SIMToolbox 2.0: MATLAB toolbox for structured illumination fluorescence microscopy

Supplementary documentation and user's guide

Pavel Křížek,[1] Tomáš Lukeš,[2,3] Martin Ovesný,[1] Jakub Pospíšil,[2] Vojtěch Terš,[2] Karel Fliegel,[2] and Guy M. Hagen[4,*]

[1]First Faculty of Medicine, Charles University in Prague, Prague, Czech Republic
[2]Department of Radioelectronics, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic
[3]Laboratoire d'Optique Biomédicale LOB, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
[4]UCCS center for the BioFrontiers Institute, University of Colorado Colorado Springs, Colorado Springs, Colorado, USA.
[*]ghagen@uccs.edu

About this text

This is the supplementary documentation for a MATLAB toolbox which provides tools for processing data acquired by structured illumination microscopy (SIM). The toolbox contains methods for processing both optical sectioning (OS-SIM) and super-resolution (SR-SIM) data, and for calibration and recreation of arbitrary illumination patterns in the camera image. A graphical user interface is also available, as are repertoires with line-grid and dot-grid illumination patterns. The toolbox is distributed under the terms of the GNU GPLv3 license and the source code and the associated documentation are available at the project website: http://mmtg.fel.cvut.cz/SIMToolbox.

Acknowledgments

This work was supported by the UCCS center for the BioFrontiers Institute and by the Czech Technical University in Prague[1]. T.L. acknowledges a SCIEX fellowship[2]. This project was also supported in part by the Czech Science Foundation. The authors thank the following people for their contributions to this project: Zdeněk Švindrych, Lubomír Kováčik, Michal Křížek, Evgeny Smirnov, Josef Borkovec, and Christian Lanctôt. The authors thank Carl Zeiss (Munich, Germany) for access to the Zeiss Elyra system, and the Institute of Molecular Genetics (Prague, Czech Republic) and BIOP lab at EPFL, (Lausanne, Switzerland) for access to the Nikon N-SIM system.

[1] SGS14/148/OHK3/2T/13, SGS16/167/OHK3/2T/13, and SGS18/141/OHK3/2T/13
[2] Project code 13.183

Contents

1 About

Structured illumination microscopy (SIM) works by acquiring a set of images at a given focal plane using widefield detection. Each image in the set is made with a different position of an illumination mask but with no mask in the detection path [1]. Subsequent image processing is needed to yield an optically sectioned (OS-SIM) image [1–3] or a super-resolution (SR-SIM) image [4, 5], i.e., an image with resolution beyond the Abbe limit. Here, we present a MATLAB toolbox for processing data acquired by structured illumination microscopy. The toolbox contains methods for processing both OS-SIM and SR-SIM data. Finally, SIMToolbox makes available an alternative method for SR-SIM, maximum a posteriori probability estimation (MAP-SIM) [6]. SIMToolbox offers users a graphical user interface (GUI). Users can also use the supplied functions to create their own programs.

To increase both the flexibility and the performance of structured illumination microscopy, our group uses a reflective ferroelectric liquid crystal on silicon (LCOS) microdisplay (a spatial light modulator, or SLM) to create illumination patterns with arrays of lines and/or dots with an arbitrary spatial frequency [6–8]. This flexibility enables us to easily compromise between scanning speed (i.e., the number of patterns), the desired signal to noise ratio (SNR), and (in OS-SIM) the optical section thickness, and thus to find the most suitable SIM pattern for a given sample and optical setup. Use of a calibration procedure allows us to determine a mathematical model describing a one-to-one mapping between pixels of the microdisplay (used to create the illumination pattern) and pixels of the camera chip, and thus to recreate an arbitrary digital illumination mask in the acquired data [7]. Higher performance data processing methods can be used for image reconstruction if such a digital mask is available.

To facilitate use of SLMs in SIM, we provide detailed guidelines for building a SIM system with a LCOS microdisplay and LED (or laser) illumination, and a set of line-grid and dot-grid illumination patterns for optical sectioning and super-resolution SIM. The patterns are intended for our particular microdisplay, but can be adapted for other SLMs. SIMToolbox also offers an algorithm for spatial calibration of the microscope and methods to recreate the digital illumination mask in the camera image.

In order to achieve faster processing of long image sequences of live cells, we made a new implementation of MAP-SIM which takes advantage of computer graphics cards (GPUs) and the CUDA programming environment. This extension is freely available as a part of SIMToolbox.

1.1 Compatibility notes

The SIMToolbox GUI was compiled with MATLAB 2017a and tested in Windows 7, 8 and 10. The GUI is a stand-alone program and does not require MATLAB to be installed.

Approximate processing times measured using a conventional computer (Intel® Core™ i7-4770 CPU, 3.40 GHz; NVIDIA® Quadro® K2000 GPU, 2 GB; RAM 32 GB) are shown in the Table 1.

To use the MATLAB functions within SIMToolbox (i.e., without the GUI), MATLAB must be installed. The functions were mainly developed with 64bit MATLAB versions 2017a in Windows 10. When using SIMToolbox functions without the GUI, the MATLAB

"Image Processing Toolbox" is required. SIMToolbox also requires the "MATLAB YAML" package by Kota Yamaguchi [9] to convert MATLAB objects to/from YAML file format. Note that this package is installed automatically when using the GUI.

Table 1: Approximate processing times for different algorithms within SIMToolbox. Processing time was always measured for one 2D image. The name of the test image and its size is specified in the first column.

Sample	SR-SIM	MAP-SIM (CPU)	MAP-SIM (GPU)
Actin (green LED) LCOS (1392 x 1040px)	8.89 *s*	19.11 *s*	6.82 *s*
Actin_Nikon (512 x 512 px)	1.19 *s*	1.77 *s*	0.61 *s*
Mito_Zeiss (1280 x 1280 px)	7.68 *s*	14.30 *s*	5.47 *s*

1.2 Installation of the GUI (A standalone application)

1. Download the GUI installer from the project website: http://mmtg.fel.cvut.cz/SIMToolbox.
2. Un-ZIP the file and run "MyAppInstaller_web.exe".
3. MATLAB runtime is also required. If it is not already installed, it will be automatically downloaded and installed. "MyAppInstaller_web.exe" will guide you through the installation process.
4. We recommend following the default installation settings.

1.3 Installation of the MATLAB functions (not necessary if the standalone application GUI is used)

1. Unzip the file *SIMToolbox.zip* with the toolbox into your working directory. It also contains the YAML package [9]. The file can be downloaded from the project website: http://mmtg.fel.cvut.cz/SIMToolbox.
2. Open MATLAB and change the path to the toolbox directory.
3. Run the user interface by running the function *SIMToolbox*. It initializes all necessary paths by itself.

1.4 Input / Output

Input images need to be stored in 8-bit or 16-bit **.tif* files. We recommend recording the data using a 16-bit camera output. This is especially important for SR-SIM applications.

Generally, the convention for storing the data is one big file *name.tif* containing all acquired images in the following order: phase - angle -z plane - time. If your image stack has a different arrangement, you can use an option "prepare directory" which allows to reorder an input tif stack as described later in Section 2.4.

Data are stored in a directory called *SomePath\name*, where *name* is a user-specified name. The data directory has to also contain:

1. a description file about the data acquisition protocol which can be either an Andor IQ protocol *name.txt*, or a user-specified file *name.info*, see Section 6.1,

2. a pattern repertoire **.repz* containing SIM pattern information, see Section 4.2

The "prepare directory" feature can be used to automatically generate these files if they are not already present, see Section 2.4.

Results with optically sectioned and/or super-resolution images are saved to the directory *SomePath\name\results* as 32-bit **.tif* files. This can be changed in the function *config.m*.

1.5 Help with SIMToolbox

To get help using SIMToolbox, please consult this supplementary documentation. A detailed description of algorithms used for the data processing is also commented in the source code, including numerous examples of how to use the functions. Users can also consult our previous publications [6–8, 10].

1.6 Updating the toolbox

Please check the project website for updates. http://mmtg.fel.cvut.cz/SIMToolbox.

1.7 Reporting a bug

In the case of a malfunction, please let us know by email: lukestom@fel.cvut.cz. Please specify the malfunction, and describe the procedure to reproduce the error. Specify also your platform, i.e., MATLAB version, the operating system, and version of SIMToolbox.

2 User interface

SIMToolbox allows users to process SIM data from various systems and offers numerous settings for each step of the image reconstruction process. In this section, we introduce the graphical user interface (GUI) and show several examples of how to use the SIMToolbox for processing SIM data acquired by a) a SIM system using a LCOS microdisplay, b) Zeiss Elyra, c) Nikon N-SIM (or other SIM systems with a similar image acquisition strategy).

2.1 Workflow overview

Input data should be organized as a stack of images in a TIF file. The pattern angles and phases need to be organized in the following order: angle1 - phase 1 phase 2... phase n; angle 2 - phase 1 phase 2... phase n, etc. This order is then repeated for each Z section. If the stack contains more time points, the whole structure is further repeated for each time point.

The input image stack is stored in a folder with the same name together with a *repz* file (an archive) that contains all the illumination patterns and a text file that contains information about the acquisition. This folder is referred to as the "Data directory". After processing, the results are saved automatically into this folder. Together with the SIMToolbox, we have included pre-generated *repz* files for various illumination patterns and illumination strategies. We also make publicly available a package of MATLAB functions that allows one to generate repz files for any kind of illumination pattern of your choice. The text file with acquisition details can be exported from your scientific camera. At present, SIMToolbox is only able to automatically read text files generated by Andor IQ software.

If you acquired your SIM images from a commercial system like Zeiss Elyra or Nikon N-SIM and you do not have the acquisition data text file, SIMToolbox allows you to generate this description file together with a repz file with all the pattern information. As described below, the dialog menu "**Prepare data dir**" feature generates the aforementioned files and prepares a directory which is ready for further use.

2.2 Launching the program

SIMToolbox can be launched using MATLAB or as a standalone application which runs even without MATLAB installed (i.e., the GUI). If you want to launch the program from MATLAB, open and run the function called SIMToolbox. All the other functions will be automatically added into the MATLAB path. In order to use the standalone application, it is necessary to install an appropriate MATLAB runtime environment as decribed in Section 1.2.

2.3 Processing images acquired by a SIM system with a liquid crystal on silicon (LCOS) microdisplay

This section contains information pertaining to home-built SIM systems, particularly those using a high speed ferroelectric liquid crystal on silicon (LCOS) microdisplay (type SXGA-3DM, Forth Dimension Displays, Dalgety Bay, Scotland;

please see Section 4 for more details). However, some of the elements discussed here are also useful for processing data acquired with commercial systems. See below for more details about working with data acquired with commercial systems.

Use this test file: TestFiles/SIM_LCOS/Actin_LCOS

2.3.1 Opening a data directory

Figure 1 shows screenshots of the main menu (a) after startup and (b) during the selection of a pattern sequence. Most of the processing options are disabled after startup and they become available after the data are loaded. In this example, the directory already contains an image data stack in a TIF file, the metadata in a text file and a repz file containing the illumination patterns. In the main menu, users can directly load the directory by clicking on the "Data directory" button and choosing the right folder in a standard dialog window. Once the data directory is loaded, panel "Data" shows important information about the image stack such as the number of pixels, pixel sizes, number of planes along the z-axis, number of time points etc. The abbreviation "seq" (sequence) denotes the number of images in one pattern sequence. By "pattern sequence" we mean all pattern angles and phases for each pattern angle that are used to acquire SIM images at one z plane. In the example in Figure 1, the pattern sequence has 24 illumination patterns. A set of patterns with a particular timing setup is referred to as a "running order".

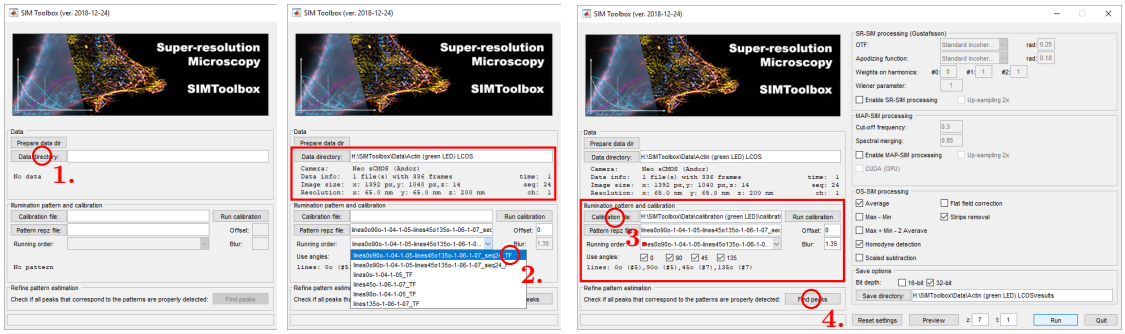
2.3.2 Illumination pattern information and opening a calibration file

Choose the following *running order*:
"lines0o90o-1-04-1-05-lines45o135o-1-06-1-07_seq24_TF".

Choose the following *calibration file*:
"calibration_LIN.yaml"

This name indicates that the pattern used in this experiment was used 5 phases at 0° and 90° and 7 phases at 45° and 135°. In total, the pattern sequence has 5+5+7+7 = 24 images. "TF" at the end of the running order name refers to a triggering scheme that has been used with the LCOS microdisplay during data acquisition. More details about the illumination patterns and an explanation of how users can create patterns with your own system can be found in the sections 4.2 and 4.2.4.

We next see checkboxes for all of the angles within the loaded running order. By changing these checkboxes, the reconstruction can use only the chosen pattern angles. In this example, the calibration is known. The calibration file can be opened by clicking on the "Calibration file" button. If there is a calibration file, we strongly recommend to use it. The calibration precisely describes the projection of the pattern from the LCOS microdisplay onto the sample. Therefore, the processing methods perform best when the calibration is in place. You can adapt the calibration procedure to your own SIM system which is equipped with a LCOS microdisplay or DMD. The calibration process is described in Section 5.2. After the calibration data are acquired, SIMToolbox allows users to analyze the data and to generate a calibration file using the button "*Run calibration*".



a b c

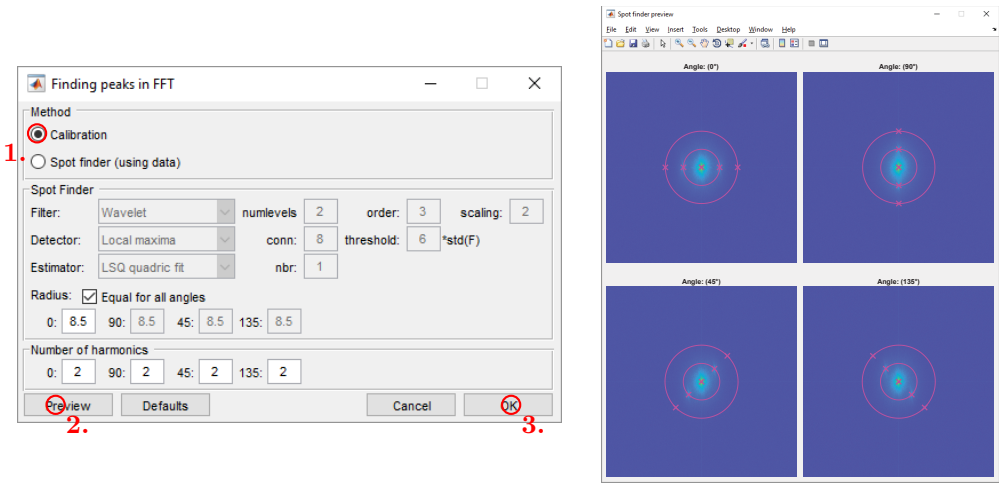
Figure 1: Main dialog of SIMToolbox. (a) Opening a data directory after startup. (b) Data properties and a selection of a running order (pattern sequence, if needed). (c) Details about the chosen illumination patterns and calibration (if available).

2.3.3 Refinement of pattern estimation

For the pattern estimation it is essential to detect the peaks in Fourier space which correspond to the pattern frequency and angle. The button "Find peaks" opens a dialog menu shown on Figure 2 where one can check if the peaks are detected properly. If the calibration file is provided, the pattern estimation uses the calibration information. In this example, the program automatically selected the calibration option as shown in Figure 2. The preview option allows users to inspect the positions of detected peaks in Fourier space. Detected peaks are marked by purple crosses. In Figure 2, we show detected peaks for 0°, 90°, 45°, and 135°. Pattern estimation from calibration is very precise and is thus preferred. **If the calibration is not known, patterns can be estimated directly from image data.** The "Find peaks" dialog menu contains numerous settings for precise tuning of the peak detection according to the investigated data. Sections 2.4 and 2.5 show how to adjust these settings for processing data acquired by Zeiss and Nikon SIM systems.

