ユーザーズマニュアル

実空間可視化システム

発行日 2025 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. 実空間可視化システムとは

実空間可視化システム(Real Space Visualization System)は、外部カメラから取得したキャプチャを基に、動的な対象（人、ロボット、台車など）および準静的な物体（机、椅子など）の位置情報を推定したデータを仮想空間（Choreonoid などのロボットシミュレータ）のモデルデータに反映させ、リアルタイムで更新するシステムです（図 1-1)。

本書はこの「実空間可視化システム」のユーザーマニュアルです。上記は会津大学 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。

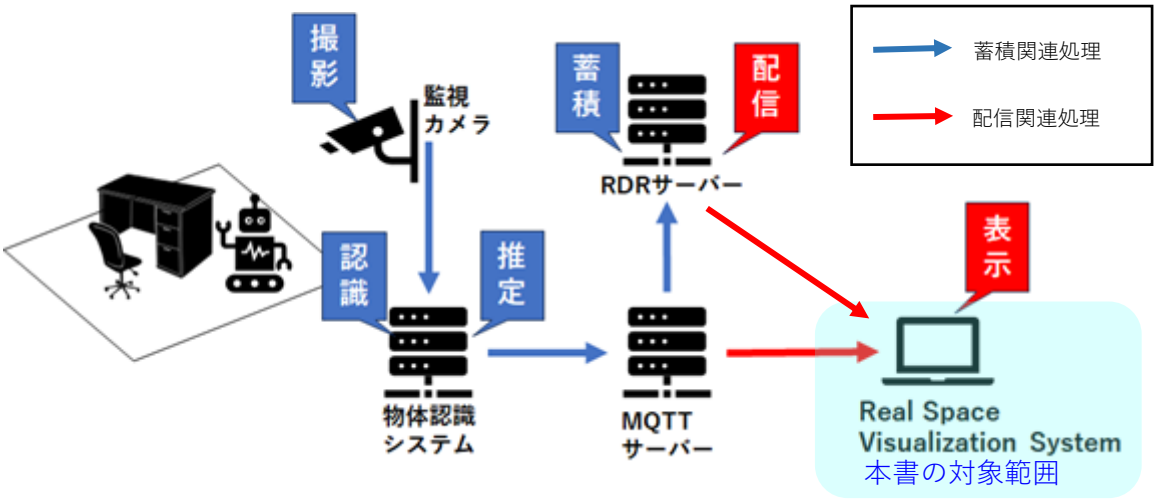

図 1-1 実空間可視化システム構成一例

1.2. 動作環境

動作環境一覧を表 1-1 に示します。

表 1-1 動作環境一覧

環境		バージョン	補足
OS	Ubuntu	20.04 LTS	-
CPU	Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz	-	-
GPU	-	-	-
メモリ	16GB 以上	-	-
ストレージ(SSD)	512GB 以上	-	-
開発言語	Python	3.8 系	-
ミドルウェア	Choreonoid	c6d0afc6f0dc677cb732488bd983e9efd004bca2	コミット ID
	ROS	Noetic	-
依存ライブラリ	paho-mqtt	1.6.1	MQTT Python クライアント
	requests	2.22.0	HTTP 向け Python ライブラリ
	numpy	1.22.4	数学関数ライブラリ
	numpy-quaternion	2023.0.2	クォータニオン用ライブラリ

1.3. 前提事項/注意事項

導入にあたっての前提ならびに注意事項を表 1-2 に示します。

表 1-2 前提ならびに注意事項

前提事項	(1) インストールマニュアル_実空間可視化システムに沿って、動作環境構築済みであること (2) MQTT Broker が起動していること (3) 物体認識システムが起動していること (4) RDR 上の RDBS が起動していること
注意事項	無し

