2023 年 3 月 14 日
機能ロボット学研究室(MA-1)

近接点地図構築ソフトウェアの環境構築と使用方法

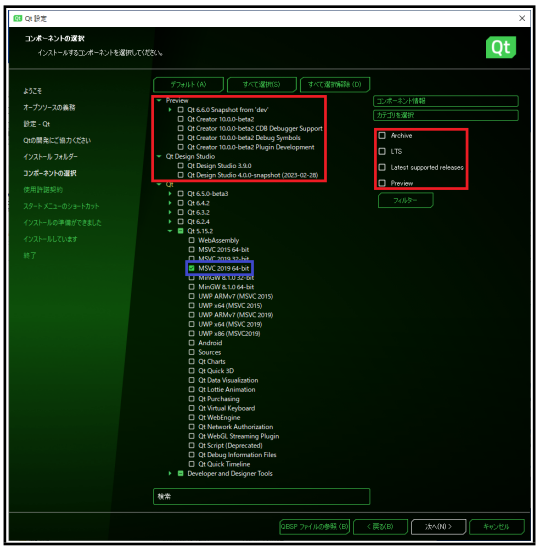
M2 和田鼓太郎

1 Visual Studio Community 2019 のインストール

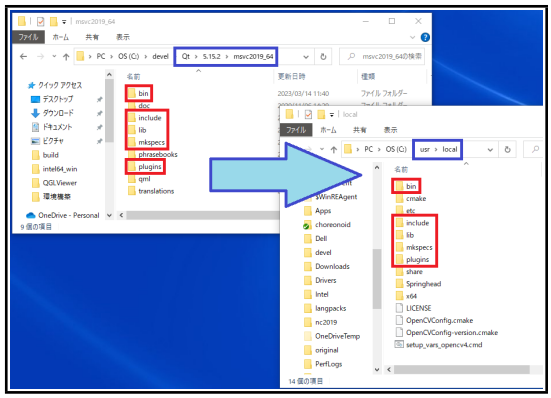
Qt が MSVC2019 か MSVC2015 としか互換性が無いようで、G2O のインストールに必要になる．インストール方法は省略.

2 Qt5 のインストール

1. `https://www.qt.io/` で Qt Account を登録
2. `https://www.qt.io/download-open-source` からオンラインインストーラをダウンロード
3. インストーラを起動し，カスタムインストールを選択
4. コンポーネントを選択
   i 全てのチェックボックスのチェックを外す
   ii `Qt` → `Qt 5.15.2` → `MSVC2019 64-bit` を選択
5. インストール
6. `Qt/5.15.2/msvc2019_64` フォルダ内にある `bin`, `include`, `lib`, `mkspecs`, `plugins` フォルダを `usr/local` にコピー