2.3.4 Image processing methods and results

SIMToolbox offers several processing methods for SIM data as described in details in Section 3. These methods include super-resolution algorithms SR-SIM (3.3.1), MAP-SIM (3.3.2), and optical sectioning algorithms (3.2). The "Run" button launches processing of the whole image stack. The program automatically creates a folder "Results" in the current data directory and saves all reconstructed images into this folder. The processing can take several minutes depending on the size of the input image stack and performance of your computer. We recommend use of the "Preview" option before processing the whole stack. The preview option shows results for one plane of the current stack (Figure 4). The z plane and time point that will be shown in the preview can be changed as shown in Figure 3.



a b

Figure 2: Estimation of illumination patterns using a calibrated microdisplay-camera system. (a) Dialog menu “Find peaks” for peak detection in Fourier space (b) preview of detected peaks for pattern angles 0°, 90°, 45°, and 135°. **The “spot finder” options will be necessary when processing data from commercial systems. The various options are provided to allow robust determination of the peak positions even in noisy data. In that case, users will have to experiment with these settings to correctly detect the peaks.**

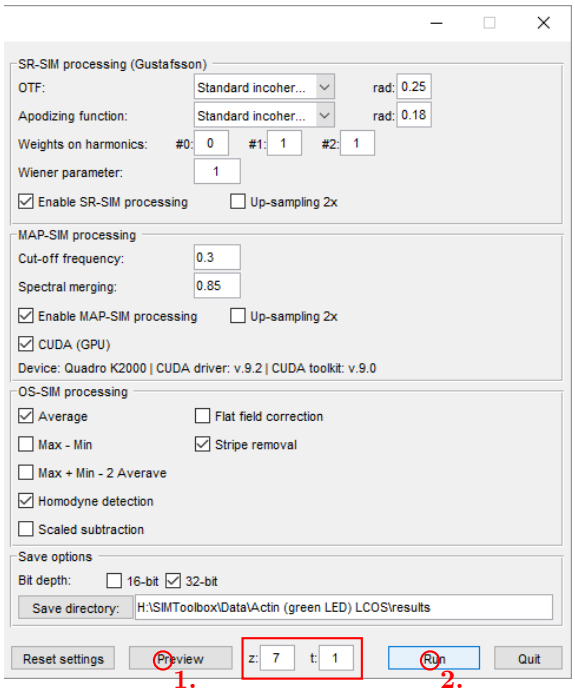


Figure 3: Choice of SIM methods and reconstruction parameters.

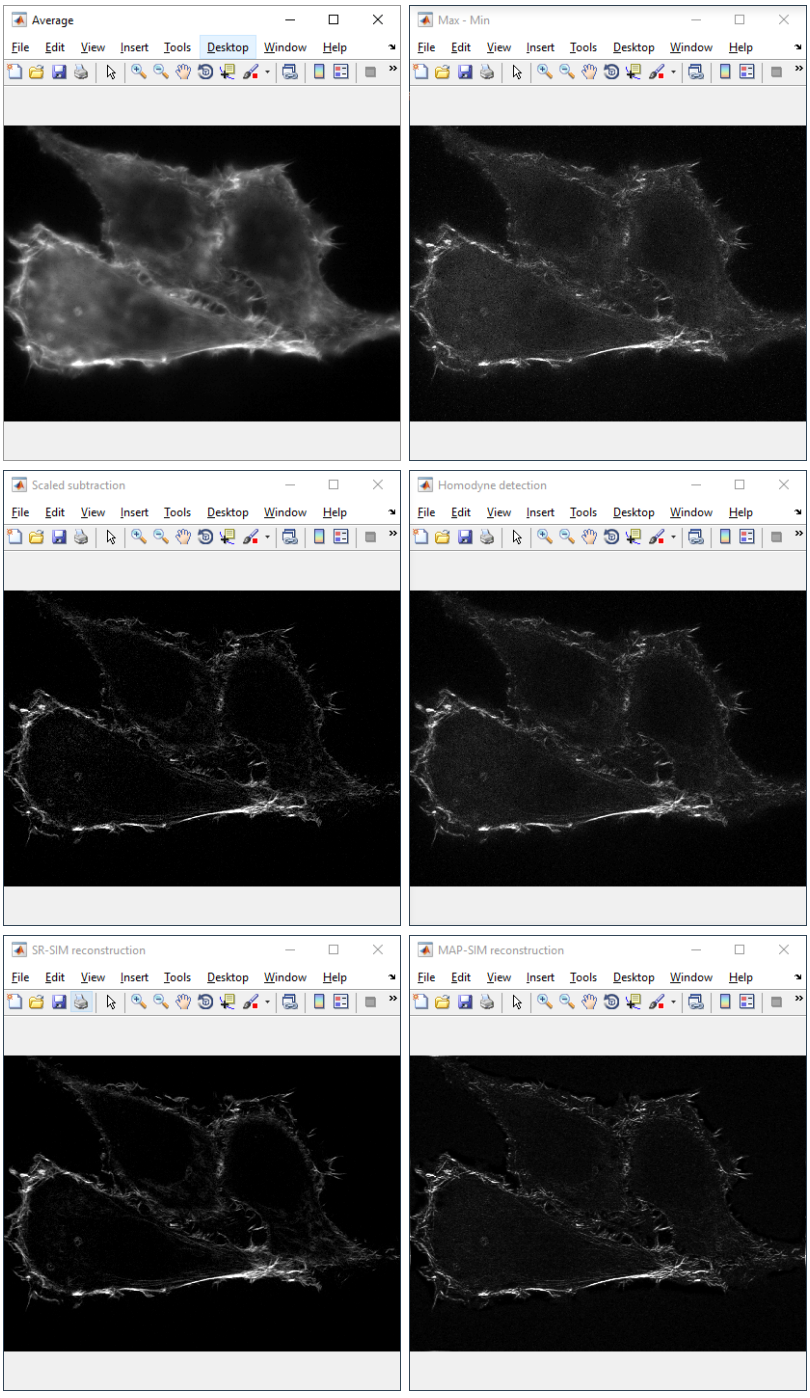


Figure 4: Results from one optical section of the test data from the LCOS SIM system using a calibration file (TestFiles/SIM_LCOS/Actin_LCOS). This data was used to create Fig. 1 of the main paper and Fig. 23 of this supplement.

2.4 Processing images acquired by Zeiss Elyra

Use this test file: TestFiles/Mito_Zeiss.tif

Working with data acquired by Zeiss Elyra is similar to the data processing described in the previous section. Users first need to to prepare the data directory. In the main menu, click the "*Prepare data dir*" button. Figure 5 shows dialog for data directory preparation. First select the image data and then fill in the rest of the parameters which can not be read automatically from the input data. Pattern angles can be approximate values, which can be estimated by inspecting the FFT of the images at each pattern angle using a program such as Image J and an on-screen protractor. The user can enter any value for the angles. These values are used only to refer to the angles in the main menu (check boxes "Use angles"). The precise values are estimated automatically during the pattern estimation process. The program creates a data directory with the same name as the input image stack. The image stack will be moved into this directory. During this process, the program also generates a "*repz*" file with pattern information and a description file with data information that have been entered into the "Prepare data dir" menu. If the arrangement of the input tif staf is different then phase-angle-z plane-time, you can rearrange the stack (check the checkbox). For rearrangement procedure, you have to specify current arrangement of the stack. Please make sure what is the arrangement of your stack and if the rearrangement is really necessary.

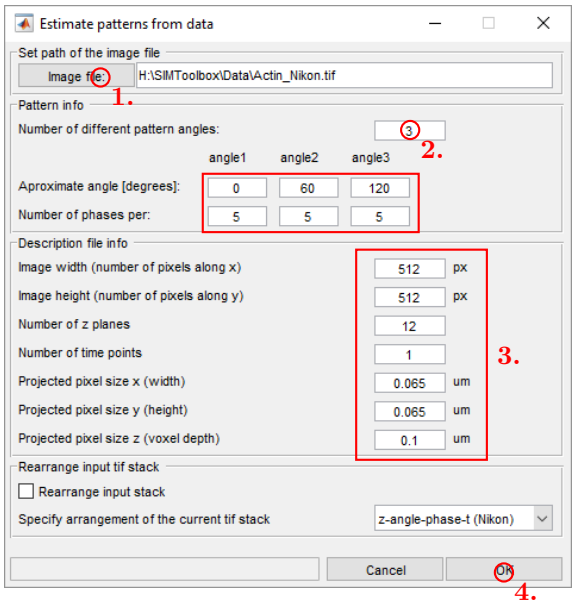
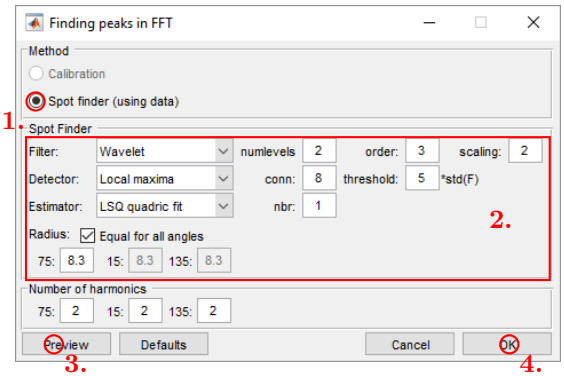
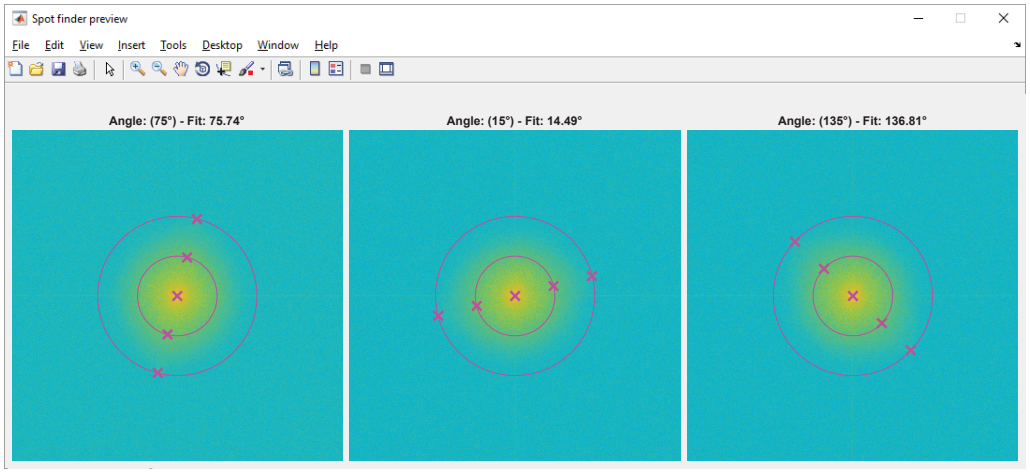


Figure 5: Menu "Prepare data dir". Select your tif stack, specify parameters. The program works with tif stack where the consecutive images fullfill the following arrangement: phases - angle - z plane - time. If the arrangement of your input tif stack is different, you can rearrange the stack. Please make sure what is the current arrangement of your input stack. If the input stack has an improper arrangement of images, the output super-resolved images will be full of strong, stripy artifacts!

When processing Zeiss or Nikon data, there will be no calibration available. Therefore, the patterns have to be estimated directly from the frequency spectrum of the data. To make sure that the patterns will be estimated correctly, please check the "Find peaks" menu (Figure 6). Users need to adjust the settings such that all of the peaks in the FFT that correspond to the pattern frequencies (and only the correct peaks) are detected, i.e. marked by purple crosses in the preview. An example of the results for one optical section of the test data "Mito_Zeiss.tif" is shown in Figure 7.



a



b

Figure 6: (a) Menu "Find peaks", the pattern parameters are estimated from the data, example of settings for Zeiss Elyra system. (b) Preview of peaks detected in Fourier space for one pattern angle.

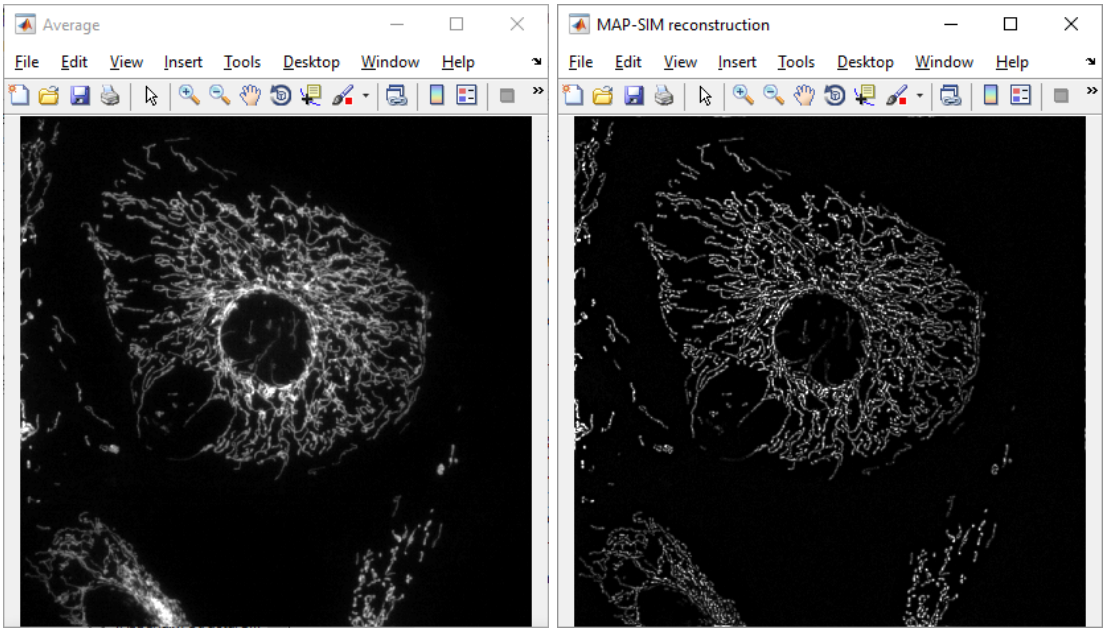
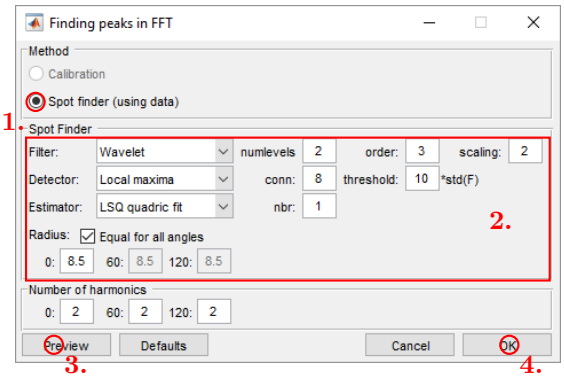


Figure 7: Results from one optical section of the test data from a Zeiss Elyra SIM system (TestFiles/Mito_Zeiss.tif).

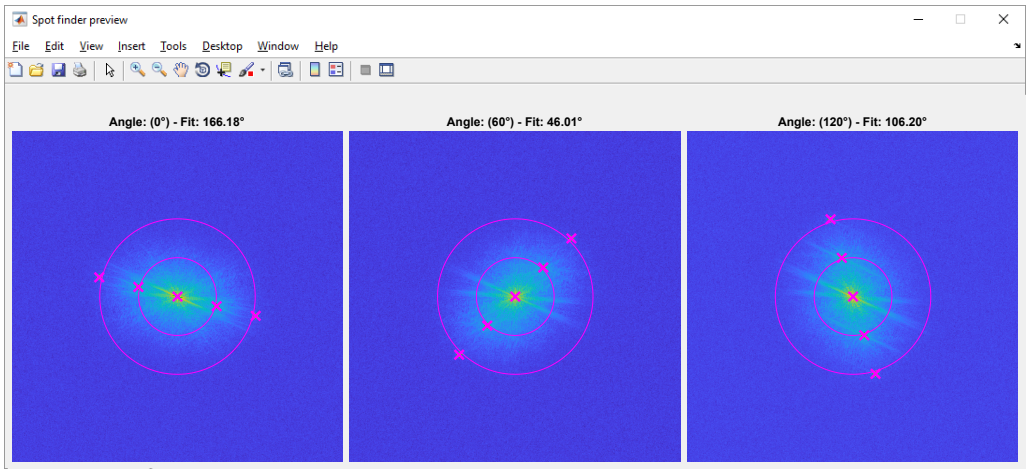
2.5 Processing images acquired by Nikon N-SIM

Use this test file: TestFiles/Actin_Nikon.tif

When processing data from Nikon N-SIM system, the process is the same as described in Section 2.4. The recommended settings for pattern estimation is shown in Figure 8. Figure 9 shows results for one optical section of the test image stack Actin_Nikon.tif.



a



b

Figure 8: Menu “Find peaks”, calibration is unknown and patterns are estimated from the data, example of settings for Nikon N-SIM system.

