実装

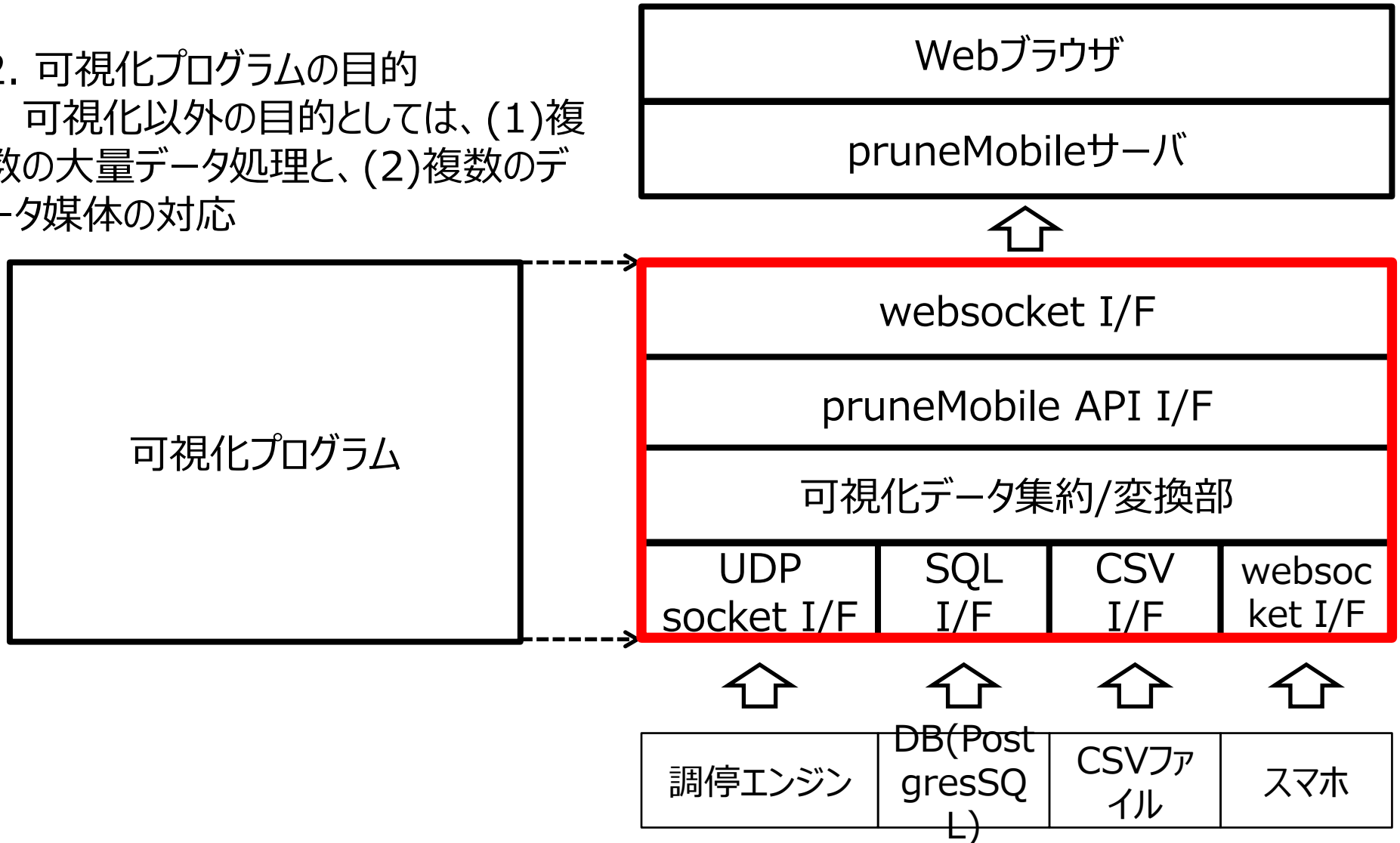
2. 開発対象



2.2. 可視化プログラムコア部の構成

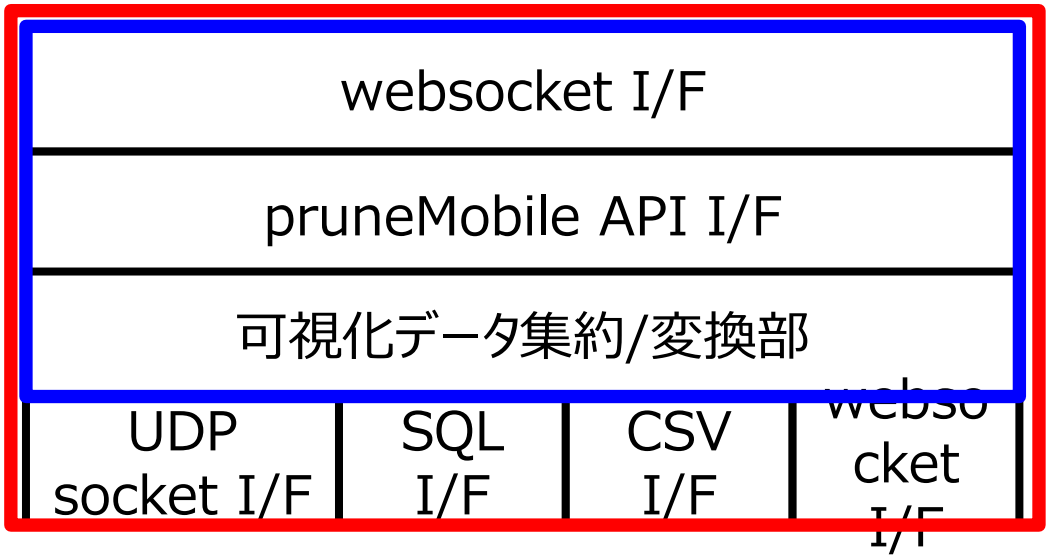
1.開発対象 **赤字の範囲**

2. 可視化プログラムの目的
　可視化以外の目的としては、(1)複数の大量データ処理と、(2)複数のデータ媒体の対応