(a) Components

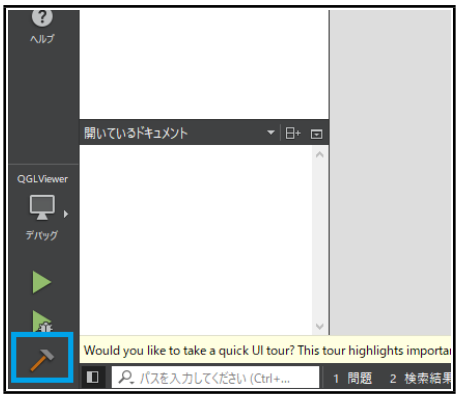


(b) Copying

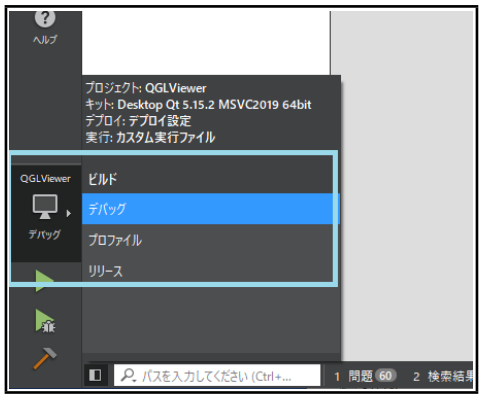
Fig. 1: Installing Qt5

3　QGLViewer のインストール

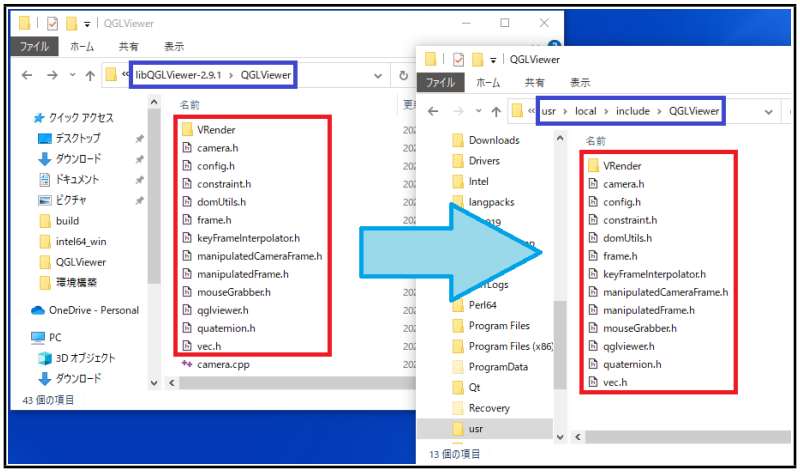
1. `http://libqglviewer.com/installWindows.html` からソースファイルをダウンロードし解凍
2. Qt Creator を起動 (Qt5 と一緒にインストールされている)
3. Qt Creator のバージョン, キットが `5.15.2 MSVC2019 64bit` になっていることを確認
4. Qt Creator で `QGLViewer/QGLViewer.pro` を開く
5. 左下のトンカチマークを押し，Debug モードでビルドを行う
6. 左下のモニタマークを押し，Release モードに変更
7. 左下のトンカチマークを押し，Release モードでビルドを行う
8. Qt Creator を終了する
9. `QGLViewer` フォルダ内のヘッダファイルを `usr/local/include/QGLViewer` にコピー
   (`VRender` フォルダ内のヘッダファイルも参照関係が保たれるようにコピー)
10. `QGLViewer` フォルダ内の `lib`, `dll`, `prl` ファイルを `usr/local/lib` にコピー
11. `QGLViewer` フォルダ内の `dll` ファイルを `usr/local/bin` にコピー



(a) Debug Build



(b) Changed to Release

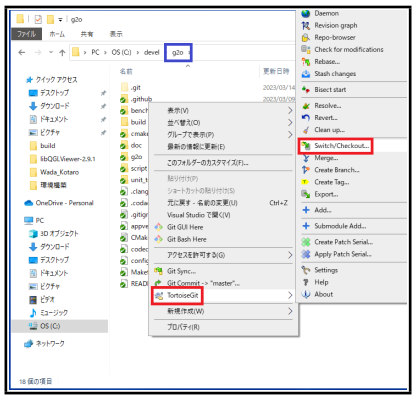


(c) Copying

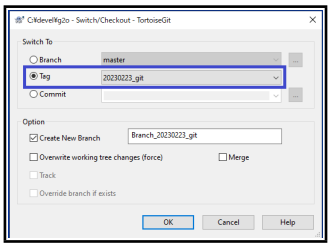
Fig. 2: Installing QGLViewer

4　G2O のインストール

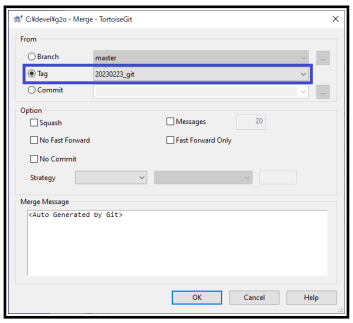
1. git リポジトリ `https://github.com/RainerKuemmerle/g2o.git` をクローン
2. G2O をタグ `20230223_git` でチェックアウトしマージ
3. git リポジトリ `https://github.com/WadaKotaro/prox_g2o.git` をクローン
4. prox_g2o のパッチ `prox_g2o/0001-Pose-graph-optimization-using-Proximity-Point-Pair-m.patch` を G2O に適用する
5. Cmake-Gui で G2O を Build (Cmake-Gui の使用方法は田崎先生が作成したマニュアル**ソフトウェア開発環境の構築方法**`.pdf` を参照すること)
   - `CMAKE_INSTALL_PREFIX` : `C:/usr/local`
   - `CMAKE_PREFIX_PATH` : `C:/usr/local`
   - エントリ `G2O_BUILD_...` 系は `G2O_BUILD_APPS`, `G2O_BUILD_SLAM2D_TYPES` 以外不要なので，チェックを外しても良い
   - Qt5, QGLVIEWER 関連のエントリを適切に設定
   - SuiteSparse 関連は無視してよい (Configure 中に警告を出すが無視しても正常に機能する)
6. Visual Studio 2019 を用いて，生成されたソリューションファイル `g2o/build/g2o.sln` を Debug, Release モードで ALL_BUILD し INSTALL



