make_zmap 説明書

知能システム研究部門
吉見 隆

2012.04.11

1 あらまし

このプログラムは、kinect の計測結果を使って箱の中のエレベーションマップを出力する。また、kinect の計測結果では欠落を生じてしまうペットボトルのために、ふたの位置から相対的に定義された肩の部分を合成する。

2 構成

プログラムは，以下の要素からなる。

1. kinect サーバー (kinect-server)
2. キャリブレーション (kinect-calib)
3. 計測プログラム
4. zmap 作成プログラム (make_zmap)

それぞれが一つの実行プログラムの形になっている（例外あり)。個々のプログラムの説明を次に示す。

2.1 kinect サーバー (kinect-server)

USB インターフェースに接続した kinect に対してキャプチャ命令を出し，距離画像をレンジデータ（三次元座標値および色) の形でファイルに出力する．現在のところ、プログラムが想定する kinect の台数は一台である．

使いかた:

- `$ sudo kinect-server`

オプションは無い.

プログラム開始後，しばらくすると実行したコンソールにキャプチャシーケンスを表示しはじめる。その状態になればデータの取得が可能。数秒の時間が必要である。

“/tmp/KinectTrigger” という固定名のファイルに対して，出力先のファイル名を書き込むと、そのファイル名にレンジデータを出力する．ファイル名はフルパスで指定する．レンジデータ出力後，プログラムによって “/tmp/KinectTrigger” は削除される.

注意事項

- このプログラムは pcl に依存している。コンパイル前にあらかじめインストールを済ませておくこと
- ポートのアクセス権限の関係で，実行時には sudo で実行する必要がある.

2.2 キャリブレーション (kinect-calib)

通常 kinect で取得される三次元データは kinect の座標系で、画像の右方向が X 下方向が Y、奥方向が Z である．ロボット座標系に変換するためには，4x4 の変換行列 T.mat が必要である。

キャリブレーションプログラムは，あらかじめ与えられた複数の座標にロボットに張りつけたのクロスマークを移動させ、それを kinect で撮影する。各位置のロボット座標値におけるクロスマークの三次元座標値と、レンジデータの輝度情報を使って計算された kinect 座標系のクロスマークの三次元座標値からもっとも適当な変換行列 T.mat を計算する.

T.mat の計算時にはクロスマークから指先の中心部までのオフセットをプログラム内で定数としてもち、それを反映したものを出力する.

使いかた: [1]

1. PA10 を開始位置に移動
   - $ cd /src/PA10/GManipulator.v4.1/client
   - $ ./manip abs_jmove1 -60
   - $ ./manip abs_jmove -60 65 0 90 0 -65 -180
2. プログラムを起動。一回の移動ごとにプロンプトを出すので，安全を確認しながら操作する.

[1] ここの内容は、以下のページで見られる。
http://choreonoid.org/GraspPlugin/i/?q=ja →(ページ下でログイン) → 開発者限定情報 → PA10 → PA10 で kinect のキャリブ

- $ cd /src/PA10/calib.yoshimi
- $ ./calib_pos_kinect

3. 終了して原点に復帰

- $ cd /src/PA10/GManipulator.v4.1/client
- $ ./manip abs_jmove -60 0 0 90 0 90 0
- $ ./manip abs_jmove1 0

出力データは以下の二つである．

- /tmp/tmat-YYYYMMDDHHMMSS.d に 4x4 の matdata
- 標準出力に点データが出力される．

```
# robot coordinates (%d) ← データ個数
xr yr zr
...
# camera coordinates (%d) ← データ個数
xc yc zc
...
```

2.3 計測プログラム

kinect で計測した三次元データをロボット座標系のレンジデータにして出力する．kinect-calib で作成した変換行列ファイル T.mat が必要となる．

現在のところ、シェルコマンドで実行しており，以下の手順で行う．将来的にはシェルスクリプトなどで置き換えるかもしれない．

1. Console 1 で kinect-server を立ち上げる．

- $ sudo kinect-server

2. Console 2 でデータ取得をする．

- $ echo "/tmp/data.rng" > /tmp/KinectTrigger
- Console 1 に表示あり．/tmp/KinectTriger が unlink されるのを待つ (一瞬)
- $ cp /tmp/data.rng .
- $ range_move -i data.rng -o data_R.rng -m tmat-YYYYMMDDHHMMSS.d

2.4 zmap 作成プログラム (make_zmap)

kinect-measure で取得したレンジデータファイルを対象に，ロボット座標系の XY 平面に投影した通い箱内のエレベーションマップをつくる。ペットボトルのキャップに類似した領域が存在すればそれを中心にした円形領域を合成する。この円形領域は擬似的にペットボトルの肩を模したもので、通常の kinect の撮影ではデータが欠落する部分である．

2.4.1 コマンドオプション

コマンドオプションには次のものがある．

これ以外にもパラメータとしては初期セグメンテーション関係のパラメータおよびペットボトルの蓋および肩の部分のサイズに関するパラメータがあるが、これらパラメータのオプションによる変更機能はまだ実装されていない．

セグメンテーション関係のパラメータはデフォルト値が `range_segmentation.h` 内で定義されており、 `main_make_zmap.c`:arg_p() 内で、opt.segarg.XXX の変数に代入されている。それ以外のパラメータは `make_zmap.h` でデフォルト値 (CAP_MIN, CAP_MAX, SHOLDER_RHO, DIFF_Z) が定義されており、tt main_make_zmap.c:main() 内で関数を呼ぶ際に使用されている。変更したい場合には適宜上の部分を修正すること．

