CGDH Tools: Getting started in computer generated display holography

Petr Lobaz
Dept. of Computer Science and Engineering, Faculty of Applied Sciences,
University of West Bohemia, Pilsen, Czech Republic

ABSTRACT

First steps in computer generated display holography can be difficult for both students, researchers and engineers. One of the main reasons is that in order to make a computer generated hologram, one has to implement the whole chain that calculates optical fields, makes a hologram and allows to simulate its reconstruction. The aim of CGDH Tools is to provide the basic building blocks of such a chain and to show a few case studies that demonstrate step by step how to use them. CGDH Tools is a set of Octave/MATLAB/Scilab functions and scripts that try to be as simple and clear as possible; moreover, they are extensively documented. The functionality includes light propagation calculations (convolution based, angular spectrum based and single Fourier transform based with optional local frequency clipping), object wave calculations (point cloud, wavefront recording plane, stereogram, billboard based method, etc.), hologram observation simulation and various handy utilities. CGDH Tools can be used for teaching holography or wave optics, for simple diffractive optical element design or it can be used as a starting point when implementing advanced calculation methods. CGDH Tools can be freely downloaded at holo.zcu.cz.

Keywords: computer generated holography, display holography, Fourier optics, script, Matlab, Octave, Scilab

1. INTRODUCTION AND RELATED WORK

Holography is a method of capturing and reconstructing light that can be used (among others) for the creation of ultra-realistic 3-D images[1]. Digital holography tries to employ digital electronics somewhere in the process, e.g. by capturing light using a CCD sensor, modification of captured data by means of signal processing or modifying light using a spatial light modulator[2]. Computer generated holography tries to get rid of capturing real light – it tries to generate a hologram (a special diffractive structure) computationally[3]. Computer generated display holography is a special kind of computer generated holography. It tries to calculate a hologram of a 3-D scene for display purposes. It can be thought as an extension of computer graphics. While computer graphics tries to make a synthetic photograph of a 3-D scene, computer generated display holography tries to make a synthetic hologram of such a scene. The resulting hologram is a special pattern that creates a perfect illusion of the 3-D scene when "printed" and properly illuminated.

Computer generated display holography is quite difficult for beginners. It requires programming skills, proficiency in digital signal processing, Fourier optics and classical (optical) holography. Additional skills such as computer graphics, photography, 3-D imaging or geometrical optics are more than welcome. Anyone who wants to start with computer generated display holography faces an uncomfortable problem: even the most simple task requires implementation of the whole imaging chain. This means that simulation of light wave of a 3-D scene, hologram recording and hologram reconstruction must be implemented somehow, usually computationally. Moreover, parameters of the calculation such as angles, distances or light intensities must be chosen carefully. If anything goes wrong, the output of the imaging chain is usually completely useless. For a beginner, it is close to impossible to track the origin of the error.

Unfortunately, a beginner has usually no simple enough base to start with. Most texts on holography use mathematical expressions that are far away from working code. Research articles usually do not contain any code, or the code does not cover the whole imaging chain. There are some monographs with example scripts, but 1. they are not focused on display holography, 2. they usually provide only isolated pieces of code, not the whole chain. For example, Voelz[4] or Schmidt[5] show many scripts for light propagation simulation, lens simulation, etc., but do not show any examples related to holography. Picart[6], Poon and Liu[2] or Poon and Banerjee[7] show scripts for digital holography, but they mostly focus on technical holography rather than display holography of 3-D scenes. Vijayakumar and Bhattacharya[3] discuss computer generated holography in context of technical diffractive optical elements only. Here "technical holography" means

holography for other purposes than 3-D display, such as for laser beam shaping, digital holographic microscopy, surface testing, etc.

There are also some software implementations available, but they are either too simple (just a few techniques) or not educational. Shimobaba[8] and Matsushima[9] describe production-ready libraries for computer generated display holography that are not intended to explain basic principles. MIT Media Lab published some of their hologram generation software tools[10], but again they can be hardly useful for a beginner. Müller[11] provides beginner-friendly tools for light propagation, but any extension to computer generated display holography is missing. Some basic scripts were provided by Lobaz[12] as a part of a tutorial focused on computer generated display holography, but the scripts showed a complete imaging chain with just one simple hologram generation technique.