1.4. 関連資料

関連資料を表 1-3 に示します。

表 1-3 関連資料

No	資料名	備考
1	インストールマニュアル_実空間可視化システム	-

2.動作手順

2.1.ディレクトリ構成

Real_Space_Visualization_System のディレクトリ構成を図 2-1 に示します。

```
Real_Space_Visualization_System      -----  ルートディレクトリ
├ tls                                -----  SSL サーバ証明書格納ディレクトリ
|   ├ ca.crt                        -----  認証局の証明書
|   ├ client.key                    -----  クライアント証明書の秘密鍵
|   └ client.crt                    -----  クライアント証明書
├ cnoid_item_base_list.yaml          -----  モデルデータ配置用基準ファイル
├ create_yaml.py                     -----  YAML ファイル作成処理クラス
├ mqtt_interface.py                  -----  MQTT 通信処理クラス
├ object_setting.py                  -----  物体表示事前設定クラス
├ object_viewer.py                   -----  物体表示メインクラス
├ object_viewer_controller.py        -----  物体表示操作クラス
├ rearrangement_obstacles.py         -----  モデルデータ配置処理クラス
├ socket_server.py                   -----  ソケット通信処理クラス
├ location_conf.json                 -----  ロケーション、モデル設定ファイル
└ mqtt_conf.json                     -----  MQTT サーバー接続設定ファイル
```

図 2-1 ディレクトリ構成

2.2. ロケーション、モデル設定

ロケーション、モデル設定ファイル仕様として、ファイル形式は JSON、改行コード: LF、文字コードは UTF-8 とします。パラメータを表 2-1、記述例を図 2-2 に示します。

表 2-1 ロケーション、モデル設定ファイル仕様

項目	型	説明
location	string	ロケーション名を指定(“lictia_1f” of “nagato_ic”)
field_body	dict	フィールドの body 設定項目
name	string	body 名
translation	array	配置する座標[x, y, z]
rotation	string	配置する姿勢(オイラー角) [x, y, z]
url	string	body ファイルの格納パス
static_bodys	array	静的物体の body 設定項目
name	string	body 名
translation	string	配置する座標[x, y, z]
rotation	string	配置する姿勢(オイラー角) [x, y, z]
url	string	body ファイルの格納パス
semi_static_bodys	array	準静的物体の body 設定項目
name	string	body 名
height	double	物体の高さ
url	string	body ファイルの格納パス
obstacle_bodys	array	動的物体の body 設定項目
name	string	body 名
height	double	物体の高さ
url	string	body ファイルの格納パス

```
{
  "location": "lictia_1f",
  "field_body": {
    "name": "LICTiA1F",
    "translation": [8.997, -1.691, -0.05],
    "rotation": [0, 0, 90],
    "url": "${SHARE}/LICTiA/model/LICTiA1F.body" },
   "static_bodys": [
    { "name": "DigitalSignage_47v",
      "translation": [0.6695,1.248, 0.952],
      "rotation": [0, 0, 180],
      "url": "${SHARE}/LICTiA/model/DigitalSignage_47v.body"}],
  "semi_static_bodys": [
    { "name": "table",
      "height": 0.7,
      "url": "${SHARE}/LICTiA/model/Table-120x120.body"},
    { "name": "chair",
      "height": 0.4,
      "url": "${SHARE}/LICTiA/model/Office_chair.body"}],
  "obstacle_bodys": [
    { "name": "person",
      "height": 0.6,
      "url": "${SHARE}/LICTiA/model/Person.body"},
    { "name": "megarover",
      "height": 0.05,
      "url":
"${SHARE}/cnoid_turtlebot_bringup/model/Waffle_pi_LiDAR_2d.body"},
    { "name": "trolley",
      "height": 0.2,
      "url": "${SHARE}/LICTiA/model/Trolley.body"}]
}
```

図　2-2 ロケーション、モデル設定ファイル記述例

2.3. MQTT サーバー接続設定

MQTT サーバー接続設定ファイル仕様として、ファイル形式は JSON、改行コード: LF、文字コードは UTF-8 とします。パラメータを表 2-2、記述例を図 2-3 に示します。

表 2-2 MQTT サーバー接続設定ファイル仕様

項目	型	説明
host	string	ホスト名(もしくは IP アドレス)を指定
port	integer	ポート番号を半角数字で指定
ca_certs	string	CA 認証局ファイルのパスを指定
certfile	string	クライアント証明書のパスを指定
keyfile	string	クライアント秘密鍵のパスを指定

```
{
    "host": "localhost",
    "port": 8080,
    "ca_certs": "./tls/ca.crt",
    "certfile": "./tls/clt.crt",
    "keyfile": "./tls/clt.key"
}
```