(a) Checkout

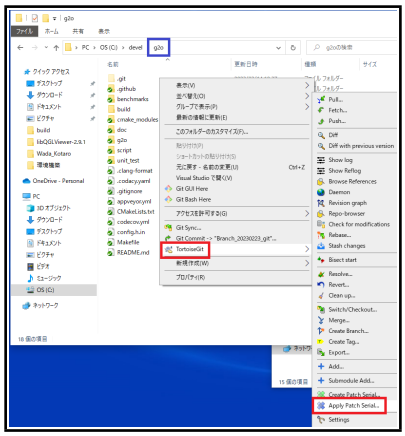


(b) Tag : 20230223_git



(c) Merge : 20230223_git

Fig. 3: Checkout G2O



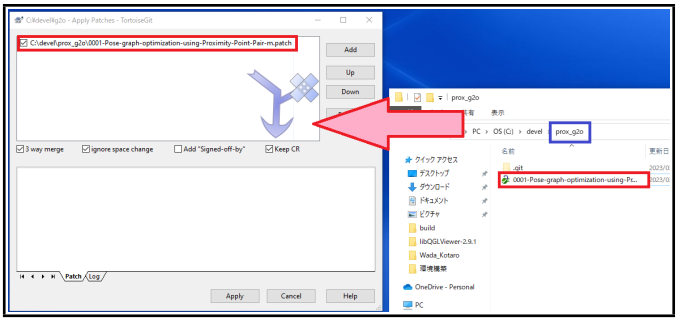


Fig. 4: Apply patch to G2O

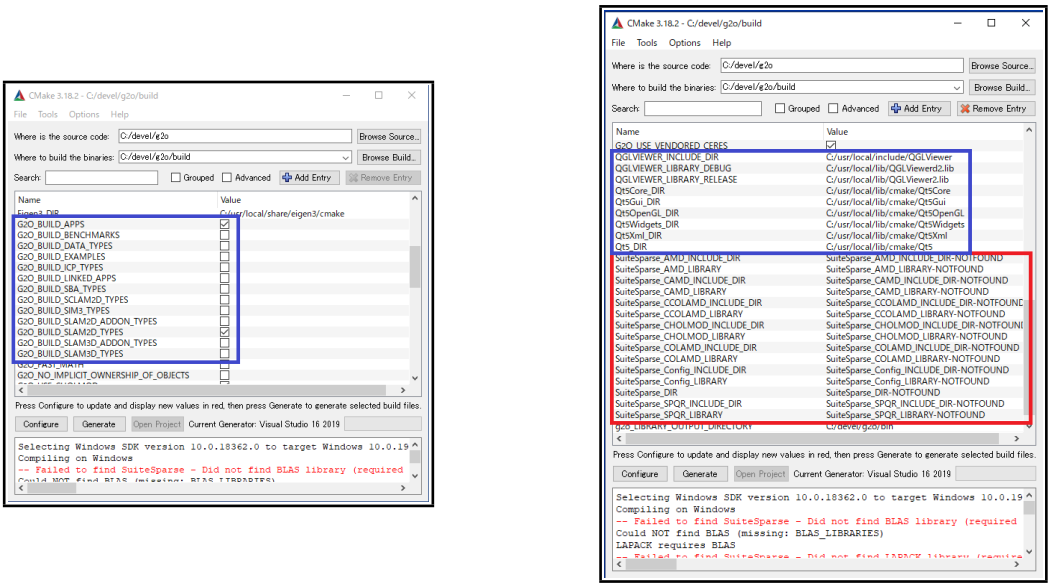


Fig. 5: Entries for build G2O

5 MapConstructor のインストール

1. git リポジトリ `https://github.com/WadaKotaro/MapConstructor.git` をクローン
2. Cmake-Gui で Build
   - `CMAKE_INSTALL_PREFIX` : `C:/usr/local`
   - `CMAKE_PREFIX_PATH` : `C:/usr/local`
   - Boost, expat, Eigen3, GLEW, GLUT(freeglut), MKL, Scenebuilder, Springhead, g2o 関連のエントリを適切に設定
3. Springhead のマクロ `isnan` と Eigen3 のクラス `isnan` が重複定義されエラーを出すので，Springhead の `Env.h` のマクロ `isnan` の定義をコメントアウトする
   (その場しのぎ的な策であるが MapConstructor や Systembuilder, Omnia は問題なく機能する)
4. Visual Studio を用いて，生成されたソリューションファイル
   `MapConstructor/build/MapConstructor.sln` を Debug, Release モードで ALL_BUILD する
5. プロジェクト `MapConstructor` をスタートアッププロジェクトに設定し F5 キーで実行することで，MapConstructor による近接点地図の構築ができる

