

This 'readme' file describes how to use the following files to implement single-pixel holography (SPH) and reconstruct holographic images.

Statement of system requirements:

1. All the files in this .zip package can be successfully executed using the MATLAB v2016a (including the later versions) and the Visual Studio 2013 (including the later versions).
2. In this work, we run these files with the MATLAB v2020b and the Visual Studio 2019 community version. The operating system is Windows 10.
3. During the data acquisition, the physical presence of the digital micromirror device (DMD) and the data acquisition card (DAC) is required to execute respective C++ files successfully. Both the DMD and the DAC are the required non-standard hardware for the experiment.

The installation guide of MATLAB and Visual Studio can be found on the official website. It generally takes about one hour of installation time on a desktop personal computer.

Contents:

This package of 'Data_and_supporting_files' includes three separate parts with respective functionalities. The corresponding file folders are ordered based on the sequence during experiments:

*) Preparation of the Hadamard-like patterns (MATLAB code)

*) Image acquisition for SPH (C++ code)

*) Holographic reconstruction (MATLAB code)

Preparation of the Hadamard-like patterns (MATLAB)

The file 'DMD_hadamard256.m' in this file folder is used to generate Hadamard-like patterns with orders up to 256×256 . Hadamard-like patterns with different orders can be generated as well by adjusting the parameter "number" in this file. The execution of this file with commercialized software MATLAB generates a sequence of Hadamard-like patterns in the format of .txt binary files. To minimize phase drifting caused by environmental disturbance during the experiment, the first order of Hadamard-like patterns $\tilde{H}_1(\vec{r})$ (DC signal) is inserted after every 16 modulation patterns as the tracking base. Therefore, 73,728 patterns are generated in a sequence and it takes about 25 minutes to finish this one-time preparation.

A DMD (Texas Instrument DLP7000&DLPC410) is employed to project Hadamard-like patterns to the sample. The software and the control board of the DMD are provided by Vialux company (model number Vialux V7001). The Hadamard-like patterns in the format of .txt binary files can be uploaded to the random-access memory (RAM) in advance. Unfortunately, we are unable to provide a series of .txt files of Hadamard-like patterns in this folder as it occupies a large memory of about 50 Gbit (less than the 64Gbit of the RAM of the DMD).

Image acquisition for SPH (C++ code)

This file folder contains three subfolders, which are named 'joint_256bitplane8', 'include_header_DMD' and 'include_header_NIdaq'.

*) File folder of 'joint_256bitplane8':

This folder includes some necessary files of the Visual Studio C++ project to control both the DMD and the DAC. In this project, the execution of an integrated C++ source file named 'joint_256bitplane8.cpp' enables the DMD to display the desired Hadamard-like patterns with full speed and the DAC to acquire data subsequently. After preloading the desired Hadamard-like patterns onto the DMD, it takes about 3 seconds to finish the data acquisition.

Please visit the website <https://visualstudio.microsoft.com/zh-hans/> to download the Visual Studio 2019 community version.

*) File folder of 'include_header_DMD':

This folder contains a header file alp.h, and corresponding import library LIB files. Including alp.h and linking the LIB files to the Visual Studio C++ project are prerequisite to controlling the DMD (The header declares functions and constant values, while the LIB files load the DLL file and import the recognized functions in the DMD when starting the C++ project).

Please visit the website <https://www.vialux.de/en/download.html> to download control files.

*) File folder of 'include_header_NIdaq':

This folder contains a series of header files and corresponding import library LIB files. Including header files and linking the LIB files to the Visual Studio C++ project are prerequisite to controlling the DAC (These header files declare functions and constant values, while the LIB files load the DLL file and import the recognized functions in the DAC when starting the C++ project).

Please visit the website

<https://www.ni.com/zh-cn/support/downloads/drivers/download.ni-daqmx.html#288248>

to download control files.

The successful execution of these C++ files requires the correct configuration of environment variables in a “normal” desktop computer as well as the physical presence of the DMD and the DAC.

Holographic reconstruction (MATLAB)

This folder contains the main MATLAB file of 'singlepixel_dataprocessing_complex.m' for holographic reconstruction, supporting subfunctions, and an example of reconstruction with raw data.

Here we introduce in sequence:

*) singlepixel_dataprocessing_complex.m: This file reads the measured raw data and calls all the supporting subfunctions to reconstruct holographic images using compressive sensing with different sampling ratios. Subfunctions such as correcting system-induced phase contaminations and phase drifting during the data acquisition process are included. The total execution time of this main file takes about 45 minutes, including

holographic reconstruction with and without the sample, removing phase contaminations, and the denoising process.

*) `biological_sample.mat`: The raw data of imaging a slice of rat tail. The format of the data has been converted from `.tdms` to `.mat` file.

*) `background_curvature.mat`: The raw data used to correct for phase contaminations from system aberrations. The format of the data has been converted from `.tdms` to `.mat` file.

*) `biological_sample_rawdata.tdms`: The raw data of imaging a slice of rat tail. The data was collected through DAC and was in the format of TDMS.

*) `background_curvature_rawdata.tdms`: The raw data used to correct for phase contaminations from system aberrations. The data was collected through DAC and was in the format of TDMS.

*) `smoothfit.m`: A custom-designed fitting function to correct the phase drifting during data acquisition.

*) `imaging-amplitude.fig`: The amplitude image reconstructed in this example.

*) `imaging-phase.fig`: The phase image reconstructed in this example.

*) `mcolor2.mat`: A custom-designed colormap of phase distribution in consideration of phase wrapping.

*) `BM3D.m`: A denoising algorithm that employs block-matching 3D (BM3D) filtering.

*) file folder of 'results': A folder contains output results including many intermediate variables generated during the reconstruction process. For example, it contains holographic output with and without compressive sensing, crude images before correcting phase contaminations, computed field coefficients of each pattern, and the corresponding statistical property. Note that these files in the folder 'results' will be rewritten when running either 'singlepixel_dataprocessing_complex.m' or 'image_processing_CS.m'.