図 2-3 MQTT サーバー接続設定ファイル記述例

2.4. Choreonoid 起動

(1) 1 つ目のターミナルで ROS マスターを起動します。

```
$ roscore
```

(2) 2 つ目のターミナルで Real_Space_Visualization_System のディレクトリに移動したのち Choreonoid を起動します。

```
$ cd ***/***/Real_Space_Visualization_System
$ rosrun choreonoid_ros choreonoid
```

2.5. スクリプトの読み込み

(1) Choreonoid 画面のファイル→読み込み→Python スクリプトを選択し、object_viewer.py を読み込んでください。

(2) 読み込まれたスクリプトはアイテムに表示されます。

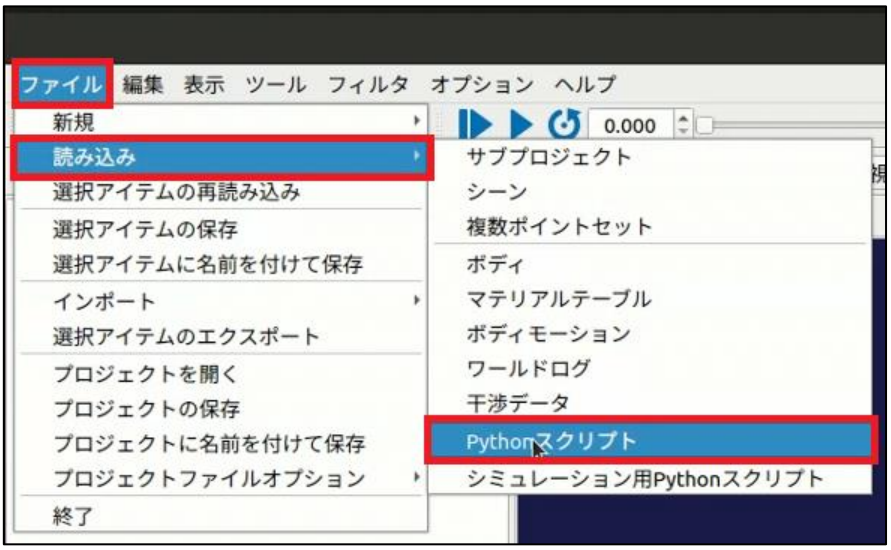

図 2-4 スクリプト読み込み

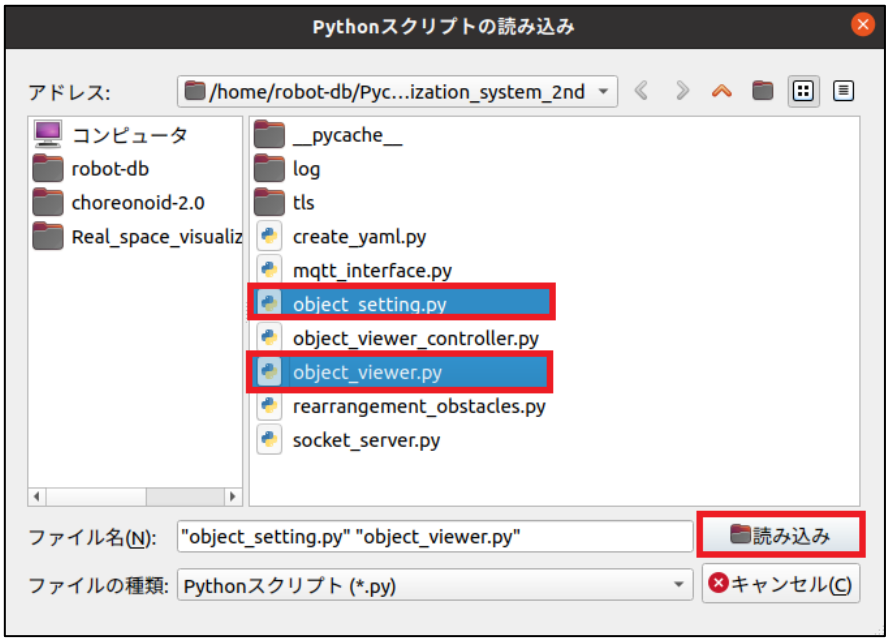

図 2-5 スクリプト読み込み

2.6. スクリプトの実行(セッティング)

