国際VHF通信とAIS情報の対応付けに関する基礎研究

修士学位論文

国際 VHF 通信と AIS 情報の対応付けに関する基礎研究

平成 30 年度
（2018 年 9 月）

東京海洋大学大学院
海洋科学技術研究科
海運ロジスティクス専攻

善住　大輔

Abstract

Marine VHF will be an important factor for study of marine traffic and USV (Unmanned Surface Vehicles). For example, sometimes communications between vessels using Marine VHF will not be maneuvering to the law. This influences marine traffic. Because there are cases like this, we must collect a lot of Marine VHF radio data and study it. Also, for USV and vessels to coexist, it is necessary for an USV to know Marine VHF's originating vessel. This is to avoid pleasure boats and fishing boats. Additionally, USV has a problem that it must be able to deal with distress messages from Marine VHF without DSC function. These problems can be solved by knowing the location of the caller.

This is a basic study to associate Marine VHF and AIS information. In this study, as the distance gets away, we used the feature that the sound noise becomes large and estimated the distance by sound noise. First, we confirmed that there is a correlation between sound noise and distance. This was confirmed using a specific low power radio instead of Marine VHF. The power spectrum of the voice received by the specified low power radio unit becomes strong between about 256 Hz and 4 kHz. The volume of the sound noise increases as the distance increases. The volume of the sound voice does not change until the distance where the radio wave arrives steady. If that distance is exceeded, the volume of the sound increases as the distance increases, and it approaches the sound volume of the sound noise. As the distance increases, the power spectrum of the noise becomes stronger from about 4 kHz toward the lower frequency, and finally the power spectrum of the noise between about 256 Hz and 4kHz becomes stronger.

And, tried Marine VHF's sound to learn using MLP (Multi-Layer Perceptron), accuracy is not good, but correlation between Marine VHF's sound and distance was observed.

目次

第1章　はじめに

長年の間、世界中で海上交通流や海上交通に関する様々な研究が行われており、それによって船舶事故の減少や安全性の向上、効率的な運航、船員の負担の減少などが行われてきた。そして、近年は自動運航船の開発が盛んに進められており、より安全性が高くて効率的な運航を目指して研究が進められている。しかし、日本においての自動運航船の研究は、世界から見たときに後れを取っており、海洋国家日本が世界をリードするようなイニチアチブをとるには、今後精力的な研究が必要である。

自動車の自動運転に注目をすると、自動運転を実現するために人工知能（AI）の研究が積極的に進められている。自動車にたくさんのセンサーを設置してエンジンなどの自動車自体の情報や、周辺環境の障害物や白線、標識といった外部の情報を収集して、自動車に関するビッグデータを構築して分析をしている。近年は GPU（Graphics Processing Unit）などでコンピュータの計算処理能力が向上し、ビッグデータをディープラーニングすることができるようになった。そして、このディープラーニングによって自動運転の研究が飛躍的に進展し、もう間もなく自動運転車が普及する直前まで来ている。

ただし、自動運転車が普及するにあたって解決しなければならない問題がある。それは、人間との共存方法である。人が自動車を運転するときは、アイコンタクトやパッシング、手による指示などが行われており、そういったコミュニケーションが今後の自動運転車には必要となる。もちろん、自動運転車から一方通行なコミュニケーションではなく、自動運転車を取り巻く人間からのコミュニケーションに対しても対応しなくてはならない。

船舶についても自動車業界と同じような流れが起きている。AIS（Automatic Identification System）を搭載した船舶が増え、近年は船舶 IoT（Internet of Things）や海洋 IoT が進み、船舶や海洋についてのビッグデータの構築は着実に進められている。日本が自動運航船で世界の上位になるには船舶 IoT や海洋 IoT などから得た、より多種多様で膨大な量のビッグデータを構築する必要がある。また、自動運航技術だけでなく、海上交通に関する研究をより進めていくことは、よりインテリジェンスな自動運航船の開発につながるため、今後ますます海上交通に関する研究が必要であると考える。

無論、人間との共存についても考える必要がある。軍事目的の船舶や商船などは自動運航船に置き換えることができたとしても、漁船やプレジャーボートといった船舶は自動運航船になることは考えにくい。港湾においてはこういった船舶が多く、自動運航船はこれらの船舶とのコミュニケーションをとることで、避航操船を行う、遭難への対処を行うなどの行動ができるようにならなければいけない。

船舶間におけるもっとも有効なコミュニケーションツールは国際 VHF 無線通信（以下、国際 VHF とする）であり、自動運航船と非自動運航船が国際 VHF でコミュニケーションをとれるようになることが必要である。国際 VHF でコミュニケーションをとるためにも、国際 VHF が海上交通にどのような影響を及ぼしているのか、どうコミュニケーションをとることで共存ができるのかを研究する必要があり、海上交通に関する研究で国際 VHF に着目することは今後重要である。

国際 VHF の通信内容を使った海上交通の研究はすでにいくつかある [1, 2]。国際 VHF 通信が音声データである以上、通信内容を分析するには人間が音声を聞いて内容を理解する必要があるが、国際 VHF 通信は携帯電話に比べてノイズが多くクリアな音質ではないため、正確に内容を聞き取りためには繰り返し聞く必要がある。また、彼らの研究では国際 VHF 通信を分析するのに重要な情報として、発信船舶の位置が重要となってくる。そこで、国際 VHF 通信から発信船舶の位置を自動的に推定し、AIS やレーダーなどの情報と組み合わせて分析を進めることができれば、よりスムーズな海上交通に関する研究、さらには自動運航船の研究につながると考える。

国際 VHF 通信とそれを発している船舶(発信船舶)の情報、すなわち、AIS やレーダーの情報を対応付ける方法は 3 つ考えられる。

第一に音声認識によって国際 VHF 通信の内容に含まれる船名から航行している位置を推定し、発信船舶を特定する方法が考えられる。この方法は主に船名によってデータを抽出可能な AIS に適用することが可能である。この方法を行うには国際 VHF で用いられる特殊な用語に対するビッグデータを用意して音声認識技術の精度を高める必要がある。

第二に指向性アンテナと電波の受信強度を用いて国際 VHF 通信の距離と方位を推定し発信船舶を特定する方法が考えられる。この方法は AIS に限らずレーダーの情報との対応付けにも用いることができ、AIS 非搭載の船舶にも対応できる。この方法を行うには、電波法に抵触する装置の改造を必要とする。

第三に音声解析を行い発信船舶の距離を推定する方法が考えられる。この方法は、国際 VHF 通信に含まれる雑音と音声の関係から、距離を推定するものであり、上に挙げた 2 つの方法に比較して、実現が容易であり、本研究ではこの方法について研究を行うこととした。

第 2 章で研究背景・目的を述べ、第 3 章で音声データの分析方法を説明する。第 4 章では国際 VHF 通信から得た音声ノイズと距離の関係について、第 5 章では特定小電力トランシーバーによる音声ノイズと距離の分析について述べる。そして、第 6 章では第 5 章の実験方法で大量の音声データを使った音声ノイズと距離の分析を述べて、第 7 章では第 6 章で得た音声データを使った、多層パーセプトロンによる音声ノイズによる距離推定モデルの作成について話す。そして、第 8 章で総括する。

第2章　研究背景

国際 VHF 通信と AIS 情報を対応付けることは非常に重要であり、これからの時代に必要なシステムであると考える。本章では海上交通に関する研究、及び自動運航船において国際 VHF 通信の重要性を述べた後、AIS 情報の対応付けについて述べる。そして、国際 VHF 通信と AIS 情報を対応付ける手法を 3 つ提案し、その中で音声のノイズ量による距離推定を行った理由を説明する。

2.1　海上交通に関する研究における重要性

今日まで海上交通に関する研究が積極的になされている。海上交通を研究することで、船舶事故の削減やヒューマンエラーを軽減するといった船舶交通の安全や、輻輳海域での交通整理、経済的な運航方法に関して寄与している。

2.1.1　船舶・海洋に関するビッグデータの必要性

近年ビッグデータが注目を集めている。そして、そのビッグデータを統計学や人口知能などといった手法でデータマイニングすることにより、これまで分からなかった各データの相関や傾向などを分析することが様々なところで行われている。当然のことながら、海上交通に関する研究においてもビッグデータを集めて、データマイニングすることによって、更なる海上交通の発展に寄与すると考えられ、船舶や海洋におけるビッグデータを収集するためのシステムの構築は非常に重要であるといえる。

船舶のビッグデータの一例として AIS を挙げる。AIS は 2002 年 7 月 1 日に発効された「1974 年の海上における人命の安全に関する条約（SOLAS74）」第 V 章により、国際航海に従事する 300 総トン数以上のすべての船舶、すべての旅客船、国際航海に従事しない 500 総トン数以上の貨物船のいずれかに該当する船舶は、AIS を搭載することが義務付けられている。これにより船舶の識別符号、船種、位置、針路、速力、航行状況及びその他の安全に関する情報を入手することが可能になった。また、AIS 受信機を衛星に搭載した衛星 AIS によって遠洋を含む地球全域の AIS 情報を取得することができるようになった [3]。そして、リアルタイムな AIS 情報を Web サイトで見ることができる「Marine Traffic [4]」や、過去の大量の AIS 情報などから得られた航行履歴を提供するサービス [5]などといったビジネスが登場し、誰もが簡単に世界中の AIS 情報を入手することができるようになっている。

海上交通の研究ではよく AIS を用いて、船舶の航行に関する分析やシミュレーションなどが行われている。例えば、滝本の研究 [6]では、海上交通流シミュレーションに AIS 情報を活用することで海上交通流の変化の予測、衝突危険な状態や見合い関係の発生頻度の推定に成功した。これには 1 週間分の AIS 情報が用いられた。また、海上交通以外にも AIS 情報は使われており、Google と国際的な海洋保護団体 Oceana、SkyTruth は「Global Fishing Watch [7]」という Web サイトを立ち上げ、世界の漁業活動を可視化して持続可能な漁業と水産資源の管理をするプロジェクトを行っている。これは AIS 情報で漁船の動きをディープラーニングして、漁船の種類や漁具の判定をしている。

AIS 情報の他にも船舶に関する情報はいくつかあり、それらを収集して船舶・海洋ビッグデ

ータを構築して、解析や利用をしている。例えば、庄司は海洋ブロードバンドネットワークによる船陸協調運航を目指して、2009 年に「先端ナビゲートシステム（Advanced Navigation System）」を構築し、船舶運航に関する多種多様な情報を収集、解析などしている [8]。先端ナビゲートシステムは東京湾のレーダー画像、ARPA 情報、東京湾 AIS 情報、東京湾風向風速情報、東京湾 Web カメラ情報、沿岸波浪数値予報モデル情報、全球波浪数値予報モデル情報、船舶航海情報、船舶機関情報、船舶レーダー情報、船舶 Web カメラ情報、全球海図情報などをデータベースに蓄積しており、海洋 GIS（Geographic Information System）で複数の情報を重畳して表示することで、これらのデータを組み合わせた解析・分析ができる。

近年は自動運航船の研究開発が盛んに行われており、そのような背景からも船舶・海洋に関する情報を収集してビッグデータを構築することは必要であり、このビッグデータをデータマイニングすることによって、海上交通や水産資源管理などに大きく寄与すると考える。

2.1.2　国際 VHF 通信ビッグデータの必要性

瀬田らの研究 [1]や田崎らの研究 [2] によると、AIS 搭載義務により国際 VHF 通信による避航意思の確認が増加しているが、意思疎通が取れないケースがあるという。そこで彼らの研究では船舶間コミュニケーションをどのように行われているかを調査するため、AIS 情報の取得と国際 VHF 通信の聴取を行い、国際 VHF 通信で聞き取った船名から AIS 情報と照らし合わせることで、どういった種類の船舶が、どのような位置関係の時に、どういう通信をするのかを分析した。

瀬田らの研究では、事前に国際 VHF 通信を行うことにより操船行動の意思を確認することで、操船者の危険に対する許容範囲が拡大されて、本来なら危険と感じる状況でも相手船に接近するような操船を可能にすることが分かった [1]。田崎らの研究では、輻輳海域や航路等が指定されている海域では航法に従った避航行動を行うことができない場合があることから、最接近から 15～20 分前などかなり早い段階で通信を行って二船間で操船意図を確認していたことが分かった。また、同航する二船間について相対的な位置関係について通信が多いことが分かった [2]。これらから、国際 VHF 通信によって通常では危険と思われる操船や、航法に従わない操船が行われる可能性があるということが言える。つまり、国際 VHF 通信はイレギュラーな操船に使われることがあり、海上交通において国際 VHF 通信を解析することは、イレギュラーな操船に対しての分析にも非常に有効であると思われる。よって、国際 VHF 通信のビッグデータを構築することは、海上交通の研究に寄与すると考える。

2.2　自動運航船での可能性

自動運航船の研究開発が積極的に進められている中、自動運航船と非自動運航船の共存方法が問題として挙がっている。本研究では、国際 VHF 通信は自動運航船において重要なファクターの 1 つであり、AIS、レーダー、カメラなどの各種船舶データに加えて国際 VHF 通信を対応付けることで、よりインテリジェンスな自動運航船になると考える。

2.2.1　自動運航船の動き

近年、世界中で無人船や自動運航船の開発が進められている。日本においては国土交通省海事局が主体となって、船舶・船舶機器のインターネット化（IoT）・ビックデータを活用した安全性・効率性の高い船舶「IoT 活用船」や液化天然ガス等の環境に優しい代替燃料に対応した「代替燃料船」など先進船舶の導入等を推進しており [9]、IoT 活用船から自動運航船へと段階的に技術の開発・実用化を目指している [10]。

アメリカ国防高等研究局（DARPA）は ACTUV（Anti-Submarine Warfare Continuous Trail Unmanned Vessel）プログラムで建造された「Sea Hunter 図 1　Sea Hunter (ACTUV)」（図 1）の試験が成功し、アメリカ海軍研究局（ONR）に正式に移管したことを発表した [11]。Sea Hunter は自律航行が可能な対潜水艦というミッションを行う無人実験艇であり、近いうちに実験段階から軍事運用に移行するとみられる。

図 1　Sea Hunter (ACTUV)

（出典）Wikipedia「Sea Hunter」https://en.wikipedia.org/wiki/Sea_Hunter

ノルウェーの Kongsberg Maritime と YARA International は世界初のゼロエミッション、自律航行するコンテナ船「YARA Birkeland」を共同開発しており、2020 年の完全自律航行を目指して開発を進めている [12]。また、Kongsberg Maritime と Wilhelmsen は自律航行船のデザインや開発、コントロールシステムやロジスティクスなど、自動運航船におけるバリューチェーンを行う会社「Massterly」を設立すると発表した [13]。ノルウェーの海運関連企業は商船の自動運航化に向けて、企業が積極的に動いているのが見て取れる。

EU では 2012 年から 2015 年にかけて「MUNIN（Maritime Unmanned Navigation through Intelligence in Networks）」と呼ばれる、自動運航船の運用に関する技術コンセプトの開発と技術的、経済的、法的な面での検証をするプロジェクトを行った [14]。MUNIN は結論として次のような結論を出している。

1. 自動運航船は海運会社の収益性を高める可能性を秘めている。
2. 無人船は有人船に比べて衝突・沈没のリスクが 10 倍減少する可能性があるが、無人船はサイバー攻撃や海賊のリスクが懸念される。
3. 無人船の法律についても責任問題や船舶の設計基準等の懸念事項はあるが、法的に大きな障害にはならない。

よって、無人船はより持続可能な海運業化の目的に貢献することができると結論付けている。

ヨーロッパの企業や大学が協力して自動運航船の研究開発を行っている「AAWA（Advanced Autonomous Waterborne Applications Initiative）」は、技術、安全とセキュリティ、社会と法律、経済とビジネスモデルの 4 つについて研究している [15]。AAWA は遠隔及び自動運航船を作る技術はすでにあり、安全性についても既存の船舶と同じくらい安全である。法律については責任問題を含む、様々な運航シナリオのケーススタディを調査した上で、法律の変更ができる。また、遠隔及び自動運航船はこれまでの海運業界とその中での役割とは異なるものであるとされていると言っている。AAWA に参加している Rolls Royce は、2020 年頃には遠隔及び自動運航船を限定海域、2025 年頃までには近海域、それ以降は外洋で運航することを目指している [16]。

中国では 2017 年に中国船級社（CSS）と珠海市政府、武漢理工大学、雲洲智能公司が共同で中国初の小型無人スマート貨物船プロジェクトを開始した [17]。そして、2018 年 2 月には珠海市沖に無人船試験場の建設に着手した [18]。無人船試験場の総面積は約 770 平方キロメートルで、完成すれば世界最大でアジアでは初めての無人船試験場となる。中国は陸地から遠く離れた場所に広がる島の補給船を無人化することを目標としており、無人船にすることで貨物用のスペースがより広くなり、無人化に伴う人件費を削減で、補給にかかる高い費用を削減できると考えている。そして、中国の経済・外交権構想「一路一帯（One Belt, One Road）」に参加している、東南アジアなどの多くの島を持つ国で大きなマーケットを作るとみられる。

小型船舶については USV（Unmanned Surface Vehicles）や ASV（Autonomous Surface Vehicles）といった遠隔または自動運航船がすでに存在し、世界中で使用されている。風力で航行できる「SAILDRONE [19]」（図 2）や海軍向けの「MAST（Maritime Autonomy Surface Testbed） [20]」（図 3）があり、海洋観測や魚の数の推定、偵察任務といった軍事目的など、様々な用途で使用されている。日本でも 2016 年に海上保安庁が海洋観測のために、Maritime Robotics の「Wave Glider (R)」という波と太陽エネルギーのみで動く USV を導入している [21, 22]。

このように自動運航船は世界中で国、民間問わず、技術開発から法律に至るまで、着々と研究が進められており、小型船舶や軍事目的に至ってはすでに実用化されているものもある。自動運航船が世界中で運航される時代がやってくるのも、時間の問題といえるだろう。

図 2　SAILDRONE

（出典）Wikipedia「Unmanned surface vehicle」

https://en.wikipedia.org/wiki/Unmanned_surface_vehicle

図 3　MAST（Maritime Autonomy Surface Testbed）

（出典）Wikipedia「Unmanned surface vehicle」

https://en.wikipedia.org/wiki/Unmanned_surface_vehicle

2.2.2　自動運航船と非自動運航船の衝突予防での利用

2.2.1 で述べたように世界中で自動運航船の開発が進められているが、自動運航船を普及するにあたり、すべての船舶が一気に自動運航船に切り替えられるのは難しく、必ず自動運航船と非自動運航船が混在する期間が生じる。また、一般に港湾には大型船舶だけでなく、プレジャーボートや小型漁船といった小型船舶が存在し、これらが自動運航船に切り替わるのは考えにくい。つまり、自動運航船が普及したとしても、必ず自動運航船と非自動運航船が混在する。

船舶同士が衝突をしないようにするためには、見張りが不可欠である。有人船では目視やレーダー、AIS といった見張りにより衝突を回避しており、必要に応じて国際 VHF 通信を用いてお互いが意思疎通を図り、衝突を回避している。自動運航船はレーダーやカメラ、AIS 等の様々なセンサーを組み合わせ、海上衝突予防法に従って衝突を避けるが、非自動運航船と意思疎通をとって避航することができないと想定される。しかし、非自動運航船から見た場合、自動運航船がどのタイミングで、どのように避航動作を行っているかを事前に知ることができない。そこで、自動運航船の意思表示をするために、自動運転車が歩行者や他のドライバーと意思疎通を図るために、音やライト、動き、LED 表示板などを用いるような形で [23]、自動運航船が他の操船者に対して意思表示をすることが必要と思われる。ただし、これは自動運航船から非自動運航船に対する意思表示のみである。

自動運航船と非自動運航船が意思疎通をするには、国際 VHF 通信が有効である。しかし、AIS 非搭載船舶は船名と位置情報が対応した情報が提供できない。そこで、国際 VHF 通信とレーダーやカメラなどの情報を対応付けることによって、どの位置にいる船舶から受信したかを推測することができると考える。

2.2.3　非自動運航船の遭難における捜索、救助での利用

海難が発生した際には陸や周囲の航行船舶に救難信号を発信する。SOLAS74 条約が適用される船舶は GMDSS（Global Maritime Distress and Safety System）が導入されており、DSC（Digital Selective Calling; デジタル選択呼出装置）付き国際 VHF 機器の装備が義務付けられている [24]。DSC 機能（ch70）は遭難の際にボタン操作のみで遭難警報を発することができ、GPS 等の外部機器が接続されているものは位置情報も同時に発信する [25, 26]。

しかし、総トン数 100 トン未満のプレジャーボートや小型漁船といった小型船舶は、DSC 付き国際 VHF 機器の搭載は任意であり、このような小型船舶で海難が発生した場合、ch16 にて遭難通報をしなければならない。もし、遭難した船舶が位置情報を伝えることができない、もしくは聞き取れなければ捜索範囲が広くなってしまい、人命のリスクが増加すると想定される。だが、遭難通報の内容や電波強度・品質から位置を推定することができれば捜索範囲を特定することができ、周辺を航行する船舶が自動運航船だとしても、遭難者の下へ駆け付けて救命艇やいかだを出すなどの対応が可能ではないかと考える。

2.2.4　国際 VHF 通信と各種センサー情報の対応付けの重要性

2.2.2、2.2.3 の例で述べたように、国際 VHF 通信は自動運航船と非自動運航船と共存する上で必要であると考える。現在の自動運航船の研究で使用されている AIS やレーダー、カメラなどの各種センサーと融合させることはインテリジェンスな操船に有益である。また、自動運航船が安全航行をするためにも国際 VHF 通信を傍聴する必要がある。そのためにも、国際 VHF 通信の発信者の特定、通信内容の理解などが求められる。

2.3　オートマチックな国際 VHF 通信と AIS 情報の対応付け

2.1 にて国際 VHF 通信のビッグデータ構築の重要性、2.2 にて国際 VHF 通信と各種センサーなどの情報との対応付けの重要性を述べた。以上の点から、オートマチックに国際 VHF 通信の分析ができる必要があるといえる。2.1 の海上交通の分析では自動的に国際 VHF 通信と AIS 情報が対応付けられることによって、よりスムーズで大量のデータ分析をすることが可能になると思われる。また、2.2 の自動運航船においても AIS 情報との対応付けは必要であると考える。よって、本研究ではオートマチックな国際 VHF 通信と AIS 情報の対応付けを行うシステムを研究することにした。

オートマチックに国際 VHF 通信と AIS 情報を対応付ける手法として、音声認識による手法、電波伝搬距離と指向性アンテナによる手法、音声ノイズによる手法の 3 つを提案する。

2.3.1　提案 1：国際 VHF 通信の音声認識による対応付け

船舶は国際 VHF 通信で通話することにより船舶間コミュニケーションを図っている。その通話内容には、船名やコールサインなどの通話する船舶を特定する情報と、その船舶間でやりとりをする通話内容の情報が含まれている。それらを音声認識することで、発信者と受信者の船舶を特定し、AIS 情報と対応付ける手法を提案する。なお、その概要を図 4 に示す。

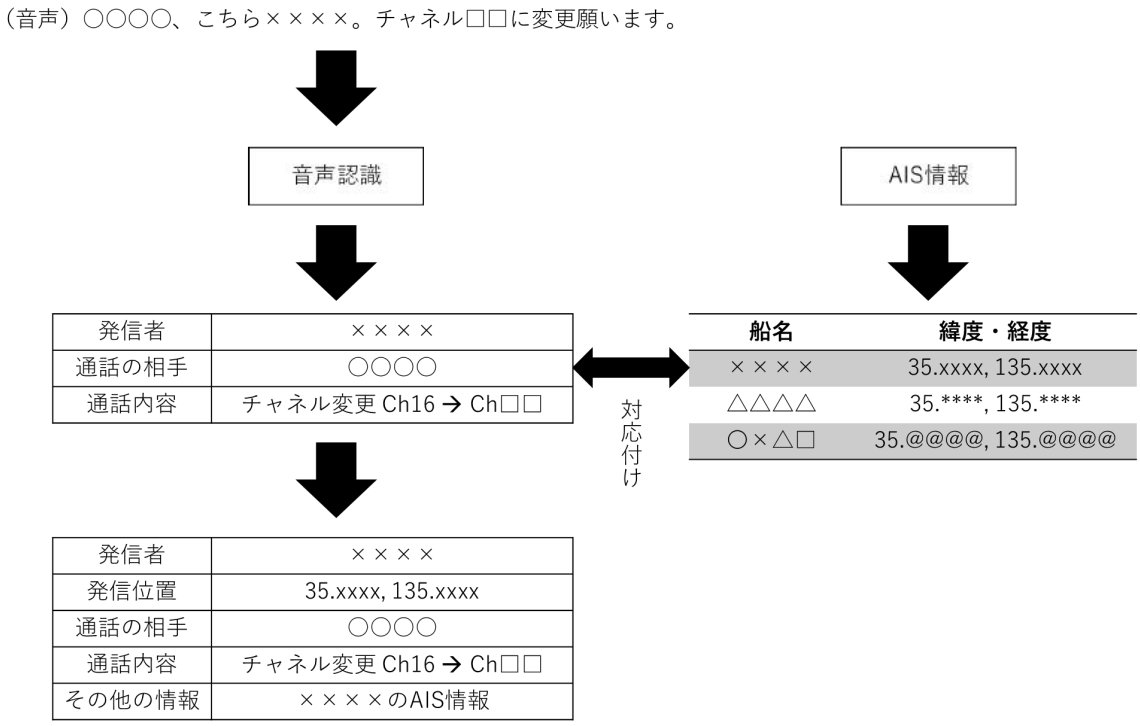

図 4　国際 VHF 通信の音声認識による対応付けの概要

音声認識技術の発達と可能性

近年は音声認識技術が発達してきている。特に Apple や Google、Microsoft、IBM、Amazon などは音声認識を積極的に研究開発、商品化を進めており、我々は日常生活においてスマートフォンやスマートスピーカー（図 5）など音声認識や自然言語処理などを用いたバーチャルア

シスタントを利用する機会が増えてきている。

図 5　スマートスピーカー（左: Amazon Echo, 右: Google Home）
（出典）Wikipedia「スマートスピーカー」https://ja.wikipedia.org/wiki/スマートスピーカー

Mary Meeker によると 2016 年においてモバイル検索の 20%は音声検索で行われており、調査会社の ComScore は 2020 年までにインターネット検索の 50%は音声検索を通して行われると予想している [27]。つまり、音声認識技術は日常で利用されるほどに高精度になってきていることが分かる。

最近では Google が「Duplex」という AI が自然な会話で電話予約を代行するシステムを発表した [28]。これはオンライン予約に対応していない、電話でのみ予約や注文ができるレストランや美容院などに、AI システムが代行して電話で予約、注文のやりとりをするシステムである。AI の音声は人間のようなイントネーションで会話をし、途中で相槌を打つといった対応をするため、あたかも人間が予約の電話をかけたように感じるという。Duplex は「TensorFlow Extended（TFX） [29]」で構築した RNN（再帰型ニューラルネットワーク）でトレーニングされている。これは Duplex が予約や注文に関する会話を認識、理解できるということであり、ディープラーニングの会話の種類の幅を増やすことで、より専門性の高い会話が必要なところでも応用が可能である。

Google の「Cloud Speech API [30]」は最新のディープラーニングのニューラルネットワークアルゴリズムを用いて、音声を高精度で認識する Web API である。110 の言語と方言を認識し、録音された音声だけでなくリアルタイムストリーミング音声も処理することができる。特に優れているのはノイズ低減をすることができ、雑音の多い音声もノイズ除去をすることなく処理することができるとされている。Google の他にも Microsoft の「Bing Speech API [31]」、IBM の「Speech to Text [32]」、Amazon の「Amazon Transcribe [33]」といった Web API が各社で提供されており、これらも Cloud Speech API と同様に音声を高精度で認識することができる。そして、Web API で提供されているため、インターネットに接続できるシステムに簡単に音声認識機能

を組み込むことができる。しかし、日常で使われている言葉に対しては高い認識率であるが、専門用語などの日常で使われないような言葉に対しては、発音の似ている日常で使われている言葉で認識することが多く、専門用語の学習ができていないと考えられる。現状として海事用語などを正しく認識するのは難しいが、開発者が海事用語を学習させることができれば、海事での利用は考えられる。

先はディープラーニングを用いた最新の音声認識機能について述べたが、すでに音声認識技術は多くの研究機関や企業が様々な手法によって研究開発をしている。日本においても「Julius [34]」というオープンソースで高性能な大語彙連音声認識エンジンが、京都大学と情報処理振興事業協会､奈良先端科学技術大学院大学､名古屋工業大学によって研究開発されている｡Julius は PC やスマートフォン上で、数万語の連続音声認識をほぼリアルタイムで実行でき、単語辞書や言語モデル、音響モデルなどの各モジュールを組み替えることで、小語彙の音声対話システムからディクテーションまで多くの用途に応用することができ、オフライン環境においても音声認識することができる。専門用語といった一般に使われない言葉についても、ユーザーが単語辞書や言語モデル、音響モデルをカスタマイズすることで認識させることが可能であり、現時点で海事用語などを音声認識させるには適していると考える。また、オフライン環境で使用することが可能であることから、衛星通信を用いたインターネット通信しかできない環境においても、国際 VHF 通信を音声認識できるのではないかと考える。しかし、専門用語の単語を登録することは容易であるが、言語モデルや音響モデルまでカスタマイズするには、大量の学習データが必要とされる。また、多言語対応についての課題がある。

オフラインで音声認識するという点では Android に「SpeechRecognizer [35]」という API がある。これは通常 Google のサーバーと通信して音声認識を行っているが、オフラインの言語パッケージをダウンロードすることでオフラインでの音声認識が可能である。しかし、Google の Could Speech API と同様に専門用語などを学習していないため、現時点では海事用語などの音声認識には向かないが、将来的に海事用語が学習されれば利用価値はあると思われる。

音声認識技術はディープラーニングの登場後、非常に高精度で音声を認識することができるようになった。海事用語といった専門用語についての認識率も、このさき学習されることによって音声認識することは可能であると予測される。また、現状において限られた専門用語の音声（単語）を認識することができると考えられ、今後は国際 VHF 通信の通信内容を文字に起こすことが可能なのではないかと考える。

国際 VHF 通信の音声認識による対応付けの課題

国際 VHF 通信の音声データは、国際 VHF 通信の発信機及び受信機自身が発生させる電磁ノイズ、混信、気象海象、発信者の周囲の環境音ノイズなどにより多くのノイズが混入することや、話者の発音やスピード、訛りなどの影響によって、聞き手が音声を聞き取れないことがある。

無線機にはノイズスケルチのような、無信号時のノイズを遮断する機能がある。無信号時のノイズを遮断すると受信した時のみ音声が聞こえることから、どのタイミングで受信したかが分かり、国際 VHF 通信と AIS 情報を対応付ける重要な手がかりとなる。だが、受信した信号

に強いノイズが混入している場合は聞き手が音声を聞き取れないこともあり、通信している船舶の特定が困難になる場合がある。また、ノイズスケルチ機能は信号を受信後、その信号がある一定以上の電波強度があるかを判断した後にスケルチが解放されるため、受信から音声を出力するまでに若干の遅延が生じる。発信者の会話のタイミングが発信と同時、もしくは発信より若干早いと、会話の最初の部分が途切れるといった現象が生じる。無線通信において相手船や VTS（Vessel Traffic Service）を呼び出す際、初めに相手船や VTS の名前を呼び、次に自船の名前を言うため、このようなことが起きると誰を呼び出しているのかを特定できない場合がある。

本研究でこのシステムを開発するには、まず初めに人間が音声を聞き取ることができなければいけないが、以上の点から人間が聞き取れる音声は限られている。また、人間が聞き取れない音声を音声認識するのは非常に難しいと考えられる。通話内容などを分析するために音声認識をすることは大切ではあるが、AIS 情報との対応付けだけに焦点を向けた場合に、船名を聞き取ることができなければ対応付けられないことか、この手法を研究する時点で大きな課題があると思われる。

また、同時に複数の通信があった場合、混信してしまうために音声を得ることができない。よって、同時通信に対して本手法は有効でないと考えられる。

2.3.2　提案 2：電波伝搬距離と指向性アンテナによる対応付け

電波の強さは自由空間において距離の 2 乗に比例して減衰する [36]。そこで、受信した電波の強さから発信者と受信者の距離を求め、AIS の位置情報などと対応付ける手法を提案する。この手法は音声認識を用いた手法とは違い、AIS 情報以外のレーダーなどの情報を利用することができるため、AIS 非搭載船にも有用性があると考える。なお、そのイメージを図 6 に示す。

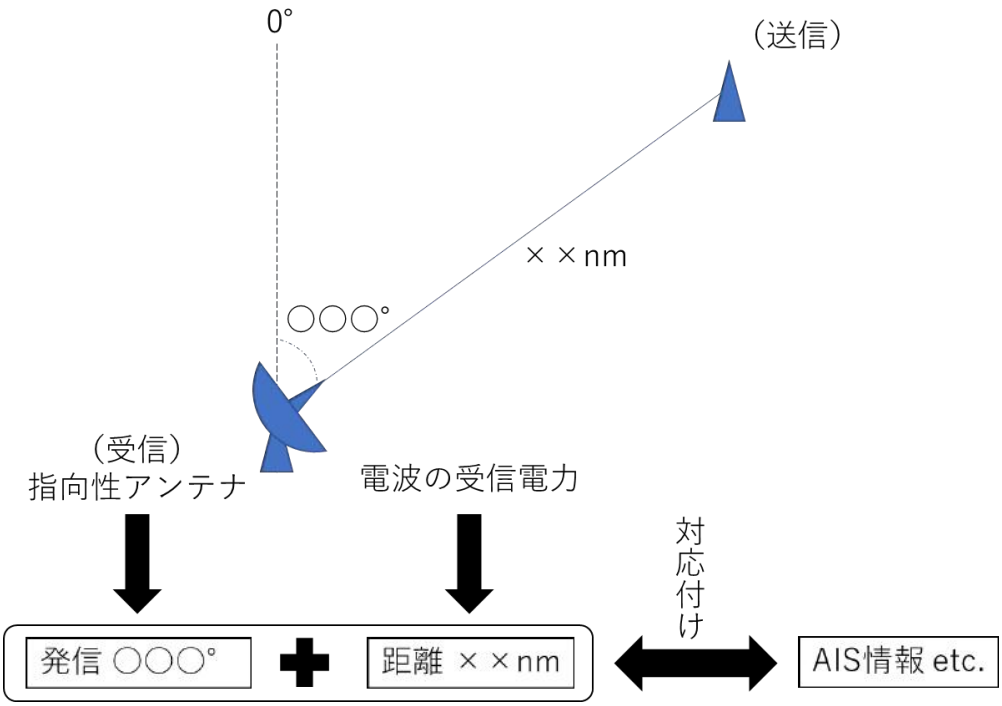

図 6　電波伝搬距離と指向性アンテナによる対応付け（イメージ）

電波の特性と電波伝搬距離について

電波を波長（周波数）で分類した場合、国際 VHF や AIS で使用される電波「超短波（Very High Frequency）」は、波長は 10m～1m、周波数は 30MHz～300MHz（国際 VHF や AIS は 156MHz～174MHz）である。特徴としては、空間波による見通し範囲の通信が可能で、雨や霧の影響を受けにくく、情報を多くのせることができる。しかし、ラジオダクトなどによる以上伝搬で遠くの送信局の妨害を受けることもある。また、直接波と大地反射波が主となるため直進性が高く、回折現象が起きにくく、電離層では反射せずに通抜する特徴がある。ちなみに、衛星 AIS はこの特徴を利用している [37]。

地球表面を平面と仮定した時、超短波の送受信距離と送受信電力の関係は、直接波と大地反射波の合成波によって表すことができる [38]。送受信距離と二波（直接波と大地反射波）の関係を図示したものを図 7、及び図 8、関係式を式 1 から式 7、式中・図の記号の説明を表 1 に示す [38, 39]。

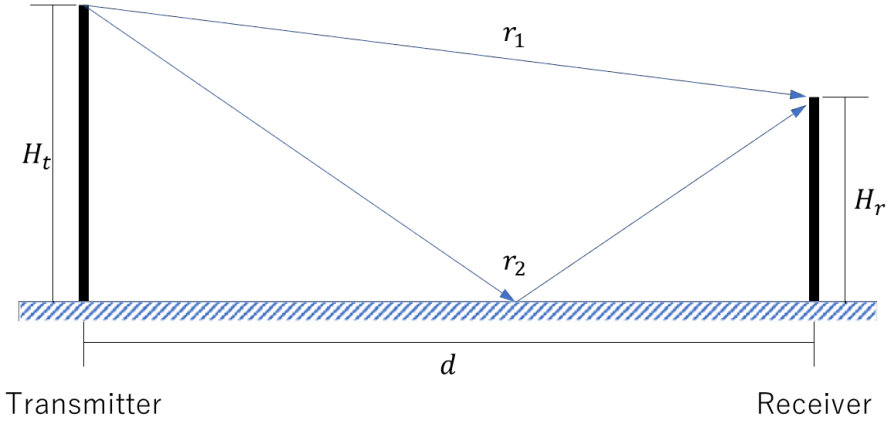

図 7　送受信距離と直接波、大地反射波の合成

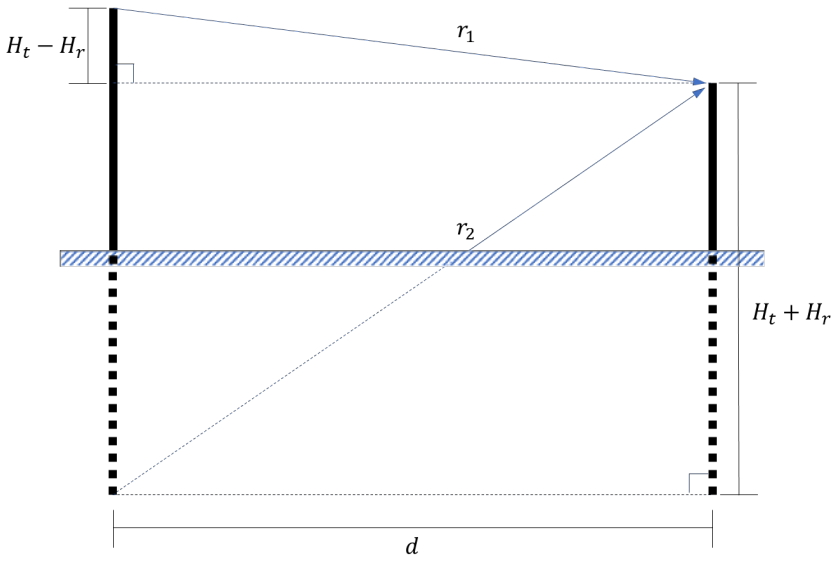

図 8　送受信距離と直接波、大地反射波の距離の算出

$$\beta = \frac{2\pi}{\lambda} \quad (1)$$

$$r_1 = \sqrt{d^2 + (H_t - H_r)^2} \quad (2)$$

$$r_2 = \sqrt{d^2 + (H_t + H_r)^2} \quad (3)$$

$$l = r_2 - r_1 \cong \frac{2H_t \cdot H_r}{d} \quad (4)$$

$$E_f = \frac{\sqrt{30P_t \cdot G_t}}{r_1} \quad (5)$$

$$E_r = 2E_f \left| sin\left(\frac{\beta \cdot l}{2}\right) \right| \quad (6)$$

$$P_r = \frac{{E_r}^2 \cdot \lambda^2 \cdot G_r}{480\pi^2} \quad (7)$$

表 1　送受信距離と送受信電力の関係式の記号の説明

λ	通信周波数
β	波数
d	送受信距離
H_t	送信アンテナの高さ
H_r	受信アンテナの高さ
r_1	直接波の距離
r_2	大地反射波の距離
l	二波の距離の差
P_t	送信アンテナの送信電力
P_r	受信アンテナの受信電力
G_t	送信アンテナの利得
G_r	受信アンテナの利得
E_f	自由空間における電界強度
E_r	受信アンテナの電界強度

式 1 は通信周波数λから波数βを求める。式 2 と式 3 は図 7 の直接波と大地反射波の関係を図 8 に置き換え、送受信距離と送受信アンテナ高さの関係を三平方の定理を用いることで、直接波の距離r_1と大地反射波の距離r_2を求める。式 4 は直接波の距離r_1と大地反射波の距離r_2の差lである。式 5 は自由空間における電界強度E_fを送信電力P_tと送信アンテナの利得G_t、直接波の距離r_1から求める。式 6 は受信アンテナの電界強度E_rを自由空間における電界強度E_fと定位相β、直接波と大地反射波の距離の差lから求める。位相差と距離の差によって 2 波の干渉を求めている。式 7 は受信電力P_rを受信アンテナの電界強度E_rと通信周波数λ、受信アンテナの利得G_rから求める。

受信電力が急激に減衰するところをブレークポイントという。ブレークポイントを超えてしまうと電波を受信できない。ブレークポイントは式 8 で求めることができる。

$$d_{bp} = \frac{2\sqrt{2}\pi H_t H_r}{\lambda} \qquad 8$$

受信アンテナに直接波と大地反射波の二波が到達した時、二波の位相差によってお互いに干渉が生じて、合成波が強くなったり、打ち消しあったりする。二波モデルによる距離と受信電力の関係を図 9 に示す。全体的に受信電力を見たとき、距離が離れるにしたがって受信電力は減少するが、二波が干渉によって打ち消しあったとき、その受信電力は急激に減少していることが見て取れる。よって、実際にこれらの式を用いて距離を推定するには、受信電力を取得して距離を推定し続けて、干渉の影響を受けても問題がないようにしなければならない。

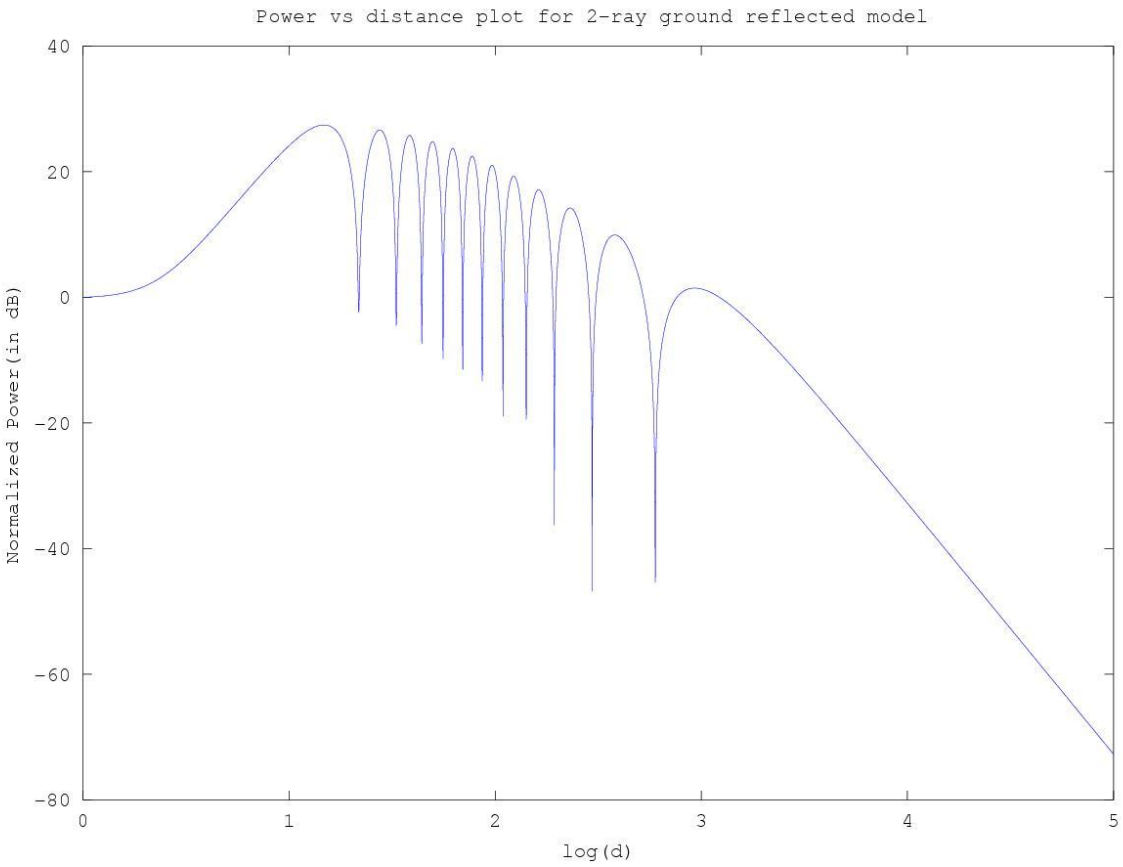

図 9　二波モデルによる距離と受信電力の関係

（出典）Wikipedia「Two-ray ground-reflection model」

https://en.wikipedia.org/wiki/Two-ray_ground-reflection_model

また、実際に受信機で測定できる受信電力は、受信アンテナの受信電力P_rからアンテナと受信機を繋ぐケーブルや受信機本体での損失を差し引いた電力になる。実際に計測した受信電力から理論式で推定するには単純な計算にはならず、他にも送受信間に障害物（他の船舶など）がある場合は電波が回折することもあり、測定値から理論式を用いて距離を推定するには複雑

な要因が関与することを考えなければならない。

指向性アンテナによる方位の特定

指向性アンテナ（ビームアンテナ）は電波の送受信において、特定の方向に対して電波を放射し、特定の方向の電波だけ受信できるアンテナである。指向性アンテナで受信することで、電波の発信方向に対して大きな利得を得ることができ、他の方向からのノイズは拾わないため、どの方向から電波が発信されたかを特定することができる。指向性アンテナの代表例として八木・宇田アンテナ（図 10）やパラボラアンテナ（図 11）がある。

図 10　八木・宇田アンテナ

（出典）Wikipedia「Yagi-Uda antenna」https://en.wikipedia.org/wiki/Yagi%E2%80%93Uda_antenna

図 11 パラボラアンテナ

（出典）Wikipedia「Parabolic antenna」https://en.wikipedia.org/wiki/Parabolic_antenna

一般的に国際 VHF 通信や AIS 情報を受信するアンテナは、無指向性アンテナが用いられて

いる。無指向性アンテナの代表例としてホイップアンテナがある。無指向性アンテナはどの方向にも電波が均等に放射し、全方向から電波を受信することができる。また、ホイップアンテナの場合はアンテナをコイルすることにより、図 12 のような短いアンテナにすることができるため、携帯型トランシーバーなどにも用いられている。

図 12　トランシーバーの短いホイップアンテナ

（出典）Wikipedia「Whip antenna」https://en.wikipedia.org/wiki/Whip_antenna

電波伝搬距離と指向性アンテナによる対応付けの課題

本手法は指向性アンテナを用いることで、電波の発信者の方向を的確に知ることができる点が最大の利点である。また、複数の国際 VHF が同時通信をしたとしても、少なくとも何かしらの国際 VHF 通信についての距離と方位は知ることができると考えられる。

SOLAS 条約により空中線電力（送信機がアンテナに対して供給する電波の電力）が定められており、海岸局では 50W、船舶局では 25W とされている。通達距離は約 50km であり、使用する周波数帯は 156～174MHz の間の周波数を使用する [40]。小型船舶においても携帯型の場合は 5W 以下、据付型の場合は 25W 以下の国際 VHF を使用できるとされている [24]。このように国際 VHF がある程度スペックが定められている点から電波伝搬距離の推定がしやすいため、電波の減衰傾向を調べることにより、発信者の距離を推定できるのではないかと思われる。

基本的に国際 VHF 通信は、あらゆる船舶に情報を発信することや、遭難時に周囲の船舶に救難信号を出すことを考えたとき、全方位に対して電波を放射する必要があることから、指向性アンテナを用いることは望ましくはない。指向性アンテナをローテーターによって回転させて送受信をするにも、周辺船舶などが電波の途切れなく受信をするにはローテーターを高速回転させる必要があると思われる。もし、この手法の有用性が認められたとしても、すべての船舶の国際 VHF 通信がローテーターを設置するには法律の改正とコストが問題となる。そのため、あくまで本手法は電波の発信者の距離と方向を求めることで AIS 情報と対応付けるもので

あり、現状としては本来の国際 VHF 通信の使用はしないことが前提条件となる。

2.3.3　提案 3：音声ノイズでの距離推定による対応付け

国際 VHF 通信は距離が離れると音声のノイズが大きくなり、ある一定距離以上離れると音声が聞き取れなくなる。本手法はこのノイズの大きさから距離を推定し、その距離の同心円上にいる船舶を特定することで、AIS の位置情報と対応付ける。ここで、スピーカーやイヤホンから出力される音声のノイズのことを、音声ノイズと呼ぶ。この手法は音声認識を用いた手法とは違い、AIS 情報以外のレーダーなどの情報を利用することができるため、電波伝搬距離と指向性アンテナによる手法と同様に、AIS 非搭載船にも有用性があると考える。ただし、距離が同じで場所（方位）の異なる船舶については、発信者を特定できない課題はあるが、発信者を絞り込むことができるので、有用性はあると思われる。なお、そのイメージを図 13 に示す。

