インストールマニュアル

ロボット DB システム

2 次元グリッドマップ生成・配布

発行日 2021 年 3 月 31 日
公立大学法人会津大学
株式会社東日本計算センター

目　次

1．はじめに 1
1.1．動作環境 2
1.2．使用機器 3
2．システムの導入 3
2.1. MongoDB 3
2.2. PostgreSQL 5
2.2.1．インストール 5
2.2.2．ユーザー情報 DB/テーブル作成と設定 6
2.2.3. 2 次元グリッドマップ DB/テーブル作成と設定 8
2.3. embulk 10
2.4. Pymongo 11
2.5. psycopg2 11
2.6. Pandas 12
2.7. NumPy 12
2.8. ROS 13
2.8.1．リポジトリ登録 13
2.8.2．インストール 13
2.8.3. rosdep の初期化 13
2.8.4. ROS の環境設定 13
2.8.5．ワークスペース作成 14
2.8.6. ROS の環境再設定 14
2.8.7．動作確認 14
2.9. octomap 15
2.10. map_server 15
2.11. PCL 15
2.12. python-pcl 16
2.13. Apache2 17
2.14. Tomcat 19
2.14.1．準備 19
2.14.2．インストール 19
2.14.3．サービスファイル設定 19
2.14.4．起動 20
2.15. Apache と Tomcat の連携 21
2.15.1．プロキシ機能の有効化 21

2.15.2. 仮想ホストの設定 21
2.15.3. Apache の再起動 22
2.16. Spring 23
2.16.1. 準備 23
2.16.2. ダウンロード 23
2.16.3. 日本語化 23
2.16.4. maven のインストール 24
2.16.5. 設定 25
2.16.5.1. Maven 25
2.16.6. プロジェクトファイルのインポート 29
2.16.7. プロジェクトの更新 30
2.16.8. database.properties の設定 31
2.16.9. application.properties の設定 32
2.16.10. RDBSConstants.java の設定 32
2.16.11. War ファイルの生成 33
2.16.11.1. ターゲットディレクトリのクリーン 33
2.16.11.2. War ファイルの生成 33
3. 2 次元グリッドマップ生成 34
3.1. 3 次元点群データのマージ 34
3.2. ROS パッケージ構築 37
3.3. 2 次元グリッドマップ変換 38
4. Web アプリケーションの設定 39
4.1. Web アプリケーション用ディレクトリ作成 39
4.2. ディレクトリ所有者設定 39
4.3. Tomcat シンボリックリンク参照有効化 40
5. Web アプリのデプロイ 41
5.1. war ファイルの設定 41
5.2. シンボリックリンクの設定 41

1. はじめに

ロボット DB システム(以下、本システム)は、異種複数ロボットの内部、外部データ等を LICTiA サーバー上に配置された共通のデータベース(以下、DB)に蓄積し、そのデータが利用できる環境を提供します。本書は、LiDAR が捉えたデータから生成した 3 次元地図データを基に 2 次元グリッドマップに変換し、それを Web 上でダウンロードする機能の環境構築手順を示します。本システムは会津大学 産学連携ロボット研究開発支援事業の一環として開発したものです。

1.1. 動作環境

動作環境一覧は以下のとおりです。

表 1-1 動作環境一覧

環境		バージョン	補足
OS	Ubuntu	18.04 LTS	-
CPU	Intel(R) Core(TM) i7 ファミリー or Intel(R) Xeon(R) CPU E5 ファミリー 且つ 4 コア以上推奨	-	-
メモリ	16GB 以上推奨	-	-
ストレージ	512GB 以上推奨	-	-
開発言語	Python	2.7.17[1]	-
	C++	C++11	-
	Java	8	-
ミドルウェア	MongoDB	4.2 系	-
	PostgreSQL	11 系	-
	ROS	Melodic	-
	Apache	2.4 系	-
	Tomcat	9.0 系	-
依存ライブラリ	embulk	0.9.23	-
	Pymongo	3.11.0	-
	psycopg2	2.8.5	-
	Pandas	0.23.4	-
	Numpy	1.16.4	-
	PCL	1.8.1	-
	Boost	1.65.0	-
	python-pcl	0.3.0rc1	-
	octomap	0.6.5	-
	map_server	1.16.7	-

[1] 本システムは Python3 未検証。よって、依存ライブラリも 2 系に対応したものとなる。

1.2. 使用機器

表 1-2 使用機器一覧

No	機器名称	個数	補足
1	PC	1	-

2. システムの導入

各ライブラリの導入手順を示します。バージョンは本システムの環境構築時点となります。ライブラリによっては最新版もしくは任意の世代までの公開となっている場合があります（本書執筆時点では MongoDB, PostgreSQL のみ）。その際は最新版を導入してください。

2.1. MongoDB[2]

(1) 公式鍵をインポートします。

```
$ wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -
```

(2) MongoDB の list ファイルを作成します。

```
$ echo "deb https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" |
sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
```

(3) パッケージを更新します。

```
$ sudo apt update
```

(4) インストールをします。本システムでは 4.2.9 とします。

```
$ sudo apt install mongodb-org=4.2.9 mongodb-org-server=4.2.9 mongodb-org-shell=4.2.9
mongodb-org-mongos=4.2.9 mongodb-org-tools=4.2.9
```

バージョンアップデートせずに固定したい場合は、以下を実行します。

```
$ echo "mongodb-org hold" | sudo dpkg --set-selections
$ echo "mongodb-org-server hold" | sudo dpkg --set-selections
$ echo "mongodb-org-shell hold" | sudo dpkg --set-selections
$ echo "mongodb-org-mongos hold" | sudo dpkg --set-selections
$ echo "mongodb-org-tools hold" | sudo dpkg --set-selections
```

[2] https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

(5) サービスを起動します。

```
$ sudo service mongod start
```

(6)

サービスの自動起動を有効化します。

```
$ sudo systemctl enable mongod.service
```

(7) MongoDB シェルを起動します。以下のようなメッセージが表示されれば、インストール完了です。

```
$ mongo
MongoDB shell version v4.2.9
connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
：
MongoDB server version: 4.2.9
：
>
```

※：設定ファイルについては、本書では触れませんので、ユーザー様で適切に設定してください。

2.2. PostgreSQL[3]

2.2.1. インストール

(1) PostgreSQL 公式の apt リポジトリを/etc/apt/sources.list.d/pgdg.list に追加します。

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ bionic-pgdg main" >>
/etc/apt/sources.list.d/pgdg.list'
```

(2) 公式鍵をインポートします。

```
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key
add -
```

(3) パッケージを更新します。