(1) 読み込んだ object_setting.py スクリプトを選択し、右クリックから実行を押下して下さい。

図 2-6 スクリプト実行

(2) スクリプトを実行するとシーンに設定ファイルの field_body と static_bodys に設定したモデルが表示されます。表示後、視点をお好みの位置に変更して下さい。

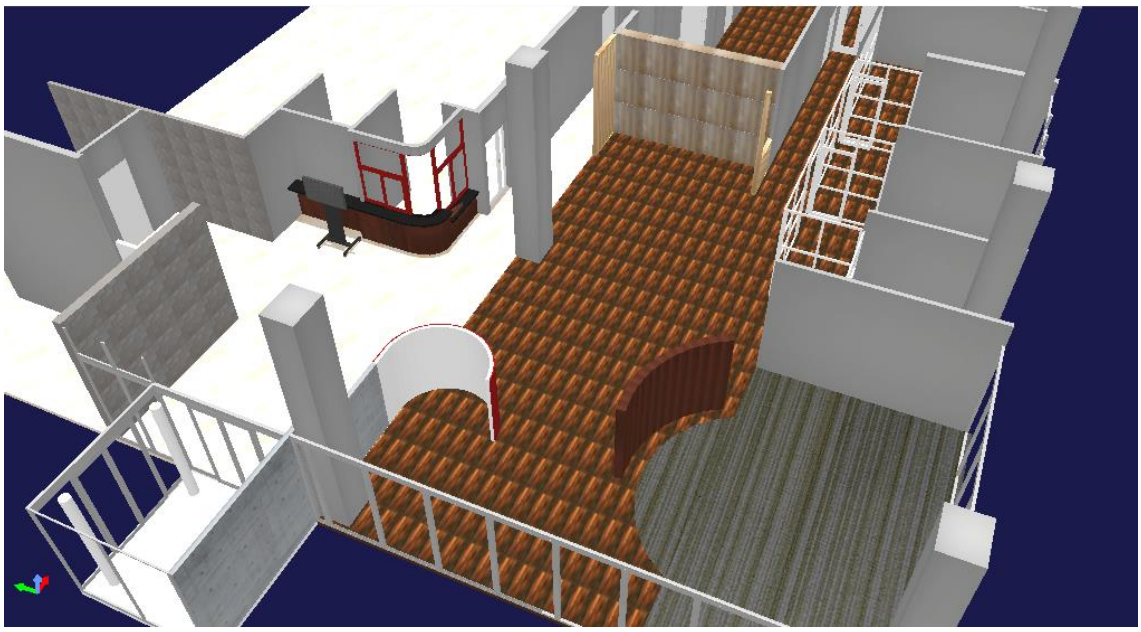

図 2-7 スクリプト実行しモデルが出力された状況

2.7. スクリプトの実行(実行)