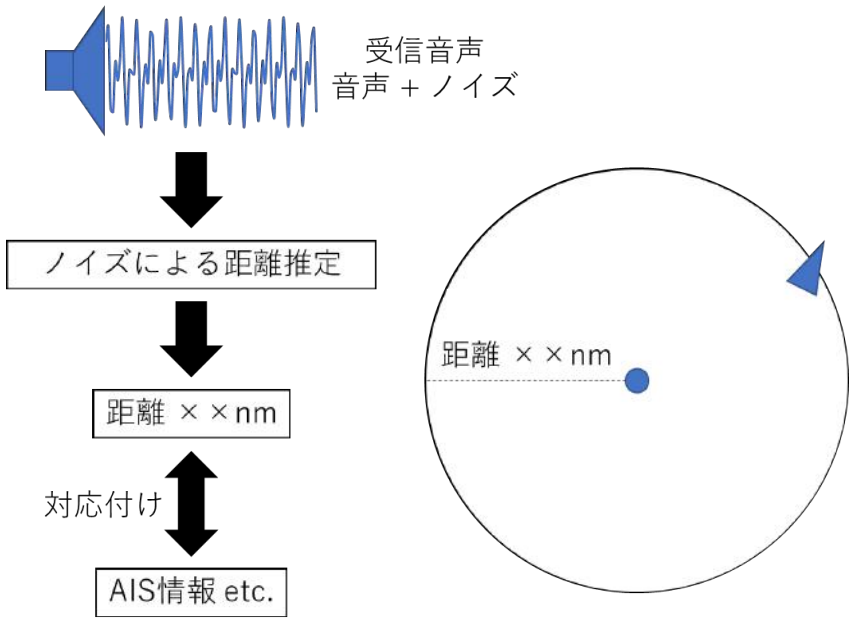

図 13　音声ノイズでの距離推定による対応付け（イメージ）

FM 通信について

国際 VHF 通信は FM 方式で変調されている。FM（Frequency Modulation）は周波数変調とも言い、情報を搬送波の周波数の変化で伝達する変調方式の 1 つである。FM はアナログ（連続的、比例量的）な情報通信のための変調方式でノイズに強い特徴があり、国際 VHF 通信の他にも FM ラジオやアマチュア無線、消防無線、タクシー無線、アナログテレビジョン放送の音声信号などに使われている。

FM は送信したい信号波に変調用の搬送波を使って変調をかけ、搬送波の疎密によって送信波が表される（図 14）。送信波は搬送波の疎密で信号を伝えるため、振幅は常に一定であり、ノイズによって振幅を変化させたとしても、振幅の大小は信号に無関係であるため信号に影響を与えない。また、振幅の幅を大きくしても信号に影響がないことから、送信波の出力（振幅）を大きくすることで、遠くまで電波を飛ばすことができるため、FM はノイズに強いとされて

いる。また、周波数偏移（周波数変調における周波数変化の幅）を大きくすることによって、ダイナミックレンジ（識別可能な信号の最小値と最大値の比率）や占有帯域幅（搬送波の変調で占める周波数の範囲）が広がり、SN 比（信号対ノイズ比）を高くすることができるため、よりノイズに対して強くすることができる [41]。

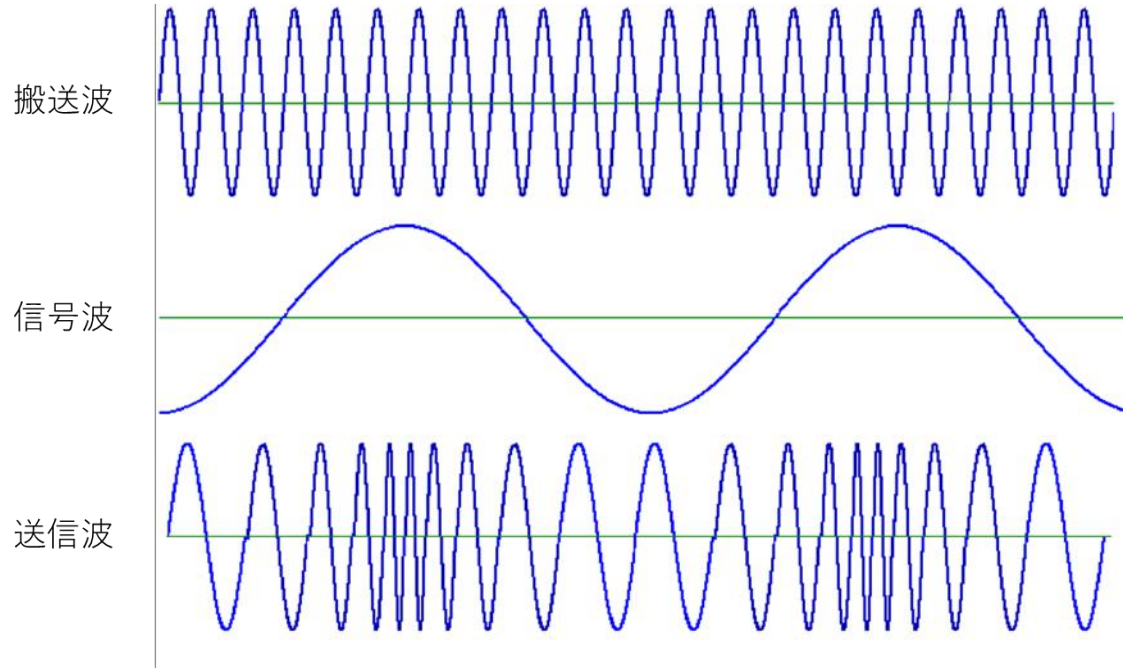

図 14　FM 変調のイメージ

（出典）Wikipedia「周波数変調」https://ja.wikipedia.org/wiki/周波数変調

しかし、FM の受信には弱肉強食特性と呼ばれる性質があり、同一または非常に近接した周波数を複数受信（混信）した場合、弱い電波は強い電波によってかき消される（マスキング）起こるため、電波を受信していたとしても他の強い電波によって聞こえないことがある。

距離の増加による SN 比の悪化

SN 比とは信号の分散をノイズの分散で割った、信号（Signal）とノイズ（Noise）の電力比である。SN 比が大きいほどノイズが少なく、信号の品質が良い。逆にノイズが大きくなると SN 比は小さくなり、信号の品質は悪くなる。通常 SN 比は常用対数で表され、単位はデシベル（dB）を使う。SN 比を式 9、SN 比で用いる記号を表 2 に示す。P_S, P_Nは電力や輝度などの物理量、A_S, A_Nは電流、もしくは電圧の実効値（二乗平均平方根）である。デシベルを用いるので、電力比は常用対数の 10 倍、電流比・電圧比は常用対数の 20 倍をする。測定した信号が真の信号とノイズが足し合わされた信号（$S+N$）であるとき、$(S+N)/N$比で表す場合もある。

$$S/N\ [dB] = 10\ log_{10}\frac{P_S}{P_N} = 20\ log_{10}\frac{A_S}{A_N} \qquad 9$$

表 2　SN 比の記号の説明

P_S	信号の電力や物理量
P_N	ノイズの電力や物理量
A_S	信号の電流（電圧）
A_N	ノイズの電流（電圧）

ノイズは 2.3.1 の課題で述べたように様々な要因によるノイズがあり、常に受信機はこのノイズを拾っている。距離が近いときは発信された信号強度が強いため、音声の信号を取り出すことができるが、距離が離れるにしたがって 2.3.2 の電波伝搬距離で述べたように信号強度が減衰していく。FM の特性上ノイズによって送信波の振幅が増加して音声信号に影響を与えることはないが、距離が離れることで送信波が減衰して振幅が小さくなる。したがって、常に受信しているノイズに対する送信波の信号強度が下がり SN 比が悪化していく。

2.3.2 で述べたように国際 VHF 通信の空中線電力（電波の出力）は、海岸局は 50W、船舶局は 25W（携帯型は 5W）と定められている。電波の到達距離は電波の出力によってある程度決まるため、SN 比の悪化傾向についても電波の出力ごとに同じ傾向がみられると考えられる。

ノイズの種類

ノイズは周波数に対するエネルギー、パワースペクトル密度（Power Spectral Density; PSD）（以下、PSD とする）の傾向によって、いくつかの種類に分類される。大きく分類すると、全周波数で均一なエネルギーをもつ（PSD が平坦）ものをホワイトノイズ（白色雑音）、逆に PSD が平坦でないものをカラードノイズ（有色雑音）と呼ぶ。また、カラードノイズにも PSD の分布傾向によって、いくつかの種類に分類される。

ホワイトノイズ

ホワイトノイズは全ての周波数で同じ強度となるノイズで、PSD は平坦になる。このノイズを音で聞いたとき「シャー」のような音がする。ホワイトノイズと呼ばれる由来は、光がすべての周波数成分を同等に含むと白色になることから由来している。ホワイトノイズの例を図 15、ホワイトノイズに近似させ生成したノイズのスペクトルを図 16 に示す。なお、ホワイトノイズを生成するときは正規乱数を用いて生成されるが、正規乱数は正規分布を持つような乱数であるため、ホワイトガウスノイズとなる。

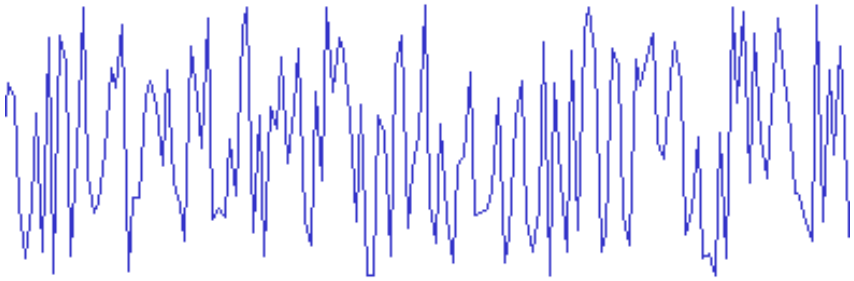

図 15　ホワイトノイズの例

出典）Wikipedia「ホワイトノイズ」https://ja.wikipedia.org/wiki/ホワイトノイズ

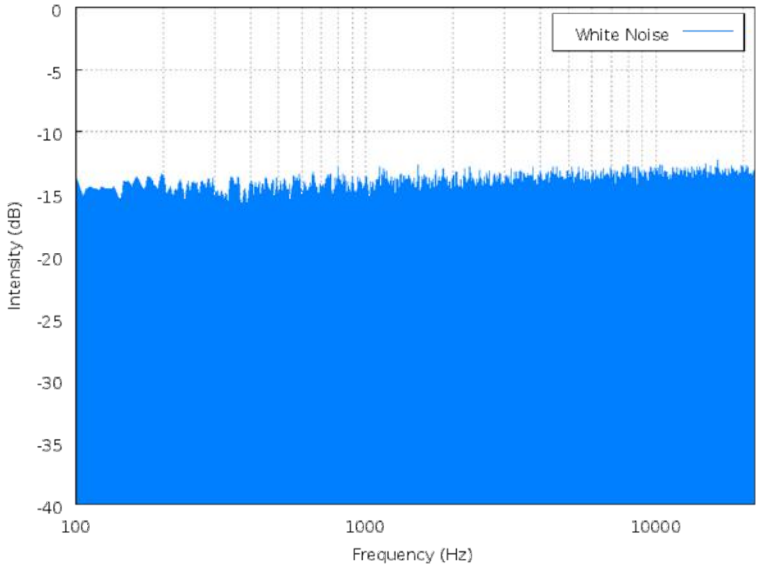

図 16　ホワイトノイズのスペクトル

（出典）Wikipedia「カラードノイズ」https://ja.wikipedia.org/wiki/カラードノイズ

ピンクノイズ

ピンクノイズはパワーが周波数に反比例するノイズで、光がこのような周波数成分を持つとピンク色に見えることから由来する。$1/f$ゆらぎ（PSD が周波数fに反比例するゆらぎ）を持つノイズなので、$1/f$ノイズとも呼ばれる。PSD は周波数が低いほどパワーが強く、ホワイトノイズに対して 1 オクターブあたり 3dB 減衰する LPF（Low Pass Filter; 低域通過フィルタ）を通すとピンクノイズになる。図 17 にピンクノイズのスペクトルを示す。

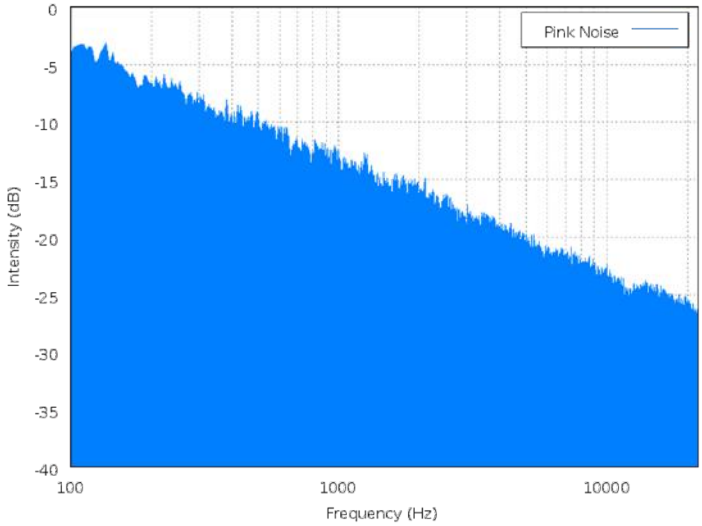

図 17　ピンクノイズのスペクトル

（出典）Wikipedia「カラードノイズ」https://ja.wikipedia.org/wiki/カラードノイズ

ブラウニアンノイズ

ブラウニアンノイズ（ブラウンノイズ）はレッドノイズとしても知られており、ブラウン運動によって生成されたノイズをいう。名前は色ではなく、ブラウン運動を発見したロバート・ブラウンの名前から由来する。この PSD は$1/f^2$に比例し、ピンクノイズ以上に低い周波数ほど強いエネルギーを持ち、オクターブ当たり 6dB 降下する。ブラウニアンノイズはホワイトノイズの積分によって生成することができる。図 18 にブラウニアンノイズのスペクトルを示す。

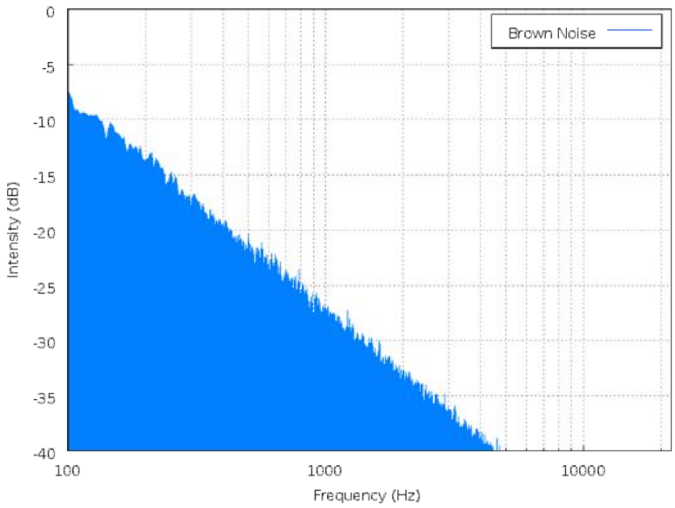

図 18　ブラウニアンノイズのスペクトル

（出典）Wikipedia「カラードノイズ」https://ja.wikipedia.org/wiki/カラードノイズ

ブルーノイズ

ブルーノイズはアジュールノイズとも呼ばれる。ピンクノイズとは逆に、PSD は周波数の上がるにつれてオクターブあたり 3dB 上昇し、密度はfに比例する。図 19 にブルーノイズのスペクトルを示す。

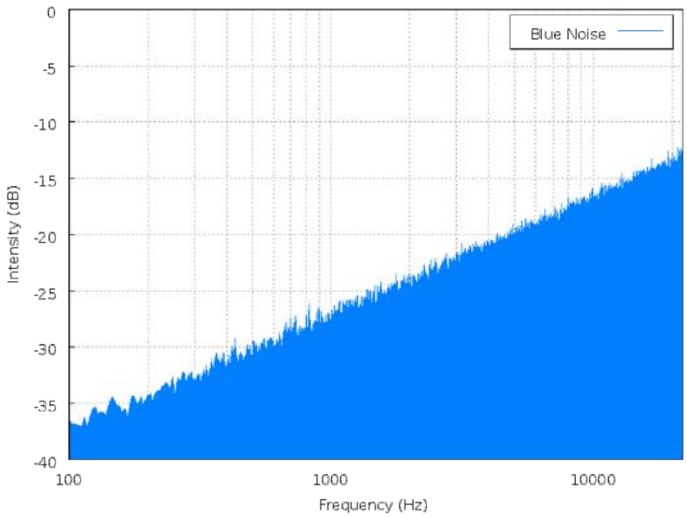

図 19　ブルーノイズのスペクトル

（出典）Wikipedia「カラードノイズ」https://ja.wikipedia.org/wiki/カラードノイズ

パープルノイズ

パープルノイズはバイオレットノイズとも呼ばれる。ブラウニアンノイズとは逆に、PSD の周波数が上がるにつれてオクターブあたり 6dB 上昇し、密度はf^2に比例する。ホワイトノイズを微分したものと等しい。図 20 にパープルノイズのスペクトルを示す。

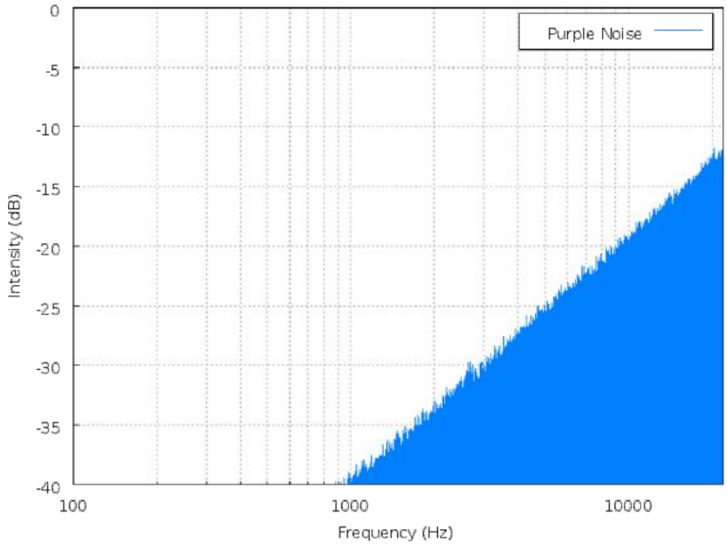

図 20　パープルノイズのスペクトル

（出典）Wikipedia「カラードノイズ」https://ja.wikipedia.org/wiki/カラードノイズ

グレイノイズ

グレイノイズは音響心理学的な等ラウドネス曲線に沿った PSD を持つノイズである。全周波数に対して等しいラウドネスを持っており、個人差はあるが人間が聞いたときに、どの周波数成分も同じ大きさに聞こえる特徴がある。図 21 にグレイノイズのスペクトルを示す。

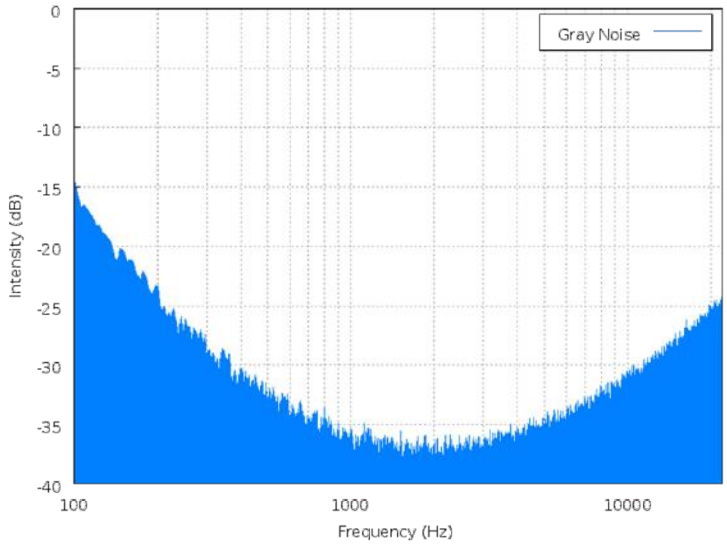

図 21　グレイノイズのスペクトル

（出典）Wikipedia「カラードノイズ」https://ja.wikipedia.org/wiki/カラードノイズ

音の性質について

そもそも音とは、音波という媒体を振動させることで生じる縦波（疎密波）である。また、音の性質を決める要因は、大きさと高さ、音色の 3 つであるといわれている。音の大きさは媒体を振動する幅の大きさ（振幅）で、振動する幅が大きいほど音は大きくなる。音の高さは振動する波長が短い（周波数が高い）ほど、音は高くなる。音色は物理的に異なる 2 つの音が、同じ大きさ、同じ高さであっても異なった感じに聞こえるものである。

人間の耳が聞くことのできる音の高さは 20Hz～20kHz まで聞くことができる。ただし、人の耳は周波数によって感度が異なり、物理的に同じ音圧であっても周波数によって感じる音の大きさが異なる。人の耳が聞き取れる音の周波数と音圧レベルの関係を等高線で表したものを、等ラウドネス曲線といい、ISO 226 として国際標準規格化されている。図 22 に等ラウドネス曲線を示す。

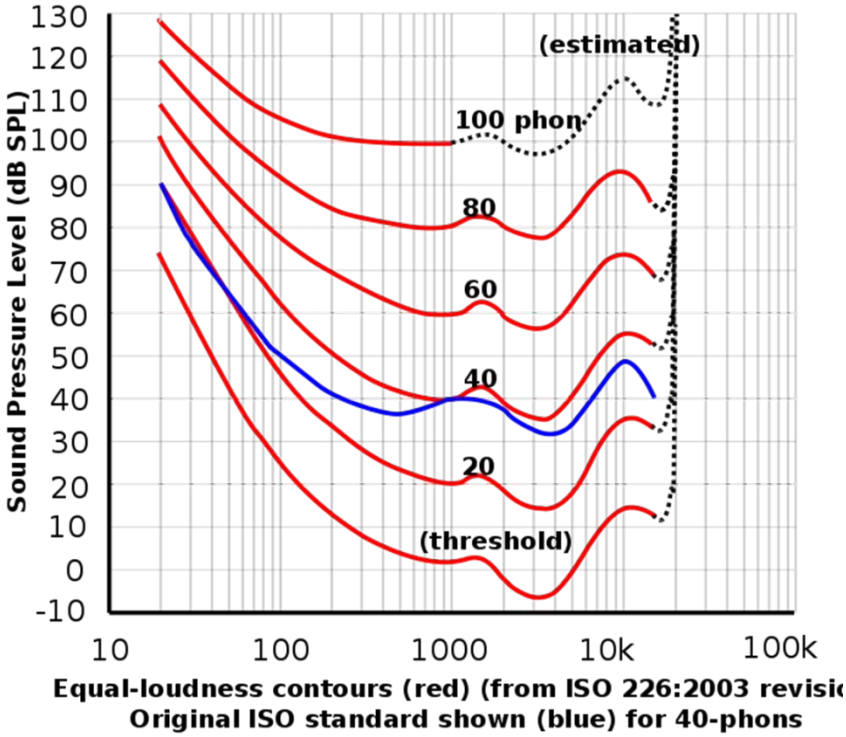

図 22　等ラウドネス曲線

（出典）Wikipedia「等ラウドネス曲線」https://ja.wikipedia.org/wiki/等ラウドネス曲線

人の声の音（音声）は声帯で発した原音を喉、口、鼻などによって倍音を増幅しており、増幅した音声が特定の周波数において音量のピークがいくつか出現する。この出現したピークの周波数をフォルマントといい、周波数の低い順に第一フォルマント、第二フォルマントのように数字を当てていく。声帯は会話をするとき約 100～200Hz 付近で声帯を振動させており、喉、口、鼻などで共鳴させることにより、特定帯域ごとに倍音が増幅される。人間が音声を聞くときは、第一フォルマント（約 500～1000Hz）と第二フォルマント（約 1500～3000Hz）によって、母音の弁別を大体行うことができ、逆に主要フォルマント（約 500～3000Hz）を音声から除去

すると、発音された母音が認識できなくなる。つまり、音声の大半は約 500～3000Hz で構成されているといえる。

音声ノイズでの距離推定による対応付けの課題

本手法は特別なアンテナや送受信機を用いることなく、出力される音声のみで距離を推定できる可能性があることである。もし、この手法が確立されれば非常に安いコストで、法的な問題を抱えることなく国際 VHF 通信と AIS 情報などを対応付けるシステムを構築できる。

課題としては距離と音声ノイズの関係が明らかにされていないため、この点を解明する必要がある。しかし、受信機から出力されるノイズは、様々な要因が与えるノイズがほぼ常に一定であるとしたとき、距離による電波の減衰とノイズの関係は相関があると予想される。もし、この相関関係が明らかになれば、国際 VHF の出力（50W, 25W, 5W）は定められていることから、距離と音声ノイズの相関がみられるのではないかと考えられる。また、音声の主要フォルマント周波数は約 500～3000Hz であることから、国際 VHF 通信の音声の約 500～3000Hz の範囲は音声部分であり、それ以外の周波数は音声の構成に不要な音、すなわち音声ノイズであると考えられる。つまり、音声に関係ない周波数範囲（約 500～3000Hz 以外）から音声ノイズを測定することで、距離と音声ノイズの関係を明らかにすることができると考えられる。また、ノイズの傾向や種類が明らかになれば、その傾向から主要フォルマント周波数に乗った音声ノイズからも距離が推定できると思われる。

また、2.3.1 の課題と同様に、同時に複数の通信があった場合は混信してしまうために音声を得ることができない。よって、同時通信に対して本手法は有効でないと考えられる。

2.3.4　各提案手法の利点と欠点、課題

これまで国際 VHF 通信と AIS 情報などの対応付けについて、3 つの手法を提案した。これらについて利点と欠点、課題を表 3 にまとめる。

提案 1（2.3.1）の音声認識による AIS 情報との対応付けは、音声認識を用いるので通信内容をテキスト化することができ、2.1 で述べた国際 VHF 通信の解析・ビッグデータ化に非常に有効であると考える。また、コストが安い点や受信機やアンテナを改造する必要がないので法的問題はクリアできる利点がある。しかし、音声認識をしたときに船名やコールサインなどから発信船舶を特定するため AIS 情報が必要であり、非 AIS 搭載船には有効ではない。スケルチ機能や発信者の発声と発信のタイミングによって、音声の最初や最後が途切れてしまい、音声認識ができない場合がある。混信によっても音声は聞き取れないことから、同時通信には対応できない。

提案 2（2.3.2）の指向性アンテナと電波伝搬距離による対応付けは、指向性アンテナで方位を特定し、電波の受信強度から距離を推定するため、この手法のみで発信船舶を特定することが可能である。また、音声が聞き取れなくても距離推定ができ、混信しても指向性アンテナで 1 つの発信については位置を特定することができ、他の AIS やレーダーなどの情報と対応付けることができると考えられる。しかし、アンテナの変更や受信機の改造により、設備投資のコストと法的な問題が想定される。距離が離れるにしたがって電波は減衰するが、電波の干渉で

受信強度が強くなったり、弱くなったりするので、干渉しても正しく距離を推定できる方法を確立する必要もある。

提案 3（2.3.3）の音声ノイズによる対応付けは、提案 2 のように改造する必要はないため、コストと法的問題はないと思われる。また、音声ノイズで推定するため、音声が聞き取れる必要はなく、AIS やレーダーなどと対応付けることで、非 AIS 搭載船舶にも対応ができる。しかし、音声ノイズによる距離の推定方法を確立する必要があり、混信により同時通信にも対応ができないという課題がある。

表 3　各提案手法の利点と欠点、課題

	提案 1	提案 2	提案 3
利点	・通信内容が分かる ・コストが安い ・法的問題がない	・指向性アンテナによる方位と電波伝搬距離から、単独で発信船舶を特定できる ・音声が聞き取れなくても距離推定ができる ・同時通信に対応できる（同時通信した内の 1 つのみ）	・距離推定で受信エリアを計算し、AIS やレーダーなどと対応付けることで発信船舶を特定できる ・音声が聞き取れなくても距離推定ができる ・コストが安い ・法的問題がない
欠点	・対応付けられるのは AIS 情報のみ（船名やコールサインなどからに限る） ・ノイズなどで音声認識できない音声は対応付けられない ・スケルチ機能や発信者のタイミングによっては、音声の最初や最後の部分が途切れる ・音声認識アルゴリズムが認識出る専門用語が極めて少ない ・同時通信は対応できない	・コストが高い ・法的な問題が想定される ・通信内容はわからない ・電波の干渉によって距離推定が誤る可能性がある。	・通信内容はわからない ・音声ノイズによる距離の推定方法が確立されていない ・同時通信に対応できない
課題	・音声認識できる専門用語を増やす ・ノイズに対する認識率の向上	・電波の干渉を考慮した距離の推定方法の確立	・音声ノイズによる距離の推定方法の確立

これらの利点、欠点、課題と、本研究の目的を考えたとき、海上交通の研究において必要な国際 VHF 通信のビッグデータの構築は、通信内容から船舶間でどういった通信をしているかを分析するために、提案 1（2.3.1）の音声認識は必要であると思われる。それとは別に、自動運航船などで活用することを考えた場合は発信船舶の特定が重要になるため、提案 2（2.3.2）と提案 3（2.3.3）は有効であるが、コストと法的問題を考えれば、提案 3 の音声ノイズによる距離推定で情報を対応付けるのが最も有効であると思われる。

この 2 点から、提案 1 の音声認識と提案 3 の音声ノイズによる距離推定を合わせ、AIS 情報やレーダーなどと対応付けることが、国際 VHF 通信のビッグデータの構築や、自動運航船の国際 VHF 通信の傍聴に必要なのではないかと考える。

2.4　本研究の目的

2.3.4 で述べたように国際 VHF 通信を他の情報と対応付けるには、音声認識による対応付けと音声ノイズによる距離推定を組み合わせるのが有効である。そこで、本研究では音声ノイズでの距離推定による情報の対応付けについて研究する。音声ノイズが距離に応じてどのように変化するかを調べ、そこから距離推定モデルを構築することを目指す。

第3章　音声データの分析方法

本章では音声データの分析方法として音波の振幅、周波数スペクトル、パワースペクトル（PSD）、サウンドスペクトログラムの 4 つについて述べる。

3.1　音波の振幅

音波の振幅を表したものを図 23 に示す。この図は縦軸に振幅、横軸に時間をとっており、振幅による音の大きさや波長の長短を見ることがきる。音声の信号がある振幅とそうでない振幅から差を求めることで、距離に応じて音声ノイズがどう増加するかを観察する。

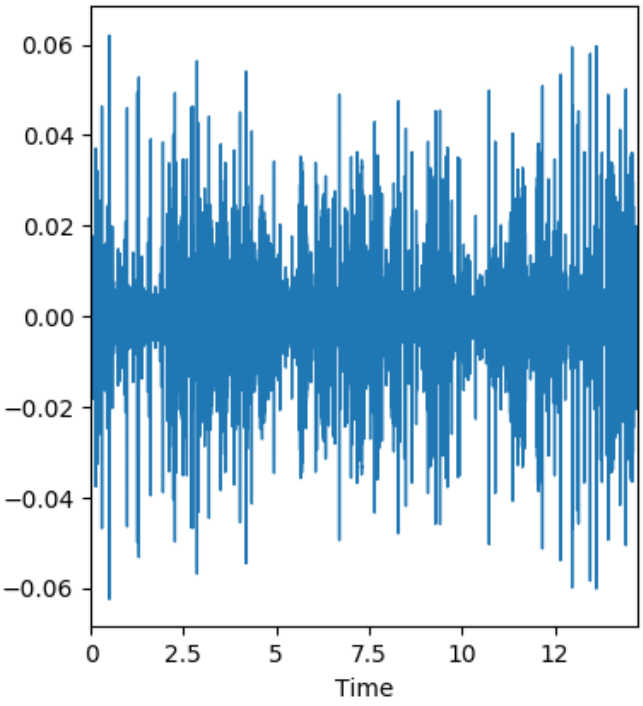

図 23　音波（縦軸：振幅、横軸：時間）

3.2　周波数スペクトル

音波をフーリエ変換することにより、ある瞬間の音を周波数成分に分解することで、その音を構成する周波数とその強さを見ることができる。これを周波数スペクトルという。一般的に音波をフーリエ変換するには、FFT（Fast Fourier Transform; 高速フーリエ変換）を用いることが多い。図 24 に FFT した周波数スペクトルを示す。この図は縦軸に振幅、横軸に周波数をとっている。また、振幅の 2 乗（V^2rms）でパワースペクトルとなる。パワースペクトルにすることで、周波数における信号強度を見ることができる [42]。

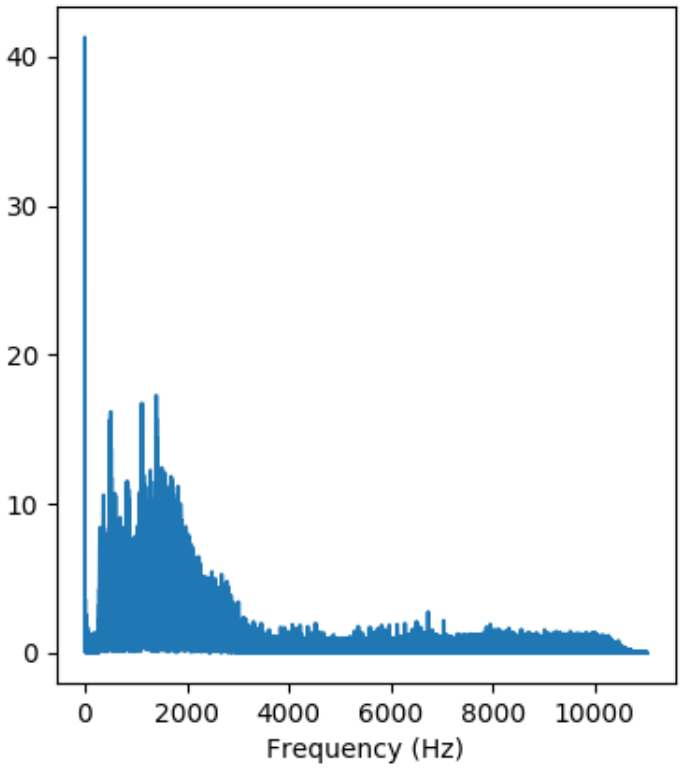

図 24　周波数スペクトル（縦軸：振幅、横軸：周波数）

周波数分解能Δfと時間窓長T、サンプリング周波数f_s、サンプリング点数Nの関係を式 10 に示す。FFT はサンプリング周波数f_sとサンプリング点数Nによって、周波数分解能Δfが決まる。Δfが小さいほど周波数分解能が高く、時間窓長Tが大きいほうが高くなる。すなわち、サンプリング点数Nを大きくするか、サンプリング周波数f_sことで周波数分解能を高くすることができる [42]。

$$\Delta f = \frac{1}{T} = \frac{f_s}{N} \qquad 10$$

3.3　パワースペクトル密度（PSD）

FFT 周波数分解能Δfに依存しない、単位周波数幅（1Hz 幅）当たりのパワー値として表現するスペクトルを PSD（Power Spectral Density Function; パワースペクトル密度関数）という。図 25 に PSD を示す。この図は縦軸にV^2/Hz、横軸に周波数をとっている。PSD を求める方法に Welch 法による PSD 推定がある [42]。

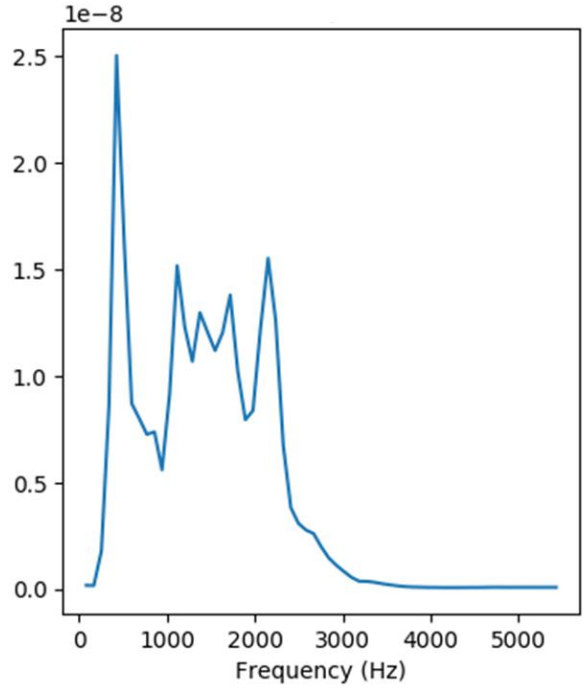

図 25　PSD（縦軸：PSD [V^2/Hz]、横軸：周波数）

3.4　サウンドスペクトログラム

周波数スペクトルやパワースペクトル、PSD はある時刻におけるスペクトルであるが、時間の次元を加えたものをサウンドスペクトログラムという。図 26 にサウンドスペクトログラムを示す。

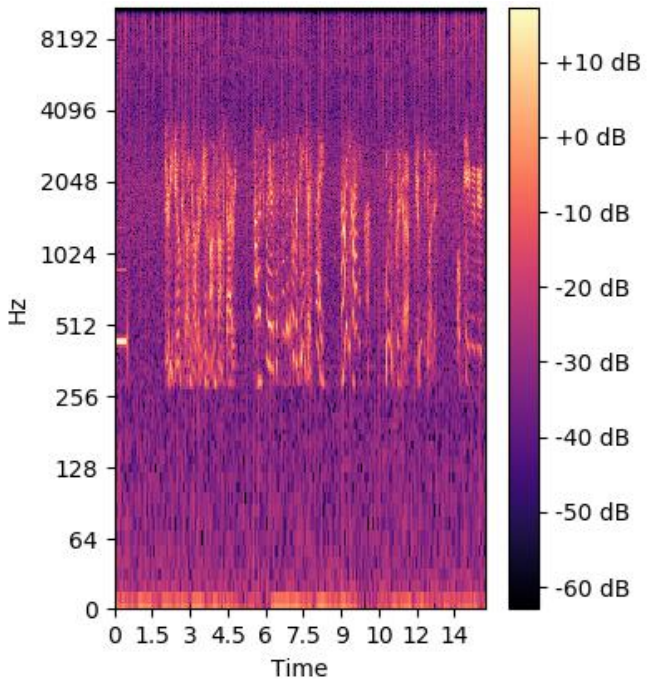

図 26　サウンドスペクトログラム（STFT）

サウンドスペクトログラムは STFT（short-time Fourier transform; 短時間フーリエ変換）で窓関数をシフトしながらフーリエ変換を行うことで求める。また、スペクトログラムの信号の強さを PSD などに変えることで、時間変化における PSD の変化をみることもできる。

第4章　国際 VHF からの音声ノイズと距離の関係

本研究では国際 VHF 通信の音声ノイズから距離を推定するために、実際に電波を発信した船舶を音声データから特定し、AIS 情報の船名と位置情報から距離を求め、音声ノイズと距離の関係を明らかにする実験を行った。

4.1　実験方法

この実験では国際 VHF 通信を携帯型（5W）で受信し、その音声をボイスレコーダーに録音する。同時に AIS を受信しログデータを記録する。その後、録音した音声データを聞いて船名が特定できるデータのみを抽出し、その受信時刻と船名から電波を発信した船舶の AIS 位置情報をもとに、発信した船舶の距離を求め、音声ノイズと距離の関係を分析する。実験に使用した機器の一覧表を表 4、システムの概要を図 27 に示す。

表 4　使用機器一覧

AIS 受信機	AMEC / CYPHO-101G AISRECEIVER
GPS 受信機	Global Sat / BU-355S4（PS/2 コネクタ）
国際 VHF（携帯型）	STANDARD HORIZON / HX851
ねじ込み式プラグ変換アダプタ	STANDARD HORIZON / CT-91
Raspberry Pi（コンピュータ）	Raspberry Pi 1 Model B
ボイスレコーダー	東芝 / DMR-1800V GREEN HOUSE / GH-KANADT8
USB メモリ	ELECOM / MF-MSU3A16GPN
バッテリー	八洲電業 / DLG-FCMINI-B

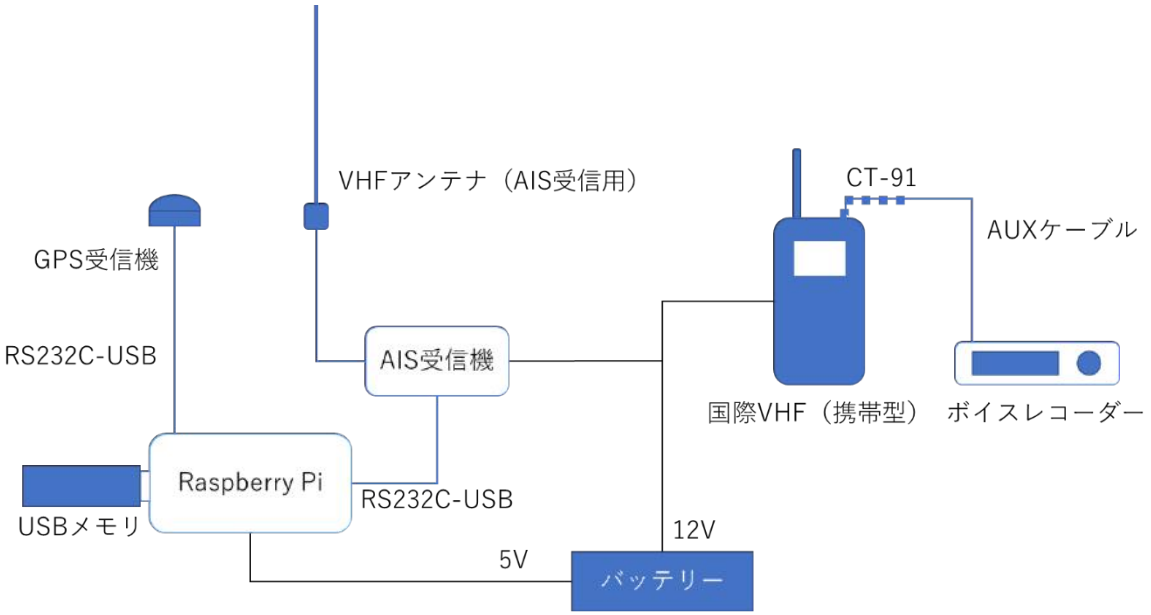

図 27 国際 VHF から音声ノイズと距離を分析する実験でのシステム概要図

国際 VHF 通信の音声の録音については、国際 VHF（携帯型）のスピーカー出力とボイスレコーダーのマイク入力を AUX 端子でつなぐ必要があるのだが、HX851 のイヤホンマイクの端子が 1 ピンの 3.5φ ねじ込み式であるため、CT-91 の変換アダプタを用いてイヤホン 3.5φ、マイク 2.5φ の 2 ピンに変換する。そして、CT-91 のイヤホン端子（3.5φ）とボイスレコーダーの AUX 端子（3.5φ）を AUX ケーブルでつなぎ、国際 VHF 通信の音声をボイスレコーダーで録音する。この時、国際 VHF 通信のチャネルは最も通信が多いと思われる ch16 に設定した。

AIS は VHF アンテナから AIS 受信機を用いてデータを取得した。GPS も GPS 受信機でデータを取得した。受信した AIS と GPS データは RS232C-USB ケーブルを通して Raspberry Pi に送り、minicom を使ってシリアル通信をログとして USB メモリに保存した。なお、詳しいやり方については 4.2 で述べる。

国際 VHF（携帯型）はバッテリーを内蔵しているが、長期的受信を可能にするため、バッテリーから 12V 給電した。また、AIS 受信機も外部電源が必要なため、バッテリーから 12V 給電した。Raspberry Pi と GPS 受信機については 5V 給電した。なお、ボイスレコーダーについては DMR-1800V（東芝）を使用していたが、乾電池が途中で切れたため、2 回目から GH-KANADT8（GREEN HOUSE）を使用した。なお、GH-KANAD8 も給電しながらの使用はできなかったが、内臓バッテリーの電池持ちが非常に良いため、最初の録音以降はこのボイスレコーダーのみを使用している。

実際に機器を接続した様子を図 28 に示す。

図 28　国際 VHF 通信と AIS 情報を収集するシステム
（左：AIS 受信機・Raspberry Pi・ボイスレコーダー・バッテリー、右：国際 VHF）

図 29　機器を接続して設置した様子

4.2　Raspberry Pi で AIS・GPS のログを取得する方法

AIS と GPS の受信データを記録するには、コンピュータで 2 つのシリアル通信を受信してログファイルを保存する必要がある。しかし、一般的なコンピュータはバッテリーの電池持ちが期待できないため、本研究では Raspberry Pi に Micro USB Micro-B 端子から 5V 給電することで長時間におけるログ取得を可能にした。図 30 は本研究で使用した Raspberry Pi Model B の写真である。

Raspberry Pi とはイギリスのラズベリーパイ財団によって主に教育目的で開発された、ARM プロセッサ搭載のシングルボードコンピュータである。小型で非常に安価であり、主に IoT 機器として趣味や業務に広く使われている。最近の Raspberry Pi はシングルボードコンピュータとしては非常にハイスペックな Raspberry Pi 3 Model B+や、超小型（65mm×30mm）の Raspberry Pi Zero WH などが登場している。OS も Debian をカスタマイズした Raspbian だけでなく、サードパーティの OS（Ubuntu, Windows10 IoT Core, RISC OS, OSMC など）が登場し、好きな OS で IoT 機器などの開発や、動画や音楽などのメディア再生に特化した機器を開発することがで

きるようになっている。

本研究では 2012 年に発売された Raspberry Pi Model B を用いて、シリアル通信の受信とログの保存を行った。CPU はシングルコア 700MHz、メモリ 512MB と低スペックではあるが、非常に高度な処理をするわけではないので問題はない。ストレージは SD カード（8GB）で、OS は Raspbian を使用した。ログデータは SD カードに保存するストレージ容量がなくなる可能性があったため、外部ストレージとして USB メモリに保存した。

AIS と GPS のログを取得するための事前準備を 4.2.1、ログの取得方法を 4.2.2、ログの取得の停止方法を 4.2.3、AIS のデコードと UTC timestamp の対応付けについて 4.2.4 に記す。

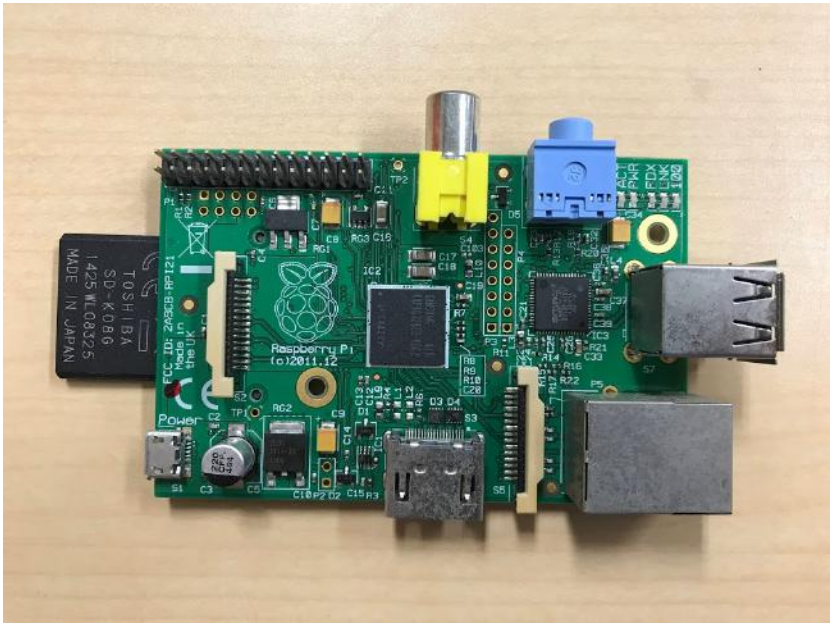

図 30　Raspberry Pi Model B

4.2.1　Raspberry Pi の事前準備

Raspberry Pi で AIS・GPS のログを取得するために、まず下記のように事前準備を行う必要がある。

1. Raspberry Pi Software Guide [43]に従って、Raspbian をインストールする。
2. Raspberry Pi をインターネットに接続した状態で起動する。
3. ターミナルを起動し、”sudo apt-get install minicom screen”を実行する。
4. AIS 受信機をセットアップし、AIS と GPS のシリアルケーブル（RS232C-USB）を Raspberry Pi の USB ポートに差し込む。なお、差し込む順番で USB ポート名との対応関係が変わる。本研究では AIS、GPS の順に差し込み、AIS の USB ポートは”/dev/ttyUSB0”、GPS の USB ポートは”/dev/ttyUSB1”とした。
5. ターミナルに“sudo minicom -s”を入力して実行し、minicom を起動する。
6. 「画面とキーボード」を選択し、次に「コマンドのためのキー」を選択し、”^B”と入力する。これは screen でセッションを作成した際に、コマンドの重複を避けるためである。
7. 「シリアルポート」を選択し、表 5 に従って設定を行い、名前を付けて保存する。