`-i` *fin* 入力ファイル名を指定する．デフォルトは標準入力。

`-o` *fout* 出力ファイル名を指定する．デフォルトは標準出力。

`--dth` *dth* レンジデータに対する depth edge のための隣接点との閾値。デフォルトは 5.0[mm](`make_zmap.h`:DEF_DTH で定義)。

`--zth` *zth* 箱領域を抽出するための高さ閾値．ロボット座標系で箱の縁から少し下の高さを指定する。デフォルトは -50.0 [mm] (`make_zmap.h`:DEF_Z_TH で定義)。

`--pitch` *pitch* zmap の X および Y 方向のメッシュのサイズ。デフォルトは 1.0[mm] (`make_zmap.h`:DEF_PITCH で定義)。

2.4.2 出力ファイルフォーマット

出力ファイルには，make_zmap のオプション，通い箱の長方形の XY 平面におけるパラメータ，マップの XY 平面内のオフセットおよびサイズ，三次元点の配列、領域分割数、領域のパラメータ，キャップ領域の数とリストを出力する．プログラムの中間データなどをほとんどすべて含んでいるが，ラベル画像だけは含んでいない。

1. `opt_make_zmap=`*z_th dth pitch*

   z_th 箱領域を検出する際に用いるZの閾値で、箱の縁よりわずかに低い値を用いる。

   dth レンジデータの距離エッジをつける際の閾値 [mm]。

   pitch XY 平面のメッシュのサイズ [mm]。

2. `opt_make_zmap.segarg=`*lshort lext deltalen width0 width1 mergespeed*
   セグメンテーションを行う関数のためのパラメータ

   lshort 除去される微小セグメントの長さ

   lext 延長処理するときの長さ

   deltalen 微小セグメント除去をする際の長さ測定の最小距離

   width0 強制マージする際の微小領域幅

   width1 テクスチャ領域にマージする際の微小領域幅

   mergespeed 強制マージするときの分割数

3. `rectangle.[xy][min|max]=`*xmin xmax ymin ymax*
   長方形領域の XY 座標系における範囲 [mm]

4. `rectangle.corner=`*corner[4][2]*
   長方形頂点の XY 座標 [mm]

5. `rectangle.center=`*corner[2]*
   長方形中心の XY 座標 [mm]

6. `rectangle.dir=`*ux uy vx vy*
   長方形軸方向ベクトル

7. `rectangle.[uv][min|max]=`*umin umax vmin vmax*
   長方形軸方向の最大最小値

8. `zmap.mapsize_[xy]=`*mapsize_x mapsize_y*
   zmap の XY 方向の画素数

9. `zmap.offset_[xy]=`*offset_x offset_y* zmap の画素 (0,0) の XY 座標

10. `zmap.pitch=`*pitch*
    zmap の 1 画素に相当する長さ [mm]

11. X Y Z
    zmap の各点の座標値．(mapsize_x * mapsize_y) 個のデータがある。gnuplot データへの変換のために、mapsize_x 個ごとに空行が入っている．このデータには prefix がつかない。

12. `zmap.Z_[NoData|OutRect]=`*Z_NoData Z_OutRect*
    zmap でデータの欠落部分と長方形外の部分を示す Z の値を示す。有効な Z 座標値から重複しないように自動的に計算されたもの．

13. `zmap.dz=`*dz*
    箱内の領域分割をする際の高さ閾値 [mm]

14. `nlabel=`*nlabel*
    領域数

15. `regprop[`*ir*`].g[0-1]=`*regprop[ir].{g[0] g[1]}*
    領域 *ir* の二次元重心 (*ir* は 0 .. *nlabel-1* の値。以下同様)

16. `regprop[`*ir*`].v[0-1][0-1]=`*regprop[ir].{v[0][0] v[0][1] v[1][0] v[1][1]}*
    領域 *ir* の二次元主軸ベクトル

17. `regprop[`*ir*`].[min|max][0-1]=`*regprop[ir].{min[0] max[0] min[1] max[1]}*
    領域 *ir* の二次元主軸ベクトル方向の最大最小値

18. `caplist.n=`*caplist.n*
    ふたのサイズ条件を満たす領域の数

19. `caplist.d[`*ic*`]=`*caplist.d[ic]*
    ふたのサイズ条件を満たす領域番号。(*ic* は 0..caplist.n-1 の値)

2.4.3 入出力関数

`make_zmap.h` 内で定義された関数 `input_zmap()` および `output_zmap()` を使うことでファイル入出力が可能である。ソースプログラムは `zmap_io.c` にある。

2.4.4 可視化スクリプト

zmap ファイルは、以下のユーティリティによって可視化できる。

`script_zmap_to_matdata.sh` このスクリプトは，zmap データを matview 表示用の matdata に変換するものである。データ以外の部分を削除し，gnuplot 表示用に挿入してある空行も削除する．さらに、先頭行を追加する．入力ファイル名と出力ファイル名が必要である．

`script_zmap_to_plot.sh` このスクリプトは，zmap データを gnuplot 表示用の三次元データに変換するものである。データ以外の部分を削除する．入力ファイル名と出力ファイル名が必要である．

2.4.5 関数について

ファイル `make_zmap.c` には、プログラム内のほとんどの関数が含まれる。ヘッダファイル `make_zmap.h` をインクルードすることで、利用できる。ライブラリ化はしていない。以下に有用な関数について述べる．

`check_inside_rect(double x, double y, RectAngle *rect)` 点(x,y)が長方形 rect 内部にあるかどうかの判定を行う．