(1) 読み込んだ object_viewer.py スクリプトを選択し、右クリックから実行を押下して下さい。

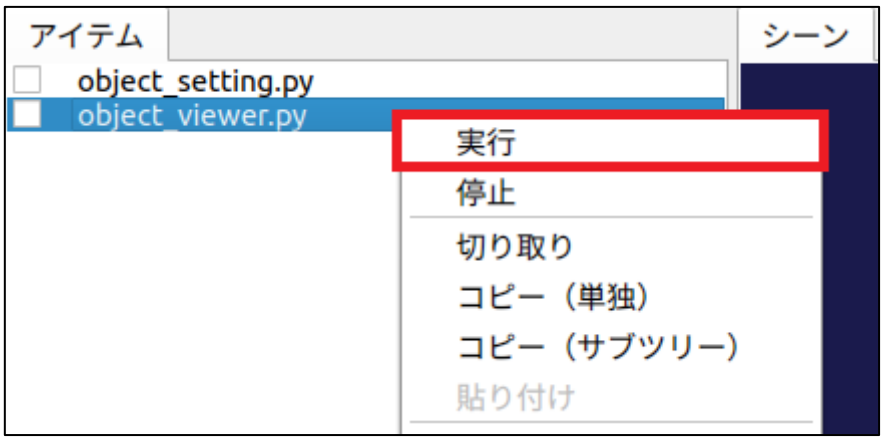

図 2-8 スクリプト実行

(2) スクリプトを実行するとシーンに物体認識システムで認識された物体が表示されます。

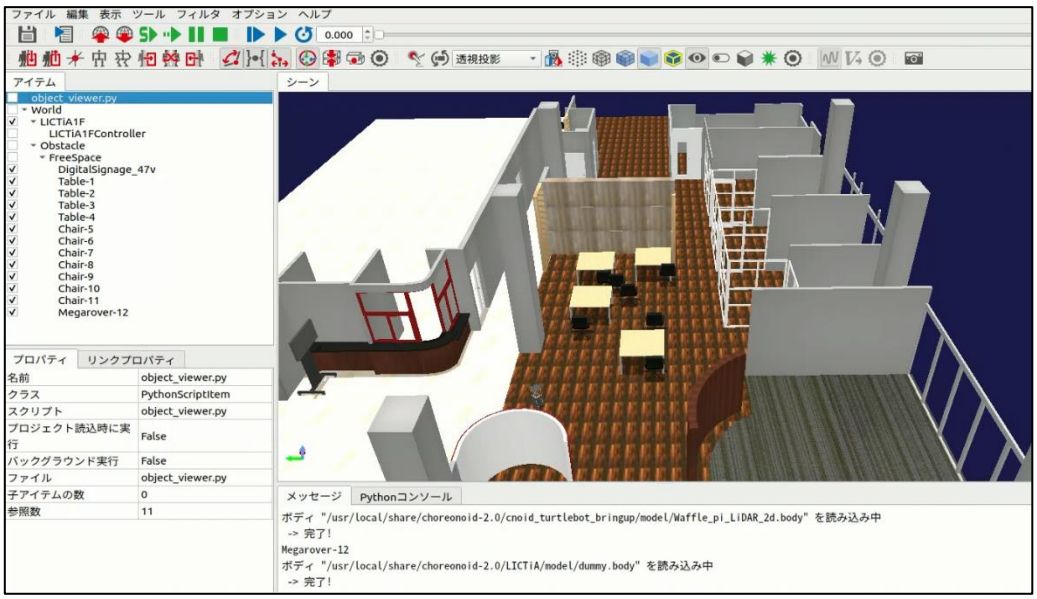

図 2-9 認識した物体がシーンに表示されている状況

2.8. スクリプトの終了

スクリプト実行中は Choreonoid を操作することが出来ません、スクリプトを終了する場合はコンソールから以下のコマンド実行して下さい。

(1) スクリプトの終了

```
$ cd ***/***/Real_Space_Visualization_System
$ python object_viewer_controller.py    'stop'
```

