Available online at www.sciencedirect.com
ScienceDirect
SoftwareX ▮ (▮▮▮▮) ▮▮▮–▮▮▮
SoftwareX
www.elsevier.com/locate/softx

Light-Field Imaging Toolkit

Jeffrey Bolan, Elise Hall, Chris Clifford, Brian Thurow*

Advanced Flow Diagnostics Laboratory, Auburn University, AL, United States

Received 6 August 2015; received in revised form 19 May 2016; accepted 25 May 2016

Abstract

The Light-Field Imaging Toolkit (LFIT) is a collection of MATLAB functions designed to facilitate the rapid processing of raw light field images captured by a plenoptic camera. An included graphical user interface streamlines the necessary post-processing steps associated with plenoptic images. The generation of perspective shifted views and computationally refocused images is supported, in both single image and animated formats. LFIT performs necessary calibration, interpolation, and structuring steps to enable future applications of this technology.

Keywords: Plenoptic; Light field; MATLAB

Code metadata

Current code version	*V2.31*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-15-00046
Legal Code License	*GNU General Public License, version 3 (GPL-3.0)*
Code versioning system used	*git*
Software code languages, tools, and services used	*MATLAB*
Compilation requirements, operating environments & dependencies	*Requires MATLAB 2009a or higher on Microsoft Windows or Linux*
If available Link to developer documentation/manual	*See included documentation PDF at* https://github.com/AFDL/LFIT
Support email for questions	thurow@auburn.edu

1. Motivation and significance

Recently, the technique of light-field imaging with a plenoptic camera has emerged [1–4], which is capable of capturing 3D information about a scene in a single snapshot. This technique uses what is known as a plenoptic camera to capture the four-dimensional (4D) light field—the spatial and angular information associated with the distribution of light rays in a volume [5]. A plenoptic camera is identical in most respects to a conventional camera; the key difference lies in the addition of a microlens array. This planar array of microlenses is inserted inside the camera a short distance in front of the image sensor. Light rays passing through the main lens of the camera are focused onto the microlens array. Depending on the incident angle of a given light ray on the main lens, the ray will be focused by the microlens it strikes onto a different pixel on the image sensor. In this way, both the angular and spatial information in a scene are sampled. This combined spatial and angular data can be computationally processed to generate new, unique images of the volume from a single raw plenoptic image. This allows for 3D information to be collected using an otherwise typical camera. The potential applications of this tool are myriad, ranging from microscopy [6] to 3D particle image velocimetry [7] to range finding [8,9].

While most post-processing is largely optional in a conventional camera, raw plenoptic images require significant post-processing to be of any use. A plenoptic camera uses a standard 2D image sensor but encodes the aforementioned spatial and angular data into thousands of microimages formed by the microlens array. A given microimage corresponds to a single spatial sample of the light field, where each pixel within the given

* Corresponding author.
E-mail address: thurow@auburn.edu (B. Thurow).

http://dx.doi.org/10.1016/j.softx.2016.05.004
2352-7110/© 2016 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

microimage corresponds to a different angular sample. To extract this information, spatial and angular coordinates must be associated with each pixel in the image. In essence, although a plenoptic camera may be used identically to a traditional camera, special software is required to extract 3D data from the 2D camera sensor. This need for specialized software, which prior to this work was not openly available, is a significant barrier for researchers wishing to utilize a plenoptic camera in their work.

The Light-Field Imaging Toolkit (LFIT) was written to streamline and standardize the processing of raw plenoptic images. Initially, LFIT was created in 2014 for use in the Auburn University Advanced Flow Diagnostics Laboratory (AFDL) by combining several independently developed codes internal to the lab. Since then, LFIT has been enhanced and extended across a number of versions. Today, LFIT is composed of dozens of functions and several thousand lines of code, with a fully functioning graphic user interface (GUI). This suite of functions is now provided to assist other groups working with plenoptic data. This paper provides a high level overview of LFIT and its functionality.