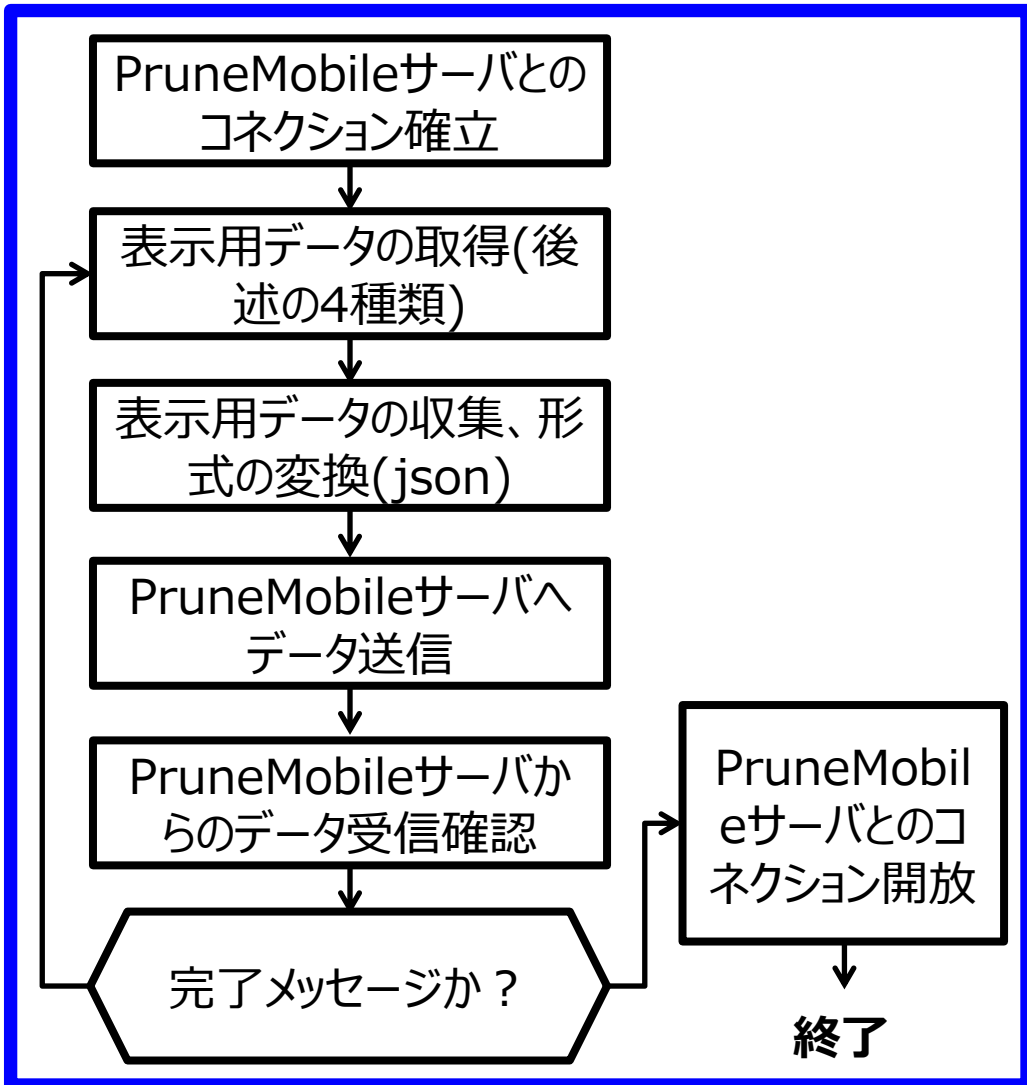


2.3. 可視化プログラムコア部アルゴリズム

(a)プログラムスタック



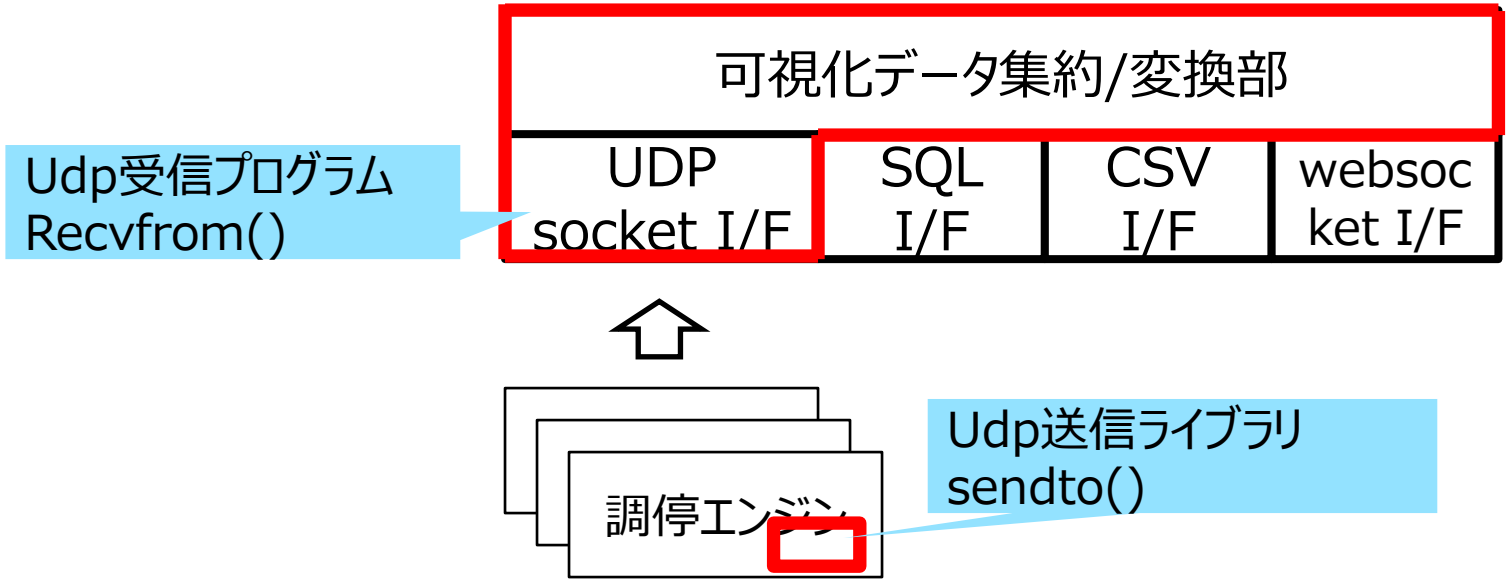
(b)フローチャート



2.3.1 可視化プログラム データI/F —— プログラム組み込みタイプ

1.シミュレータの中に通信用I/Fを組み込んで、シミュレーション結果をリアルタイムに取り出す。通信方式にはUDPを採用した。採用理由は(1)データの高速転送が可能であることと、(2)どのプログラムでも実装が用意であること、(3)描画用に逐次データ送信を行うため、データロストを認容できること、である。