```
$ sudo apt update
```

(4) インストールをします。

```
$ sudo apt -y install postgresql-11
```

(5) サービスの自動起動を有効化します。

```
$ sudo systemctl enable postgresql
```

(6) サービスを起動します。

```
$ sudo systemctl start postgresql
```

(7) PostgreSQL シェルを起動します。以下のようなメッセージが表示されればインストール完了です。¥q で切断してください。

```
$ sudo -u postgres psql
psql (11.7 (Ubuntu 11.7-1.pgdg18.04+1))
"help" でヘルプを表示します。
postgres=# ¥q
```

※：設定ファイルについては、本書では触れませんので、ユーザー様で適切に設定してください。

[3] https://www.postgresql.org/download/linux/ubuntu/

2.2.2. ユーザー情報 DB/テーブル作成と設定

(1) postgres ユーザー(OS ユーザー)のパスワードを設定します。すでに設定済みでしたら、次にお進みください。

```
$ sudo passwd postgres
新しい UNIX パスワードを入力してください:
新しい UNIX パスワードを再入力してください:
```

(2) postgres ユーザー(PostgreSQL ユーザー)のパスワードを設定します。

```
$ su - postgres
postgres@pcname:~$ psql
psql (11.7 (Ubuntu 11.7-1.pgdg18.04+1))
"help" でヘルプを表示します。
postgres=# ALTER ROLE postgres WITH PASSWORD '(パスワード)';
```

(3) DB の作成ならびにロール(ユーザー)の作成をします。本書では説明の便宜上、ユーザー情報 DB は rdr_user と命名します。

```
$ su - postgres
$ postgres@pcname:~$ createdb rdr_user
$ postgres@pcname:~$ psql rdr_user
rdr_user=# CREATE ROLE (ロール名) WITH LOGIN PASSWORD '(パスワード)';
```

作成されたことを確認します。確認後、ログアウトします。

```
rdr_user=# ¥l
名前      | 所有者   | エンコーディング | 照合順序     | Ctype(変換演算子) |      アクセス
権限
-----------+----------+------------------+-------------+-------------------+-----------------
------
postgres  | postgres | UTF8             | ja_JP.UTF-8 | ja_JP.UTF-8       |
rdr_user  | postgres | UTF8             | ja_JP.UTF-8 | ja_JP.UTF-8       |
rdr_user=# ¥q
postgres@pcname:~$ exit
```

(4) ユーザー情報 DB のオーナーを(3)で作成したユーザーに変更します。

```
$ sudo -u postgres psql
postgres=# ALTER DATABASE “rdr_user” OWNER TO (ロール名);
ALTER DATABASE
postgres=#
```

(5) (3)の方法でデータベース一覧を表示し、所有者が(4)で指定したロール名であることが確認できます。

(6) ユーザー情報 DB にデータを格納するテーブルを作成するために、rdr_user に切り替えます。

```
postgres=# ¥c rdr_user
データベース "rdr_user" にユーザ "postgres" として接続しました。
rdr_user=#
```

(7) ユーザー情報テーブル users を作成します。すべてワンラインで記述しますが、本書では説明の便宜上データ毎に改行表示します。

```
rdr_user=# CREATE TABLE public.users(
    user_id integer NOT NULL,
    user_name character varying(256),
    password character varying(256),
    PRIMARY KEY (user_id)
);
```

(8) ユーザー情報を users に登録します。本書では予め用意したユーザー情報(CSV 形式)をインポートして登録することとします。なお、password はハッシュ値を保存しますので、ユーザー様でハッシュ化(ユーザー様で定義したパスワード文字列＋ソルト値)します。ハッシュ化アルゴリズムは SHA256 です。ここで定義しましたソルト値はソースコード(詳細は 2.16.10 参照)にも定義します。

```
rdr_user=# ¥COPY users (user_id, user_name, password) from '/home/(ユーザー名)/user.csv' with csv;
```

(9) 作成したテーブルのオーナーを変更します。

```
rdr_user=# ALTER TABLE public.users OWNER to (ロール名);
```

(10) オーナーが変更されたことを確認し、ログアウトします。

```
rdr_user=# ¥dt
                リレーション一覧
 スキーマ | 名前 |     型     |  所有者
----------+------+----------+----------
 public   | users | テーブル | (ロール名)

rdr_user=# ¥q
postgres@pcname:~$ exit
```

2.2.3.2 次元グリッドマップ DB/テーブル作成と設定

(1) DB を作成します。今回はオーナー指定もします。本書では説明の便宜上、2 次元グリッドマップ DB は grid_map_2d と命名します。

```
$ su – postgres
$ postgres@pcname:~$ createdb grid_map_2d -O “(ロール名)”
$ postgres@pcname:~$ psql grid_map_2d
grid_map_2d=#
```

作成されたこと、指定した所有者であることを確認します。確認後、ログアウトします。

```
grid_map_2d=# ¥l
名前      |  所有者  | エンコーディング |  照合順序   | Ctype(変換演算子) |      アクセス
権限
-----------+----------+-----------------+------------+-------------------+-----------------
------
grid_map_2d | (ロール名) | UTF8              | ja_JP.UTF-8 | ja_JP.UTF-8       |
```

(2) 2 次元グリッドマップテーブル maps を作成します。すべてワンラインで記述しますが、本書では説明の便宜上データ毎に改行表示します。

```
grid_map_2d=# CREATE TABLE public.maps(
    robot_id character varying(256), ¥
    location character varying(256), ¥
    datetime timestamp with time zone, ¥
    map_yaml_path character varying(256), ¥
    map_pgm_path character varying(256) ¥
);
```

(3) 作成したテーブルのオーナーを変更します。

```
grid_map_2d=# ALTER TABLE public.maps OWNER to (ロール名);
ALTER TABLE
```

(4) オーナーが変更されたことを確認し、ログアウトします。

```
grid_map_2d=# ¥dt
              リレーション一覧
 スキーマ | 名前 |     型     |  所有者
----------+------+----------+----------
 public   | maps | テーブル | (ロール名)

grid_map_2d=# ¥q
postgres@pcname:~$ exit
```

2.3. embulk[4]

(1) 公式に記載されているとおり、embulk は Java を必要とするため、JRE(Java SE Runtime Environment)もしくは JDK(Java SE Development Kit)をインストールしてください。ここでは OpenJDK 8 をインストールします。

```
$ sudo apt update
$ sudo apt install openjdk-8-jdk
```

(2) Java が複数のバージョンを共存している場合、Java8 に切り替えます。以下コマンド実行後に、2 を入力して Enter キーを押します。

```
$sudo update-alternatives --config java
alternative java (/usr/bin/java を提供) には 2 個の選択肢があります。
選択肢 パス 優先度 状態