表 5　minicom の設定

	AIS	GPS
シリアルデバイス	/dev/ttyUSB0	/dev/ttyUSB1
速度/パリティ/ビット	38400 8N1	4800 8N1

4.2.2　ログの取得方法

Raspberry Pi で AIS と GPS のログを取得する方法を下記に記す。なお、Raspberry Pi を操作するために、別のコンピュータから SSH 通信でイーサネットケーブルを介して操作する。そのため、ログの取得に関する操作には別のコンピュータが必要である。OS が Windows の場合は、事前に TeraTerm をインストールする必要がある。

ログを取得する際には、AIS と GPS の記録時刻から受信タイミングを同期する必要があるので、タイムスタンプを記録するように設定する。また、screen コマンドでセッションを作成し、SSH 通信を遮断後もログを取得し続けるように設定する。

1. Raspberry Pi を起動する。
2. Raspberry Pi にイーサネットケーブル、USB メモリを接続する。
3. Windows の場合は TeraTerm、macOS や Linux の場合はターミナルを起動し、”ssh pi@raspberrypi.local”を実行する。
4. パスワードが聞かれるので、Raspberry Pi のパスワードを入力する（初期設定は”raspberry”）。
5. “cd /media/pi/[USB メモリのパス]”を入力し、USB メモリのディレクトリに移動する。
6. “screen”を実行し、そのまま Enter キーを押す。
7. AIS の USB ポートを接続（事前準備で”/devttyUSB0”を設定した方）し、”sudo minicom [AIS の設定名]”を実行する。
8. Ctrl-B + Z キーを押し、N キーを押す。
9. 再び Ctrl-B+Z キーを押し、L キーを押す。そして、AIS のログのファイル名を指定する。
10. Ctrl-A+D キーを押し、セッションを移動する。
11. “screen”を実行し、そのまま Enter キーを押す。
12. GPS の USB ポートを接続（事前準備で”/devttyUSB1”を設定した方）し、”sudo minicom [GPS の設定名]”を実行する。
13. Ctrl-B + Z キーを押し、N キーを押す。
14. 再び Ctrl-B + Z キーを押し、L キーを押す。そして、GPS のログのファイル名を指定する。
15. Ctrl-A + D キーを押し、セッションを移動する。
16. 再び Ctrl-A + D キーを押し、ログアウトする。

4.2.3　ログの取得の停止方法

データ収集が完了してログの取得を停止するには、下記を実行する。

1. Raspberry Pi にイーサネットケーブル、USB メモリを接続する。
2. Windows の場合は TeraTerm、macOS や Linux の場合はターミナルを起動し、”ssh

pi@raspberrypi.local”を実行する。

3. パスワードが聞かれるので、Raspberry Pi のパスワードを入力する（初期設定は”raspberry”）。
4. “screen -ls”を実行して、実行中のセッションを確認する。
5. “screen -r [仮想番号]”を実行して、AIS 受信のセッションか GPS 受信のセッションのどちらかに移動する。
6. Ctrl-B + Q キーを押して、minicom を終了する。これでログは保存される。
7. Ctrl-A + K キーを押し、Y キーを押す。これでセッションを kill する。
8. “screen -r”を実行して、もう片方のセッションに移動し、6 と 7 を実行する。
9. “sudo umount /media/pi/[USB メモリのパス]”を実行して、USB メモリをアンマウントする。
10. “sudo halt”を実行して、Raspberry Pi の電源を切る。

4.2.4　AIS デコード及び AIS・GPS の UTC timestamp の対応付け

本研究で用いている AIS 受信機は本来、専用の GPS 受信機があるのだが、それは壊れてしまっているため、別の GPS 受信機を使用した。このことで AIS をデコードした際に AIS を受信した UTC timestamp が GPS と同期されないため、12:00:00 を受信開始時刻としたデコードデータが出力されてしまうという問題があった。そこで、ログを取得する際に minicom で AIS と GPS のシリアル信号を受信した時刻について、Raspberry Pi の時計でタイムスタンプを記録し、ログデータをデコードする際に AIS と GPS のログのタイムスタンプを対応付けることで、AIS の UTC timestamp を GPS の UTC timestamp に置きなおして AIS デコードするようなプログラムを開発した。そのプログラムのソースコードを付録 A に載せる。

このプログラムは python3 で実行する。使い方を下記に記す。

1. 事前準備

   pip3 で以下のコマンドを実行し、ライブラリをインストールする。

   - “pip3 install libais”
   - “pip3 install pymap3d”
   - “pip3 install argparse”

2. 実行方法

   次のコマンドで実行することができる。Options については表 6 に記す。

   “python3 TimestampMatching.py [Options]”

   （例）AIS をデコードして、JSON で出力する

   “pyhton3 TimestampMatching.py -a [AIS ファイルパス] -g [GPS ファイルパス] -o [出力ファイルパス] -j”

   （例）AIS の緯度・経度を補間して距離を求め、CSV で出力する

   “python3 TimestampMatching.py -a [AIS ファイルパス] -g [GPS ファイルパス] -o [出力ファイルパス] -c -l -r 35.313034,139.783935,0”

表 6　TimestampMatching.py の Options

-a [AIS ファイルパス]	AIS ログのファイルパス（必須）
-g [GPS ファイルパス]	GPS ログのファイルパス（必須）
-o [出力ファイルパス]	出力（デコードデータ）のファイルパス （拡張子は書かないこと・必須）
-j	JSON で出力する （デフォルトは JSON で出力される）
-c	CSV で出力する
-d [スライドする日数]	AIS, GPS のタイムスタンプをスライドする日数 2 日後なら”-d 2”、3 日前なら”-d -3”
-l	AIS（Message Type 1, 2, 3）の緯度経度を 1 秒毎に線形補間する
-r [緯度],[経度],[高度]	指定した緯度・経度・高度から AIS（Message Type 1, 2, 3）の緯度・経度までの方位（degrees）、仰角（degrees）、距離（m）を計算する カンマ区切り、スペースなしで指定（”-l”必須）

4.3　データ収集の場所と環境

実験に必要なデータの収集を 2018 年 5 月 11 日、14 日、19 日の 3 日間、千葉県の富津岬にある明治百年記念展望台、東京海洋大学（越中島キャンパス）3 号館の屋上、東京湾アクアラインの海ほたる展望デッキで、合計 7 回データの収集を行った。図 31 にデータ収集場所の地図、11 日の木更津の天気を表 7、14 日の江戸川臨海（新木場）の天気を表 8、19 日の木更津の天気を表 9 に示す。また、表 10 に各データについての概要を示す。

国際 VHF の音量については、HX851 のメモリ 1.75 で固定した。また、スケルチについても同様に HX851 のメモリで設定をしている。

1 回目の録音ではボイスレコーダーは DMR-1800V（東芝）を使用した。録音の設定は SP（標準）モードにした。録音レベル調整は仕様上、自動調整となっていた。また、このボイスレコーダーは録音データをコンピュータに移動する手段がなかったため、録音終了後に Windows 10 のマイク端子とボイスレコーダーをつなぎ、ボイスレコーダーの再生音声を Voice Recorder というアプリで録音しなおした。2 回目以降の録音では GH-KANADT8（GREEN HOUSE）を使用し、録音設定は WAV（384kbps）とした。

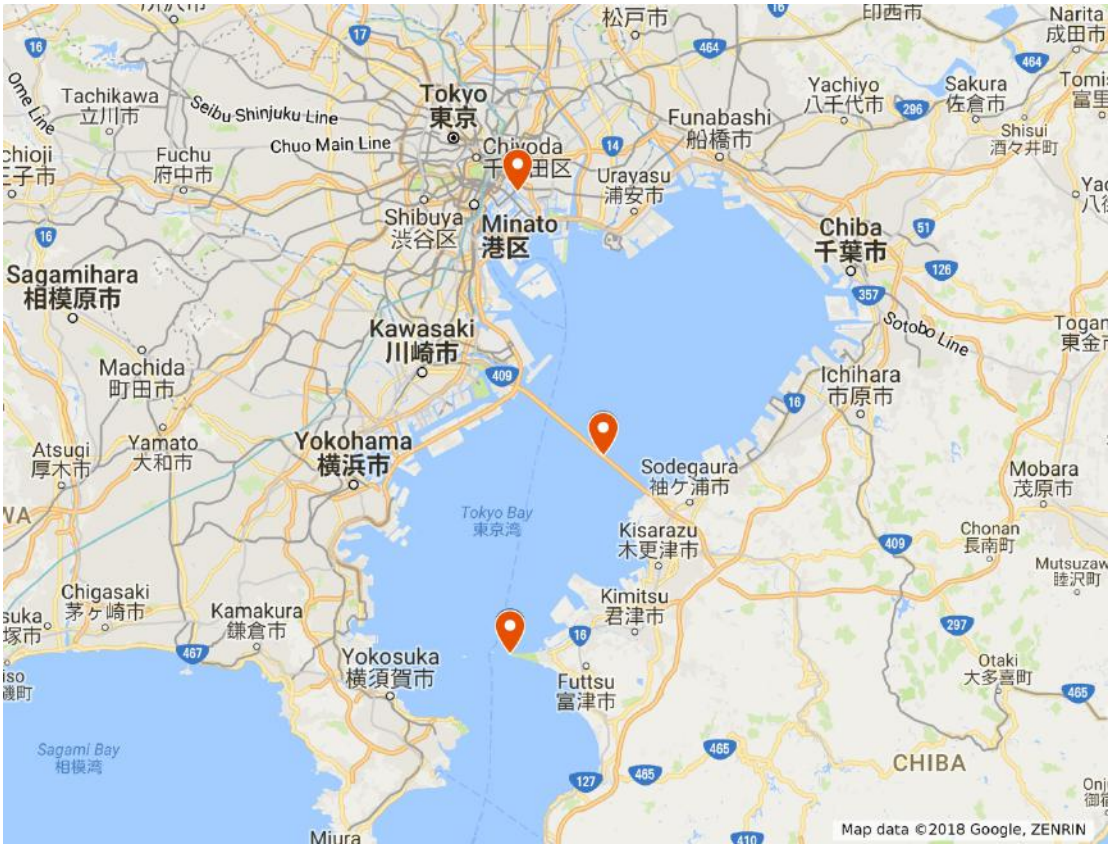

図 31　国際 VHF 通信と AIS 情報のデータ収集場所

（出典）Google「Google Maps」https://www.google.com/maps/

表 7　木更津の天気（5 月 11 日）

5 月 11 日	降水量（mm）	気温（℃）	風速・風向（m/s）		日照時間（h）
			風速	風向	
13:00	0	21.5	4.5	西南西	1
14:00	0	20.8	4.5	南西	1
15:00	0	20.8	4.4	南西	1
16:00	0	20.8	4.3	南西	1
17:00	0	20	3.2	南南西	1
18:00	0	18.6	2.9	南南西	0.6
19:00	0	17.5	1.7	南南西	0
20:00	0	17.1	3	南西	

（出典）気象庁「過去の気象データ検索」http://www.data.jma.go.jp/obd/stats/etrn/index.php

表 8　江戸川臨海（新木場）の天気（5 月 14 日）

5 月 14 日	降水量（mm）	気温（℃）	風速・風向（m/s）		日照時間（h）
			風速	風向	
13:00	0	27.9	4.4	北北西	1
14:00	0	28.8	5.7	北北西	1
15:00	0	28.5	8.4	北西	1
16:00	0	28	7	北	1
17:00	0	27.3	6.3	北北西	1
18:00	0	26	4.7	北北西	1
19:00	0	24.1	3.9	北北東	0.1
20:00	0	22.8	1.7	北北東	
21:00	0	21	2.1	北東	

（出典）気象庁「過去の気象データ検索」http://www.data.jma.go.jp/obd/stats/etrn/index.php

表 9　木更津の天気（5 月 19 日）

5 月 19 日	降水量（mm）	気温（℃）	風速・風向（m/s）		日照時間（h）
			風速	風向	
10:00	0	22	2.5	南西	0
11:00	0	23.1	1.4	南南西	0
12:00	0	24.4	1.3	北西	0.1
13:00	0	25.2	4.4	東北東	0.8
14:00	0	23	4.5	東南東	0.1
15:00	0	20.8	4.9	東	0
16:00	0	19.9	4.6	東	0
17:00	0	18.5	4.6	東	0.1

（出典）気象庁「過去の気象データ検索」http://www.data.jma.go.jp/obd/stats/etrn/index.php

表 10　国際 VHF 通信と AIS 情報のデータ収集概要

	1 回目	2 回目	3 回目	4 回目
日付	2018 年 5 月 11 日	2018 年 5 月 11 日	2018 年 5 月 11 日	2018 年 5 月 14 日
場所	明治百年記念 展望台	明治百年記念 展望台	明治百年記念 展望台	東京海洋大学 越中島キャンパス 3 号館 屋上
緯度	35.312802	35.312802	35.312802	35.666897
経度	139.785402	139.785402	139.785402	139.792313
スケルチ	0	0	0	0
開始時刻	12:01:49	13:35:45	16:33:48	14:44:38
終了時刻	13:23:13	15:30:25	19:41:10	20:08:19
録音時間	1:21:24	1:54:40	3:07:22	5:23:41
	5 回目	6 回目	7 回目	
日付	2018 年 5 月 19 日	2018 年 5 月 19 日	2018 年 5 月 19 日	
場所	明治百年記念 展望台	海ほたる 展望デッキ	海ほたる 展望デッキ	
緯度	35.312792	35.464797	35.464797	
経度	139.785382	139.873260	139.873260	
スケルチ	3	3	3	
開始時刻	11:10:27	14:46:42	16:24:54	
終了時刻	13:03:36	16:22:45	16:43:19	
録音時間	1:53:09	1:36:03	0:18:25	

4.4　音声データの分析結果

録音した音声データを聞き取り、そこで聞こえた音声から通信が行われた時刻、発せられたメッセージ、発信者、受信者、話者の性別、話者が日本人か、聞き取りやすさはどうかをまとめた。なお、「○○、こちら××。チャネル□□に変更願います。」「はい、□□。」のように、1 つ 1 つのメッセージでは船名などが入っていなくても会話の流れで船名が特定でき、かつ通話が聞き取れるものは、船名を特定したうえで聞き取れる通信と解釈している。

1 回目の音声データの発信者と通話回数の関係を表 11 に示す。1 回目の合計受信回数は 161 回あり、そのうち船舶からの発信は 60 回（37%）、VTS からの発信は 101 回（63%）だった。船舶からの発信のみでみると、60 回中 26 回（43%）は一部、またはほぼ聞き取れた。逆に通信が聞き取れなかったのは 60 回中 34 回（57%）あった。ポートラジオからの発信で発信者が特定できなくて、聞き取れなかったものは 101 回中 7 回（7%）であり、9 割はどの VTS からの通信かが特定できた。

4 回目の音声データは合計受信回数が 9 回しかなく、うち 5 回は内容が聞き取れなかった。

船舶からの発信は1回､横浜ポートラジオから1回､東京ポートラジオから2回発信があった。最もデータ収集時間が長かったが、船舶からの発信は1回しか聞き取れなかった。これは東京海洋大学（越中島キャンパス）と海岸の間にビルやマンションがあることや、東京港以外に入港する船舶の電波が受信できていない可能性が考えられる。

表 11　1回目の発信者と通話回数の関係

船舶からの通信で一部、またはほぼ聞き取れた	26	16%
船舶からの通信で聞き取れない	34	21%
ポートラジオからの通信で聞き取れない	7	4%
東京マーチスからの通信で一部、またはほぼ聞き取れた	39	24%
東京ポートラジオからの通信で一部、またはほぼ聞き取れた	14	9%
横浜ポートラジオからの通信で一部、またはほぼ聞き取れた	21	13%
川崎ポートラジオからの通信で一部、またはほぼ聞き取れた	11	7%
千葉ポートラジオからの通信で一部、またはほぼ聞き取れた	9	6%
第1回の受信合計	161	

6回目の音声データの発信者と通信回数の関係を表 12 に示す。6回目の合計受信回数は311回あり、そのうち船舶からの発信は169回（54%)、VTSからの発信は142回（46%）だった。船舶からの発信のみで見ると、169回中61回（36%）は一部、またはほぼ聞き取れた。逆に通信が聞き取れなかったのは169回中108回（64%）あった。

表 12　6回目の発信者と通信回数の関係

船舶からの通信で一部、またはほぼ聞き取れた	61	20%
船舶からの通信で聞き取れない	108	35%
東京ポートラジオからの通信で一部、またはほぼ聞き取れた	9	3%
東京マーチスからの通信で一部、またはほぼ聞き取れた	108	35%
横浜ポートラジオからの通信で一部、またはほぼ聞き取れた	10	3%
川崎ポートラジオからの通信で一部、またはほぼ聞き取れた	9	3%
千葉ポートラジオからの通信で一部、またはほぼ聞き取れた	6	2%
第6回の受信合計	311	

以上のデータを合わせると、一部またはほぼ聞こえた船舶の通信は 88 回であった。これらの音声データを聞き取るにはかなりの時間を要した。特にスケルチを0に設定した音声は、常にノイズが入っており、とても人が聞き取りにくい音声に対してはかなり時間を要した。6回目はスケルチを3に設定した分、弱い電波はカットされるためスケルチ0より音声データを分析しやすかった。しかし、二波モデルを考えたときに、必ずしも距離が遠い電波がカットされ

るのではないため、少し離れた距離の音声データが聞き取れなくなる可能性がありうる。

距離を推定するモデルを構築するには、様々な距離で発信した電波を、数多く分析しなければいけない。まず、データ量を考えると東京湾に面した障害物のない場所でデータ収集を行う必要がある。また、電波の受信が勝手にカットされないようにスケルチを 0 にする必要がある。ただし、スケルチを 0 にすると音声データの分析にものすごく時間がかかる。その上で、AIS 情報との対応付けをしなければならないため、この実験方法は得られるデータ数が少ないわりに、非常に時間がかかる。

逆に VTS のデータ数が船舶のデータ数より多い。これは、VTS の通信は国際 VHF 通信の出力が 50W であることから、船舶よりも遠いところまで電波が届くためと考えられる。また、船舶の通信より電波が強く、聞き取りやすいことが分かる。VTS の電波を発信するアンテナの位置が分かれば、VTS からの通信回数は多いので、距離を推定するモデルが構築できると思われたが、VTS はテロ・安全対策の関係でアンテナの位置は公表していない。そのため、VTS の通信から距離を推定するモデルの構築はできなかった。

国際 VHF から電波を送信することができれば、任意の送受信間の距離を設定してデータを収集することができるのだが、国際 VHF で電波を発信するには免許が必要であるため、海上で通信されている音声を受信してデータを収集した。しかし、このやり方では上記のようにデータ数が少なすぎるため、距離を推定するモデルを作成するには時間がかかる。したがって、この実験では距離を推定するモデルの構築はできないと判断し、残りの音声データの分析は断念した。

第5章　特定小電力トランシーバーによる音声ノイズと距離の分析

第 4 章の実験では国際 VHF 通信から通話内容から発信している船舶を特定して、AIS で距離を求めて分析を行おうとしたが、時間がかかる割にデータが少ないために距離を推定するモデルの構築はできないと判断した。データ量を増やすには国際 VHF ではなく、免許がなくても電波を発信できる特定小電力トランシーバーが有効であると考えた。そこで、特定小電力トランシーバーを使ってでデータを収集し、その音声ノイズと距離を分析することで、距離を推定するモデルができるか調べることにした。

5.1　特定小電力トランシーバーについて

無線電話用特定小電力無線局は通称「特定小電力トランシーバー」と呼ばれており、国内のみでの使用に限り、電波法に基づく免許を要しない特定小電力無線局の一種である。400MHz 帯を使用して近距離の音声通話を行うことができる。国際 VHF の周波数は 156.025MHz から 162.025MHz であることから、特定小電力トランシーバーは国際 VHF の周波数よりも高い。それにより、雨滴、水蒸気などの影響を受けて電波の減衰が大きくなる特徴がある [37]。

音声通信に使われる電波形式は、国際 VHF も特定小電力トランシーバー、どちらも同じ F3E（FM の単一チャネルのアナログ信号）が使われている [44, 45]。

ここで FM 復調器の出力における SN 比の求め方を式 11 に示す [46]。

$(S/N)_{out} = \frac{3\pi A_C^2 A_B^2 k_{FM}^2}{2N_0\omega_S^3}$	11

ω_Sは搬送波の角周波数であり、この式から SN 比はω_S^3（搬送波の角周波数）に反比例することが分かる。よって、国際 VHF と特定小電力トランシーバーは周波数が違うことで、SN 比に影響が出る。

特定小電力トランシーバーは周波数の違いによる国際 VHF の SN 比の違い、雨滴や水蒸気による減衰があるため、国際 VHF とは異なるノイズや音声が生じる可能性が考えられる。だが、電波形式は同じ F3E であるため、特定小電力トランシーバーを使って音声解析を行って距離を推定することで、国際 VHF と何か共通した結果を得られると考える。

特定小電力トランシーバーは大きく分けて 2 つある。1 つはレジャーや重要度の低い業務で使用される 421MHz 帯及び 440MHz 帯、422MHz 帯を使用するもの。一般に広く利用されており、誰でも簡単に入手することが可能である。空中線電力が最大 10mW であることから近距離通信しかできないという制限があり、通信距離の目安として市街地では 100～200m、郊外では 1～2km、見通しの良い場所では約 2km といわれている。空中線（アンテナ）は無線機本体に装着されていなければならず、アンテナを外す、給電線を使用することはできない。また、絶対利得が 2.14dB 以下でなければならない。通信時間についても制限があり、3 分間通信したら自動的に通信を終了し、2 秒経過しなければその後の通信を行わない機能を有さなければならない（一部例外がある）。また、送信装置から電波を発射する場合にそのチャネルが空いているかを調べるキャリアセンス機能が必要である。

もう 1 つは主に工場やプラントなどの事業所構内、建設・工事現場などで使用されている 413MHz 帯及び 454MHz 帯を使用するものがある。これは空中線電力が 1mW 以下の制限はあ

るが、給電線の使用が可能であり、通信時間やキャリアセンスの制限はない特徴がある。

本実験で使用した特定小電力トランシーバーは UBZ-LM20（KENWOOD）を使用した。**表 13** に UBZ-LM20 の仕様、図 32 に写真を示す。このトランシーバーは 442MHz 帯を使用しているトランシーバーで、通信時間に制限がついているタイプである。なお、UBZ-LM20 は電話外部マイク・スピーカーを接続することができ、マイク端子は 3.5φ ミニプラグ、スピーカーは 2.5φ ミニプラグになっている。

表 13　UBZ-LM20（KENWOOD）の仕様

送受信周波数	422.200～422.300 MHz（h1～h9ch） 422.050～422.175 MHz（1～11ch） 12.5 kHz ステップ
電波形式	F3E, F2D
周波数安定度	±4ppm
消費電流	送信時 70mA 以下 受信定格出力時 120mA 以下 （ラウドネス OFF 時） 受信待ち受け時 50mA 以下 セーブ時（平均）約 10mA
性能保証温度範囲	－10℃～＋50℃
電源電圧	定格電圧 DC 4.5 V（－接地）
送信電力	10mW
低周波出力	90mW 以上 （定格電圧、8Ω 負荷、10%歪）
受信感度	－8dBμ 以下（12dB SINAD）
寸法（突起物含まず）	幅 55.5×高さ 103.9×奥行 26mm
重量（重さ）	約 180g（単 3 アルカリ電池 3 本を含む）

図 32　UBZ-LM20（KENWOOD）

5.2　実験方法

この実験では 2 台のトランシーバー（UBZ-LM20）を用いて、1 台は音楽再生デバイスとマイク端子を AUX ケーブルで接続し、もう 1 台はボイスレコーダーとスピーカー端子を AUX ケーブルで接続した。なお、スピーカー端子は 2.5φ ミニプラグのため、スピーカー端子に 2.5φ ミニプラグ（オス）－3.5φ ミニプラグ（メス）変換アダプタを接続した。この実験のシステムの概要図を図 33 に示す。

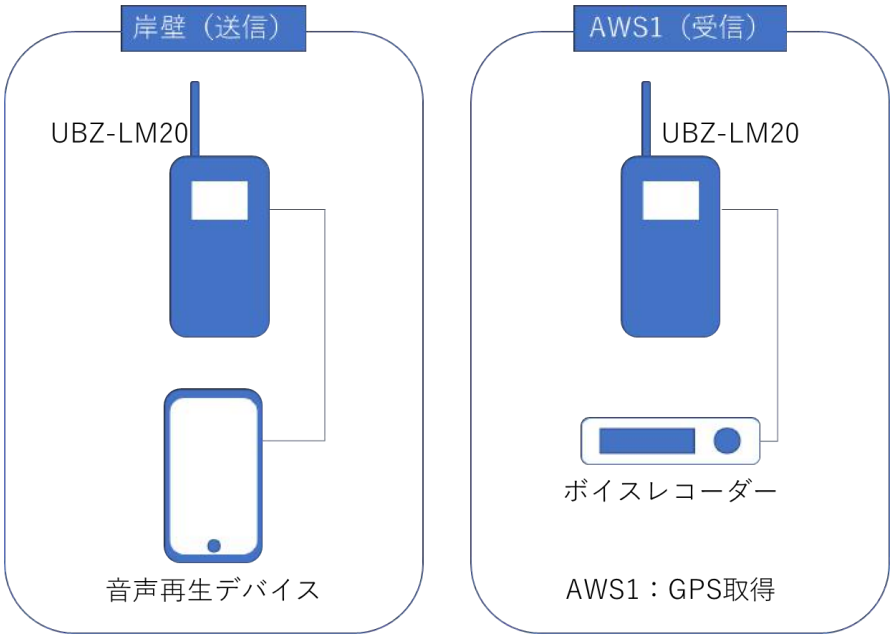

図 33　特定小電力トランシーバーによる実験でのシステム概要図

実験場所は送信を千葉県浦安市港（〒279-0024）の夕日の見える東屋付近の岸壁、受信をAWS-1という自動見張りシステムの実験を行うために使用している小型船舶を使い、東京湾の東京ディズニーシー沖で行った。AWS-1は自動見張りシステムで位置情報や船舶の運動（ロール・ピッチ・ヨーなど)、カメラの画像などを常時記録しており、AWS-1の位置情報からトランシーバーが信号を受信した時刻における距離を算出する。

この実験では最初にAWS-1はなるべく岸壁近くに近寄り、そこから沖に向かって走っていく。送信側は音声を再生して電波を発信し、受信側はボイスレコーダーでその音声を録音する。これを音声が受信できなくなるまで行う。

なお、この実験で作成したプログラムのソースコードを付録Bに載せる。

5.3　AWS-1について

AWS-1は松本洋平が所有する自動見張りシステム（Automatic Watching System; AWS）の実験を行うための小型船舶である。AWSはカメラ、AIS、レーダー、GPS、ジャイロセンサーから得た情報をコンピュータで統合、計算することで、衝突の恐れがある船舶を海上衝突予防法に従って自動的に避航する機能を有するソフトウェアである。AWS-1はAWSを実験するためにYAMAHA YF-24というと24ft小型船舶にカメラ、AIS、GPS、ジャャイロセンサー（AHRS)、コンピュータ（JetsonTK1、JetsonTX1、Zedboard）を搭載している。この電子システムはAWSによって、AWS1のオートパイロットに対して舵角と速力の指示を出している。また、AWS-1はコンピュータからオートパイロットに対して指示を出して操船するため、コンピュータにジョイスティック・コントローラーを接続して操船する、事前にウェイポイントを緯度・経度で設定して､そのウェイポイントを通過するように操船させることが可能である｡表 14にAWS-1の概要、図 34にAWS-1の外観、図 35にAWS-1の電子システム概略図を示す。

表 14　AWS-1（YAMAHA YF-24）の概要

船種	YAMAHA YF-24
船名	AWS-1
全長	7.20m
全幅	2.59m
全深さ	1.64m
質量	2600kg
最大乗員	8人

（出典）横佐古竜太「航海画像における色情報を用いた距離推定」

図 34　AWS-1 の外観

（出典）横佐古竜太「航海画像における色情報を用いた距離推定」

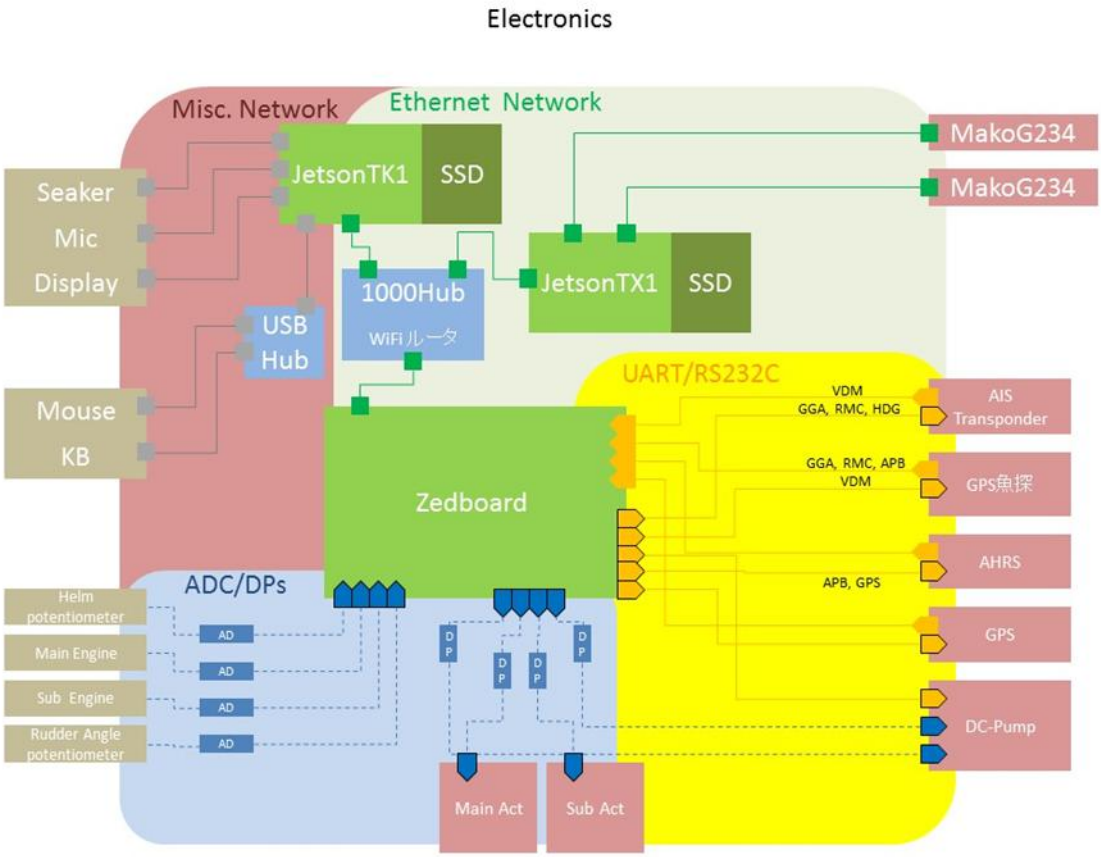

図 35　AWS-1 の電子システム概略図

（出典）横佐古竜太「航海画像における色情報を用いた距離推定」

AWS-1 は電子システムの各種データを記録しており、それをログデータとしてファイルに出

力することが可能である。この実験ではこの機能を利用して GPS の緯度・経度情報を取得し、そこから音声データを受信した時の距離を算出している。

5.4 音声データについて

第 4 章とは違い、この実験には音声データを用意する必要がある。そこで、この実験では以下の 7 つの音データを用意し、約 16 秒の音声データを作成した。

- 440Hz のビープ音
- 英語（男声）
- 英語（女声）
- 日本語（男声）
- 日本語（女声）
- “OK, Google”（日本語・男声）
- Soundless（音のないインターバル部分）

収録時にノイズがない、BGM などの効果音のない音声を使うため、英語はニュースのアナウンス、日本語は日本語学習用の音声を使用した。“OK, Google”は録音した音声をスマートフォンアプリ「Google アシスタント」に聞かせたときに、距離に応じてどこまで認識できるかどうかを調べるために収録した。また、音声では話者によって声質や音量が変わるため、440Hz のビープ音で距離による音声ノイズの傾向を調べた。Soundless は音のないインターバル部分を指しており、この部分の音声ノイズは純粋なノイズ音が得られる。この Soundless も分析することで音声ノイズの傾向を調べた。

なお、ビープ音の作成、音データの結合は「Audacity [47]」というフリーの音声編集ソフトを使用した。これはオープンソースであり、Windows や macOS、Linux などといった OS で動作する。また、Audacity はスペクトルやスペクトログラムの表示、ノイズリダクション機能など、音声編集や解析に使う様々な機能を有している。

図 36 に Audacity で見た実験音声データを示す。Audacity の上側は波形、下側はスペクトログラムを表示している。

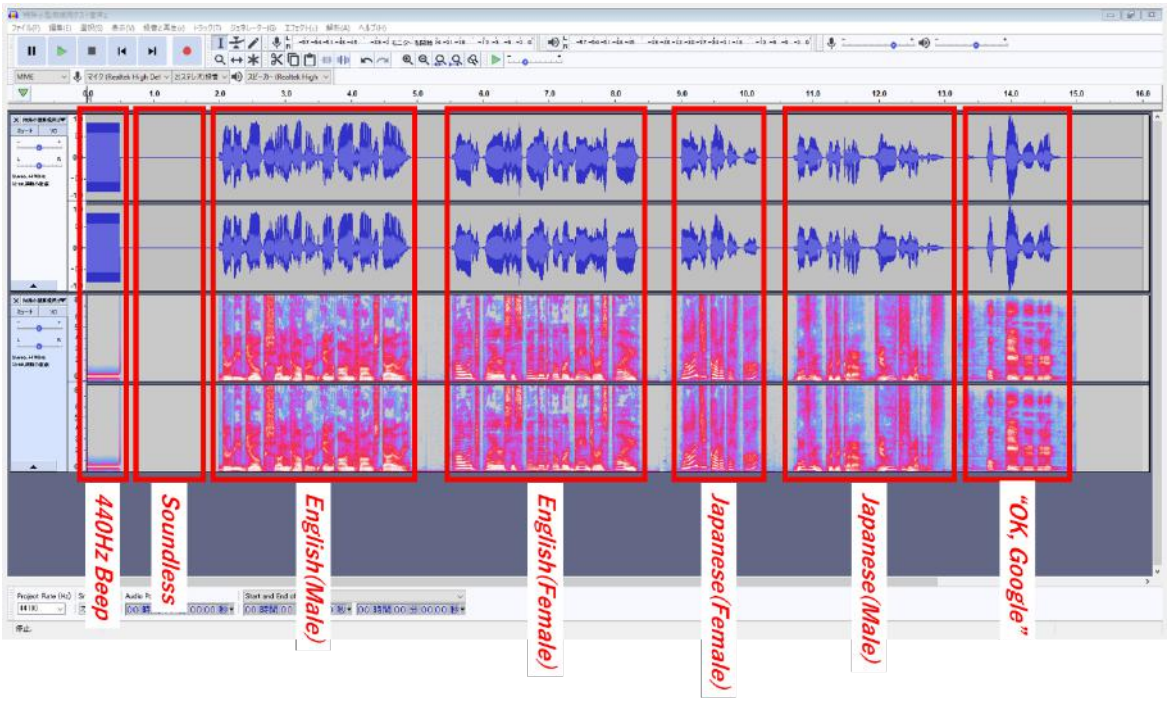

図 36 Audacity で見た実験音声データ（上：波形、下：スペクトログラム）

5.4　データ収集の場所と環境

実験は 2018 年 6 月 1 日、9 日の 2 日行った。ただし、1 日は送受信間距離を約 100～200m 間隔に設定してデータを収集し、その実験結果と反省を踏まえて 9 日は音声を連続再生・受信をしてデータを収集したため、1 日と 9 日では実験方法と結果の算出方法が異なる。

特定小電力トランシーバーの UBZ-LM20 は両日とも 8ch で行い、音量は一定にした。

5.4.1　6 月 1 日の実験について

1 日の実験では、送受信間距離を約 50～100m 間隔に設定してデータを収集した。まず、AWS-1 は送信側になるべく近い位置に停船し、送受信を行う。その後、約 100～200m 程度沖に移動し、再び送受信を行う。これを音声が聞き取れなくなるまで繰り返した。この実験時における送信側の設置の様子を図 37、送信側に接近した時の AWS-1 を図 38 に示す。

なお、トランシーバーはスケルチ OFF に設定したのだが、電源を一度切ると設定が初期化されてスケルチ ON になってしまっていた。当時はその仕様を知らずに行っていたため、これらの実験ではスケルチがある状態の結果になっている。

図 37　送信側の設置の様子

図 38　送信側に接近した時の AWS-1

音データの分析には Audacity のコントラスト機能とスペクトル表示機能を使用している。これは任意の音声データの範囲を指定することで、音量の実効値（dB）とスペクトルを調べることができる。この実験で得た音声データはこの機能を使い、7 つの音データの部分をマウスのドラック&ドロップで指定し、距離ごとにそれぞれの音データを分析した。

なお、スペクトル表示機能における Audacity の設定は表 15 のとおりである。また、実験で使用した音データの各スペクトルを図 39（線形表示)、図 40（対数表示）に示す。

表 15　スペクトル表示機能における Audacity の設定

アルゴリズム	スペクトラム表示
関数	Hanning window
サイズ	2048

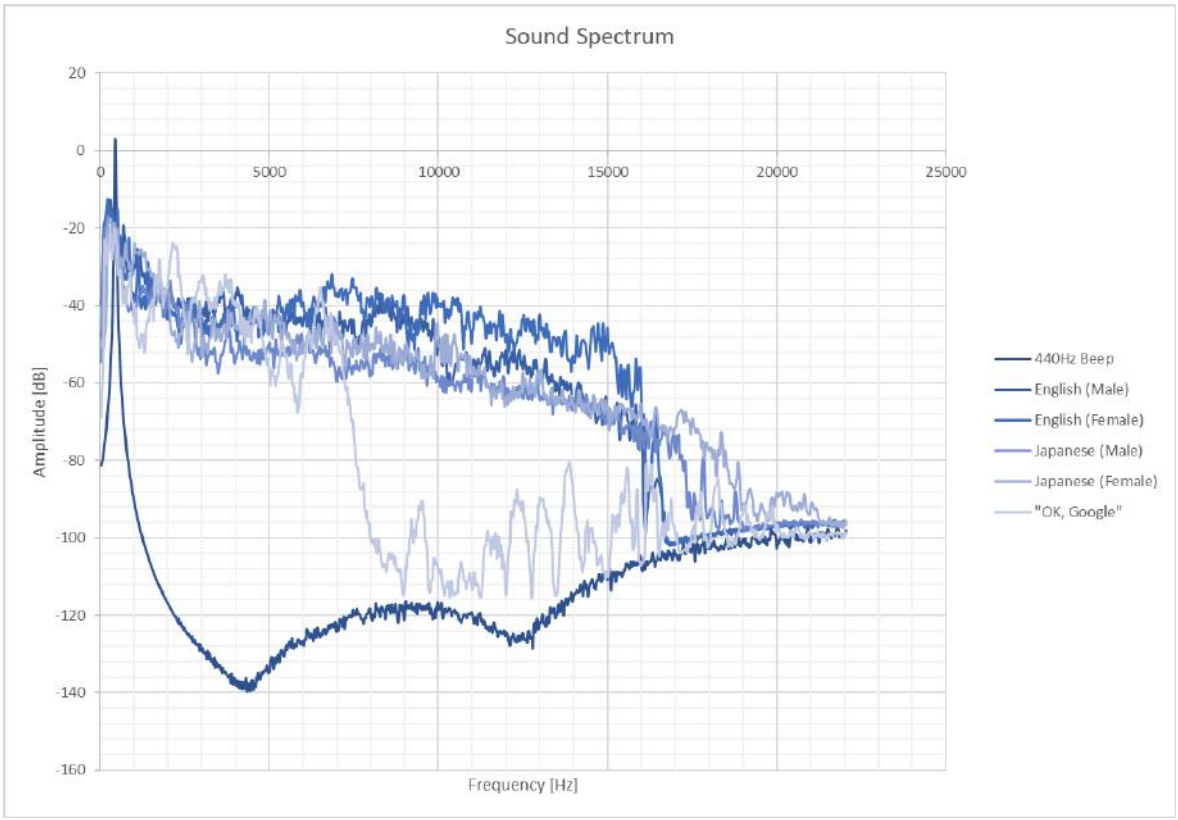

図 39　実験で使用した各音データのスペクトル（線形表示）

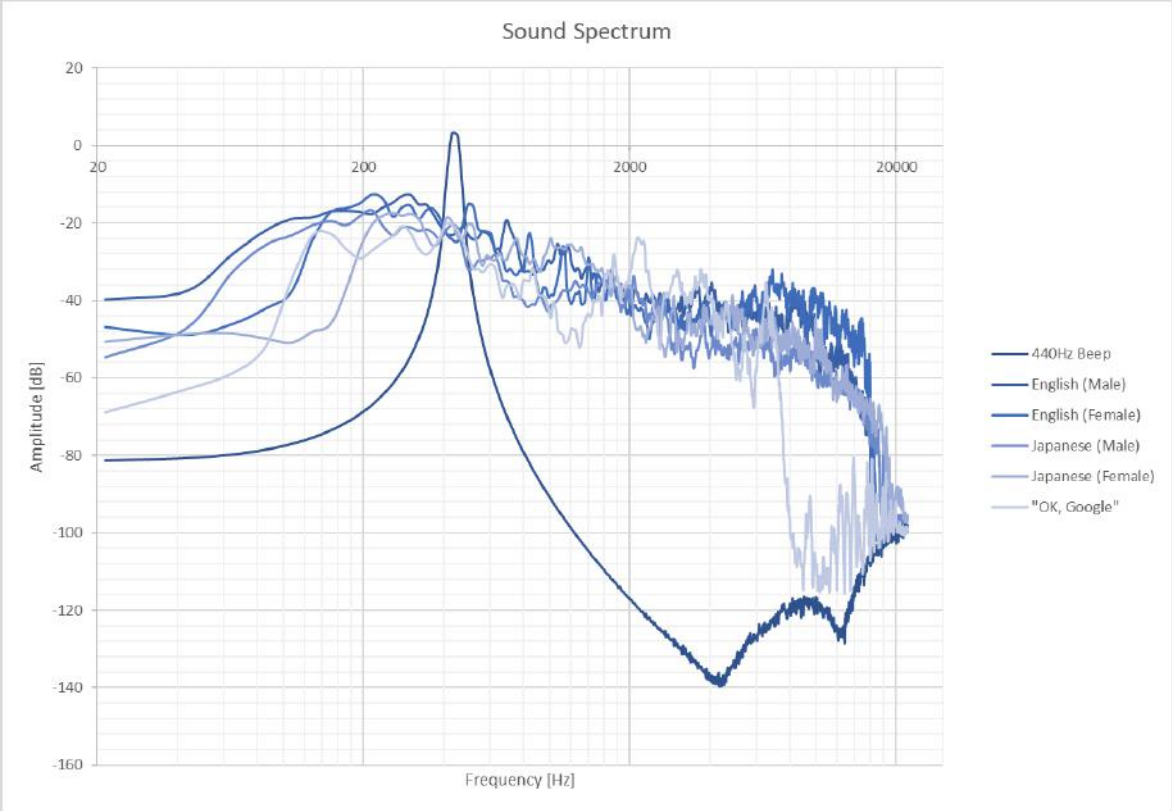

図 40　実験で使用した各音データのスペクトル（対数表示）

5.4.2　6 月 1 日の実験結果

この日の江戸川臨海（新木場）の天気を表 16 に示す。この日は東から南東あたりから風が吹いていて、徐々に風速が上がっていった。それに伴い、海面に波が立ったため AWS-1 も動揺が大きくなっていった。

表 16　江戸川臨海（新木場）の天気（6 月 1 日）

6 月 1 日	降水量（mm）	気温（℃）	風速・風向（m/s）		風速・風向（m/s）		日照時間（分）
			平均	風向	最大瞬間	風向	
14:20	0	24.9	3.6	南東	7.7	南南東	0
14:30	0	25	2.7	東北東	4.5	東	1
14:40	0	23.9	3.3	東	7.5	東南東	3
14:50	0	24	5.5	南東	9.2	南東	0
15:00	0	24	6.7	南東	9.4	南東	1
15:10	0	24	7.5	東南東	10.7	東	0

（出典）気象庁「過去の気象データ検索」http://www.data.jma.go.jp/obd/stats/etrn/index.php

音データは合計 12 回録音した。その受信した時刻と距離、観測者の受信した音データに対

する主観を表 17 に示す。距離は 118m から録音を開始し、2060m まで離れながら録音をした。1055m の地点で音声ノイズが目立ち始め、1265m の地点でホワイトノイズのような「サー」音が聞き取れるようになった。1620m の地点まで来ると、徐々に音声が途切れはじめ、1880m 以降はスケルチによって音が途切れる部分が出てくる。2060m の地点で音データによっては完全に無音になる部分があった。

また、実験では録音する際には 30 秒～1 分間再生・録音したが、その間で一部音が途切れる瞬間があった。観測者からは、上空の飛行機が近づいてくると受信しなくなり、通り過ぎた瞬間に受信するという報告があった。これは、飛行機が電波に対して何らかの影響を与えたために、電波強度が急激に下がって受信しなくなったと考えられる。

表 17　6 月 1 日の音データの時刻・距離・観測者の主観

No.	放送時刻	距離 [About m]	観測者の主観
1	14:25	118	
2	14:28	245	
3	14:31	400	
4	14:36	740	
5	14:40	880	
6	14:43	1055	ノイズが目立ち始める
7	14:47	1265	サー音がしっかり聞き取れる
8	14:49	1400	
9	14:52	1620	日本語女性, 男性, Google の音声が途切れ始める. 2 番目以降も所々途切れる
10	14:55	1760	2 回目の音声が英語女性から途切れ始め、日本語男性から音がなくなる.
11	14:58	1880	英語男女が無音あり. 1 回目はかなり無音
12	15:01	2060	英語女性が無音あり, それ以降無音, 1 回目か 2 回目かもわからない

音量の実効値による分析

各音データの音量の実効値（RMS Volume）と距離の関係をグラフにしたものを図 41 に示す。距離 2060m の時、英語（女声）、日本語（男声・女声）、“OK, Google”は音声が受信できなかった。

種類別にみると 440Hz のビープ音が最も音量が大きく、Soundless が最も音量が小さかった。各種音声データはビープ音よりは音量が小さかった。各種音声データがビープ音より小さいのは、音声には音量の起伏があるため、実効値にしたときに音量が小さい部分の影響を受けているとみられる。

