Quasi-Euclidean Epipolar Rectification: user's guide

Pascal Monasse

June 25, 2014

1 Introduction

This document is a short user's guide to software program `rectifyQuasiEuclidean` that puts an image pair in rectified epipolar geometry.

2 Installation

2.1 Requirements

Dependencies of the software are the following:

- `libpng` and `zlib` for reading/writing images in PNG format (`http://libpng.sourceforge.net/index.html` and `http://www.zlib.net/`)
- `CMake` for building the software (`http://www.cmake.org/`)
- C++ compiler (GNU `g++`)

Most linux distributions propose easy to install packages for these (if not already installed by default). If you need to install them, be sure to use the developper's versions (package with extension `-dev`), so that you get header files and not only the libraries.

The third party libraries `libpng` and `zlib` are optional: if they are not found on the system, local versions will be built and used.

2.2 Build

The build process has three steps:

1. Decompress the archive.
2. Launch `cmake` to generate `Makefile`.
3. Launch `make` for compilation and link.

2.2.1 Decompress the archive

To decompress, you can input in a shell the command

```
$ tar xzf rectify-quasi-euclidean_<version>.tgz
```

Replace ¡version¿ with the version number.

2.2.2 Launch cmake

Create a folder of your choice where to install the software, for example `build`, and go to that folder:

```
$ cd rectify-quasi-euclidean_<version>
$ mkdir build
$ cd build
```

Launch `cmake` with argument the base folder containing the source codes (there is a file `CMakeLists.txt` in that folder):

```
$ cmake ../src
```

This checks the availability of the dependencies and outputs `Makefile` in case of success. To build optimized version of programs, the variable `CMAKE_BUILD_TYPE` must be modified using

```
$ cmake -D CMAKE_BUILD_TYPE:string=Release ../src
```

or with utility `ccmake` (notice the double `c`).

By default, the third party libraries for image input/output are searched on the build machine and used if present. You can choose to skip selectively this search and use the provided ones with the option:

```
$ cmake -DWITH_INCLUDED_LIBPNG:bool=ON ../src
```

Another options is `WITH_INCLUDED_ZLIB`. However, the option `WITH_INCLUDED_ZLIB` is ignored if `WITH_INCLUDED_LIBPNG` is not set.

2.2.3 Launch make

To build, simply type

```
$ make
```

You can also use the option '-j2' to launch two parallel compilations (or more if you have additional cores). The executable files are then in folder `bin` and libraries in `lib`. For example, you get `lib/libNumerics.a` and `bin/rectify`.

By default, static libraries are produced. If you prefer dynamic ones, you can set to ON the variable `BUILD_SHARED_LIBS`, either by adding the option when launching `cmake`

```
$ cmake -D BUILD_SHARED_LIBS:bool=ON ../src
```

or by using the utility `ccmake` before calling `make` again.

3 Usage

3.1 Testing your installation

As a test of your build, you can launch the following:

```
$ build/bin/rectifyQuasiEuclidean data/CarcassonneSmall/im[12].png out1.png out2.png
```

This launches the pipeline on the images `im1.png` and `im2.png` of folder

`data/CarcassonneSmall`

In case of success, you can visually compare the resulting images `out1.png` and `out2.png` to images in folder `data/CarcassonneSmall`.

3.2 Program workflow

1. SIFT: find SIFT points and correspondences between both input images.
2. AC-RANSAC (aka ORSA): find correspondences compatible with F matrix.
3. Rectify: compute rectifying homographies.
4. Homography: apply the homographies and output images.

3.3 Example

```
$ build/bin/rectifyQuasiEuclidean data/CarcassonneSmall/im[12].png out1.png out2.png
sift: im1: 550 im2: 508 matches: 261
Remove 30/261 duplicate matches
  nfa=-332.077 inliers=204 precision=1.36755 im2 (iter=0,sample=2,23,27,28,87,198,218)
  nfa=-363.582 inliers=229 precision=2.2345 im2 (iter=3,sample=14,139,76,216,134,62,152)
  nfa=-423.3 inliers=225 precision=1.02865 im2 (iter=7,sample=154,103,41,186,166,33,114)
  nfa=-437.086 inliers=219 precision=0.708313 im2 (iter=62,sample=154,99,224,177,210,198,6
  nfa=-439.23 inliers=218 precision=0.666444 im2 (iter=64,sample=11,150,91,185,31,20,229)
  nfa=-446.762 inliers=221 precision=0.688147 im2 (iter=103,sample=12,111,161,19,163,188,2
  nfa=-450.868 inliers=220 precision=0.633637 im2 (iter=583,sample=39,5,217,63,23,64,66)
F= [ -1.72604e-09 -1.31828e-08 -5.07353e-05;  -4.83721e-08 -1.49106e-08 -0.00204647;  8.53
LM iterations: 50 f=1082.87
K_left: [ 1082.87 0 274.279;  0 1082.87 142.5;  0 0 1 ]
K_right: [ 1082.87 0 237.584;  0 1082.87 142.5;  0 0 1 ]
Initial rectification error: 10.2437 pix
Final rectification error: 0.149013 pix
$
```