＊ 0 /usr/lib/jvm/java-11-openjdk-amd64/bin/java 1111 自動モード
1 /usr/lib/jvm/java-11-openjdk-amd64/bin/java 1111 手動モード
2 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java 1081 手動モード
現在の選択 [*] を保持するには 、さもなければ選択肢の番号のキーを押してください: 2
```

(3) embulk をインストールします。curl がない場合は、先にインストールします。

```
$ sudo apt install curl
$ curl --create-dirs -o ~/.embulk/bin/embulk -L "https://dl.embulk.org/embulk-0.9.23.jar"
$ chmod +x ~/.embulk/bin/embulk
$ echo 'export PATH="$HOME/.embulk/bin:$PATH"' >> ~/.bashrc
$ source ~/.bashrc
```

(4) 本システムでは以下の Output プラグインを使用するので、それぞれインストールします。

```
$ embulk gem install embulk-output-mongodb
$ embulk gem install embulk-output-postgresql
```

[4] https://github.com/embulk/embulk#linux--mac--bsd

2.4. Pymongo[5]

(1) pip でインストールしますので、python と pip をインストールします。

```
$ sudo apt install python
$ curl https://bootstrap.pypa.io/2.7/get-pip.py -o get-pip.py
$ sudo python get-pip.py
```

(2) version(本システムでは 3.11.0)を指定する場合は、以下のように実行します。

```
$ sudo python -m pip install pymongo==3.11.0
```

2.5. psycopg2[6]

(1) 公式に記載されているとおり、以下のライブラリのインストールが前提条件ですので、無い場合はインストールします。

```
$ sudo apt install gcc c++ make
$ sudo apt install python-dev
$ sudo apt install libpq-dev
```

(2) インストールされたかどうかを確認するため、version を確認します。結果が以下のように、インストールした PostgreSQL のバージョンと同一であればインストール成功です。

```
$ pg_config –version
PostgreSQL 11.7 (Ubuntu 11.7-1.pgdg18.04+1)
```

エラーが発生した場合は、パスが通ってないので、以下を実行後、version の確認を再度行います。

```
$ export PATH=/usr/lib/postgresql/11/bin/:$PATH
```

(3) version(本システムでは 2.8.5)を指定する場合は、以下のように実行します。

```
$ sudo pip install psycopg2==2.8.5
```

[5] https://api.mongodb.com/python/current/installation.html

[6] http://initd.org/psycopg/docs/install.html

2.6. Pandas[7]

Python2.7 を使用しますので、version 0.23.4 を指定します[8]。

```
$ sudo pip install pandas==0.23.4
```

2.7. NumPy[9]

Python2.7 を使用しますので、version 1.16 系を指定します。

```
$ sudo pip install numpy==1.16.4
```

[7] https://pandas.pydata.org/

[8] https://pandas.pydata.org/pandas-docs/stable/whatsnew/v0.24.0.html

[9] https://www.scipy.org/scipylib/download.html

2.8. ROS

2.8.1. リポジトリ登録

ROS は apt には無いので、package.ros.org からインストールできるようにするために、リポジトリ(と鍵)を登録します。

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/roslatest.list'
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key F42ED6FBAB17C654
$ sudo apt update
```

2.8.2. インストール

```
$ sudo apt install ros-melodic-desktop-full
$ sudo apt install python-rosinstall
```

2.8.3. rosdep の初期化

rosdep はパッケージが必要とする外部ライブラリやツール等のシステム依存物をダウンロード/インストールするための機能です。

```
$ sudo apt install python-rosdep
$ sudo rosdep init
$ rosdep update
```

2.8.4. ROS の環境設定

ROS 関連の環境変数を自動設定するため、.bashrc に設定を追加します。この設定が終わると、ROS 関連のコマンドが使えるようになります。

```
$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

2.8.5. ワークスペース作成

開発を行うためのワークスペースを準備します。名前や作成場所は任意ですが、一般的には catkin_ws が使われますので、それにあわせた例とします。cakin_make を実行すると、ワークスペース内に必要なファイルが作成されます。

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
$ cd ~/catkin_ws
$ catkin_make
```

2.8.6. ROS の環境再設定

ROS ワークスペースを認識させるため、再度.bashrc に設定を追加します。

```
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

2.8.7. 動作確認

ROS 関連のソフトウェアを動作させるには、roscore コマンドをまず起動する必要があります。ここでは、テストのために起動を確認します。起動すると多くのメッセージが表示されますが、started core service と表示されれば問題はありません。確認後は、Ctrl-C で終了します。

```
$ roscore
```

2.9. octomap

3 次元点群データから 2 次元グリッドマップに変換するため、OctoMap の ROS パッケージ群をインストールします。

```
$ sudo apt install ros-melodic-grid-map
$ sudo apt install ros-melodic-octomap ros-melodic-octomap-rviz-plugins
$ sudo apt install ros-melodic-octomap-mapping ros-melodic-octomap-msgs ros-melodic-octomap-ros ros-melodic-octomap-server
```

2.10. map_server

2 次元グリッドマップ変換結果を保存するために、map_server をインストールします。

```
$ sudo apt install ros-melodic-map-server
```

2.11. PCL[10]

3 次元点群データを扱うためのライブラリです。ダウンロード先は任意です。

```
$ cd (ダウンロード先)
$ wget --no-check-certificate https://github.com/PointCloudLibrary/pcl/archive/pcl-1.8.1.tar.gz
$ tar -xvzf pcl-1.8.1.tar.gz
$ cd pcl-pcl-1.8.1
$ mkdir build && cd build
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make -j2
$ sudo make install
```

[10] https://github.com/PointCloudLibrary/pcl/

2.12. python-pcl

python で点群データを扱うための PCL の Python バインディングです。前提条件として Cython と NumPy が必要になりますが、すでにインストール済みです。

(1) python-pcl を GitHub から clone します。Git がインストールされていなければ、APT コマンドでインストールしてください。clone 先は任意です。

```
$ cd (clone 先のディレクトリ)
$ git clone https://github.com/strawlab/python-pcl.git
$ cd python-pcl
```

(2) インストールにあたり、スクリプト(setup.py)内の VTK ライブラリのバージョンを Ubuntu 18.04 向けに 7.0→6.3 に修正します。修正箇所は python-pcl の issue[11]に記載してありますので、注釈のリンク先で確認して修正してください。執筆時点では setup.py の 726 行目と 752 行目を修正することになります。

(3) ビルドします。

```
$ sudo python setup.py build_ext -i
$ sudo python setup.py install
```

(4) インストールされたことを確認します。インポートできたら、Ctrl-D キーで終了します。

```
$ python
>>> import pcl
>>>
```

[11] https://github.com/strawlab/python-pcl/issues/307

2.13. Apache2

(1) Apache2 をインストールします。