距離ごとにみると Soundless は距離が離れるにつれて、徐々に音量が大きくなっている。他

の音データは距離が離れても、音量は大きく変化しなかった。つまり、人間が話していないときのノイズ音の音量は、距離が離れるにしたがって大きくなることが分かる。

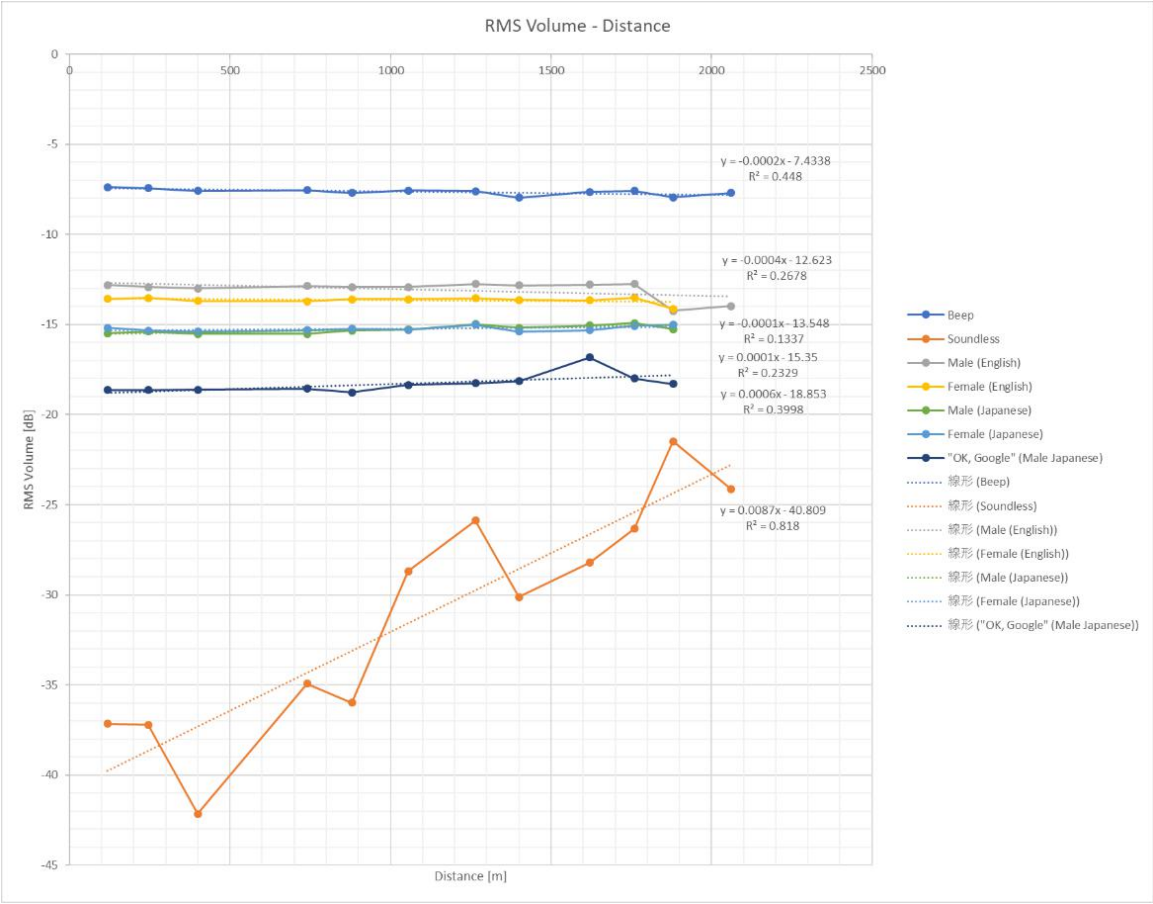

図 41　6 月 1 日の音量の実効値と距離

440Hz のビープ音のスペクトルによる分析

440Hz のビープ音のスペクトルを図 42（線形表示）、図 43（対数表示）に示す。

線形表示の方を見ると約 2kHz～6kHz の間で急激に音量が下がっている。また、この周波数帯では距離ごとの音量に差が生じている。約 6kHz 以上は 1400m を除いて距離が近いほうが音量は小さく、離れるにしたがって音量が大きい傾向はあるが、約 2kHz～6kHz 間のような音量の変化はなく、周波数が大きくなっても音量の変化はあまりない。1400m の音量は 1265m の音量より小さく、1055m に近い音量である。

対数表示を見ると約 200Hz 以下では 2060m が最も音量が大きく、次に 1880m が大きい。それ以外はほぼ似た音量である。そして、どの音量も約 200Hz にかけて小さくなり、約 200Hz～300Hz の間で音量が急激に大きくなる。約 200Hz～約 2kHz は距離ごとの音量の差はほぼなく、徐々に音量は小さくなる。約 2kHz～約 6kHz の間は、線形表示で述べたように距離ごとに音量の差が生じている。118m～1265m までは距離が離れるにつれて音量が小さくなるが、1400m は音量が 1265m より大きくなり、それ以上の距離になると再び音量が小さくなる。また、440Hz のビープ音は通常 440Hz のところで音量が最大になるが、トランシーバーを通した 440Hz のビープ音のピーク音量は約 500Hz のところにあり、次のピークが約 375Hz で、440Hz

の音量は 375Hz～500Hz の間で音量が小さくなっている。

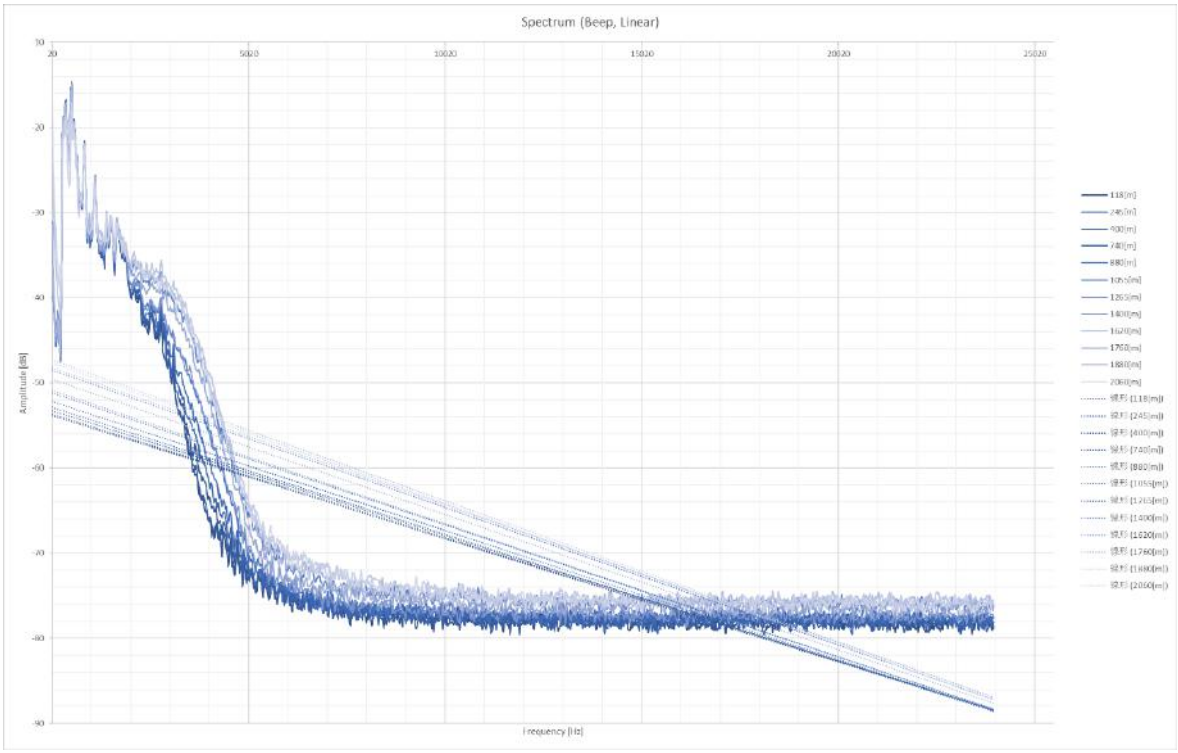

図 42　440Hz のビープ音のスペクトル（線形表示）

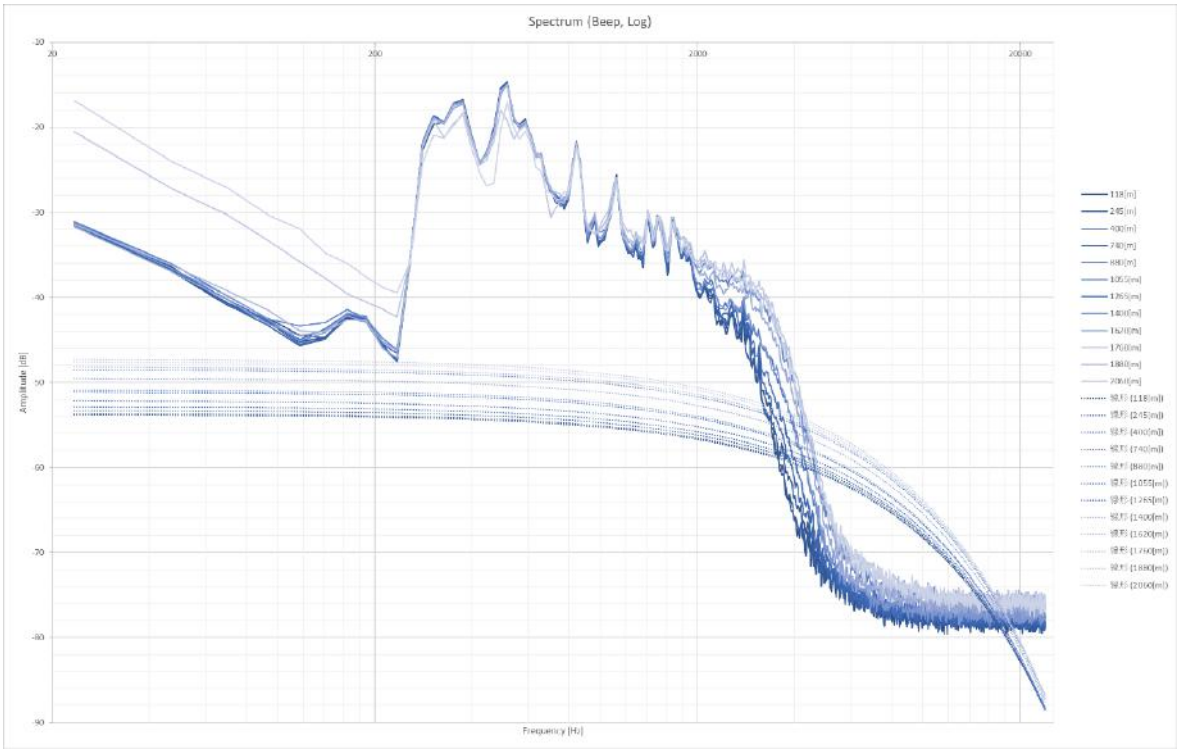

図 43　440Hz のビープ音のスペクトル（対数表示）

Soundless のスペクトルによる分析

Soundless のスペクトルを図 44（線形表示）、図 45（対数表示）に示す。

線形表示で見たときに、3kHz～6kHz の間で音量が急激に下がっている。また、この周波数帯では距離ごとの音量に差が生じている。約 6kHz 以上は距離が近いほうが音量は小さく、離れるにしたがって音量が大きい傾向はあるが、周波数が大きくなっても音量の変化はあまりない。1265m 以下は距離が離れるにしたがって音量が大きくなり、1400m で音量が下がる。そして、1400m 以上も距離が離れるにしたがって音量が大きくなる。これは約 1kHz～3kHz の間でも同じような傾向が見られる。

対数表示を見ると約 200Hz 以下の距離による音量の差は、線形表示の約 6kHz 以上のような傾向が見られる。880m 以下と 1400m は約 20Hz～50Hz の間で音量が急激に小さくなって、それ以外は約 200Hz にかけて緩やかに音量が小さくなっている。約 200Hz～300Hz の間でどの距離の音量も急激に大きくなる。118m の音量は約 300Hz～400Hz の間で急激に上がり、約 1kHz にかけて緩やかに音量が下がる。約 1kHz～約 2kHz は距離ごとの音量の差はほぼなく、約 2kHz～約 6kHz の間は、線形表示で述べたような傾向を示している。

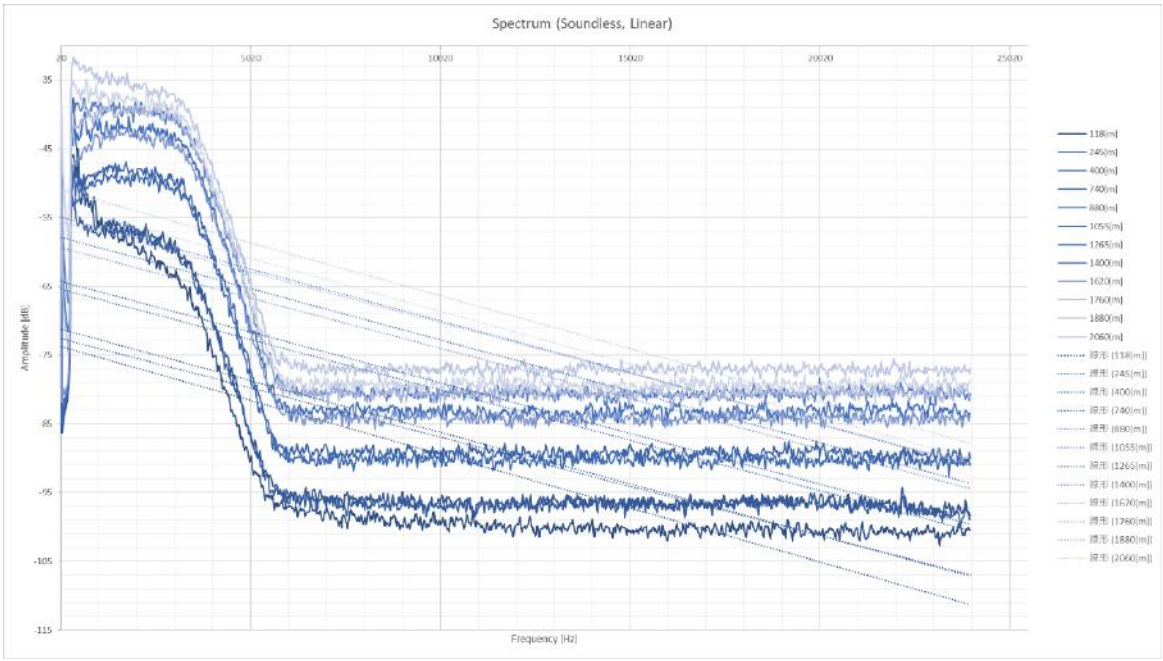

図 44　Soundless のスペクトル（線形表示）

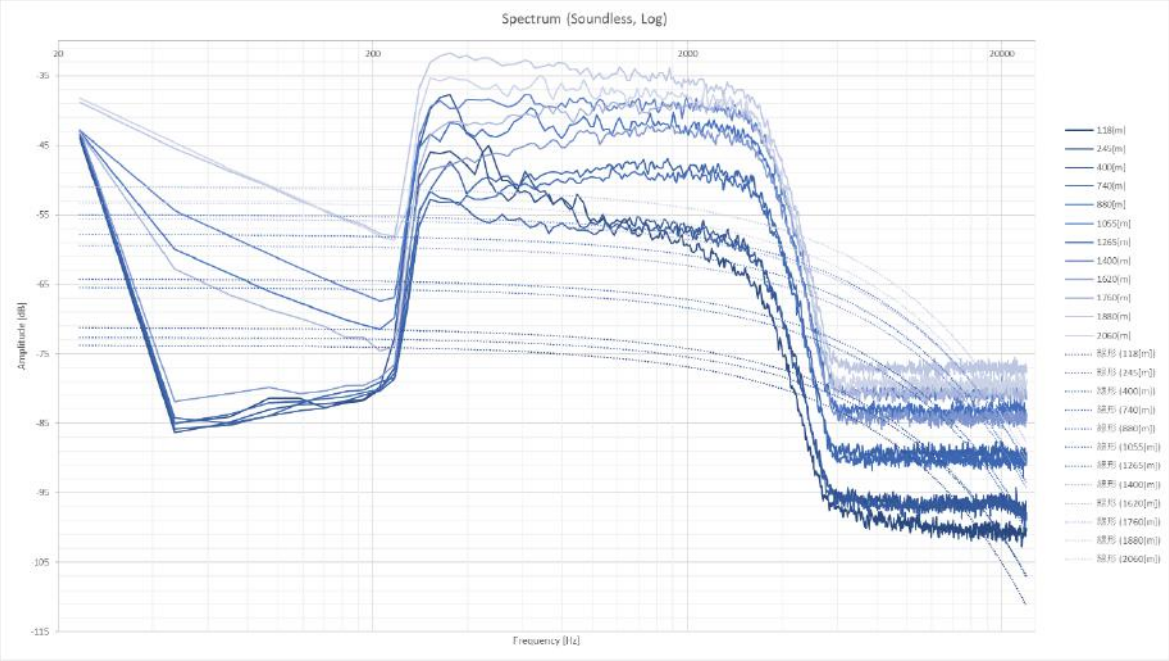

図 45　Soundless のスペクトル（対数表示）

音声（英語・日本語・”OK, Google”）のスペクトルによる分析

英語（男声）のスペクトルを図 46（線形表示）、図 47（対数表示）、英語（女声）のスペクトルを図 48（線形表示）、図 49（対数表示）、日本語（男声）のスペクトルを図 50（線形表示）、図 51（対数表示）、日本語（女声）のスペクトルを図 52（線形表示）、図 53（対数表示）、”OK, Google”のスペクトルを図 54（線形表示）、図 55（対数表示）に示す。

音声によって音量のピーク値の差異はあるが、どの音声も傾向として約 200Hz～300Hz で音量が急激に上がり、約 2kHz～6kHz にかけて音量が急激に下がり、約 6kHz 以上は音量の変化はほぼない。また、1265m 以下は距離が離れるにしたがって音量は下がり、1400m で音量があがる。そして、1400m 以上も距離が離れるにしたがって音量が下がる傾向がある。

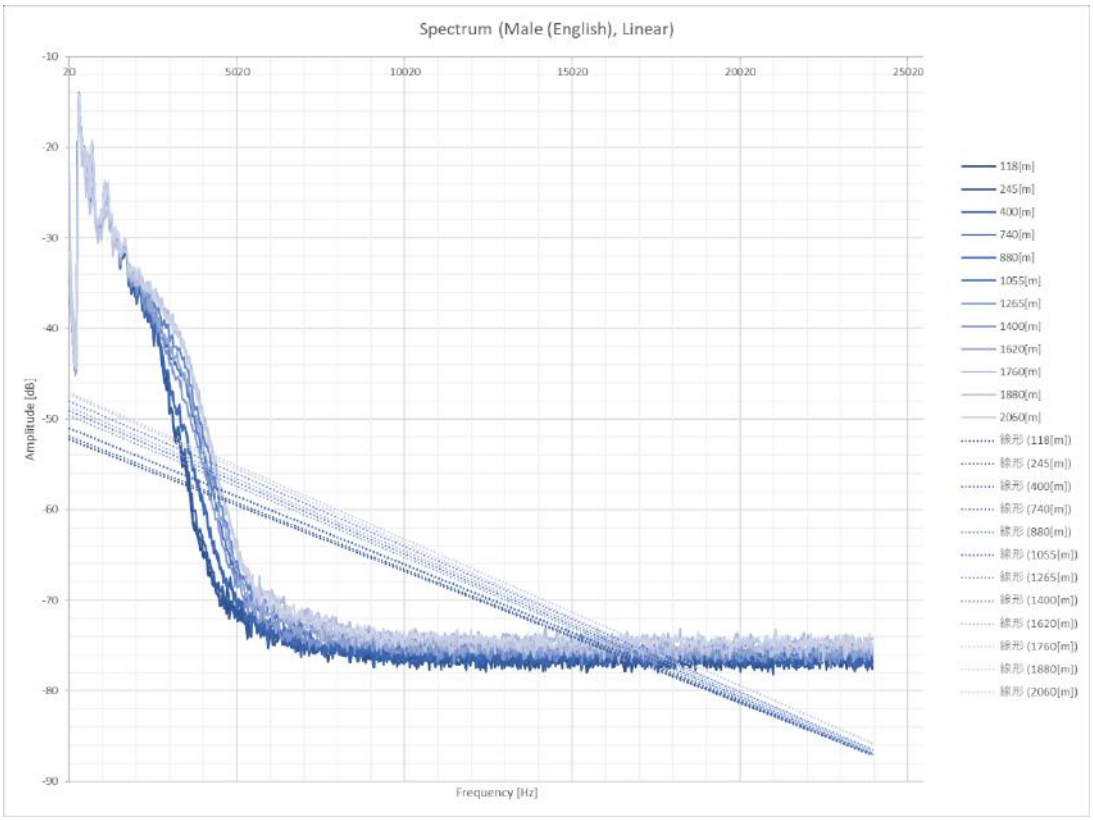

図 46　英語（男声）のスペクトル（線形表示）

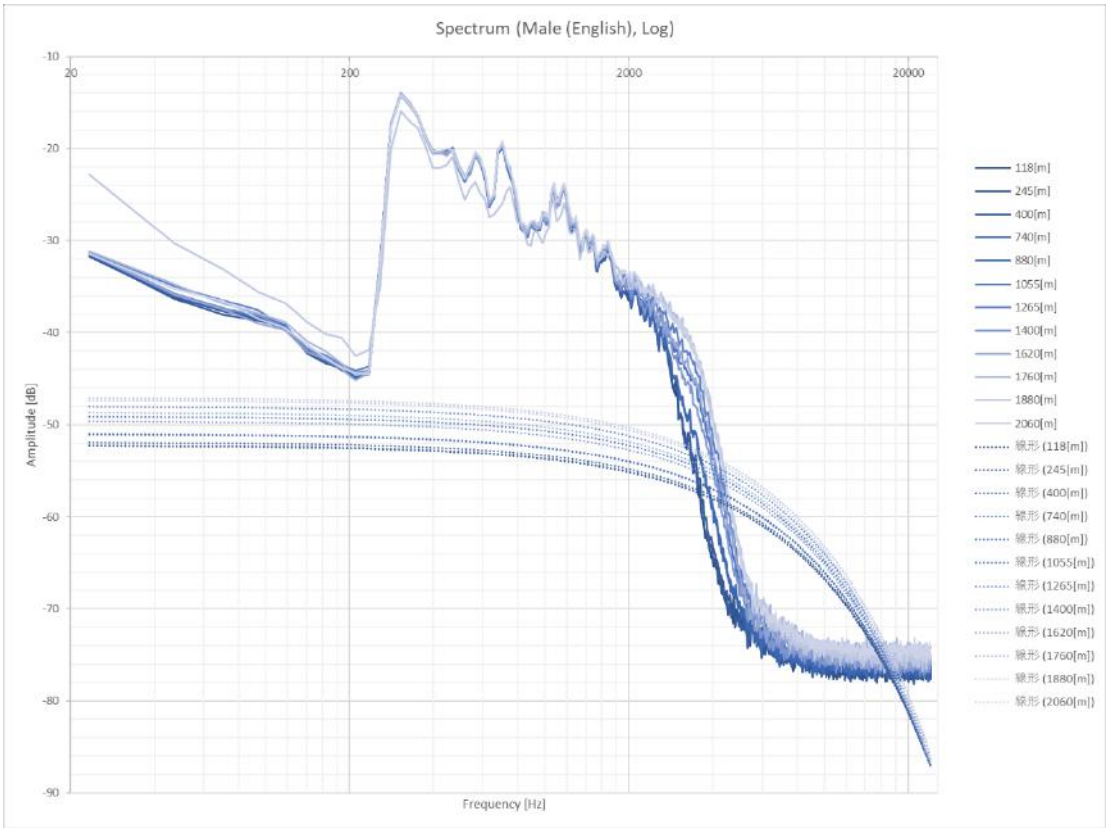

図 47　英語（男声）のスペクトル（対数表示）

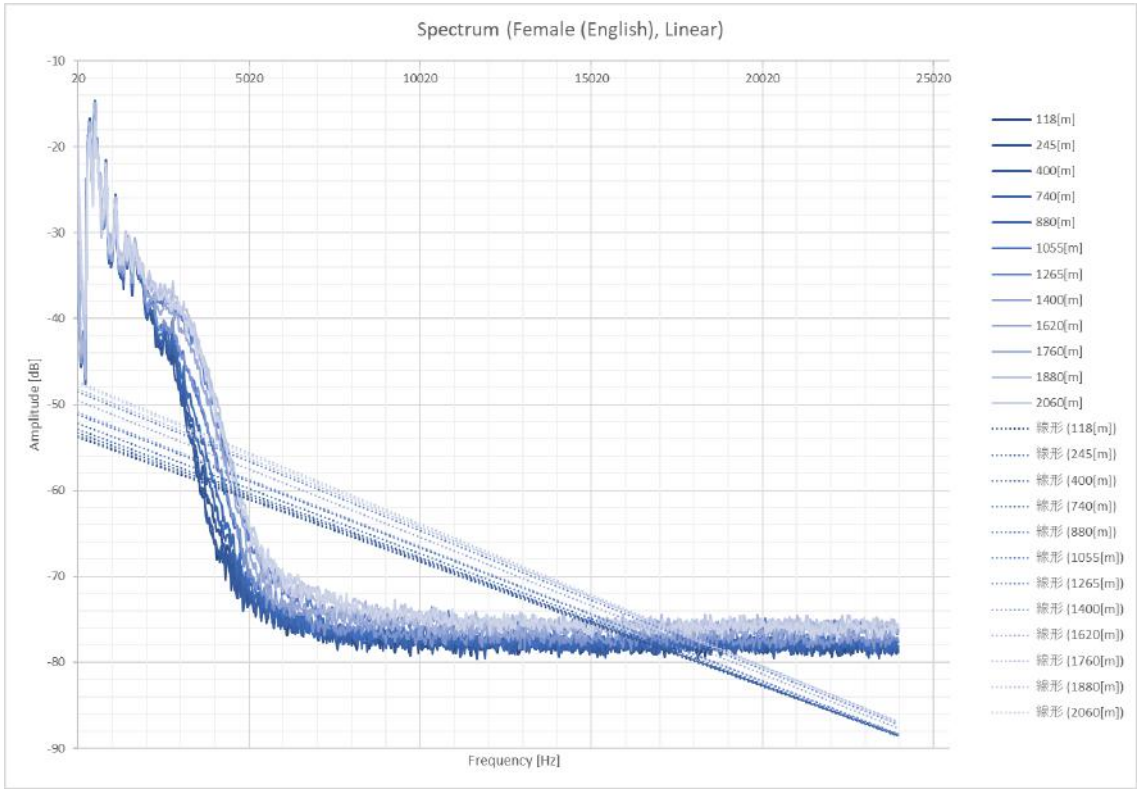

図 48　英語（女声）のスペクトル（線形表示）

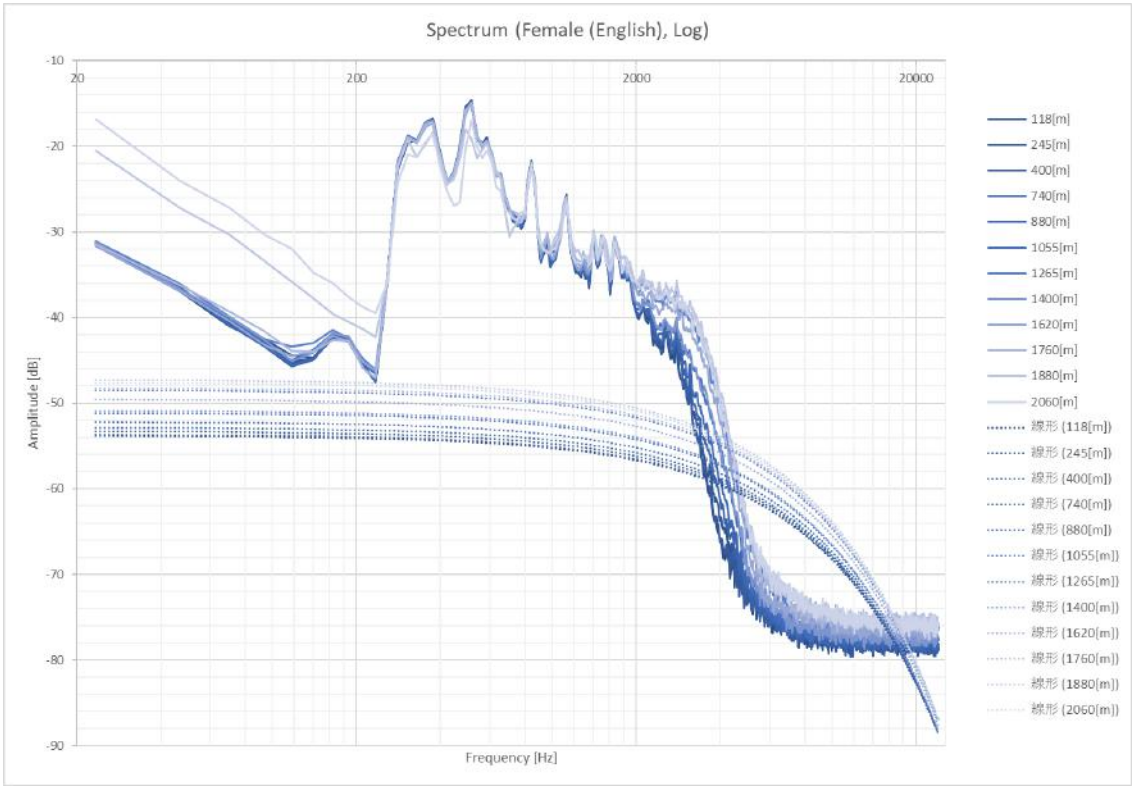

図 49　英語（女声）のスペクトル（対数表示）

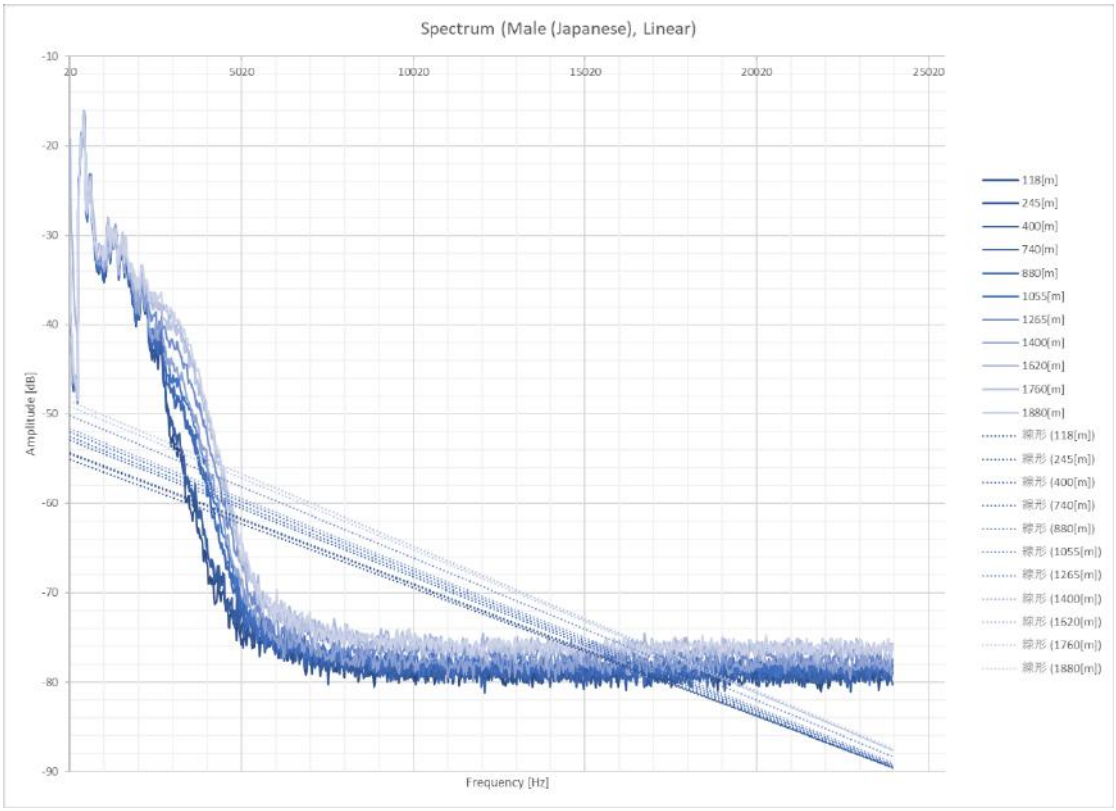

図 50　日本語（男声）のスペクトル（線形表示）

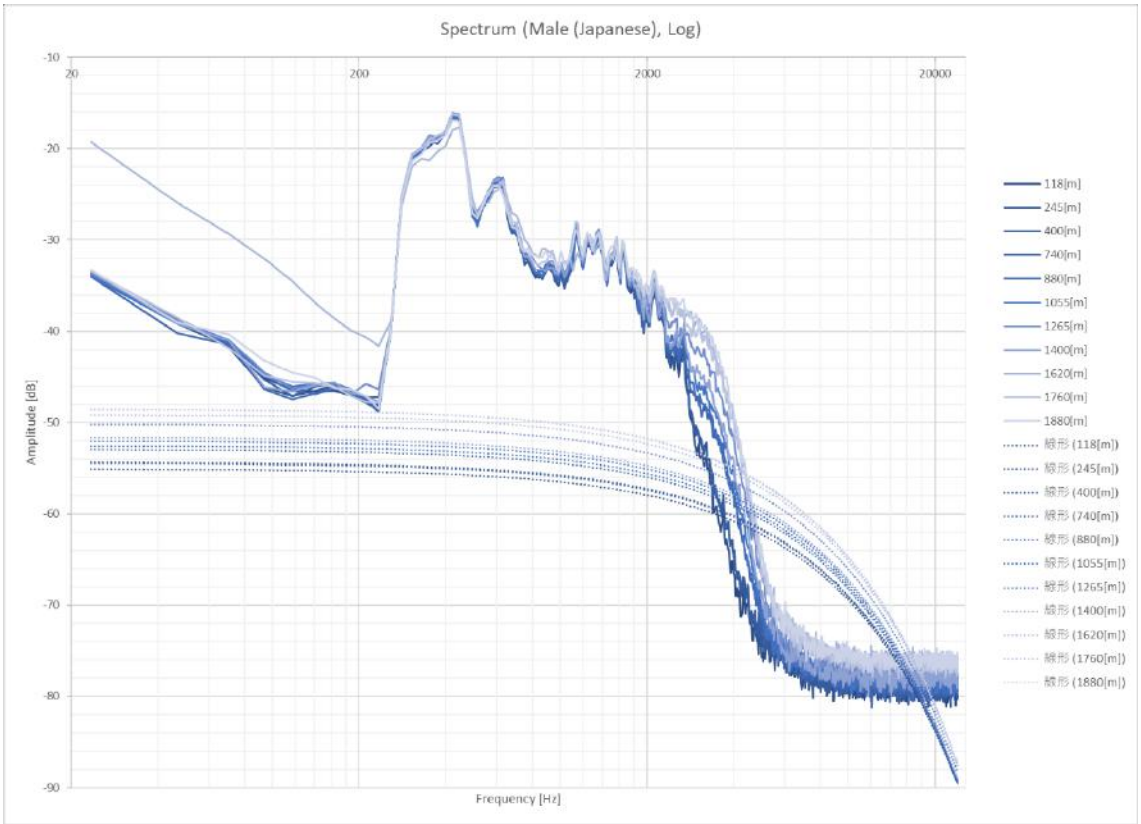

図 51　日本語（男声）のスペクトル（対数表示）

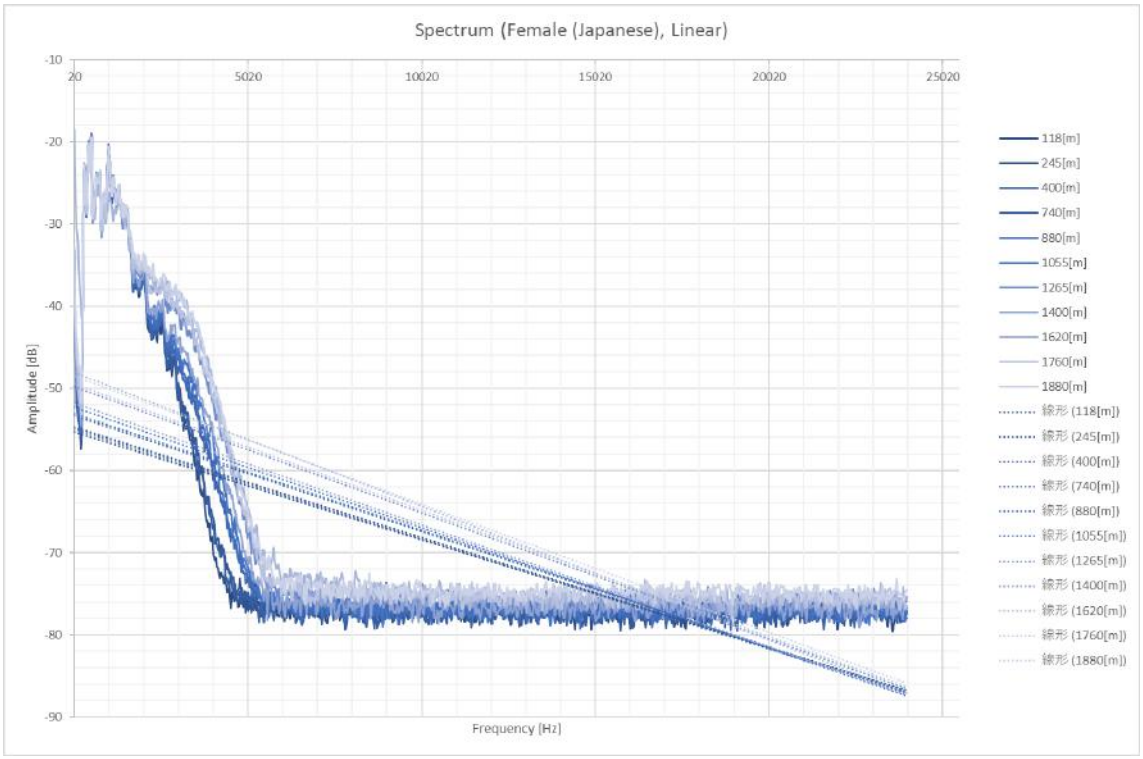

図 52　日本語（女声）のスペクトル（線形表示）

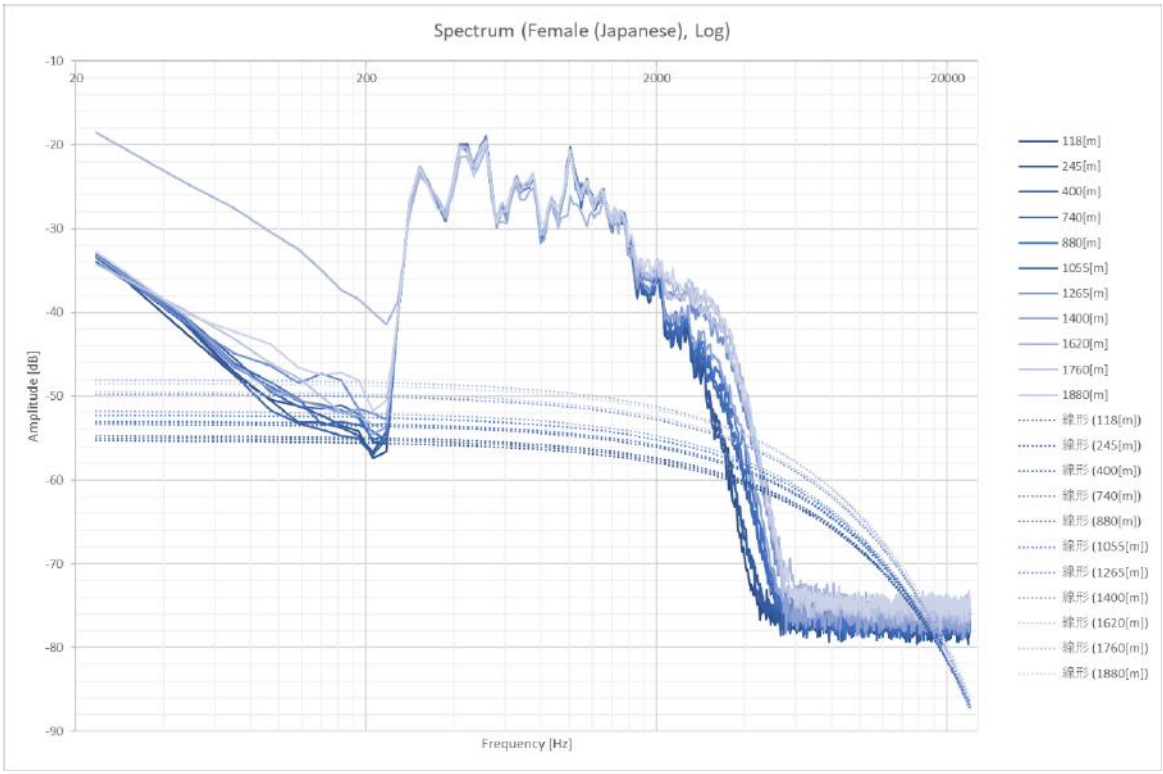

図 53　日本語（女声）のスペクトル（対数表示）

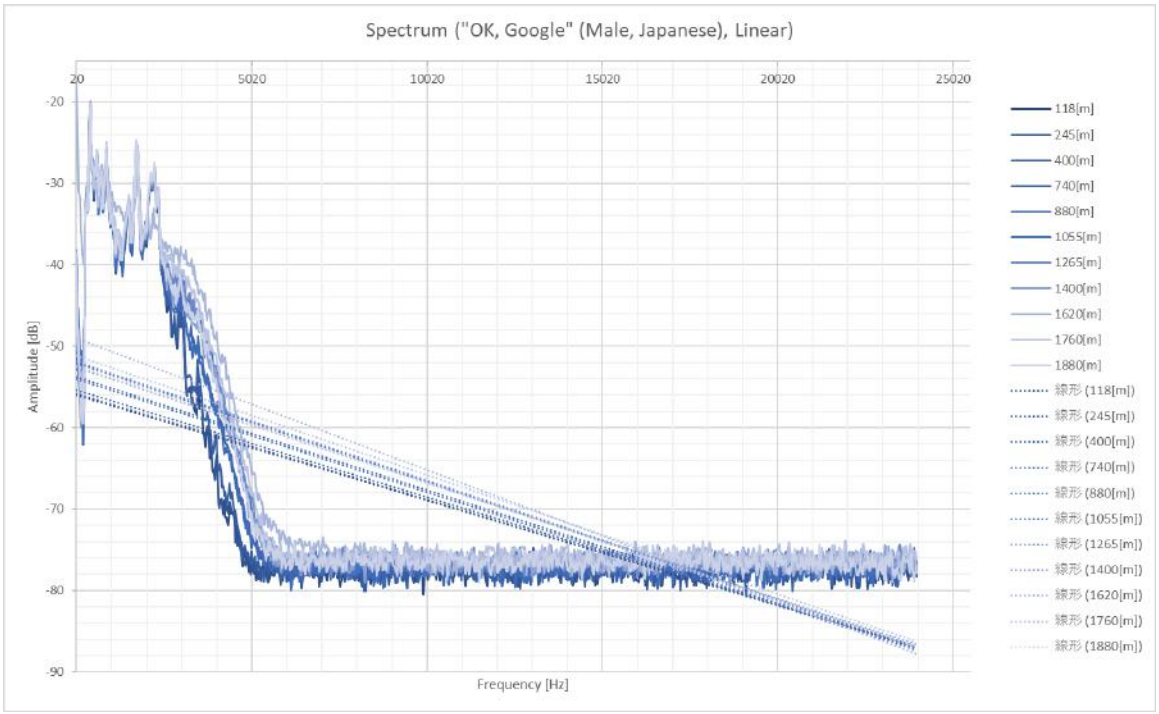

図 54　"OK, Google"のスペクトル（線形表示）

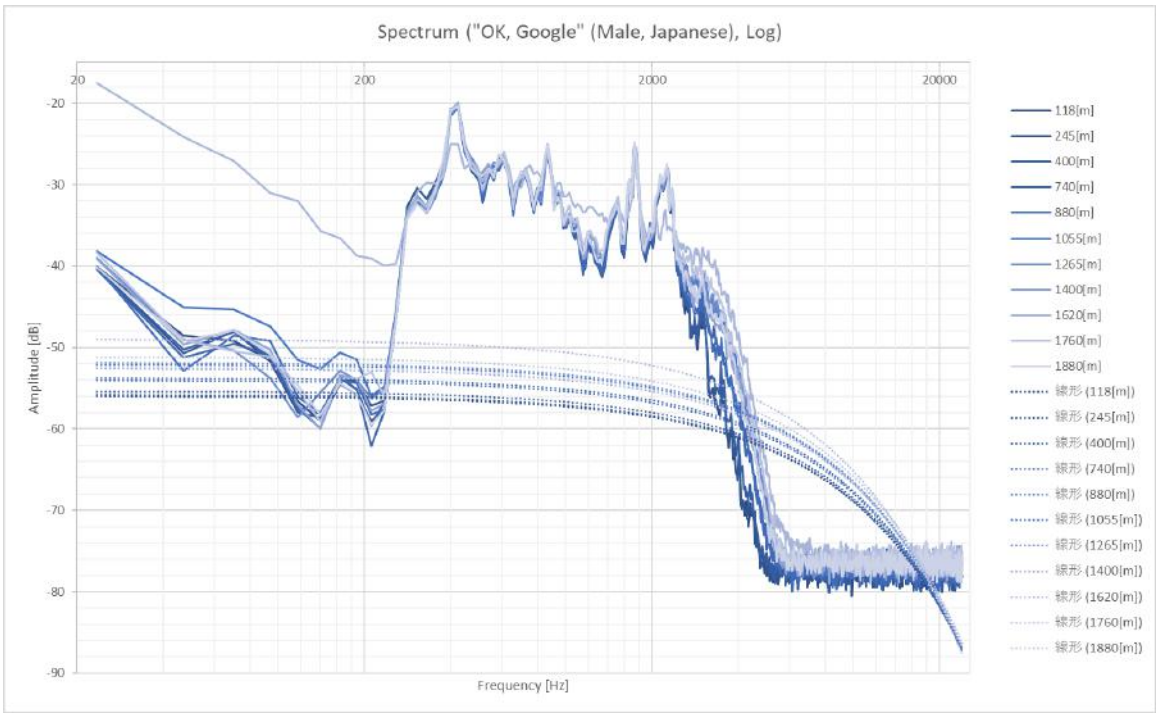

図 55　"OK, Google"のスペクトル（対数表示）

5.4.3　6 月 1 日の実験のまとめ

この 6 月 1 日の実験では以下の特徴が見られた。

1. Soundless の音量の実効値は、距離が離れるにしたがって音量が大きくなる。
2. 約 200Hz～300Hz 間は音量が急激に大きくなる。

3. 約 2kHz～6kHz 間は音量が急激に小さくなる。
4. 約 6kHz 以上は周波数による音量の変化はない。
5. 約 2kHz 以上において、1265m までは距離が離れるに従って音量が小さくなる。
6. 約 2kHz 以上において、1400m になると 1265m より音量が大きくなる。
7. 約 2kHz 以上において、1400m 以上も距離が離れるにしたがって音量が小さくなる。

全体を通してみると、これらの特徴から音声ノイズと距離には相関があり、約 2kHz 以上においては基本的に距離が離れるに従って音量が下がる傾向がある。

5.4.4　6 月 9 日の実験について

6 月 1 日の実験では音声ノイズと距離に相関があると仮説を立てた。ただし、1 日の実験結果のスペクトルはそれぞれの音全体のスペクトルで比較しているもので、スペクトログラムのような時間における周波数とパワーの変化については比較していない。

また、1 日の実験はデータ数が少ないことと、Audacity で人間がそれぞれの音を選択して音量の実効値とスペクトルを算出するのに時間がかかることから、この実験のやり方を改良することにした。

実験用音データは 7 種類の音データの再生時間が分かっているため、録音データ内に録音されている実験用音データの開始時点が分かれば、自動的に 7 種類の音データを分析して波形、スペクトラム（FFT)、パワースペクトル密度（PSD)、スペクトログラム（STFT）を出力するようにプログラムを作成した。

また、実験用音データを連続再生・録音し続けても、AWS-1 は常に GPS により位置情報を取得しているので、連続再生・録音しているすべての音データに対して距離を求めることができる。UBZ-LM20 は 3 分ごとに電波が発信できなくなり、次の発信ができるまで 2 秒のインターバルがある仕様から、3 分 2 秒ごとに電波を発信することは可能である。UBZ-LM20 は 1 回発信ボタンを押すことで、3 分間電波を発信し続ける機能があるため、実験用音データの連続再生・発信・録音ができる。

よって、1 日の実験方法を一部変更し、実験用音データは音声再生デバイスでリピート再生し、その間はボイスレコーダーで録音し続ける。そして、UBZ-LM20 は 1 度の発信で 3 分間発信し続けるように設定し、3 分 2 秒ごとに電波を発信する。そして、連続録音データをプログラムで分析を行った。なお、スケルチは OFF にしたため、スケルチによる音データのカットはない。

5.4.5　6 月 9 日の実験結果

実験は 10:11 に開始し、1 時間 15 分 25 秒データを録音した。この日の江戸川臨海（新木場）の天気を表 18 に示す。この日の風はそこまで吹き上がらなかったため、1 日に比べて AWS-1 の動揺は小さかった。

