Light-Field Viewer

Designed by Maximilian Diebold
HCI

June 2015

1 Introduction

This toolbox allows to visualize vertical, horizontal and cross structured light field. The light fields are displayed either as animation or as EPIs at the selected position in the center viewer. In this viewer it is possible to interactively change the horopter real time (EPI mode) and apply it to the entire light field to watch the related animation afterwards. Additionally it is possible to save all shifted images to generate an gif animation of the underlying light field at a given Horopter. This all is a very supporting in terms of light field analysis, especially the zooming function in the animation window.
Aside this also the openCV camera calibration for single cameras is implemented. To use the calibration, a modified version of the xml-config file, provided by openCV needs to be loaded to unlock the calibration features in the control terminal.
At last, the structure tensor orientation estimation with occlusion handling is implemented and can be executed in different modes such as single or accumulate mode.
A detailed explanation of each toolbox part is given in the following sections. For each part different examples are provided in the data-subfolder.

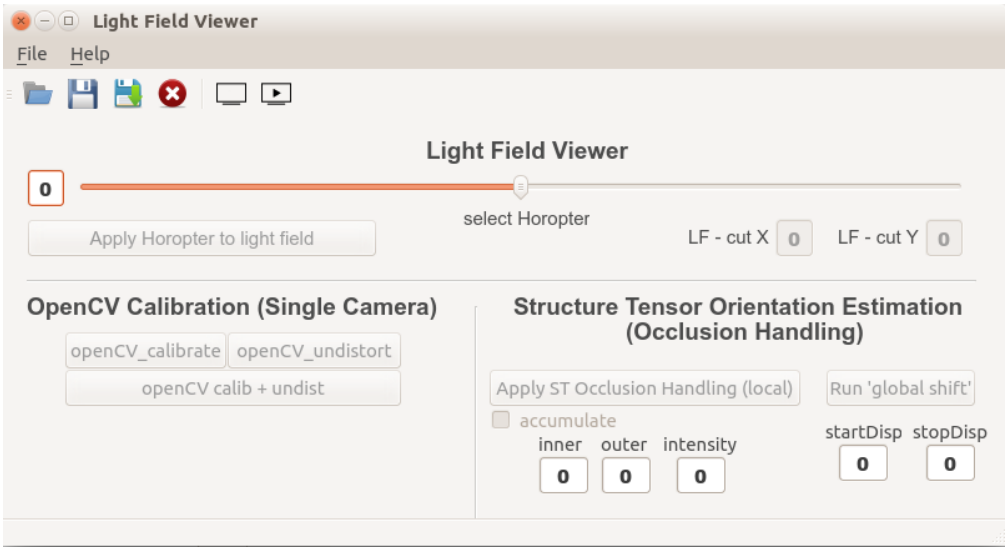

Figure 1: Illustrates the toolbox front end without any loaded ini-file. In the initial state no segments are unlocked. Dependent on the loaded configuration file different segments with its buttons will unlock.

2 Light-Field Viewer

To visualize light fields all related images need to be located in a folder which is defined in the ini-file, see figure 2. In addition also the type of the light field is a necessary information to display the data correctly. The viewer supports three different kind of light fields which are horizontal, vertical or cross light-fields. To load cross light fields it is important to consider that at the given location defined in the ini-file, a 'h'-folder for the horizontal and a 'v'-folder for the vertical light field direction is present in which the related images are located. This is important, because in contrast to horizontal or vertical light fields, for cross light fields the software searches the image data in the two described sub-folders instead of the current location. As third parameter the horopter of the light field can be set optionally, to place the center position of the slider to an initial horopter value unequal zero. The last option is to invert the images direction. Image sequences having the inverted order are visualized correctly but the disparity becomes negative. To avoid this without renaming the images it is possible to invert the load order.

```
[GeneralInputs]
;directory of your image sequence (either absolute path, or relative to the executable)
imagefile="/home/mdiebold/LFrepos/tools/20150520_LFViewer/v1.0/data/Aloe"
;define if you want to load a horizontal, vertical or 4D light field
type="horizontal"
;set horopter
horopter=50
;inverts the image order while loading them in the light-field container
invert=0
```

Figure 2: Shows the content of an ini-file. For the visualization only the parameter are needed. The first is the location of the images. The second the type of light field and the third one is a preset horopter which is optional.

