

2019年度 卒業研究論文

MATLAB/Simulinkによる
RADAR計測システムの開発

Development on RADAR measurement system
using MATLAB/Simulink

一関工業高等専門学校
電気情報工学科 秋田研究室
佐藤 凌雅

2020年3月3日

概要

本研究は、路面の段差検知システムの実現を目的としている。提案するシステムはRADARを用いて路面を常に監視し、車両前方の段差の有無を検知する。システム構築にはMATLAB/Simulinkを用いたモデルベース開発を採用し、ソフトウェア設計の効率向上を図った。なお、提案した手法の有効性をシミュレーションを通して検証した。

キーワード：RADAR, モデルベース開発

In this study, I aim to realize a detect system the steps before. The proposing system constantly monitors the road surface using RADAR and detects the presence of a step ahead of the vehicle. In order to improve the efficiency of software design, model-based development using MATLAB/Simulink was adopted for system construction. The effectiveness of the proposed method is verified through a simulation.

Keyword: RADAR, Model-Based Development

目次

第1章 緒言	1
1.1 はじめに	1
1.2 本論文の構成	2
第2章 車載センサの種類	3
2.1 カメラ	3
2.2 LiDAR	4
2.3 RADAR	4
2.4 総括	5
第3章 RADAR	7
3.1 RADARの概要	7
3.2 ミリ波	9
3.3 本研究で使用するRADAR	10
第4章 開発手法	12
4.1 モデルベース開発	12
4.2 MATLAB/Simulink	13
4.3 提案システムの開発手法	13

第5章	システムの構築	14
5.1	テストデータの収集方法	14
5.2	構築した信号処理アルゴリズム	17
第6章	システムの検証	18
6.1	信号処理結果	18
6.2	考察	19
6.3	課題点	20
第7章	結言	22
7.1	本論文のまとめ	22
7.2	今後の展望	23
付録A	RADARの有用性の検証	24
A.1	検証手法	24
A.2	測定結果	27
A.3	結論	28
付録B	自由空間伝搬損失の補償	30
B.1	フリスの自由空間伝搬損	30
B.2	補償手法	31
付録C	ソースコード	32
C.1	測定に使用したプログラム	32
C.2	得られるデータの例	41
C.3	Simulinkにデータを転送するMATLABコード	42
参考文献		44

第1章

緒言

1.1 はじめに

自動車の走行時に生じる振動は乗車している人に負担となり、乗り心地の快適性を損ねる。特に上下方向の振動を軽減する装置としてはバネとダンパを用いるサスペンションが広く用いられている。その中でも、アクティブサスペンションは自動車の揺れを検知し、車内で感じる揺れを抑えるように電子制御することが可能である。

このアクティブサスペンション制御の研究は古くから行われており、1989年には市販車に装備されるまでにその技術は進歩している[1]。しかし、その多くはストロークセンサや加速度センサーがバネ下の動きを検知してから作動するもので、車両が段差に侵入した際の最初の衝撃を完全に吸収できない。そこで、車載センサを用いて路面の状態を測定し、車体に衝撃が加わるタイミングでサスペンションを調整することにより、既存のアクティブサスペンション技術では吸収しきれない最初の衝撃を軽減できると考える。

本研究では、このアクティブサスペンションの動作判定の前段として、路面の段差検知システムの実現を目的とする。

提案するシステムはRADARを用いて路面を常に監視し、車両前方の段差の有無を判別

する。システム構築にはMATLAB/Simulinkを用いたモデルベース開発を採用し、ソフトウェア設計の効率向上を図る。なお、提案した手法の有効性をシミュレーションを通して検証する。

1.2 本論文の構成

本論文は7章より構成され、第1章は緒論であり、本文は第2章から始まる。2章では現在主に使用されている車載センサについて説明し、3章でRADARの原理の説明と研究で使用するRADARについて紹介する。4章ではモデルベース開発について説明を行い、5章で実際に提案するシステムについて述べる。6章で検証結果について説明する。最後に、結論を7章に示す。

第2章

車載センサの種類

この章では自動車の運転支援システムや自動運転に活用されているセンサについて説明する。

2.1 カメラ

一般的に自動運転やADAS向けのカメラは車内にあるルームミラーの裏側などに配置されており、車両の進行方向を向いている。その場合、前方カメラはウインドガラスを挟んで前方の画像を撮影し、画像処理用プロセッサが撮影した画像・映像の解析をリアルタイムで行う。この過程を経て、車両の前方に車両や障害物や人がいるかを検知することができる[2]。

カメラは種々の対象物を検出・認識することができ、対象物に応じて複数の用途に利用することができる。道路上の白線を認識し、その位置から自車のレーン逸脱を警報する機能、前方の車両や歩行者を検知して、衝突の危険がある際に警報を出し、緊急時には自動でブレーキを掛ける機能、等々、様々な用途に用いることができる。

なお、前方を2台のカメラ（ステレオカメラ）を用いると、2台のカメラの映像の視差か

ら物体までの距離を推測することが可能になる[3].

一方、カメラで画像・映像を撮影するということは、基本的には人の目で見るという仕組みと類似の原理であることから、夜間や逆光に加え、濃霧、豪雨、豪雪などの悪天候の場合は検出能力が低下することが課題の一つとされている[2].

2.2 LiDAR

LiDAR は、赤外線のレーザ光をパルス状に照射し、物体に反射されて帰ってくるまでの時間から距離を計測するセンサである。動作原理が後述するRADARと類似しているため、別名レーザRADARとも呼ばれる。細く絞ったレーザ光を可動ミラーによって方向を変えてスキャンすることで物体の方位も検出することができる。このようなタイプのセンサをスキャン LiDAR と呼ぶ。LiDAR は、ミリ波RADARに比べてさらに波長の短い電磁波である赤外光を使っているため、検出の際の空間分解能が高いことが特長である。この特長を生かし車の進路の安全な場所の検出に使うことができる。

ただし、赤外光を用いるため、豪雨、豪雪、霧などの悪天候時に検出性能が低下するという短所がある[3].

また、LiDARは高価で屋外で使用できる性能のものになると1個あたり数百万円もする。さらに、LiDARは測定時に物理的に回転する。耐久性の観点から量販車に実装することは現状のままでは少し厳しい[4].

2.3 RADAR

ミリ波RADARは、ミリ波と呼ばれる周波数帯が30GHz～300GHzの非常に波長の短い電波を照射し、物体に反射されて帰ってくる電波を検出することにより、物体までの距離

と方向を検出するセンサである。非常に高い周波数の電波は直線性が強く、電波でありながらレーザのように扱うことができる。この周波数を波長にすると1~10mmとミリオーダーの長さになることから、「ミリ波」と呼ばれている。

ミリ波センサのミリ波は直線性が強いため、雨や雪が降っている悪天候な状況でも、遠くまで検出することが可能となる。また、ミリ波は光ではなく電波であるため、トンネルや対向車のライトが当たるなどのように明るさが急激に変化する条件でも、明るさに左右されず検出できる[5]。

2.4 総括

前述した、カメラ、ミリ波RADAR、LiDARの三種類のセンサは、いずれも長所と短所がある。カメラによる検出は、物体の識別が可能であり、車両や歩行者など自動車を安全に走行させるうえで重要な物体を、他の物体と区別して検出することができる。しかしながら、カメラの画像は人間の目で見る画像と同じ原理に基づくものであり、夜間や逆光など光源が不適切なシーンや、濃霧、豪雨、豪雪などの悪天候のシーンでは、人間と同じく検出能力が低下する。

これに対しミリ波RADARは、自らの発する電波を利用した検出のため、光源や天候に左右されず良好な検出特性を維持できる。また対象物体までの距離を正確に計測できる特長もある。しかし、検出の際の空間分解能が他のセンサに比べて劣るため、物体の識別は困難であり、また段ボール箱や発泡スチロールなど、電波の反射率の低い物体の検出が難しいという課題がある。