2. OBJECTIVES OF CGDH TOOLS

"CGDH Tools" is a set of functions for Octave/MATLAB/Scilab distributed with demonstration scripts showing either some details or complete imaging chains. The main purpose of CGDH Tools is educational. This means that the code tries to be as simple, short and readable as possible, it focuses mainly on ideas, not tricky optimizations. This makes porting to other languages straightforward. Furthermore, CGDH Tools can provide reference results when implementing optimized, high-performance versions.

The educational intent determined the programming language. MATLAB[13] is a widespread general environment used for numerical calculations and many opticians, as well as computer scientists, can use it. Its programming language allows to express mathematical ideas very efficiently, it is easy to analyze and display the results, it is easy to play with the code. Octave[14] is an open source alternative to MATLAB and it is, in fact, the main development platform. Scilab[15] is also an open source alternative to MATLAB that uses slightly different syntax; its main advantage over the current version of Octave is support of large matrices.

While CGDH Tools intent is "just educational", it can be used for simple production work as well. For instance, it has been used to make commercial diffractive structures that project an image at a specified distance; or to explore some unusual effects in security holograms.

While the code tries to be simple and short, the calculations try to be as general as possible if it does not obfuscate the main idea. For example, light propagation calculations based on convolution do not require the same size of the source and the target optical field, which is a usual restriction in most implementations. Such a "general implementation" does not make the code more difficult to understand and it is easy to optimize the code if this feature is not necessary. Moreover, implemented methods try to be as exact as possible. For example, constant factors are often dropped in practical calculations as the results are usually normalized anyway. On the other hand, if there are several methods that calculate the same thing, it is more difficult to compare them or to use them interchangeably when constant factors have been dropped.

The code tries to emphasize similarities between various methods and encourages to understand differences. For example, calculations in holography use special functions known as the spherical wave, the Rayleigh-Sommerfeld convolution kernel, the Fresnel convolution kernel, etc. It is necessary to evaluate these functions, to evaluate their local frequencies, etc. It would be possible to implement each evaluation as a single script, but such implementation would hide similarities between the functions. Thus, there is usually one script that takes a string parameter, for example **'SphericalWave'** or **'RayleighSommerfeldExact'**, and the code is organized in such a way it is obvious that some functions have many common properties.

The most important aspect of CGDH Tools is that they are built around *demos* – scripts that show the whole imaging chains, for example making a hologram of an image and reconstructing its real image. Each demo contains many parameters to play with. For example, it is possible to select between ready-made experiments, choose various hologram recording methods, etc. The demos explain principles of the methods and at the same time, they show how to use CGDH Tools core functions. It is worth emphasizing that demos also show examples of calculation parameters that really work.

The code tries to use easy to understand variable names (for example, **referenceWave** and **objectWave** instead of more common **r** and **o**). The code tries to avoid "magic constants" – for example, instead of **takePhotograph(..., 100e-3, ...)**, it uses **focalLength = 100e-3; takePhotograph(..., focalLength, ...)**. Last but not least, the functions and demos are extensively commented.

3. SIMPLE EXAMPLE

Let us show a simple example of using CGDH Tools. The following script first calculates an off-axis hologram of an image with a diffuse surface. Subsequently, the script illuminates the hologram with a suitable reconstruction wave and calculates a real image formed on a screen.

First, we choose a suitable wavelength and prepare the source image with a diffuse surface. All dimensions are in meters, i.e. the source image dimensions are 10 mm × 10 mm (USAF target plus a wide black border, see Figure 1). It is assumed that all planes in the calculation are parallel to the plane $z = 0$. Note that point coordinates are written as row vectors [x, y, z]. Functions in **bold** style are part of CGDH Tools.

```
lambda = 500e-9;
source = loadImage('image.png');
source = applyRandomPhase(source);
sourceHeight = 10e-3;
sourceWidth = 10e-3;
sourceCenter = [0, 0, 0];
```

Determine sampling distance (distance between pixel centers) of the source image. Please note that sampling distances can be different in x and y directions.

```
sourceSize = size(source);
sourceDeltaYX = calculateDelta(sourceSize, [sourceHeight, sourceWidth]);
```