```
$ sudo apt install apache2
```

(2) インストール完了確認を行います。以下のようにインストールした情報が出力されます。

```
$ apache2 -v
Server version: Apache/2.4.29 (Ubuntu)
Server built:    2020-08-12T21:33:25
```

(3) クライアントに不要なサーバー情報を公開しないように設定を変更します。テキストエディタは使い慣れたものを使用してください。以下は nano で編集する場合のコマンドです。

```
$ sudo nano /etc/apache2/conf-available/security.conf
```

表 2 -1 security.conf の変更箇所

行数	変更前	変更後
25	ServerTokens OS	ServerTokens Prod
36	ServerSignature On	ServerSignature Off

(4) ディレクトリインデックスのファイル指定を変更するため、表 2-2 のように編集します。

```
$ sudo nano /etc/apache2/mods-available/dir.conf
```

表 2 -2 dir.conf の変更箇所

行数	変更前	変更後
2	DirectoryIndex ¥ index.html index.cgi ¥ index.pl index.php ¥ index.xhtml	DirectoryIndex ¥ index.html index.php

(5) ServerName とタイムアウト時間(sec)の設定を行うために、apache2.conf を表 2-3 のように編集します。

```
$ sudo nano /etc/apache2/apache2.conf
```

表 2 -3 apache2.conf の設定内容

行数	変更前	変更後
70	-	ServerName UoaServer
86	Timeout 300	Timeout 14400

(6) これまでの設定内容を反映させるために、Apache の再起動を行いましたら設定完了です。

```
$ sudo /etc/init.d/apache2 restart
```

2.14. Tomcat

2.14.1. 準備

Tomcat を実行するための専用ユーザーを作成します。

```
$ sudo useradd -M tomcat
```

2.14.2. インストール

(1) opt ディレクトリへ移動し Tomcat をダウンロードし、解凍してください。本書では 9.0.4 で説明しますが、9.0.4 以降のバージョンでも問題ございません。

```
$ cd /opt/
$ sudo curl -O https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.4/bin/apache-tomcat-9.0.4.tar.gz
$ sudo tar -xzvf /opt/apache-tomcat-9.0.4.tar.gz
```

(2) 展開されたディレクトリの所有者を Tomcat ユーザーに変更します。

```
$ sudo chown -R tomcat:tomcat /opt/apache-tomcat-9.0.4
```

2.14.3. サービスファイル設定

(1) サービスとして起動させるために、service ファイル(ファイル名：tomcat.service)を新規作成します。記述内容は以下のとおりです。

```
$ sudo nano /etc/systemd/system/tomcat.service
```

図 2-1tomcat.service に記載する内容

```
[Unit]
 Description=Apache Tomcat 9
 After=network.target

 [Service]
 User=tomcat
 Group=tomcat
 Type=oneshot
 PIDFile=/opt/apache-tomcat-9.0.4/tomcat.pid
 RemainAfterExit=yes
```

```
ExecStart=/opt/apache-tomcat-9.0.4/bin/startup.sh
ExecStop=/opt/apache-tomcat-9.0.4/bin/shutdown.sh
ExecReStart=/opt/apache-tomcat-9.0.4/bin/shutdown.sh;/opt/apache-tomcat-9.0.4/bin/startup.sh

[Install]
WantedBy=multi-user.target
```

(2) service ファイル作成完了後、パーミッションを設定します。

```
$ sudo chmod 755 /etc/systemd/system/tomcat.service
```

(3) Tomcat を自動で起動させるための設定をします。

```
$ sudo systemctl enable tomcat
```

2.14.4.起動

Tomcat を起動します。Web ブラウザから”http://localhost:8080/”にアクセスし、図 2-2 が出力されればインストールは完了です。

```
$ sudo service tomcat start
```

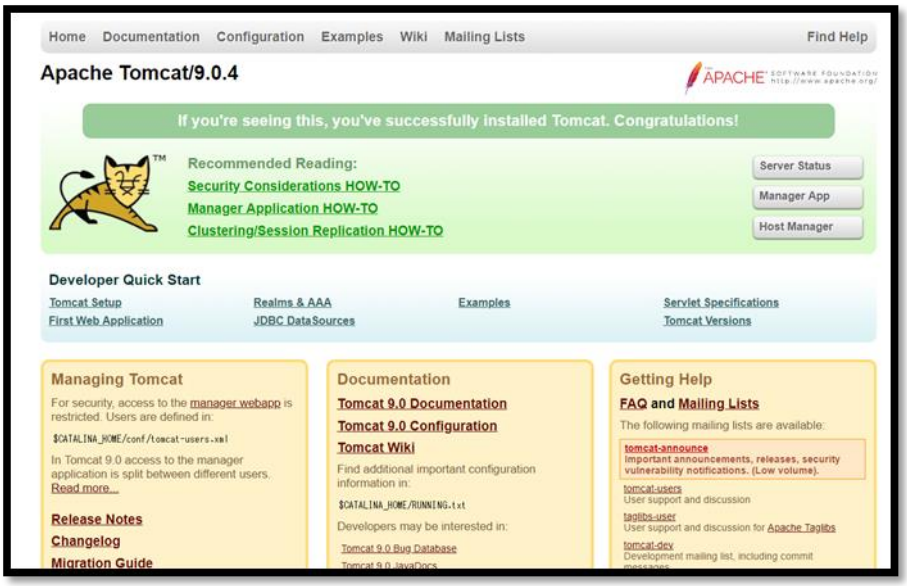

図 2 -2 Tomcat 接続画面

2.15. Apache と Tomcat の連携

現状の設定では Apache と Tomcat が独立して動作するため、リクエストを Apache で待ち受けて Tomcat に渡すように設定を変更します。

2.15.1. プロキシ機能の有効化

プロキシ機能を組み込むため mod_proxy と mod_proxy_ajp のモジュールを有効にします。

```
$ sudo a2enmod proxy
$ sudo a2enmod proxy_ajp
```

2.15.2. 仮想ホストの設定

(1) 仮想ホストの設定ファイルを新規作成します。ファイル名は任意ですが、本書では uoa_service.conf と命名し説明します。図 2-3 のように記述し保存終了します。

```
$ sudo nano /etc/apache2/sites-available/uoa_service.conf
```

図 2 -3 uoa_service.conf に記載する内容

```
<VirtualHost *:80>
        ServerName uoa_service.com
        ServerAdmin admin@uoa_service.com

        <Location />
            ProxyPass ajp://localhost:8009/
            Order allow,deny
            Allow from all
        </Location>
        ErrorLog /var/log/apache2/error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog /var/log/apache2/access.log combined
</VirtualHost>
```

(2) uoa_service.conf のシンボリックを作成し仮想ホストを有効化します。その後、デフォルトで設定されているシンボリックを削除します。