LiDARは、赤外線のレーザ光を用いるため、電波の反射率が低い物体も検出できる。特に段ボール箱、木材、発泡スチロールなど、路上散乱物として走行の妨げになる物体も検出可能である。またスキャン LiDAR では高い空間分解能で距離と方位を検出できるた

め、物体検出だけでなく、それらの間のフリースペースの検出も可能である。ただし、赤外光を用いるため、豪雨、豪雪、霧などの悪天候時に検出性能が低下するという短所がある。

表2.1に、各種センサーの機能・性能を相互比較した例を示す。

表2.1 各種センサーの機能・性能別相互比較（5段階評価:5がベスト）[6]

機能・性能	LiDAR	レーダー	可視カメラ
近傍の物体検知	2	4	2
測定距離	4	4	5
分解能	4	3	5
暗い場所での動作	5	5	1
明るい場所での動作	5	5	4
雪・霧・雨の際の動作	3	5	2
色彩/コントラスト	1	1	5
検出速度	4	5	2
センサーの寸法	1	5	5
価格	1	5	5

本研究は路面の段差検知システムの実現を目的としている。そのため、他のセンサと比較して近傍の物体検知と検出速度に優位性があり、かつ安価であるRADARが最適なセンサであると考える。

第3章

RADAR

この章ではRADARの基本原理、研究に使用するRADARの性能を示し、LiDARとの比較を行う。

3.1 RADARの概要

RADARは電波を発射し遠方にある目標物を検知し、そこまでの距離を測る電波検知装置である[7]。

RADARシステムは以下のような簡易なブロック図（図3.1）で表される。

- 送信アンテナ部

電波を効率よく物標に放射し、反射波を捉える装置

- 信号処理部

受信したデータ列を検出や測距など目的を達するために行うソフトウェア処理

- 検波部

受信波を信号処理しやすい信号形式に変換する装置

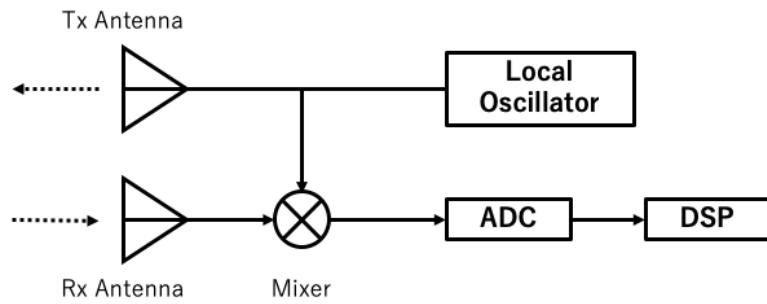


図3.1 RADARのブロック図

例えば、RADARではパルス波（矩形波）を図3.2のように物標方向に照射し、その反射信号から距離に対する反射強度を得ることができる。物標の位置はこの反射波の強度のピークが発生している箇所に存在すると考えることができる。なお、この反射強度は物体の比誘電率に依存する。障害物のない箇所でも空気による若干の反射は発生するが、空気の比誘電率は物標より小さくなるため反射は少なくなり、物標の検知が可能となる。

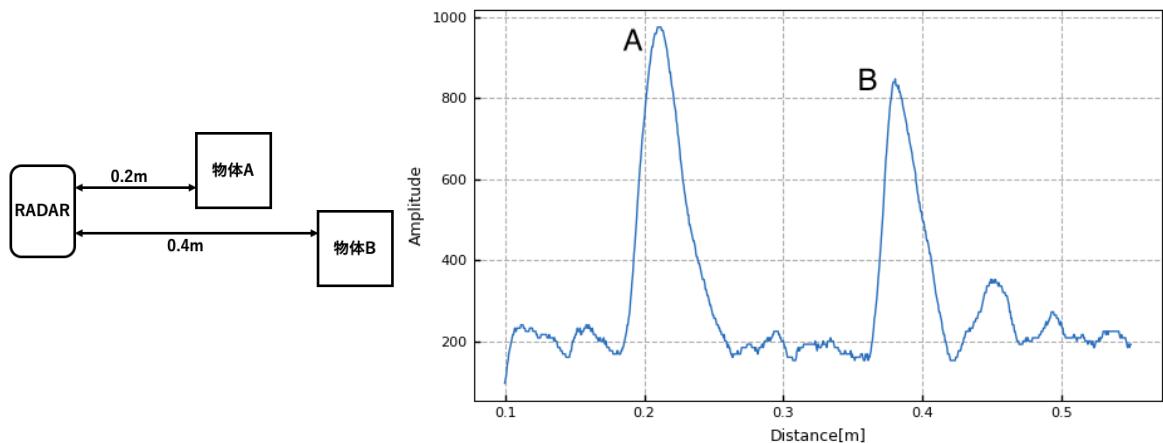


図3.2 物体の位置と反射強度の関係

3.2 ミリ波

ミリ波の波長は図3.3に示すように1mm～10mmと非常に短い。

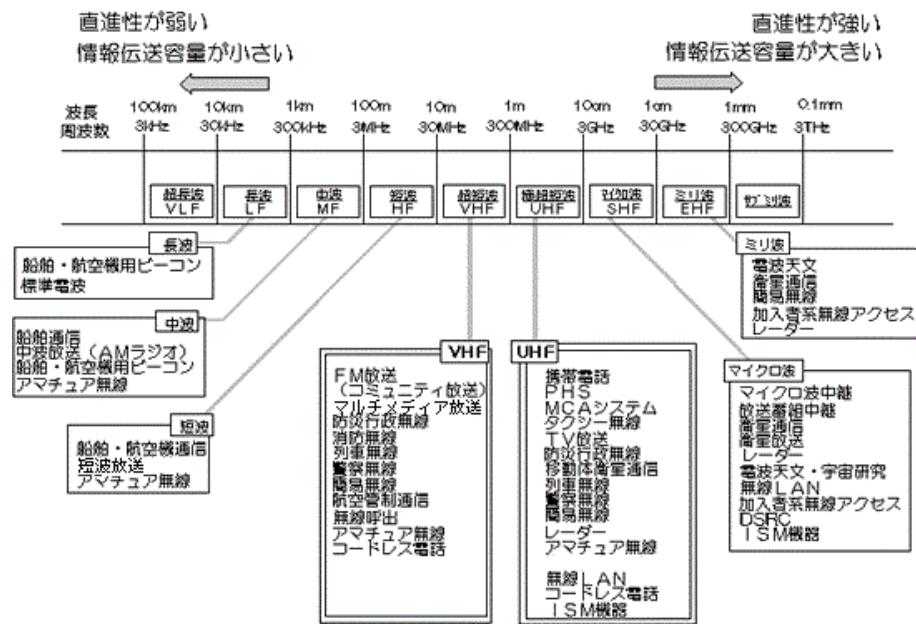


図3.3 電波の分類[8]

波長が1cm以下のマイクロ波と比較した際の特徴を以下に述べる[9]。

1. 広帯域性

RADARの分解能は帯域幅の逆数となる。ミリ波は広い帯域幅を利用できるため高分解能・高精度化が可能となる。

2. 装置の小型軽量化

アンテナ開口径は波長に比例するため、周波数が高いほど装置サイズを小さくできる。

3. 鋭い指向性

空間的に高密度の利用が可能となる。また、お互いの電波の干渉も低くできる。

3.3 本研究で使用するRADAR

本研究ではAcconeer社製のXC112/XR112評価キットを用いる。この評価キットは同社製のA111というパルスRADARを搭載したRADARセンサボード（XR112）とRaspberryPiとの接続用コネクタボード（XC112）から構成される。図3.4, 図3.5にXC112およびXR112の外観を示す。