表 18　江戸川臨海（新木場）の天気（6 月 9 日）

6 月 9 日	降水量（mm）	気温（℃）	風速・風向（m/s）		風速・風向（m/s）		日照時間（分）
			風速	風向	最大瞬間	風向	
10:00	0	26.6	4.8	南南東	5.6	南南東	10
10:10	0	27	4.5	南南東	5.4	南南東	10
10:20	0	27.4	4.5	南南東	5.5	南南東	10
10:30	0	27.3	4.3	南南東	5.3	南南東	10
10:40	0	26.5	4.1	南南東	5.4	南	7
10:50	0	26.9	4.3	南	5.5	南	9
11:00	0	27.1	4.4	南	5.5	南	10
11:10	0	27.4	4.5	南	5.9	南	9
11:20	0	26.7	4.5	南	5.9	南	8
11:30	0	27	4.5	南	5.8	南	8
11:40	0	27.4	4.7	南	5.9	南	9
11:50	0	27.4	4	南	5.7	南	9

（出典）気象庁「過去の気象データ検索」http://www.data.jma.go.jp/obd/stats/etrn/index.php

この音データを Audacity で見たときの波形とスペクトログラムを図 56 に示す。28 分 40 秒ごろから 36 分 0 秒ごろの間の波形の振幅が、他の振幅に比べて小さくなっているが、再生して音データを確認したみたところ、若干音声ノイズに変化を感じたが、音声に対しての変化は感じられなかった。また、スペクトログラムを確認する分には、振幅が小さくなる部分の前後ではスペクトログラムに大きな変化は見られなかった。

連続再生した音データを実験音データの長さ（約 16 秒）ごとに分割した結果、246 分割することができた。実験時、音楽再生デバイスでリピート再生したでの、プログラムである一定間隔で音データを区切ることで分割することができると考えたのだが、リピートをするたびに再生するタイミングが違ったため、プログラムで実験音データの長さに区切るのはできなかった。また、特定小電力トランシーバーの特性による 3 分毎の通信遮断により、発信は人間のタイミングで行ったため、実験音データの最初の部分が通信遮断の時だと、その実験音データを切り出すことができなかった。距離がかなり離れたところでも、実験音データが聞き取れなかったため、そういった実験音データの最初の部分が特定できなかったものは、今回の分析には含まれていない。

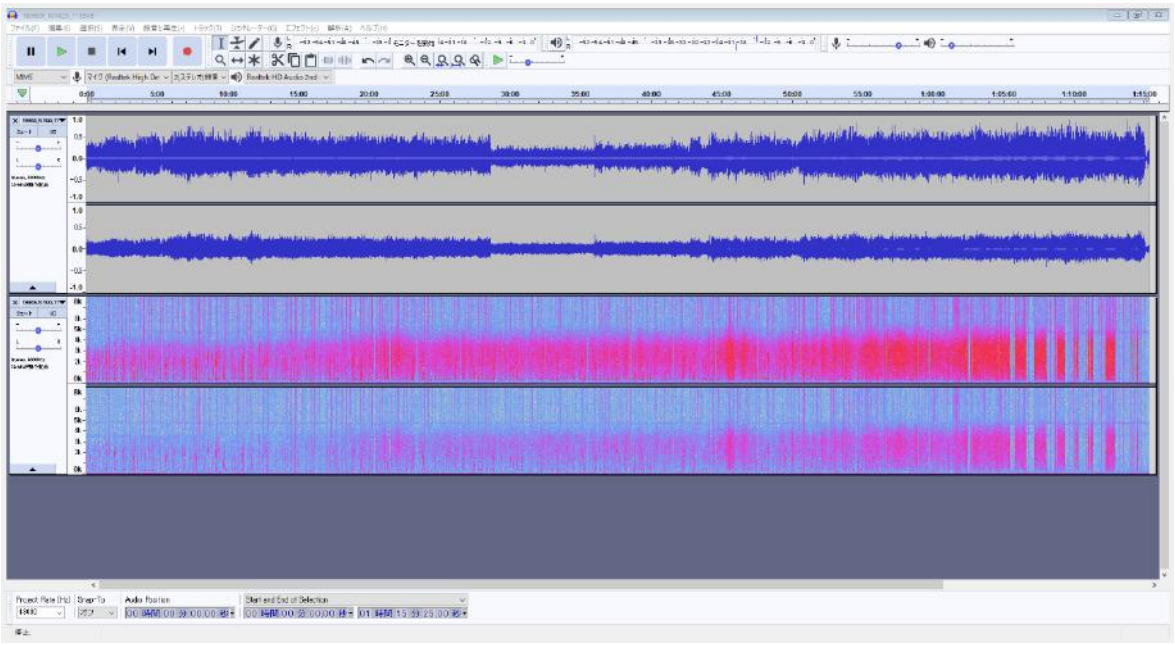

図 56　6 月 9 日の実験音データ（上：波形、下：スペクトログラム）

246 分割した実験音データごとの送受信間距離を図 57 に示す。実験開始時は 120m 離れており、終了時は 2219m であった。

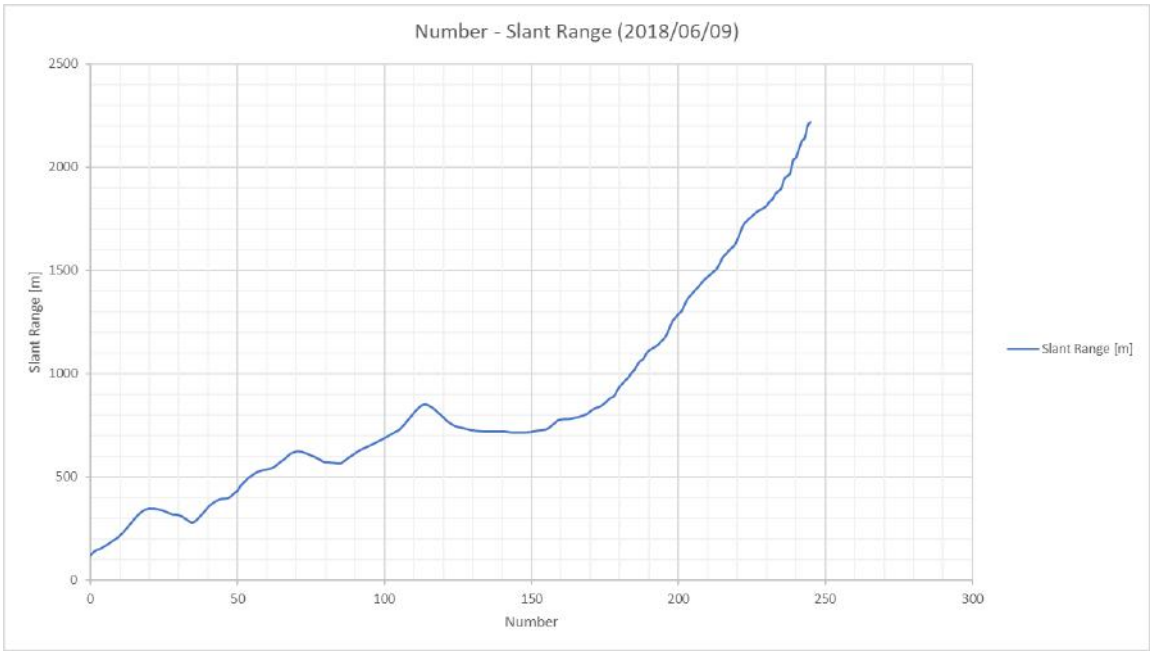

図 57　実験音データごとの送受信間距離

音量の実効値による分析

1 日と同様に、音の種類ごとに音量の実効値を計算した。音量の実効値と距離の関係を図 58 に示す。

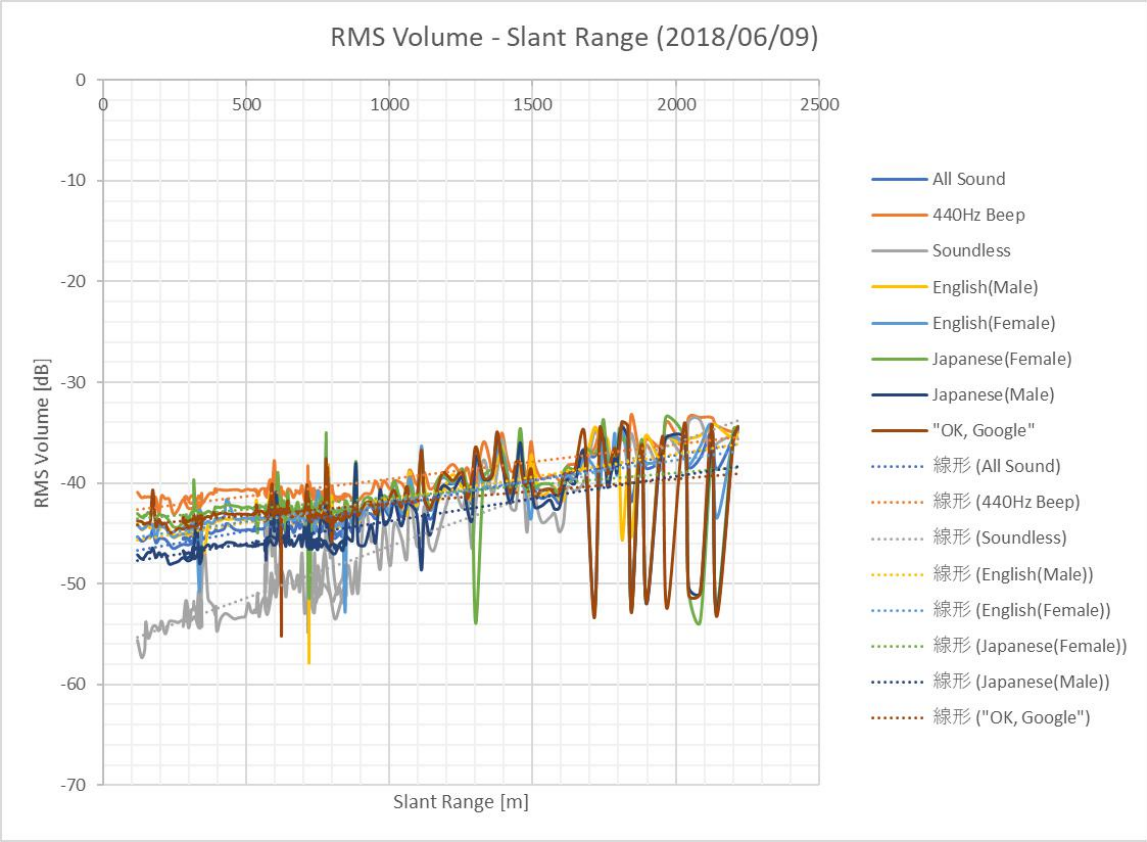

図 58　6 月 9 日の音量の実効値と距離

1 日の実験の比較項目に All Sound を追加した。これは実験音データ全体の音量の実効値を計算したものである。

Soundless を除く他の音データは主に 1700m 付近以降、急激な音量の減少が目立つ。スペクトルを確認すると音声ノイズも受信しておらず、ほぼ完全に無音状態になっていた。実験当日は飛行機が近くを飛んでいるのが観測された。これは 1 日の実験で飛行機が近くを飛ぶと電波を受信しなくなる現象があったのではないかと考える。

Soundless は距離が近いと音量が小さく、距離が離れるに従って徐々に大きくなるのが分かる。これは 1 日の実験結果と同様であることが分かる。

Soundless 以外の音データは 900m 程度までは音量に大きな変化はないが、1000m を超えたあたりから音量が少し大きくなっている。線形近似も緩やかに音量が大きくなっていることから、距離が離れるにしたがって音量が上がることがいえる。

波形による分析

音が途切れていないデータの中から、距離ごとに音データの波形を比較する。距離 120m の波形を図 59、距離 497m の波形を図 60、距離 806m の波形を図 61、距離 1003m の波形を図 62、距離 1272m の波形を図 63、距離 1470m の波形を図 64、距離 2219m の波形を図 65 に示す。なお、各図の No Sound は Soundless のことである。

波形からいえるのは、どの音データも距離が遠くになるにつれて、Soundless の振幅が徐々に

大きくなり、他の音データについても距離が遠くなることで音声ノイズと思われる振幅が目立ち始める。特に 1003m のデータからは顕著に大きくなり、最終的に音の信号が音声ノイズによって被されてしまい、ほぼ音の信号の振幅が見られなくなることが分かる。

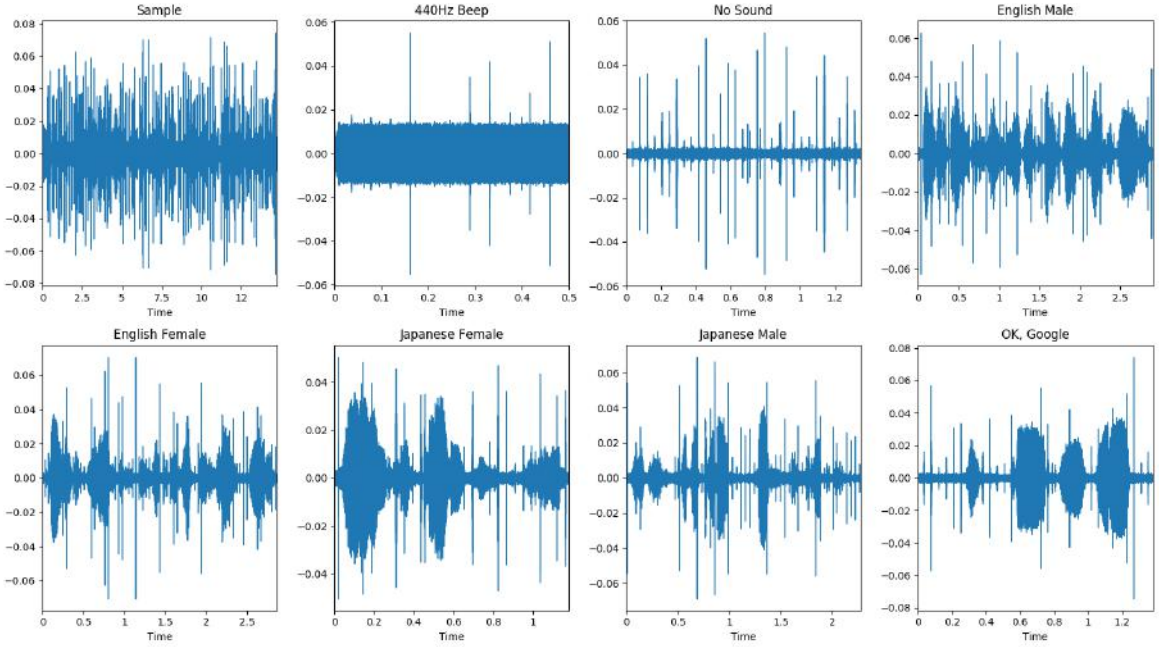

図 59 距離 120m の波形

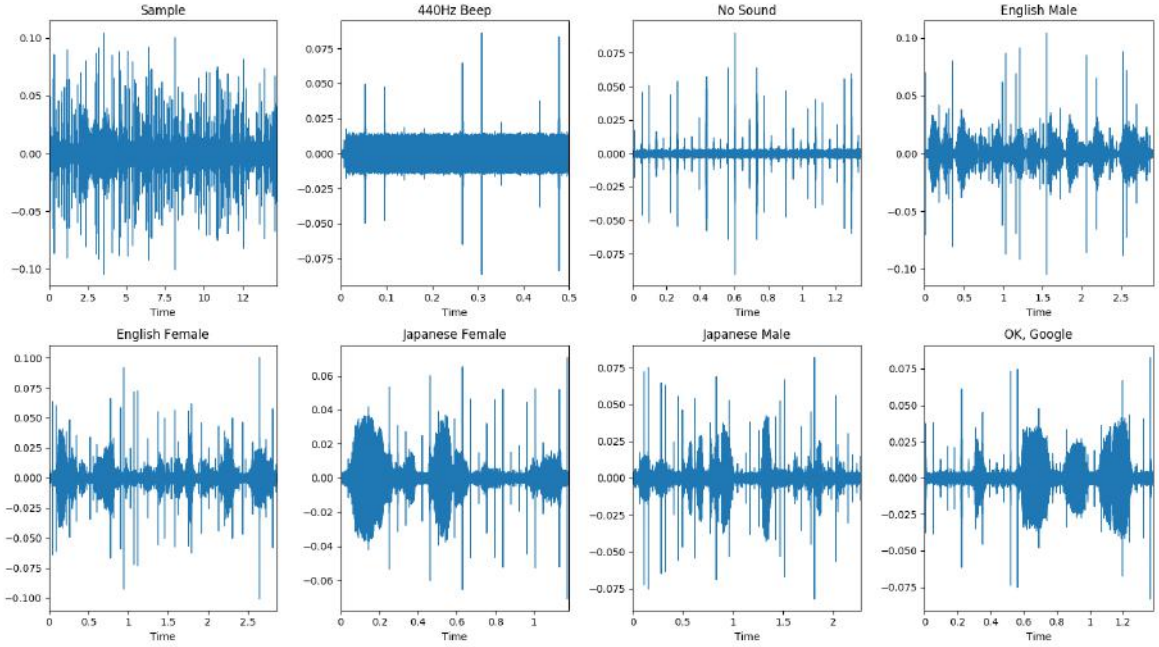

図 60 距離 497m の波形

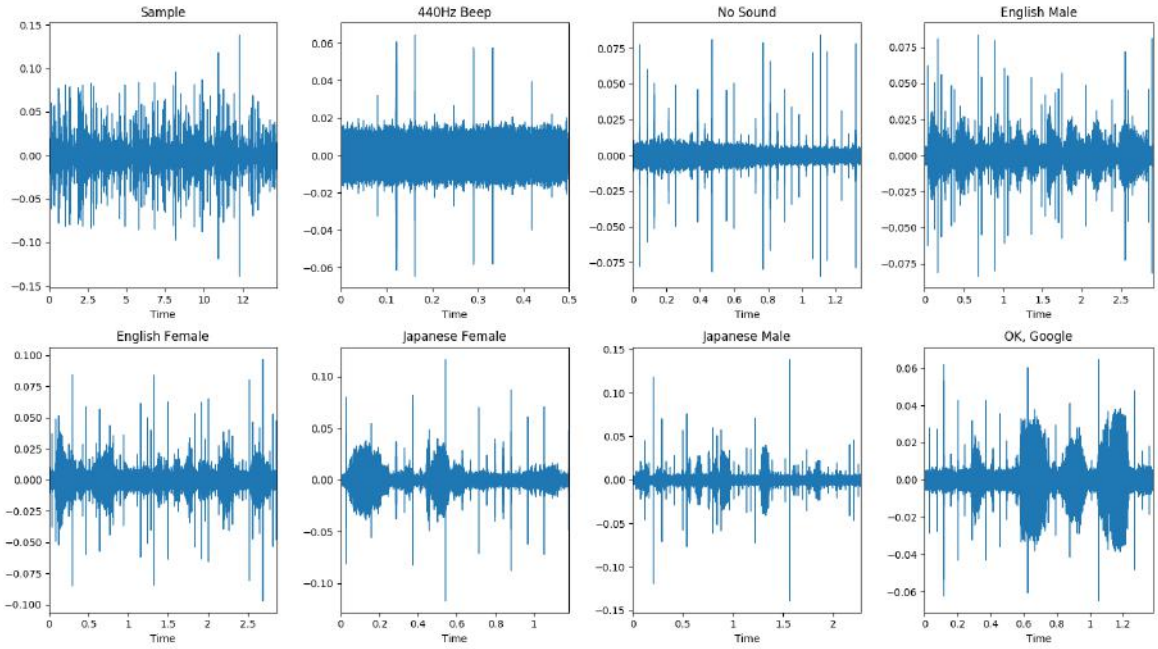

図 61　距離 806m の波形

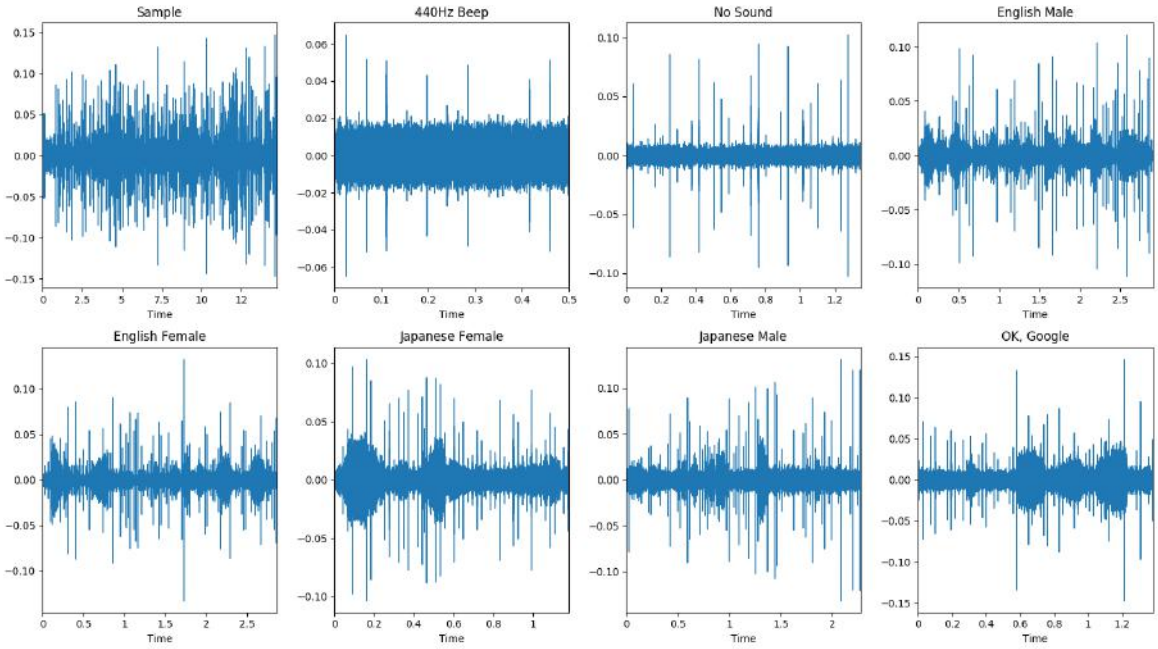

図 62　距離 1003m の波形

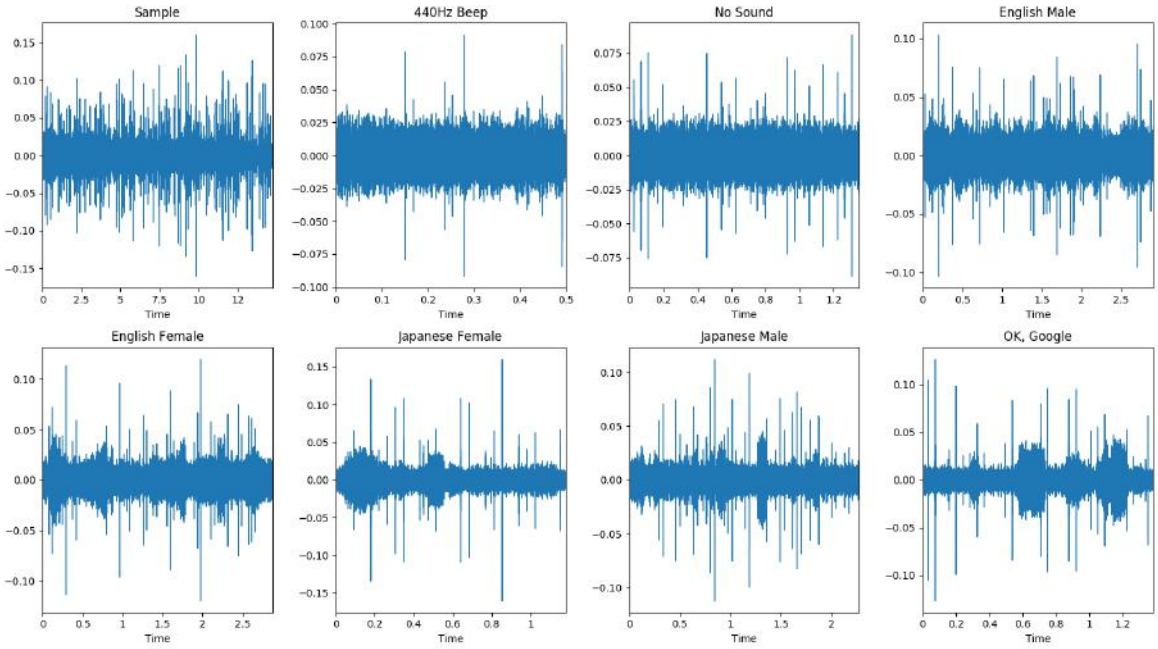

図 63　距離 1272m の波形

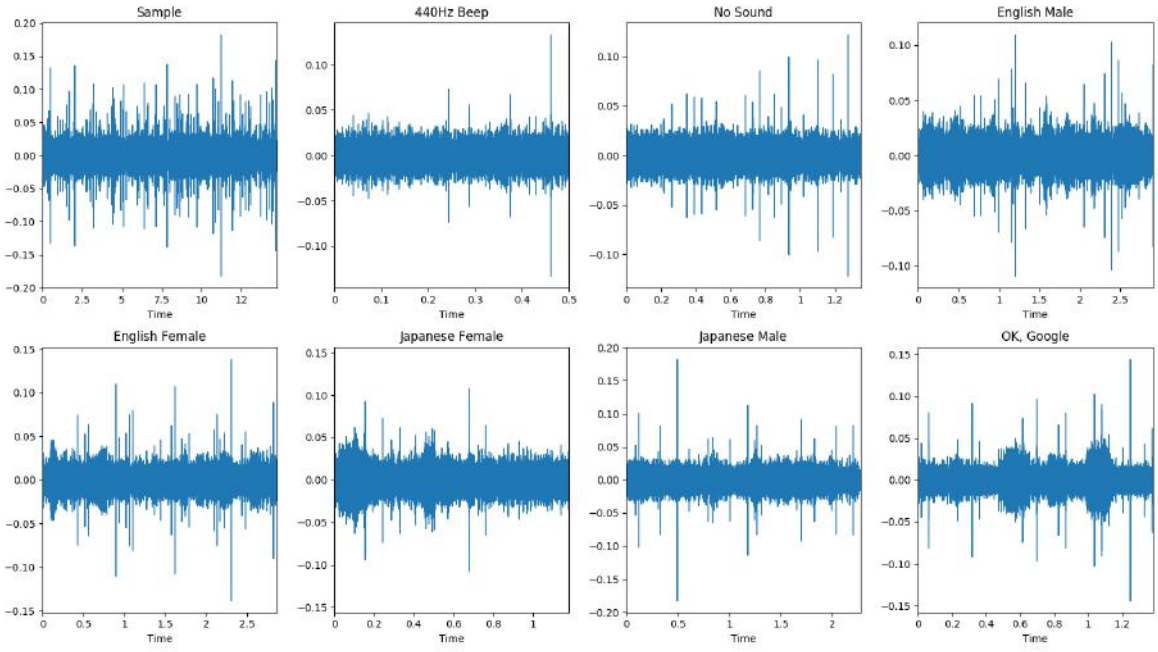

図 64　距離 1470m の波形

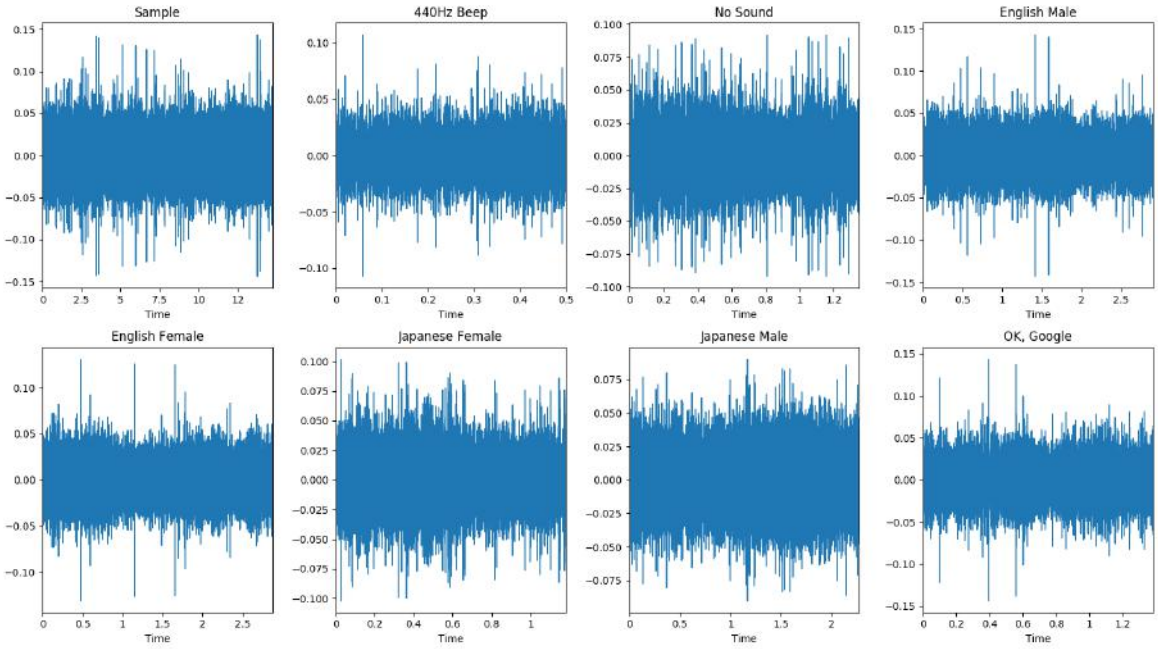

図 65　距離 2219m の波形

パワースペクトル密度（PSD）による分析

音が途切れていないデータの中から、距離ごとに音データのパワースペクトル密度（PSD）を比較する。距離 120m の PSD を図 66、距離 497m の PSD を図 60、距離 806m の PSD を図 61、距離 1003m の PSD を図 62、距離 1272m の PSD を図 63、距離 1470m の PSD を図 64、距離 2219m の PSD を図 65 に示す。なお、各図の No Sound は Soundless のことである。

1272m 以下のパワーについては、どれも多少の差異はあるが大きな変化はない。しかし、1470m はパワーが大きくなっており、2219m はさらに大きくなっている。

周波数帯でみると、約 2000Hz～4000Hz 間のパワーは距離が離れるに従って大きくなっている。1272m の英語と日本語の男声のパワーについては、約 2000Hz 以下と約 2000Hz～4000Hz の各ピーク値の差がかなり小さい、もしくは約 2000Hz～4000Hz の方がパワーは上回っている。1470m になると Soundless 以外の音データのパワーは、1272m よりも各ピーク値の差が小さくなる。特に人間の音声（440Hz のビープ音以外）が顕著である。2219m になると、人間の音声（440Hz のビープ音以外）はどれも Soundless に非常に近いパワーになる。440Hz のビープ音も約 2000～4000Hz のパワーは大きくなる。

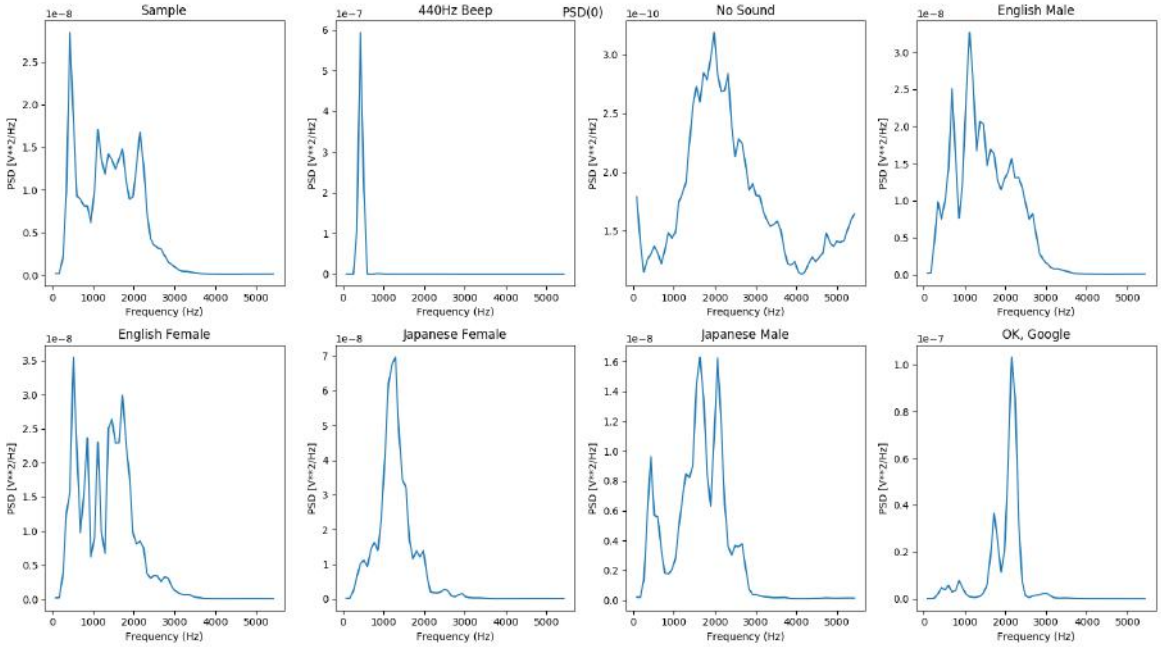

図 66　距離 120m のパワースペクトル密度

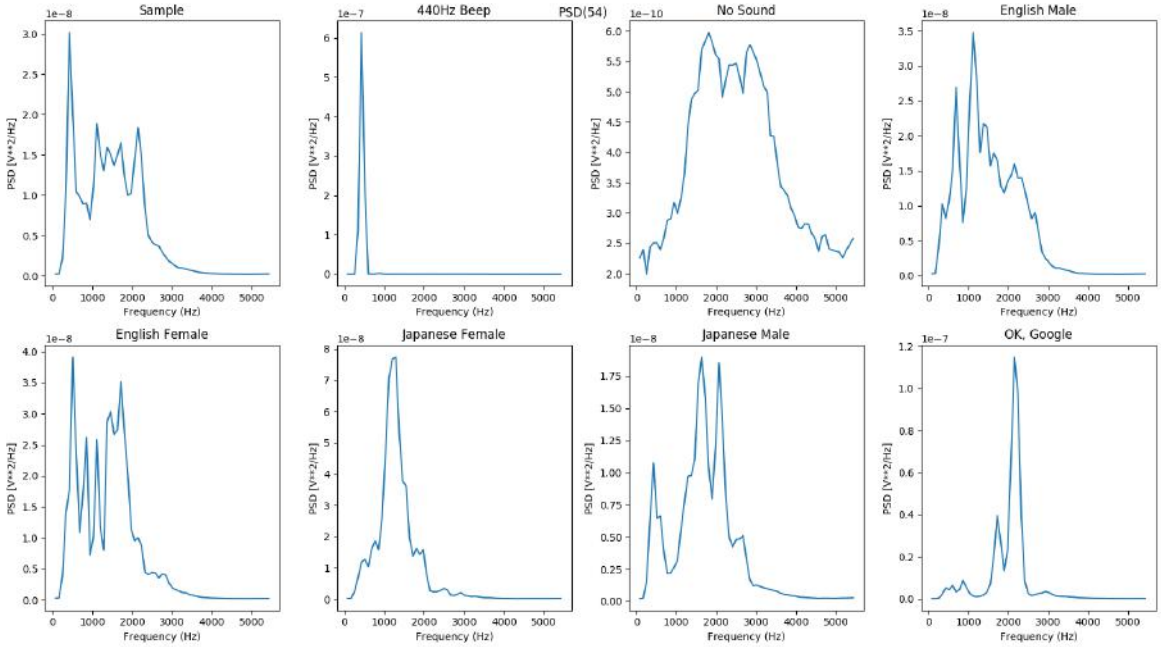

図 67　距離 497m のパワースペクトル密度

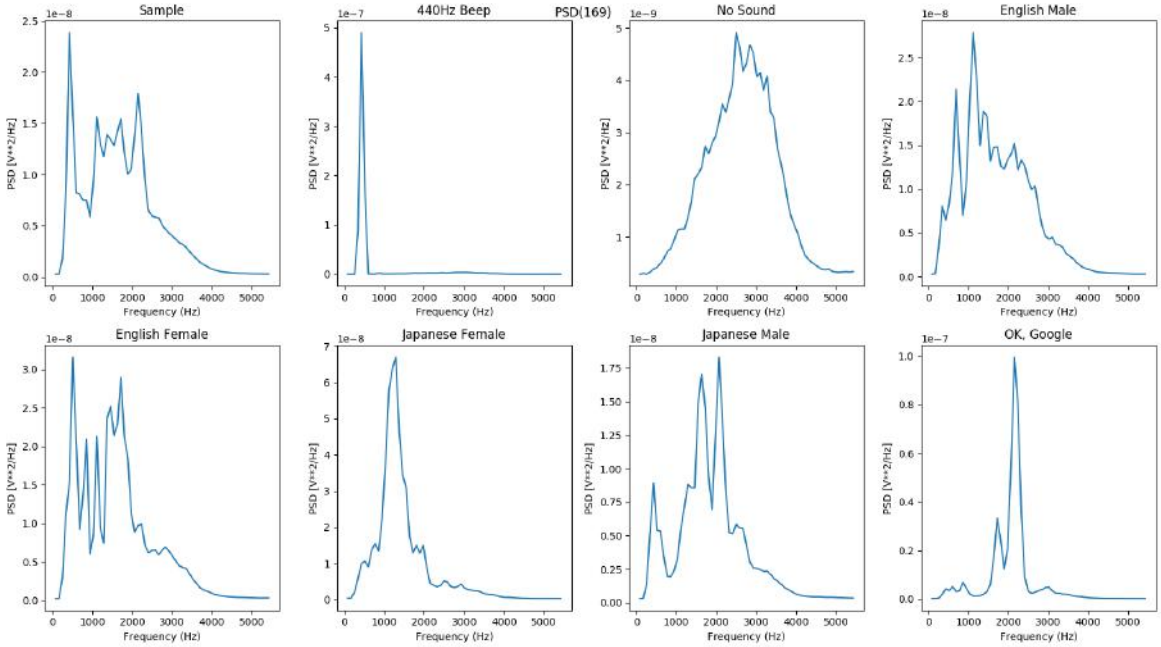

図 68　距離 806m のパワースペクトル密度

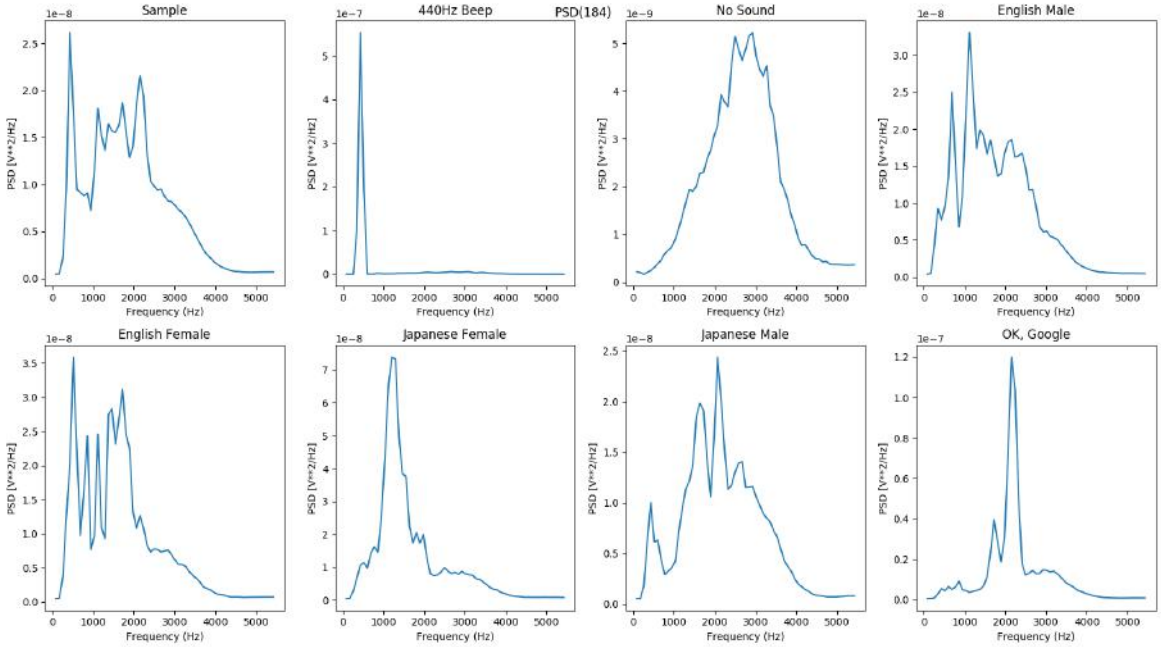

図 69　距離 1003m のパワースペクトル密度

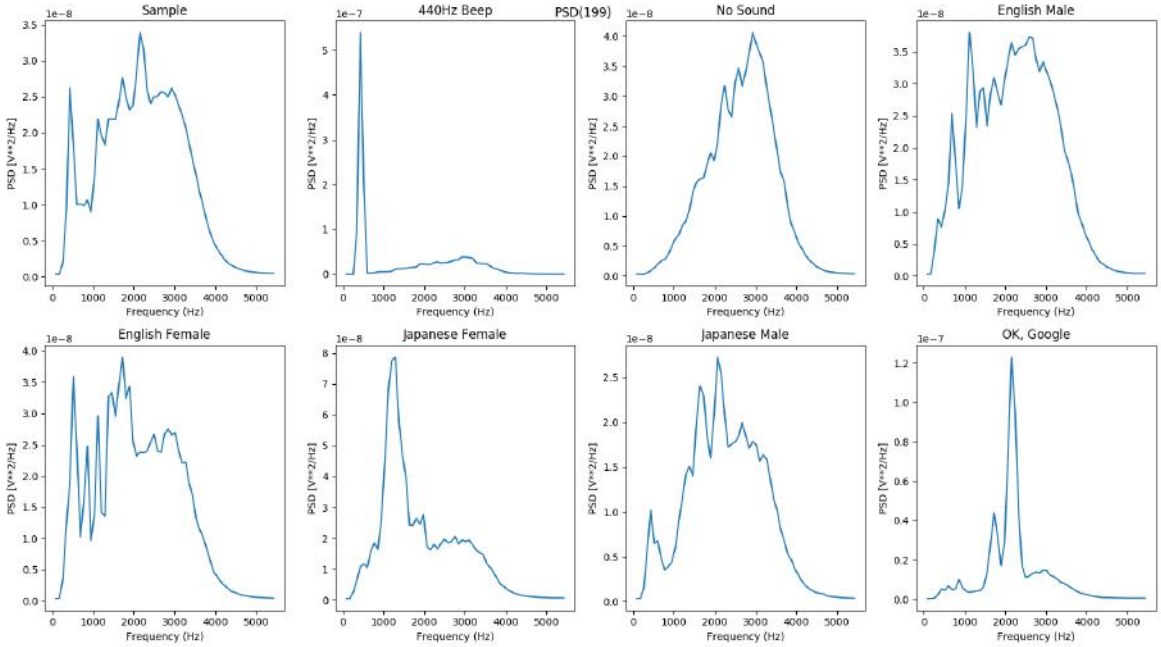

図 70　距離 1272m のパワースペクトル密度

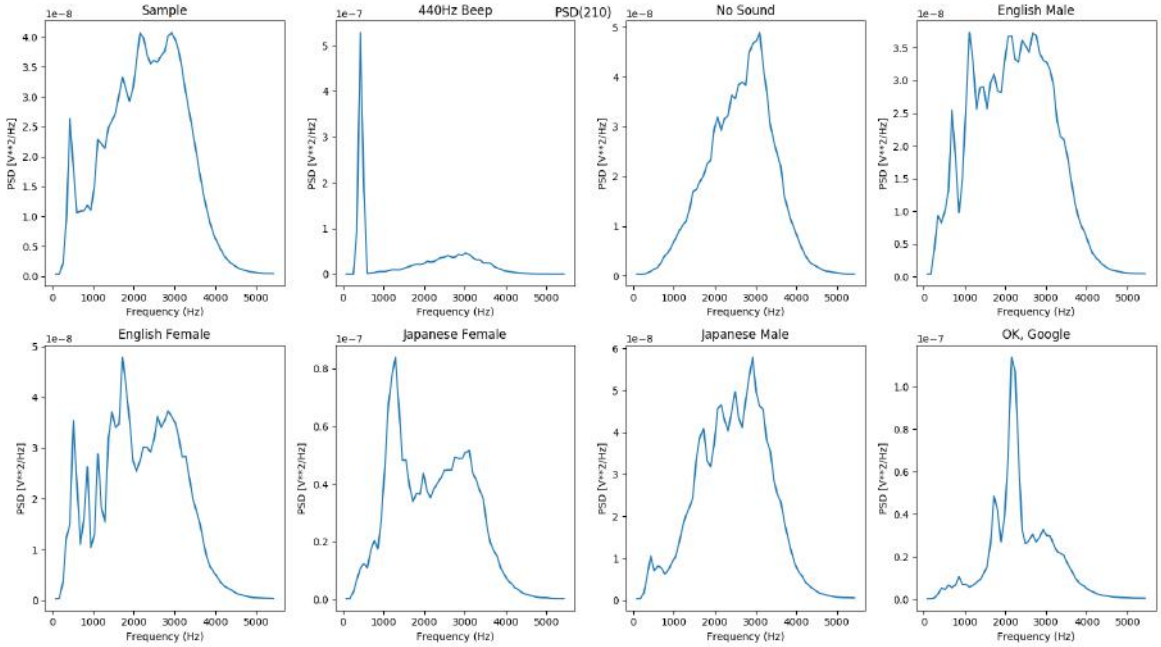

図 71　距離 1470m のパワースペクトル密度

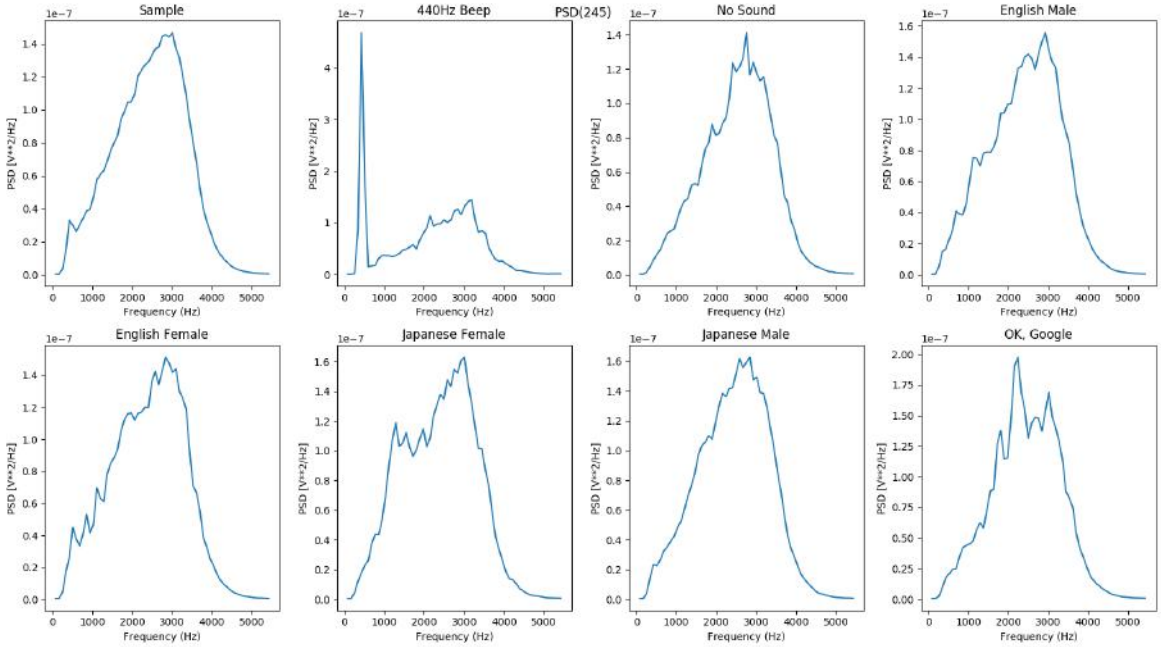

図 72　距離 2219m のパワースペクトル密度

スペクトログラム（STFT）による分析

音が途切れていないデータの中から、距離ごとに音データのパワースペクトル密度（PSD）を比較する。距離 120m の PSD 図 73 を、距離 497m の PSD を図 74、距離 806m の PSD を図 75、距離 1003m の PSD を図 76、距離 1272m の PSD を図 77、距離 1470m の PSD を図 78、距離 2219m の PSD を図 79 に示す。なお、各図の No Sound は Soundless のことである。