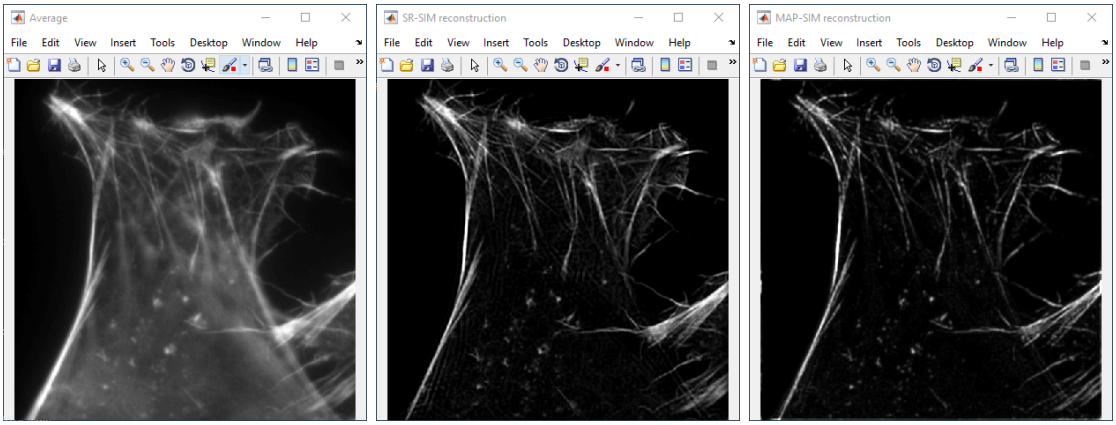


Figure 9: Results from one optical section of the test data from Nikon SIM system (TestFiles/Actin_Nikon.tif).

3 Methods for processing SIM data

3.1 Image preprocessing

3.1.1 Flat-field correction

Flat-field correction is a standard procedure used to suppress image artifacts caused by uneven illumination. The technique requires a flat-field image I_F obtained with full illumination (all microdisplay pixels ON) and a background image I_D obtained with all microdisplay pixels OFF. Flat-field correction of an image I is obtained by

$$I_C\left(u,v\right) = m\frac{I\left(u,v\right) - I_D\left(u,v\right)}{I_F\left(u,v\right) - I_D\left(u,v\right)}, \tag{1}$$

where (u,v) indicates the u,v pixel position in the camera image, and m is an average intensity value of the difference between the flat-field and background images $I_F - I_D$. Flat-field and background images can be obtained during the calibration process described in Section 5.5.

The toolbox function for flat-field correction of a stack or a sequence of images is:

```
IMseq = seqflatfield(IMseq, IMflat, IMdark);
```

3.1.2 Stripe removal

Images created by OS-SIM or SR-SIM reconstruction methods sometimes contain slight patterned artifacts. These are usually attributed to minor fluctuations of the light source or camera exposure time. To correct this, each image of one complete scanning sequence can be normalized such that the average intensity of all images in this sequence is the same. This procedure was suggested by Cole et al. [11] and gives satisfactory results. The appropriate function in the toolbox is:

```
IMseq = seqstriperemoval(IMseq);
```

3.2 Optical sectioning methods

Several computational approaches for obtaining optically sectioned images from SIM data are reviewed in [1]. Essentially, there are two approaches for data processing. The first type reconstructs optically sectioned images without any information except for the number of illumination patterns N. The second type requires, in addition, knowledge of the exact position of the illumination mask in the camera image.

If more than a single orientation of a line-grid illumination pattern is used to acquire the data, images with the same pattern orientation are processed independently and the results for different orientations are combined by averaging. This is not typically done in OS-SIM, but is easy to accomplish with DMD or LCOS-based microscopes. Such averaging should help make the resulting optically sectioned PSF more isotropic.

We note here that OS-SIM can actually improve lateral resolution compared to widefield microscopy (by a factor of about 1.4), similar to the confocal microscope [12].

3.2.1 Widefield image

Let $I_n(u,v)$ be intensity values of the camera image captured at a given frame n in a sequence of N illumination patterns, and (u,v) indicates the u,v pixel position in the camera image. A widefield image can be recovered from SIM data as an average of all images

$$I_{\mathrm{WF}}(u,v) = \frac{1}{N}\sum_{n=1}^{N} I_n(u,v). \tag{2}$$

The appropriate function in the toolbox is:

```
Iwf = seqwf(IMseq);
```

3.2.2 Simple methods

Optical sections can be recovered as follows

$$I_{\mathrm{C1}}(u,v) = \max_{n=1,\ldots,N} I_n(u,v) - \min_{n=1,\ldots,N} I_n(u,v)\ , \tag{3}$$

$$I_{\mathrm{C2}}(u,v) = \max_{n=1,\ldots,N} I_n(u,v) + \min_{n=1,\ldots,N} I_n(u,v) - \frac{2}{N}\sum_{n=1}^{N} I_n(u,v), \tag{4}$$

$$I_{\mathrm{C3}}(u,v) = \left|\sum_{n=1}^{N} I_n(u,v)\exp\left(2\pi \mathrm{i}\frac{n}{N}\right)\right|. \tag{5}$$

The approach described by Equation (3) applies maximum minus minimum intensity projection [13,14]. Here the "max" term is assumed to contain mainly contributions from parts of the sample that are in focus and the "min" term mainly contributions from out

of focus regions. The method in Equation (4) is claimed to further increase the image quality of optical sections [13]. Finally, the method in Equation (5) is a form of homodyne detection [3], which is a technique based on detecting frequency-modulated signals by interference with a reference signal. This is our prefered method when a calibrated illumination mask is unavailable.

We note here that not all of the possible processing methods are appropriate for all of the illumination masks. For example, when using dot-grid patterns, the homodyne detection method (5) will not work.

The appropriate functions in the toolbox are:

```
Ic1 = seqcfmaxmin(IMseq);
Ic2 = seqcfmaxmin2av(IMseq);
Ic3 = seqcfhomodyne(IMseq);
```

3.2.3 Methods with illumination mask (scaled subtraction methods)

The methods described in this section require knowledge of the exact position of the illumination pattern in the acquired data. The digital illumination mask can be determined by the approach described in Section 5. Optical sectioning methods available in the toolbox work on the following principles

$$I_{\mathrm{C4}}(u,v) = \sum_{n=1}^{N} I_n(u,v)\, M_n^{\mathrm{on}}(u,v)\,, \tag{6}$$

$$I_{\mathrm{C5}}(u,v) = \sum_{n=1}^{N} I_n(u,v)\, M_n^{\mathrm{on}}(u,v) - \frac{1}{1-N}\sum_{n=1}^{N} I_n(u,v)\, M_n^{\mathrm{off}}(u,v)\,, \tag{7}$$

$$I_{\mathrm{C6}}(u,v) = \frac{\sum_{n=1}^{N} I_n(u,v)\, M_n^{\mathrm{on}}(u,v)}{\sum_{n=1}^{N} M_n^{\mathrm{on}}(u,v)} - \frac{\sum_{n=1}^{N} I_n(u,v)\, M_n^{\mathrm{off}}(u,v)}{\sum_{n=1}^{N} M_n^{\mathrm{off}}(u,v)}\,, \tag{8}$$

$$I_{\mathrm{C7}}(u,v) = I_{\mathrm{C6}}(u,v)\, \frac{N\sum_{n=1}^{N} M_n^{\mathrm{on}}(u,v) - \left(\sum_{n=1}^{N} M_n^{\mathrm{on}}(u,v)\right)^2}{N\sum_{n=1}^{N} \left(M_n^{\mathrm{on}}(u,v)\right)^2 - \left(\sum_{n=1}^{N} M_n^{\mathrm{on}}(u,v)\right)^2}\,. \tag{9}$$

Here $M_n^{\mathrm{on}} \in [0,1]$ is the intensity of the (usually binary) digital illumination mask in the camera image for a given frame n, and $M_n^{\mathrm{off}} = 1 - M_n^{\mathrm{on}}$. The method in Equation (6) creates images corresponding to the conjugate (mostly confocal) light [15]. Methods in Equations (7)–(9) apply various forms of scaled subtraction of the out of focus light [1, 7, 13, 15–18].

The first term in all these methods represents the conjugate (mostly confocal) light and the second term the non-conjugate (mostly out of focus) light [15]. The method in Equation (7) is suitable only for non-overlapping pattern sequences, e.g., one pixel lines with one pixel shifts of the pattern. The method in Equation (8) corrects for possible pattern overlaps when using multi-pixel lines and shifts and thus is more universal. For

example, patterns using two pixel wide lines may be shifted by one pixel or by two pixels. The method in Equation (9) corrects for small pixel-to-pixel variations in the mask intensity (non-binary masks) [1]. We typically use Equation (7).

The appropriate functions in the toolbox are:

```
Ic4 = seqcfmaskonly(IMseq, Mask);
Ic5 = seqcfscaledsub1px(IMseq, Mask);
Ic6 = seqcfscaledsub(IMseq, Mask);
Ic7 = seqcfscaledsubbeta(IMseq, Mask);
```

3.3 Super-resolution methods

Super-resolution methods are limited to image stacks only with line-grid patterns. The available methods are SR-SIM (the Gustafsson-Heintzmann method) [4, 5] and MAP-SIM [6].

3.3.1 The Gustafsson-Heintzmann method

The method was developed and experimentally demonstrated by Gustafsson et al. [4] and by Heintzmann et al. [5]. The method has subsequently been adopted and expanded upon by several groups [6, 8, 19–33]. Let us denote the true fluorophore density of the specimen $f(x)$ and its Fourier transform $F(k)$. The observed images $g_n(x)$ using structured illumination are given by

$$g_n(x) = (m_n(x)f(x)) * h(x)\,, \tag{10}$$

where $h(x)$ is the point spread function $m_n(x) = 1 + \alpha\cos(2\pi(x \cdot p + \varphi_n))$ represents the n-th illumination mask and p is a modulation vector. After Fourier transformation, this equation can be rewritten as

$$G_n(k) = H(k)[F(k) * M_n(k)]\,. \tag{11}$$

The Fourier transform of the illumination mask is $M_n(k) = \delta(k) + \frac{\alpha}{2}\delta(k-p)e^{2i\pi\varphi_n} + \frac{\alpha}{2}\delta(k+p)e^{-2i\pi\varphi_n}$. Then by substitution we get

$$G_n(k) = H(k)\left(F(k) + \frac{\alpha}{2}F(k-p)e^{2i\pi\varphi_n} + \frac{\alpha}{2}F(k+p)e^{-2i\pi\varphi_n}\right). \tag{12}$$

To separate the spectral components at least three images with different phase shifts of the illumination pattern need to be acquired. With equally separated phase shifts $(0, 2\pi/3, -2\pi/3)$, the following system of equations can be solved for each pattern orientation:

$$\begin{bmatrix} G_1(k) \\ G_2(k) \\ G_3(k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{2i\pi/3} & e^{-2i\pi/3} \\ 1 & e^{-2i\pi/3} & e^{2i\pi/3} \end{bmatrix} \times \begin{bmatrix} H(k)F(k) \\ \frac{\alpha}{2}H(k)F(k-p) \\ \frac{\alpha}{2}H(k)F(k+p) \end{bmatrix}. \tag{13}$$

The solutions $C_n(k)$ represent the shifted spectra multiplied by non-shifted OTFs:

$$\begin{bmatrix} C_1(k) \\ C_2(k) \\ C_3(k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{-2i\pi/3} & e^{2i\pi/3} \\ 1 & e^{2i\pi/3} & e^{-2i\pi/3} \end{bmatrix} \times \begin{bmatrix} G_1(k) \\ G_2(k) \\ G_3(k) \end{bmatrix}, \tag{14}$$

these three separated components then need to be shifted:

$$\begin{aligned} CS_1(k) &= C_1(k) \\ CS_2(k) &= C_2(k+p) \\ CS_3(k) &= C_3(k-p) \end{aligned} \tag{15}$$

and combined through a generalized Wiener filter:

$$S(k) = \frac{\sum_{n=1}^{N} OTF_n(k)^* CS_n(k)}{\sum_{n=1}^{N} |OTF_n(k)|^2 + w}\,, \tag{16}$$

where $S(k)$ is the reassembled image in reciprocal space, OTF_n is the n-th shifted OTF of the imaging system (in analogy to Equation (15)), N is the number of acquired patterned illumination images and w is a small constant that controls the noise of the reconstructed image.

The reassembled image is then apodized with the desired OTF profile to decrease ringing artifacts caused by sharp edges of the new reassembled Fourier spectrum. The resulting image is finally transformed back to real space to get the super-resolution reconstruction $s(x)$.

The toolbox provides the following functions to proceed through the described steps:

```
% Calculate position of peaks of individual spectral components.
% This is important for the shifting step.
M = sim_findpeaksassign(imginfo, ptrninfo, calinfo, Z, cfg);

% Extract spectral components from a sequence of images.
C = sim_extract(G, M);

% Shift the extracted FFT components.
CS = sim_shiftspectra(G, M);

% Generate the shifted OTF, combine the shifted spectra and apodize
% the reassembled image. The parameter for a Wiener filter is stored in
% `cfg.wiener` and the settings of apodization are set in `cfg.apodize`.
s = sim_combine(sim\_addOTF(CS, M, OTF), cfg);
```

Apodization Apodization is used to shape the frequency spectra of the super-resolution images in order to suppress noise and other reconstruction artifacts. As we deal with images, two-dimensional rotationally symmetric low-pass filters are usually used. Traditionally, a triangle apodization is used [21,34]. To guarantee a non-negative PSF, apodization based on application of the Lukosz bound was suggested [35]. There are also other types of apodization and in the toolbox we provide several functions.

Let ω denote the spatial frequency, ω_c is the cut-off frequency, and $\bar{\omega} = \frac{\omega}{\omega_c}$ stands for the normalized spatial frequency. No apodization of the frequency spectra over the whole range of spatial frequencies ω is defined as

$$A_{\text{none}}(\omega) = 1\,. \tag{17}$$

Apodization using Butterworth and Gaussian shaped filters are defined as

$$A_{\text{Butter}}\left(\bar{\omega}\right) = \left(1+\bar{\omega}^{2n}\right)^{-1}, \tag{18}$$

$$A_{\text{Gauss}}\left(\bar{\omega}\right) = \exp\left(-0.5\left(\bar{\omega}\sqrt{2\ln 2}\right)^{2}\right). \tag{19}$$

Apodization by circular, triangle, cosine bell (Hann window), standard OTF, and Lukosz bound based functions are defined for $\omega \leq \omega_c$ by the following functions

$$A_{\text{circular}}\left(\bar{\omega}\right) = 1, \tag{20}$$

$$A_{\text{triangle}}\left(\bar{\omega}\right) = \left(1-\bar{\omega}\right)^{q}, \tag{21}$$

$$A_{\text{cosbell}}\left(\bar{\omega}\right) = \cos\left(\frac{\pi}{2}\bar{\omega}\right)^{2}, \tag{22}$$

$$A_{\text{standard}}\left(\bar{\omega}\right) = \frac{\pi}{2}\left(\arccos\left(\bar{\omega}\right) - \bar{\omega}\sqrt{1-\bar{\omega}^{2}}\right) \tag{23}$$

$$A_{\text{Lukosz}}\left(\omega,\omega_c\right) = \min\left\{\cos\left(\frac{\pi\omega}{\omega+\omega_c}\right), \cos\left(\frac{\pi\omega}{\omega+\sqrt{2}\omega_c}\right)\right\}, \tag{24}$$

otherwise it is zero. The shape of all apodization functions described above is shown in Figure 10 and the appropriate functions in the toolbox are called by a command:

```
A = apodize_NAME(siz, params);
```

where `NAME` stands for `none`, `butterworth`, `circular`, `cosbell`, `gauss`, `lukosz`, `standard`, `triangle`, or `file`, depending on the apodization filter.

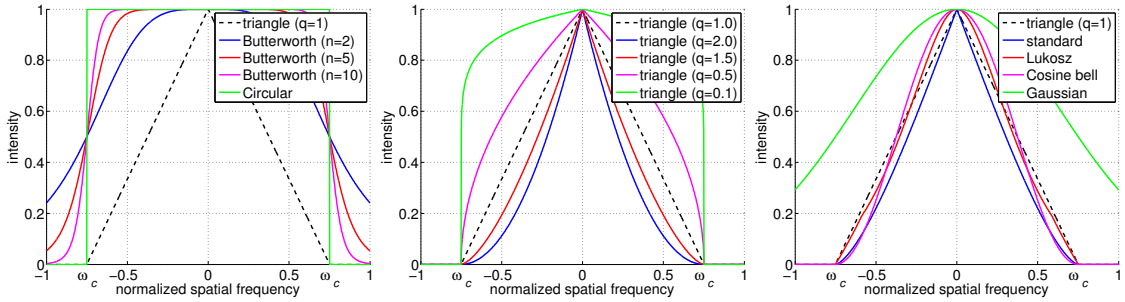


Figure 10: Profiles of apodization functions with $\omega_c = 0.75$.

3.3.2 MAP-SIM

Maximum a posteriori probability SIM (MAP-SIM) is a new, versitile image reconstruction method for super-resolution structured illumination microscopy [6]. The method was originally developed for SIM systems where the illumination patterns are projected on the sample using a microdisplay, but can also be used to process SR-SIM data. MAP-SIM provides resolution enhancement in all three dimensions using two-dimensional illumination patterns. The method is based on combining, via spectral merging in the frequency

domain, results from maximum a posteriori probability estimation (for resolution improvement) and results from homodyne detection (for optical sectioning). Image acquisition in structured illumination microscopy can be described as

$$y_k = HM_k x + n_k, \tag{25}$$

where $k = 1, \ldots, K$ indexes the sequence of illumination patterns, y_k denotes a vectorized (column stacked), diffraction limited, low-resolution (LR) image acquired by the camera using the k-th illumination pattern. x, and n_k represent an unknown, vectorized high-resolution (HR) image and additive noise, respectively. All of these vectors contain m elements. H is an $m \times m$ matrix which models the convolution between the HR image and the PSF of the system, and M_k is an $m \times m$ diagonal matrix in which the elements represent the k-th illumination pattern.