```
$ sudo a2ensite uoa_service
$ sudo a2dissite 000-default.conf
```

2.15.3. Apache の再起動

これまでの設定内容を反映させるために、Apache2 を再起動します。

```
$ sudo /etc/init.d/apache2 restart
```

2.16. Spring

DB の接続設定ファイルの編集や War ファイルの生成に必要な Spring Tools をインストールします。

2.16.1. 準備

本書では説明の便宜上ホームディレクトリ(/home/(ユーザー名))とします。

2.16.2. ダウンロード

(1) 以下のサイトから、Spring Tools 4 for Eclipse の LINUX 64-BIT 版をダウンロードし、そのファイルをホームディレクトリに移動します。本書では ver4.8.0 を用いての説明をします。

https://spring.io/tools

(2) 解凍します。

```
$ tar xvzf spring-tool-suite-4-4.8.0.RELEASE-e4.17.0-linux.gtk.x86_64.tar.gz
```

2.16.3. 日本語化

(1) 以下から Eclipse の日本語化プラグイン(Pleiades)の Linux 版をダウンロードし、そのファイルをホームディレクトリに移動します。

https://mergedoc.osdn.jp/

(2) 解凍します。

```
$ unzip pleiades.zip -d pleiades
```

(3) 展開した pleiades の中にある" features"内の" jp.sourceforge.mergedoc.pleiades"と"plugins"内の" jp.sourceforge.mergedoc.pleiades"ディレクトリを展開した Spring Tool Suite にある"features"と"plugins"にコピーします。

```
$ cp -r pleiades/plugins/jp.sourceforge.mergedoc.pleiades sts-4.8.0.RELEASE/plugins/.
$ cp -r pleiades/features/jp.sourceforge.mergedoc.pleiades sts-4.8.0.RELEASE/features/.
```

(4) sts-4.8.0.RELEASE ディレクトリ内の設定ファイル(SpringToolSuite4.ini)を編集します。ファイル末尾に以下を追記し保存終了します。

```
-Xverify:none
-javaagent:plugins/jp.sourceforge.mergedoc.pleiades/pleiades.jar
```

図 2-4 追記内容

2.16.4. maven のインストール

Java のビルドツールである Maven をインストールします。

```
$ sudo apt install maven
$ mvn -v
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 1.8.0_275, vendor: Private Build, runtime: /usr/lib/jvm/java-8-openjdk-amd64/jre
Default locale: ja_JP, platform encoding: UTF-8
OS name: "linux", version: "5.4.0-60-generic", arch: "amd64", family: "unix"
```

2.1 6.5. 設定

2.1 6.5.1. Maven

2.1 6.5.1.1. インストール

(1) Spring Tool Suite を起動します。

```
$ cd ~/sts-4.8.0.RELEASE
$ ./SpringToolSuite4
```

(2) インストールした Maven を適用するための設定をおこないます。ツールバーの[ウィンドウ]から設定を開きます。Maven の中にあるインストールを選択し、”Maven の起動で使用するインストールの選択”にある”追加”を押下します。

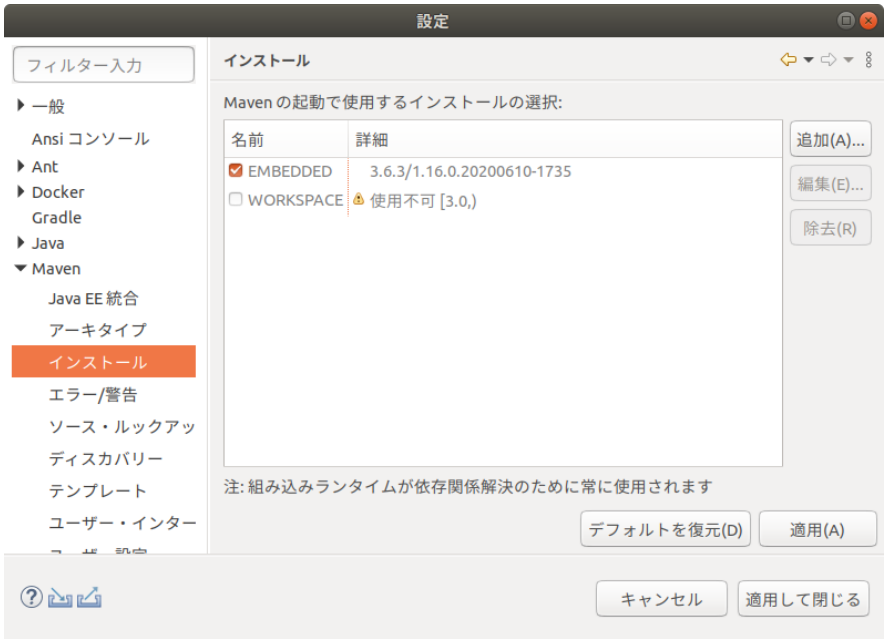

図 2-5 設定(Maven インストール)

(3) 新規 Maven ランタイムウィンドウの“ディレクトリー...”ボタンを押下し、インストールした Maven のパスをインストールホーム設定します。パスは /usr/share/maven です。記述したら、完了ボタンを押下します。

図 2-6 設定(新規 Maven ランタイム)

(4) 追加された maven を選択し、”適用”を押下して設定完了です。

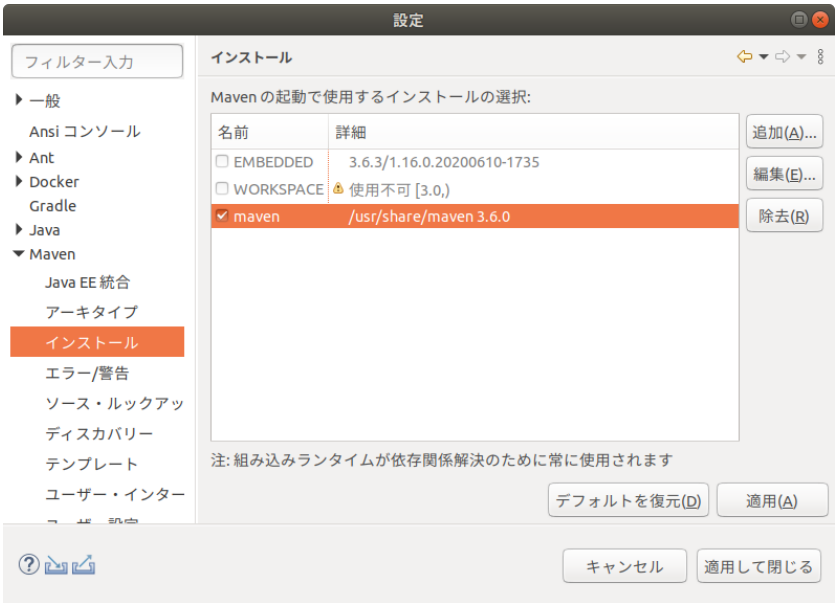

図 2-7 設定(Maven インストール選択)

2.1 6.5.1.2. ユーザー設定

プロキシ環境下でインターネット接続している場合は Maven のユーザー設定を変更する必要があります。

(1) Maven をインストールした際に設定ファイルを格納されているディレクトリ(/etc/maven)にある settings.xml をリネームしバックアップします。

