LO-RANSAC Software Library Manual

(Building and Usage)

Karel Lebeda, karel@lebeda.sk

The code of LO-RANSAC is now publicly available under the GNU GPL license. The home page can be found at http://cmp.felk.cvut.cz/software/LO-RANSAC/. In case you use this software in an academic work, please cite the following paper: [4]. This manual was originally published as a part of Karel Lebeda's Master's Thesis [3].

1 Dependencies

The library has only one dependency – LAPACK: Linear Algebra Package [1]. In Linux distributions, this is often available via package management system (e.g. package liblapack). We were informed that on Windows it is possible to replace LAPACK by Intel MKL library. Then, besides changing the path in a Makefile (see below), it is necessary to replace LAPACK #includes by <mkl.h>. This was reported as working with MS Visual Studio 2008 on Windows 7 x64. Another possibility is to use LAPACK for Windows. However, we were not able to check reliability of either of these solutions.

The LO-RANSAC library also uses several procedures from CCMATH library [2]. Nevertheless, since this is included in the distribution archive, it is not necessary to be concerned about those at all.

2 Building the library

The unpacked LO-RANSAC directory contains a license file, information about authors with contact information, README file with content similar to this chapter and src subdirectory, containing the source files. In the src directory, the Makefile is prepared for an easy build. The debug/release build can be switched there by simply (un)commenting lines (turning on/off optimization or debugging information).

As the LAPACK installation usually does not include the header files, it is necessary to obtain those elsewhere. E.g., Matlab carries a set of headers along. The path to the headers should be specified in the Makefile as well.

Running the Makefile compiles all the source files into C objects and then link these together with LAPACK library. A file libransac.a is the result. Then it is possible to link this file to other software using -lransac -L. E.g. for the use in Matlab, MEX-sources are prepared. These should be compiled as follows:

```
mex loransacH.mex.c -o loransacH.mexglx -lransac -L. -llapack.
```

This example is for 32-bit Linux. The proper MEX-file extension should be used, according to the particular platform (use the mexext command in Matlab).

The following error has been reported when trying to run LO-RANSAC MEX-files from Matlab (Matlab 2013a on Ubuntu 12.04):

```
Invalid MEX-file '/.../loransacF.mexa64':
libgfortran.so.3: version 'GFORTRAN_1.4' not found (required by liblapack.so.3gf)
```

In such a case, it is necessary to preload the gfortran dynamic library before Matlab is started. One solution is to type

```
export LD_PRELOAD='/.../libgfortran.so.3'
```

in the command line, before starting Matlab (from the command line; replace ... with your path to libgfortran.so.3).

3 Application Programming Interface

The main gateway to the library consists of two functions for geometry estimation: `ransacH` and `ransacF` for homography and epipolar geometry estimation (respectively). Their arguments are summarized in Table 3. Since the library is often called from Matlab MEX-files, the matrices are stored column-wise (in a column-major order). In these arguments, all the essential RANSAC parameters are possible to set. Also, the `ransac*` functions return inliers/outliers bipartitioning of the data points by filling the `inl` array by ones and zeroes, and runtime statistics (number of samples drawn, local optimisations performed and models rejected). The score of the best geometry hypothesis found is returned in a special structure, containing the number of inliers `I` and the widened truncated quadratic score `J` (note that this is implemented as a *gain* function to be maximized, to be consistent with the inlier count). Its definition is:

```
typedef struct { unsigned I; double J; } Score;
```

If it is necessary to change some of the inner LO-RANSAC constants or parameters (e.g. the number of inner samples in LO, or the cost function), this can be done in the file `rtools.h`. On the other hand, for the cases when an "as simple as possible" interface is needed, we have prepared wrapping functions `ransacHsimple` and `ransacFsimple`. In these, all the parameters are set to their default values wherever it makes any sense. Furthermore, nothing but the geometry matrix is returned. The default values can be found in the Table 3 as well.

For an example of a typical LO-RANSAC usage see the files `testH.c` and `testF.c`. Furthermore, these indicate a proper function of the LO-RANSAC library. The Makefile compiles them with the library, thus they can be easily run from the system terminal (e.g. `./testH` for homography). Small deviations from the expected values in the exact tests do not mean necessarily that there is a problem. There can be differences, e.g. between platforms. The reported values were obtained on 32b Linux.