Size of the hologram will be the same as the source image size. Let us place the hologram 700 mm from the source image.

```
hologramSize = sourceSize;
hologramDeltaYX = sourceDeltaYX;
hologramCenter = [0, 0, 0.7];
```

The screen used for projection of the real image will be placed 700 mm from the hologram, its size and sampling distance will be the same as of the source image.

```
screenSize = sourceSize;
screenDeltaYX = sourceDeltaYX;
screenCenter = [0, 0, 1.4];
```

The function **propagate** is one of the most important functions in CGDH Tools. It propagates an optical field given in the source plane to a given area in the target plane. Currently, both source and target planes must be parallel to the plane $z = 0$. The function **propagate** can use various algorithms to do the job – for example, convolution in the spatial domain (using 3 fast Fourier transforms) with the exact Rayleigh-Sommerfeld convolution kernel[1]. Moreover, it can filter out high local frequencies to avoid aliasing. For convenience, the function **propagate** returns x and y coordinates of the target plane samples together with the complex matrix describing the optical field.

```
[objectWave, hologramX, hologramY] = propagate(source, sourceDeltaYX, sourceCenter, …
  hologramSize, hologramDeltaYX, hologramCenter, …
  lambda, '3fft', 'RayleighSommerfeldExact', 'localExact');
```

We are going to use a plane reference wave. Its direction vector is given by angles α and β. The angle α is the angle between the axis x and the direction vector minus $\pi/2$, the angle β is given with respect to the axis y. Thus, $\alpha = \beta = 0$ gives a direction vector [0, 0, 1].

For an off-axis hologram, we need a direction vector that is not parallel to the z axis. Moreover, the plane wave should be representable with given sampling distance, for example twice as large as is the current one. Note that **hologramDeltaYX(1)** is the sampling distance in the y direction.

```
referenceAlpha = 0;
referenceBeta = getMaxDiffAngle(lambda, 2.0*hologramDeltaYX(1));
referenceDirection = getVectorFromAngles(referenceAlpha, referenceBeta);
```

Make a suitable reconstruction wave that creates a perfect real image on the screen:

```
reconstructionAlpha = -referenceAlpha;
reconstructionBeta = -referenceBeta;
reconstructionDirection = getVectorFromAngles(reconstructionAlpha, reconstructionBeta);
```

The reference wave amplitude should be comparable to the object wave amplitude, i.e. the square root of the object wave intensity. The reconstruction wave amplitude can be arbitrary, for example 1.

```
objectWaveIntensity = getWaveIntensity(objectWave);
[xx, yy] = meshgrid(hologramX, hologramY);
referenceWave = sqrt(objectWaveIntensity) * …
  planeWave(referenceDirection, lambda, xx, yy, hologramCenter(3));
reconstructionWave = …
  planeWave(reconstructionDirection, lambda, xx, yy, hologramCenter(3));
```

CGDH Tools provides several methods for calculating a hologram with given object and reference waves. Here we are going to use the physically based one, i.e. simulation of their interference.

```
hologram = calculateHologram(objectWave, referenceWave, 'classical');
```

Finally, we can illuminate the hologram with the reconstruction wave and propagate the optical field to the screen plane. For convenience, CGDH Tools provides a function that displays an image in various useful ways; here we display the absolute value of the optical field on the screen, i.e. the square root of its intensity, see Figure 1. Please note that most displays require a gamma-corrected input[16], i.e. a pixel value should be equal to $intensity^{1/\gamma}$, where γ is typically 2.2. The absolute value of the optical field is the square root of its intensity, i.e. it is a reasonable approximation of the gamma-corrected input.

```
[screenWave, screenX, screenY] = propagate(…
  hologram .* reconstructionWave, hologramDeltaYX, hologramCenter, …
  screenSize, screenDeltaYX, screenCenter, …
  lambda, '3fft', 'RayleighSommerfeldExact', 'localExact');

displayImage(screenX, screenY, abs(screenWave), 'positive');
```