```
$ sudo mv /etc/maven/settings.xml /etc/maven/settings_bk.xml
```

(2) バックアップ後、settings.xml を作成します。

```
$ sudo nano /etc/maven/settings.xml
```

(3) 記述内容を図 9-7 に示します。プロキシ情報は<host>と<port>に記述します。記述完了したら保存終了します。

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                      http://maven.apache.org/xsd/settings-1.0.0.xsd">
<proxies>
    <proxy>
      <id>optional</id>
      <active>true</active>
      <protocol>https</protocol>
      <host>your.proxy.address</host>
      <port>****</port>
    </proxy>
</proxies>
</settings>
```

(4) Spring Tool Suite の設定を開き、Maven にあるユーザー設定を選択します。ユーザー設定にあるグローバル設定およびユーザー設定の参照からそれぞれ先ほどの settings.xml を選択し、適用を押下することで変更内容を反映します。[図 2-8]

図 2-8 設定(Maven ユーザー設定画面)

2.1 6.6. プロジェクトファイルのインポート

(1) ツールバーの[ファイル]からインポートを選択します。

(2) インポート画面が表示されますので、「一般」ディレクトリから「既存プロジェクトをワークスペースへ」を選択し、”次へ”を押下します。

(3) “アーカイブ・ファイルの選択”を選択し、”参照”からプロジェクトを指定します。本書では RDBS と命名されたプロジェクトをインポートします。ファイルを選択後、”完了”を押下するとプロジェクトがインポートされます。

図 2-9 プロジェクトのインポート

2.1 6.7. プロジェクトの更新

(1) インポートしたプロジェクトを右クリックし、[Maven]-[プロジェクトの更新]を選択します。

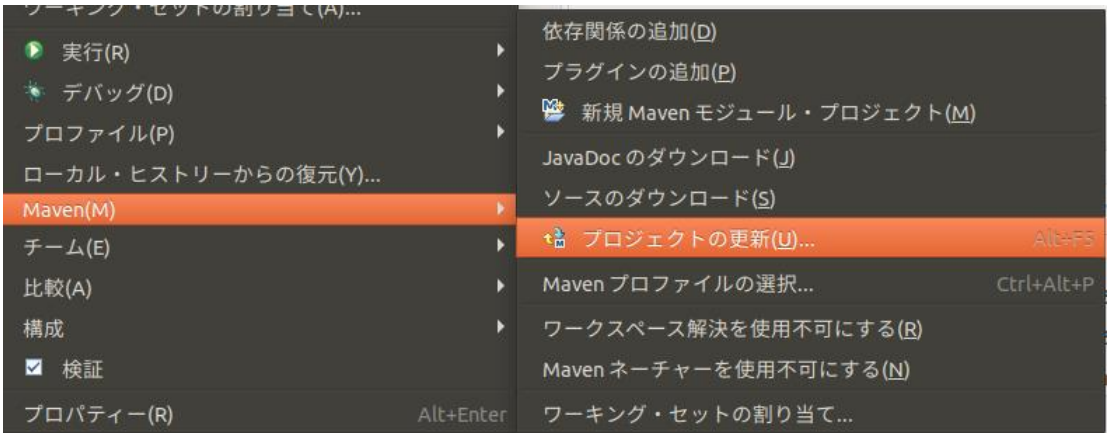

図 2-10 Maven プロジェクトの更新

(2) プロジェクトにチェックがされていることを確認し、OK を押下します。

図 2-11 Maven プロジェクト更新 - 確認

2.1 6.8. database.properties の設定

Spring Tool Suite プロジェクトには DB 接続設定用ファイルがあり、DB やユーザーなどに応じて変更が必要になります。

(1) パッケージ・エクスプローラから RDBS プロジェクトの/src/main/resource にある database.properties を選択して開きます。

(2) database.properties に 2.2.2(3)で作成したユーザー、パスワード、DB 名にそれぞれ記述します。以下はその一例となります。

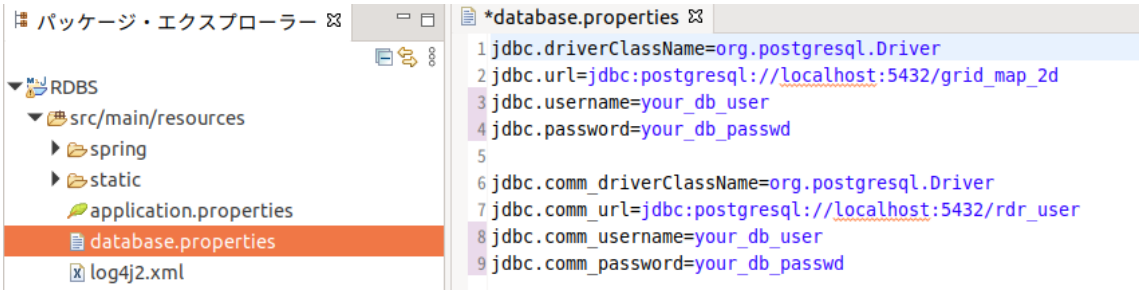

図 2-12 database.properties の設定

表 2-4 database.properties 設定値

項目	詳細	変更要否
jdbc.driverClassName	2 次元グリッドマップ DB へアクセスするためのライブラリ名称	否
jdbc.url	2 次元グリッドマップ DB への URL	DB アドレス/ポート番号を変更している場合は要
jdbc.username	2 次元グリッドマップ DB に接続するユーザーを設定	要
jdbc.password	2 次元グリッドマップ DB に接続する際のパスワードを設定	要
jdbc.comm.driverClassName	ユーザー情報 DB へアクセスするためのライブラリ名称	否
jdbc.comm.url	ユーザー情報 DB への URL	DB アドレス/ポート番号を変更している場合は要
jdbc.comm.username	ユーザー情報 DB に接続するユーザーを設定	要
jdbc.comm.password	ユーザー情報 DB に接続する際のパスワードを設定	要

2.16.9. application.properties の設定

アプリケーション設定用ファイルに、RDR のルートパスとダウンロードパスを設定します。説明の便宜上、ホームディレクトリにデータ格納することで説明しますが、ユーザー様で /home/(ユーザー名)/rdr の部分を適切なパスに設定ください。application.properties は RDBS プロジェクトの/src/main/resource にあります。