全ての音データにおいて 256Hz～4096Hz にパワーがあり、距離が遠くなるにつれて、特定の範囲のパワーが大きくなる傾向が見られる。497m と 806m の間で 2048Hz～4097Hz のパワーが増加し、1003m で 1024Hz～4097Hz、それ以降は 512Hz～4097Hz のパワーが増加していく。Soundless についても距離が離れるに従って、2048Hz～4097Hz から徐々に全体的にパワーが強くなっていく。この強くなっていくパワーが音声ノイズであると考えられる。

440Hz のビープ音は距離に関係なく 440Hz に 10dB 以上のパワースペクトルがあるのが各図から見て取れる。また、人間の音声についても距離に関係なく声紋があり、その声紋のパワースペクトルについても距離に応じた変化は各図から見られない。しかし、ビープ音と人間の音声は、距離が離れるにつれて 440Hz のスペクトル・声紋と音声ノイズのパワーの差が小さくなっていく。このことから、距離が離れるにしたがって音声ノイズのパワースペクトルが強くなり、音声のパワースペクトル（声紋など）と音声ノイズのパワースペクトルの差は小さくなっていく。

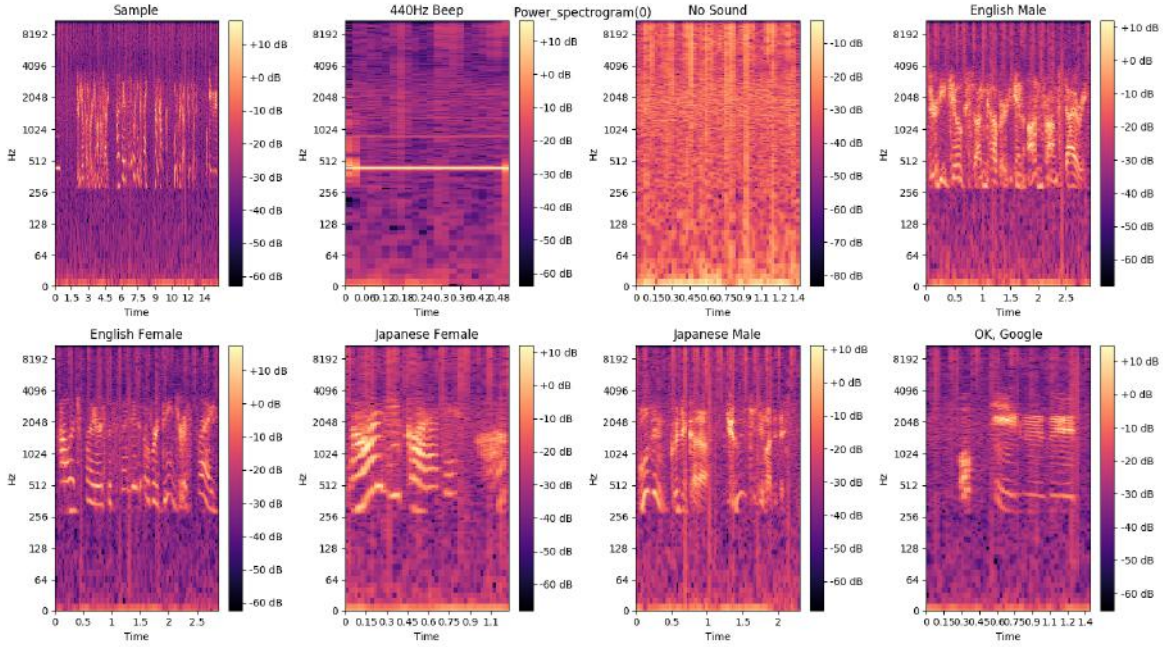

図 73　距離 120m のスペクトログラム

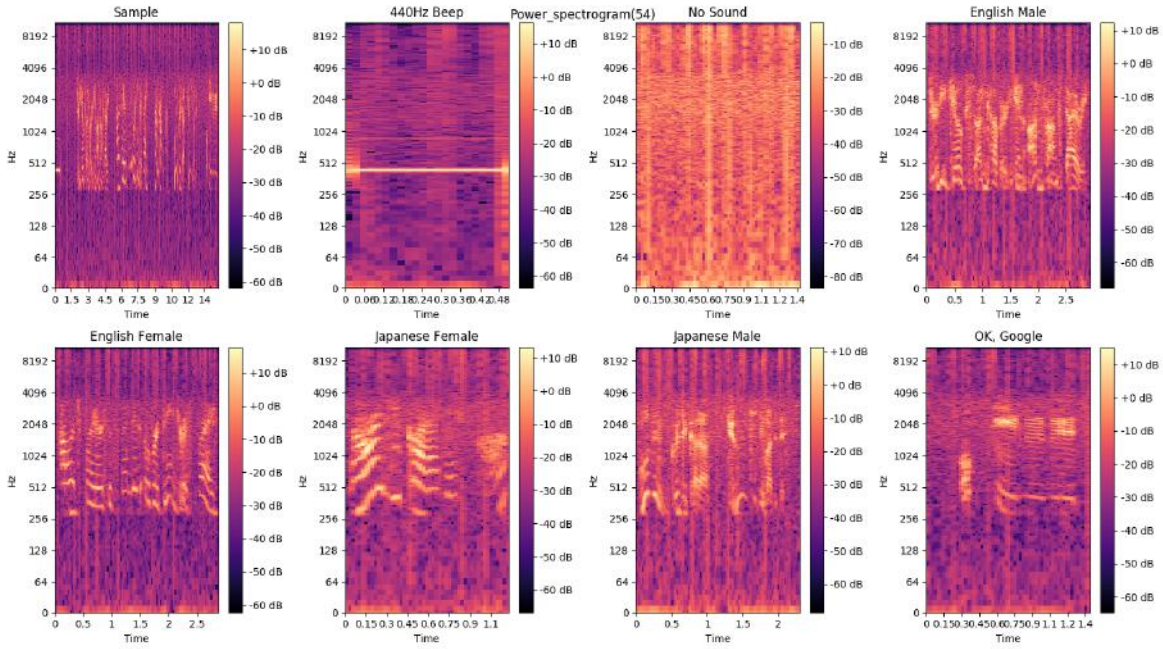

図 74　距離 497m のスペクトログラム

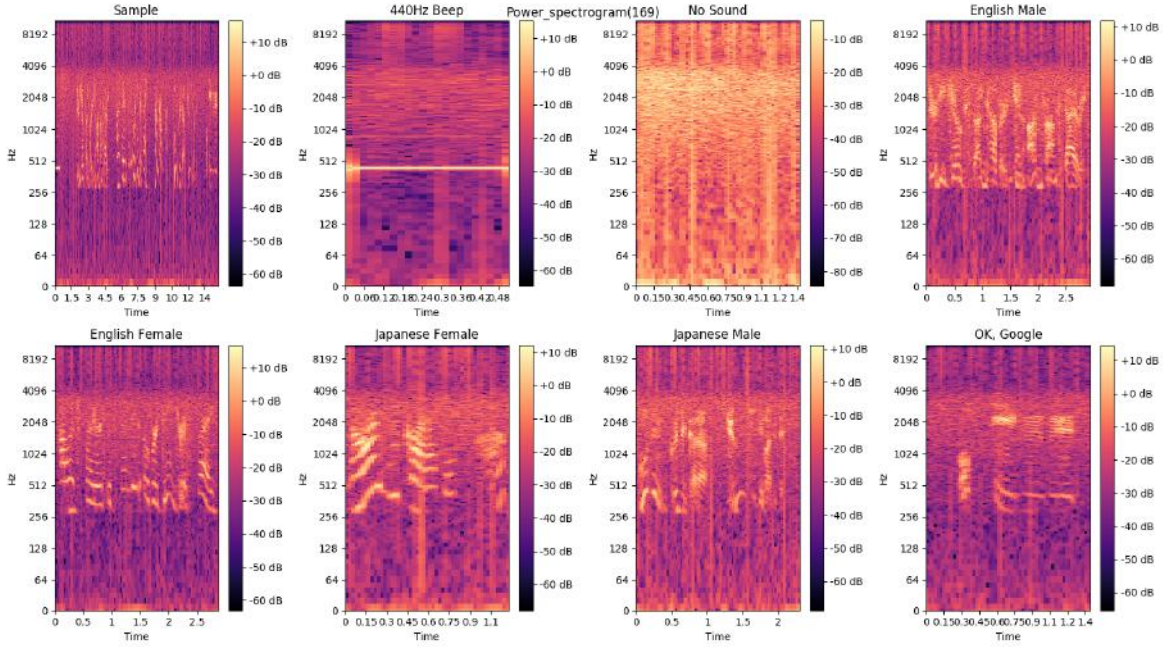

図 75　距離 806m のスペクトログラム

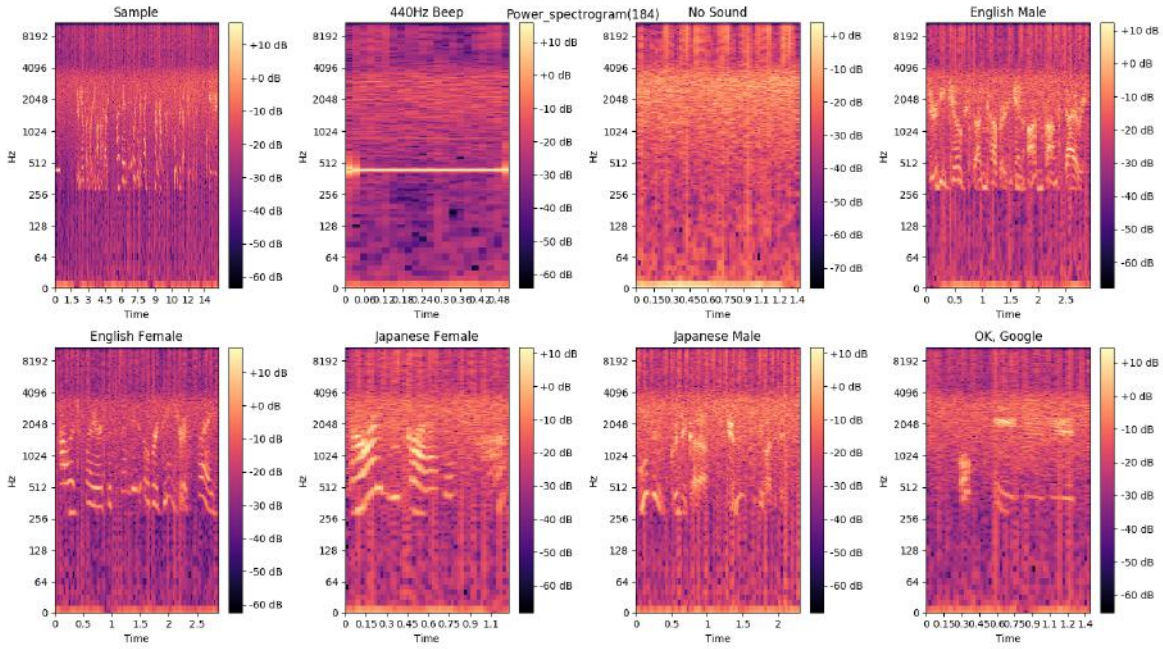

図 76　距離 1003m のスペクトログラム

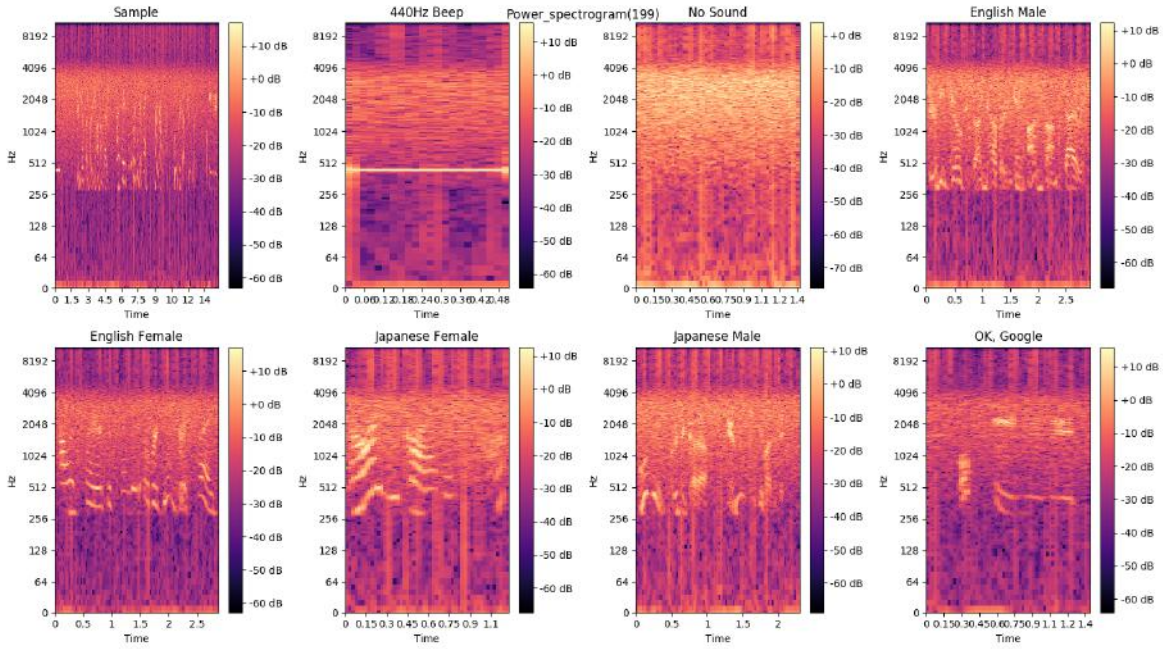

図 77　距離 1272m のスペクトログラム

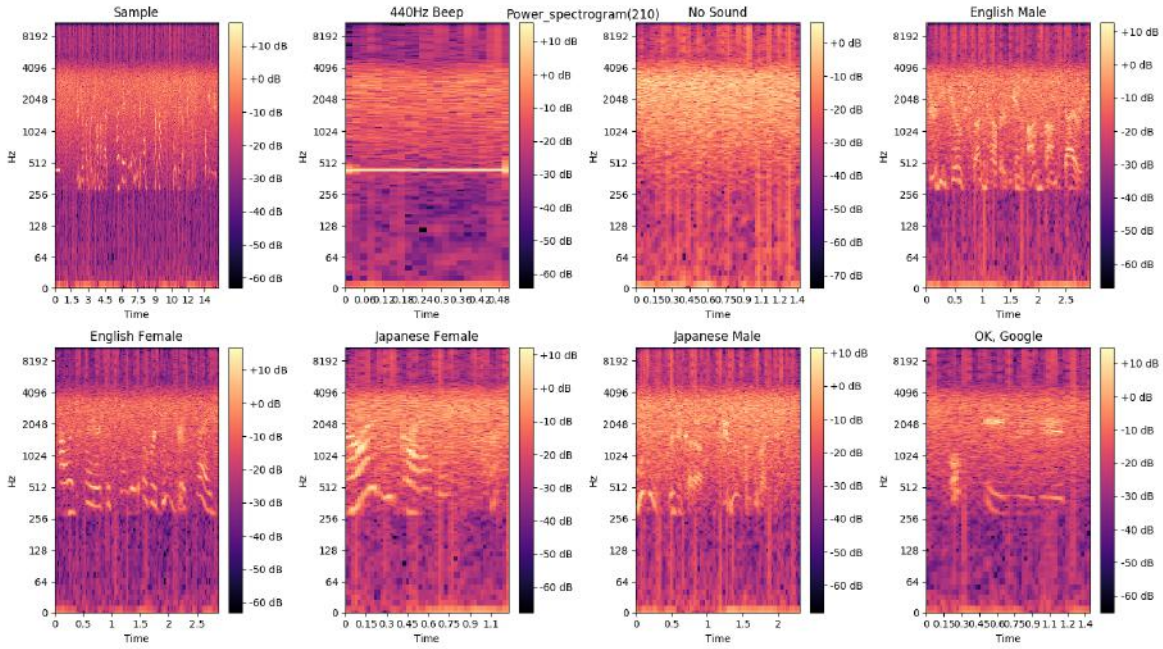

図 78　距離 1470m のスペクトログラム

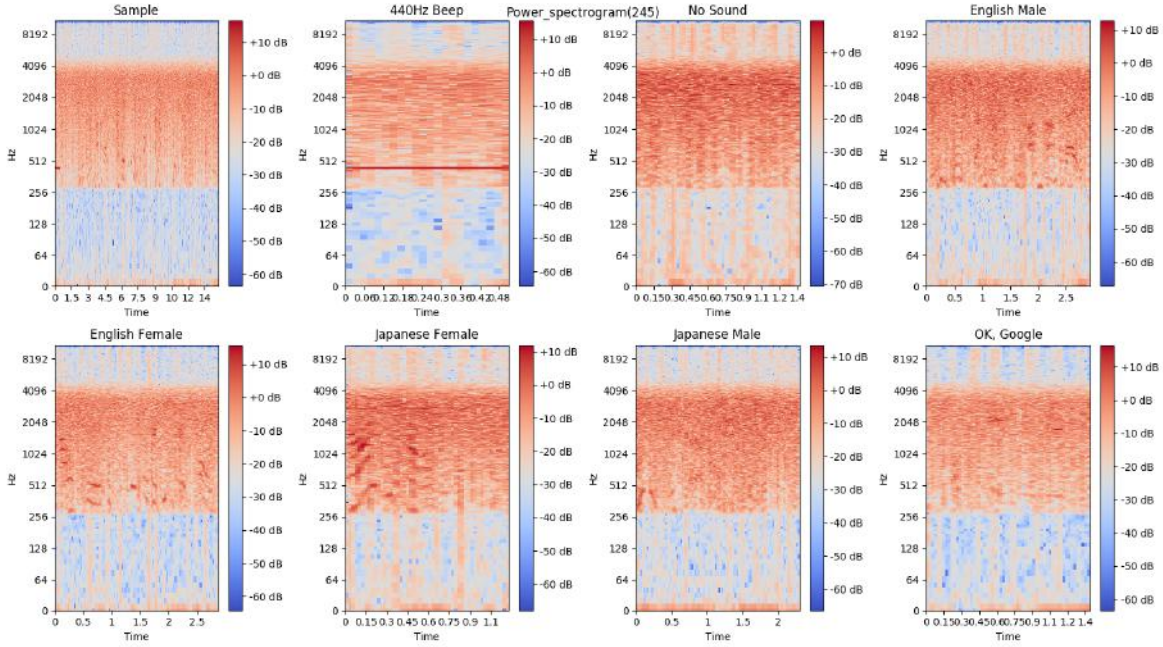

図 79　距離 2219m のスペクトログラム

5.3.6　6 月 9 日の実験のまとめ

この 6 月 9 日の実験では以下の特徴が見られた。

1. Soundless の音量の実効値は、距離が離れるに従って音量が大きくなる。
2. Soundless 以外の音量の実効値は約 1km を超えたあたりから、距離が離れるに従って音量が大きくなり、音の信号は音声ノイズの振幅に被されていく。
3. 各音データ全体のスペクトルで見たとき、約 2kHz～4kHz のパワーは距離が離れるにしたがって大きくなる。
4. 各音データ全体のスペクトルで見たとき、約 1.5km～2km になると、人間の音声のスペクトルは Soundless のスペクトルに近づく。
5. スペクトログラムで見たとき、音データは 256Hz～4096Hz の間にパワーが集中している。
6. スペクトルグラムで見たとき、距離が近いときは高周波数帯（2048Hz～4097Hz）で音声ノイズのパワーが強くなり、1km を超えたあたりから距離が遠くなるにつれて、徐々に高周波数帯から低周波数帯の間（512Hz～4097Hz）で音声ノイズのパワーが強くなる。
7. 距離が離れるにしたがって音声ノイズのパワースペクトルが強くなっていき、音声データのパワースペクトルとの差は小さくなる。

まず、スケルチを OFF にしたので 2km を超えても音データを受信することができたが、電波を受信しないときがあり、電波が不安定であった。これは飛行機の影響だったのかは不明なため、検証する必要がある。

特定小電力トランシーバーの郊外の通信目安距離（約 1km～2km）の間では、Soundless だけでなく音の信号にたいしても音声ノイズが乗ることが分かった。約 1km までは音声ノイズは 2kHz～4kHz にかけてパワーが強くなり、そこから距離が離れるにしたがって徐々に低周波数

帯まで音声ノイズのパワーが強くなる傾向が見られた。このことから、通信目安距離までの音声ノイズとその距離以上での音声ノイズには、何かしらの違いや特徴があると考えられる。

5.5　音声データの分析結果

6 月 1 日、9 日の実験結果から、2 つの実験に共通する特徴をまとめる。

1. 音声や音がない受信音声データ（Soundless）は距離に応じて音量が大きくなる。
2. 通信目安距離以上においては、音声などの受信音声データについても距離に応じて音量が大きくなり、音声や音がない受信音声データ（Soundless）の波形に近づいていく。
3. 受信音声データのパワーは 256Hz～4kHz 付近の間に集中している。
4. 距離が離れるにしたがって、約 4kHz から低い周波数に向かって音声ノイズのパワースペクトルが強くなり、最終的に約 256Hz から約 4kHz 間の音声ノイズのパワースペクトルが強くなる。
5. 距離が離れるにしたがって、音声と音声ノイズのパワースペクトルの差は小さくなる。

また、440Hz のビープ音は音声ノイズとのピークの差が、人間の音声とは違って顕著に表れた。また、ビープ音を人間が聞いたときに、距離が離れても音声ノイズに埋もれることなく聞き取れた。これはビープ音が単純な sin 波の音で、人間の音声のように複雑な波でないために、音声ノイズの波と違いによるものではないかと思われる。

第6章　大量の音声データを使った音声ノイズと距離の分析

第 5 章で音声ノイズと送受信間距離に相関があることが確認できた。そこで、より多くの人間の音声でデータをとることにした。第 5 章の実験では人間の音声は 5 種類しかなかったため、これで音声ノイズによる距離推定モデルをつくるにはデータ不足である点と、人によって音量や声の高さに個人差があるため、こういった個人差の影響を受けないようなモデルを構築するために、大量の音声データを使って人間の声に対しての音声ノイズと送受信間距離の相関関係を調べることにした。

6.1　大量の音声データについて

大量の音声データを用意するのに、多くの実験協力者を募ってデータを作成するのは非常に時間がかかってしまうため、「Common Voice [48]」のデータを使用することにした。

Common Voice は Mozilla が進める、一般の人が利用できるオープンな音声データベースである。Common Voice の音声データベースは、世界中の人が読み上げた文章の音声データが大量にあり、英語であれば 1522575000 人（2018 年 7 月 22 日現在）のスピーカーが収録した音声データを、Common Voice の Web サイトからダウンロードすることができる。Common Voice プロジェクトは、Deep Learning による音声認識技術の開発などを行う際に、大企業だけしか使えなかった大量の音声データを、誰でも利用できるようにするプロジェクトである。Common Voice の音声データベースは、このプロジェクトによって集められた音声データである。著作権については CC0（いかなる権利も保有しない）に設定されているため、すべての人が著作権による制限を受けることなく、自由に追加や再利用などをすることができる。また、このプロジェクトに参加して音声データベースに登録することも自由にでき、Common Voice の Web サイトなどで文章を読むだけで、音声データを登録することができる。

また、Mozilla が進めている DeepSpeech プロジェクト [49]では、この音声データセットを使って、機械学習をするためのソースコードを GitHub 上に公開している。このソースコートは TensorFlow [50]で動かすことができ、誰でも音声認識の研究・開発をすることができるようになっている。

Mozilla と同じように音声認識のために大量の音声データセットを作成して、機械学習用の学習データを作っているものはいくつかあり、LibriSpeech や TED-LIUM コーパス、VoxForge、Tatoeba などがある。

Common Voice から音声データセットをダウンロードすると、コーパスがいくつかに分割されている。分割されたコーパスの名前と内容について、表 19 に示す。また、ダウンロードしたデータセットには各コーパスに含まれる音声の文章と、音声と文章が一致しているに投票した人数、不一致に投票した人数、スピーカーの年齢、性別、アクセント（米国英語や英国英語、オーストラリア英語など）が、コーパスごとに CSV でまとめられている。

今回の実験では Common Voice の「cv-valid-dev」にある音声データのみを使用した。データ数としては 4076 個の音声データがあり、すべて再生するには 3 時間ほど時間がかかる。

表 19　Common Voice のコーパスの名前と内容

cv-invalid	少なくとも 2 人以上が聞いたときに文章と音声が「不一致」であるサブセット。
cv-valid-dev	少なくとも 2 人以上が聞いたときに文章と音声が「一致」しており、音声認識の「開発と実験」用に分割したサブセット。
cv-valid-train	少なくとも 2 人以上が聞いたときに文章と音声が「一致」しており、音声認識の「トレーニング」用に分割したサブセット。
cv-valid-test	少なくとも 2 人以上が聞いたときに文章と音声が「一致」しており、音声認識の「テスト」用に分割したサブセット。
cv-other-dev	文章と音声の一致について投票した人数が 2 人未満、もしくは「一致」と「不一致」が同じ票数で、音声認識の「トレーニング」用に分割したサブセット。
cv-other -train	文章と音声の一致について投票した人数が 2 人未満、もしくは「一致」と「不一致」が同じ票数で、音声認識の「開発と実験」用に分割したサブセット。
cv-other -test	文章と音声の一致について投票した人数が 2 人未満、もしくは「一致」と「不一致」が同じ票数で、音声認識の「トレーニング」用に分割したサブセット。

6.2　実験方法

第 5 章で行った実験の反省を踏まえて、次のように実験を行った。なお、基本的に第 5 章の 6 月 9 日の実験方法をベースにした。

1. 特定小電力トランシーバーUBZ-LM20 を 2 台用意し、受信を行うトランシーバーはスケルチを OFF にする。今回の実験ではチャネルを 8ch に設定する。送信を行うトランシーバーは 1 度送信ボタンを押すと、3 分間電波を発信し続けるモードに設定する。
2. 第 5 章の実験と同様に、送信を行うトランシーバーには音楽再生デバイス、受信を行うトランシーバーにはボイスレコーダーを接続し、録音を開始する。
3. 受信を行うトランシーバー、ボイスレコーダーをタッパー内に固定し、そのタッパーを三脚に固定する。そして、三脚を見通しの良い岸壁に設置する。
4. 送信を行うトランシーバーは AWS-1 のデッキ内に固定し、音楽再生デバイスで音声データを再生し続ける。

5. 実験者がトランシーバーの送信ボタンを押したときに、その時刻を同時に記録する。

前回の実験の反省として、UBZ-LM20 は送信開始から 2 分 50 秒経過すると、「ピッ」と送信終了 10 秒前の合図をする音が流れる。今回は純粋に人間の音声のみで分析用データを作成したいので、送信開始時刻を記録しておき、その送信開始時刻から 2 分 40 秒間の録音データのみを抽出するプログラムを作成した。これによって、人間の音声のみで分析データを生成することができる。

距離に関しては前回同様、AWS-1 のログデータを使用する。上記のように送信開始時刻から 2 分 40 秒ごとにデータを抽出するため、ログデータについても送信開始時刻から 2 分 40 秒間のデータのみを抽出して、距離を算出している。

実際に音声ノイズから距離を算出することを想定した時、リアルタイム性があるほうが実用的である。そこで、分析する音声データはパワースペクトル（PSD）をスペクトログラムのように時間単位で生成したデータに加工した（以降、パワースペクトログラムと呼ぶ)。前回とのスペクトログラムの違いは、パワーの単位が dB から PSD[V^2/Hz]になる点である。

また、実際にトランシーバーで話すときは、話者によって音量などが違うことから、cv-valid-dev の音声はノーマライズなどの音声処理はしなかった。そのため、スピーカーごとに音量や環境音ノイズに差がある。

なお、この実験で作成したプログラムのソースコードを付録 C に載せる。

6.3 データ収集の場所と環境

実験は 2018 年 6 月 23 日と 24 日の 2 日間行った。受信を行うトランシーバーの設置位置を表 20、23 日の天気を表 21、24 日の天気を表 22 に示す。また、受信を行うトランシーバーの設置の様子を図 80、送信を行うトランシーバーの設置の様子を図 81

表 20 受信を行うトランシーバーの設置位置

	6 月 23 日	6 月 24 日
緯度	35.6328616524447	35.63299479897658
経度	139.925418198239	139.92510245194651
高度	1.9986038208007812	2.6601700624567854

表 21　江戸川臨海（新木場）の天気（6 月 23 日）

6 月 23 日	降水量（mm）	気温（℃）	風速・風向（m/s）		日照時間（分）
			風速	風向	
11	0	23	0.9	南東	0
12	0	22.4	2.7	南東	0
13	0	22.4	3.3	東南東	0
14	0	21.5	4.4	南南西	0
15	0.5	21	2.3	南南西	0
16	3	19.9	3.3	南南東	0

（出典）気象庁「過去の気象データ検索」http://www.data.jma.go.jp/obd/stats/etrn/index.php

表 22　江戸川臨海（新木場）の天気（6 月 24 日）

6 月 24 日	降水量（mm）	気温（℃）	風速・風向（m/s）		日照時間（分）
			風速	風向	
10	1.5	20.6	2.7	北北西	0
11	0	21.3	1.6	北北西	0
12	0	23	1.8	西北西	0
13	0	24.1	1.3	北北東	0.3
14	0	24.5	4.7	南	0.8

（出典）気象庁「過去の気象データ検索」http://www.data.jma.go.jp/obd/stats/etrn/index.php

図 80　受信を行うトランシーバーの設置の様子（6 月 23 日）

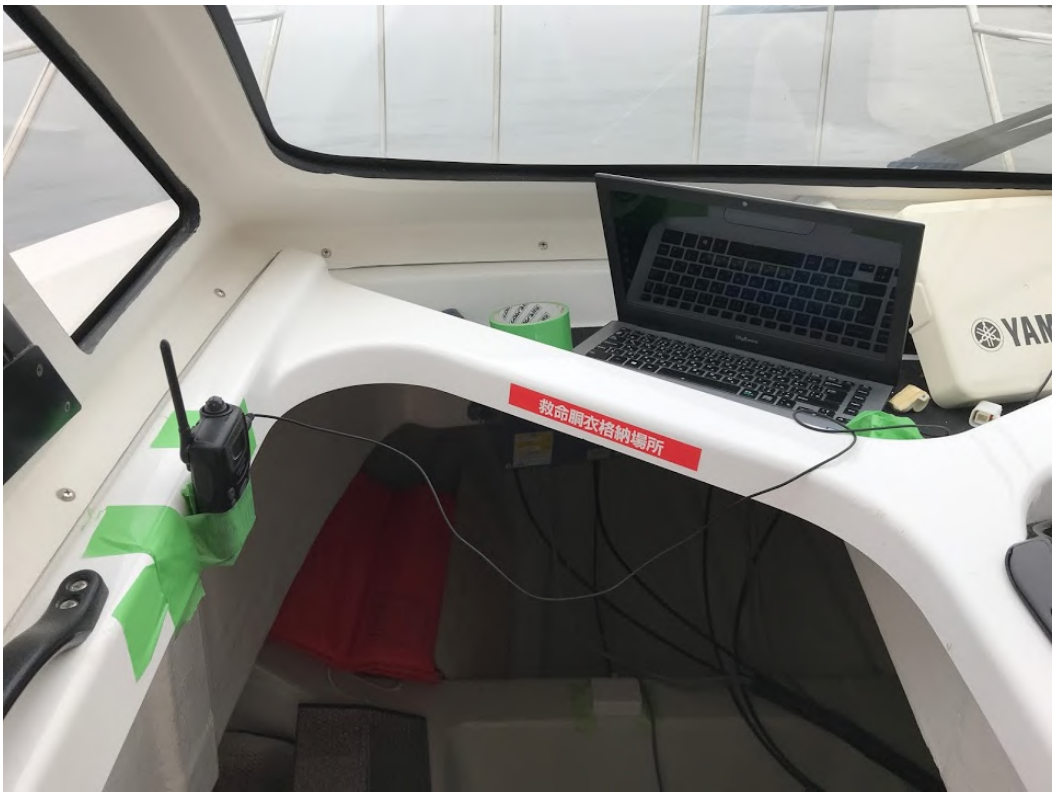

図 81　送信を行うトランシーバーの設置の様子（6 月 23 日）

天気については 2 日間とも風はそこまで強くなく、AWS-1 の動揺は小さかった。ただし、途中で小雨が降ったりしたため、多少電波が減衰している可能性が考えられる。

分析用の音声データのデータ量としては、23 日は送信回数 44 回、計 1 時間 57 分 20 秒の分析用データが得られた。24 日は送信回数 39 回、計 1 時間 44 分 0 秒の分析用データが得られた。よって、2 日間合わせて 3 時間 41 分 20 秒の分析用の音声データが得られたことになる。

6.4　実験結果

23 日、24 日の 2 日間で行った実験で得られた分析用の音声データをパワースペクトログラムに変換した。それをパワースペクトル（PSD）と距離の関係を 3 次元グラフに表示したものを図 82 に示す。なお、周波数は常用対数をとっている。

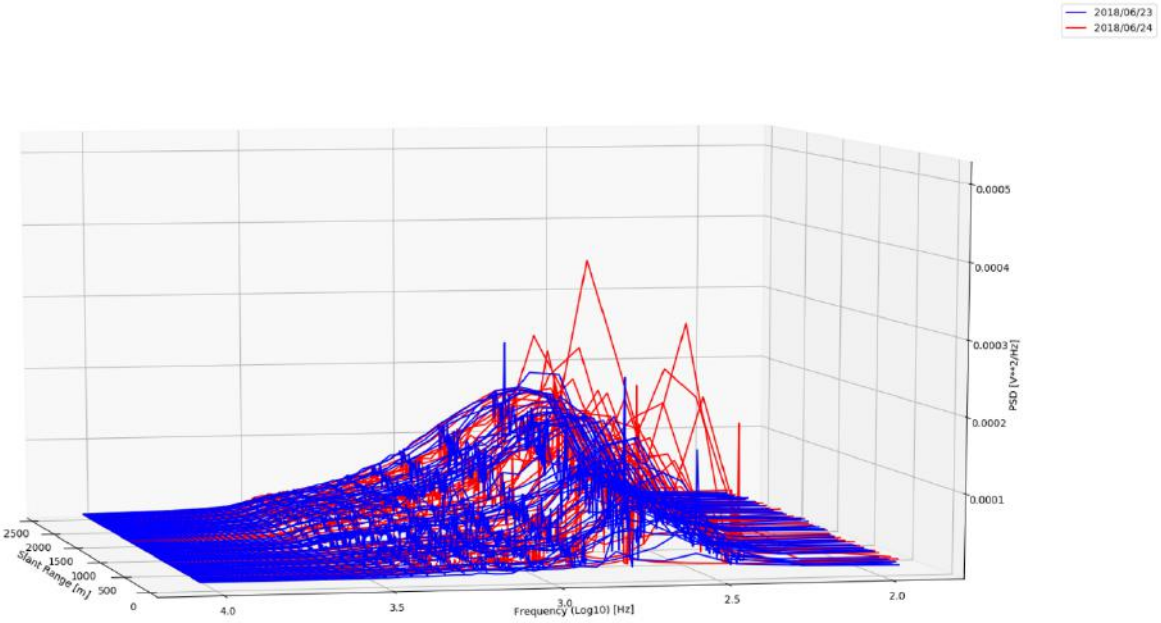

図 82　パワースペクトル（PSD）と距離（6 月 23 日・24 日）

23 日、24 日ともに距離が離れるに応じて、約 300Hz～4kHz の PSD が強くなっている。PSD が最も強い周波数は、どの距離においても約 300Hz～500Hz 付近がピークであり、約 1000m を超えたあたりから、急激に約 300Hz～500Hz をピークとして約 300Hz～1kHz の範囲で大きな山型になっている。24 日については PSD が急激に強くなる部分がある。約 1kHz～4kHz の部分は距離が離れるに応じて、緩やかに PSD が強くなっている。

距離ごとにおける周波数と PSD の関係を図 83 に示す。このデータは 23 日、24 日を合わせたデータであり、各グラフに書いてある距離の前後 5m における周波数ごとの PSD 平均しており、録音された音声によってピーク値が変わるため、なるべく同じくらいのピーク値である 90m、500、990m、1500m、2010m、2430m のデータを並べて表示した。

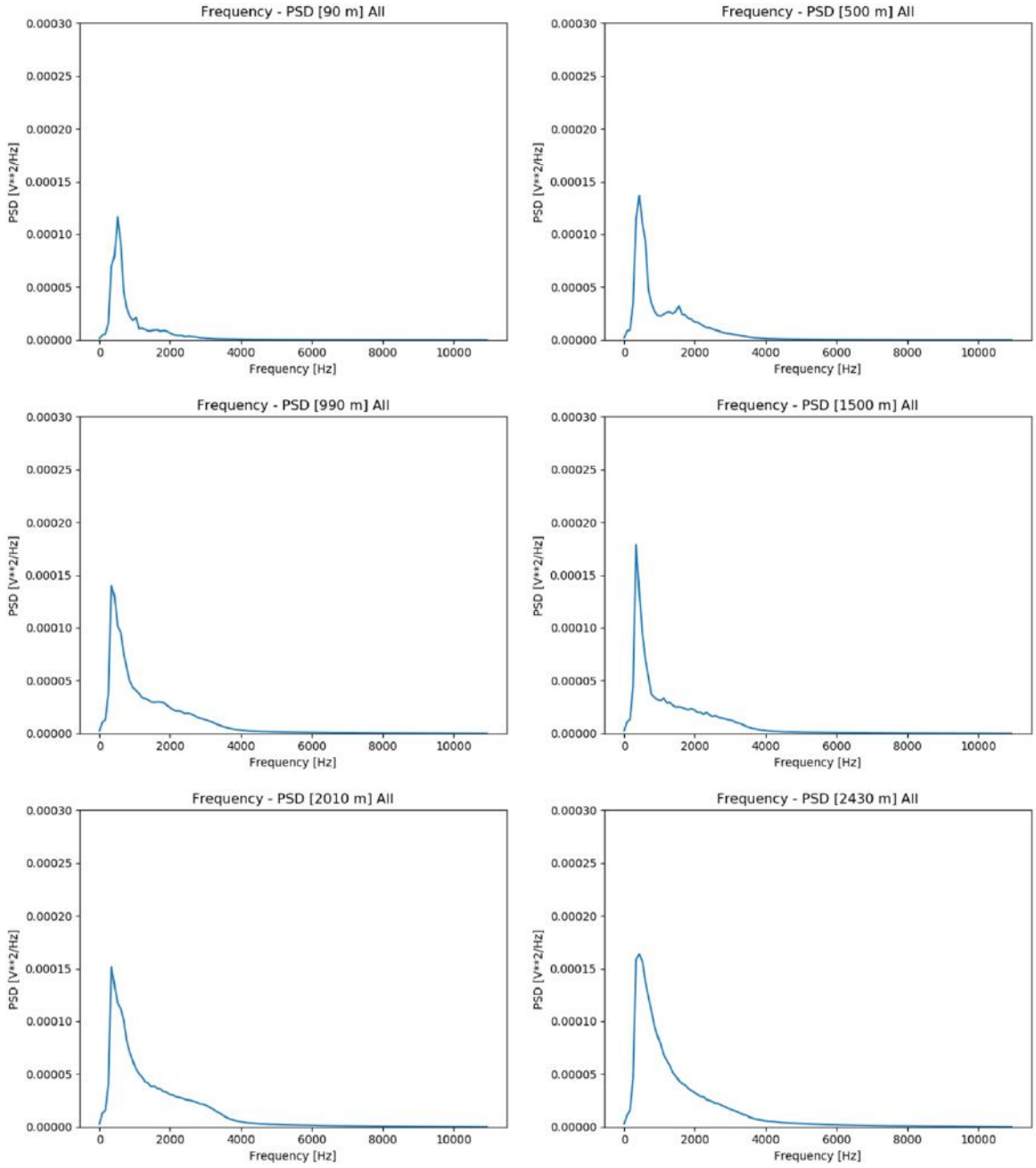

図 83　距離ごとにおける周波数と PSD

この図だけでなく、指定した距離の前後のグラフも比較した結果、90m から 990m にかけて約 2kHz をピークに、約 2kHz～4kHz の範囲で PSD が徐々に大きくなっている傾向がある。約 500Hz～1kHz の間のピーク値が 90m と 1500m 以上で比べると 1500m 以上のほうが大きい。また、2010m 以上になると 1kHz 付近（PSD ピーク値の周波数より少し上）の PSD が強くなっている傾向が見られた。

ただし、図 84 のように 10m の差で PSD が大きく変動する例もいくつか見られた。これは再生した音声データをノーマライズしていないことや、再生する音声データの切り替え時や音節部分に無音時間があるために、近い距離の PSD に差が生じることがあると考えられる。

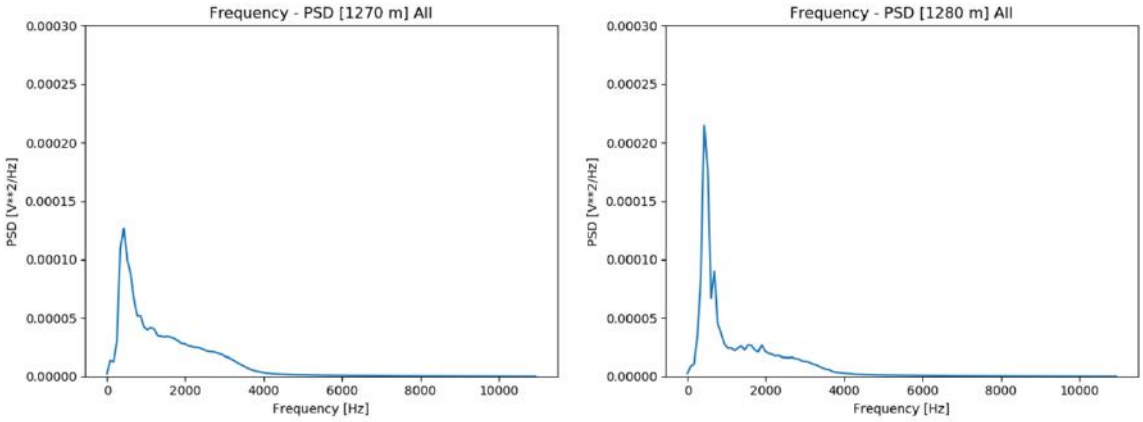

図 84　PSD が大きく変動する例

話者が話している時と、そうでない時で PSD には差があるが、特定小電力トランシーバーの通信目安距離を考慮すると、全体的に 1km 以下は約 2kHz～4kHz で PSD が強くなることが確かめられた。また、1km 以上になると約 500Hz～1kHz の PSD も強くなることが分かった。これは第 5 章の結果とほぼ一致する形でないかと思われる。

第7章　多層パーセプトロンを用いた音声ノイズによる距離推定モデルの作成

第 6 章では人間の音声の個人差があっても、音声のパワースペクトルと距離には相関があることが確かめられた。しかし、距離推定モデルを作成するために回帰問題を解いて近似曲線を作成するにも、距離ごとに計算すべき周波数帯が異なるため、パワースペクトルを与えただけで距離を推定するには、計算が複雑になると思われる。そこで、距離推定のモデルを作成するにあたって、複雑な回帰問題を解くのに優れていて、近年注目されているニューラルネットワークライブラリ・フレームワークを用いて多層パーセプトロンを構築し、音声ノイズと距離の関係を学習させることにより距離推定モデルを生成する。

7.1　Keras について

近年は、DNN をはじめとするディープラーニングについてプログラムするライブラリが多数作られており、このライブラリを使うことで誰でも、複雑なプログラムを書くことなくディープラーニングをすることができる。

中でも注目を最も集めているニューラルネットワークライブラリ・フレームワークが TensorFlow [50]である。TensorFlow は Google が開発をしているライブラリで Python や C、JavaScript などで動作しており、環境構築が容易で GPU にも対応している。2.3.1 で紹介したような Google の Duplex は、TensorFlow によって RNN（再帰型ニューラルネットワーク）を構築して学習・運用している。世界中の多くのユーザーが TensorFlow を使っていることや、チュートリアル、リファレンスが充実しているのも魅了である。

Keras [51]は Python で書かれた、TensorFlow または CNTK [52]、Theano [53]上で実行可能な高水準のニューラルネットワークライブラリである。コンセプトとしてユーザーフレンドリー、モジュール性、及び拡張性を掲げており、他のライブラリに比べて学習コストをかけることなく、手軽にディープラーニングを行うことができる。さらに柔軟性もあって生産性が高く、ユーザーはより多くのアイデアを試すことができる。TensorFlow と組み合わせて使われることが多く、GPU に対応した TensorFlow であれば、Keras も GPU に対応することができるので、より高速な学習をすることができる。Keras は事業や研究で非常によく使われており、ユーザー数は 20 万人以上の個人ユーザー（2017 年 11 月現在）がいる。また、最近は TensorFlow のソースコードの中で、Keras のライブラリを使うことができるようになっている。

Keras の基本的な使い方は、次の 4 ステップで使うことができる。モデルの構築や最適化アルゴリズム、損失関数、評価関数などは、すでにライブラリで用意されており、そのライブラリを呼び出すだけで複雑な式を書くことなく、実行できるようになっている。また、学習率を変える、自作の損失関数、評価関数などを作ることも容易である。

1. 学習データを用意する。
2. モデル（レイヤー、ノードや活性化関数）を構築する。
3. 最適化アルゴリズムと損失関数、評価関数を指定する。
4. バッチサイズや学習回数を指定して、学習をさせる。

7.2　実行環境・設定

この実験ではでは第 6 章で取得した距離とパワースペクトログラムのデータを使い、Keras で多層パーセプトロンによるトレーニングを行うことで、距離推定モデルの作成、学習、評価を行った。まず、パワースペクトログラムの各周波数の PSD を入力値とし、その時の距離を教師データとした。本研究は距離を推定したいので、DNN は回帰分析を行うように設計した。

Keras はマウスコンピュータの NEXTGEAR-NOTE i400 シリーズ 9 台を使用し、Keras・TensorFlow-GPU をインストールした。なお、同じシリーズでスペックが異なる 2 種類のコンピュータを使用しているが、どちらにも NVIDIA® GeForce® GT650M（2GB）の GPU を搭載しており、GPU を使ったディープラーニングが可能である。コンピュータのスペックを表 23 に示す。また、トレーニングの様子を図 85 に示す。

表 23　実験で使用したコンピュータのスペック表

シリーズ名	NEXTGEAR-NOTE i400SA8-W7	NEXTGEAR-NOTE i400PA8-SP-W7
製造元	株式会社マウスコンピュータ	
CPU	Intel® Core™ i7-3640QM CPU @ 2.40GHz 2.40GHz	Intel® Core™ i7-3840QM CPU @ 2.80GHz 2.80GHz
メモリ（RAM）	4.0 GB	16.0GB
GPU	NVIDIA® GeForce® GT650（2GB）	
システムの種類	64 ビット	
OS	Windows 7 Professional	

図 85　トレーニングの様子

Keras はモデルを作成して、そのモデルを fit 関数でフィッティングすることで、学習が開始される。しかし、パワースペクトログラムと距離のデータは 2GB 以上あり、このやり方では学習用データをまとめて呼び出してメモリを使用するため、メモリが 4GB のコンピュータではメモリ不足に陥ることがあった。そこで、パワースペクトログラムと距離のデータをランダムシャッフルして 10 分割し、9 つのデータセットを学習用データ、1 つのデータセットをテスト用データにした。Keras で学習をするプログラムを書く際は、事前にデータを読み込んで fit 関数を実行するのではなく、毎回ランダムに学習用データセットを呼び出して、train_on_batch 関数を使用して学習させた。

最適化アルゴリズムにはいくつか手法があるが、この実験では最も優れているといわれている Adam を使用した。なお、過学習を防止するために Adam の学習率をe^{-5}と設定している。また、各レイヤーの活性化関数にもいくつか種類があるが、これについても一般に優れているといわれている ReLU を使用した。ただし、この実験における DNN は回帰問題を取り扱っているため、最後のレイヤー（出力）については Linear を使用している。

過学習防止のためにモデル内の各レイヤーの間に、Dropout を設定している。通常はトレーニングごとにすべてのレイヤーのノードが更新されてフィッティングを行うのだが、すべてのノードを更新してしまうと過学習を起こす可能性が高くなる。そこで、Dropout によって毎回トレーニングをするときにノード数を制限し、ランダムに各ノードを少しずつフィッティングさせることで過学習を防止する。この実験では Dropout を 0.5 に設定しており、毎回トレーニングで更新するノードを 50%に制限している。

epoch 数は 1000、batch サイズは 256 に設定した。epoch とはデータセット全体に対する 1 回の処理単位、batch とは *N* の sample（データセットの 1 つの要素）のまとまりで、訓練中は batch の処理結果によりモデルが 1 回更新される。一般に batch は、それぞれの入力のみの場合に比べて、入力データのばらつきをよく近似し、batch が大きいほど、その近似は制度がよくなる。しかし、その分 batch 処理には時間がかかるため、モデルの更新が遅くなる。また、メモリの使用量も多くなる。逆に batch が小さいとモデルの更新は早くなり、メモリ使用量も少なく済むが、入力データのばらつきの影響を大きく受けやすくなる。そのため、一般的にはメモリ領域を超えなくて済む最大の batch サイズを選ぶことが推奨されている。