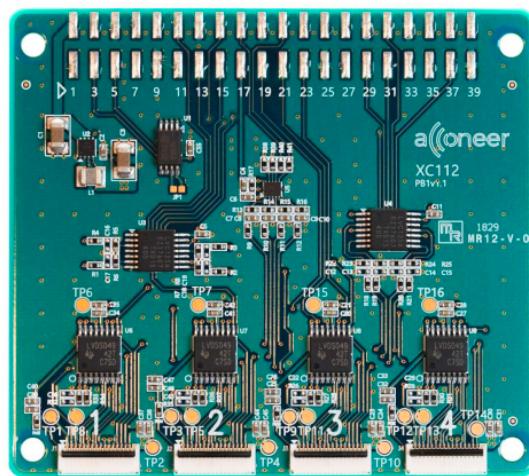


図3.4 XC112[11]

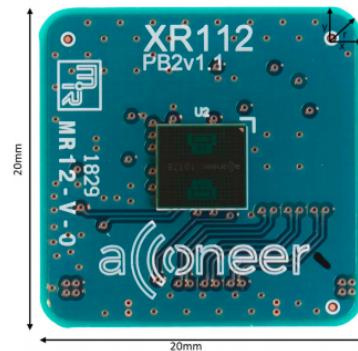


図3.5 XR112[13]

評価キットのブロック図を図3.6に示す。

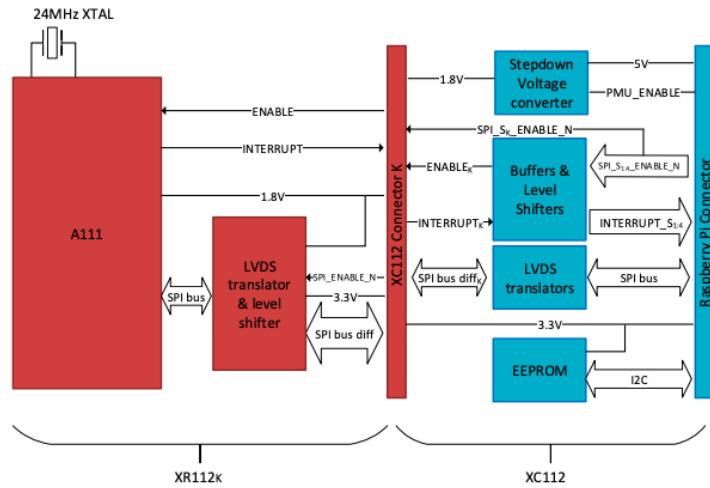


図3.6 XC112/XR112評価キットのブロック図[12]

シングルボードコンピュータのRaspberryPiを使用することを想定しており、本研究でもRaspberryPi3 ModelBをRADAR制御用コンピュータとして使用した。評価キットに使用されているA111は周波数60GHz（波長 5 mm）のミリ波RADARである[12]。

第4章

開発手法

この章ではモデルベース開発という開発手法について概説し、提案システムの開発プロセスについて説明する。

4.1 モデルベース開発

モデルベース開発 (Model Based Design / Development MBD) とは、シミュレーション可能なモデルを用いるソフトウェア開発手法のことを指す。制御系MBDでは、制御器および制御対象、またはその一部をモデルで表現し、机上シミュレーション/リアルタイムシミュレーションにより制御アルゴリズムの開発・検証を行う。このモデル化とシミュレーションは、ハードウェアが利用できない場合（システム開発初期）には特に有益である。さらに、コード生成ツールを用いて、制御器モデルから実際の制御器（マイコン等）に組み込む制御用Cプログラムを作成することも可能となる。コード生成により、時間を節約し手作業でのコーディングによるエラーを防ぐことができる[14]。

4.2 MATLAB/Simulink

本研究では、信号処理アルゴリズムの開発にMATLAB/Simulinkを使用した。MATLABはThe MathWorks社が開発している数値解析ソフトウェアであり、その中で使うプログラミング言語の名称でもある[15]。また、SimulinkはMATLABと統合化されたモデルベース開発のためのソフトウェアである。モデルにMATLABアルゴリズムを組み込み、シミュレーションの結果をMATLABにエクスポートして詳細な解析を行うことができる[16]。

4.3 提案システムの開発手法

提案システムの開発は、具体的には以下の開発プロセスに沿って行った。

1. 制御対象のデータ収集

RADAR信号処理アルゴリズムを構築する上で、シミュレータに入力するためのテストデータを実機から収集した。

2. 信号処理アルゴリズム設計

RADAR信号から所望のデータ（路面の段差の有無）を算出する信号処理器をSimulink上に構築し、シミュレーションを行った。

3. アルゴリズムの検証

事前に収集したRADAR信号のデータを解析し、アルゴリズムの有効性を検証した。

第5章

システムの構築

この章では、構築した段差検知システムについて説明を行う。

5.1 テストデータの収集方法

段差検知の信号処理アルゴリズムを構築する上で、シミュレータに入力するためのテストデータの収集を行った。

5.1.1 分解組立型電気自動車PIUS

今回、RADARの処理を行うために株式会社モディーが開発、製造を行っているPIUSと呼ばれる分解組立型電気自動車を使用した。このPIUSは組立、分解ができるよう部品点数を極力減らしているものの、自動車の持つ基本性能を十分に備えており、前進、ニュートラル、後進のシフトスイッチ、ウィンカー、前照灯、サイドブレーキと本物の自動車を運転するときに使用している装備を備えている[17]。図にPIUSの外観を示す。



図5.1 分解組立型電気自動車PIUS[17]

5.1.2 製作した筐体

このPIUSにRADARセンサを固定する筐体を3DCADで設計し、3Dプリンタで製作した。この筐体をPIUSに取り付けた様子を図5.2に示す。



図5.2 センサの取り付けの様子

5.1.3 測定環境

データ収集は本校敷地内にある段差を用いて行った。この段差の上を約30km/hで走行させ、RADARのデータを取得した。また、同時に車体に加わる衝撃を評価する目的で、加速度の測定も行った。測定環境の模式図を図5.3、段差の写真を図5.4に示す。

RADAR取り付け角度は地面とのなす角が45度になるようにした。また、データの取得は毎秒300回行う設定とした。

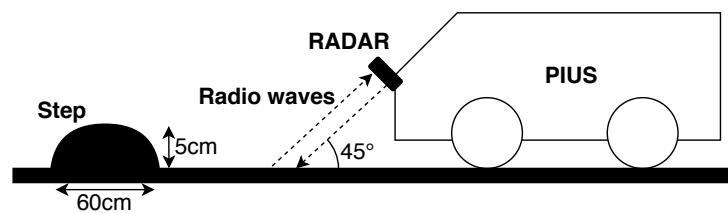


図5.3 測定環境の模式図



図5.4 データ収集に使用した段差の写真

5.2 構築した信号処理アルゴリズム

RADARのデータから段差の有無を検出するアルゴリズムはSimulink上で作成した。

図5.5に開発した信号処理アルゴリズムのブロック図を示す。なお、図中のbinarizeブロックは第一引数のuが第二引数のthreshold以上であれば1、それ以下であれば0を出力するものである。