2. Software description

LFIT is a suite of MATLAB functions, each of which facilitates some aspect of the processing of a raw plenoptic image. A single main demonstration script is provided with the package, which invokes the graphical user interfaces (GUI) and calls the various included functions as appropriate. Most users will be most comfortable with the GUI mode, although a scripted batch mode is also included for automated processing. It should be noted that the individual functions can be used independently of the main demonstration script. That is, all the functionality of LFIT can be accessed via the command line or user-written MATLAB scripts that call LFIT functions. For specialized applications, this may be the preferred route.

Broadly speaking, the overall task of processing a raw plenoptic image can be categorized into these steps: Import, Microlens Calibration, Radiance Array Formation, and Computational Imaging. These steps are described in more detail below. Additional practical details are given in the documentation included with LFIT. An exhaustive description of the theory behind light field imaging with a plenoptic camera is beyond the scope of this paper; for an extended discussion of the theory from a practical perspective with regard to LFIT, refer to [10]. Also note that the nomenclature used in LFIT is largely adopted from Ng; see [4,11] for a full description of light field imaging.

2.1. Import

LFIT first requires that several basic parameters be defined before processing any images. These parameters are defined in the first GUI window, shown below in Fig. 1. Specifically, a directory containing calibration image(s) associated with the raw plenoptic image(s) to be processed is defined. The raw plenoptic image directory and output file paths are also specified. The magnification at the nominal focal plane is input by the user, along with the focal length of the main lens. Other parameters such as the focal length of the microlens array are automatically loaded when specifying the plenoptic camera used in the experiment. Alternatively, the user may select the 'other camera' option and will be prompted to input these parameters.

2.2. Microlens calibration

After the above parameter definitions, the first step in post-processing a plenoptic image is the microlens calibration step. To determine the center of each microimage, a series of calibration images are captured during the experiment of a white surface with the aperture of the main lens stopped down to a small size. LFIT first averages the calibration images to reduce the influence of sensor noise. After averaging, the user is prompted to identify three microlens centers in a specified pattern. The spatial relationship between these points is used to initiate a lens finding algorithm. This algorithm marches through each microimage in the scene, identifying the centroid of each spot. These centroids define the centers of each microlens.

2.3. Radiance array formation

The goal of this step is to structure the data from the raw plenoptic image into a 4D radiance array, indexed by (i, j, k, l) which correspond to (u, v, s, t) coordinates. The (u, v) coordinates are associated with the angular data, and the (s, t) coordinates are associated with the spatial data. With the center of each microimage identified via the above calibration, the pixels behind each microimage are then interpolated onto a uniform (u, v) grid because the microlens array is not perfectly aligned with the image sensor. From an algorithmic perspective, it is computationally more efficient to have the sampled (u, v) data lie on a uniform grid; otherwise, it would be later necessary to perform scattered data interpolation, which is a comparatively slow (factor of 10) process.

The known coordinates in the interpolation are derived from the calculated center of the microlens in the calibration step. In (u, v) coordinates, the microlens center will be located at $(0, 0)$. However, it is unlikely that the calculated center of the microlens perfectly aligns with a pixel in image coordinates. Because of this, the imaged pixels correspond to a sampling of slightly offset (u, v) coordinates. Thus, the known offset (u, v) values are associated with the microimage data. Using this, a new microimage can be created by interpolating for new values of (u, v) on a uniform integer-valued (not offset) grid. This process is repeated for every microimage in the raw plenoptic image, as each microimage will be non-uniformly offset relative to the center of its respective microlens. For cameras with a rectangular array, the (s, t) coordinates for the microimages are determined by assuming a non-rotated (s, t) grid centered on the center microimage. For cameras with a hexagonal array, it is necessary to resample the scattered microimage (s, t) data onto a rectilinear grid for algorithmic purposes. For both types of (u, v) interpolation, the standard MATLAB function *interpn* is used. For the hexagonal array type requiring (s, t)