The PSF of a fluorescence microscope can be modeled as an Airy disk. In Fourier space this leads to an OTF of the form

$$OTF\left(f\right) = \frac{1}{\pi}\left[2\cos^{-1}\left(\frac{|f|}{f_c}\right) - \sin\left(2\cos^{-1}\left(\frac{|f|}{f_c}\right)\right)\right], \tag{26}$$

where f is the spatial frequency. The reconstruction of the HR-MAP image can be expressed as a minimization of the following cost function [6]

$$x_{\text{HR-MAP}} = \arg\min_{\text{x}}\left[\sum_{k=1}^{K}\left\|y_k - HM_k x\right\|^2 + \lambda\Gamma\left(x\right)\right], \tag{27}$$

where $\Gamma(x)$ is a regularization function [6]. The first term in Eq. 27 describes the mean square error between the estimated HR image and the observed LR images. The second term is the regularization term and its contribution is controlled by the parameter λ which is a small positive constant defining the strength of the regularization.

A high resolution image (HR-MAP), the result of MAP estimation (Eq. 27), contains details unresolvable in a widefield microscope, but does not provide suppression of out of focus light (optical sectioning). On the other hand, the homodyne detection method (Eq. 5) provides images (LR-HOM) with optical sectioning. In the MAP-SIM method, both LR-HOM and HR-MAP images are calculated and then merged in the frequency domain to obtain the final HR image (MAP-SIM). Low pass filtering is applied to the LR-HOM image and a complementary high pass filter is applied to the HR-MAP image. The weighting scheme is described by

$$\begin{aligned}\text{x}_{\text{MAP-SIM}} = \mathcal{F}^{-1}\Bigg\{&(1-\beta)\mathcal{F}\left\{\text{x}_{\text{LR-HOM}}\right\}\exp\left(-\frac{\text{f}^2}{2\rho^2}\right)\\ &+\beta\mathcal{F}\left\{\text{x}_{\text{HR-MAP}}\right\}\left(1-\exp\left(-\frac{\text{f}^2}{2\rho^2}\right)\right)\Bigg\},\end{aligned} \tag{28}$$

where $\mathcal{F}, \mathcal{F}^{-1}$ denotes the Fourier transform operator and its inverse, respectively, f is the spatial frequency, ρ is the standard deviation of the Gaussian filter, and β is a positive weighting coefficient. This coefficient can be adjusted in the GUI by changing the value of "Spectral merging". Recommended value should be in the range 0.7 - 0.9. For proper

spectral weighting of LR-HOM and HR-MAP images, the algorithm uses a normalized cut-off frequency of the widefield image which can be set in the GUI as "Cut-off" frequency parameter. Recommended value should be in the range 0.2 - 0.5. For more details about the MAP-SIM method please refer to [6].

The toolbox provides the following functions to proceed through the described steps:

```
% Reconstruct super-resolution image using MAP-SIM method if the ...
    calibration is not known - patterns are estimated from the data.
ptrn = sim\_findpeaksassign(imginfo, ptrninfo, calinfo, Z, cfg);
MaskOn = genMasks(seq,ptrn);
ptrninfo.MaskOn = MaskOn;
im.mapsim  = mapsim(seq(angles), MaskOn(angles), IMhom, cfg.msm);
% If the calibration is known - mapsim is calculated using the known ...
    illumination masks.
im.mapsim = mapsim(seq(angles), MaskOn(angles), IMhom, cfg.msm);
```

3.3.3 MAP-SIM using GPU

CUDA can significantly decrease processing time of MAP-SIM reconstruction. An average speed-up of real-life samples is about 20 times. However, this improvement depends on the graphics card used for the processing. The reconstruction speed-up will be probably lower on all mobile platforms. GPU processes most of the data in single-precision. Therefore, the results of GPU processing can be slightly different in compare with processing on CPU with double-precision. However, this difference should be acceptable in most scenarios.

The CUDA reconstruction uses the same resources for all Z-Planes. By doing so, we can eliminate the time needed to create and destroy these resources repeatedly. However, an attention is required when using CUDA processing core. Before processing all Z-planes, it is necessary to create these resources. When all Z-planes are reconstructed, the GPU memory must be released. See the calling convention:

```
MapcoreCudaPrepare([nx, ny, numSeq, maxIter, lambda, alpha, thresh, fc]);
for z = 1:Z
    ...
    IMmap = MapcoreCudaProcess(IMseq, OTF, Masks);
    ...
end
MapcoreCudaFinish();
```

where `IMseq` is the input sequence of illuminated images, `OTF` is OTF model of the microscope, `Masks` is corresponding image masks, `maxIter` is maximum numer of iterations, `numSeq` is the number of images per one Z-Plane, `lambda` is the normalization coefficient, `alpha` is the intial step for Barzilai-Borwein method, `thresh` is the convergence threshold, `fc` is the cut off frequency used for apodizaditon, `nx` and `ny` are X and Y-dimension of the image data (Number of columns and rows, respectively), and `Z` is the total number of Z-Planes.

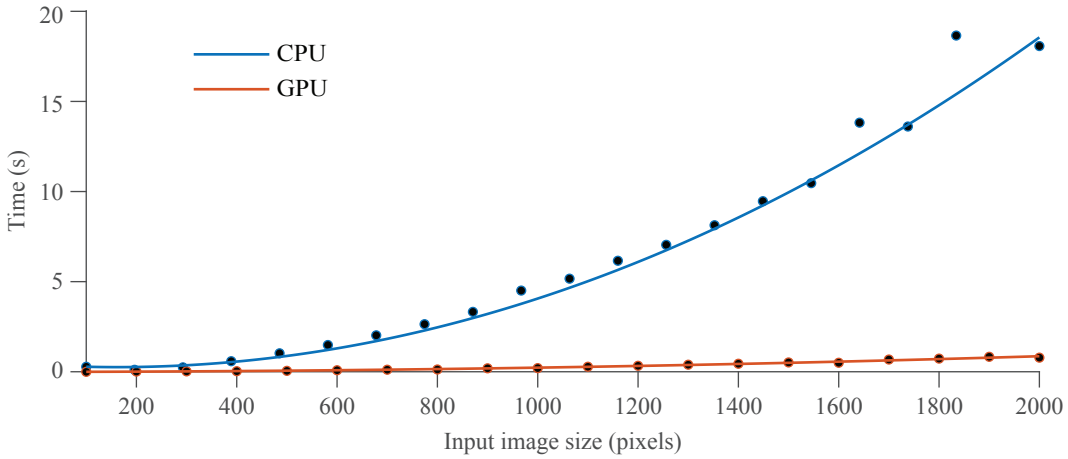


Figure 11: The processing times for various size inpput images when SIMToolbox is running with or without CUDA acceleration (2 iterations).

Testing rig:

- Motherboard: MSI P55-GD80
- CPU: Intel Core i5-750@2,67GHz
- GPU: MSI nVidia GeForce GTX960@1190MHz (ForceWare 355.60)
- Ram: 10GB DDR3@1600MHz CL9
- Matlab R2014a 64-Bit on Windows 7 SP1 (64-Bit)
- Mexfile Cuda Runtime: 6.5 Compiled with Visual Studio 2013

CUDA requirements:

- Graphics card with Compute Capability of 2.0 or higher (GeForce 400+)
- Relatively new nVidia drivers (version 6050+)
- 64-Bit Windows version
- Visual Studio 2013 C++ redistributable 64-bit

3.3.4 Note on TIRF-SIM

Total internal reflection fluorescence (TIRF) can be employed in SIM experiments [22]. In some cases, the illumination pattern spatial frequency that can be generated will be outside of the frequency passband of the objective. This is not because of Stoke's Shift, but rather because the NA for imaging in a TIRF objective is significantly lower than the NA that can be used for illumination. This has been reported for other high NA oil immersion objectives [36]. This phenomenon has been reported in the SIM literature as well (supplementary figure S2 of ref [22]). In SIM, this means that the excitation pattern may only barely be visible in the camera images, or may not be visible at all. As such, no peaks will be visible in the frequency spectrum. In this case, SIMToolbox will be able to process the data only if a calibrated microdisplay is used such as is described here. Without the calibration, SIMToolbox depends on finding peaks in the frequency spectrum. If the peaks are absent, as can occur in TIRF-SIM, SIMToolbox will be unable to process the data.

4 (Appendix) Structured illumination microscopy using a liquid crystal on silicon (LCOS) microdisplay

What follows is a description of the structured illumination microscope setup that we developed, and of functions within SIMToolbox that are useful for similar setups. We provide these details and software elements in the hope that researchers building their own systems will find them useful.

4.1 Microscope setup

Our setup (see Figure 12) is based on an IX71 microscope equipped with UPLSAPO 100× /1.40 NA and 60× /1.35 NA oil immersion objectives (Olympus). We used a NEO sCMOS camera (Andor; 2560 × 2160 pixels; pixel size 6.5 μm). Focus was adjusted using a piezo-Z stage (NanoScan-Z100, Prior Scientific; resolution 1.5 nm). The desired illumination patterns were produced by a high speed ferroelectric liquid crystal on silicon (LCOS) microdisplay (SXGA-3DM, Forth Dimension Displays; 1280 × 1024 pixels, 13.62 μm pixel pitch). Similar LCOS microdisplays have been used previously in SIM [6–8,23,37,38], and in other fast optical sectioning systems such as programmable array microscopy (PAM) [16,17,39] and light sheet microscopy [40]. The display was illuminated by a home-built, three channel LED system based on high power LEDs (PT-54, Luminous Devices) with emission maxima at 460 nm, 525 nm, and 623 nm. The output of each LED was filtered with a band pass filter (Chroma; 450 − 490 nm, 505 − 555 nm, 633 − 653 nm, resp.), and the three wavelengths were combined with appropriate dichroic mirrors. The light was then vertically polarized with a linear polarizer (Edmund Optics). We imaged the microdisplay into the microscope using a polarizing beam splitter cube (Newport) and 180 mm focal length tube lens (U-TLU, Olympus). Sample fluorescence was isolated with a dual band filter set appropriate for Cy3 and Cy5, or a single band set for GFP (TIRF-grade, Chroma). The image sequences were acquired using Andor IQ software, which controlled an input/output computer card (PCIM-DDA06/16, Measurement Computing) to move the piezo-Z stage and to control LED illumination.

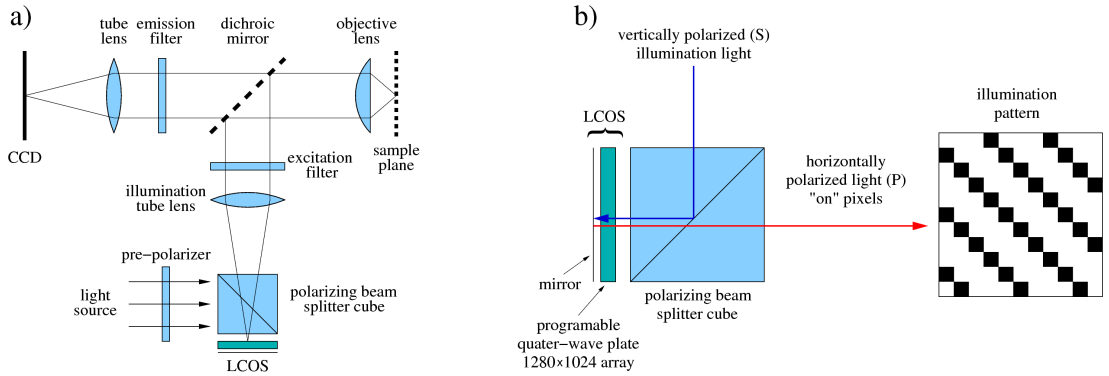


Figure 12: Structured illumination microscope: (a) the microscope setup, (b) principle of illumination pattern generation using a LCOS microdisplay.

When using a 100× objective, single microdisplay pixels are imaged into the sam-

ple with a nominal size of 136.2 nm, thus as diffraction-limited spots. At the shortest wavelength used in our system, 460 nm, the Abbe limit for a 1.40 NA objective is $\lambda/2\mathrm{NA} = 164.3$ nm, which is larger than the 136.2 nm microdisplay pixel size we used. Nyquist-limited sampling of the specimen by the SIM pattern would imply an optimal microdisplay pixel size of $< \lambda/4\mathrm{NA} = 82.1$ nm as imaged in the sample. However, we did not observe obvious patterned artifacts in the reconstructed images and so judged that a pixel size of 136.2 nm was adequate. This is a matter for further investigation. These relationships can easily be changed by choosing a different illumination tube lens and objective. A separate matter is the CCD pixel size. With a 100× objective the camera pixel size is 65 nm in the sample (when using an Andor NEO sCMOS camera), implying Nyquist-limited imaging at the fluorescence wavelength (~ 550 nm).

Figure 13 shows how to connect the LCOS-based microscope system. We used Andor cameras and software, but a similar system can be built using components from other vendors. We chose Andor IQ software because it provides, via a DDA06/16 computer card, generic user-configurable inputs and outputs that can be synchronized with the camera exposures. Andor IQ also controls other microscope hardware and provides for image analysis. This saves considerable time in unnecessary programming when building such a system. A 74LS123N monostable vibrator uses the rising edge of the camera FIRE signal to produce a pulse which is connected to the LCOS synchronization input and causes the microdisplay to begin showing a pattern. The falling edge of the FIRE signal is used by a second 74LS123N to instruct the microdisplay to advance to the next pattern. We found that this was necessary to ensure that, in long sequences, the microdisplay reliably changed to the next pattern after each camera exposure. Selecting a channel (i.e., red, green, or blue excitation) in IQ is linked to an enable signal produced by the DDA06/16 card. This is combined by a logical AND gate with the enable signal from the microdisplay controller. This signal is further combined with the camera FIRE signal using a second AND gate. In this way, the LED is only activated during camera exposures, and only when the proper pattern is present on the microdisplay. This prevents unwanted illumination, thereby reducing photobleaching.

Figure 14 shows a schematic of the optical setup. Several details are worth mentioning. The size of the Luminous Devices PT54 LED emitter is 2.0 x 2.7 mm. The magnification ratio from the emitter to the objective back focal plane is given by the ratio of the focal lengths of the illumination tube lens (180mm) and the LED collector lens (50 mm), $180/50 = 3.6$. Thus the LED emitter will be imaged into the objective back focal plane as a source of about 7.2 x 9.7 mm. This roughly fills the back aperture of an Olympus 60× or 100× objective, which are approximately 8 mm in diameter. Shorter focal length collector lenses will collect more light from the LEDs but will not increase the throughput of the system. We used an Olympus U-TLU tube lens for imaging the microdisplay into the microscope. This ensures that optical aberrations are kept to a minimum.