After loading the light field into the viewer the buttons and input fields of the light-field viewer as well as for the structure-tensor orientation estimation become available. To set the parameter for the structure tensor the values can entered directly in the input fields or can be preset in the ini-file as shown in figure 5. Aside the unlocked buttons also the main window opens, see figure 4 on the left side. It contains the center view of the light field and shows all available EPI (vertical and horizontal). The shown light file is part of the Middlebury dataset and can be seen as sparsely sampled light field containing 7 views in horizontal direction captured. Thus the horizontal EPI contains information from 7 views while the vertical EPI just contains one column. When the main window gets closed it can be re-opened by clicking the monitor button in the control terminals menu bar. The other monitor button, having the play symbol inside starts the light field animation for the applied horopter which opens in its own window, as

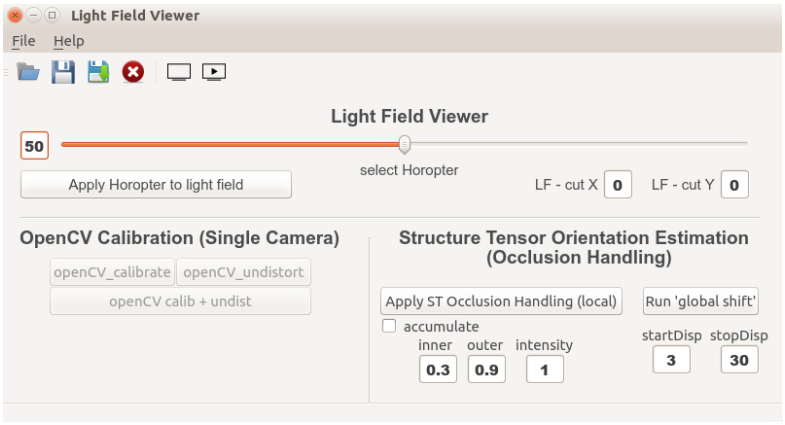

Figure 3: Shows the front end of the LF Viewer after loading an ini-file. The buttons and input areas of the light-field viewer and the structure tensor orientation estimation section are available now.

shown in figure 4 on the right side. This animation window is controllable with some keys and the mouse to evaluate the light fields in more detail.

The available keys are:

ESC	Close Window
S	Display next image and stop (to leave press any other key)
Space	Change from vertical in horizontal direction and back.
Mouse wheel	Zoom in and out
Mouse click and hold	Move the image position (in zoomed images)

To see the light field at a different horopter it is important to press the button "Apply Horopter to light field". After pressing the button the selected Horopter gets applied to the entire light field. The animation window updates the output automatically if the computation of the new horopter is finished. While watching the animation or when saving the light field at different horopter, it is possible to cut the light field in vertical (LF-cut Y) or horizontal (LF-cut X) direction to remove the appearing black borders due to the applied horopter.

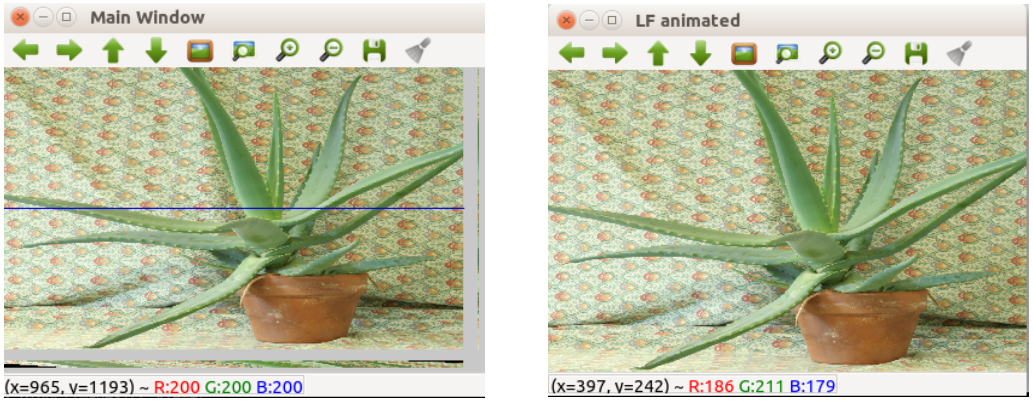

Figure 4: Illustrated the toolbox front end without any loaded ini-file. In the initial state no segments are unlocked. Dependent on the loaded configuration file different segments with its buttons will unlock.

