Read me (v2.0, April 2016)

This is the readme file for:

- line extraction.
- vanishing point estimation (and line clustering and rotation estimation).

I) General information

The code is in Matlab and was mainly implemented by Jean-Charles Bazin. If you have any questions, do not hesitate to contact me (jean-charles.bazin@inf.ethz.ch). The code is dirty and not optimized, but it does the job ;-)

Please do *not* redistribute the code.

If you use the code, you must cite one of the following papers:

- Line extraction:

- "Motion Estimation by Decoupling Rotation and Translation in Catadioptric Vision" by Jean-Charles Bazin, Cedric Demonceaux, Pascal Vasseur, and Inso Kweon, CVIU, 2010.

- **Vanishing point estimation, line clustering and rotation estimation**:

- "Globally optimal consensus set maximization through rotation search" by Jean-Charles Bazin, Yongduek Seo, and Marc Pollefeys, ACCV, 2012.
- "3-line RANSAC for orthogonal vanishing point detection" by Jean-Charles Bazin and Marc Pollefeys, IROS, 2012.

II) How to run the code

II-1) LINE EXTRACTION

Let's first extract lines. We provide here the code for omnidirectional images acquired by a Ladybug camera, but the same method can be applied for other wide field-of-view images (such as catadioptric, and fisheye lens) and also for perspective images, as explained in the following.

Go to the folder named CVIU2009_LineExtraction_Ladybug. The main function is Fn_Demo_Ladybug_MAIN.m for Ladybug (omnidirectional) images, and Fn_Demo_Perspec_MAIN.m for conventional (perspective) images. You can set some parameters, such as the filename of the input image and an optional binary mask (for example to ignore the parts of the car that carries the camera).

You can play with some parameters in Fn_Detection_OmniLine, for example ThresholdPixel. Please refer to the comments in the code and our CVIU 2009 paper for the details and meaning of the parameters.

Note: by setting the calibration parameters (Hc, phi, etc...) and the projection model (flag_ProjectionModel), you can apply the code for any central images (e.g. fish eye).

We provide two input images:
- AMRO_65000.png acquired by a (conventional) perspective camera. Thanks to Bastien!
- LadybugIndoor.png acquired by a Ladybug camera. Thanks to Olivier!

Run the main function. If everything works fine, a figure should pop up with the extracted lines in red (see example in the below Figure1).

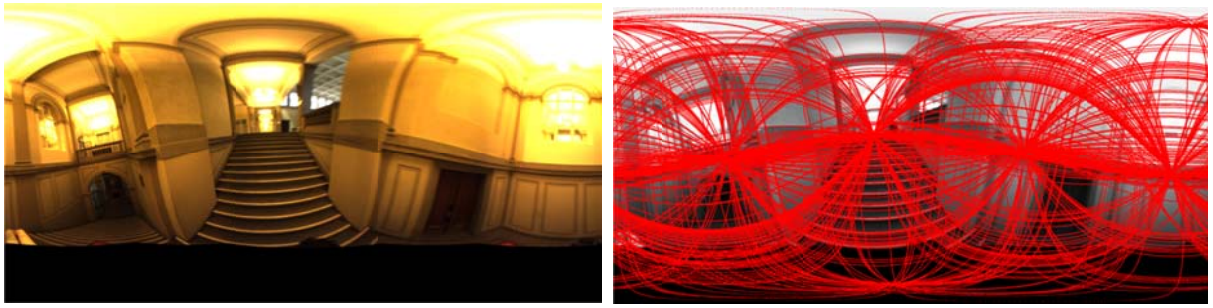

Figure1: input image (left) and extracted lines (right)

II-2) LINE CLUSTERING AND ROTATION ESTIMATION

To estimate the vanishing points (and line clustering and rotation estimation), our approach extracts the lines and then cluster them. We assume the vanishing points are orthogonal.

Our method returns:

- the vanishing points: their 3D directions and the coordinates of their projection in the images (i.e. where the parallel lines intersect in the images).
- the line clustering: i.e. which line belongs to which vanishing point.
- the rotation of the camera: the three rotation angles and the rotation matrix.

In the previous section, we saw how to extract the lines. Let's now see how to cluster these lines. We provide two approaches:

- our IROS 2012 approach which runs a 3-line RANSAC
- our ACCV 2012 approach which maximizes the number of inliers (i.e. the number of clustered lines) in a globally optimal way via branch-and-bound

The code calls the line extraction function that was explained in the above section. An important parameter for clustering is InlierThreshold. It controls when a line is considered an inlier (see our ACCV 2012 or IROS 2012 paper for details).

II-2-A) IROS 2012 (RANSAC)

Go to the folder named IROS2012_LineClustering_ByRANSAC. The main function is Fn_main.m.
You can control the number of RANSAC iterations in Fn_Find_Rotation_by_MultiRansac (see NbRansacIterations for example).

If everything works fine, a figure should pop up with the clustered lines displayed in different colors (red, green, blue).