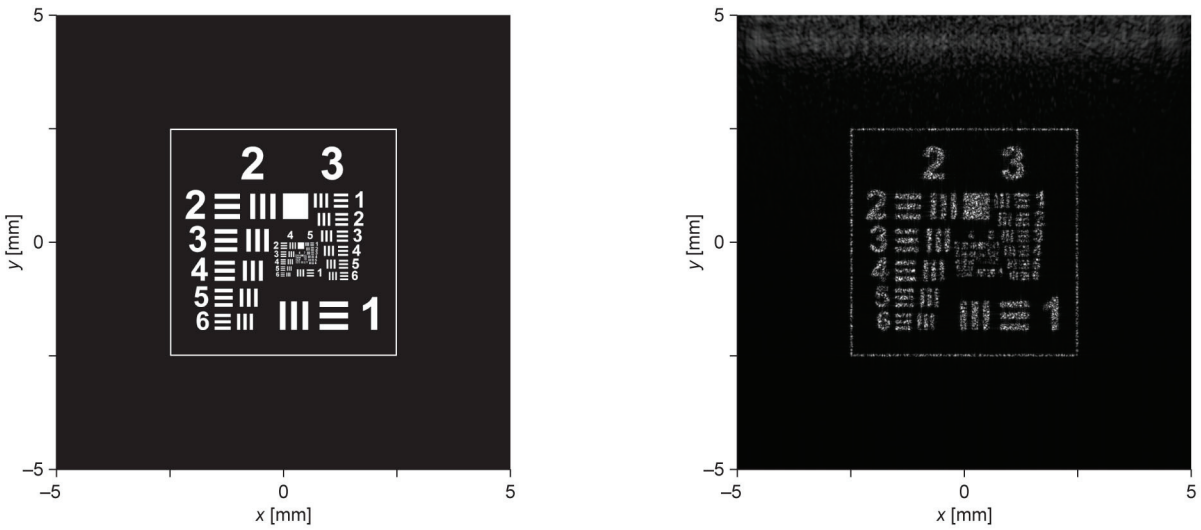

Figure 1. Left: Original image. Right: Reconstructed real image. The speckle noise is caused by a diffuse surface of the original image. Top of the image is lighter due to undiffracted reconstruction wave.

4. IMPLEMENTED FUNCTIONS AND DEMOS

The functionality of CGDH Tools is built around demos. It is thus the best to introduce its possibilities by introducing the demos. Please note that the demos are numbered in the CGDH Tools package. This numbering shows a recommended order to explore them.

Basic functions

Digital holography and Fourier optics uses variety of special complex functions, such as various chirps, impulse responses of free space (e.g. the Rayleigh-Sommerfeld convolution kernel, its simplified form, its Fresnel approximation), transfer functions of free space (e.g. the transfer function used in the angular spectrum decomposition method, its version without evanescent waves, its Fresnel approximation) and others. These functions must be sampled, thus it is necessary to understand their local frequency properties. The demo **basic_functions** shows these functions, shows the area where current sampling conditions are fulfilled, and shows an approximation of this region by a rectangle. Figure 2 shows the real part of the exact Rayleigh-Sommerfeld convolution kernel. It is clearly visible that the aliasing-free zone is bounded by curved lines (hyperbolas) and that the region is well approximated by a rectangle.

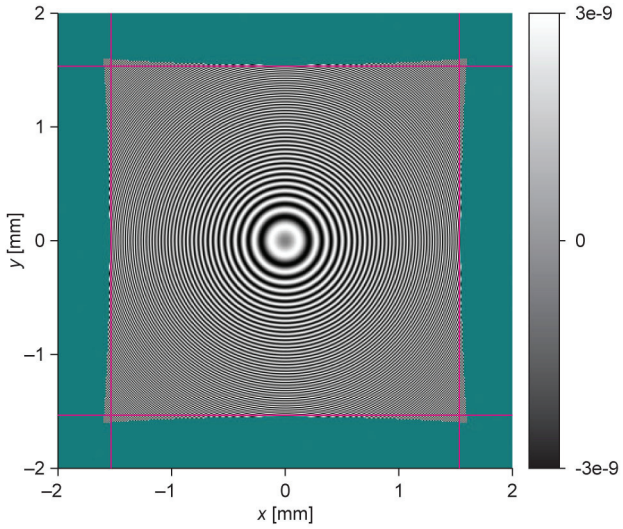

Figure 2. The real part of the exact Rayleigh-Sommerfeld convolution kernel. The green color depicts the area where the function cannot be properly sampled using given sampling distance.