3 Structure Tensor Orientation Estimation with Occlusion Handling

For each loaded light field also the structure tensor orientation estimation with occlusion handling can be applied. Thus it is important to set the result location in the ini-file, as shown in figure 5. At this location the results of the structure tensor computation is saved automatically which is important to set because there is no visualization output of structure tensor result. By pressing the button "Apply ST Occlusion Handling(local)" the algorithm computes the disparity for the light field at the given horopter and saves the result. When changing the horopter and applying the algorithm again by pressing the button "Apply ST Occlusion Handling(local)" the old result gets overwritten. To avoid this the accumulate flag needs to be set. Now the results of each local estimation are combined by taking for each pixel the disparity value having the largest coherence value. To compute light fields having a large disparity range one can also define the start and the stop disparity and press "Run 'global shift'". The algorithm computes now the light field in the entire range automatically and merges the views.
The implemented structure tensor uses bilateral filter instead of Gaussian filter which is important for the occlusion handling. Because of this an additional intensity threshold can be set. To preset the parameter, it is also possible to define the parameter in the ini-file as already shown in figure 5

```
[StructureTensor]
;directory of your results(either absolute path, or relative to the executable)
resultLocation="/home/mdiebold/LFrepos/tools/20150520_LFViewer/v1.0/demo"
;baseline between two neighboring cameras
baseline=0.016
;focal length of the camera
focalLength=3740
;start disparity (smaller one)
dispStart=3
;stop disparity (larger one)
dispStop=30
;inner Gaussian Filter
innerSigma=0.3
;outer Gaussian Filter
outerSigma=0.9
;bilateral Filter Intensity threshold
bilateralThreshold=1
```

Figure 5: Illustrates the content of the ini-file with parameters for the structure tensor section.

4 OpenCV Camera Calibration

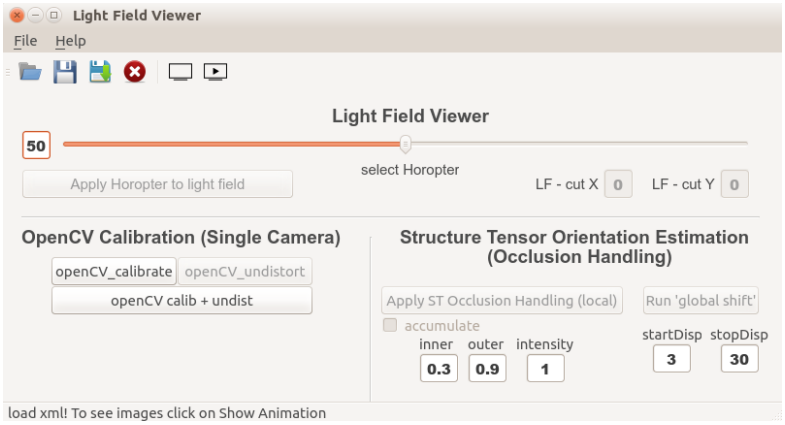

Figure 6: Illustrated the toolbox front end without any loaded ini-file. In the initial state no segments are unlocked. Dependent on the loaded configuration file different segments with its buttons will unlock.

To use the openCV camera calibration, a XML-file as shown in figure 7 is needed to set all needed parameter. The first parameter "CalibrationImages" defines the location of the calibration images while the second parameter "ImagesToRectify" defines the location of the images used for the undistortion. All other parameter are calibration related and explained in the XML-file itself. The calibration result is saved in the same folder as the xml-file is located and is named as defined by the parameter "Write_outputFileName". A calibration result is shown in figure 8. Saved is the camera matrix and the distortion coefficients also the path to the images to undistort as already defined in the xml-file. This parameter value appears at this position again to apply the undistortion with the already computed