2.9. スクリプトの再実行

(1) スクリプト終了後、再度 object_viewer.py スクリプトを選択し、実行を行うとスクリプトが再度実行されます。

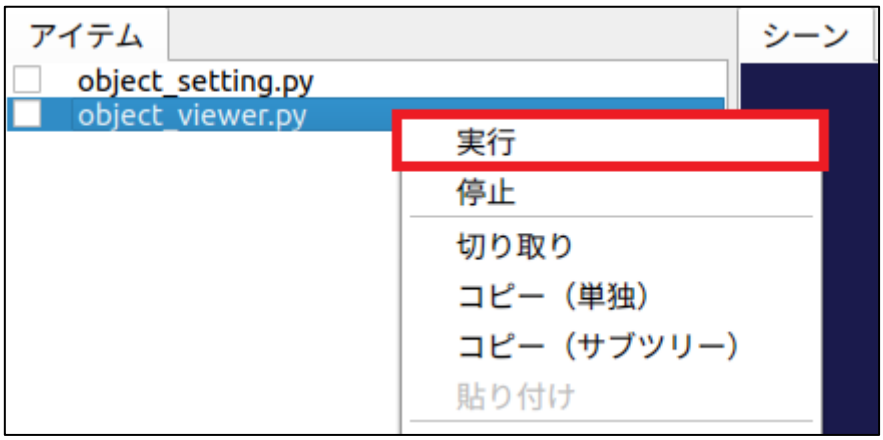

図 2-10 スクリプト実行

2.10. Choreonoid 終了

スクリプト終了後、Choreonoid 動作中に画面右上の[X]ボタンを押下またはファイル→終了を選択すると”現在のプロジェクトは保存されていません。プロジェクトを閉じる前に保存しますか？”とポップアップが表示され、「無視」を押下すると終了します。

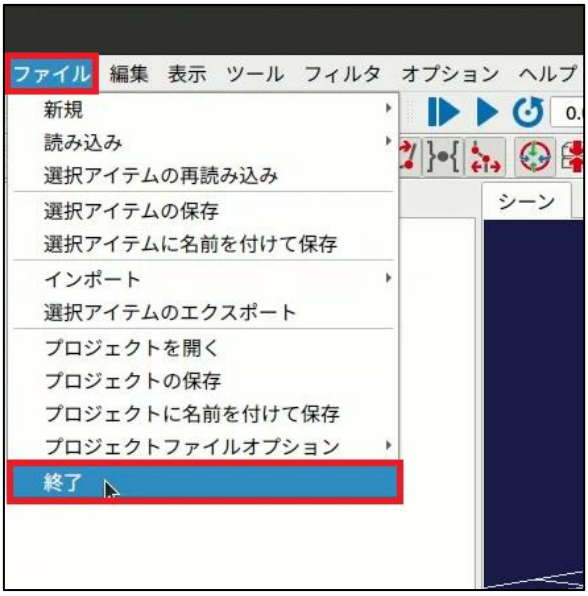

図　2-11 Choreonoid 終了

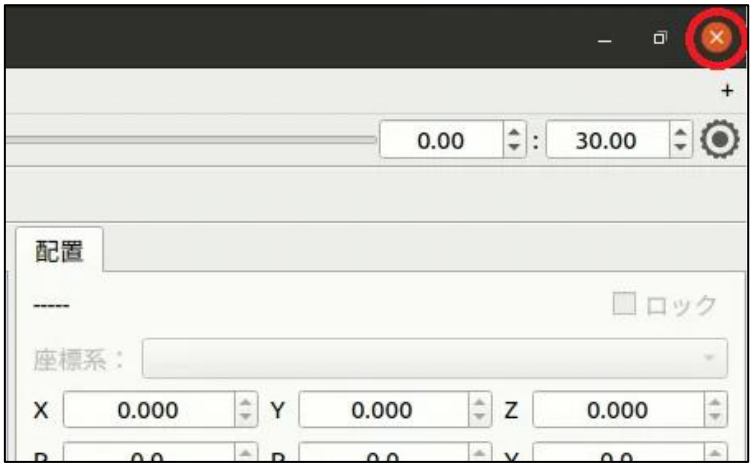

図　2-12 Choreonoid 終了

3. エラーメッセージ

エラー発生時のメッセージを以下に示します。

表 3-1 エラーメッセージ一覧

No	状態	エラーメッセージ
1	MQTT 接続エラー	Unable to connection for MQTT server.
2	設定ファイル読込エラー	Unable to read setting file.
3	物体更新時エラー	Unable to update

4. 注意事項

表 4-1 注意事項一覧

No	内容
1	スクリプト実行中は画面操作は実行できません、スクリプト操作の終了を実施後に画面操作を実施して下さい。
2	スクリプト実行後にモデルデータを削除し、再度スクリプトを実行すると、うまく動作しない場合があります。その際は Choreonoid を再起動じ、再度スクリプトを実行して下さい。

著作権