Remark: You can get slightly different results in each run because of the stochastic nature of the RANSAC algorithm.

4 Troubleshooting

Please send an email to the maintainer Pascal Monasse (monasse@imagine.enpc.fr) describing your problem.

List of files

```
rectify-quasi-euclidean_<version>:
BUILD.txt  data  doc  LICENSE.txt  README.txt  src

rectify-quasi-euclidean_<version>/data:
CarcassonneSmall

rectify-quasi-euclidean_<version>/data/CarcassonneSmall:
H_im1.png  H_im2.png  im1.png  im2.png
```

```
rectify-quasi-euclidean_<version>/doc:
userguide.pdf  userguide.tex

rectify-quasi-euclidean_<version>/src:
third_party     libIO       libMatch     libOrsa       main.cpp  sift
CMakeLists.txt  libLWImage  libNumerics  libTransform  rectify   warp

rectify-quasi-euclidean_<version>/src/third_party:
jpeg-9a  libpng-1.6.12  tiff-4.0.3  zlib-1.2.8

rectify-quasi-euclidean_<version>/src/third_party/jpeg-9a:
...

rectify-quasi-euclidean_<version>/src/third_party/libpng-1.6.12:
...

rectify-quasi-euclidean_<version>/src/third_party/tiff-4.0.3:
...

rectify-quasi-euclidean_<version>/src/third_party/zlib-1.2.8:
...

rectify-quasi-euclidean_<version>/src/libIO:
CMakeLists.txt  cmdLine.h  io_png.c  io_png.h  nan.h

rectify-quasi-euclidean_<version>/src/libLWImage:
LWImage.cpp  LWImage.h

rectify-quasi-euclidean_<version>/src/libMatch:
CMakeLists.txt  match.cpp  match.h

rectify-quasi-euclidean_<version>/src/libNumerics:
ccmath_svd.cpp  homography.cpp  matrix.h      rodrigues.cpp
CMakeLists.txt  homography.h    numerics.cpp  rodrigues.h
computeH.cpp    matrix.cpp      numerics.h    vector.cpp

rectify-quasi-euclidean_<version>/src/libOrsa:
CMakeLists.txt    fundamental_model.cpp  orsa.cpp        orsa_model.hpp
conditioning.cpp  fundamental_model.hpp  orsa.h
conditioning.hpp  main.cpp               orsa_model.cpp

rectify-quasi-euclidean_<version>/src/libTransform:
CMakeLists.txt    map_image.cpp  spline.h
gauss_convol.cpp  map_image.h    TransformSize.cpp
gauss_convol.h    spline.cpp     TransformSize.h

rectify-quasi-euclidean_<version>/src/rectify:
CMakeLists.txt  main.cpp  rectify.cpp  rectify.h

rectify-quasi-euclidean_<version>/src/sift:
```

```
CMakeLists.txt     domain.cpp  im1.png      numerics.cpp  splines.h
demo_lib_sift.cpp  domain.h    im2.png      numerics.h
demo_lib_sift.h    filter.cpp  library.cpp  prova.png
demo_sift.cpp      filter.h    library.h    splines.cpp

rectify-quasi-euclidean_<version>/src/warp:
CMakeLists.txt  warp.cpp
```

List of authors

- Toni Buades (toni.buades@uib.es): `sift`
- Nicolas Limare (nicolas.limare@cmla.ens-cachan.fr): `libIO`
- Lionel Moisan (Lionel.Moisan@parisdescartes.fr): `libTransform`, `libOrsa`
- Pascal Monasse (monasse@imagine.enpc.fr): `homography`, `libIO`, `libLWImage`, `libMatch`, `libNumerics`, `libTransform`, `libOrsa`, `rectify`, `sift`, packaging, documentation
- Pierre Moulon (pmoulon@gmail.com): `libOrsa`
- Zhongwei Tang (tangfrch@gmail.com): `libTransform`, `libOrsa`, `sift`