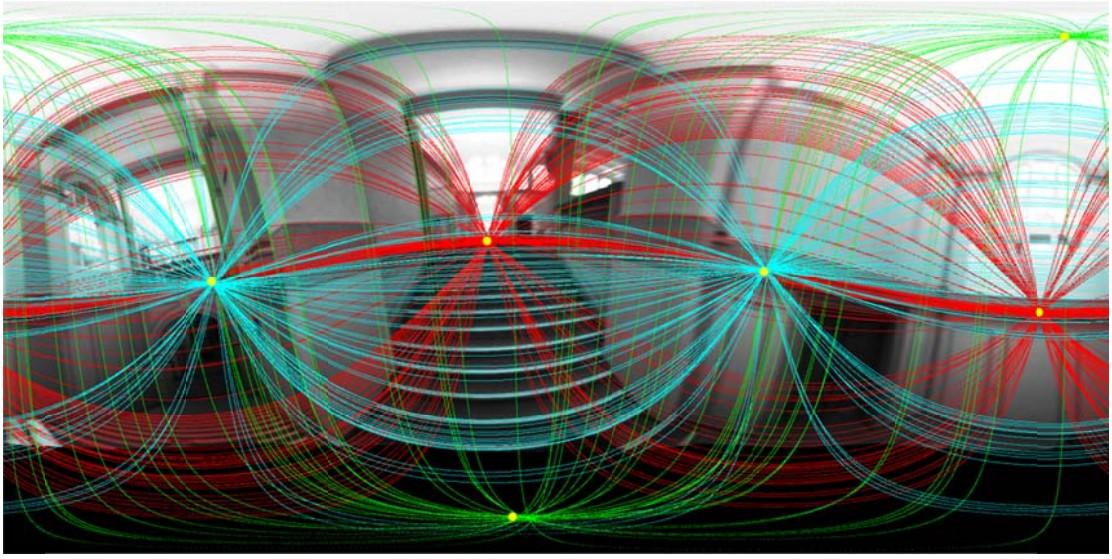

Figure2: given the lines shown in Figure1, line clustering result obtained by our 3-line RANSAC. Each color corresponds to one cluster of lines. 223 inlier lines were detected.

II-2-B) ACCV 2012 (Branch and Bound)

Go to the folder named ACCV2012_LineClustering_BnB. The main function is Fn_main.m.

The search can be performed in depth or breadth first search (set FlagDepthOrBreadthSearch). For depth first search, you can choose the selection of the "best" cube by setting flag_DFS_SelectionMethod.

If everything works fine, a figure should pop up with the clustered lines displayed in different colors (red, green, blue).

The number of inlier lines detected by our branch-and-bound approach should be equal to or higher than by the RANSAC approach, since our BnB approach is globally optimal.

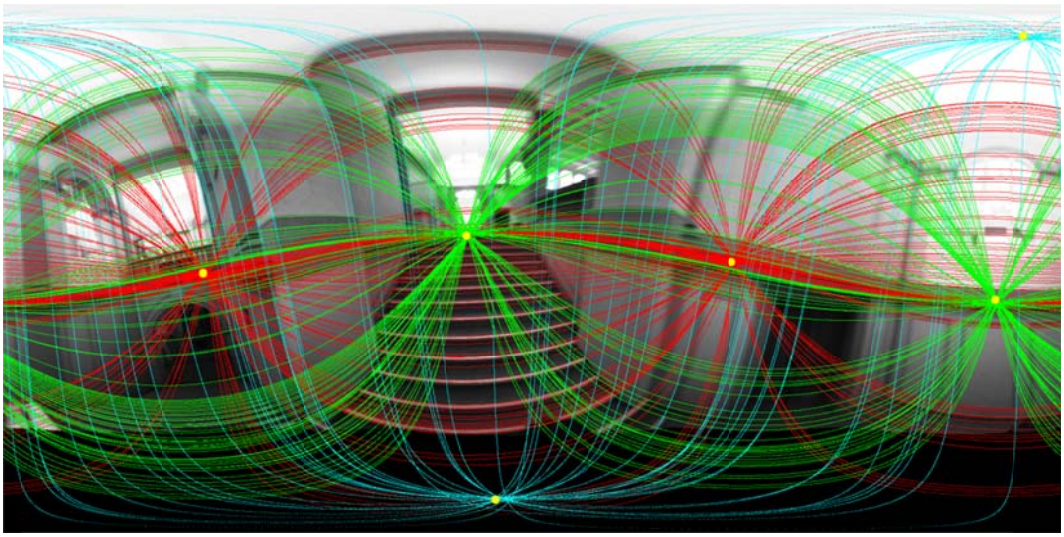

Figure3: given the lines shown in Figure1, line clustering result obtained by our globally optimal branch-and-bound approach. Each color corresponds to one cluster of lines. 224 inlier lines were detected.

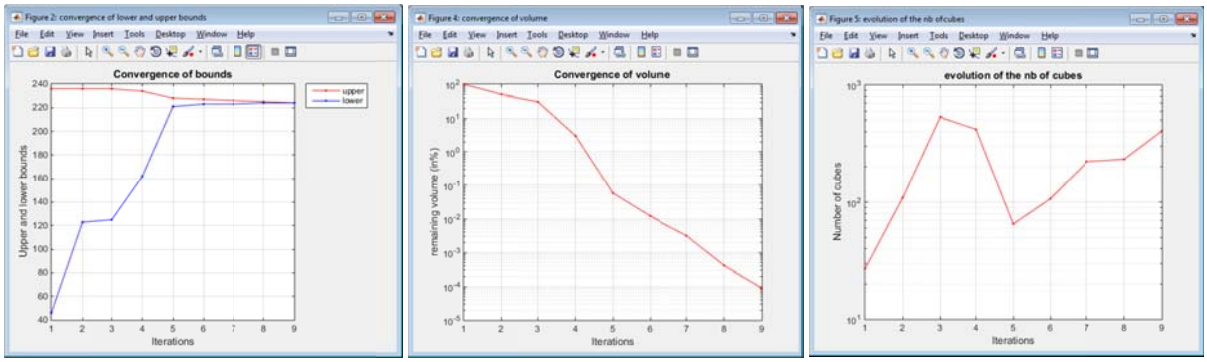

Figure4: analysis of the branch-and-bound procedure: convergence of the bounds, convergence of the volume and evolution of the number of cubes.