Our illumination setup follows a Köhler-type design in which images of the LEDs are formed in the back focal plane of the objective. The illumination is therefore uniform on the microdisplay and in the sample. The Luminous Devices LED emitters are structured, and so imaging the LEDs onto the microdisplay is not recommended. A mounting frame is available for the microdisplay. In our setup, this frame acts as a field aperture, and delineates the illuminated area. We also found that use of flat dichroic mirrors was important

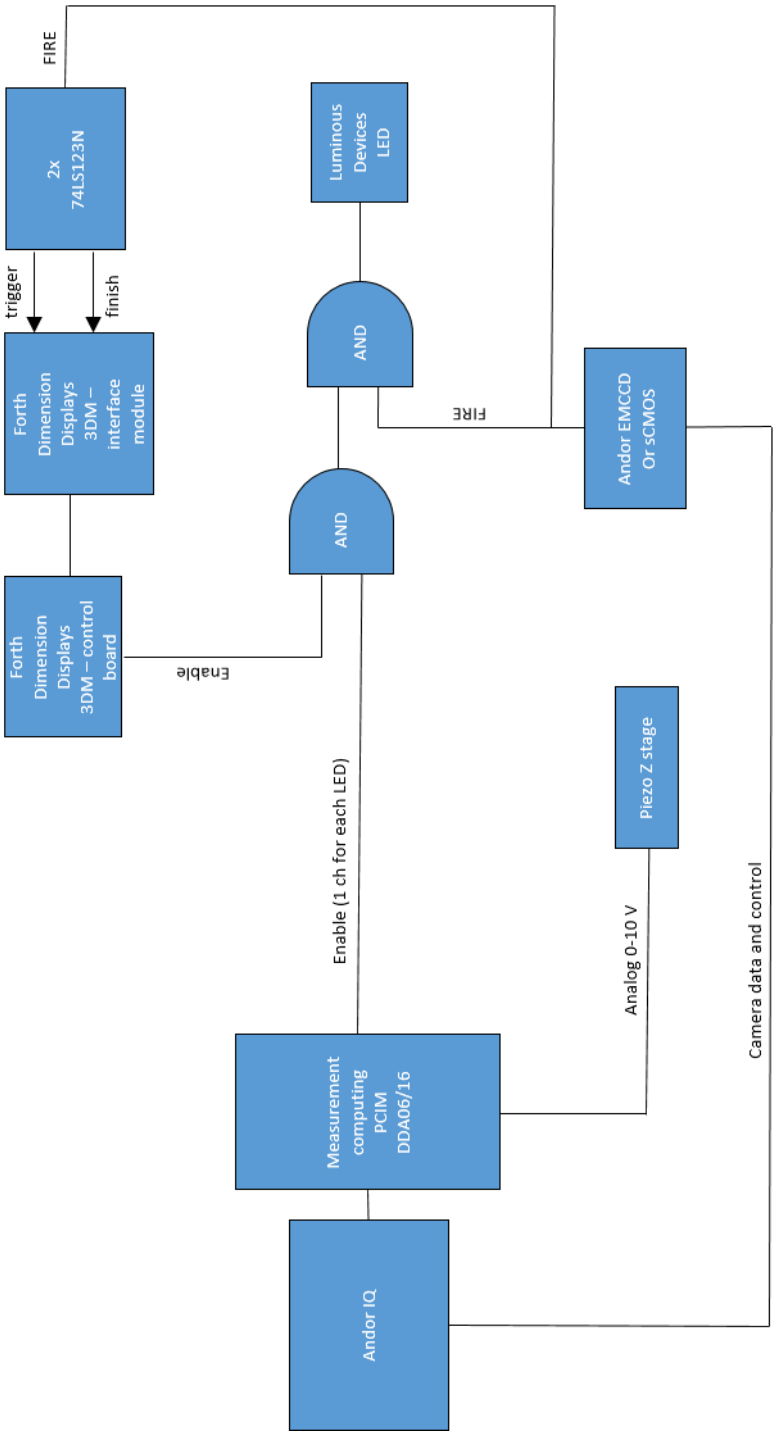


Figure 13: Connection diagram. See text for details.

for good imaging quality. Wavefront distortions introduced by typical dichroics are easily visible in the focused illumination pattern. "TIRF-grade" dichroics with 2 mm thick substrates are available from Chroma, as well as specially designed filter cubes. These metal cubes ensure that the dichroics are mounted without strain (twisting). We used the cubes without excitation filters, because the excitation filters need to be mounted in front of the LEDs.

In our setup, the LCOS and LED system are placed behind the IX71 microscope. This is preferred because it allows one to use the filter sets in the turret, thereby allowing fast dichroic changes. It also allows one to visualize the illumination pattern by eye. This helps greatly when aligning the system and makes identifying aberrations (e.g. vignetting or distortion from a non-flat dichroic) much easier. It also allows one to use conventionally oriented dichroics, and to mount the camera in the usual place.

Figure 15 shows one possible timing scenario for the LCOS system. The different timing schedules are supplied by Forth Dimension Displays and are not user-adjustable. The timing schedule shown below is called "300 µs 1 bit balanced." In this particular schedule, 300 µs lit periods are followed by 845 µs dark periods (duty cycle 26%). The "1 bit" designation indicates that the display will show each pattern for the same length of time. 8 bit schedules are also possible, which can be used to make gray scale images. However this is generally not desirable for SIM applications. During the dark periods, the microdisplay shows the inverse of the current pattern. This is required to avoid charge build up within the LCOS device. During these "compensation" periods, the light source must be switched off. This is easily accomplished with LED light sources. In this example, LCOS display cycles are synchronized with 3 ms camera exposures. Longer exposures will result in more lit periods being integrated by the camera. We typically used exposures from 50 to 500 ms depending on the sample. In later experiments, we switched to a schedule that uses 2 ms lit periods and also has a higher duty cycle for illumination.

4.2 Illumination patterns

Most strategies in structured illumination microscopy assume that a set of illumination patterns required for image reconstruction consist of N equal movements of the same pattern such that the sum of all of the patterns results in homogeneous illumination. The illumination patterns are usually created from a grid of lines [3,7], or from dots in a square or triangular grid [13,41,42]. The mark-to-area ratio (MAR) [1] of the pattern is defined as the fraction of pixels which are considered to be illuminated in a unit area of the pattern.

For the toolbox, we generated several illumination sequences with line and dot grid patterns. The package of these illumination sequences can be also downloaded from the project web site http://mmtg.fel.cvut.cz/SIMToolbox. **The illumination patterns are designed for a high speed ferroelectric liquid crystal on silicon (LCOS) microdisplay** (SXGA-3DM, Forth Dimension Displays, Dalgety Bay, Scotland; 1280 × 1024 pixels, 13.62 μm pixel pitch). Each pattern sequence is stored in the so called *Repertoire* (`*.repz`), which is a zip file containing images of patterns (`*.bmp`; 1-bit images in our case), a compiled timing schedule (`*.seq`; for example 300 μs 1-bit balanced sequence in our case) with timing of the display, and a text file (`*.rep`) describing the so called running orders. A *Running order* is a short program defining the order of displayed images

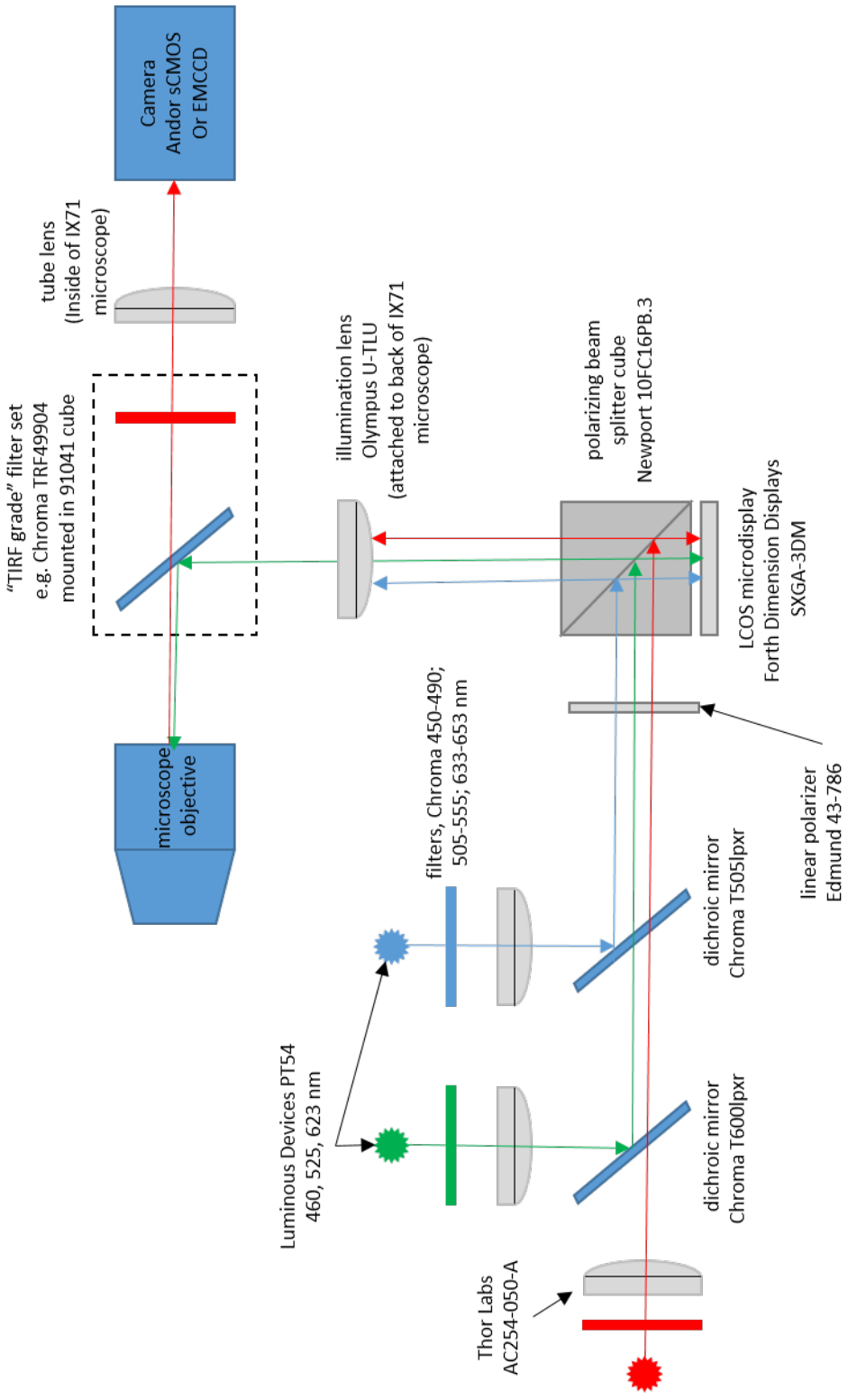


Figure 14: Optical setup. See text for details.

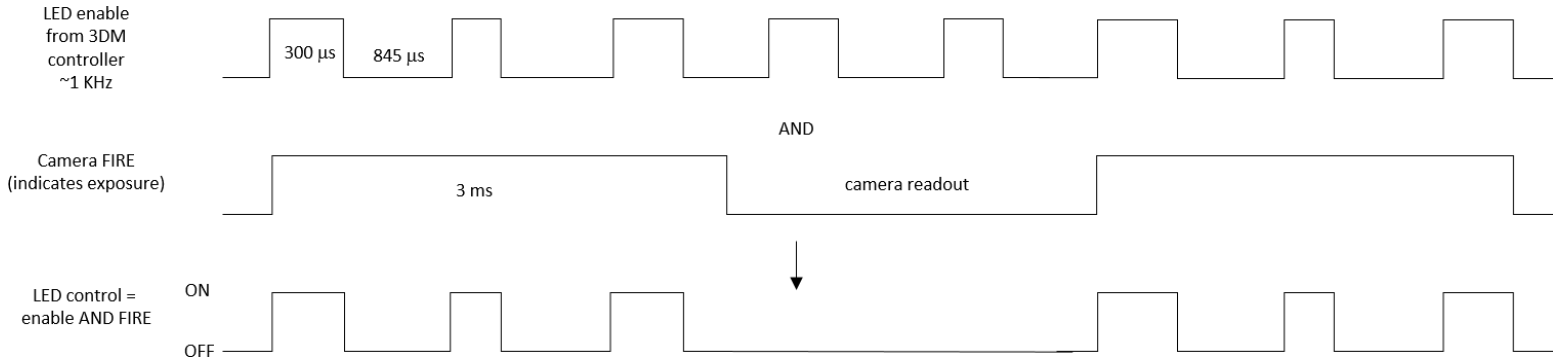


Figure 15: Timing diagram showing a schematic of one possible timing schedule.

and reaction of the microdisplay's driver board to input synchronizing signals, e.g., to show next image when a trigger is detected. There can be several independent running orders stored in one repertoire. For our purpose, we added a pattern description text file (*`*.yaml`*) with various information about the stored illumination patterns. All repertoires with illumination patterns are available in the directory *`patterns`*.

4.2.1 Line-grid patterns

Line-grid illumination patterns contain single and/or multiple orientations of lines. Repertoires are stored in files *`linesANGLEo-ON-OFF-STEP-NUM.repz`*, where *`ANGLE`* stands for orientation of the line-grid, *`ON`* defines thickness of each line in pixels, *`OFF`* is the gap in between the lines in pixels, *`STEP`* is the shift of the pattern in pixels between each image acquisition to obtain a phase-shifted illumination mask, and *`NUM`* is the number of patterns in one scanning sequence. Repertoires with single a orientation are created for angles 0°, 90°, 45°, and 135° and with *`ON`* = 1–6 pixels, *`OFF`* = 1–40 pixels, and *`STEP`* = 1–3 pixels. Repertoires with multiple orientations are designed such that the period and MAR of the pattern are as close as possible in different orientations (keeping it the same is not possible due to pixelated microdisplay image). These repertoires are created for orientations {0°, 90°}, {45°, 135°}, {0°, 63.4°, 116.6°}, {26.6°, 90°, 153.4°}, and {0°, 90°, 45°, 135°} and with *`ON`* = 1–3 pixels, *`OFF`* = 1–15 pixels, and *`STEP`* = 1–3 pixels. Examples of line-grid illumination patterns are shown in Figure 16. Note that some combinations of *`ON/OFF/STEP/ANGLE`* are not possible due to the necessary condition of the pattern's homogenity (the sum of all of the patterns should be uniform illumination).

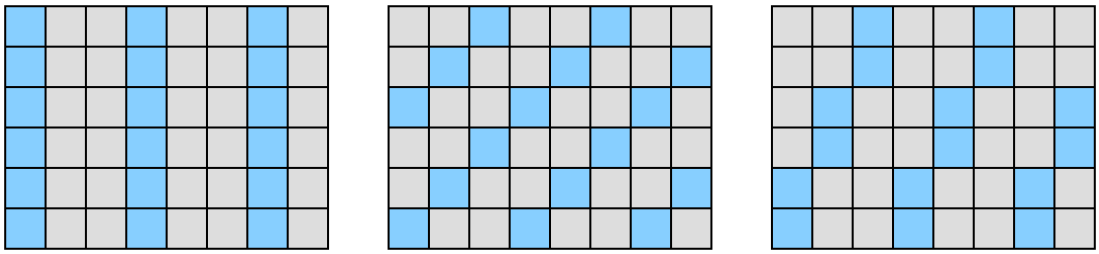


Figure 16: Example of line-grid illumination patterns. Left: line-grid pattern at 90° with *`ON`* = 1, *`OFF`* = 2, *`STEP`* = 1, *`NUM`* = 3, period of lines 3 pixels, and MAR = 1/3. Middle: line-grid pattern at the angle of 45° with *`ON`* = 1, *`OFF`* = 3, *`STEP`* = 1, *`NUM`* = 3, period of lines 2.12 pixels and MAR = 1/3. Right: line-grid pattern at the angle of 63.4° with *`ON`* = 1, *`OFF`* = 3, *`STEP`* = 1, *`NUM`* = 3, period of lines 2.66 pixels and MAR = 1/3.

4.2.2 Dot-grid patterns

Repertoires with dot-grid illumination patterns were created in a square or triangle grid. Patterns with a square grid are stored in files *`dotsSQ-ON-OFF-STEP-NUM.repz`* and patterns with a triangle grid are stored in files *`dotsTRI-ON-OFF-STEP-NUM.repz`*. Dots are of size *`ON`* ×*`ON`* pixels with a gap of several *`OFF`* pixels in between the dots, *`STEP`* is the shift of the pattern in pixels between images, and *`NUM`* is the number of patterns in one

scanning sequence. We generated patterns with *ON* = 1–3 pixels, *OFF* = 1–13 pixels, and *STEP* = 1–3 pixels. Examples of dot-grid illumination patterns are shown in Figure 17. Note that some combinations of *ON/OFF/STEP* are not possible due to the necessary condition of the pattern's homogeneity (the sum of all of the patterns should be uniform illumination).

We note here that not all of the possible processing methods are appropriate for all of the illumination masks. For example, when using dot-grid patterns, the homodyne detection method (Eq. 5) will not work.

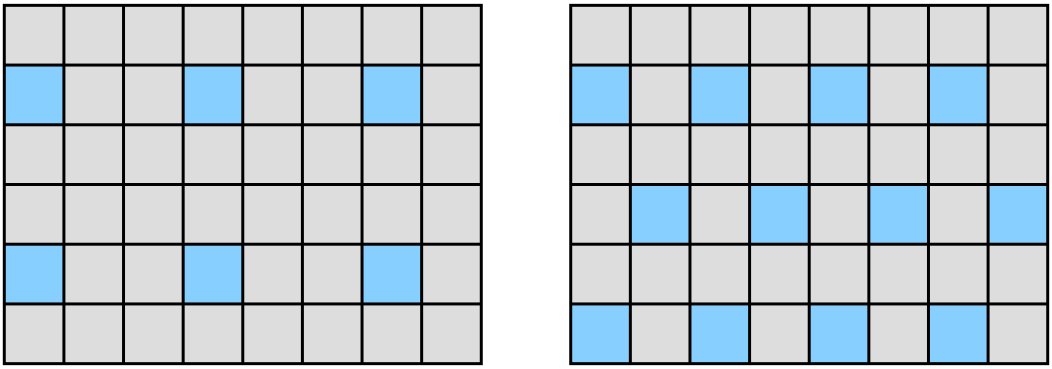


Figure 17: Examples of dot-grid illumination patterns. Left: square grid with *ON* = 1, *OFF* = 2, *STEP* = 1, *NUM* = 9, horizontal and vertical period of 3 pixels, and MAR = 1/9. Right: triangle grid with *ON* = 1, *OFF* = 1, *STEP* = 1, *NUM* = 4, horizontal and vertical period of 2 pixels, and MAR = 1/4.