5.1 実行中に ntdll.dll にてエラー 0xc0000005, 0xc0000374 が発生する場合

1. Application Verifier を起動する
2. `MapConstructor/build/Release/MapConstructor.exe` を Application Verifier でテストするアプリケーションに追加し保存
3. 再度 Visual Studio で MapConstructor を実行

ntdll.dll のエラーは何らかの理由でヒープ破損し発生していると思われる．同条件で実行しても発生しない場合があり再現性が低いため，詳細な原因は不明である．

6 MapConstructor の使用方法

Config Files の xml ファイルで実行内容を設定し F5 キーで実行する.

6.1 platform.xml ファイル

platform タグの要素 name を用いて実行する xml ファイルを指定する. (拡張子 .xml は書かない.) 実行する xml ファイルは MapConstructor/conf に配置しなければ認識されないので注意.

以降に説明するタグは, 実行する xml ファイルに書く.

6.2 SEQUENCE タグ

実行時に行うタスクを羅列する. タスクの指定は task タグの要素 name を用いて指定する. また, タスクによっては実行内容の指定が必要なものもある. 実行内容の指定は要素 name にスペース区切りで書いて指定する. (例 : タスク名 BeforePGO の実行内容 Save を行う場合→ <task name="BeforePGO Save"/>) スペース区切りで実行内容を指定しているため, 各タスクの name にはスペースを使わないこと.

6.3 MapLoader タグ

Omnia を用いて生成した時系列データやループが保存された csv ファイルを読み込む.

要素	内容
name	SEQUENCE のタスク指定に用いるタスクの名前 (重複しないように注意, スペースは使わないこと)
mapNum	読み込む時系列データの数を指定する
loadFormat	読み込む形式をスペース区切りで羅列する
MatchFile	ループ (近接点対) を保存しているファイル名
PoseRefFile	ループ (相対ポーズ) を保存しているファイル名 (MapConstructor にて生成される)

- Map タグを用いて各時系列データの読み込み設定を指定する
- loadFormat には Loc, Prox, Geo, PC, Movement, Match, PoseRef が指定可能

6.3.1 Map タグ

要素	内容
ID	ループ検出の際に用いた時系列データ ID (Match ファイルが時系列データを識別するのに必要)
LocFile	各フレームのロボットポーズを保存しているファイル名
ProxFile	各フレームの近接点位置を保存しているファイル名
GeoFile	RTK 測位結果を保存しているファイル名
PCFile	各フレームの点群位置を保存しているファイル名
MovementFile	各フレームのオドメトリを保存しているファイル名 (MapConstructor にて生成される, Omina で読み込める Movement ファイルとは形式が違うので注意)

6.4 MapSaver タグ

時系列データやループを csv ファイルに書き出す.

要素	内容
name	SEQUENCE のタスク指定に用いるタスクの名前 (重複しないように注意, スペースは使わないこと)
mapNum	書き出す時系列データの数を指定する
saveFormat	読み込む形式をスペース区切りで羅列する
MatchFile	ループ (近接点対) を保存するファイル名
LocMatchFile	ループが定義されたノード対の位置を保存するファイル名 (gnuplot 用)
AbsProxMatchFile	ループが定義された近接点対の位置を保存するファイル名 (gnuplot 用)
PoseRefFile	ループ (相対ポーズ) を保存するファイル名

- Map タグを用いて各時系列データの書き出し設定を指定する
- loadFormat には Loc, Prox, AbsProx, Geo, LocGeo, Movement, Match, LocMatch, AbsProxMatch, PoseRef が指定可能

6.4.1 Map タグ

要素	内容
ID	書き出す時系列データの ID
LocFile	各フレームのロボットポーズを保存するファイル名
ProxFile	各フレームの近接点位置を保存するファイル名
AbsProxFile	各フレームの近接点位置 (絶対座標) を保存するファイル名 (gnuplot 用)
GeoFile	RTK 測位結果を保存するファイル名
LocGeoFile	RTK 測位結果 (時系列データ座標系) を保存するファイル名 (gnuplot 用)
MovementFile	各フレームのオドメトリを保存するファイル名