Light propagation

One of the most important tasks in computer generated display holography is the calculation of coherent light propagation in free space. CGDH Tools offers basically three types of calculation of light propagation between parallel planes (source and target):

- **3fft**: convolution of the source phasor with a convolution kernel in the spatial domain calculated using three fast Fourier transforms. Supported convolution kernels are the exact Rayleigh-Sommerfeld kernel, the simplified Rayleigh-Sommerfeld kernel (without "minus 1/*r*" term) and their Fresnel approximation.
- **2fft**: convolution of the source phasor with a convolution kernel in the frequency domain, i.e. application of the transfer function. This calculation uses just two fast Fourier transforms. Supported transfer functions are the exact one from the angular spectrum decomposition method, its alternative without evanescent waves support and its Fresnel transform. Note that 3fft methods are generally more precise for large propagation distances, while 2fft methods are generally preferred for small propagation distances.
- **1fft**: approximation of light propagation calculated using just one fast Fourier transform. Supported methods are the Fresnel approximation and the Fraunhofer approximation. Note that in the 3fft and the 2fft methods, the sampling distances used in the source and the target planes must be the same. In the 1fft method, sampling distance in the target plane is given by the propagation distance and the size of the source optical field.

Special functions used in the calculation (transfer functions etc.) can be further filtered in order to avoid their aliasing. Supported filtering modes are **localExact** and **localRect**, where high-frequency parts of the function are set to 0. The **localExact** filtering affects those samples where the local frequency of the function is too high. The **localRect** filtering approximates the aliasing-free zone by a rectangle and sets all samples outside of it to zero. In particular, the **localRect** method applied in the angular spectrum based propagation method is known as band-limited angular spectrum method[17].

All methods have been carefully implemented so that their results are directly comparable. Provided demos **compare3fft2fft**, **compare3fft1fft** and **compare1fft1fft** allow to calculate light propagation using two different methods and compare the results. For example, Figure 3 shows the propagation of light (λ = 500 nm) diffracted by a rectangular sine grating (f = 20 mm^{-1}) of size 1 mm × 1 mm to a distance 200 mm by 3fft and 1fft methods using the Fresnel approximation. Note that the images are mostly the same including absolute values.

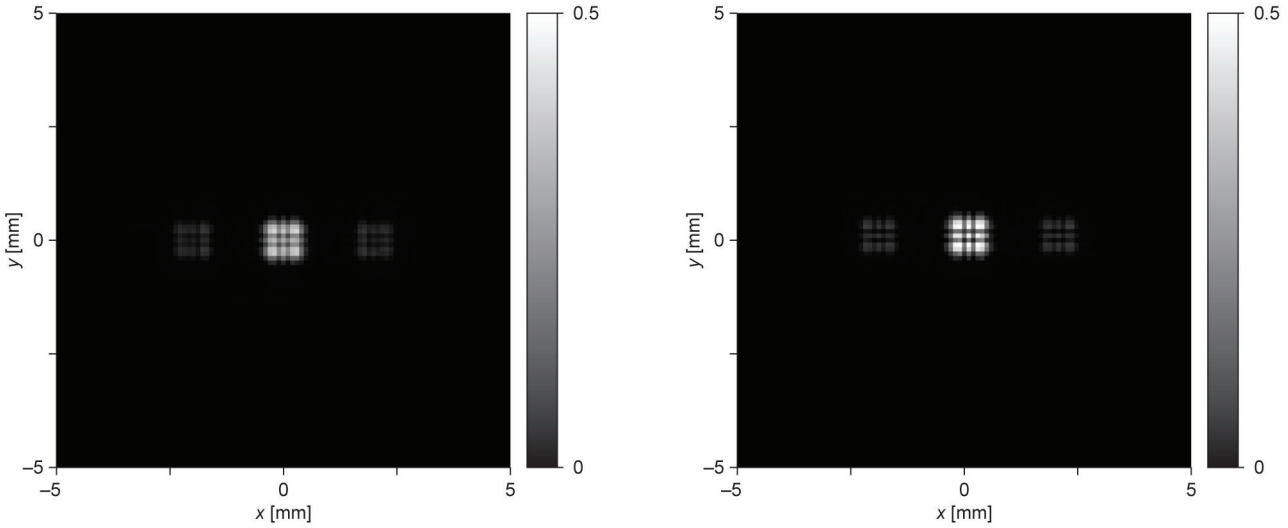

Figure 3. Left: calculation using 3fft method (Fresnel approximation).
Right: calculation using 1fft method (Fresnel approximation).