4.2.3 Calibration patterns

Because the microdisplay lets us to create an arbitrarily configured "known scene" , we established the calibration using a chessboard pattern [7]. Four orientation markers are placed into the chessboard pattern in order to define the microdisplay coordinate system. The sequence of illumination patterns also contains a white (all microdisplay pixels ON) and a black (all microdisplay pixels OFF) image to compensate for possible illumination artifacts (flat-field correction), see Figure 18, and a negative chessboard pattern which alternates with the positive one during the image sequence acquisition to keep photo-bleaching of the sample uniform. The recommended calibration sample is a thin fluorescent film. A thick fluorescent plastic slide is not recommended. Repertoires with the calibration sequence are stored in files *calibr_boxSIZE_seq3.repz*, where *SIZE* defines the chessboard box size in pixels and were generated for *SIZE* = 8–32 pixels. In the actual microscope setup, we typically use a box size of 14 pixels. The calibration procedure and image processing involved is described in more detail in Section 5.

4.2.4 Creating a pattern repertoire

Example of a *repz* file:

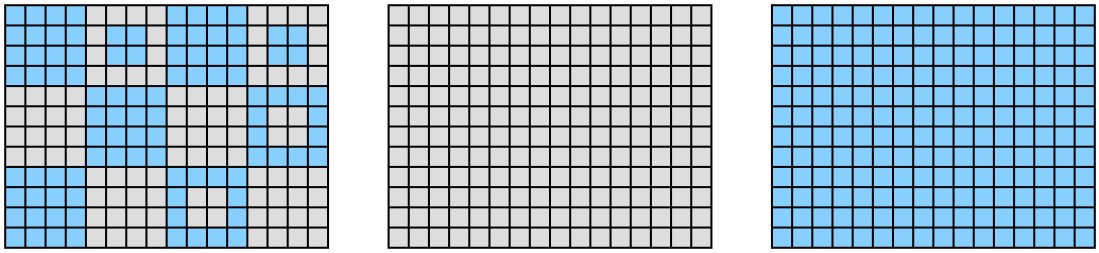


Figure 18: Example of illumination sequence for camera-microdisplay calibration. Left: chessboard pattern of $SIZE = 4$ pixels with four orientation markers. Middle: pattern with all pixels off to obtain a background image. Right: pattern with all pixels on to obtain a flat-field image.

```
lines90o-1-01-1-02.repz
  |
  +-- 48449 300us 1-bit Balanced.seq3  # timing of the microdisplay
  +-- lines90o-1-01-1-02.rep           # repertoire description
  +-- lines90o-1-01-1-02.yaml          # repertoire description
  +-- ptrn001_lines90o-1-01-1-02.bmp   # 1-bit BMP file (binary mask, 1st phase)
  +-- ptrn002_lines90o-1-01-1-02.bmp   # 1-bit BMP file (binary mask, 2nd phase)
  +-- white.bmp                        # 1-bit BMP file (binary mask, all ones)
```

Note that *rep* and *yaml* files must have the same name as the *repz* file, while the other files can have any name. Their purpose is further specified inside the *repz* file, which defines the running orders used by the microdisplay.

Listing of *lines90o-1-01-1-02.rep*:

```
SEQUENCES        #please consult the 3DM product documentation for more information
A "48449 300us 1-bit Balanced.seq3"
SEQUENCES_END

IMAGES
1 "white.bmp"
1 "ptrn001_lines90o-1-01-1-02.bmp"
1 "ptrn002_lines90o-1-01-1-02.bmp"
IMAGES_END

DEFAULT "WHITE"
[
 <(A,0) >
]

"lines90o-1-01-1-02_TF"        #TF means the 3DM unit listens for both
[                                                  #TRIGGER and FINISH signals
 <t(A,1) >
 {f (A,1) }
 <t(A,2) >
 {f (A,2) }
]
```

```
"lines90o-1-01-1-02_F"        #F means the 3DM unit listens only for FINISH signals
[
 {f (A,1) }
 {f (A,2) }
]
```

The file specifies where the information about microdisplay timing is stored and what images are available. Then three different running orders are defined. Each frame of a running order has assigned a timing information, the images declared above and information about when the image is displayed with respect to the trigger signal (*trigger* or *finnish*).

The *yaml* file is required for analysis only and its format is specified in Section 6.2.

5 (Appendix) Calibration of illumination patterns - for use with LCOS or DMD SIM systems

The position of the illumination pattern in the camera images might be determined by analyzing the raw images in some way. In our hands this proved both difficult and inaccurate, particularly with sparse samples. Previously, we introduced a procedure that allows one to determine a mathematical model describing the one-to-one mapping between the microdisplay and the camera sensor and thus to create a digital illumination mask in the camera image [7]. Having such a model lets us determine the exact position of an arbitrary illumination pattern in the camera image, even for sparse samples. In the following sections we briefly summarize the principles of the method and describe functions available in the toolbox.

We performed the calibration before each experiment. One calibration is needed for each emission wavelength.

5.1 One-to-one correspondence mapping between microdisplay and camera

The illumination patterns created on the microdisplay are projected to the camera chip as follows. An optical ray originating at a point $\boldsymbol{x} = [x, y]^\top$ in the plane of the microdisplay illuminates the sample by passing through an illumination tube lens and the microscope objective. Fluorescence from the sample is collected by the objective and imaged at the point $\boldsymbol{u} = [u, v]^\top$ on the camera sensor. A diagram is shown in Figure 19.

Let us express the position of points $\boldsymbol{x}, \boldsymbol{u}$ in projective coordinates: $\tilde{\boldsymbol{x}} = [x, y, 1]^\top$, $\tilde{\boldsymbol{u}} = [u, v, 1]^\top$ and let us assume that there are no distortions in the path of the optical ray. Using projective coordinates allows us to describe affine transformations (e.g., translation, rotation, scaling, reflection) by a single matrix multiplication. It can be shown [43] that any point (or pixel) from the microdisplay plane can be unambiguously mapped to the camera sensor (and vise versa) using a linear projective transformation

$$\alpha \tilde{\boldsymbol{u}} = H \tilde{\boldsymbol{x}}\,, \tag{29}$$

where H is a constant 3×3 projective matrix, and $\alpha \neq 0$ is an appropriate scaling factor. The projective matrix H is unique for a given setup and has to be determined, see Section 5.2. The linear mapping model in Equation (29) can be further extended to correct for radial distortion and decentering components [7].

5.2 Camera-microdisplay calibration

Camera-microdisplay calibration is a procedure which determines numerical values of the projective matrix H in Equation (29). Calibration proceeds by finding corresponding pairs of points between the microdisplay and the camera image. Each such correspondence provides one Equation (29). This results in a system of equations which can be solved by least squares methods [7, 43]. Typically hundreds of points are used to correct for uncertainty in the measurements.

As the microdisplay lets us create an arbitrarily configured “known scene” , we established the calibration using the chessboard illumination pattern, see Section 4.2.3. The

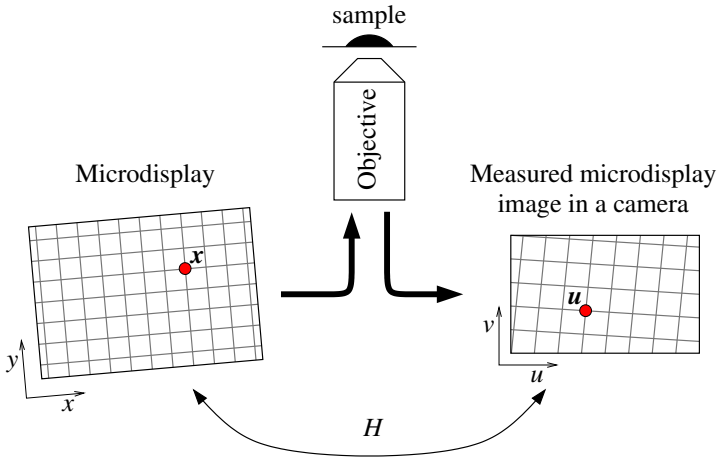


Figure 19: One-to-one mapping between the microdisplay and the camera sensor. A point $\boldsymbol{x}$ on the microdisplay is imaged at point $\boldsymbol{u}$ on the camera. The rotations are exaggerated to help illustrate that they can be compensated for by the algorithm.

known positions of corners on the microdisplay are used as reference points and four markers in the chessboard center define the orientation. We project the chessboard illumination pattern to the camera sensor using a sample made of a thin fluorescent film. The microdisplay and camera images are shown in Figure 20. Calibration proceeds by finding the positions of the corners in the camera image and by finding corresponding pairs of points between the microdisplay and the camera. We usually acquire a Z-stack of images 2 μm thick with a step of 50 nm to compensate for possible sample tilt. This ensures all of the corners are sharply in focus somewhere in the image stack.

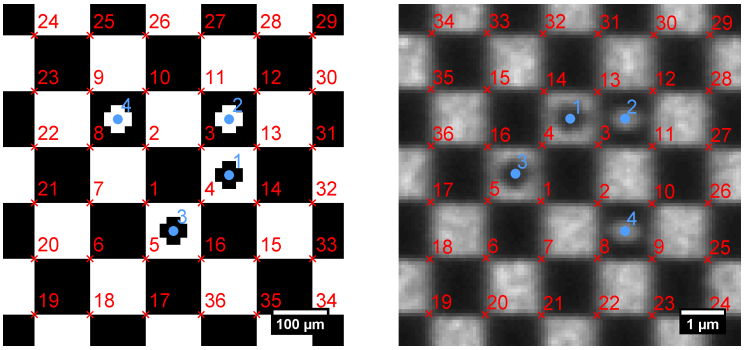


Figure 20: Chessboard calibration pattern. Four markers define the orientation of the coordinate system. Left: microdisplay image with known positions of corners and markers. Right: raw camera image of the same area with detected corners and markers (100 × /1.45 NA objective; our SIM setup causes rotation and flip of the microdisplay image). Point correspondences are indicated by numbers. Numbering starts from the center. Adapted from [7].

If a list of point correspondences between the microdisplay and camera images is

available, the following toolbox functions can be used to compute the linear mapping model in Equation (29).

```
map = calhomolin(wpts, ipts);
```

Here `ipts` contains u, v coordinates of corners localized in the camera image, `wpts` are the corresponding x, y coordinates of the reference points on the microdisplay, and `map` contains information about the mapping model. Using the mapping model, any point can be transformed between the microdisplay and camera images by:

```
ipts = calw2i(wpts, map);
wpts = cali2w(ipts, map);
```

Automatic processing of a stack of images (including markers and corners detection, correspondence matching, and computation of the mapping model) acquired using the calibration illumination sequence (Section 4.2.3) is provided by the function:

```
calprocess(datadir);
```

Here `datadir` is a string with the path to the data and the data processing can be configured in the file *`calconfig.m`*. The parameters of the mapping model are saved into the same directory as the calibration data as *`filemask_LIN.yaml`*. Note that the calibration repertoire *`*.repz`* used for data acquisition has to be present in the same directory as the data.

5.3 Detection of markers and their alignment

To determine the coordinate system of the microdisplay in the camera image, four orientation markers are placed into the chessboard center. The detection of the markers is based on the assumption that the average intensity of the squares with markers is in between the intensity of white and black squares. A binary mask with squares marked in black and the rest in white is obtained by $|I - t| > t$, where I is a slightly blurred camera image, and t is an average intensity of the image I. The position of markers is determined as local intensity maxima in a distance transform [44] which was applied to the the binary mask. The alignment of markers and orientation of the coordinate system is based on sorting Euclidean distances between points A, B, C, D such that $l_{AB} < l_{AC} < l_{AD}$, see Figure 21.

The toolbox provides the following functions to detect and sort the positions of the markers, and also a simple user interface to select/check position of the markers manually:

```
markers = markers_detect(im, sigma);
markers = markers_check(im, markers);
markers = markers_sort(markers);
```

Here `im` is the input camera image, `sigma` is the blurring factor of an applied Gaussian filter, and `markers` contain x, y coordinates of the detected markers.

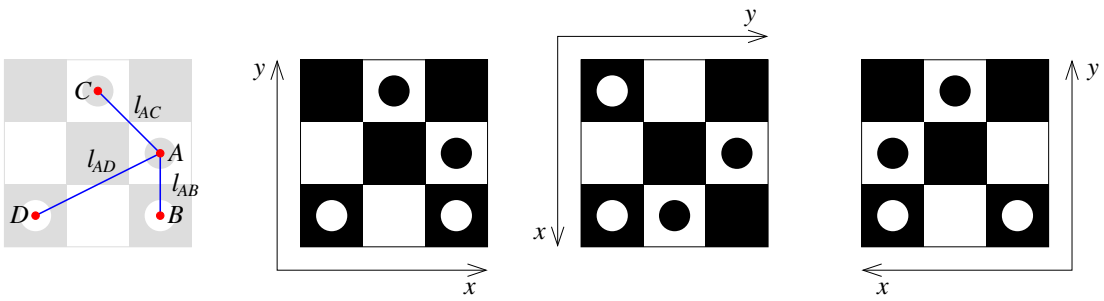


Figure 21: The four markers are recognized according to the Euclidean distances between points A, B, C, D such that $l_{AB} < l_{AC} < l_{AD}$. The orientation of the coordinate system is based on the positions of the markers.

5.4 Detection of corners

Our corner detector is based on convolution of the input image with two 3×3 kernels $K, -K$ to calculate response to the situation where a black square is on the top left and to the situation where a white square is on the top left. The resulting response of the detector is computed as $G = (K * I)^2 + (-K * I)^2$, where

$$K = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}.$$

The positions of the corners are determined as local intensity maxima above a user-specified threshold in the response G. Note that the input image I has to be slightly blurred before applying the convolution operation in order to reduce the influence of noise.

The function performing the corner detection is:

```
[corners, G] = corners_detect(im, sigma, thr);
```

Here `im` is the input camera image, `sigma` is the blurring factor of the Gaussian filter, `thr` is a threshold value to the response `G` of the corner detector, and `corners` contain x, y coordinates of the detected corners.

Sub-pixel position of the corners is estimated by fitting the values of G by a quadratic bivariate function in a local neighborhood of corners. The toolbox also provides a strategy based on determining the intersection of local intensity gradients [45].

The toolbox functions are:

```
[ysub, xsub] = subpix2d_fitquadric(y, x, G, nbr);
[ysub, xsub] = subpix2d_gradintersect(y, x, G, nbr);
```

Here `x` and `y` are the coordinates of the detected corners, `xsub` and `ysub` are their sub-pixel estimates, and `nbr` is the size of the local neighborhood.

5.5 Flat-field and background images

Flat-field and background images are used to suppress image artifacts caused by uneven illumination, see Section 3.1.1. The automatic calibration procedure run by function *calprocess* computes the flat-field image as an average of all images obtained with full illumination and saves the result to the file *filemask_FFwhite.tif*. The background image is computed as an average of all images obtained with all microdisplay pixels OFF and the result is saved to the file *filemask_FFblack.tif*. We ususally collected these images automatically as part of the calibration acquisition.

5.6 Correspondence matching

The automatic calibration procedure run by function *calprocess* determines the point correspondences between the microdisplay and the camera image as follows. First, the linear mapping model in Equation (29) is initialized using the known position of the markers on the microdisplay and the detected markers in the camera image. Then the algorithm proceeds in concentric squares with increasing edge size and with the origin in the center of the markers. The known positions of the corners in the microdisplay are mapped to the camera coordinate system and the correspondences are detected as nearest neighbors. The mapping model is updated on the fly with newly matched correspondences.

5.6.1 Determining pattern position in the camera image

The mapping model in Equation (29) allows us to determine the exact illumination pattern position in the camera image. Assuming a known camera-microdisplay calibration and a known illumination pattern on the microdisplay, the digital illumination mask in the camera image can be recreated by the following steps:

```
ptrn2camera(imginfo, ptrninfo, calinfo);    % create a look-up table
IMptrn = ptrnload(ptrninfo, 'runningorder', numro, 'number', no);
IMmask = ptrn2camera(IMptrn);
```

or the following command can be used to recreate the whole illumination sequence:

```
Mask = ptrnmaskprecompute(imginfo, ptrninfo, calinfo, ...
        'runningorder', numro, 'sigma', sigma);
```

Here `imginfo`, `ptrninfo`, and `calinfo` are structures with information about the image stack, illumination pattern repertoire, and camera-microdisplay calibration, respectively, `IMptrn` is a microdisplay illumination pattern in a specified running order `numro` and a pattern position `no`, `IMmask` is the recreated digital illumination mask in the camera image, `Mask` is a stack of digital illumination masks as projected to the image stack, and `sigma` is a blurring factor which approximates the measured point spread function (PSF) of the microscope by smoothing the illumination mask with a Gaussian filter.