*application.properties　RDBSConstants.java

```
1 app.downloadPath=/home/your_user_name/rdbs/web/download/tmp
2 app.rdr_root_path=/home/your_user_name/rdbs
```

図 2-13 application.properties の設定

2.16.10. RDBSConstants.java の設定

17 行目の HASH_SALT の値を定義してください。

2.16.11. War ファイルの生成

Tomcat にデプロイする War ファイルを Spring Tool Suite で生成します。

2.16.11.1. ターゲットディレクトリのクリーン

RDBS プロジェクトを右クリックし、実行から Maven clean を実施します。コンソールに完了が通知されると以前の Maven install で生成されたファイル(プロジェクト内の target ディレクトリ内)が削除されます。

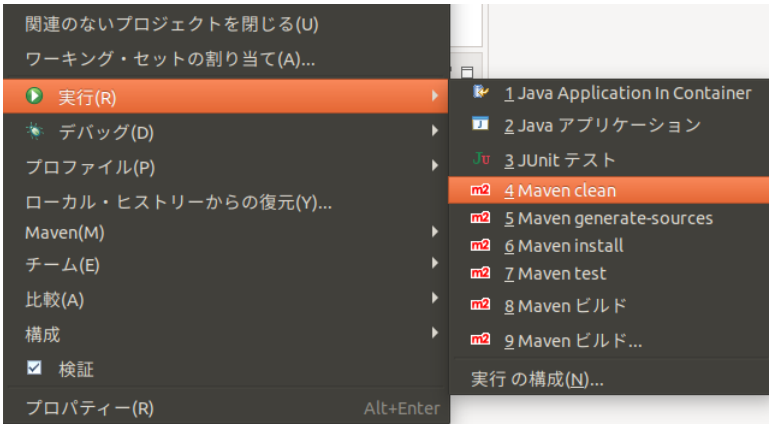

図 2-14 Maven clean

2.16.11.2. War ファイルの生成

プロジェクトを右クリックし、実行から Maven install を実施します。コンソールに完了が通知されるとプロジェクト内の target ディレクトリに War ファイルが生成されます。

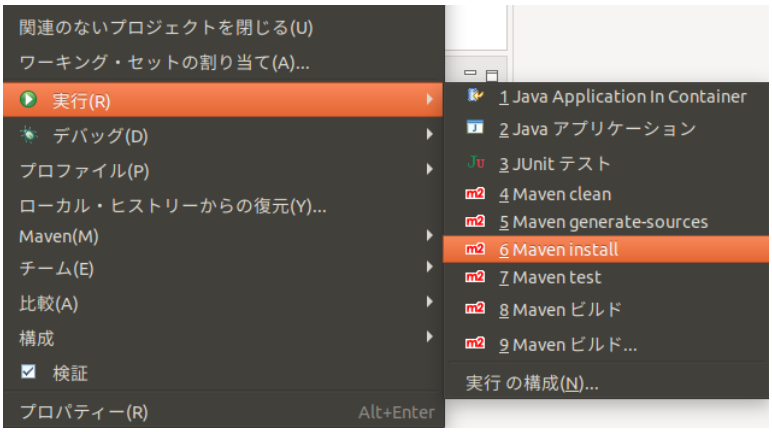

図 2-15 Maven install

3.2 次元グリッドマップ生成

3.1.3 次元点群データのマージ

2 次元グリッドマップの変換にあたり、SLAM(LOAM)後の 3 次元点群データをダウンサンプリングしたあとにマージします。

(1) 入手したソースコードをホームディレクトリに配置します。

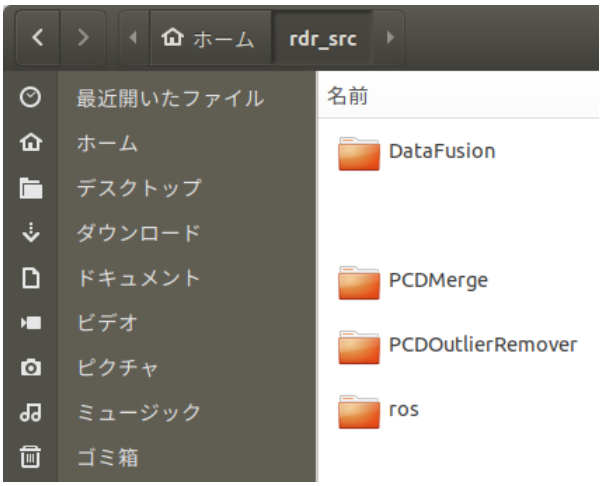

図 3-1 ソースディレクトリ

(2) 対象となる 3 次元点群データのファイルリストを作成します。ファイルは以下に格納されている前提で説明します。

```
$ cd ~/rdr_src/PCDMerge/
$ find ~/rdr/daisha2_1/3d_map/rtf_office/20201010/192800/*.pcd > filenames.txt
```

(3) 必要に応じて、パラメータ設定ファイル(params.ini)を編集します。パラメータ詳細は表 3-1 を参照してください。ここでは 10 ファイル毎にレーザー番号 6 のみの点群データをマージします。

```
[Params]
thin_out_rate=10
trim_method=l
laser_num=6
lidar_h_lower=-0.10
lidar_h_upper=0.10
lower_x=-4.5
lower_y=-2.8
lower_z=-2.5
upper_x=4.2
upper_y=2.0
upper_z=9.5
```

表 3-1 パラメータ詳細

引数	型	説明
thin_out_rate	int	マージ対象 PCD ファイル読込間引き率
trim_method	string	点群データトリミング手法 l：レーザー h：LiDAR 高さ
lower_x/y/z	float	測定環境外の点群データの各方向の下限値
upper_x/y/z	float	測定環境外の点群データの各方向の上限値

(4) ビルドします。

```
$ mkdir build && cd build
$ cmake ..
$ make
```

(5) 実行します。実行引数詳細は表 3-2 を参照ください。

```
$ ./pcd_merge [pcd_filenames_list] [merge_params]
```

表 3 - 2 アプリケーション引数仕様

引数	型	説明
pcd_filenames_list	string	マージ対象 PCD ファイルパスをリストアップしたファイル(txt 形式)を指定 (3.1.3(2)で生成したファイル)
merge_params	string	マージに必要な各種パラメータが定義されたファイル(ini 形式)を指定 (3.1.3(3)で設定したファイル)

(6) マージ出力された 3 次元点群データ merged.pcd を 2 次元グリッドマップに変換するには、座標系カラム位置の修正が必要ですので、以下のように編集/上書きします。

```
$ sed -i s/x y z/y z x/ merged.pcd
```