`Score`	`ransacH`	`(double * u, int len, double th, double conf,`
		`int max_sam, int do_lo, int inlLimit,`
		`double * H, unsigned char * inl, unsigned * stats);`

`Score`	`ransacF`	`(double * u, int len, double th, double conf,`
		`int max_sam, int do_lo, int inlLimit,`
		`double * F, unsigned char * inl, unsigned * stats);`

`void`	`ransacHsimple`	`(double *u, int len, double th, double *H);`
`void`	`ransacFsimple`	`(double *u, int len, double th, double *F);`

Table 1: Application Programming Interface. A user can choose between a more detailed parameter settings and the simple variant where as many as possible parameters are set to their respective default values. See table 3 for a description of the arguments.

`[H]`	=	`loransacH(TC_PAIRS, THRESHOLD)`
`[H, INL, STATS]`	=	`loransacH(TC_PAIRS, THRESHOLD` `[, LO_ON, CONF, INL_LIMIT, MAX_SAM, RAND_SEED])`
`[F]`	=	`loransacF(TC_PAIRS, THRESHOLD)`
`[F, INL, STATS]`	=	`loransacF(TC_PAIRS, THRESHOLD` `[, LO_ON, CONF, INL_LIMIT, MAX_SAM, RAND_SEED])`

Table 2: Matlab MEX-files Application Programming Interface. Again, the only mandatory inputs are the tentatively corresponding pairs of points (here called `TC_PAIRS` instead of `u`) and the inlier/outlier error threshold. A matrix, representing the geometry, is always returned; optionally also with the inliers/outliers separation. One additional parameter is present here – the seed for pseudorandom generator (`srand` in C).

Name	Direction	Type	Size	Default value (F/H)
`F/H`	output	double	3×3	NA
fundamental matrix / homography matrix (M^*)				
`inl`	output	binary	1×len	NULL (not returned)
inliers/outliers separation ($\mathcal{I}^*$)				
`stats`	output	unsigned	1×3	NULL (not returned)
runtime statistics (#samples, #LOs, #rejected models)				
`u`	input	double	6×len	NA
input data – tentative correspondences				
`len`	input	int	scalar	NA
length of input tentative correspondences (N)				
`th`	input	double	scalar	NA
squared inlier-outlier error threshold (θ^2, px^2)				
`conf`	input	double	scalar	0.95
user-required probability of obtaining the best solution (η_0)				
`max_sam`	input	int	scalar	1 000 000
maximal number of samples drawn				
`do_lo`	input	binary	scalar	1
LO on/off switch				
`inlLimit`	input	int	scalar	49/28
maximal number of inliers for iterative lest squares, 0 = no limit				

Table 3: The list of LO-RANSAC arguments. This table applies to C interface, all the properties nevertheless hold even for the MEX interface. For the interface of the MEX-files please refer to the Table 2.

The input matrix `u` is of size $6\times N$. It consists of the homogeneous coordinates of both tentatively corresponding points – i^{th} column can be expressed as $[x_i, y_i, w_i,\ x'_i, y'_i, w'_i]^\top$ or $[\mathbf{x}_i^\top,\ \mathbf{x}_i'^\top]^\top$. On the output, a model of the desired geometry is returned, satisfying

$$\text{rank}\mathbf{H} = 3\ ,$$

$$\mathbf{x}_i \times \mathbf{H}\mathbf{x}'_i = 0 \qquad \forall i : \texttt{inl}(i) = 1$$

for the homography estimation, and

$$\text{rank}\mathbf{F} = 2\ ,$$

$$\mathbf{x}_i^\top \mathbf{F}\mathbf{x}'_i = 0 \qquad \forall i : \texttt{inl}(i) = 1$$

for the epipolar geometry estimation, with the tolerance θ of the correspondence Sampson's error ($e_i^2 \leq \theta^2$). Please note that the notation used in the CMP WBS-Demo is used here, instead of the more common $\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = 0$ and $\mathbf{x}'^\top_i \mathbf{F}\mathbf{x}_i = 0$.

References

[1] LAPACK: Linear algebra package. *http://www.netlib.org/lapack/*.

[2] D. A. Atkinson. CCMATH: Mathematics software library. *http://freecode.com/projects/ccmath*.

[3] K. Lebeda. Robust Sample Consensus. Master's thesis, Czech Technical University in Prague, 2013.

[4] K. Lebeda, J. Matas, and O. Chum. Fixing the locally optimized RANSAC. In *Proceedings of the British Machine Vision Conference*, pages 95.1–95.11, 2012.