5.6.2 Determining pattern position in the frequency domain

For the purpose of SR reconstruction according to the Gustafsson-Heintzmann method (see Section 3.3.1), we need to find precise positions of peaks in spectra of the modulation pattern. This is available through function *sim_findpeaks*. There are two methods the function can use for computation. The traditional approach is to find local intensity maxima of the power spectrum near the theoretical modulation frequency. A new, alternative approach introduced here for the first time utilizes the camera-microdisplay calibration which allows one to determine the position of the peak with high precision. The approach to be used is selected by setting the parameter *method.type* of *sim_findpeaks* to either *'spotfinder'* for the traditional approach, or *'calibration'* for the alternative approach. Finding local intensity maxima This method is applied on the Fourier transform of raw images. The algorithm first filters the image with a lowpass or bandpass filter to reduce noise, then finds local maxima in the filtered image and finally refines the found position to obtain positions with sub-pixel precision. The function *spotfinder(imraw, filter, detector, estimator)* has been developed for this purpose. All possible types of filters, detectors and estimators can be found in sub-directory *./utils/spotfinder*. The local maxima approach has a significant disadvantage since the spots are difficult to find precisely due to significant intensity variations in power spectra, especially in data with low SNR. Alternative approach with calibrated patterns When we utilize the camera-microdisplay calibration, we determine the exact positions of the peaks, even with low SNR data. This is possible since the approach calculates the peaks directly from information about the illumination patterns and is thus independent of the quality of the acquired images. The algorithm works with a theoretical 2-dimensional sine wave parametrized by orientation angle and period of the illumination pattern that would ideally be displayed on the microdisplay. This sine wave is virtually transformed from the microdisplay into the camera using a linear transformation obtained from the calibration process. Finally, from the exact knowledge of the parameters of the sine wave in the camera, the positions of the peaks in the Fourier power spectrum can be directly calculated.

There are several advantages to this method. In our hands, the method is robust, and is able to correctly locate the Fourier peaks in a variety of data we have worked with. Any residual errors are very small [7]. The method is also able to correct for radially symmetric distortions (barrel or pincushion distortions). Once the calibration data is available, the actual calculations are fast. Finally, the method is useful for both digital micromirror device (DMD) and LCOS-based microscope systems. DMD and LCOS devices have become more popular in microscopy as these technologies have matured.

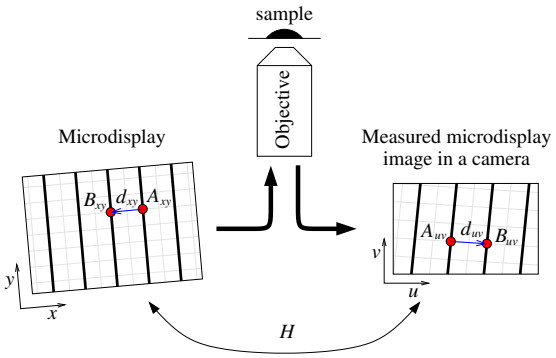


Figure 22: Determining pattern period or frequency in a camera image and the position of spectral components in the frequency domain. Rotation and scaling are exaggerated in the diagram.

6 (Appendix) YAML File structure, other software notes

YAML files are Python-based text files used for (de)serialization of objects, therefore the main constructs in these files are objects and arrays of objects. A simple object is defined as *`name: value`*. An object can contain sub-objects, which can be defined either as a set of indented name-value pairs, each on a separate line, or on a single line inside curly brackets, e.g., *`{x:5,y:10}`*. An array is defined in square brackets, e.g., *`[1,2,3]`*, or even *`[{x:3,y:5},{x:4},{y:1}]`*.

6.1 Data info

Description of a dataset can be stored in *`dataset.info`* file. Despite the file extension, this is a *`yaml`* file.

```
id: SIM_data
data:
  order: zs
image:
  size:
    x: 400
    y: 400
    z: 25          # number of Z planes
    seq: 15    # number of images in the sequence, e.g. 5 phases and 3 angles
    w: 1       # number of channels
    t: 1       # number of time points
  resolution:
    x: 0.065   # [um]
    y: 0.065   # [um]
    z: 0.126   # [um]
camera:
  name: PCO edge sCMOS camera (used in Zeiss Elyra system)
  size: {x: 1280.0, y: 1280.0}
  pixel: {pitch: 6.5, gap: 0.0}
  bitdepth: 16
ptrn:         # optional if no repz
```

```
  order: ap   # order of sequence
  angles: 5   # number of angles
  phases: 5   # number of phases
```

Most of the values are self-explanatory and don't need a detailed description. *id* is a unique identificator for *info* file format and is always *SIM_data*. Field *data.order* describes the order of frames in the dataset. Possible values are strings of *z,s,w,t*, which are symbols corresponding with the dimensions described in field *image.size*. Similarly, order of patterns in *repz* file is described in *ptrn.order* and its value is a string of *a*ngle and/or *p*hase.

The *info* file is automatically fetched if it is available when calling *imginfoinit*. The function returns a structure containing all the available information, which is then utilized when reading the data from a *tiff* stack using function *imgload*. This adds a semantic information from *info* file to the pure stack of images, which makes further manipulations with the data much simpler and more transparent.

6.2 Pattern

As discussed in Section 4.2.4 a *repz* file contains a *yaml* file with all information about the repertoire as shown below:

```
name: lines90o-1-01-1-02
imagesize: {x: 1280.0, y: 1024.0}
sequence: 48449 300us 1-bit Balanced.seq3
default: 0.0
runningorder:
- name: WHITE
  trigger: none
  numseq: 1.0
  data:
  - id: white
    num: 1.0
    images: [white.bmp]
- name: lines90o-1-01-1-02_TF
  trigger: TF
  numseq: 2.0
  data:
  - id: lines
    angle: 90.0
    'on': 1.0
    'off': 1.0
    step: 1.0
    num: 2.0
    MAR: 0.5
    period: 2.0
    images: [ptrn001_lines90o-1-01-1-02.bmp, ptrn002_lines90o-1-01-1-02.bmp]
- name: lines90o-1-01-1-02_F
  trigger: F
  numseq: 2.0
  data:
  - id: lines
    angle: 90.0
```

```
    'on': 1.0
    'off': 1.0
    step: 1.0
    num: 2.0
    MAR: 0.5
    period: 2.0
    images: [ptrn001_lines90o-1-01-1-02.bmp, ptrn002_lines90o-1-01-1-02.bmp]
```

The file tells us dimensions and timing of the microdisplay and lists individual running orders, including meta-information such as triggering, period, or mark-to-area ratio (MAR).

Pattern description file is automatically fetched when calling *ptrnopen*. The function unzips the *repz* file and returns a structure filled with the information obtained from the *yaml* file. The individual patterns or sequences of patterns can be read by calling *ptrnload*, which takes the info structure as an argument. To close a pattern repertoire, call *ptrnclose*.

6.3 Calibration

An example of file *calibration_RAD.yaml* as a result from the calibration process:

```
id: SIM_calibration
time: 05-Jun-2014 18:19:52
numchannels: 1.0
setup:
  display:
    name: SXGA-3DM (4DD)
    siz: {x: 1280.0, y: 1024.0}
    pixel: {pitch: 13.62 um, inter-pixel gap: 0.54 um}
    origin: {x: 644.5, y: 516.5}
  camera:
    name: iXon EMCCD (Andor)
    siz: {x: 1004.0, y: 1002.0}
    pixel: {pitch: 8.0, gap: 0.0}
    roi:
      xlim: [1.0, 1004.0]
      ylim: [1.0, 1002.0]
  tubelens: 180 mm (Olympus IX 71)
  objective: 100x / 1.40 NA (Olympus)
  immersion: oil RI 1.515 (Cargille DF)
  ptrn: {repz: calibr_box08_old_seq3.repz, ro: 1.0}
  cfg: {imFFwhite: calibration_FFwhite.tif, imFFblack: calibration_FFblack.tif,
        detmarksigmablure: 2.0, detmarkborder: 100.0, detcornsigmablure: 2.0,
        detcornthr: 0.3, detsubpixnbr: 1.0, detcorndst: 0.8}
cal:
  channel: 1.0
  name: 460 nm
  map:
    Hw2i:
    - [-0.0018176612385491003, 2.043380682759331, -517.5273964812662]
    - [2.0518560527566594, 0.0011187159485942615, -827.639931821344]
    - [9.224820960695596E-7, 7.501092268414877E-7, 1.0]
    Hi2w:
```

```
    - [-3.5641399486015636E-4, 0.48680419136118064, 403.35874688114933]
    - [0.48944593846373424, -2.3114930808478904E-5, 253.72352977852728]
    - [-2.627310449524256E-7, -1.1093862096448294E-6, 1.0]
    Rw2i:
      [546.1711580687274, 649.9019876913792, 2.465510379176542E-9,
       4.9317582792866E-16, 1.159336374781209E-25]
    Ri2w:
     [556.756690082697, 556.935867724223, -2.05698804580254E-9,
      -1.4532493673046376E-15, 6.2780696604229505E-24]
  origin: {x: 536.1801321804793, y: 494.86389075656933}
  err:
    numpts: 3782.0
    disp2cam:
      {av: 0.11736070012623095, std: 0.0787570754003634, max: 1.0907613267720675}
    cam2disp:
      {av: 0.05973459791030152, std: 0.03899923771618263, max: 0.5571146216985254}
```

The unique *id* of the calibration file is *SIM_calibration*. Among other information, details about the microdisplay are stored in *setup.display*, camera is described by *setup.camera*, pattern repertoire is specified in *setup.ptrn* and configuration of the calibration process is converted into *setup.cfg*, where values *setup.cfg.imFFwhite* and *setup.cfg.imFFblack* contain names of the files required for flat field correction.

Object *cal* is the most important one for analysis, as it contains an array of channels (wavelengths), each with its own mapping *cal.map* containing the 3×3 matrices for transformation and also parameters for correction of radial distortion from the microdisplay to the camera and vice versa as detailed in Section 5.

Calibration file is processed by calling function *calinfo*. Then functions *calw2i* and *cali2w* can be used to transform a set of points from the microdisplay to the camera and vice versa:

```
1     calinfo = calload(filename);
2     ipts = calw2i(wpts, calinfo.cal.map, calinfo.setup.camera.roi);
3     wpts = cali2w(ipts, calinfo.cal.map, calinfo.setup.camera.roi);
```

Here `ipts` and `wpts` are $n \times 2$ matrices with n rows of $\{x, y\}$ coordinates.

6.4 Manual initialization of image stacks and image sequences

Image stack `IMseq` is a stack of images acquired for all orientations and shifts (phases) of one complete sequence of illumination patterns. It is stored in a matrix of size $m \times n \times q$, where m and n correspond to image size, and q is the number of illumination patterns in the sequence.

Image sequence `seq` is an "array of structures" with fields `angle`, `IMseq`, and `num`. It can be created only for line-grid illumination patterns and its purpose is to separate images acquired with different orientations of the pattern. Field `angle` specifies orientation of the line-grid pattern, `IMseq` is a sub-stack of images acquired for a particular pattern orientation, and `num` is a number of images in the appropriate sub-stack. The image sequence also allows an easy selection of only some orientations of the pattern. For example, an image

sequence acquired with four line-grid orientations at {0°, 90°, 45°, 135°} can be reduced to angles {45°, 135°} by typing a command `seq([0 0 1 1])`.

Data for SIM processing are initialized by:

```
imginfo = imginfoinit(datadir, filemask);
```

Here `datadir` is a string with a path to the SIM data, `filemask` is a string used to search the directory for images named `filemask*.tif`, and `imginfo` is an output structure with information about the SIM data. The function requires a YAML description file `filemask.info` or an Andor IQ acquisition protocol `filemask.txt` to be present in the SIM data directory.

An image stack `IMseq` at a particular axial position `Z`, channel `W` and time `T` can be loaded by:

```
IMseq = seqload(imginfo, 'z', Z, 'w', W, 't', T);
```

Conversion of an image stack `IMseq` to an image sequence `seq` is based on the illumination pattern sequence and is performed by:

```
repzfilename = ptrndirinfo(datadir);
ptrninfo = ptrnopen(repzfilename);
seq = seq2subseq(IMseq, ptrninfo, numro);
```

Here `repzfilename` is a string with a path to the illumination pattern repertoire, `ptrninfo` contains information about the repertoire, and `numro` is a scalar specifying the running order of the repertoire used to acquire the data.

6.5 Installation of the MATLAB functions (not necessary if the standalone application GUI is used)

1. Unzip the file ***SIMToolbox.zip*** with the toolbox into your working directory. It also contains the YAML package [9]. The file can be downloaded from the project website: http://mmtg.fel.cvut.cz/SIMToolbox.

2. Open MATLAB and change the path to the toolbox directory.

3. Run the user interface by running the function ***SIMToolbox***. It initializes all necessary paths by itself.

4. Optionally, we recommend to move the YAML package into your personal toolbox directory so other programs can eventually access it. The path to the package can be initialized in your personal MATLAB start-up script ***startup.m*** by the command ***addpath('PathToToolboxes\yaml')***.

5. Optionally, we also recommend including in the MATLAB start-up script the command ***javaaddpath('PathToToolboxes\yaml\java\snakeyaml-1.9.jar')*** and to comment ***javaaddpath(YAML.JARFILE)*** in functions ***YAML.load*** and ***YAML.dump***. The function ***javaaddpath*** causes clearing of all break points and workspace variables which results in loss of the data every time you load or save a YAML file.

7 (Appendix) List of main library functions

This is a list of most important functions available in the toolbox and a brief description. For more details, see the comments and examples in the code.

7.1 Basic IO operations

`imginfoinit`	initialize structure with information about the acquired data
`imgload`	load image at specified position (Z,T,W)
`imgzload`	load z-stack at specified position (T,W)
`seqload`	load SIM image stack at specified position (Z,T,W)
`imgsave16`	save image to a 16 bit TIFF (appends to existing file)
`imgsave32`	save image to a 32 bit TIFF (appends to existing file)
`isfile`	test if file exists
`fileparts_dir`	extract directory from a string
`fileparts_name`	extract file name without extension
`fileparts_nameext`	extract file name with extension

7.2 Basic image processing

`imgflatfield`	flat field correction
`seqflatfield`	flat field correction for every image in a sequence
`seqstriperemoval`	correction of intensity fluctuations in an image sequence
`fspecialseparable`	generate kernels for separable convolution
`imfilterseparable`	image filtering using separable convolution
`seqblure`	filter every image in a sequence using separable convolution
`imgradprof`	compute average radial intensity profile of an image

7.3 Illumination patterns

`ptrndirinfo`	check if a pattern repertoire is stored in data directory
`ptrnopen`	open pattern repertoire
`ptrnload`	load image of illumination pattern from a repertoire
`prnclose`	close pattern repertoire and delete temporary files
`ptrn2camera`	map illumination pattern from microdisplay to camera
`ptrnmaskprecompute`	precompute illumination mask for SIM sequence
`ptrnchecklines`	test if running order contains only line-grid patterns
`ptrngetro`	get structure with specified running order
`ptrngetnumro`	get number of running orders in a repertoire
`ptrngetnumseq`	get number of illumination patterns in a running order
`ptrngetimagenames`	get names of images in a particular running order

7.4 Calibration of illumination patterns

`calload`	load calibration file
`calw2i`	map microdisplay coordinates to image coordinates
`cali2w`	map image coordinates to microdisplay coordinates
`calprocess`	camera-microdisplay calibration process
`calconfig`	configuration of the calibration process
`markers_detect`	detect circular markers on a chessboard
`markers_sort`	sort markers to a particular ordering
`corners_detect`	detect corners on a chessboard image
`calloadgdf`	load chessboard definition from a repertoire
`calhomolin`	compute linear homography between microdisplay and camera
`calsave`	save calibration file

7.5 OS-SIM processing methods

`seqwf`	widefield image computed from an image sequence
`seqmemlim`	set memory limit for processing image sequences

Simple methods

`seqcfmaxmin`	max-min scheme
`seqcfmaxmin2av`	max+min-2av scheme
`seqcfhomodyne`	homodyne detection scheme

Methods with illumination mask

`seqcfmaskonly`	conjugate image
`seqcfscaledsub1px`	simple scaled subtraction scheme
`seqcfscaledsub`	scaled subtraction scheme
`seqcfscaledsubbeta`	scaled subtraction scheme with correction scheme

7.6 SR-SIM processing methods

`seqpadsmooth`	pad image sequence and smooth edges by a sigmoid function
`imgrmpadding`	remove image padding
`seqfft2`	fast Fourier transform applied to every image in a sequence
`seqifft2`	inverse fast Fourier transform applied to every image in a sequence
`sim_findpeaksassign`	calculate position of peaks of individual spectral components
`sim_findpeaks`	search for peaks in Fourier space
`sim_findpeaksloadimage`	load and prepare image for pattern estimation

Gustafsson-Heintzmann method

`sim_extract`	extract spectral components from a sequence of images
`sim_shiftspectra`	shift the extracted FFT components to their proper positions in Fourier space
`sim_addOTF`	generate shifted OTFs
`sim_combine`	combine the shifted spectra and apodize the reassembled image
`sim_matchspectra`	match spectral components based on the central component as a reference

MAP-SIM

`mapsim`	reconstruct super-resolution image using MAP-SIM method according to [6]
`genMasks`	generate the illumination masks based on the estimated pattern properties

7.7 Optical transfer functions and apodizing functions

`apodize_none`	no filter
`apodize_butterworth`	2D Butterworth filter
`apodize_circular`	2D circular filter
`apodize_cosbell`	2D cosine bell filter (Hanning window)
`apodize_gauss`	2D Gaussian filter
`apodize_lukosz`	2D Lukosz filter
`apodize_standard`	2D standard incoherent filter
`apodize_triangle`	2D triangle function filter (Bartlett window)

7.8 Local intensity maximum detection

`spotfinder`	detection and position refinement of local intensity maxima

Filtering and feature enhancement

`filter_none`	no filter
`filter_average`	2D low-pass averaging filter
`filter_gauss`	2D low-pass Gaussian filter
`filter_diffav`	2D band-pass filter based on Difference-of-Averaging filter
`filter_dog`	2D band-pass filter based on Difference-of-Gaussians filter
`filter_dao`	2D band-pass filter based on shifted Gaussian filter
`filter_wavelet`	2D band-pass filter based on wavelets

Thresholding and spot detection

`detector_locmax`	detection based on finding local intensity maxima
`threshold_val`	threshold by a constant
`threshold_std_val`	threshold by standard deviation
`threshold_mustd_val`	threshold by mean and standard deviation
`threshold_fnc`	threshold by a user defined function

Sub-pixel refinement

`estimator_none`	no estimator
`estimator_lsq_gradintersect`	refinement by intersection of local gradients
`estimator_lsq_fitquadric`	refinement by fitting a quadratic function

8 (Appendix) Supplementary information: Figure 1

Here we further describe Figure 1 from the main paper.