7.3　モデルの設定と実験結果

モデルの設計、距離の正規化について、損失関数の設定を表 24 に示す。この実験では教師データの値を、そのままの距離の値を使うのではなく、ある定数で割ることで正規化し、0 から 1 の間に収めるようにしている。0.0-1.0 と書かれているのは、距離の最小値が 0.0、最大値が 1.0 となっており、0.0-0.8 と書かれているのは、距離の最小値が 0.0、最大値は 0.8 となっている。距離の最大値が 0.8 というのは、1.0 を最大としたときに、教師データの距離の最大値を 80%に収める格好となっている。

損失関数については Keras の用意している関数をそのまま利用している。mae（mean_absolute_error）は平均絶対誤差、mean_squared_logarithmic_error 平均二乗対数誤差、logcosh は予想誤差のハイパボリックコサインの対数である。Keras の説明では、logcosh は

$log(cosh\,x)$はxが小さければ、$x^2/2$とほぼ等しくなり、xが大きければ$abs(x) - log\,2$とほぼ等しくなる。つまり、平均二乗誤差とほぼ同じ働きをする。さらに、時折ある乱雑な誤った予測にそれほど強く影響されないとされている。

なお、この実験で作成したプログラムのソースコードを付録 D に載せる。

表 24　各モデルの設計・正規化・損失関数

	Model	正規化	Loss Function
1	129（入力）→500→1（出力）	0.0 - 1.0	logcosh
2	129（入力）→500→1（出力）	0.0 - 1.0	mean_squared_logarithmic_error
3	129（入力）→50→1（出力）	0.0 - 1.0	logcosh
4	129（入力）→50→1（出力）	0.0 - 1.0	mae
5	129（入力）→500→1（出力）	0.0 - 0.8	mean_squared_logarithmic_error
6	129（入力）→300→50→1（出力）	0.0 - 0.8	logcosh
7	129（入力）→300→50→300→1（出力）	0.0 - 0.8	logcosh
8	129（入力）→300→50→300→50→1（出力）	0.0 - 0.8	mae

この 1 から 8 のモデルの実験結果を図 86 から図 93 に示す。

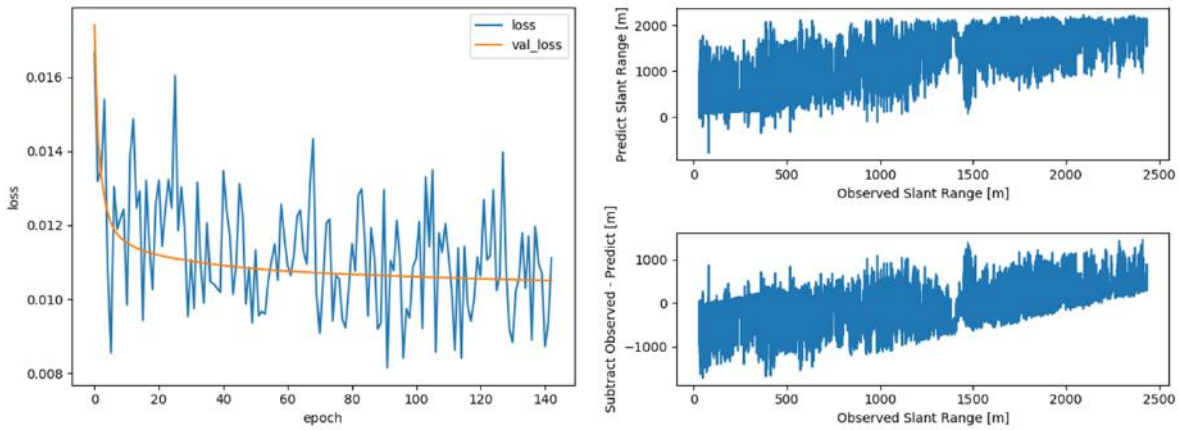

図 86　1 の実験結果

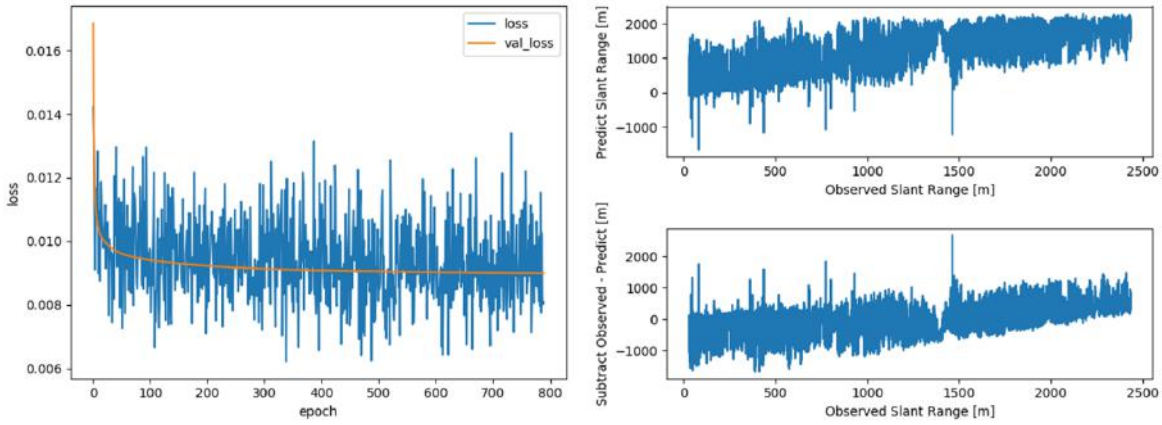

図 87　2 の実験結果

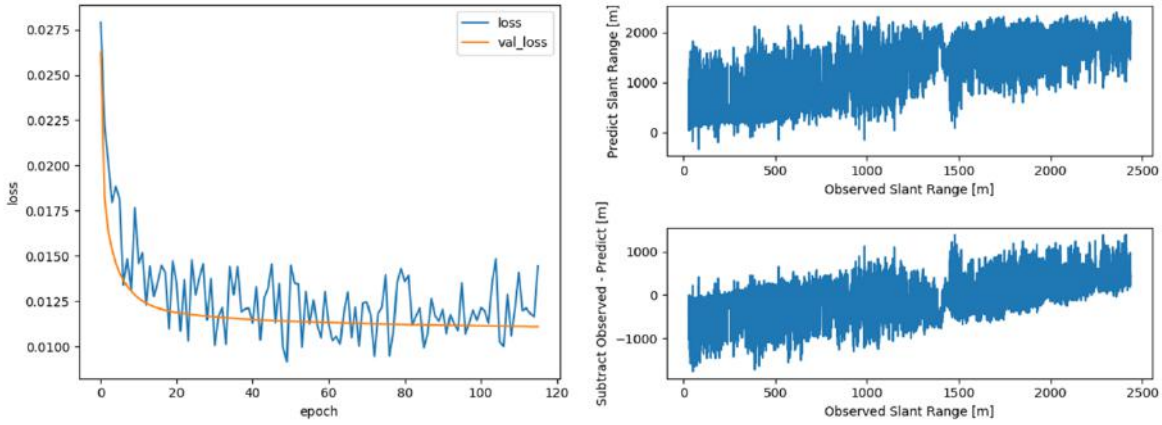

図 88　3 の実験結果

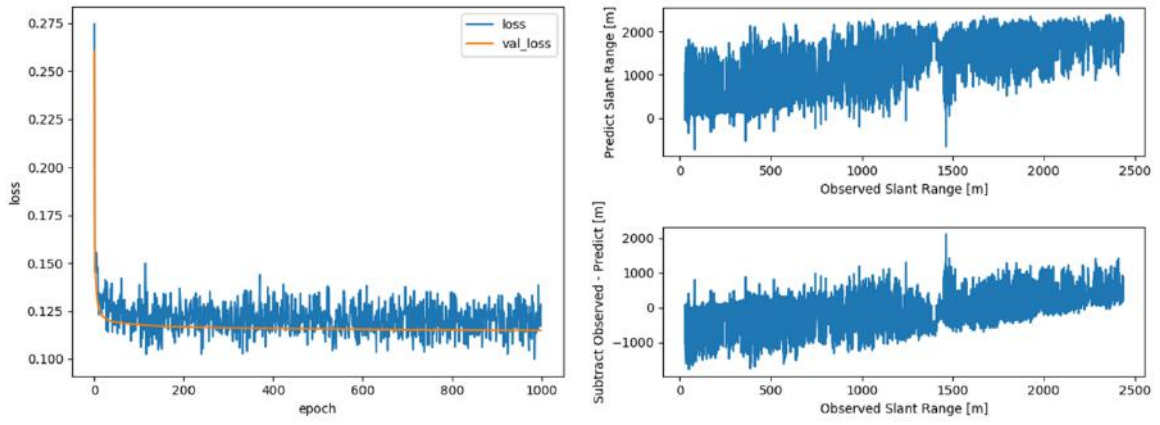

図 89　4 の実験結果

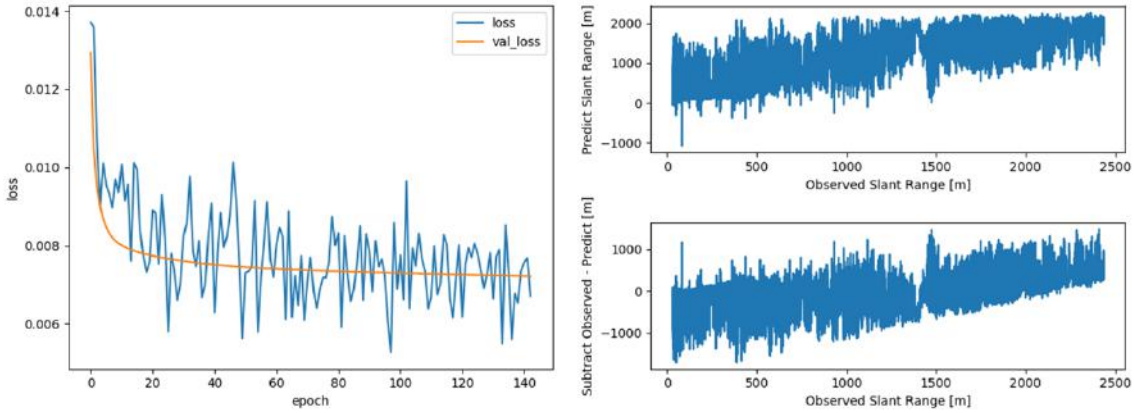

図 90　5 の実験結果

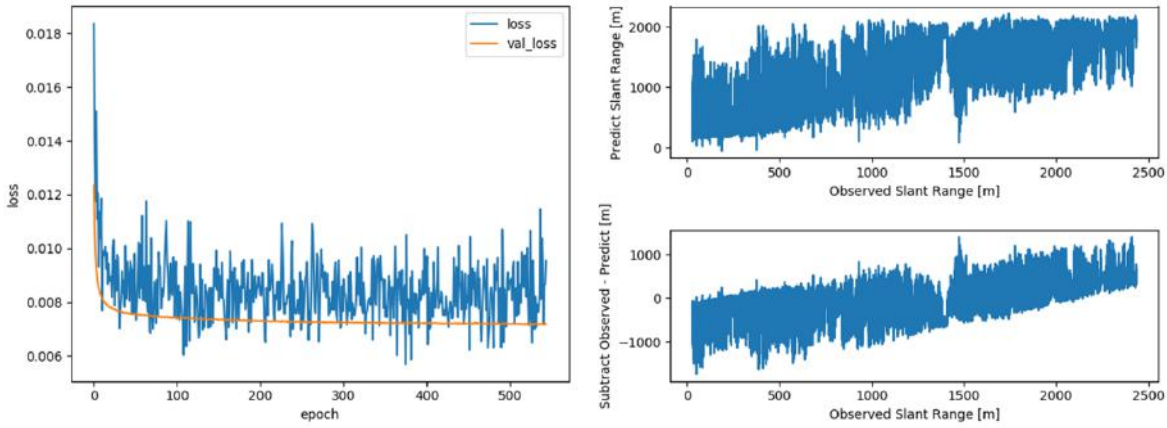

図 91　6 の実験結果

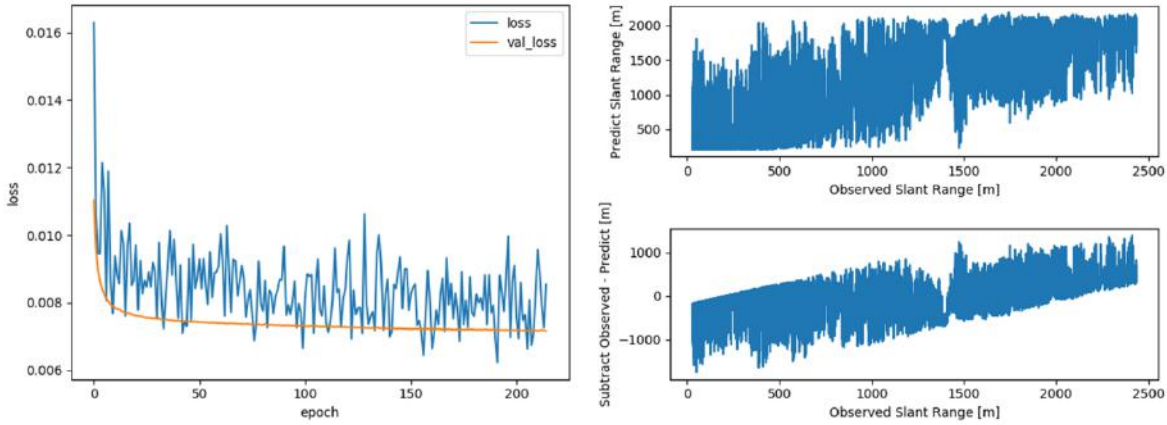

図 92　7 の実験結果

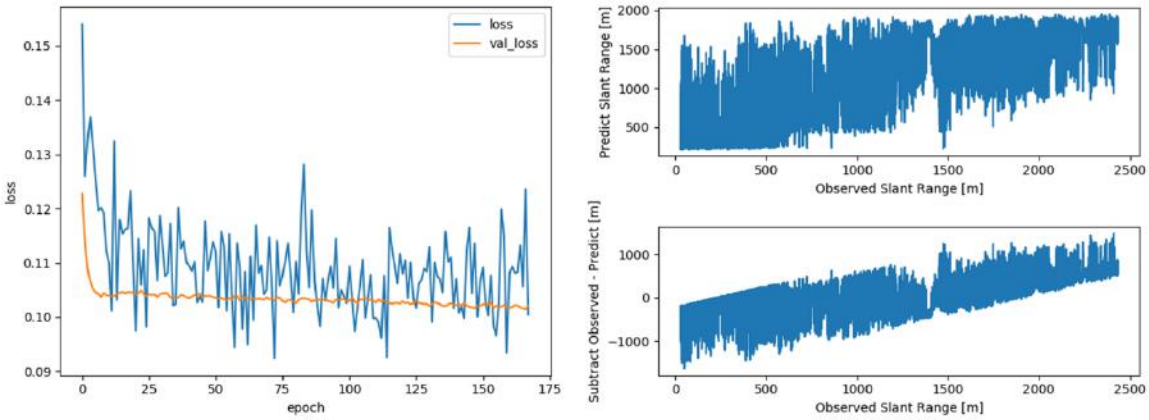

図 93　8 の実験結果

各図には左側に epoch 数に対するロス（青 loss：学習用データのロス、オレンジ val_loss：テスト用データのロス）の減少を表している。また、右側には実際の距離ごとに、テスト用データで距離を予測した結果（右上）と、実際の距離から予測した結果を引いた差を表している。

まず、すべてにおいて学習用データのロスが振動している。つまり、どのモデルも過学習を起こしているといえる。本文には載せていないがプログラムを開発するにあたり、Dropout を導入していない、最適化アルゴリズムの学習率をデフォルト値にしていた時に過学習を起こしていたので、1 から 8 の実験では過学習防止の措置をとっていたのだが、それでも過学習を起こしていた。これは圧倒的に学習用データが少ないことが原因であると考えられる。

また、テスト用データで距離を予測させた結果、かなり大きく予想値にばらつきが見られた。これは過学習を起こしたために、未知の値に対しての予想ができていないと思われる。しかし、実際の距離と予想値に相関がみられるため、PSD によって距離が近いのか遠いのかを学習できていると思われる。

正規化のやり方で比較すると、5 から 8 の距離の最大値を 0.8 にした方が、テスト用データのロスが小さい傾向が見られ、距離の予想値も 0m を下回る結果を出していない。つまり、距離の最大値を 1.0 としないほうが学習しているといえる。

8 の実験ではテスト用データのロスが振動している。これはレイヤー数を多くしすぎたことが原因であるため、トレーニングをする際にはこれよりもレイヤー数を減らす必要がある。

損失関数の違いで比較すると、2 と 5 の実験で使った mean_squared_logarithmic_error は、距離の予想値が 0m を下回っている。また、3 と 4 の実験で使った mae と logcosh では、mae は 1500m 付近の予想値が 0m を下回っている。このことから logcosh が最適な損失関数だと考えられる。

第8章　まとめ

本研究では国際 VHF 通信を AIS やレーダーなどの情報と対応付けることで、海上交通の研究に使用するための国際 VHF 通信のビッグデータや、自動運航船での国際 VHF 通信の発信船舶の特定による避航操船や救助活動に有効であると考えた。そこで、国際 VHF 通信と他の情報を対応付けるシステムを開発するにあたり、まず AIS 情報との対応付けに試みた。

国際 VHF 通信と AIS 情報を対応付ける方法については（1）音声認識で船名などから船舶を特定して AIS 情報を対応付ける方法、（2）指向性アンテナと電波伝搬距離によって位置を特定して、その位置にいる船舶の AIS 情報と対応付ける方法、（3）国際 VHF 通信のノイズから距離を推定して、その距離にいる船舶の AIS 情報と対応付ける方法を提案した。その中でも（3）の方法が最も現実的で有効であると考え、本研究では（3）の方法について研究した。

まず、国際 VHF 通信の音声から距離を推定するモデルを作成するために、携帯型の国際 VHF から東京湾の通信を録音し、その通信内容から船名などを聞き取って AIS 情報と対応付ける試みを行った。しかし、音声を聞き取って通信内容を理解するのにかなり時間がかかったこと、聞き取れた船舶の通信が少なかったこと、VTS については発信位置が不明なことから、このやり方は目的を達成する手段としては不適切と判断した。

次に国際 VHF の代わりに、免許を所持していなくても自由に使用することができる特定小電力トランシーバーを使い、音声のノイズと距離の関係を明らかにすることを試みた。特定小電力トランシーバーは変調方式が国際 VHF と同じ FM であること、電波の性質が類似していることから、特定小電力トランシーバーでモデルを作成して、国際 VHF に応用することができると考えた。特定小電力トランシーバーでノイズと距離の関係を調べた結果、以下のことが分かった。

1. 特定小電力トランシーバーが受信した音声のパワースペクトルは約256Hz～4kHz間に集中している。
2. ノイズの音量は距離が離れるに従って大きくなる。
3. 電波が安定して届く距離（最小の通信目安距離）までは、音声の音量は変わらない。
4. 電波が安定して届く距離（最小の通信目安距離）を超えた場合、距離が離れるに従って音声の音量は大きくなり、ノイズの音量に近づいていく。
5. 距離が離れるに従って、約 4kHz から低い周波数に向かってノイズのパワースペクトルは強くなり、最終的に約 256Hz～4kHz 間のノイズのパワースペクトルが強くなる。

最後に、特定小電力トランシーバーから聞こえる音のパワーと距離の関係を多層パーセプトロンによる機械学習をしてみた。結果としては学習用データ数が少なかったため過学習を起こしてしまったが、作成したモデルはトランシーバーの音のパワースペクトルと距離の関係について学習した傾向が見られた。また、教師データとなる距離を正規化する際は最大距離を 1.0 よりも下にする、損失関数は Keras の場合は logcosh にすると、学習結果が良い傾向であった。以上より、学習用データ数を増やしていくことで、機械学習による距離推定のモデル作成は可能であると考えられる。

今回の研究では、特定小電力トランシーバーの音声解析をすることで距離を推定することを

した。国際 VHF と特定小電力トランシーバーは搬送波の周波数が違うため SN 比に違いはあるが、変調方式が同じであること、電波の性質が似ていることから、国際 VHF においても似たような性質があると予想される。また、機械学習でパワースペクトルと距離には関係があることが確認できたことから、トランシーバーの音声と距離に関係性があることを明らかにした。

今後は国際 VHF と特定小電力トランシーバーの音声の性質の違いを明らかにして、国際 VHF 通信の音声と距離の関係を調べる必要がある。

参考文献

[1] 広. 瀬田, 太. 小野, 雄. 矢野 , 治. 鈴木, “VHF 無線電話通信から見た伊勢湾の海上交通状況,” 日本航海学会, 2009.

[2] 佑. 田崎, 英. 鹿島, 佳. 国枝 , 考. 竹本, “輻輳海域における国際 VHF 無線電話を用いた船舶間コミュニケーションの特徴について,” 日本航海学会, 2015.

[3] 太. 荒谷, 洋. 松倉, 剛. 瀬田 , 兼. 田村, “衛星 AIS データの特性及び利用法に関する検討,” 日本船舶海洋工学会, 2015.

[4] Marine Traffic, “Marine Traffic,” [オンライン]. Available: https://www.marinetraffic.com/. [アクセス日: 10 7 2018].

[5] 東洋信号通信社, “航行履歴データ販売,” [オンライン]. Available: https://www.toyoshingo.co.jp/service/information/data_archive.html. [アクセス日: 10 7 2018].

[6] 瀧本忠教, “海上交通流シミュレーションを用いた安全対策の評価例,” 海上技術安全研究所, 2012 年.

[7] Global Fishing Watch, “Global Fishing Watch,” [オンライン]. Available: http://globalfishingwatch.org/. [アクセス日: 11 7 2018].

[8] 庄司るり, “先端ナビゲートシステムと船舶情報の活用について,” 航法システム研究会, 2016.

[9] 国土交通省, “先進船舶導入等計画認定制度について,” [オンライン]. Available: http://www.mlit.go.jp/maritime/maritime_tk7_000022.html. [アクセス日: 21 6 2018].

[10] 国土交通省海事局, “海事生産性革命（i-Shipping）の推進,” 6 11 2017. [オンライン]. Available: https://www.nmri.go.jp/_src/4227/17kouen_1.pdf. [アクセス日: 21 6 2018].

[11] DARPA, “ACTUV “Sea Hunter” Prototype Transitions to Office of Naval Research for Further Development,” 30 1 2018. [オンライン]. Available: https://www.darpa.mil/news-events/2018-01-30a. [アクセス日: 21 6 2018].

[12] Kongsberg, “Autonomous ship project, key facts about YARA Birkeland,” [オンライン]. Available: https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/4B8113B707A50A4FC125811D00407045?OpenDocument. [アクセス日: 21 6 2018].

[13] Kongsberg, “Wilhelmsen and KONGSBERG establish world's first autonomous shipping company,” [オンライン]. Available: https://www.km.kongsberg.com/ks/web/nokbg0238.nsf/AllWeb/0CBA6362C268ED02C1258264002EB6C4?OpenDocument. [アクセス日: 21 6 2018].

[14] MUNIN, “MUNIN,” [オンライン]. Available: http://www.unmanned-ship.org/munin/. [アクセス日: 21 6 2018].

[15] AAWA, “AAWA Seminar – Helsinki, Finland,” [オンライン]. Available: http://www.rolls-royce.com/~/media/Files/R/Rolls-Royce/documents/customers/marine/ship-intel/12%20-%20AAWA%20Coordinator.pdf. [アクセス日: 1 7 2018].

[16] Rolls Royce, “Ship intelligence,” [オンライン]. Available: https://www.rolls-royce.com/products-and-services/marine/ship-intelligence. [アクセス日: 1 7 2018].

[17] 人民網日本語版, “500 トン級小型無人貨物船の開発がスタート,” 7 12 2017. [オンライン]. Available: http://j.people.com.cn/n3/2017/1207/c95952-9301803.html. [アクセス日: 21 6 2018].

[18] AFP, “中国、世界最大の無人船試験場建設に着手 アジア初,” 14 2 2018. [オンライン]. Available: http://www.afpbb.com/articles/-/3162333?cx_part=search. [アクセス日: 30 6 2018].

[19] SAILDRONE, “SAILDRONE,” [オンライン]. Available: https://www.saildrone.com/. [アクセス日: 1 7 2018].

[20] NavalDrones, “MAST,” [オンライン]. Available: http://www.navaldrones.com/MAST.html. [アクセス日: 1 7 2018].

[21] 第十管区海上保安本部, “自律型海洋観測装置（AOV）によるハイテクな海洋観測を開始,” 3 8 2016. [オンライン]. Available: http://www1.kaiho.mlit.go.jp/KAN10/soudan/kisya/20160803.pdf. [アクセス日: 1 7 2018].

[22] Maritime Robotics, “Wave Glider (R),” [オンライン]. Available: https://maritimerobotics.com/wave_glider/. [アクセス日: 1 7 2018].

[23] CNET Japan, “歩行者に意思を伝える自動運転車、新興企業 Drive.ai が実現へ,” 5 9 2016. [オンライン]. Available: https://japan.cnet.com/article/35088533/. [アクセス日: 10 7 2018].

[24] 総務省, “船舶の安全航行のための海上無線通信の現状と課題,” [オンライン]. Available: http://www.soumu.go.jp/main_content/000020944.pdf. [アクセス日: 5 7 2018].

[25] 株式会社ワイズギア, “国際 VHF の機能紹介,” [オンライン]. Available: https://www.ysgear.co.jp/marine/transceiver/vhf/. [アクセス日: 5 7 2018].

[26] 総務省, “国際 VHF の運用方法,” [オンライン]. Available: http://www.soumu.go.jp/main_content/000230458.pdf. [アクセス日: 5 7 2018].

[27] Forbes, “Optimizing For Voice Search Is More Important Than Ever,” 27 11 2017. [オンライン]. Available: https://www.forbes.com/sites/forbesagencycouncil/2017/11/27/optimizing-for-voice-search-is-more-important-than-ever/#281083044a7b. [アクセス日: 10 7 2018].

[28] Google, “Google Duplex: An AI System for Accomplishing Real-World Tasks Over the Phone,” [オンライン]. Available: https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html. [アクセス日: 10 7 2018].

[29] Google, “TFX,” [オンライン]. Available: https://www.tensorflow.org/tfx/. [アクセス日: 10 7 2018].

[30] Google, “Cloud Speech API,” [オンライン]. Available: https://cloud.google.com/speech/. [アクセス日: 10 7 2018].

[31] Microsoft, “Bing Speech API,” [オンライン]. Available: https://azure.microsoft.com/ja-jp/services/cognitive-services/speech/. [アクセス日: 10 7 2018].

[32] IBM, “Speech to Text,” [オンライン]. Available: https://www.ibm.com/watson/jp-ja/developercloud/speech-to-text.html. [アクセス日: 10 7 2018].

[33] Amazon, “Amazon Transcribe,” [オンライン]. Available: https://aws.amazon.com/jp/transcribe/. [アクセス日: 10 7 2018].

[34] Julius development team, “Julius,” [オンライン]. Available: http://julius.osdn.jp/. [アクセス日: 10 7 2018].

[35] Google, “SpeechRecognizer,” [オンライン]. Available: https://developer.android.com/reference/android/speech/SpeechRecognizer. [アクセス日: 10 7 2018].

[36] IBS Japan, “初歩の電波（無線と電波について）,” 2 3 2010. [オンライン]. Available: http://www.ibsjapan.co.jp/tech/details/elementary-electric-wave/eewave-04-13.html. [アクセス日: 12 7 2018].

[37] 隼. 今津 , 純. 樋野, 新版　電波航法, 株式会社成山堂書店, 2012.

[38] CIRCUIT DESIGN, INC., “Hight pattern calculation,” [オンライン]. Available: http://www.cdt21.com/resources/siryo7.asp. [アクセス日: 13 7 2018].

[39] 潤 . 高田 , “電波伝搬の基礎理論,” [オンライン]. Available: http://www.apmc-mwe.org/mwe2005/src/TL/TL05-01.pdf. [アクセス日: 15 7 2018].

[40] 総務省 , “海上通信,” [オンライン]. Available: http://www.tele.soumu.go.jp/j/adm/system/satellit/marine/index.htm. [アクセス日: 15 7 2018].

[41] アイティメディア株式会社, “これだけは知っておきたいアナログ用語：FM 変調,” 8 12 2014. [オンライン]. Available: http://ednjapan.com/edn/articles/1412/08/news013.html. [アクセス日: 15 7 2018].

[42] 株式会社小野測器, “FFT アナライザについて,” [オンライン]. Available: https://www.onosokki.co.jp/HP-WK/c_support/newreport/analyzer/index.htm. [アクセス日: 16 7 2018].

[43] RASPBERRY PI FOUNDATION, “Raspberry Pi Software Guide,” [オンライン]. Available: https://www.raspberrypi.org/learning/software-guide/. [アクセス日: 19 7 2018].

[44] 総務省 , “海上無線通信の現状,” [オンライン]. Available: http://www.soumu.go.jp/main_sosiki/joho_tsusin/policyreports/chousa/kaijo_senpaku/pdf/080424

_2_si3.pdf. [アクセス日: 23 7 2018].

[45] 総務省, “アマチュア無線局の電波型式を新表記に改正,” [オンライン]. Available: http://www.soumu.go.jp/soutsu/kyushu/ru/file/katashiki.pdf. [アクセス日: 22 7 2018].

[46] 浩 . 山崎 , “通信理論 I,” [オンライン]. Available: http://www.tamagawa.ac.jp/gakubu/kougaku/infceng/csl/yamazaki/print/commun/No.14.pdf. [アクセス日: 23 7 2018].

[47] Audacity, “Audacity,” [オンライン]. Available: https://www.audacityteam.org/. [アクセス日: 20 7 2018].

[48] Mozilla, “Common Voice,” [オンライン]. Available: https://voice.mozilla.org/ja. [アクセス日: 22 7 2018].

[49] Mozilla, “Project DeepSpeech,” [オンライン]. Available: https://github.com/mozilla/DeepSpeech/blob/master/README.md#common-voice-training-data. [アクセス日: 22 7 2018].

[50] Google, “TensorFlow,” [オンライン]. Available: https://www.tensorflow.org/. [アクセス日: 22 7 2018].

[51] Keras, “Keras,” [オンライン]. Available: https://keras.io/ja/. [アクセス日: 23 7 2018].

[52] Microsoft, “CNTK,” [オンライン]. Available: https://github.com/Microsoft/cntk. [アクセス日: 23 7 2018].

[53] Theano, “Theano,” [オンライン]. Available: https://github.com/Theano/Theano. [アクセス日: 23 7 2018].

付録 A　AIS デコード及び AIS と GPS の UTC timestamp を対応付けるプログラム

ソースコード：TimestampMatching.py

```
1  # coding: utf-8
2
3  import argparse
4  import ais
5  import pymap3d as pm
6  import json
7  import csv
8  import time
9  import datetime
10
11 def main():
12     parser = argparse.ArgumentParser (
13         prog='AIS, GPS Timestamp Matching',
14         usage='AIS データと GPS データのタイムスタンプを一致させるプログラム
   ',
15         description='python3 TimestampMatching.py [input AIS filepath] [input GPS
   filepath] [output filepath]',
16         epilog='Copyright 2018 Daisuke Zenju',
17         add_help=True
18         )
19
20     parser.add_argument('-a', '--ais', help='AIS filepath')
21     parser.add_argument('-g', '--gps', help='GPS filepath')
22     parser.add_argument('-o', '--output', help='Output filepath')
23     parser.add_argument('-d', '--day', help='Offset day; ex. 2 (+2day)')
24     parser.add_argument('-j', '--json', help='AIS Decode for JSON', action='store_true')
25     parser.add_argument('-c', '--csv', help='AIS Decode for CSV', action='store_true')
26     parser.add_argument('-l', '--lerp', help='Calc linear interpolation for positions
   "lat,lng,alt"', action='store_true')
27     parser.add_argument('-r', '--range', help='Calc distance between the two points, input
   lattitude, longitude, altitude. ex. 35,135,0')
28     args = parser.parse_args()
29
30     # 引数読込(AIS ファイル名, GPS ファイル名)
31     if args.ais and args.gps and args.output:
32         input_ais_file = args.ais
33         input_gps_file = args.gps
```

```
34
35              print('AIS Filepath : ' + input_ais_file)
36              print('GPS Filepath : ' + input_gps_file)
37
38              # AIS, GPS データファイル読込
39              ais_file = open(input_ais_file, 'r')
40              gps_file = open(input_gps_file, 'r')
41              ais_row = ais_file.readlines()
42              gps_row = gps_file.readlines()
43
44              # AIS, GPS Timestamp Matching
45              output_data = timestamp_matching(ais_row, gps_row)
46
47              #  日時補正
48              if args.day:
49                  offset_day = int(args.day)
50              else:
51                  offset_day = 0
52
53              # Timestamp を GPS Time(UTC)に置換
54              output_data = replace_utc(output_data, offset_day)
55
56              #  引数読込(OUTPUT ファイル名)
57              output_file = args.output
58
59              # AIS Decode & Output
60              if args.csv:
61                  print('OUTPUT(CSV) Filepath : ' + output_file + '_type[number].csv')
62                  ais_data = ais_decode(output_data)
63                  ais_decode_output(output_file, ais_data, 'csv')
64              else:
65                  print('OUTPUT(JSON) Filepath : ' + output_file + '[_types] and
    [_mmsi].json')
66                  ais_data = ais_decode(output_data)
67                  ais_decode_output(output_file, ais_data, 'json')
68
69              # Calc linear interpotion for positoins
```

```
70              if args.lerp:
71                  lerp = calc_lerp(ais_data)
72                  lerp_flag = False
73                  if args.range:
74                      base = args.range.split(',')
75                      lerp = calc_distance(lerp, float(base[0]), float(base[1]), float(base[2]))
76                      lerp_flag = True
77                  if args.csv:
78                      lerp_output(output_file, lerp, 'csv', lerp_flag)
79                  else:
80                      lerp_output(output_file, lerp, 'json', lerp_flag)
81
82                  if args.range:
83                      base = args.range.split(',')
84                      dist = calc_distance(lerp, float(base[0]), float(base[1]), float(base[2]))
85
86
87  # AIS, GPS Timestamp Matching
88  def timestamp_matching(ais_row, gps_row):
89      # AIS list
90      ais_list = []
91      for ais_line in ais_row:
92          ais_split = ais_line.split(' ')
93          msg = ais_split[2].split(',')
94          timestamp = ais_split[0].replace('[', '') + ' ' + ais_split[1].replace(']', '')
95          ais_split = [timestamp + ' ' + msg[0] , msg]
96          ais_list.append(ais_split)
97
98      # GPS dict
99      gps_data = {}
100     for gps_line in gps_row:
101         gps_split = gps_line.split(' ')
102         msg = gps_split[2].split(',')
103         timestamp = gps_split[0].replace('[', '') + ' ' + gps_split[1].replace(']', '')
104         gps_data[timestamp + ' ' + msg[0]] = msg
105
```

```
106         # Output list & AIS, GPS Match
107         output_data = []
108         flag = 0
109         for ais_l in ais_list:
110             if ais_l[1][0] != '!AIVDM':
111                 if ais_l[0] in gps_data.keys():
112                     ais_l[1] = gps_data[ais_l[0]]
113                     flag += 1
114             if flag != 0:
115                 output_data.append([ais_l[0], ais_l[1]])
116
117         return output_data
118
119
120 # Replace UTC Time and Offset day
121 def replace_utc(datas, offset_day):
122         output_data = []
123         timestamp = datetime.datetime(
124             int(datas[0][0][:4]),
125             int(datas[0][0][5:7]),
126             int(datas[0][0][8:10]),
127             int(datas[0][0][11:13]),
128             int(datas[0][0][14:16]),
129             int(datas[0][0][17:19])
130         )
131         diff_time = 0
132         diff_day = datetime.timedelta(days=offset_day)
133
134         for data in datas:
135             if diff_time == 0:
136                 if data[1][0] == '$GPRMC' or data[1][0] == '$GPGGA':
137                     utc_time = datetime.datetime(
138                         int(data[0][:4]),
139                         int(data[0][5:7]),
140                         int(data[0][8:10]),
141                         int(data[1][1][:2]),
```

```
142                             int(data[1][1][2:4]),
143                             int(data[1][1][4:6])
144                         )
145                         stamp_time = datetime.datetime(
146                             int(data[0][:4]),
147                             int(data[0][5:7]),
148                             int(data[0][8:10]),
149                             int(data[0][11:13]),
150                             int(data[0][14:16]),
151                             int(data[0][17:19])
152                         )
153                         diff_time = utc_time - stamp_time
154                 else:
155                     if data[1][0] == '$GPRMC' or data[1][0] == '$GPGGA':
156                         stamp_time = datetime.datetime(
157                             int(data[0][:4]),
158                             int(data[0][5:7]),
159                             int(data[0][8:10]),
160                             int(data[0][11:13]),
161                             int(data[0][14:16]),
162                             int(data[0][17:19])
163                         )
164                         timestamp = stamp_time + diff_time + diff_day
165                     data[0] = timestamp.strftime('%Y-%m-%dT%H:%M:%SZ')
166                     output_data.append(data)
167         output_data = sorted(output_data)
168         return output_data
169
170
171 # Offset Date
172 def offset_date(datas, date_offset):
173         output_data = []
174         diff_time = date_offset - stamp_time
175         print(diff_time)
176         for data in datas:
177                 stamp_time = datetime.datetime(
```

```
178                     int(data[0][:4]),
179                     int(data[0][5:7]),
180                     int(data[0][8:10]),
181                     int(data[0][11:13]),
182                     int(data[0][14:16]),
183                     int(data[0][17:19])
184                 )
185                 timestamp = stamp_time + diff_time
186                 data[0] = timestamp.strftime('%Y-%m-%dT%H:%M:%SZ')
187                 output_data.append(data)
188         output_data = sorted(output_data)
189         return output_data
190
191
192 # AIS, GPS Output (to File)
193 def aisgps_output(output_file, output_data):
194         output = open(output_file + '.nmea', 'w')
195         for out in output_data:
196                 msg = str(out[1])
197                 msg = msg.replace('[', '')
198                 msg = msg.replace(']', '')
199                 msg = msg.replace(' ', '')
200                 output.write(str(out[0]) + ' ' + msg + '¥n')
201         output.close()
202
203
204 # AIS Decode (dict : key[Message Type], value[Message])
205 def ais_decode(output_data):
206         utc = ''
207         msg = ''
208         flag = 0
209         length = 0
210         ais_data = {}
211
212         station_types = {
213                 0: 'All types of mobiles',
```

```
214             1: 'Reserved for future use',
215             2: 'All types of Class B mobile stations',
216             3: 'SAR airborne mobile station',
217             4: 'Aid to Navigation station',
218             5: 'Class B shipborne mobile station (IEC62287 only)',
219             6: 'Regional use and inland waterways',
220             7: 'Regional use and inland waterways',
221             8: 'Regional use and inland waterways',
222             9: 'Regional use and inland waterways',
223             10: 'Reserved for future use',
224             11: 'Reserved for future use',
225             12: 'Reserved for future use',
226             13: 'Reserved for future use',
227             14: 'Reserved for future use',
228             15: 'Reserved for future use'}
229
230         aton_types = {
231             0: 'Default, Type of Aid to Navigation not specified',
232             1: 'Reference point',
233             2: 'RACON (radar transponder marking a navigation hazard)',
234             3: 'Fixed structure off shore, such as oil platforms, wind farms, rigs.',
235             4: 'Spare, Reserved for future use.',
236             5: 'Light, without sectors',
237             6: 'Light, with sectors',
238             7: 'Leading Light Front',
239             8: 'Leading Light Rear',
240             9: 'Beacon, Cardinal N',
241             10: 'Beacon, Cardinal E',
242             11: 'Beacon, Cardinal S',
243             12: 'Beacon, Cardinal W',
244             13: 'Beacon, Port hand',
245             14: 'Beacon, Starboard hand',
246             15: 'Beacon, Preferred Channel port hand',
247             16: 'Beacon, Preferred Channel starboard hand',
248             17: 'Beacon, Isolated danger',
249             18: 'Beacon, Safe water',
```

```
250             19: 'Beacon, Special mark',
251             20: 'Cardinal Mark N',
252             21: 'Cardinal Mark E',
253             22: 'Cardinal Mark S',
254             23: 'Cardinal Mark W',
255             24: 'Port hand Mark',
256             25: 'Starboard hand Mark',
257             26: 'Preferred Channel Port hand',
258             27: 'Preferred Channel Starboard hand',
259             28: 'Isolated danger',
260             29: 'Safe Water',
261             30: 'Special Mark',
262             31: 'Light Vessel / LANBY / Rigs'}
263
264         fix_types = {
265             0: 'Undefined',
266             1: 'GPS',
267             2: 'GLONASS',
268             3: 'Combined GPS/GLONASS',
269             4: 'Loran-C',
270             5: 'Chayka',
271             6: 'Integrated navigation system',
272             7: 'Surveyed',
273             8: 'Galileo'}
274
275         # Match the output of gpsd 3.11.
276         nav_statuses = {
277             0: 'Under way using engine',
278             1: 'At anchor',
279             2: 'Not under command',
280             3: 'Restricted manoeuverability',    # Maneuverability.
281             4: 'Constrained by her draught',
282             5: 'Moored',
283             6: 'Aground',
284             7: 'Engaged in fishing',
285             8: 'Under way sailing',
```

```
286             # Reserved for future amendment of navigational status for ships
287             # carrying DG, HS, or MP, or IMO hazard or pollutant category C,
288             # high speed craft (HSC).
289             9: 'Reserved for HSC',
290             # Reserved for future amendment of navigational status for ships
291             # carrying dangerous goods (DG), harmful substances (HS) or marine
292             # pollutants (MP), or IMO hazard or pollutant category A, wing in
293             # ground (WIG).
294             10: 'Reserved for WIG',
295             # Power-driven vessel towing astern (regional use).
296             11: 'Reserved',
297             # Power-driven vessel pushing ahead or towing alongside (regional use).
298             12: 'Reserved',
299             # Reserved for future use.
300             13: 'Reserved',
301             # AIS-SART (active), MOB-AIS, EPIRB-AIS,
302             14: 'Reserved',
303             # Default (also used by AIS-SART, MOB-AIS and EPIRB-AIS under test).
304             15: 'Not defined'}
305
306         ship_types = {
307             0: 'Not available',
308             1: 'Reserved for future use',
309             2: 'Reserved for future use',
310             3: 'Reserved for future use',
311             4: 'Reserved for future use',
312             5: 'Reserved for future use',
313             6: 'Reserved for future use',
314             7: 'Reserved for future use',
315             8: 'Reserved for future use',
316             9: 'Reserved for future use',
317             10: 'Reserved for future use',
318             11: 'Reserved for future use',
319             12: 'Reserved for future use',
320             13: 'Reserved for future use',
321             14: 'Reserved for future use',
```

```
322             15: 'Reserved for future use',
323             16: 'Reserved for future use',
324             17: 'Reserved for future use',
325             18: 'Reserved for future use',
326             19: 'Reserved for future use',
327             20: 'Wing in ground (WIG), all ships of this type',
328             21: 'Wing in ground (WIG), Hazardous category A',
329             22: 'Wing in ground (WIG), Hazardous category B',
330             23: 'Wing in ground (WIG), Hazardous category C',
331             24: 'Wing in ground (WIG), Hazardous category D',
332             25: 'Wing in ground (WIG), Reserved for future use',
333             26: 'Wing in ground (WIG), Reserved for future use',
334             27: 'Wing in ground (WIG), Reserved for future use',
335             28: 'Wing in ground (WIG), Reserved for future use',
336             29: 'Wing in ground (WIG), Reserved for future use',
337             30: 'Fishing',
338             31: 'Towing',
339             32: 'Towing: length exceeds 200m or breadth exceeds 25m',
340             33: 'Dredging or underwater ops',
341             34: 'Diving ops',
342             35: 'Military ops',
343             36: 'Sailing',
344             37: 'Pleasure Craft',
345             38: 'Reserved',
346             39: 'Reserved',
347             40: 'High speed craft (HSC), all ships of this type',
348             41: 'High speed craft (HSC), Hazardous category A',
349             42: 'High speed craft (HSC), Hazardous category B',
350             43: 'High speed craft (HSC), Hazardous category C',
351             44: 'High speed craft (HSC), Hazardous category D',
352             45: 'High speed craft (HSC), Reserved for future use',
353             46: 'High speed craft (HSC), Reserved for future use',
354             47: 'High speed craft (HSC), Reserved for future use',
355             48: 'High speed craft (HSC), Reserved for future use',
356             49: 'High speed craft (HSC), No additional information',
357             50: 'Pilot Vessel',
```

```
358             51: 'Search and Rescue vessel',
359             52: 'Tug',
360             53: 'Port Tender',
361             54: 'Anti-pollution equipment',
362             55: 'Law Enforcement',
363             56: 'Spare - Local Vessel',
364             57: 'Spare - Local Vessel',
365             58: 'Medical Transport',
366             59: 'Noncombatant ship according to RR Resolution No. 18',
367             60: 'Passenger, all ships of this type',
368             61: 'Passenger, Hazardous category A',
369             62: 'Passenger, Hazardous category B',
370             63: 'Passenger, Hazardous category C',
371             64: 'Passenger, Hazardous category D',
372             65: 'Passenger, Reserved for future use',
373             66: 'Passenger, Reserved for future use',
374             67: 'Passenger, Reserved for future use',
375             68: 'Passenger, Reserved for future use',
376             69: 'Passenger, No additional information',
377             70: 'Cargo, all ships of this type',
378             71: 'Cargo, Hazardous category A',
379             72: 'Cargo, Hazardous category B',
380             73: 'Cargo, Hazardous category C',
381             74: 'Cargo, Hazardous category D',
382             75: 'Cargo, Reserved for future use',
383             76: 'Cargo, Reserved for future use',
384             77: 'Cargo, Reserved for future use',
385             78: 'Cargo, Reserved for future use',
386             79: 'Cargo, No additional information',
387             80: 'Tanker, all ships of this type',
388             81: 'Tanker, Hazardous category A',
389             82: 'Tanker, Hazardous category B',
390             83: 'Tanker, Hazardous category C',
391             84: 'Tanker, Hazardous category D',
392             85: 'Tanker, Reserved for future use',
393             86: 'Tanker, Reserved for future use',
```

```
394             87: 'Tanker, Reserved for future use',
395             88: 'Tanker, Reserved for future use',
396             89: 'Tanker, No additional information',
397             90: 'Other Type, all ships of this type',
398             91: 'Other Type, Hazardous category A',
399             92: 'Other Type, Hazardous category B',
400             93: 'Other Type, Hazardous category C',
401             94: 'Other Type, Hazardous category D',
402             95: 'Other Type, Reserved for future use',
403             96: 'Other Type, Reserved for future use',
404             97: 'Other Type, Reserved for future use',
405             98: 'Other Type, Reserved for future use',
406             99: 'Other Type, no additional information'}
407
408     for data in output_data:
409         if data[1][0] == '!AIVDM':
410             if data[1][1] == '1':
411                 msg = data[1][5]
412                 length = 0
413                 flag = 0
414             else:
415                 if flag == 0:
416                     msg = data[1][5]
417                     flag = 1
418                 else:
419                     msg += data[1][5]
420                     flag = 0
421                     length = 2
422             if flag == 0:
423                 try:
424                     decode = ais.decode(msg, length)
425                     decode['utc'] = data[0]
426                     if 'station_type' in decode:
427                         decode['station_type_text'] =
    station_types[decode['station_type']]
428
429                     if 'aton_type' in decode:
```

```
430                                 decode['aton_type_text'] = aton_types[decode['aton_type']]
431
432                             if 'fix_type' in decode:
433                                 decode['epfd_text(fix_type)'] = fix_types[decode['fix_type']]
434
435                             if 'nav_status' in decode:
436                                 decode['nav_status_text'] = nav_statuses[decode['nav_status']]
437
438                             if 'type_and_cargo' in decode:
439                                 decode['shiptype_text(type_and_cargo)'] =
    ship_types[decode['type_and_cargo']]
440
441                             # Message Type ID
442                             if decode['id'] < 4:
443                                 decode['type'] = 123
444                             else:
445                                 decode['type'] = decode['id']
446
447                             if len(utc) > 0:
448                                 if decode['type'] in ais_data:
449                                     if utc in ais_data[decode['type']]:
450                                         ais_data[decode['type']][utc].append(decode)
451                                     else:
452                                         ais_data[decode['type']].update({utc : [decode]})
453                                 else:
454                                     ais_data[decode['type']] = {utc : [decode]}
455                         except:
456                             continue
457                 else:
458                     # '$GPGGA, $GPRMC --> UTC Time'
459                     utc = data[0]
460
461         return ais_data
462
463
464 # AIS Decode Output
465 def ais_decode_output(output_file, ais_data, filetype):
```

```
466         # CSV
467         if filetype == 'csv':
468
469             for msgtype in ais_data.keys():
470                 output_file_csv = output_file + '_type' + str(msgtype) + '.csv'
471                 output = open(output_file_csv, 'w')
472                 writer = csv.writer(output)
473
474                 utc = ais_data[msgtype].keys()
475                 data = ais_data[msgtype].items()
476
477                 header = []
478                 for dat in data:
479                     for d in dat[1]:
480                         header.extend(d.keys())
481                 header = sorted(list(set(header)))
482                 writer.writerow(header)
483
484                 for dat in data:
485                     for d in dat[1]:
486                         row = []
487                         for h in header:
488                             if h in d:
489                                 row.append(d.get(h))
490                             else:
491                                 row.append('')
492
493                         writer.writerow(row)
494
495                 output.close()
496
497         # JSON
498         else:
499             # Types - Timestamp - Ships
500             output_file_jsn = output_file + '_types.json'
501             output = open(output_file_jsn, 'w')
```

```
502             json.dump(ais_data, output)
503             output.close()
504
505             # MMSI - Types - Timestamp
506             output_file_jsn = output_file + '_mmsi.json'
507             output = open(output_file_jsn, 'w')
508
509             mmsi = {}
510             for msgtype in ais_data.keys():
511                 data = ais_data[msgtype].items()
512                 for dat in data:
513                     for d in dat[1]:
514                         if 'mmsi' in d:
515                             if not mmsi.get(d['mmsi']):
516                                 mmsi.update({d['mmsi']:{msgtype:{d['utc']:d}}})
517                             else:
518                                 if msgtype in mmsi[d['mmsi']]:
519                                     mmsi[d['mmsi']][msgtype].update({d['utc']:d})
520                                 else:
521                                     mmsi[d['mmsi']].update({msgtype:{d['utc']:d}})
522
523             json.dump(mmsi, output)
524             output.close()
525
526 def calc_lerp(ais_data):
527     ships = {}
528     data = ais_data[123].items()
529     for dat in data:
530         for d in dat[1]:
531             if 'mmsi' in d:
532                 position = {
533                     'lat' : d['y'],
534                     'lng' : d['x']
535                 }
536                 if not ships.get(d['mmsi']):
537                     ships.update({d['mmsi']:{d['utc']:position}})
```

```
538                         else:
539                             ships[d['mmsi']].update({d['utc']:position})
540         result = {}
541         for mmsi in ships.items():
542             utcdatas = list(mmsi[1].keys())
543             len_utc = len(utcdatas)
544             if len_utc > 1:
545                 for i in range(len_utc - 1):
546                     start_time = datetime.datetime.strptime(utcdatas[i],
    '%Y-%m-%dT%H:%M:%SZ')
547                     end_time = datetime.datetime.strptime(utcdatas[i+1],
    '%Y-%m-%dT%H:%M:%SZ')
548                     start_unix = int(time.mktime(start_time.timetuple()))
549                     end_unix = int(time.mktime(end_time.timetuple()))
550                     sub_second = end_unix - start_unix
551                     for sec in range(sub_second):
552                         now_unix = start_unix + sec
553                         now_time =
    datetime.datetime.fromtimestamp(now_unix).strftime('%Y-%m-%dT%H:%M:%SZ')
554                         lat0 = mmsi[1][utcdatas[i]]['lat']
555                         lat1 = mmsi[1][utcdatas[i+1]]['lat']
556                         lng0 = mmsi[1][utcdatas[i]]['lng']
557                         lng1 = mmsi[1][utcdatas[i+1]]['lng']
558                         lat = lat0 + (lat1 - lat0) * sec / sub_second
559                         lng = lng0 + (lng1 - lng0) * sec / sub_second
560                         if not result.get(mmsi[0]):
561                             result[mmsi[0]] = {now_time:{'lat':lat, 'lng':lng}}
562                         else:
563                             result[mmsi[0]].update({now_time:{'lat':lat, 'lng':lng}})
564                 result[mmsi[0]].update({utcdatas[-1]:mmsi[1][utcdatas[-1]]})
565             else:
566                 result[mmsi[0]] = mmsi[1]
567         return result
568
569
570 def lerp_output(output_file, lerp_data, filetype, distance_flag):
571         if filetype == 'csv':
572             output_file_csv = output_file + '_lerp.csv'
```

```
573             output = open(output_file_csv, 'w')
574             writer = csv.writer(output)
575             if distance_flag == True:
576                 header = ['MMSI', 'UTC Time', 'Latitude', 'Longitude', 'Azimuth(deg)',
'Elevation(deg)', 'Slant Range(m)']
577             else:
578                 header = ['MMSI', 'UTC Time', 'Latitude', 'Longitude']
579             writer.writerow(header)
580             for dat in lerp_data.items():
581                 for d in dat[1]:
582                     if distance_flag == True:
583                         line = [dat[0], d, dat[1][d]['lat'], dat[1][d]['lng'], dat[1][d]['az'],
dat[1][d]['el'], dat[1][d]['range']]
584                     else:
585                         line = [dat[0], d, dat[1][d]['lat'], dat[1][d]['lng']]
586                     writer.writerow(line)
587             output.close()
588
589         else:
590             # MMSI - Time - Position
591             output_file_jsn = output_file + '_lerp_mmsi.json'
592             output = open(output_file_jsn, 'w')
593             json.dump(lerp_data, output)
594             output.close()
595
596             # Time - MMSI - Position
597             output_file_jsn = output_file + '_lerp_time.json'
598             output = open(output_file_jsn, 'w')
599             times = {}
600             for dat in lerp_data.items():
601                 for d in dat[1]:
602                     if not times.get(d):
603                         times.update({d:{dat[0]:{'lat':dat[1][d]['lat'],
'lng':dat[1][d]['lng']}}})
604                     else:
605                         if dat[0] in times[d]:
606                             times[d][dat[0]].update({'lat':dat[1][d]['lat'],
'lng':dat[1][d]['lng']})
607                         else:
```

```
608                                 times[d].update({dat[0]:{'lat':dat[1][d]['lat'],
'lng':dat[1][d]['lng']}})
609             json.dump(times, output)
610             output.close()
611
612
613 def calc_distance(lerp_data, latitude, longitude, altitude):
614     for dat in lerp_data.items():
615         for d in dat[1]:
616             az, el, dst = pm.geodetic2aer(dat[1][d]['lat'], dat[1][d]['lng'], 0, latitude,
longitude, altitude)
617             lerp_data[dat[0]][d].update({'az':az, 'el':el, 'range':dst})
618     return lerp_data
619
620
621 if __name__ == '__main__':
622     main()
```