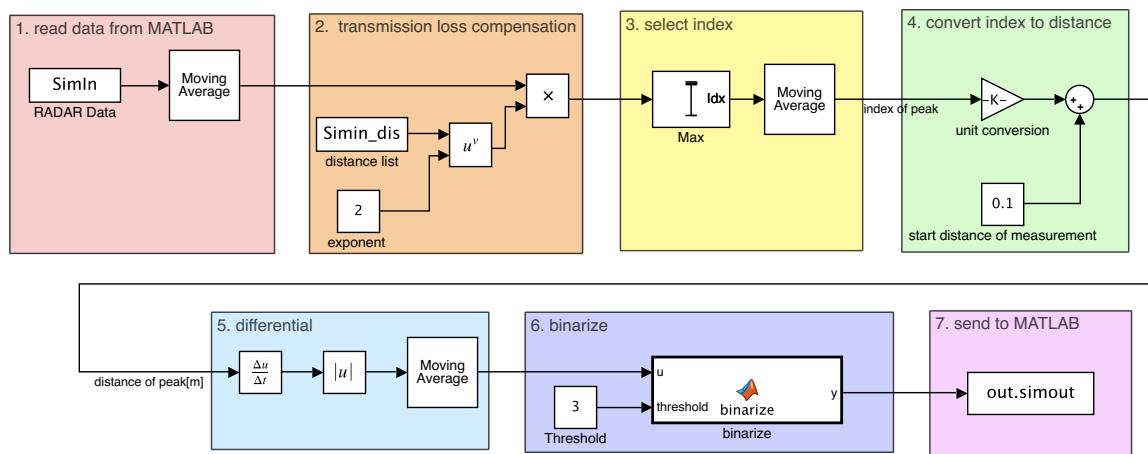


図5.5 開発した信号処理アルゴリズムのブロック図

アルゴリズムの概要は以下の通りである。リスト番号は図5.5と対応している。

1. RADARデータをMATLABのワークスペースから配列形式でSimulinkに読み込む
2. 電波は自由空間で距離の2乗に比例して減衰していくため、その影響を補償
3. 反射強度のピークのインデックスを選択
4. ピークのインデックス番号を実際の距離の単位に変換
5. 得られた路面までの距離を微分し、絶対値をとる
6. 微分値が閾値を超えた時に段差検知と判定
7. 処理結果をSimulinkからMATLABに転送する。

第6章

システムの検証

この章では、構築した段差検知システムについての検証結果を示し、その考察を述べる。

6.1 信号処理結果

取得したRADARのデータを前章で構築したアルゴリズムに入力し、動作を確認した。図5.5における「5. deffrential」の出力結果、すなわち路面までの距離の微分値の絶対値を図6.1に示す。この値が閾値（今回は0.7）を超えた場合に段差検知とする。段差検知は図中の赤線で示した箇所で発生しており、0.405秒と0.610秒で段差を検出していた。

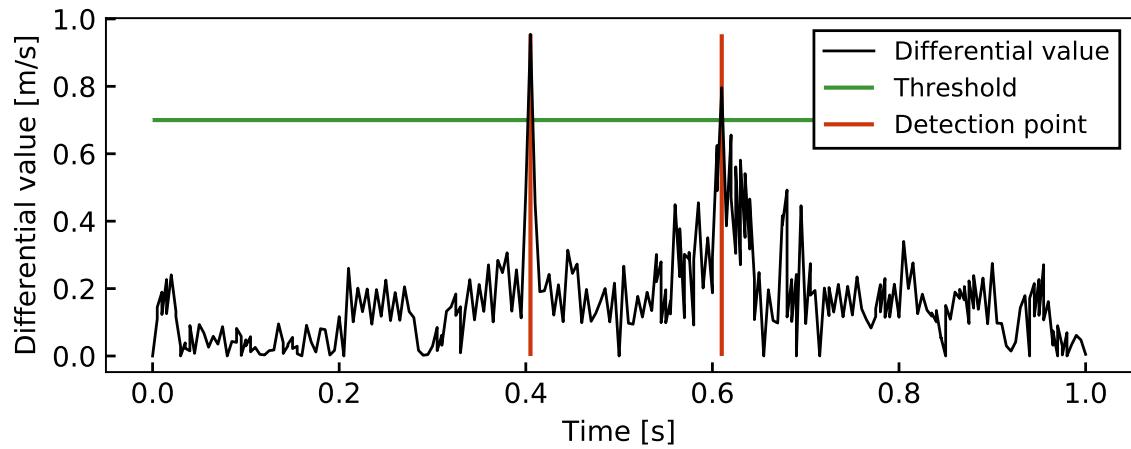


図6.1 路面までの距離の微分値の絶対値と段差検出点

6.2 考察

図6.2に走行中の車両の加速度の推移を示す。システムが段差を検知した約50ms後に 15m/s^2 以上の加速度が車体に加わっていることがわかる。このことから、提案システムは車体に衝撃が加わる前に路面の段差をあらかじめ検出することができると結論づける。

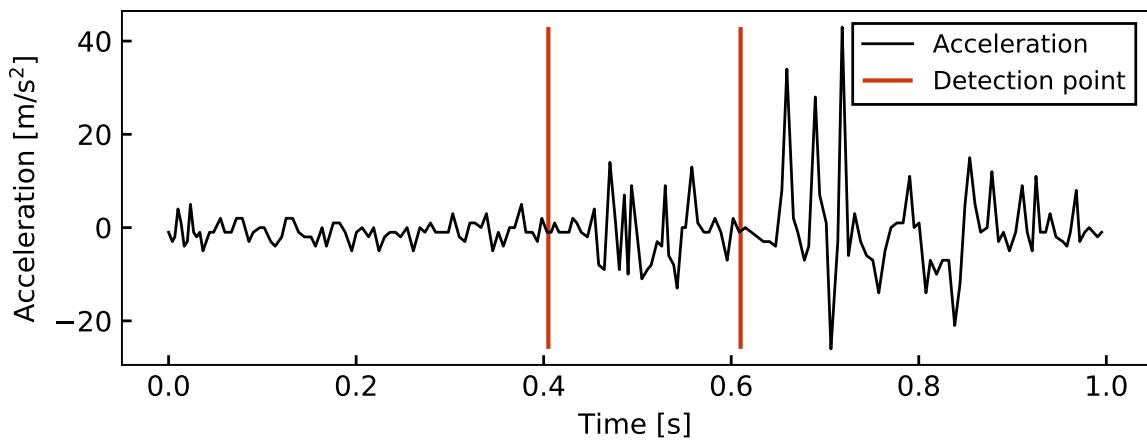


図6.2 車体に加わった上下方向の加速度と段差検出点

6.3 課題点

しかし、提案したアルゴリズムにはいくつかの問題がある。

図6.3は発車時に車体に加わった上下方向の加速度の推移と段差検出点を示している。

このデータを見ると、車両の発車時（0.6秒）時点でも段差検知と判定している。しかし、実際は車両前方には段差は存在しない。図6.4はカーブ終了直後の加速度と段差検出点のグラフである。0.750秒の地点で 33m/s^2 の加速度を検知しているが、それ以前での段差検知ができておらず、車体に衝撃が加わってから段差検知の判定をしている。