6.5 ToG2OConverter タグ

時系列データとループの情報を用いてポーズグラフを生成する．また，ポーズグラフのノードの情報を時系列データに反映する．

要素	内容
name	SEQUENCE のタスク指定に用いるタスクの名前 (重複しないように注意, スペースは使わないこと)

- Forward, Reverse の 2 つの実行内容がある

  Forward：時系列データ, ループをポーズグラフにする

  Reverse：ポーズグラフのノードのポーズを時系列データに反映する
- Node タグを用いて各時系列データのノードの形式を指定する
- Odom タグを用いて各時系列データのオドメトリ拘束形式を指定する
- Loop タグを用いて各ループ区間のループ拘束の形式を指定する

6.5.1 Node タグ

要素	内容
ID	この設定を適応させる時系列データ ID を指定する (指定しない場合は全ての時系列データに適応される)
Tag	ノードのタグ名を指定する

- Tag には VERTEX_SE2, VERTEX_PROX のいずれかが指定可能

  VERTEX_SE2：2 次元 3 自由度のポーズノード

  VERTEX_PROX：2 次元 3 自由度のポーズ + 2 次元近接点位置のノード

6.5.2 Odom タグ

要素	内容
ID	この設定を適応させる時系列データ ID を指定する (指定しない場合は全ての時系列データに適応される)
Tag	エッジのタグ名を指定する
poseRef-InfoGain	相対ポーズの各要素の逆数に対する重み行列のゲイン
infoLimit	重み行列の値の上限値

- Tag には EDGE_SE2 のみが指定可能

  EDGE_SE2：2 次元 3 自由度相対ポーズによる拘束
- Movement ファイルを用いてオドメトリ情報を読み込んだ場合はそのオドメトリ情報を用いる
  (Movement ファイルを読み込んでいない場合は連続したノード間の相対ポーズを用いてオドメトリを定義する)

6.5.3 Loop タグ

要素	内容
ID	この設定を適応させる時系列データ ID を指定する (指定しない場合は全ての時系列データに適応される)
Tag	エッジのタグ名を指定する

- Tag には EDGE_SE2, EDGE_SWITCH_SE2, EDGE_PROX, EDGE_SWITCH_PROX, EDGE_M_Est_PROX のいずれかが指定可能

  EDGE_SE2：2 次元 3 自由度相対ポーズによる拘束

  EDGE_SWITCH_SE2：2 次元 3 自由度相対ポーズによる拘束 (スイッチ変数によるロバスト化)

  EDGE_PROX：近接点対の幾何学的特性による拘束

  EDGE_SWITCH_PROX：近接点対の幾何学的特性による拘束 (スイッチ変数によるロバスト化)

  EDGE_M_Est_PROX：近接点対の幾何学的特性による拘束 (M 推定によるロバスト化)

- Match ファイルを用いて近接点対情報を読み込んだ場合は EDGE_SWITCH_PROX, EDGE_M_Est_PROX が利用可能
- PoseRef ファイルを用いて相対ポーズ情報を読み込むか，Match ファイルを用いてスキャンマッチングまたは近接点対マッチングを行なった場合は EDGE_SE2, EDGE_SWITCH_SE2 が利用可能

Tag を EDGE_PROX, EDGE_SWITCH_PROX, EDGE_M_Est_PROX にした場合，次の要素の指定が必要である．

要素	内容
proxSim-InfoGain	近接点対の類似度に対する重み行列のゲイン (0 未満に設定した場合，類似度に限らず負号を取り除いた一定値を重みにする)

Tag を EDGE_SWITCH_SE2, EDGE_SWITCH_PROX にした場合，次の要素の指定が必要である．

要素	内容
switchInfo	各スイッチ変数の重み付け

Tag を EDGE_M_Est_PROX にした場合，次の要素の指定が必要である．

要素	内容
robustKernel	ρ 関数の種類の指定
robustKernelDelta	ρ 関数のパラメータの指定

- robustKernel には Huber, GemanMcClure 等が指定可能

	proxSim-InfoGain	switchInfo	robustKernel,robustKernelDelta
EDGE_SE2	×	×	×
EDGE_SWITCH_SE2	×	◯	×
EDGE_PROX	◯	×	×
EDGE_SWITCH_PROX	◯	◯	×
EDGE_M_Est_PROX	◯	×	◯

6.6 MapOptimizer タグ

G2O を用いてポーズグラフを最適化する.

