ToF Denoising – Quick Start Guide

Frank Lenzen and Henrik Schäfer

Heidelberg Collaboratory for Image Processing (HCI),
Heidelberg University, Germany
and
Intel Visual Computing Institute,
Saarland University, Germany

frank.lenzen@iwr.uni-heidelberg.de

Software version 4/2016

June 15, 2016

1 License

2 General Notes

The algorithm implemented in this software package is described in [5]. In case that you use the output of this software in a publication, we kindly ask to cite the above paper.

Our algorithm is designed for and tested with the PMD CamCube 3. In particular, the Shadow Removal Step (see Section 5) assumes the specific lighting characteristic of the PMD CamCube. In case of other camera brands/models, we suggest to not use the shadow-removal (disabled by default) or change the properties directly in the source code.

This quick start guide for the ToF denoising software assumes Linux as the installed OS. The described installation routine was tested with Ubuntu 12.04, 13.04, 15.10 and 16.04.

After unzipping the archive, the following directory tree is created (referring to the root directory, where the file is unzipped, as ./):

directory	content
./src/	source files and CMakeLists.txt
./docu/	this document
./bin/	empty, used to store executables after compilation
./tools/	m-file to read raw data, cimg-viewer, conversion tool (see Section 8)
./calib/	generic calibration file (cf. Section 5).

For further reading on the principle of ToF cameras and denoising ToF data, we refer to [2, 3].

3 Requirements

Required packages (installed either globally or locally):

1. cmake, (optional: cmake-gui)
   Ubuntu package: cmake
2. hdf5
   Ubuntu package: libhdf5-dev (or libhdf5-serial-dev)
3. Vigra with HDF5 support, website: `https://ukoethe.github.io/vigra/`,
   github: `https://github.com/ukoethe/vigra`
   Hints:
   - can be installed locally (build directory) or globally ('make install')
   - check correct paths for hdf5 with cmake-gui
4. openCV, website: `http://opencv.org/`
   Ubuntu packages: libopencv-dev python-opencv
5. CImg[1], website: `http://cimg.sourceforge.net/`,
   Ubuntu package: cimg-dev

[1] requires the X11 and pthread libraries, which should be installed by default.

4 Installation

1. Modify the file ./src/CMakeLists.txt: if you use locally installed packages, please set the correct paths in Vigra_DIR (build directory) , Vigra_Root_DIR (parent folder to build), CIMG_DIR (path to CImg.h) and OPENCV_DIR (main directory). In case of globally installed packages the respective path is not be required.

2. Create directory *build* in *./src*.

3. In *./src/build*, call *cmake ..* (including dots !) and then *make*. This creates the executable *tof_denoising* in *./bin*.

Trouble shooting:

Vigra: Has Vigra been built/installed successfully? Is hdf5 support enabled (check with cmake-gui ..)?

OpenCV (local inst.): Has openCV been built successfully?

5 Main Parts of the Algorithm

The algorithm implemented in *tof_denoising* requires the depth and intensity images acquired with a ToF camera, stored as cimg-images (`http://cimg.sourceforge.net/`). The tool *convert2cimg* can be used to convert standard image formats to cimg, see Section 8.

As additional input, the shadow removal step (see below) requires a calibration file.

The algorithm consists of the following main steps:

1. EGDE-1: detection of definite edges in the depth image.

2. EGDE-2: detection of shadow casting edges in the depth image.

3. EGDE-3: detection of edges in the intensity image.

4. ShadowRemoval (optional, PMD CamCube): removes intensity edges caused by shadows; requires a calibration file. A generic calibration file *generic_calib.h5* is provided with this package.

5. TextureRemoval (optional): removes intensity edges caused by texture.

6. Denoising: Uses first- and second-order anisotropic total variation to remove noise. The anisotropy is determined based on the edge information retrieved in the previous steps.

As mentioned before, the Shadow Removal Step makes use of specific lighting property of the PMD CamCube. In case of other camera brands/models, we suggest to not use the shadow-removal (disabled by default) or change the properties directly in the source code.

For details on the algorithm we refer to [5].

6 Parameter File

tof_denoising requires one parameter file per ToF data set, in which all parameters (incl. input and output files) are stored. The file ending is assumed to be *.par*. The program call is *./bin/tof_denoising parameterfile.par*.

The parameters in the parameter file are organized in sections, with section names given in square brackets. Each section contains parameter entries consisting of a parameter name, a colon plus space and a parameter value (name: value). One entry per line is assumed; the rest of each line is ignored. The parameter name is assumed to be unique in each section. For some parameters, default values (see below) are available, which are used if no entry is provided in the parameter file.

The following table shortly describes the individual parameters:

Parameter name	**Type**	**Default**	**Description**
[Input]			
intensity:	string	N/A	name of existing cimg-file (float values)
depth:	string	N/A	name of existing cimg-file (float values)
[Output]			
output_file:	string	N/A	name of output file (overwriting), formats are .dat, .cimg and the standard vigra image formats[2])
additional_output:	yes/no	no	use 'yes' for additional output files (png/cimg)
path_additional_output:	string	./	sub-directory to store additional output
[EDGE-1]			
gwidth:	int	5	width of Gaussian for gradient and structure tensor calculation, only odd values allowed
sigma:	float	0.8	sigma of the same Gaussian
upsampling	0/1	1	perform upsampling during gradient calculation
hggaussangle	float	0.5	angle of the "hourglass" shaped Gaussian: 0=very thin, the larger, the less adaptive the structure tensor is.
lthreshold:	float	0.01	lower threshold for Canny
uthreshold:	float	0.1	higher threshold for Canny, larger than lthreshold
adaptiveness:	int	0	adapt thresholds to intensity ("0" = off, "1" = linear, "2" = squareroot, "-1" = reciprocal, "-2" = negative)
edgelength:	int	5	minimum length of an edge
curve1:	float	50.0	allowed max angle between edge direction of two adjacent pixels in deg
curve2:	float	80.0	allowed max angle between edge direction and adjacent edge pixel in deg