Hologram recording

Demos **hologram_real_image** and **hologram_virtual_image** show simple recording of a hologram of an image and its reconstruction. In the first case, a real image is observed on the screen, while in the second case a simple simulated camera photographs a virtual image behind the hologram. The demos further show various hologram recording setups (on-axis, off-axis, lensless Fourier) and various hologram coding methods. Supported hologram coding methods are *classical* (intensity of interference pattern created by a reference wave R and an object wave O), *bipolar*[18] (real part of OR*, where * is a complex conjugate), *shiftedBipolar* (bipolar with a constant offset to make the hologram transmittance positive), *kinoform*[19] and *complex* (both amplitude and phase modulation). Moreover, types *classical*, *bipolar* and *shiftedBipolar* can be transferred to phase modulation (bleached) to increase diffraction efficiency[1]. Hologram binarization is also supported.

Hologram of a layered scene

Demo **layered_hologram** finally shows a hologram of a 3-D scene. The scene is decomposed into a stack of parallel partially transparent layers. Light propagates from the farthest layer to the plane containing the following one. Light from the previous layer is combined with the current layer and is propagated to the next plane, and so on. The idea was proposed by Lohmann[20] and it is often used in other algorithms[21,22]. Example output of the demo is shown in Figure 4.

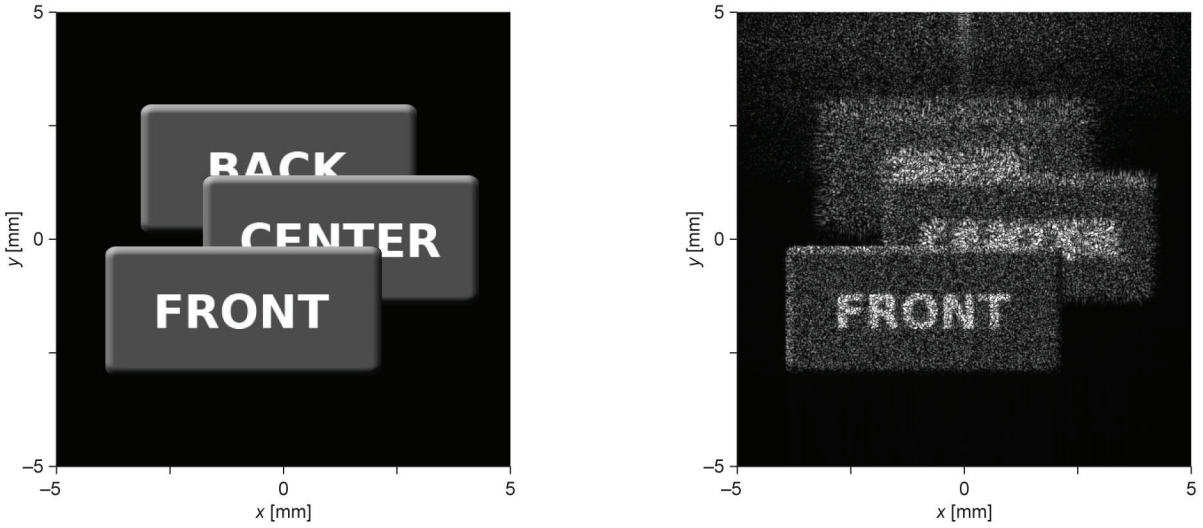

Figure 4. Left: Geometrical image of a layered scene, the layers are 25 mm apart.
Right: Real image reconstructed from a hologram of size 20 mm × 20 mm located 1475 mm from the front layer.