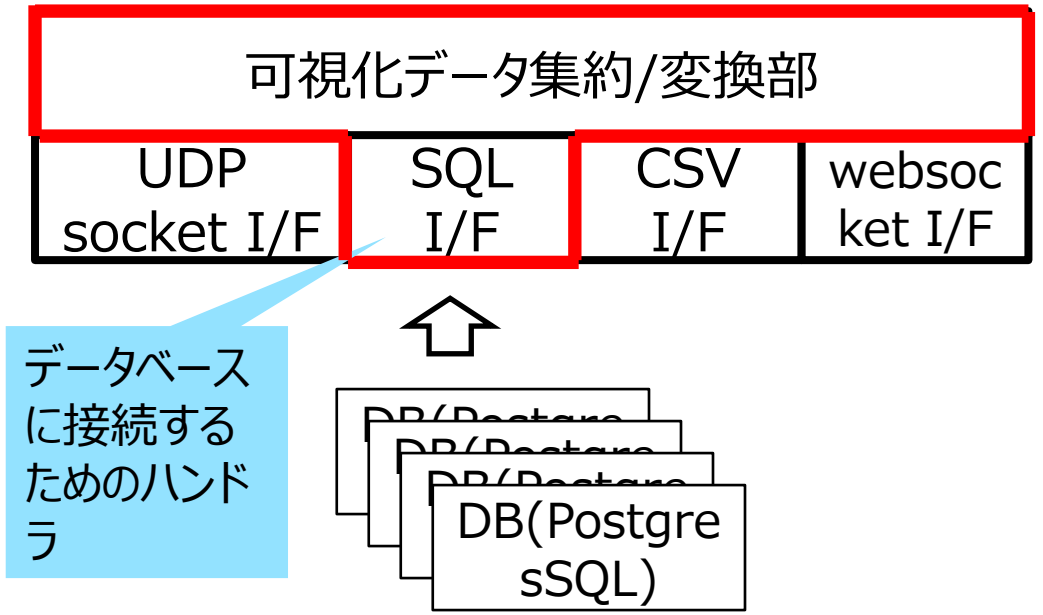
2. 開発対象



2.3.2 可視化プログラム データI/F —— DB連携タイプ

1.DBからSQL文を使って情報を取り出す方法。現状は、PostgreSQLで動作を確認済み。データ読み込みのタイミングは、プログラムのパラメータを変更することで設定する。

2. 開発対象



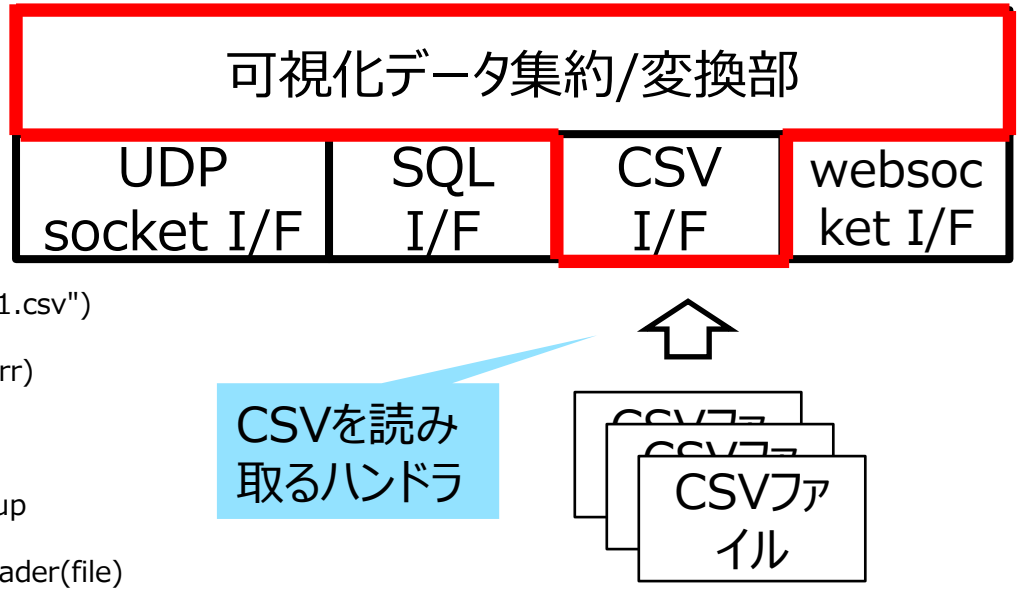

```
// db: データベースに接続するためのハンドラ
var db *sql.DB
// Dbの初期化
dbParam := fmt.Sprintf("host=localhost
port=%d user=postgres password=ca_sim
dbname=ca_sim sslmode=disable", port)
db, err := sql.Open("postgres", dbParam)
if err != nil {fmt.Println("cannot open db")
os.Exit(1)}

sql := "SELECT id, to_char(time,
'HH24:MI:SS'), user_or_bus,x, y FROM
position_log "
                rows, err := db.Query(sql)
```

2.3.3 可視化プログラム データI/F —— データファイル連携タイプ

1.CSVファイルから情報を取り出す方法。動作確認済み。データ読み込みのタイミングは、プログラムのパラメータを変更することで設定する。

2. 開発対象




```
file, err := os.Open("1.csv")
if err != nil {
        panic(err)
}
defer file.Close()

var wg sync.WaitGroup

reader := csv.NewReader(file)
var line []string
```

2.3.4 可視化プログラム データI/F —— スマホ,タブレット連携タイプ

1.スマホの位置情報取得機能から獲られた情報取り出す方法。
スマホから位置情報を取得するためには、https(wss)対応にしないと位置情報が取り出せないことに注意する(*)
(*)https://www.kobore.netの検索エンジンで、”スマホ””位置情報”で詳細なデータが取れる

2. 開発対象