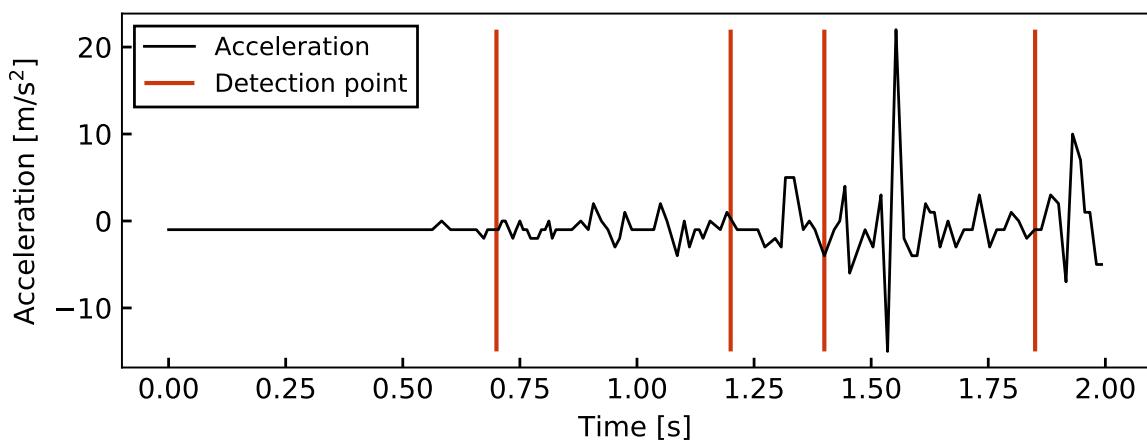


図6.3 発車時に車体に加わった上下方向の加速度と段差検出点

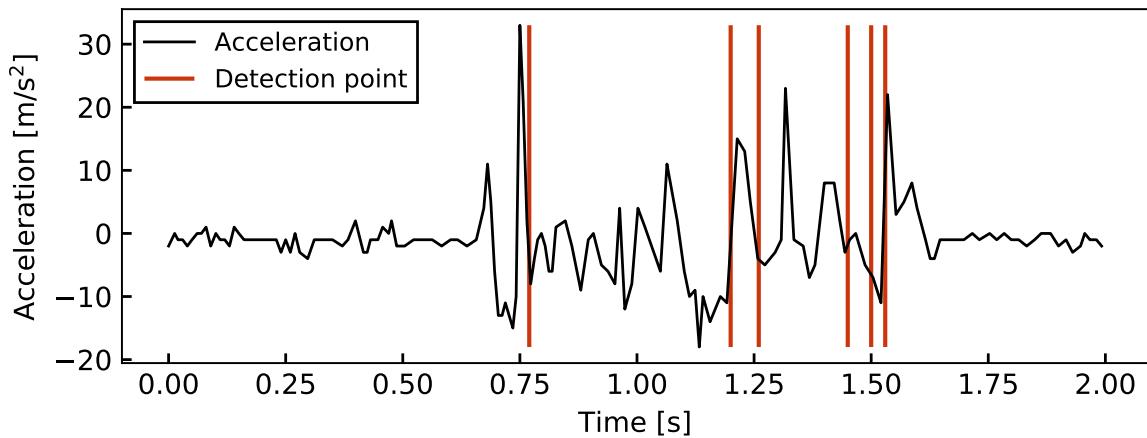


図6.4 カーブ走行後に段差に侵入した際に車体に加わった上下方向の加速度と段差検出点

現在考えうる課題点を以下にまとめる。

1. 発車時の振動による誤検知

このシステムではRADARから得られる距離の微分値（変化量）を基準に段差の有無を検出している。発車時のような車両の運動が急変する際にRADARが出力する距離の値に影響を及ぼす可能性がある。

2. 旋回時での段差非検出

RADARは車両前面の1つのみで測定を行っている。カーブの先にある段差に関しては現状のシステムでは検出することは困難である。

3. 閾値が実験的に導出されている

段差の検出に使用している微分絶対値の閾値は測定データを参考に手動でチューニングしたものであり、理論的に導出されたものでない。最終的に人が乗る車両のアクティブサスペンションの動作計画に活用することを考えると、その判定の責任を担保するためには理論的な閾値の導出が必要不可欠である。

第7章

結言

7.1 本論文のまとめ

本研究では車両前面に取り付けたRADARを用いて路面の状態を計測し、段差の有無を検出するシステムを開発した。提案した手法は、RADARから得られた路面までの距離を微分し、その絶対値が閾値を超えた時に段差検知と判定するものである。また、システムの開発効率の向上のため、MATLAB/Simulinkを使用し、モデルベースでのアルゴリズム開発を試みた。本論文での実験および考察において得られた知見を以下に示す。

- シミュレーションにおいて提案した手法を検証したところ、 15m/s^2 以上の加速度が発生するような段差の検出を行うことができた。
- 提案したアルゴリズムには6.3節に挙げられるような問題がまだ残っており、実用化に向けては改良の余地がある。
- 開発にシミュレーションソフトを使用することで、実際の車両を動作させることなく、アルゴリズムの検証ができるため、開発効率が飛躍的に向上した。

この手法を改良し、走行中の車両の前方の段差を事前に検知することが可能となれば、

車両が段差に乗り上げるのと同じタイミングでアクティブサスペンションを動作させ、段差走行時の乗り心地を改善できると考える。

7.2 今後の展望

本研究では路面の状態を計測するためにシステムをRADARを用いて構築し、検証を行った。しかしながら、6.3節に挙げられるような問題がまだ残っており、検知精度の向上のためにもアルゴリズムの改良は必須である。アルゴリズムの改良の際はRADARのみで得られるデータだけでは限界があるため、カメラやLiDAR、その他センサを併用し、センサフュージョンを行う手法が考えられる。また、現状のシステムは段差の有無を出力するものにとどまっており、実際の車両の運動制御への応用を視野にいれた場合、段差の大きさの特定についても考える必要がある。

付録A

RADARの有用性の検証

ここではRADARとLiDARにおける環境依存性について検証を行う。

A.1 検証手法

A.1.1 センサ

比較対象としてRADARと同じ価格帯のLiDARを用意した。使用したLiDARの性能と外観を表A.1、図A.1に示す。[18]

表A.1 LiDARの性能

製品名	YDLIDAR X4
検出距離	120~11,000mm
走査視野	360°
角分解能	0.5°
走査周波数	6~12Hz
電源電圧	DC5V
消費電流	450mA



図A.1 測定に使用したLiDAR

なお、LiDARは仕様上、全方向（ヨー角360度）をスキャンするが、今回の実験では壁側の1本のレーザーだけ使用する。

A.1.2 実験環境

壁との距離をセンサで24時間計測する。この際の実験環境を表A.2、実験の様子を図A.2に示す。

表A.2 実験環境

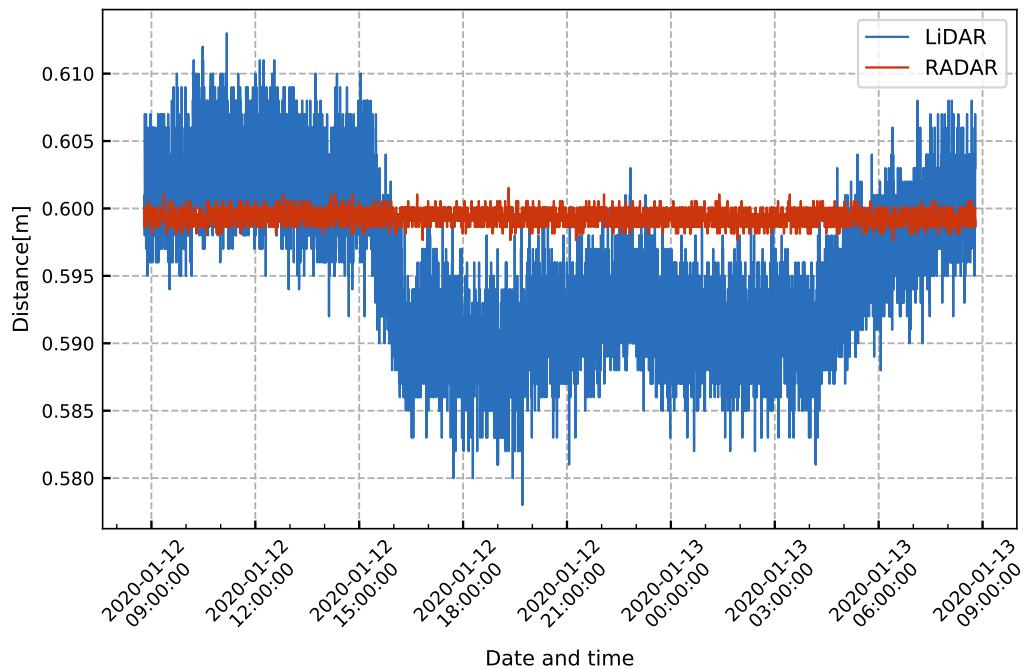
測定日時	2020/01/12 08:47～2020/01/12 08:47
測定場所	一関高専4号棟402号室
測定環境	無人、消灯、外からの外乱光あり
対象物までの距離	0.6m
測定間隔	10秒