Physically based hologram of a point cloud

One of the most frequently used techniques for making a hologram of a 3-D scene replaces the scene with a point cloud[23]. Each point emits a spherical wave, the sum of all spherical waves makes the object wave. As the basic algorithm requires a vast number of calculations, various acceleration techniques are often employed. The demo **hologram_point_cloud** shows both the basic algorithm and the accelerated one based on the idea of a wavefront recording plane[24]. In this method, the object wave is first recorded on the wavefront recording plane (WRP) in the vicinity of the point cloud. If we assume each point emits light inside a tight cone only, each point affects just a small area on the WRP and thus requires a small number of calculations. The object wave from the WRP is then propagated to the plane of the hologram. Propagation can be calculated using the fast Fourier transform. Figure 5 shows reconstructions of a hologram calculated in this way.

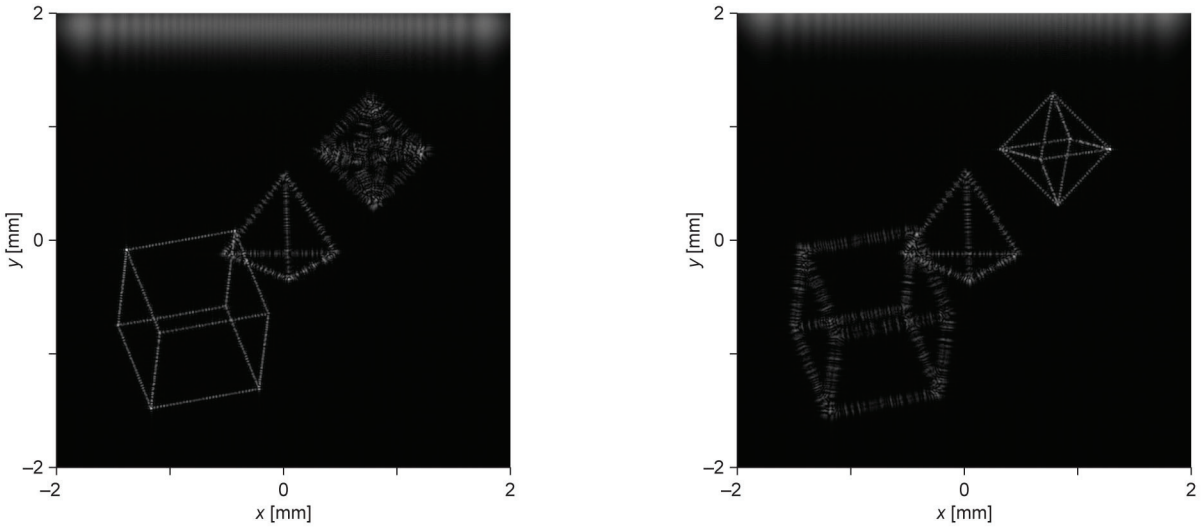

Figure 5. Left: Real image calculated by backpropagation 138 mm from the hologram.
Right: Reconstruction distance changed to 142 mm.

Stereograms

A completely different approach to making holograms of 3-D scenes splits a hologram area into small rectangles. Each rectangle contains a hologram of an image of a scene viewed from the rectangle center. These rectangles are often called subholograms. Note that a subhologram calculation can be fast as a hologram of an image can be calculated using single fast Fourier transform. Note that if the image is large and is located far from the hologram (which is usually the case), a spherical wave emitted by a point of the scene is approximated by a plane wave. It means that circular fringes caused by a spherical wave are approximated by straight segments in subholograms.

The basic algorithm[25] treats each subhologram independently, which causes loss of coherence between subholograms, see Figure 6. It means that fringes are not continuous at subhologram borders. Moreover, if the subhologram is calculated by the fast Fourier transform, plane wave directions are quantized which further degrades the hologram quality. The demo **stereogram** shows the basic algorithm and several techniques that improve it – the phase-added stereogram (PAS)[26], compensated phase-added stereogram (CPAS)[27], accurate phase-added stereogram (APAS)[28] and accurate compensated phase-added stereogram (ACPAS)[29]. In addition, the methods are implemented both utilizing and not utilizing the fast Fourier transform (FFT) in order to observe artifacts connected with quantization of plane wave direction. Last but not least, implementations without FFT allow better insight into the geometrical nature of the calculation.