付録 B　第 5 章の実験で作成したプログラム

ソースコード：read_state.py

```
1  import pandas as pd
2  import numpy as np
3  import sys
4  import pymap3d as pm
5  import csv
6  import matplotlib.pyplot as plt
7  from mpl_toolkits.mplot3d import Axes3D
8
9
10 def loadAWS1Data(fname, tstart=0, tend=sys.float_info.max):
11 df = pd.read_csv(fname, index_col=0)
12 df = df[(df.index > tstart) & (df.index < tend)]
13 df = calcElapsedTime(df, basetime=tstart)
14 df = df.reset_index().set_index('elapsed')
15 return(df)
16
17
18 def calcElapsedTime(df, basetime=0):
19 tn = df.index
20 elapsed = []
21 for t in tn:
22 elapsed.append(float(t - basetime) / 10000000.0)
23 df['elapsed'] = elapsed
24 return df
25
26
27 def calcGeodesics(df, lat0, lon0, alt0):
28 latn = df.loc[:,' lat']
29 lonn = df.loc[:,' lon']
30 altn = df.loc[:,' alt']
31
32 azn = []
33 eln = []
34 dstn = []
```

```
35 for i in range(len(latn)):
36 az, el, dst = pm.geodetic2aer(latn.iat[i], lonn.iat[i], altn.iat[i], lat0, lon0, alt0)
37 azn.append(az)
38 eln.append(el)
39 dstn.append(dst)
40 df['azimuth'] = azn
41 df['elevation'] = eln
42 df['slantrange'] = dstn
43 return df
44
45
46 '''
47 State DATA
48
49 fname = './state_15285067310000000.txt'
50 df = loadAWS1Data(fname, tstart=15285071030000000, tend=15285116280000000)
51 df = calcGeodesics(df, 35.61677460, 139.89745759, 6.57149905)
52 df.to_csv('./state_geodesics.csv')
53 '''
54
55
56 '''
57 RMS DATA
58
59 npy_rms = './npy/npy_rms/npy_rms_'
60 csv_fname = './cut.csv'
61 cut = pd.read_csv(csv_fname, header=None)
62 rms = []
63 for i in range(len(cut)):
64 t = float(cut.iloc[i, 0]) / 22050.0
65 nprms = np.load(npy_rms + str(i) + '.npy')
66 lsrms = nprms.tolist()
67 lsrms.insert(0, t)
68 rms.append(lsrms)
69 dfrms = pd.DataFrame(rms, columns=['t', 'all', 'beep', 'nosound', 'eng_m', 'eng_f', 'jpn_f',
'jpn_m', 'google'])
70 dfrms.to_csv('./rms.csv')
```

```
71 '''
```

ソースコード：mkcsv.py

```
 1 # coding: utf-8
 2
 3 import numpy as np
 4 import pandas as pd
 5 import csv
 6
 7
 8 state_path = './state_geodesics.csv'
 9 wavcut_path = './cut.csv'
10 sr = 22050.0 # sampling rate
11 psd_path = './npy/npy_psd/npy_psd_'
12 output_path = './keras/'
13
14 # Read State Data
15 state_df = pd.read_csv(state_path)
16 time_slantrange_df = state_df.loc[:,['elapsed', 'slantrange']]
17
18 # Read Wav Cat Position Data
19 wavcut_df = pd.read_csv(wavcut_path, header=None)
20 wavcut_ls = wavcut_df.values.tolist()
21
22 # Read PSD's Numpy Data
23 psdnpys = []
24 for i in range(246):
25 path = psd_path + str(i) + '.npy'
26 psdnpys.append(np.load(path))
27
28 # Add time position
29 alls = 325084.0 / 2.0 / sr
30 beep = 11025.0 / 2.0 / sr
31 nosound = (41923-12159) / 2.0 / sr
32 eng_m = (108135-44100) / 2.0 / sr
33 eng_f = (184526-121157) / 2.0 / sr
```

```
34  jpn_f = (224557-198487) / 2.0 / sr
35  jpn_m = (286422-236195) / 2.0 / sr
36  google = (325084-294637) / 2.0 / sr
37  addt = [alls, beep, nosound, eng_m, eng_f, jpn_f, jpn_m, google]
38
39  idx = []
40  #out_ls = [[] for i in range(len(addt))]
41  out_ls = []
42
43  for i in range(246):
44  sample = wavcut_ls[i][0]
45  t = float(sample) / sr
46  if (i == 0):
47  idx = list(psdnpys[i][0][0])
48  idx.append('slantrange')
49
50  out_ls2 = []
51  for j in range(len(addt)):
52  pt = t + addt[j]
53  pdf = time_slantrange_df[time_slantrange_df['elapsed'] > pt].iloc[0,:]
54  ls3 = list(psdnpys[i][j][1])
55  ls3.append(pdf['slantrange'])
56  out_ls2.append(ls3)
57  out_ls.append(out_ls2)
58
59
60  for i in range(len(addt)):
61  f = open(output_path + 'psd' + str(i) + '.csv', 'w')
62  wr = csv.writer(f, lineterminator='¥n')
63  wr.writerow(idx)
64  for j in range(246):
65  wr.writerow(out_ls[j][i])
66  f.close()
```

ソースコード：read_state.py

```
# coding: utf-8

import librosa
import librosa.display
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
import csv


### MEMO
# default: n_fft = 2048, win_length = n_fft,
# hop_length = win_length / 4, D:np.ndarray[shape=(1+n_fft/2, t)]
# frame = T * sr / hop_length

# RMS = sqrt(1/n sum=(for i in d i^2))
# RMS(db) = 20*log(RMS(V))

def rms(data):
sum2 = 0
for d in data:
sum2 += d * d
result = 20* np.log10(np.sqrt(sum2 / len(data)))
return result


def calc_fft(data):
ffts = []
for y in ys:
N = len(y)
hann = np.hanning(N)
wdat = hann * y
wfft = np.fft.fft(wdat)
freq = np.fft.fftfreq(N, d=1.0/sr)
ffts.append([freq, wfft])
return ffts
```

```
37
38
39  def calc_rms(data):
40  rmss = []
41  for i in data:
42  rmss.append(rms(i))
43  print(rmss)
44  return rmss
45
46
47  def calc_psd(data):
48  psds = []
49  for y in ys:
50  freq, psd = signal.welch(y, sr)
51  psds.append([freq, psd])
52  return psds
53
54
55  def calc_stft(data):
56  stfts = []
57  for y in ys:
58  speg = librosa.stft(y)
59  stfts.append(speg)
60  return stfts
61
62
63  def split_wav(row, sr, offset):
64  # Sampling Rate (Hz) : 22050
65  beep = row[0+offset:11025+offset]
66  nosound = row[12159+offset:41923+offset]
67  eng_m = row[44100+offset:108135+offset]
68  eng_f = row[121157+offset:184526+offset]
69  jpn_f = row[198487+offset:224557+offset]
70  jpn_m = row[236195+offset:286422+offset]
71  google = row[294637+offset:325084+offset]
72  dat = [row[0+offset:325084+offset], beep, nosound, eng_m, eng_f, jpn_f, jpn_m,
    google]
```

```
73  return dat
74
75
76  def plt_wav(data, sr, title):
77  titles = [
78  'Sample', '440Hz Beep', 'No Sound', 'English Male', 'English Female',
79  'Japanese Female', 'Japanese Male', 'OK, Google']
80  xlabel = 'Samples'
81  ylabel = 'Amplitude'
82  plt.figure(figsize = (16, 9))
83  for i in range(0,len(titles)):
84  y = data[i]
85  plt.subplot(2, 4, i + 1)
86  plt.title(titles[i])
87  librosa.display.waveplot(y, sr=sr)
88  plt.tight_layout()
89  plt.savefig('./fig/fig_wav/' + title + '.png')
90
91
92  def plt_fft(data, title, flag):
93  titles = [
94  'Sample', '440Hz Beep', 'No Sound', 'English Male', 'English Female',
95  'Japanese Female', 'Japanese Male', 'OK, Google']
96  plt.figure(figsize = (16, 9))
97  plt.suptitle(title)
98  for i in range(0,len(titles)):
99  freq = data[i][0]
100 amp = np.abs(data[i][1])
101 n = len(freq)
102 plt.subplot(2, 4, i + 1)
103 plt.plot(freq[1:int(n/2)], amp[1:int(n/2)])
104 plt.xlabel('Frequency (Hz)')
105 #plt.ylabel('Amplitude')
106 plt.ylabel('PSD [V**2/Hz]')
107 plt.title(titles[i])
108 plt.tight_layout()
```

```
109
110 if flag == 0:
111 # FFT
112 plt.savefig('./fig/fig_fft/' + title + '.png')
113 elif flag == 1:
114 # PSD
115 plt.savefig('./fig/fig_psd/' + title + '.png')
116
117
118 def plt_stft(data, title):
119 titles = [
120 'Sample', '440Hz Beep', 'No Sound', 'English Male', 'English Female',
121 'Japanese Female', 'Japanese Male', 'OK, Google']
122 plt.figure(figsize = (16, 9))
123 plt.suptitle(title)
124 for i in range(0,len(titles)):
125 D = data[i]
126 plt.subplot(2, 4, i + 1)
127 librosa.display.specshow(
128 librosa.amplitude_to_db(librosa.magphase(D)[0]),
129 y_axis='log', x_axis='time')
130 plt.title(titles[i])
131 plt.colorbar(format='%+2.0f dB')
132 plt.tight_layout()
133 plt.savefig('./fig/fig_stft/' + title + '.png')
134
135
136
137
138 # 音声ファイル読み込み
139 wav_fname = './180609_101823_113348.wav'
140 rowy, sr = librosa.load(wav_fname)
141 print ('Data Length : ' + str(len(rowy)))
142 print('Sampling Rate : ' + str(sr) + '[Hz]')
143
144 # 切り取り情報読み込み
```

```
145 csv_fname = './cut.csv'
146 with open(csv_fname, 'r') as cf:
147 reader = csv.reader(cf)
148 for i, st in enumerate(reader):
149
150 #if i < 230:
151 # continue
152
153 print(i, st[0])
154
155 # 音声ファイル分割
156 print('分割')
157 ys = split_wav(rowy, sr, int(st[0]))
158
159 print(len(ys[0]))
160
161 # RMS
162 #print('RMS')
163 #rmss = calc_rms(ys)
164 # FFT
165 #print('FFT')
166 #ffts = calc_fft(ys)
167 # PSD
168 print('PSD')
169 psds = calc_psd(ys)
170 # STFT
171 #print('STFT')
172 #stfts = calc_stft(ys)
173
174
175 # SAVE
176 #np.save('./npy/npy_wav/npy_wav_' + str(i) + '.npy', ys)
177 #np.save('./npy/npy_rms/npy_rms_' + str(i) + '.npy', rmss)
178 #np.save('./npy/npy_fft/npy_fft_' + str(i) + '.npy', ffts)
179 #np.save('./npy/npy_psd/npy_psd_' + str(i) + '.npy', psds)
180
```

```
181    #plt_wav(ys, sr, 'Sound_Wave(' + str(i) + ')')
182    #plt_fft(ffts, 'FFT(' + str(i) + ')', 0)
183    plt_fft(psds, 'PSD(' + str(i) + ')', 1)
184    #plt_stft(stfts, 'Power_spectrogram(' + str(i) + ')')
185    #plt.show()
```

付録 C　第 6 章の実験で作成したプログラム

ソースコード：mkcsv.py

```
1  # coding: utf-8
2
3  import numpy as np
4  import pandas as pd
5  import csv
6
7
8  state_path = './state_geodesics.csv'
9  wavcut_path = './cut.csv'
10 sr = 22050.0 # sampling rate
11 psd_path = './npy/npy_psd/npy_psd_'
12 output_path = './keras/'
13
14 # Read State Data
15 state_df = pd.read_csv(state_path)
16 time_slantrange_df = state_df.loc[:,['elapsed', 'slantrange']]
17
18 # Read Wav Cat Position Data
19 wavcut_df = pd.read_csv(wavcut_path, header=None)
20 wavcut_ls = wavcut_df.values.tolist()
21
22 # Read PSD's Numpy Data
23 psdnpys = []
24 for i in range(246):
25 path = psd_path + str(i) + '.npy'
26 psdnpys.append(np.load(path))
27
28 # Add time position
29 alls = 325084.0 / 2.0 / sr
30 beep = 11025.0 / 2.0 / sr
31 nosound = (41923-12159) / 2.0 / sr
32 eng_m = (108135-44100) / 2.0 / sr
33 eng_f = (184526-121157) / 2.0 / sr
34 jpn_f = (224557-198487) / 2.0 / sr
```

```
35  jpn_m = (286422-236195) / 2.0 / sr
36  google = (325084-294637) / 2.0 / sr
37  addt = [alls, beep, nosound, eng_m, eng_f, jpn_f, jpn_m, google]
38
39  idx = []
40  #out_ls = [[] for i in range(len(addt))]
41  out_ls = []
42
43  for i in range(246):
44  sample = wavcut_ls[i][0]
45  t = float(sample) / sr
46  if (i == 0):
47  idx = list(psdnpys[i][0][0])
48  idx.append('slantrange')
49
50  out_ls2 = []
51  for j in range(len(addt)):
52  pt = t + addt[j]
53  pdf = time_slantrange_df[time_slantrange_df['elapsed'] > pt].iloc[0,:]
54  ls3 = list(psdnpys[i][j][1])
55  ls3.append(pdf['slantrange'])
56  out_ls2.append(ls3)
57  out_ls.append(out_ls2)
58
59
60  for i in range(len(addt)):
61  f = open(output_path + 'psd' + str(i) + '.csv', 'w')
62  wr = csv.writer(f, lineterminator='¥n')
63  wr.writerow(idx)
64  for j in range(246):
65  wr.writerow(out_ls[j][i])
66  f.close()
```

ソースコード：mkdataset.py

```
1  # coding: utf-8
2
```

```
3  import awsTime
4  import pandas as pd
5  import numpy as np
6
7  import tqdm
8  import types
9
10 start_time = 1529801210.4710002 * 10000000 # Unix Time (microsec)
11 sr = 22050 # sampling rate
12
13 using_time = 160 # sec (2m40s --> 160 microsec)
14
15 logtime_path = 'log0624.txt' # timestamp
16 state_path = 'state0624.csv' # state(slantrange)
17 npy_path = 'log/log0624/log0624' # spectrogram(PSD)
18
19 outpath = 'dataset/dataset0624'
20
21 # load datas
22 logtime = awsTime.readUnixtime(logtime_path)
23 state = pd.read_csv(state_path)
24 f = np.load(npy_path + '-freq.npy')
25 t = np.load(npy_path + '-time.npy')
26 Sxx = np.load(npy_path + '-STFT.npy')
27 sxx = Sxx.T
28
29
30 state = state.loc[:, ['t', 'dst']]
31 state['t'] -= start_time
32 state['t'] /= 10000000.0
33 state = state[state['t'] >= 0]
34 state = state[state['t'] < t[-1]]
35
36
37 state_t = np.array(state['t'])
38 idx = np.searchsorted(state_t, t, side='left')
```

```
39 piil = ((idx == len(state_t)) | (np.fabs(t - state_t[np.maximum(idx-1, 0)]) < np.fabs(t-state_t[np.minimum(idx, len(state_t)-1)])))
40 idx[piil] -= 1
41
42 dstls = []
43 for i in tqdm.tqdm(range(len(idx))):
44 ist = idx[i]
45 dst = state.iloc[ist, :]
46 dstls.append(dst['dst'])
47
48 dset_t = []
49 dset_sxx = np.empty([0, 129])
50 dset_dst = []
51 for lt in tqdm.tqdm(logtime):
52 slt = lt - start_time / 10000000.0
53 elt = slt + using_time
54 selt = [slt, elt]
55 idx = np.searchsorted(t, selt, side='left')
56 dset_t.append(t[idx[0]:idx[1]])
57 dset_dst.append(dstls[idx[0]:idx[1]])
58 dset_sxx = np.r_[dset_sxx, sxx[idx[0]:idx[1]]]
59
60 dset_t = [flatten for inner in dset_t for flatten in inner]
61 dset_dst = [flatten for inner in dset_dst for flatten in inner]
62
63 dset_t = np.array(dset_t)
64 dset_dst = np.array(dset_dst)
65
66 np.savetxt(outpath + '-t', dset_t)
67 np.savetxt(outpath + '-dst', dset_dst)
68 np.savetxt(outpath + '-stft', dset_sxx)
69 np.savetxt(outpath + '-f', f)
```

ソースコード：wav.py

```
1 # coding: utf-8
2
3
```

```
 4  import numpy as np
 5  import scipy
 6  import librosa
 7  import librosa.display
 8  import matplotlib as mpl
 9  import matplotlib.pyplot as plt
10  from matplotlib.colors import LogNorm
11  import time
12
13  start = time.time()
14
15  wavfname = 'VOR002.WAV'
16  logfname = 'log/log0624'
17
18  # read wav file
19  y, sr = librosa.load(wavfname)
20
21  elapsed_time = time.time() - start
22  print('ReadingTime (Wave Read): ' + str(elapsed_time))
23
24  print('Sample rate (Hz): ' + str(sr))
25  print('Num of Frame: ' + str(len(y)))
26  print('Sound Time (sec): ' + str(len(y) / sr))
27
28  # calc spectrogram (PSD)
29  f, t, Sxx = scipy.signal.spectrogram(y, sr)
30
31  # save logs'
32  np.save(logfname + '-freq', f)
33  np.save(logfname + '-time', t)
34  np.save(logfname + '-STFT', Sxx)
35
36  elapsed_time = time.time() - start
37  print('ReadingTime (Calc Spectrogram(PSD)): ' + str(elapsed_time))
38
39
```

```
40  # save sound wave, spectrogram
41  mpl.reParams['agg.path.shunksize'] = 10000000000
42
43  plt.subplot(211)
44  librosa.display.waveplot(y, sr=sr, x_axis='time')
45  plt.title('Sound Wave')
46
47  plt.subplot(212)
48  plt.pcolormesh(t, f, Sxx, norm=LogNorm())
49  pp = plt.colorbar(orientation='vertical')
50  pp.set_label('PSD [V**2/Hz]')
51  plt.title('Spectrogram (PSD)')
52  plt.ylabel('Frequency [Hz]')
53  plt.xlabel('Time [sec]')
54
55  plt.tight_layout()
56  plt.savefig(logfname + '-plot.png')
57
58  elapsed_time = time.time() - start
59  print('ReadingTime (Plot Wave & Spectrogram): ' + str(elapsed_time))
```

ソースコード：awsState.py

```
1   # coding: utf-8
2   import pandas as pd
3   import pymap3d as pm
4
5   def addAER(fname, lat0, lon0, alt0):
6   df = pd.read_csv(fname, index_col='t')
7   azimuth = []
8   elevation = []
9   slantrange = []
10  for i in range(len(df)):
11  lat = df.iat[i, 5]
12  lon = df.iat[i, 6]
13  alt = df.iat[i, 7]
14  az, el, sr = pm.geodetic2aer(lat, lon, alt, lat0, lon0, alt0)
```

```
15  azimuth.append(az)
16  elevation.append(el)
17  slantrange.append(sr)
18  df['azimuth'] = azimuth
19  df['elevation'] = elevation
20  df['slantrange'] = slantrange
21  return df
22
23
24  if __name__ == '__main__':
25  fname = 'state_15297178270000000.txt'
26  outname = 'state0623.csv'
27  lat = 35.6328616524447
28  lon = 139.925418198239
29  alt = 1.9986038208007812
30
31  data = addAER(fname, lat, lon, alt)
32  data.to_csv(outname)
```

ソースコード：awsTime.py

```
1  # coding: utf-8
2  import datetime
3
4  def read_timelog(fname):
5  f = open(fname, 'r')
6  lines = f.readlines()
7  f.close
8  return lines
9
10
11  def timelog2unixtime(timelog):
12  timelog = timelog.replace('[', '').replace(']', '')
13  splittime = timelog.split(' ')
14  month = splittime[1]
15  day = splittime[2]
16  time = splittime[3]
```

```
17  year = splittime[4]
18  dl = year + '-' + month + '-' + day + ' ' + time
19  dt = datetime.datetime.strptime(dl, '%Y-%b-%d %H:%M:%S.%f')
20  return dt.timestamp()
21
22
23  def readUnixtime(fname):
24  timeline = read_timelog(fname)
25  unixtime = []
26  for tline in timeline:
27  utime = timelog2unixtime(tline)
28  unixtime.append(utime)
29  return unixtime
```

ソースコード：drawGraph.py

```
1   import numpy as np
2   import pandas as pd
3   import matplotlib.pyplot as plt
4   from mpl_toolkits.mplot3d import Axes3D
5   from matplotlib.colors import LogNorm
6   import math
7
8   #logf(0: linear, 1: log)
9   logf = 1
10
11  stft_path23 = 'dataset/dataset0623/dataset0623-stft'
12  t_path23 = 'dataset/dataset0623/dataset0623-t'
13  f_path23 = 'dataset/dataset0623/dataset0623-f'
14  dst_path23 = 'dataset/dataset0623/dataset0623-dst'
15
16  stft_path24 = 'dataset/dataset0624/dataset0624-stft'
17  t_path24 = 'dataset/dataset0624/dataset0624-t'
18  f_path24 = 'dataset/dataset0624/dataset0624-f'
19  dst_path24 = 'dataset/dataset0624/dataset0624-dst'
20
21  t23 = pd.read_csv(t_path23, delim_whitespace=True, header=None)
```

```
22  f23 = pd.read_csv(f_path23, delim_whitespace=True, header=None)
23  stft23 = pd.read_csv(stft_path23, delim_whitespace=True, header=None)
24  dst23 = pd.read_csv(dst_path23, delim_whitespace=True, header=None)
25  dst23.columns = ['dst']
26
27  t24 = pd.read_csv(t_path24, delim_whitespace=True, header=None)
28  f24 = pd.read_csv(f_path24, delim_whitespace=True, header=None)
29  stft24 = pd.read_csv(stft_path24, delim_whitespace=True, header=None)
30  dst24 = pd.read_csv(dst_path24, delim_whitespace=True, header=None)
31  dst24.columns = ['dst']
32
33  dst_stft23 = pd.concat([dst23, stft23], axis=1)
34  dst_stft23 = dst_stft23.sort_values(['dst'])
35
36  dst_stft24 = pd.concat([dst24, stft24], axis=1)
37  dst_stft24 = dst_stft24.sort_values(['dst'])
38
39  dst_min23 = math.floor(dst_stft23['dst'].min())
40  dst_max23 = math.ceil(dst_stft23['dst'].max())
41
42  dst_min24 = math.floor(dst_stft24['dst'].min())
43  dst_max24 = math.ceil(dst_stft24['dst'].max())
44
45  mean_ls23 = []
46  for i in range(dst_min23, dst_max23):
47  if i % 10 == 0:
48  choose = dst_stft23[(dst_stft23['dst'] >= dst_min23) & (dst_stft23['dst'] < i)]
49  mean = choose.mean()
50  mean_ls23.append(mean)
51  dst_min23 = i
52  choose = dst_stft23[dst_stft23['dst'] >= dst_min23]
53  mean = choose.mean()
54  mean_ls23.append(mean)
55
56  mean_ls24 = []
57  for i in range(dst_min24, dst_max24):
```

```
58  if i % 10 == 0:
59  choose = dst_stft24[(dst_stft24['dst'] >= dst_min24) & (dst_stft24['dst'] < i)]
60  mean = choose.mean()
61  mean_ls24.append(mean)
62  dst_min24 = i
63  choose = dst_stft24[dst_stft24['dst'] >= dst_min24]
64  mean = choose.mean()
65  mean_ls24.append(mean)
66
67  mean_df23 = pd.DataFrame(mean_ls23)
68  mean_df23 = mean_df23.dropna(how='all')
69  mean_df23 = mean_df23.reset_index(drop=True)
70
71  mean_df24 = pd.DataFrame(mean_ls24)
72  mean_df24 = mean_df24.dropna(how='all')
73  mean_df24 = mean_df24.reset_index(drop=True)
74
75  dst_df23 = mean_df23['dst']
76  mean_df23 = mean_df23.drop('dst', axis=1)
77
78  dst_df24 = mean_df24['dst']
79  mean_df24 = mean_df24.drop('dst', axis=1)
80
81  f23 = np.array(f23)
82  dst23 = np.array(dst_df23)
83  mean23 = np.array(mean_df23)
84
85  f24 = np.array(f24)
86  dst24 = np.array(dst_df24)
87  mean24 = np.array(mean_df24)
88
89  F23, D23 = np.meshgrid(f23.T[0], dst23)
90  F24, D24 = np.meshgrid(f24.T[0], dst24)
91
92  fig = plt.figure()
93  ax = Axes3D(fig)
```

```
94
95  print(mean23.shape)
96
97  if logf == 1:
98  #log
99  ax.plot_wireframe(D23, np.log10(F23), mean23, color='blue', label='2018/06/23')
100 ax.plot_wireframe(D24, np.log10(F24), mean24, color='red', label='2018/06/24')
101 ax.set_ylabel('Frequency (Log10) [Hz]')
102 else:
103 #linear
104 ax.plot_wireframe(D23, F23, mean23, color='blue', label='2018/06/23')
105 ax.plot_wireframe(D24, F24, mean24, color='red', label='2018/06/24')
106 ax.set_ylabel('Frequency [Hz]')
107
108 ax.set_xlabel('Slant Range [m]')
109 ax.set_zlabel('PSD [V**2/Hz]')
110 plt.legend()
111 plt.show()
112
113 """
114 F, D = np.meshgrid(np_f.T[0], np_dst.T[0])
115
116 fig = plt.figure()
117 ax = Axes3D(fig)
118 ax.plot_wireframe(F, D, np_stft)
119 ax.set_xlabel('x')
120 ax.set_ylabel('y')
121 ax.set_zlabel('z')
122 plt.show()
123
124 """
125
126 """
127 np_stftn = np.loadtxt(stft_path)
128 np_t = np.loadtxt(t_path)
129 np_f = np.loadtxt(f_path)
```

```
130  np_dst = np.loadtxt(dst_path)
131
132  plt.subplot(211)
133  plt.pcolormesh(np_t, np_f, np_stft.T, norm=LogNorm())
134  pp = plt.colorbar(orientation='horizontal')
135  pp.set_label('PSD [V**2/Hz]')
136  plt.title('Spectrogram (PSD)')
137  plt.ylabel('Frequency [Hz]')
138  plt.xlabel('Time [sec]')
139
140  plt.subplot(212)
141  plt.plot(np_t, np_dst)
142  plt.title('Slant Range')
143  plt.xlabel('Time [sec]')
144  plt.ylabel('Slant Range [m]')
145
146  plt.tight_layout()
147  plt.suptitle('Log 2018/06/23')
148  plt.subplots_adjust(top=0.9)
149
150  #plt.savefig('log0623-plot.png')
151  plt.show()
152  """
```

ソースコード：wagiri.py

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  # read data
6  datpath = './pickle/dataset0624.pkl'
7  #datpath = './pickle/dataset.pkl'
8  fpath = './dataset/dataset0623/dataset0623-f'
9  dat_df = pd.read_pickle(datpath)
10 f_df = np.loadtxt(fpath)
11
```

```
12  # rename columns
13  idx = f_df.tolist()
14  idx.append('slant_range')
15  dat_df.columns = idx
16
17  # sort by slant_range and reset index
18  dat_df = dat_df.sort_values('slant_range')
19  dat_df = dat_df.reset_index(drop=True)
20
21  dat_min = 0.0
22  dat_max = 0.0
23  for i in range(129):
24  mn = dat_df[idx[i]].min()
25  mx = dat_df[idx[i]].max()
26  if (mn < dat_min):
27  dat_min = mn
28  if (mx > dat_max):
29  dat_max = mx
30  print(dat_min, dat_max)
31
32  """
33  # save range-psd graphs
34  for i in range(129):
35  plt.plot(dat_df['slant_range'], dat_df[idx[i]])
36  plt.ylim([dat_min, dat_max])
37  plt.ylabel('PSD [V**2/Hz]')
38  plt.xlabel('Slant Range [m]')
39  #title = 'Slant Range - PSD [' + str(int(idx[i])) + ' Hz] 18/06/23'
40  #path = './wagiri/day23/range_psd/range_psd_23_' + str(int(idx[i])) + '.png'
41  title = 'Slant Range - PSD [' + str(int(idx[i])) + ' Hz] All'
42  path = './wagiri/all/range_psd/range_psd_' + str(int(idx[i])) + '.png'
43  plt.title(title)
44  plt.tight_layout()
45  plt.savefig(path)
46  print('save: ' + path)
47  plt.cla()
```

```
48  """
49  # save freq-psd graphs
50  for i in range(30, 2450, 10):
51  df = dat_df[(dat_df['slant_range'] > i-5) & (dat_df['slant_range'] < i+5)]
52  df = df.mean()
53  plt.plot(idx[0:128], df[0:128])
54  plt.ylim([dat_min, 0.0003])
55  plt.ylabel('PSD [V**2/Hz]')
56  plt.xlabel('Frequency [Hz]')
57  title = 'Frequency - PSD [' + str(i) + ' m] 06/24'
58  path = './wagiri/day24/freq_psd/freq_psd_24_' + str(i) + '.png'
59  #title = 'Frequency - PSD [' + str(i) + ' m] All'
60  #path = './wagiri/all/freq_psd/freq_psd_' + str(i) + '.png'
61  plt.title(title)
62  plt.tight_layout()
63  plt.savefig(path)
64  print('save: ' + path)
65  plt.cla()
```

付録 D　第 7 章の実験で作成したプログラム

ソースコード：learn.py

```
1  from keras.models import Sequential
2  from keras.layers import Dense, Activation, Dropout
3  from keras.callbacks import EarlyStopping,
   ModelCheckpoint, TensorBoard, CSVLogger
4  from keras.utils import plot_model
5  import keras.backend as K
6  from keras import optimizers
7  import os
8  import random
9  import glob
10 import math
11 import json
12 import csv
13 import argparse
14 import os
15 import time
16 import gc
17 from tqdm import tqdm
18 import pandas as pd
19 import numpy as np
20 import matplotlib.pyplot as plt
21
22 #mport pdb; pdb.set_trace()
23
24
25
26 class TrainModel:
27 def __init__(self, directory, test_path, norm_x=0.0,
   norm_y=0.0):
28 self.dir = directory
29 self.test_path = directory + test_path
30 self.norm_x = norm_x
31 self.norm_y = norm_y
32
33 ## data files
```

```
34  self.datpath = glob.glob(self.dir + '*')
35  self.datpath = [p.replace(os.sep, '/') for p in self.datpath]
36  ## input size
37  test = pd.read_pickle(self.test_path)
38  self.input_shape = test.shape[1] - 1
39  del test
40
41
42  # model 設計
43  def set_model(self):
44  ## Model
45  self.model = Sequential()
46  self.model.add(Dense(300,
47  activation='relu',
48  kernel_initializer='random_uniform',
49  bias_initializer='zeros',
50  input_shape=(self.input_shape,)))
51  self.model.add(Dropout(0.5))
52  self.model.add(Dense(50,
53  activation='relu',
54  kernel_initializer='random_uniform',
55  bias_initializer='zeros',
56  input_shape=(self.input_shape,)))
57  self.model.add(Dropout(0.5))
58  self.model.add(Dense(300,
59  activation='relu',
60  kernel_initializer='random_uniform',
61  bias_initializer='zeros',
62  input_shape=(self.input_shape,)))
63  self.model.add(Dropout(0.5))
64  self.model.add(Dense(50,
65  activation='relu',
66  kernel_initializer='random_uniform',
67  bias_initializer='zeros',
68  input_shape=(self.input_shape,)))
69  self.model.add(Dropout(0.5))
```

```
70  self.model.add(Dense(1))
71
72  ## Optimizer
73  self.optimizer = optimizers.Adam(lr=1e-5)
74
75  ## Loss
76  self.loss = 'mae'
77
78  # model 生成, metrics 設定
79  def make_model(self, **option):
80  self.set_model()
81  self.history = {'loss':[], 'val_loss':[]}
82
83  if ('metrics' not in option) or (option['metrics'] is None):
84  self.hist_metrics = None
85  self.model.compile(optimizer=self.optimizer,
    loss=self.loss)
86  else:
87  self.hist_metrics = list(option['metrics'])
88  for metric in self.hist_metrics:
89  self.history[metric] = []
90  self.model.compile(optimizer=self.optimizer,
    loss=self.loss, metrics=self.hist_metrics)
91
92  if 'model_path' in option:
93  self.model_path = option['model_path']
94  else:
95  self.model_path = None
96
97  if 'log_path' in option:
98  self.log_path = option['log_path']
99  else:
100 self.log_path = None
101
102 self.model.summary()
103
104
```

```
105 # training
106 def training(self, epochs=1, batch_size=32,
    test_file='dat0.pkl'):
107 if self.log_path is not None:
108 self.epochs = epochs
109 self.batch = batch_size
110 self.save_summary(self.log_path)
111
112 for epoch in tqdm(range(epochs)):
113 ## train
114 for train_path in random.sample(self.datpath,
    len(self.datpath)):
115 if train_path == self.test_path:
116 continue
117
118 else:
119 ## load train file --> shuffle --> split x, y
120 train = pd.read_pickle(train_path)
121 train = train.sample(frac=1.0)
122 train_x = np.array(train.iloc[:,:self.input_shape])
123 train_y = np.array(train.iloc[:,-1])
124 del train
125
126 ## normalize
127 train_x /= self.norm_x
128 train_y /= self.norm_y
129
130 ## train on batch
131 start = 0
132 for i in range(batch_size, len(train_x), batch_size):
133 self.train_score =
    self.model.train_on_batch(train_x[start:i],train_y[start:i])
134 start = i
135 if start < len(train_x):
136 self.train_score =
    self.model.train_on_batch(train_x[start:],train_y[start:])
137
138 del train_x, train_y
139
```

```
140 ## test
141 test = pd.read_pickle(self.test_path)
142 test_x = np.array(test.iloc[:,:self.input_shape])
143 test_y = np.array(test.iloc[:,-1])
144 del test
145 test_x /= self.norm_x
146 test_y /= self.norm_y
147 self.test_score = self.model.test_on_batch(test_x,test_y)
148 self.eval_score = self.model.evaluate(test_x,test_y)
149 del test_x, test_y
150
151 # log loss, val_loss
152 self.log_history()
153 if self.hist_metrics is None:
154 print('epoch: {}, loss: {}, val_loss: {}'.format(epoch,
    self.train_score, self.test_score))
155 else:
156 print('epoch: {}, loss: {}, val_loss: {}'.format(epoch,
    self.train_score[0], self.test_score[0]))
157 if self.log_path is not None:
158 self.save_history_graph(self.log_path)
159 self.predict(self.log_path)
160 self.evaluate(self.log_path)
161 if self.model_path is not None:
162 self.save_model(self.model_path)
163
164 def save_summary(self, path=None):
165 if path is not None:
166 with open(self.log_path + '_summary.txt', 'w') as f:
167 f.write('Epoch: {}¥n'.format(self.epochs))
168 f.write('Batch Size: {}¥n'.format(self.batch))
169 f.write('Loss Func: {}¥n'.format(self.loss))
170 f.write('Optimizer: {}¥n'.format(self.optimizer))
171 f.write('Metrics: {}¥n'.format(self.hist_metrics))
172 f.write('Evaluate File: {}'.format(self.test_path))
173 plot_model(self.model, to_file=self.log_path +
    '_summary.png', show_shapes=True)
174
```

```
175
176 # save model
177 def save_model(self, path=None):
178 self.model.save(path + '.h5')
179 self.model.save_weights(path + '_weights.h5')
180 open(path + '.json', 'w').write(self.model.to_json())
181 open(path + '_weights.txt',
    'w').write(str(self.model.get_weights()))
182
183
184 # log loss, val_loss
185 def log_history(self):
186 if self.hist_metrics is None:
187 self.history['loss'].append(self.train_score)
188 self.history['val_loss'].append(self.test_score)
189 else:
190 self.history['loss'].append(self.train_score[0])
191 self.history['val_loss'].append(self.test_score[0])
192 for i, metric in enumerate(self.hist_metrics):
193 self.history[metric].append(self.test_score[i+1])
194 # plot loss, val_loss
195 def save_history_graph(self, path=None):
196 if path is not None:
197 plt.plot(self.history['loss'], label='loss')
198 plt.plot(self.history['val_loss'], label='val_loss')
199 plt.xlabel('epoch')
200 plt.ylabel('loss')
201 plt.legend()
202 plt.tight_layout()
203 plt.savefig(path + '_history.png')
204 with open(path + '_history.json', 'w') as f:
205 json.dump(str(self.history), f)
206
207
208 def predict(self, path=None):
209 test = pd.read_pickle(self.test_path)
210 test_x = np.array(test.iloc[:,:self.input_shape])
```

```
211 test_y = np.array(test.iloc[:,-1])
212 del test
213 test_x /= self.norm_x
214 test_y /= self.norm_y
215 if path is not None:
216 test_x = self.model.predict_on_batch(test_x)
217 test_x *= self.norm_y
218 test_y = test_y * self.norm_y
219 test_x = pd.DataFrame(test_x, columns=['predict'])
220 test_y = pd.DataFrame(test_y, columns=['slantRange'])
221 predict_pd = pd.concat([test_y, test_x], axis=1)
222 del test_x, test_y
223 predict_pd = predict_pd.sort_values('slantRange')
224 predict_pd.to_csv(path + '_predict.csv', index=False,
    header=False)
225
226 plt.subplot(211)
227 plt.plot(predict_pd['slantRange'], predict_pd['predict'])
228 plt.xlabel('Observed Slant Range [m]')
229 plt.ylabel('Predict Slant Range [m]')
230 plt.subplot(212)
231 plt.plot(predict_pd['slantRange'],
    predict_pd['slantRange'] - predict_pd['predict'])
232 plt.xlabel('Observed Slant Range [m]')
233 plt.ylabel('Subtract Observed - Predict [m]')
234 plt.tight_layout()
235 plt.savefig(path + '_predict.png')
236 plt.close()
237
238 # evaluate
239 def evaluate(self, path=None):
240 print('Evaluate Score')
241 if self.hist_metrics is None:
242 print(' val_loss: {}'.format(self.eval_score))
243 else:
244 print(' val_loss: {}'.format(self.eval_score[0]))
245 for i in range(len(self.hist_metrics)):
```

```
246 print(' {}: {}'.format(self.hist_metrics[i],
    self.eval_score[i+1]))
247 if path is not None:
248 with open(path + '_evaluate.txt', 'w') as f:
249 f.write('Evaluate Score¥n')
250 if self.hist_metrics is None:
251 f.write('val_loss: {}¥n'.format(self.eval_score))
252 else:
253 f.write('val_loss: {}¥n'.format(self.eval_score[0]))
254 for i in range(len(self.hist_metrics)):
255 f.write('{}: {}¥n'.format(self.hist_metrics[i],
    self.eval_score[i+1]))
256
257
258 perser = argparse.ArgumentParser()
259 perser.add_argument('--batch', help='batch size',
    type=int, default=32)
260 perser.add_argument('--epoch', help='epoch', type=int,
    default=1)
261 perser.add_argument('--metrics', help='metrics',
    action='append', default=None)
262 perser.add_argument('--testfile', help='use test file',
    default='dat0.pkl', type=str)
263 perser.add_argument('--name', help='logs, model name',
    default='log', type=str)
264 args = perser.parse_args()
265
266 directory = './pickle/train/'
267 #norm_x = 0.001970208249986172
268 #norm_y = 2433.9076138204205
269 norm_x = 0.00236424989
270 norm_y = 2920.68913658
271
272 log_path = './logs/' + args.name
273 model_path = './models/' + args.name
274
275 train = TrainModel(directory, args.testfile, norm_x,
    norm_y)
276 train.make_model(metrics=args.metrics,
    log_path=log_path, model_path=model_path)
277 train.training(epochs=args.epoch, batch_size=args.batch)
```

ソースコード：make_pickle.py

```
1  import pandas as pd
2  import time
3
4  def mk_pickle(path_x, path_y, path_out):
5  print('read file: {}'.format(path_x))
6  dfx = pd.read_csv(path_x, delim_whitespace=True, header=None)
7  print('read file: {}'.format(path_y))
8  dfy = pd.read_csv(path_y, delim_whitespace=True, header=None)
9
10 df = pd.concat([dfx, dfy], axis=1)
11 del dfx, dfy
12
13 print('write file: {}'.format(path_out))
14 df.to_pickle(path_out)
15
16
17 dataset23_path = [
18 './dataset/dataset0623/dataset0623-stft',
19 './dataset/dataset0623/dataset0623-dst']
20 dataset24_path = [
21 './dataset/dataset0624/dataset0624-stft',
22 './dataset/dataset0624/dataset0624-dst']
23
24
25 #mk_pickle(dataset24_path[0], dataset24_path[1], './pickle/dataset0624.pkl')
26
27
28 print('read')
29 data = pd.read_pickle('./pickle/dataset.pkl')
30 print(data.info())
31
32 data = data.sample(frac=1.0)
33 step = int(data.shape[0]*0.1)
34
35 start = 0
36 end = data.shape[0]
```

```
37  count = 0
38  for i in range(step, end, step):
39  dat = data[start:i]
40  print('save: {}, {}, {}'.format(count, start, i))
41  dat.to_pickle('./pickle/train/dat{}.pkl'.format(count))
42  start = i
43  count += 1
44  if start < end:
45  dat = data[start:]
46  print('save: {}, {}, {}'.format(count, start, end))
47  dat.to_pickle('./pickle/train/dat{}.pkl'.format(count))
```