calibration to different image sequences without recomputing the calibration each time. Thus only the file, as named at "Write_outputFileName" in the xml-file, containing the calibration result needs to be reloaded. Then the "openCV_undistort" button unlocks and the undistortion can be applied to the related images. The undistortion also unlocks right after the calibration is finished successfully. To run both at once the button "openCV_calib + undist" needs to be clicked where right after the calibration the images get undistorted.

```
<?xml version="1.0"?>
<opencv_storage>
  <Settings>
     <!--Images which should be calibrated-->
      <CalibrationImages>"/home/mdiebold/LFrepos/tools/20150520_LFViewer_Linux/LFViewer_Linux/data/calib_data/Images"</CalibrationImages>
     <!--Images which should be rectified (optional if needed)-->
      <ImagesToUndistort>"/home/mdiebold/LFrepos/tools/20150520_LFViewer_Linux/LFViewer_Linux/data/calib_data/Images"</ImagesToUndistort>
      <!-- Number of inner corners per a item row and column. (square, circle) -->
      <BoardSize_Width>8</BoardSize_Width>
      <BoardSize_Height>6</BoardSize_Height>
      <!-- The size of a square in some user defined metric system (pixel, millimeter)-->
      <Square_Size_height>26.4</Square_Size_height>
      <Square_Size_width>26.4</Square_Size_width>
      <!-- The type of input used for camera calibration. One of: CHESSBOARD CIRCLES_GRID ASYMMETRIC_CIRCLES_GRID -->
      <Calibrate_Pattern>"CHESSBOARD"</Calibrate_Pattern>
      <!-- The input to use for calibration.
      To use an input camera -> give the ID of the camera, like "1"
      To use an input video  -> give the path of the input video, like "/tmp/x.avi"
      To use an image list   -> give the path to the XML or YAML file containing the list of the images, like "/tmp/circles_list.xml"
      -->

      <!--  If true (non-zero) we flip the input images around the horizontal axis.-->
      <Input_FlipAroundHorizontalAxis>0</Input_FlipAroundHorizontalAxis>
      <!-- Consider only fy as a free parameter, the ratio fx/fy stays the same as in the input cameraMatrix.
      Use or not setting. 0 - False Non-Zero - True-->
      <Calibrate_FixAspectRatio>1</Calibrate_FixAspectRatio>
      <!-- If true (non-zero) tangential distortion coefficients are set to zeros and stay zero.-->
      <Calibrate_AssumeZeroTangentialDistortion>0</Calibrate_AssumeZeroTangentialDistortion>
      <!-- If true (non-zero) the principal point is not changed during the global optimization.-->
      <Calibrate_FixPrincipalPointAtTheCenter>0</Calibrate_FixPrincipalPointAtTheCenter>
      <!-- The name of the output log file. -->
      <Write_outputFileName>"_calibration_data.ini"</Write_outputFileName>
      <!-- If true (non-zero) we write to the output file the feature points.-->
      <Write_DetectedFeaturePoints>0</Write_DetectedFeaturePoints>
      <!-- If true (non-zero) we write to the output file the extrinsic camera parameters.-->
      <Write_extrinsicParameters>0</Write_extrinsicParameters>
  </Settings>
</opencv_storage>
```

Figure 7: Shows the modified version of the openCV XML-file to calibrate single cameras. It is used to apply the calibration with its necessary parameter but also defines a location to apply the image undistortion right after the calibration.

```
[DATUM]
calibration_Time="06/14/2015 10.20"
[cameraMatrix]
fx=3042.8690846586405
fy=3042.8690846586405
cx=1857.2076630083575
cy=1223.4096798567984
sx=0
[Distortion_Coefficients]
k1=-0.083887834996867375
k2=0.072968072879188911
k3=0.00034205994191790351
k4=0.0011665931442137515
k5=0.045255389626470945
Avg_Reprojection_Error=0.73784759889307749
[images to rectify]
imagefile="/home/mdiebold/LFrepos/tools/20150520_LFViewer_Linux/LFViewer_Linux/data/calib_data/Images"
```

Figure 8: Shows the result of the openCV camera calibration. It is automatically saved in a file called _calibration_data.ini. It contains the camera matrix values and the distortion coefficients necessary to un-distort the images.