(7) merged.pcd を任意のディレクトリにコピーします。

```
$ mkdir -p ~/rdr/daisha2_1/merged/rtf_office/20201010/192800/
$ cp merged.pcd ~/rdr/daisha2_1/merged/rtf_office/20201010/192800/.
```

3.2. ROS パッケージ構築

マージした 3 次元点群データを ROS topic として配信するための環境を構築します。

(1) catkin ワークスペースの src ディレクトリに移動し、rdbs という名前でパッケージを作成します。

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg rdbs sensor_msgs std_msgs rospy
```

(2) src ディレクトリ内に rdbs/launch ディレクトリを作成し、rdr_src 内の ROS の launch ファイルをコピーします。

```
$ mkdir -p ~/catkin_ws/src/rdbs/launch
$ cd ~/catkin_ws/src/rdbs/launch
$ cp ~/rdr_src/ros/launch/pointcloud_to_gridmap.launch .
```

(3) ビルドします。

```
$ cd ~/catkin_ws
$ catkin_make
```

3.3.2 次元グリッドマップ変換

(1) ターミナル(Ctrl+Alt+T)を起動し、別タブを 3 つ起動します。1 つ目のタブで roscore を実行します。

```
$ gnome-terminal --tab --tab --tab
$ roscore
```

(2) 2 つ目のタブで、3.3.1(5)の 3 次元点群データを topic として配信します。説明の便宜上、改行表示していますが、実際はワンラインです。

```
$ rosrun pcl_ros ¥
pcd_to_pointcloud ~/rdr/daisha2_1/merged/rtf_office/20201010/192800/merged.pcd 0.5 ¥
_frame_id:=/base_footprint cloud_pcd:=/pcmap
```

(3) 3 つ目のタブで、2 次元グリッドマップ変換の launch ファイルを実行します。

```
$ cd ~/catkin_ws/src/rdbs/launch
$ roslaunch rdbs pointcloud_to_gridmap.launch
```

(4) 4 つ目のタブで、指定したパスに 2 次元グリッドマップを生成/保存します。

```
$ mkdir -p ~/rdr/daisha2_1/2d_grid_map/rtf_office/20201010/192800/
$ cd ~/rdr/daisha2_1/2d_grid_map/rtf_office/20201010/192800/
$ rosrun map_server map_saver
```

(5) 新たにターミナルを起動し、2 次元グリッドマップ DB にマップを登録します。以下は入力例となりますので、実際のロボット ID/ロケーション/日付と(4)のファイルパスを登録します。説明の便宜上、改行表示していますが、実際はワンラインです。

```
$ su – postgres
$postgres@usename:~$ psql grid_map_2d
grid_map_2d=# insert into public.maps(¥
robot_id, location, datetime, map_yaml_path, map_pgm_path) ¥
values ¥
('daisha2_1', 'rtf_office', '2020-10-10 19:28:00', ¥
'/daisha2_1/2d_grid_map/rtf_office/20201010/192800/map.yaml', ¥
 '/daisha2_1/2d_grid_map/rtf_office/20201010/192800/map.pgm');
```

4. Web アプリケーションの設定

システムに必要な環境構築が完了しましたら、Web アプリケーションの設定を実施します。本システムは説明の便宜上、以下のようにしますが、ユーザー様で適切に構成ください。

```
/ホームディレクトリ/
  └rdr/
    └daisha2_1/
      └2d_grid_map/                ← 2 次元グリッドマップ
        └(ロケーション)/(年月日)/(時分秒)/
          └map.pgm
          └map.yaml
      └3d_map/                     ← SLAM 後の 3 次元点群データ
        └(ロケーション)/(年月日)/(時分秒)/
          └unixtime.pcd
                :
      └merged/                     ← マージした 3 次元点群データ
        └(ロケーション)/(年月日)/(時分秒)/
          └merged.pcd
    └web/
      └download/                   ← 地図ダウンロード用の Zip ファイルを格納
```

4.1. Web アプリケーション用ディレクトリ作成

システムに必要なデータ格納ディレクトリ等のディレクトリを作成します。マップ系のディレクトリは今までに作成済みですので、ここでは、download ディレクトリのみを作成します。ディレクトリのパーミッションは必要に応じて適切に設定してください。

```
$ mkdir -p ~/rdr/web/download/tmp
```

4.2. ディレクトリ所有者設定

web ディレクトリの所有者を Tomcat 専用ユーザーに変更します。

```
$ sudo chown -R tomcat:tomcat ~/rdr/web/
```

4.3. Tomcat シンボリックリンク参照有効化

Tomcat のシンボリックリンク参照設定を有効にします。context.xml ファイルを開き <Context> タグの下に以下 1 行を挿入して保存終了します。

```
$ sudo nano /opt/apache-tomcat-9.0.4/conf/context.xml
```

表 4-1 追記する文章

```
<Resources className="org.apache.catalina.webresources.StandardRoot"
allowLinking="true"/>
```

5. Web アプリのデプロイ

5.1. war ファイルの設定

(1) Tomcat のサービスを停止します。

```
$ sudo service tomcat stop
```

(2) RDBS.war ファイルを Tomcat の webapps ディレクトリにコピーし、パーミッションの設定をします。本書では便宜上ユーザーディレクトリに配置されているものとします。

```
$ cd ~
$ sudo cp RDBS.war /opt/apache-tomcat-9.0.4/webapps/.
$ sudo chmod 755 /opt/apache-tomcat-9.0.4/webapps/RDBS.war
```

(3) Tomcat のサービスを開始します。

```
$ sudo service tomcat start
```

(4) 別タブを起動し、Tomcat のユーザーで war ファイルが展開されていることを確認します。（RDBS ディレクトリの更新日時で確認してください。）

```
$ sudo su - tomcat
$ ls -l /opt/apache-tomcat-9.0.4/webapps/RDBS
```

5.2. シンボリックリンクの設定

(1) Tomcat のサービスを停止します。

```
$ sudo service tomcat stop
```

(2) Tomcat のユーザーでシンボリックリンクを作成します。リンク元はデータ格納元となります。本書の例では以下ですが、ユーザー様の環境に読み替えてください。

```
$ sudo su – tomcat
$ ln -s /your_home_dir/rdr/web ¥
  /opt/apache-tomcat-9.0.4/webapps/RDBS/resources/data
```

(3) Tomcat のサービスを開始します。

```
$ sudo service tomcat start
```

著作権

本文書の著作権は公立大学法人 会津大学に帰属します。
この文書のライセンスは以下のとおりです。
クリエイティブ・コモンズ 表示 2．1日本
http://creativecommons.org/licenses/by/2.1/jp/