図A.2 RADARとLiDARの比較実験の様子

A.2 測定結果

RADAR, LiDARから得られた観測データを図A.3に示す。また、センサの平均値、分散値を表A.3にまとめると。

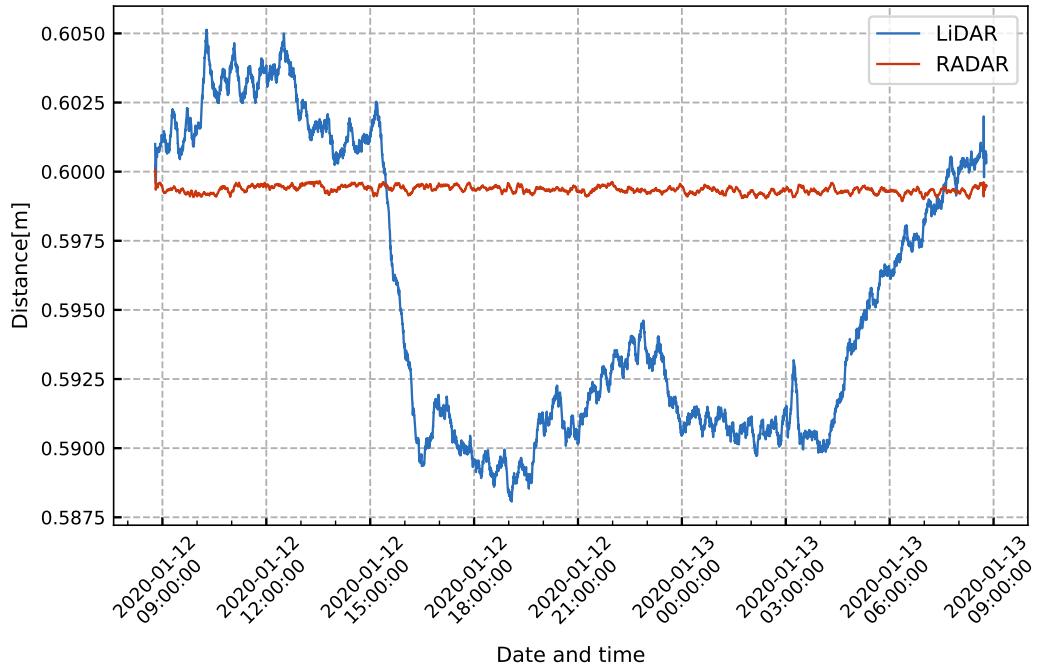


図A.3 LiDARとRADARの測定値の推移

表A.3 実験結果

	LiDAR	RADAR
平均	0.59538803	0.59935253
中央値	0.59500000	0.59958800
分散	0.00003424	0.00000024
標準偏差	0.00585180	0.00049350

図A.3のデータに対して移動平均フィルタを施した。フィルタ処理後のデータを図A.4に示す。



図A.4 LiDARとRADARの測定値の推移（移動平均フィルタ処理後）

A.3 結論

表A.3からLiDARと比較してRADARの方が値のばらつきが小さいことがわかった。しかし、LiDARとRADARでは測定原理が異なるため、単純にこのデータだけでRADARの方が優れているとは言い切れない。そこで、図A.4を見ると、LiDARは時間経過に対応して得られるデータが変動している。測定を行なった2020年1月12日の一関市の日没は16:33、1月13日の日の出は6:54であった。LiDARの測定値が大きく変動している時間と日没、日の出の時間はおおよそ一致していることから、この値の変動は外からの

太陽光の影響を受けた可能性がある。また、測定を行なった部屋では温度、湿度の管理を行なっていなかったため、光以外の環境的要因によって測定値にばらつきが出た可能性も考えられる。

一方でRADARは24時間を通してほぼ一定の値を取り続けていた。このことから、RADARはLiDARと比較して外乱の影響を受けにくいと考えられる。以上より、RADARは他の車載センサと比較して環境変化に対しての優位性があると結論づける。

付録B

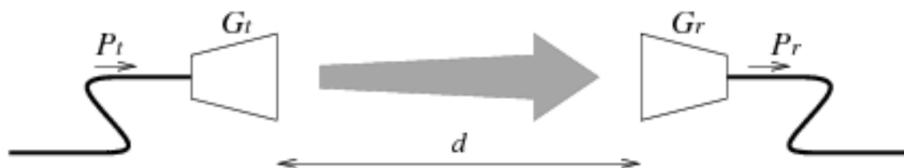
自由空間伝搬損失の補償

ここでは、提案する信号処理アルゴリズム（図5.5）の「2. transmission loss compensation」部で用いられている自由空間伝搬損失の補償の手法について述べる。

B.1 フリスの自由空間伝搬損

電波が物標に照射されると様々な方向に散乱される。この散乱の現象は物標の表面形状や粗さ、材質、入射角等の条件に大きく依存する。

ここでは図B.1に示すようなアンテナによる電波の伝送モデルを考える。ここで、送受信アンテナ間の距離 d 、送受信アンテナの利得を G_t 、 G_r 、波長 λ 、そして送信電力 P_t が与えられればフリスの伝達公式により（B.1）式のように受信電力が得られる[7]。



図B.1 電波の伝送モデル

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi d} \right)^2 = \frac{G_t G_r}{L_d} \quad (\text{B.1})$$

ここで, L_d はフリスの自由空間伝搬損と呼ばれ, (B.2) 式のように距離 d の二乗に比例し, 波長 λ の二乗に反比例する.

$$L_d = \left(\frac{4\pi d}{\lambda} \right)^2 \quad (\text{B.2})$$

B.2 補償手法

前節で示した通り, 電波は自由空間に置いて伝搬距離の二乗に比例して減衰していく. RADARでの測定の際, この自由空間伝搬損によって, 近傍の電波の反射強度は大きくなり, 遠方の電波の反射強度は小さくなる. 正確に路面の位置を推定するため, この電波の減衰を補償する必要がある.

得られたRADARのデータの距離別の強度全てに対して, 電波の往復距離の二乗を乗算することで減衰を補償した.

付録C

ソースコード

ここでは測定に使用したC言語のソースコード, 得られるデータの例, MATLABからSimulinkへデータを転送するコードを示す.

C.1 測定に使用したプログラム

```
1 #include <stdbool.h>
2 #include <stddef.h>
3 #include <stdint.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <sys/stat.h>
7 #include <time.h>
8 #include <sys/time.h>
9
10 #include <wiringPi.h>
11 #include <wiringSerial.h>
12
13 #include "acc_driver_hal.h"
14 #include "acc_rss.h"
15 #include "acc_service.h"
16 #include "acc_service_envelope.h"
17 #include "acc_sweep_configuration.h"
```

```

18
19 #include "acc_version.h"
20
21 #define REQ_DATA_NUM 128 // Number of
22 // data to be acquired
22 #define USE_SENSOR_NUM 2 // Number of
23 // sensors used
23 #define period_sec 1.0 / 50.0 // Execution
24 // cycle[s]
24 int use_sensor_id[USE_SENSOR_NUM] = {1, 3}; // List of
25 // sensor IDs to be used
25
26 double distance_to_object[3] = {0}; // Distance to object[m]
27 int initied = 0;
28 uint16_t count = 0;
29
30 static acc_service_status_t
31 // execute_envelope_with_blocking_calls(int sensor_num,
32 // acc_service_handle_t handle);
31 static acc_service_handle_t createHandle(
32 // acc_service_configuration_t envelope_configuration);
32 void writeHeader(acc_service_handle_t handle);
33 static void configure_sweeps(acc_service_configuration_t
34 // envelope_configuration, int id);
34 static acc_hal_t hal;
35
36 double processed_data[1000] = {0};
37
38 FILE *fp[3];
39 FILE *fp_g;
40
41 int main(void)
42 {
43 // Establish communication with Nucleo
44 int fd = serialOpen("/dev/ttyACM0", 115200);
45 wiringPiSetup();
46 fflush(stdout);
47 if (fd < 0)
48 {
49 printf("can't open serialport");
50 }
51 else
52 {
53 printf("opened Serial Port");
54 }