要素	内容
name	SEQUENCE のタスク指定に用いるタスクの名前 (重複しないように注意, スペースは使わないこと)
ItrNum	Gauss-Newton 法の反復数 (0 に設定した場合，目的関数が十分に収束するまで (変化が 10^{-6}未満になるまで) 反復する)

6.7 G2OFileIO タグ

G2O ファイルの読み込み, 書き出しを行う.

要素	内容
name	SEQUENCE のタスク指定に用いるタスクの名前 (重複しないように注意, スペースは使わないこと)
LoadFileName	読み込む G2O ファイル名
SaveFileName	書き出す G2O ファイル名

- Load, Save の 2 つの実行内容がある

  Load : G2O ファイルを読み込み，ポーズグラフに反映する

  Save : ポーズグラフの情報を G2O ファイルとして書き出す

6.8 MapDirPrepro タグ

ループが定義されたノード対の進行方向の情報を用いて，各時系列データの観測開始時の方向を修正する.

要素	内容
name	SEQUENCE のタスク指定に用いるタスクの名前 (重複しないように注意, スペースは使わないこと)
clacSimEqMethod	正定疎行列連立線形方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ の解法

- clacSimEqMethod には LLT, LDLT, PartialPivLU 等の Eigen の疎行列線形システムソルバが指定可能

6.9 ProxMatcher タグ

ループが定義されたノード対に近接点対マッチングを行ないノード間の相対ポーズを計算する.

要素	内容
name	SEQUENCE のタスク指定に用いるタスクの名前 (重複しないように注意, スペースは使わないこと)
clacSimEqMethod	正定疎行列連立線形方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ の解法
maxItr	Gauss-Newton 法の最大反復数 (最大反復数に達するか，目的関数が十分に収束するまで反復する)

- Match タグを用いて相対ポーズを計算するループ区間を指定する
- clacSimEqMethod には LLT, LDLT, PartialPivLU 等の Eigen の疎行列線形システムソルバが指定可能

6.9.1 Match タグ

要素	内容
ID	ループ区間の ID (指定しない場合，全てのループ区間に近接点対マッチングを行う)

6.10 ScanMatcher タグ

ループが定義されたノード対にスキャンマッチングを行ないノード間の相対ポーズを計算する．点群の読み込みに長大な時間が必要なので注意.

要素	内容
name	SEQUENCE のタスク指定に用いるタスクの名前 (重複しないように注意, スペースは使わないこと)
clacSimEqMethod	正定疎行列連立線形方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ の解法
heightRange	マッチングに用いる点群の高さの閾値 (LiDAR センサの高さは 0.8 [m])
countSkip	マッチングに用いる点群をこの数毎に 1 つになるように間引く
distSkip	マッチングに用いる点群を各点間の距離がこの数以上になるように間引く
ICPItr	ICP アルゴリズムで点群の対応付けをする回数を指定する
ignoreRatio	外れ値除去のために無視する対応付けの割合を指定する (1 つ目は進行方向が同じ向きのノード対の場合，2 つ目は進行方向が逆向きの場合)
maxItr	Gauss-Newton 法の最大反復数 (最大反復数に達するか，目的関数が十分に収束するまで反復する)

- Match タグを用いて相対ポーズを計算するループ区間を指定する
- clacSimEqMethod には LLT, LDLT, PartialPivLU 等の Eigen の疎行列線形システムソルバが指定可能

6.10.1 Match タグ

要素	内容
ID	ループ区間の ID (指定しない場合，全てのループ区間にスキャンマッチングを行う)

6.11 gnuplot による地図データの描画

MapConstructor/gnuplot フォルダのテキストファイルで描画設定を行ない， plot.plt でグラフを fig フォルダに出力する．描画設定に用いるテキストファイルの選択は platform.txt を用いて行う.

7　G2O Viewer の使用方法

G2O をインストールした際に `g2o_viewer.exe` というアプリケーションが一緒にダウンロードされる．これはグラフィックインターフェイスで g2o ファイルを読み込み, 最適化や書き出しを行うソフトである．

g2o ファイルを読み込み (書き出し)

1. File タブの Load (Save) を選択
2. 読み込む G2O ファイルを選択
   (書き出す G2O ファイル名を設定し保存)

ポーズグラフ最適化

1. # Iterations で反復回数を設定
2. Initial Guess を実行
3. Optimize を実行

描画設定の変更

1. View タブの Drawing Options を選択
2. Value で点の大きさや線の太さを変更
   (色の変更はソースファイルを修正しビルドし直す必要がある)