Cell lines and reagents:

HEP-G2 cells were maintained in phenol red-free DMEM supplemented with 10 % FCS, 100 U/ml penicillin, 100 U/ml streptomycin, and L-glutamate (all from Invitrogen, Carlsbad, CA, USA) at 37 °C and 100 % humidity. Mowiol containing DABCO was purchased from Fluka (St. Louis, MO, USA). Atto565-phalloidin was obtained from atto-tec (Siegen, Germany).

Sample preparation:

Cells were grown on high precision #1.5 coverslips (available from Zeiss, or from Marienfield). The cells were washed with 1X PBS, then fixed with 4% paraformaldehyde at room temperature for 15 minutes. The cells were then permiabilized with 0.1% triton X-100 (Sigma) for 5 minutes at room temperature. We then incubated the cells with 5 nm Atto565-phalloidin for 30 minutes at room temperature. We then washed the coverslips several times with 1X PBS, mounted the coverslips on clean slides with Mowiol, and sealed them with clear nail polish. We typically waited 24 hours for the Mowiol to harden before imaging.

Microscopy:

The microscopy system has been previously described [6, 7]. The system is further described in Section 4.

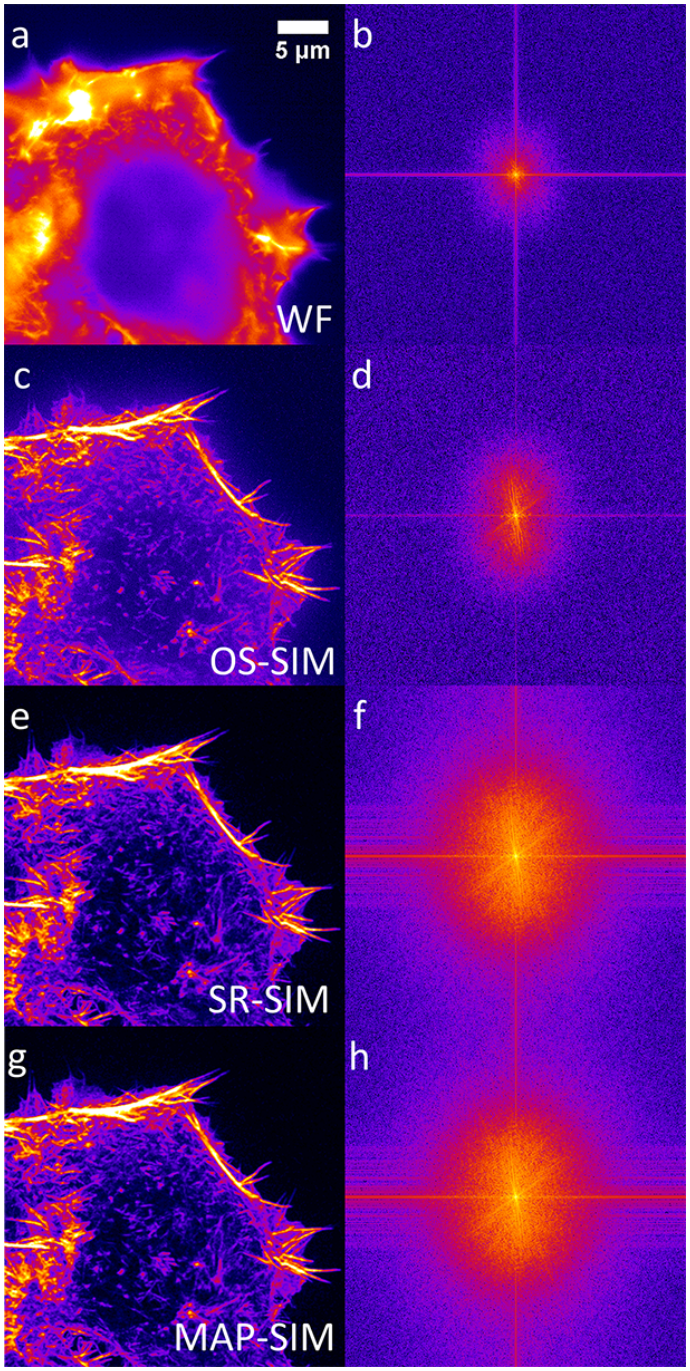


Figure 23: (Also Fig. 1 in the main paper. Widefield (WF) image, and OS-SIM, SR-SIM, and MAP-SIM reconstructions of images of a HEP-G2 cell in which the actin cytoskeleton has been labeled with Atto565-phalloidin. (a) WF. (b) FFT of WF. (c) OS-SIM, homodyne method. (d) FFT of OS-SIM image. (e) SR-SIM processing. (f) FFT of SR-SIM. (g) MAP-SIM. (h) FFT of MAP-SIM. Images are maximum intensity projections of 31 axial sections and have been individually brightness-adjusted for presentation. This data is a cropped region from: "TestFiles/SIM_LCOS/Actin_LCOS".

References

[1] R. Heintzmann. Structured illumination methods. In J. B. Pawley, editor, *Handbook of Biological Confocal Microscopy*, pages 265–279. Springer Science+Business Media, LLC, New York, 2006.

[2] T. Wilson. Optical sectioning in fluorescence microscopy. *Journal of Microscopy*, 242:111–116, 2011.

[3] M. A. A. Neil, R. Juškaitis, and T. Wilson. Method of obtaining optical sectioning by using structured light in a conventional microscope. *Optics Letters*, 22:1905–1997, 1997.

[4] M. G. L. Gustafsson. Surpassing the lateral resolution limit by a factor of two using structured illumination microscopy. *Journal of Microscopy*, 198:82–87, 2000.

[5] R. Heintzmann and C. Cremer. Laterally modulated excitation microscopy: improvement of resolution by using a diffraction grating. In *Proc. SPIE 3568*, volume 3568, pages 185–196, 1998.

[6] T. Lukeš, P. Křížek, Z. Švindrych, J. Benda, M. Ovesný, K. Fiegel, M. Klíma, and G. M. Hagen. Three dimensional super-resolution structured illumination microscopy with maximum a posteriori probability image estimation. *Optics Express*, 22, 2014.

[7] P. Křížek, I. Raška, and G. M. Hagen. Flexible structured illumination microscope with a programmable illumination array. *Optics Express*, 20(22):24585–24599, 2012.

[8] T. Lukeš, G. M. Hagen, P. Křížek, Z. Švindrych, K. Fligel, and M. Klíma. Comparison of image reconstruction methods for structured illumination microscopy. In *In Proc. SPIE 9129, Biophotonics: Photonic Solutions for Better Health Care IV*, page 91293J, 2014.

[9] K. Yamaguchi. Matlab YAML. Graduate School of Information Sciences, Tohoku University, Japan. Available at http://www.cs.sunysb.edu/~kyamagu/software/yaml/, 2011.

[10] P. Křížek and G. M. Hagen. Current optical sectioning systems in fluorescence microscopy. In A. Méndez-Vilas, editor, *Current Microscopy Contributions to Advances in Science and Technology*, Microscopy Series no. 5, pages 826–832. Formatex Research Center, Badajoz, Spain, 2012.

[11] M. J. Cole, J. Siegel, S. E. D. Webb, R. Jones, K. Dowling, M. J. Dayel, D. Parsons-Karavassilis, P. M. W. French, M. J. Lever, L. O. D. Sucharov, M. A. A. Neil, R. Juskaitis, and T. Wilson. Time-domain whole-field fluorescence lifetime imaging with optical sectioning. *Journal of Microscopy*, 203:246–257, 2001.

[12] D. Karadaglic and T. Wilson. Image formation in structured illumination wide-field fluorescence microscopy. *Micron*, 39:808–818, 2008.

[13] R. Heintzmann and P. A. Benedetti. High-resolution image reconstruction in fluorescence microscopy with patterned excitation. *Applied Optics*, 45:5037–5045, 2006.

[14] P. A. Benedetti, V. Evangelista, D. Guidarini, and S. Vestri. Us patent 6,016,367.

[15] R. Heintzmann, Q. S. Hanley, D. J. Arndt-Jovin, and T. M. Jovin. A dual path programmable array microscope (PAM): simultaneous acquisition of conjugate and non-conjugate images. *Journal of Microscopy*, 204:119–135, 2001.

[16] G. M. Hagen, W. Caarls, M. Thomas, A. Hill, K. A. Lidke, B. Rieger, C. Fritsch, B. van Geest, T. M. Jovin, and D. J. Arndt-Jovin. Biological applications of an LCoS-based programmable array microscope. In *Proc. SPIE 64410S*, volume 64410S, pages 1–12, 2007.

[17] G. M. Hagen, W. Caarls, K. A. Lidke, A. H. B. deVries, C. Fritsch, B. G. Barisas, D. J. Arndt-Jovin, and T. M. Jovin. FRAP and photoconversion in multiple arbitrary regions of interest using a programmable array microscope (PAM). *Microscopy Research and Technique*, 72:431–440, 2009.

[18] A. H. B. de Vries, N. P. Cook, S. Kramer, D. J. Arndt-Jovin, and T. M. Jovin. Generation 3 programmable array microscope (PAM) for high speed, large format optical sectioning in fluorescence. In M. R. Douglas, editor, *Emerging Digital Micromirror Device Based Systems and Applications VII*, volume 9376, page 93760C. SPIE, 2015.

[19] L. Gao, L. Shao, C. D. Higgens, J. S. Poulton, M. Peifer, M. W. Davidson, X. Wu, B. Goldstein, and E. Betzig. Noninvasive imaging beyond the diffraction limit of 3D dynamics in thickly fluorescent specimens. *Cell*, 151:1370–1385, 2012.

[20] L. M. Hirvonen, K. Wicker, O. Mandula, and R. Heintzmann. Structured illumination microscopy of a living cell. *European Biophysics Journal*, 38:807–812, 2009.

[21] P. Kner, B. B. Chhun, E. R. Griffis, L. Winoto, and M. G. L. Gustafsson. Super-resolution video microscopy of live cells by structured illumination. *Nature Methods*, 6:339–342, 2009.

[22] E. H. Rego, L. Shao, J. J. Macklin, L. Winoto, G. A. Johansson, N. Kamps-Hughes, M. W. Davidson, and M. G. L. Gustafsson. Nonlinear structured-illumination microscopy with a photoswitchable protein reveals cellular structures at 50-nm resolution. *Proceedings of the National Academy of Sciences of the United States of America*, 109:E135–E143, 2012.

[23] L. Shao, P. Kner, E. H. Rego, and M. G. L. Gustafsson. Super-resolution 3D microscopy of live whole cells using structured illumination. *Nature Methods*, 8:1044–1046, 2011.

[24] R. Fiolka, L. Shao, E. H. Rego, M. W. Davidson, and M. G. L. Gustafsson. Time-lapse two-color 3d imaging of live cells with doubled resolution using structured illumination. *Proceedings of the National Academy of Sciences of the United States of America*, 109:531–5315, 2012.

[25] L. Shermelleh, P. M. Carlton, S. Haase, L. Shao, L. Winoto, P. Kner, B. Burke, M. C. Cardoso, D. A. Agard, M. G. L. Gustafsson, H. Leonhardt, and J. W. Sedat. Subdiffraction multicolor imaging of the nuclear periphery with 3D structured illumination microscopy. *Science*, 320:1332–1336, 2008.

[26] M. Ahn, T. Kim, Y. Kim, D. Gweon, and J. Lee. Cross structured illumination for high speed high resolution line scanning confocal microscopy. *Measurement Science and Technology*, 22:015503, 2011.

[27] M. Shaw, L. Zajiczek, and K. O'Holleran. High speed structured illumination microscopy in optically thick samples. *Methods*, 88:11–19, 2015.

[28] E. Sanchez-Ortega, A. Doblas, G. Saavedra, and M. Martinez-Corral. Novel proposals in widefield 3D microscopy. In Bahram Javidi, editor, *Display Technologies and Applications for Defense, Security, and Avionics IV*, volume 7690, page 769005. SPIE, 2010.

[29] J. H. Park, J. Y. Lee, and E. S. Lee. Enhancing the isotropy of lateral resolution in coherent structured illumination microscopy. *Biomedical Optics Express*, 5:1895–1912, 2014.

[30] B. Chang, L. Chou, and Y. Chang. Isotropic image in structured illumination microscopy patterned with a spatial light modulator. *Optics Express*, 17:14710–14721, 2009.

[31] M. G. Somekh, K. Hsu, and M. C. Pitter. Stochastic transfer function for structured illumination microscopy. *J. Opt. Soc. Am. A*, 26, 2009.

[32] F. Orieux, E. Sepulveda, V. Loriette, B. Dubertret, and J.-C. Olivo-Marin. Bayesian estimation for optimized structured illumination microscopy. In *in IEEE Transactions on Image Processing vol. 21*, pages 601–614, 2012.

[33] M. Beck, M. Aschwanden, and A. Stemmer. Sub-100-nanometre resoution in total internal reflection fluorescence microscopy. *Journal of Microscopy*, 232:99–105, 2008.

[34] M. G. L. Gustafsson, L. Shao, P. M. Carlton, C. J. Wang, I. N. Golubovskaya, W. Z. Cande, D. A. Agard, and J. W. Sedat. Three-dimensional resolution doubling in wide-field fluorescence microscopy by structured illumination. *Biophysical journal*, 94(12):4957–4970, 2008.

[35] C. H. Righolt, J. A. Slotman, I. T. Young, S. Mai, L. J. van Vliet, and S. Stallinga. Image filtering in structured illumination microscopy using the lukosz bound. *Optics Express*, 21(21):24431–51, 2013.

[36] Rimas Juškaitis. Measuring the real point spread function of high numerical aperture microscope objective lenses. In *Handbook of Biological Confocal Microscopy: Third Edition*. Springer, Boston, MA, 2006.

[37] T. C. Schlichenmeyer, M. Wang, K. N. Elfer, and J. Q. Brown. Video-rate structured illumination microscopy for high-throughput imaging of large tissue areas. *Biomedical Optics Express*, 5:366–377, 2014.

[38] S. Monneret, M. Rauzi, and P. F. Lenne. Highly flexible whole-field sectioning microscope with liquid-crystal light modulator. *Journal Optics and Pure Applied Optics*, 8:S461–S466, 2006.

[39] W. Caarls, B. Rieger, A. H. B. de Vries, D. J. Arndt-Jovin, and T. M. Jovin. Minimizing light exposure with the programmable array microscope. *Journal of Microscopy*, 241:101–110, 2010.

[40] B. Chen, W. Legant, K. Wang, L. Shao, D. E. Milkie, M. W. Davidson, C. Janetopoulos, X. S. Wu, J. A. Hammer, Z. Liu, B. P. English, Y. Mimori-Kiyosue, D. P. Romero, A. T. Ritter, J. Lippincott-Schwartz, L. Fritz-Laylin, R. D. Mullins, D. M. Mitchell, J. N. Bembenek, A. Reymann, R. Bohme, S. W. Grill, J. T. Wang, G. Seydoux, U. S. Tulu, D. P. Kiehart, and E. Betzig. Lattice light-sheet microscopy: Imaging molecules to embryos at high spatiotemporal resolution. *Science*, 346:1257998, 2014.

[41] R. Heintzmann. Saturated patterned excitation microscopy with two-dimensional excitation patterns. *Micron*, 34(6):283–291, 2003.

[42] Andrew G York, Sapun H Parekh, Damian Dalle Nogare, Robert S Fischer, Marina Mione, Ajay B Chitnis, Christian A Combs, and Hari Shroff. Resolution doubling in live, multicellular organisms via multifocal structured illumination microscopy. *Nat Methods*, 9(7):749–754, 2012.

[43] M. Šonka, V. Hlaváč, and R. Boyle. *Image Processing Analysis and Machine Vision*. PWS Publishing, Boston, 2nd ed. edition, 1998.

[44] R. Kimmel, N. Kiryati, and A. M. Bruckstein. Distance maps and weighted distance transforms. *Journal of Mathematical Imaging and Vision, Special Issue on Topology and Geometry in Computer Vision*, 6:223–233, 1996.

[45] R. Parthasarathy. Rapid, accurate particle tracking by calculation of radial symmetry centers. *Nature Methods*, 9(7):724–6, 2012.