```

```

55
56 // Log file generation processing
57 char dir_name[64];
58 char filename[USE_SENSOR_NUM][64];
59 char time_str[32] = "00_00_00_00_00";
60 time_t t = time(NULL);
61 strftime(time_str, sizeof(time_str), "%m_%d_%H_%M_%S",
62           localtime(&t));
62 sprintf(dir_name, "log/%s", time_str);
63 // Generate storage folder
64 if (mkdir(dir_name, 0777) != 0)
65 {
66     printf("Folder creation failed.\n");
67     return EXIT_FAILURE;
68 }
69 for (int i = 0; i < USE_SENSOR_NUM; i++)
70 {
71     // Generate files
72     sprintf(filename[i], "%s/%d.csv", dir_name,
73             use_sensor_id[i]);
73     if ((fp[i] = fopen(filename[i], "w")) == NULL)
74     {
75         fprintf(stderr, "File open failed.\n");
76         return EXIT_FAILURE;
77     }
78 }
79
80 // Generate files
81 char gyro_file_name[128];
82 sprintf(gyro_file_name, "%s/gyro.csv", dir_name);
83 if ((fp_g = fopen(gyro_file_name, "w")) == NULL)
84 {
85     fprintf(stderr, "File open failed.\n");
86     return EXIT_FAILURE;
87 }
88
89 // Initialize RADAR driver
90 if (!acc_driver_hal_init())
91     return EXIT_FAILURE;
92
93 hal = acc_driver_hal_get_implementation();
94 if (!acc_rss_activate_with_hal(&hal))
95     return EXIT_FAILURE;
96
97 // various settings

```

```

98     acc_service_configuration_t envelope_configuration[
99         USE_SENSOR_NUM];
100    acc_service_handle_t handle[USE_SENSOR_NUM];
101    for (int i = 0; i < USE_SENSOR_NUM; i++)
102    {
103        envelope_configuration[i] =
104            acc_service_envelope_configuration_create();
105        if (envelope_configuration[i] == NULL)
106            return EXIT_FAILURE;
107
108        // Eliminate the moving average filter
109        acc_service_envelope_running_average_factor_set(
110            envelope_configuration[i], 0.0);
111
112        configure_sweeps(envelope_configuration[i],
113            use_sensor_id[i]);
114
115        handle[i] = createHandle(envelope_configuration[i]);
116        if (handle[i] == NULL)
117            return EXIT_FAILURE;
118    }
119
120    clock_t system_start_time = clock();
121    long loop_num = 0;
122
123    // Run the envelope with a blocking call
124    acc_service_status_t service_status[USE_SENSOR_NUM];
125    while (true)
126    {
127        loop_num++;
128        count++;
129
130        double now_time = (double)(clock() - system_start_time)
131            / CLOCKS_PER_SEC;
132
133        for (int i = 0; i < USE_SENSOR_NUM; i++)
134        {
135            service_status[i] =
136                execute_envelope_with_blocking_calls(i, handle[i])
137                ;
138        }
139
140        if (distance_to_object[0] != 0)
141        {
142            if (distance_to_object[0] < 0.15)

```

```

136     {
137         serialPutchar(fd, 1);
138     }
139     else
140     {
141         serialPutchar(fd, 0);
142     }
143 }
144 else
145 {
146     serialPutchar(fd, 0);
147 }
148
149 // Get acceleration data
150 int get_char;
151 int acc_data = 0;
152 get_char = serialGetchar(fd);
153 acc_data = get_char - 128;
154 fprintf(fp_g, "%.3f,%d\n", now_time, acc_data);
155
156 // Display in terminal
157 printf("\033[2J");
158 printf("\033[0;0H");
159 printf("tim: %.3f\r\n", now_time);
160 printf("dis: %.3f\n", distance_to_object[0]);
161 printf("acc: %d\n", acc_data);
162
163 // Wait for the desired loop time
164 while (true)
165 {
166     if (((double)clock() - system_start_time) / (double)
167         CLOCKS_PER_SEC) > (double)loop_num * period_sec)
168         break;
169 }
170
171 if (service_status[0] != ACC_SERVICE_STATUS_OK)
172 {
173     acc_service_envelope_configuration_destroy(&
174         envelope_configuration[0]);
175     return EXIT_FAILURE;
176 }
177 acc_service_envelope_configuration_destroy(&
178     envelope_configuration[0]);

```

```

178
179     acc_rss_deactivate();
180
181     return EXIT_SUCCESS;
182 }
183
184 // Generate handle
185 acc_service_handle_t createHandle(
186     acc_service_configuration_t envelope_configuration)
187 {
188     acc_service_handle_t handle = acc_service_create(
189         envelope_configuration);
190     if (handle == NULL)
191     {
192         printf("acc_service_create failed\n");
193     }
194
195     return handle;
196 }
197
198 double measure_start_dis;
199 double measure_len;
200 double measure_end_dis;
201 uint16_t data_len;
202 double index_to_meter;
203
204 // Run the envelope with a blocking call
205 acc_service_status_t execute_envelope_with_blocking_calls(
206     int sensor_num, acc_service_handle_t handle)
207 {
208     // Get meta data
209     acc_service_envelope_metadata_t envelope_metadata;
210     acc_service_envelope_get_metadata(handle, &
211         envelope_metadata);
212     double measure_start_dis = (double)envelope_metadata.
213         actual_start_m;
214     double measure_len = (double)envelope_metadata.
215         actual_length_m;
216     double measure_end_dis = (double)(measure_start_dis +
217         measure_len);
218     uint16_t data_len = envelope_metadata.data_length;
219     double index_to_meter = (double)(measure_len /
220         REQ_DATA_NUM); // [m / point]
221
222     if (initied != USE_SENSOR_NUM)

```

```

215     {
216         for (uint_fast16_t index = 0; index < REQ_DATA_NUM;
217             index++)
218         {
219             double now_depth = measure_start_dis + index *
220                 index_to_meter; // [m]
221             fprintf(fp[sensor_num], ",%f", now_depth);
222         }
223         initied++;
224     }
225
226     // Get sensor status
227     acc_service_status_t service_status = acc_service_activate
228         (handle);
229
230     double sec = (double)count * period_sec;
231     fprintf(fp[sensor_num], "\n%f", sec);
232
233     acc_service_envelope_result_info_t result_info;
234     // Distribute processing according to sensor status
235     if (service_status == ACC_SERVICE_STATUS_OK)
236     {
237         uint16_t data[2000];
238
239         // Get envelope data from sensor
240         // This function blocks until the next sweep arrives
241         // from the sensor and the envelope data is copied to
242         // the "data" array
243         service_status = acc_service_envelope_get_next(handle,
244             data, data_len, &result_info);
245
246         // Thin out the acquired data
247         for (int i = 0; i < REQ_DATA_NUM; i++)
248         {
249             processed_data[i] = data[(int)(data_len * i /
250                 REQ_DATA_NUM)];
251         }
252
253         // Attenuation correction (Now, Do it in MATLAB)
254         // for (int i = 0; i < data_len; i++)
255         // {
256             //     data[i] = data[i] + 0.5 * data[i] * (i *
257                 index_to_meter);
258         // }

```

```

252
253     double max_data = 0;
254     int max_data_index = 0;
255     for (int i = 0; i < data_len; i++)
256     {
257         if (max_data < data[i])
258         {
259             max_data_index = i;
260             max_data = data[i];
261         }
262     }
263
264     if (max_data > 1000)
265     {
266         distance_to_object[sensor_num] = index_to_meter *
267             max_data_index;
268     }
269     else
270     {
271         // If the intensity is lower than the threshold, treat
272         // it as non-detection
273         distance_to_object[sensor_num] = 0;
274     }
275
276     if (service_status == ACC_SERVICE_STATUS_OK)
277     {
278         for (uint_fast16_t index = 0; index < REQ_DATA_NUM;
279             index++)
280         {
281             // printf("%6u", (unsigned int)(data[index]));
282             fprintf(fp[sensor_num], ",%lf", processed_data[index
283             ]);
284         }
285     }
286     printf("\n");
287 }
288 }
289 else
290 {
291     printf("acc_service_activate() %u=>%s\n", (unsigned
292         int)service_status, acc_service_status_name_get(