CONTINUED ON NEXT PAGE

Parameter name	Type	Default	Description
[EDGE-2]			
same as in [EDGE-1]			
[EDGE-3]			
same as in [EDGE-1]			
[ShadowRemoval]			
remove_shadows:	yes/no	no	
calibrationfile:	string	default_calib.h5	hdf5 file containing a camera matrix *cmatrix*. Used only if remove_shadows:=yes
[TextureRemoval]			
remove_textures:	yes/no	no	if yes, the following parameters should be provided.
threshold:	float	0	depth threshold above which edges are not removed
thresholdg:	float	0	gradient threshold above which edges are not removed
neighborhood:	int	0	width of neighborhood in pixels on either side of the edge
[Denoising]			
steps:	int	1000	iteration steps. 1000 should suffice and can be reduced in case of weak noise to save computation time.
alpha:	float	0.01	reduced smoothing across edges, should be 1-2 orders less than beta.
beta:	float	0.1	smoothing strength for first order TV (norm of gradient), depends on noise level.
gamma:	float	0.1	smoothing strength for second order TV. Use a small value to prevent stair-casing. Use 0 to restrict to first order TV.
weight_factor:	float	3.0	choose a larger parameter to increase the weight for the depth data in regions with low intensity. Decrease the value if there are regions with low intensity and high noise, which is not smoothed out.

Setting the thresholds for EGDE-*: For each edge image a provisional result is given by the files *edge*_rawfile.cimg* (provided that in the parameter file *additional_output: yes* is set). These files show the results of the edge detection before the hysteresis step. Edge pixels above the lower threshold are set to 1, pixels above the higher threshold are set to 2. This data can be used to set the thresholds of the different edge detections to appropriate values.
For instance, all edges with no pixel above the higher threshold are removed.

7 Example Data Sets

We provide two data sets *hcibox* and *shape* in a separate zip-file. Each data set consists of intensity and depth data stored in .cimg files, a calibration file and a parameter file. Note that these data are not part of the software package and thus do not fall under the GPL license. (For details, see separate license file).

To process these data, enter the directory containing the parameter and data

files and, from this directory, call ⟨*path-to-software*⟩*/tof_denoising parameter-file.par*. For example for the *hcibox* data, assuming that they are located in *./data/hcibox*, enter this subdirectory and call *../../bin/tof_denoising hcibox.par*.

In case that you use one of these data sets in a publication, we kindly ask you to cite [4] (hcibox) or [5] (shape), respectively.

8 Tools

We provide some additional tools to display/postprocess the results of *tof_denoising*:

Matlab routine *read_dat* to read .dat-files: *data=read_dat(filename);* can be used in MATLAB to read a .dat-file produced by tof_denoising (use output_file: name.dat in the parameter file). Upon read success this routine returns a matlab matrix (2D, double values).

Simple viewer to display cimg files:
Usage: *cimg_viewer cimgfile [minvalue maxvalue].*
The data are clipped to the range [minvalue maxvalue], if these optional parameters are provided. With an open display the image can be enlarged/shrunk by pressing *enter* or *del*, respectively. Press *r* to re-load the cimg-file. To quit, just close the display.

Tool to convert standard images to cimg:
Usage: *convert2cimg input-file cimg-file.*
The input image is converted into a one-channel cimg-image with float values. Input image formats are those supported by the CImg-library. Color images are averaged over all three channels. Note that the ending of the output file is not checked to match .cimg.

Compilation of *cimg_viewer* and *convert2cimg*: create a *build* directory in *tools/*, and, inside, call *cmake ..* and *make*. The executables are stored in *./bin*.

Acknowledgements

This work was co-funded by the Intel Visual Computing Institute, Saarbrücken, Germany. The content is under the sole responsibility of the authors.

References

[1] M. Grzegorzek, C. Theobalt, R. Koch, and A. Kolb, editors. *Time-of-Flight and Depth Imaging: Sensors, Algorithms, and Applications*, volume 8200 of *LNCS*. Springer, 2013.

[2] D. Lefloch, R. Nair, F. Lenzen, H. Schäfer, L. Streeter, M. J. Cree, R. Koch, and A. Kolb. Technical foundation and calibration methods for time-of-flight cameras. In Grzegorzek et al. [1], pages 3–24.

[3] F. Lenzen, H. Schäfer, and C. S. Garbe. Denoising time-of-flight data with adaptive total variation. *Advances in Visual Computing*, pages 337–346, 2011.

[4] R. Nair, S. Meister, M. Lambers, M. Balda, H. Hoffmann, A. Kolb, D. Kondermann, and B. Jähne. *Ground Truth for Evaluating Time of Flight Imaging*, chapter 4. Volume 8200 of Grzegorzek et al. [1], 2013. to appear.

[5] H. Schäfer, F. Lenzen, and C. S. Garbe. Depth and intensity based edge detection in time-of-flight images. In *Proceedings of 3DV*. IEEE, 2013.