```

```

                service_status));
292 }
293
294 // acc_service_destroy(&handle);
295
296 return service_status;
297 }
298
299 // Sweep settings
300 void configure_sweeps(acc_service_configuration_t
301 envelope_configuration, int id)
301 {
302     acc_sweep_configuration_t sweep_configuration =
303         acc_service_get_sweep_configuration(
304             envelope_configuration);
305
306     if (sweep_configuration == NULL)
307     {
308         printf("Sweep configuration not available\n");
309     }
310     else
311     {
312         float start_m = 0.1f;
313         float length_m = 0.7f;
314         float update_rate_hz = 100;
315         acc_sweep_configuration_sensor_set(sweep_configuration,
316             id);
317         acc_sweep_configuration_requested_start_set(
318             sweep_configuration, start_m);
319         acc_sweep_configuration_requested_length_set(
320             sweep_configuration, length_m);
321         acc_sweep_configuration_repetition_mode_streaming_set(
322             sweep_configuration, update_rate_hz);
323     }
324 }
325

```

C.2 得られるデータの例

データはCSV形式で保存される。図に得られるデータテーブルの例を示す。1行目に距離[m]の値、1列目に時間[s]の値が記録される。2行、2列目以降はすべて電波の反射強度となっている。

	A	B	C	D	E	F	G
1		0.099768	0.106802	0.113836	0.12087	0.127904	0.134937
2	0	128	104	168	304	272	272
3	0.012	272	336	104	96	192	216
4	0.02	56	144	296	304	144	232
5	0.03	120	192	144	112	280	296
6	0.04	120	120	96	112	184	136
7	0.05	184	264	168	184	264	312
8	0.06	256	232	16	128	240	296
9	0.07	208	368	192	120	176	264
10	0.08	168	168	144	168	208	248
11	0.09	200	280	200	168	144	144
12	0.1	168	272	256	232	232	160
13	0.11	256	528	272	80	152	176
14	0.131	336	472	368	208	80	144

図C.1 得られるデータテーブルの例

C.3 Simulinkにデータを転送するMATLABコード

SimInを使用してSimulinkに読み込ませる。Simlink側ではfromWorkspaceブロックを使用してデータを取り込む。データは多次元配列として取り込まれる。

```
1 clc;clear;
2
3 %% Setting the import range
4 start_time = 15;
5 end_time = 35;
6
7 %% Select data to read
8 select_data = 'test_data/non-LPF-High';
9
10 %% Read RADRA data
11 % Generate matrix data from CSV
12 radar_raw_data = readmatrix(strcat(select_data, '/3.csv'));
13 radar_raw_data(end,:) = [];
14 radar_amp_data = radar_raw_data(2:end,2:end);
15 radar_time_data = radar_raw_data(2:end,1);
16 radar_dis_data = radar_raw_data(1,2:end);
17
18 %% Read acceleration data
19 acc_raw_data = readmatrix(strcat(select_data, '/acc.csv'));
20 acc_raw_data(end,:) = [];
21 acc_time_data = acc_raw_data(:,1);
22 acc_data = acc_raw_data(:,2);
23
24 %% Generate data for Simulink to read
25 SimIn = createSimuIn(radar_time_data, radar_amp_data);
26 simin_dis = createSimuIn(0, radar_dis_data);
27 simin_acc = createSimuIn(acc_time_data, acc_data);
28
29 %% Draw graph
30 subplot(2,1,1)
31 mesh(SimIn.time, radar_dis_data, SimIn.signals.values.')
32 view(2)
33 subplot(2,1,2)
34 plot(radar_dis_data, SimIn.signals.values)
35 % plot(simin_acc.time, simin_acc.signals.values)
```

```
36
37 %% Parameters
38 stopTime = SimIn.time(end);
39 indexToMeter = radar_dis_data(1, 2) - radar_dis_data(1, 1);
40
41 %% Generate SimIn
42 function SimIn = createSimuIn(times, vals)
43     SimIn.signals.values = vals;
44     SimIn.signals.dimensions = size(SimIn.signals.values, 2);
45     SimIn.time = times;
46 end
```

参考文献

- [1] 武馬修一・趙在成・神田亮・梶野英紀・土田久輔・十津憲司・大谷佳史. 電動アクティブサスペンションアクチュエータの開発. 自動車技術会論文集. 2008, 39, 5, p. 13.
- [2] 自動運転LAB. 【最新版】自動運転の最重要コアセンサーまとめ LiDAR、ミリ波レーダ、カメラ. https://jidounten-lab.com/y_2520, (参照:2020-01-24)
- [3] 松ヶ谷和沖. 自動運転を支えるセンシング技術. Denso technical review. 2016, 21, p. 13-21.
- [4] MIT Technology Review. Self-Driving Cars' Spinning-Laser Problem. <https://www.technologyreview.com/s/603885/autonomous-cars-lidar-sensors/?set=603886>, (参照:2020-01-24)
- [5] ZMP.Autonomous Driving (自動運転) の制御に使われるセンサについて. https://www.zmp.co.jp/knowledge/ad_top/dev/sensor, (参照:2020-01-24)
- [6] CleanTechnica. Tesla & Google Disagree About LIDAR — Which Is Right?. <https://cleantechica.com/2016/07/29/tesla-google-disagree-lidar-right/>, (参照:2020-01-24)
- [7] 梶原昭博 (2019) . ミリ波レーダー技術と設計. 科学情報出版.
- [8] 総務省. 周波数帯ごとの主な用途と電波の特徴. <https://www.tele.soumu.go.jp/j/adm/freq/search/myuse/summary/>, (参照:2020-02-06)
- [9] 藤村契二. 車載用ミリ波レーダの実用化技術. 電気学会論文誌. 1998, 118, p. 292.
- [10] Acconeer. EVK Getting Started Guide. <https://developer.acconeer.com/download/evk-getting-started-guide-pdf/>, (参照:2020-02-11)
- [11] Acconeer. XC112 Product Brief. <https://developer.acconeer.com/download/xc112-product-brief-pdf/>, (参照:2020-02-11)
- [12] Acconeer. XC112_XR112 User Guide. <https://developer.acconeer.com/download/>

xc112_xr112-user-guide-pdf/, (参照:2020-02-11)

- [13] Acconeer. XR112 Product Brief. <https://developer.acconeer.com/download/xr112-product-brief-pdf/>, (参照:2020-02-11)
- [14] The MathWorks,Inc. Model-Based Design with Simulink. <https://jp.mathworks.com/help/simulink/gs/model-based-design.html>, (参照:2020-02-10)
- [15] The MathWorks,Inc. Getting Started with MATLAB. <https://jp.mathworks.com/help/matlab/getting-started-with-matlab.html>, (参照:2020-02-10)
- [16] The MathWorks,Inc. Getting Started with Simulink. <https://jp.mathworks.com/help/simulink/getting-started-with-simulink.html>, (参照:2020-02-10)
- [17] PIUS. PIUS - KIT CAR. <http://www.pius-kitcar.com/pius.html>, (参照:2020-02-26)
- [18] YDLIDAR. YDLIDAR X4 Datasheet. <http://www.ydlidar.com/Public/upload/files/2019-12-18/YDLIDAR%20X4%20Datasheet.pdf>, (参照:2020-01-18)

謝辞

本研究を遂行し, 論文を執筆するにあたり, 多大なご支援とご指導を賜りました指導教官である一関工業高等専門学校 秋田敏宏准教授に深く感謝の意を表します. ご指導を通じて身に着けたことを今後の研究活動に生かし, 進学先でもさらなる成果を挙げられるよう努力していく所存です.

また, 測定データをMATLABからSimulinkへ転送するサンプルコードを提供していた
だいたアルプスアルパイン株式会社古川開発センター八鍬信康氏には心より感謝いたし
ます.

そして, 同研究室の学生の皆様にも貴重なご意見, アドバイスをいただきました. 本当に
にありがとうございました.