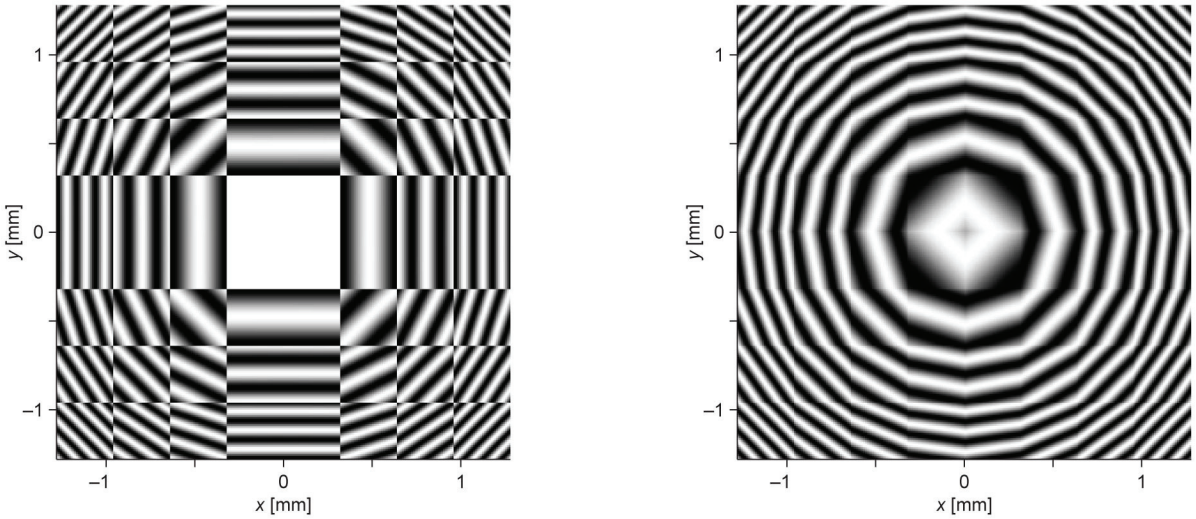

Figure 6. Left: Real part of the spherical wave emitted by a point 250 mm from the hologram plane as calculated by the basic stereogram method.
Right: The same spherical wave calculated using ACPAS method. Both images were generated using FFT.

Standalone demos

The demos in CGDH Tools and associated functions should be perfectly clear to any interested reader with basic understanding of holography. However, they could be difficult to understand for a complete beginner as the demos call CGDH Tools core functions, which further call other CGDH Tools functions, and so on. Thus, three additional demos are included that are completely self-contained: such a demo is composed of a single script that does everything from scratch.

The first demo calculates a hologram of a point cloud and reconstructs a real image. The implementation does not use any optimization, which means it is very easy to rewrite it in any imperative programming language such as C. The second demo is basically the same as the first one, but it employs matrix operations instead of loops because matrix operations are 1. highly optimized in Octave/MATLAB/Scilab, 2. code is then much easier to read. The third demo modifies the second one by taking a photograph of a virtual image rather than reconstructing a real image.

In fact, these demos are virtually the same as the scripts presented at the tutorial[12] for the audience without prior knowledge of wave optics or holography.

5. CONCLUSION AND FUTURE WORK

The current implementation of CGDH Tools contains scripts developed in Octave and fully compatible with MATLAB. A modification working in Scilab is also provided. Rewrite to other high-level languages such as Julia is not planned currently.

At the time of writing this document, CGDH Tools contained three basic methods of light propagation calculation: **3fft** (convolution calculated in the spatial domain), **2fft** (convolution calculated in the frequency domain) and **1fft** (calculation using single fast Fourier transform only). The most important missing feature is the automatic method selection; currently, the user has to decide which method to use. Another important missing feature is the optical field rotation, i.e. support for light propagation between nonparallel planes. Somewhat missing features are other propagation algorithms, most notably the double Fresnel propagation.

CGDH Tools contains all basic hologram encoding methods, i.e. the classical interference based, bipolar intensity, kinoform and full complex modulation. A somewhat missing feature is a demonstration of hologram creation without full complex representations of the object and the reference waves. Only basic hologram binarization is provided. The most important missing feature is the hologram optimization such as using an iterative Fourier transform algorithm.

Current demos show a calculation of a hologram of a point cloud, of a scene decomposed to a stack of parallel layers, and a stereogram of a point cloud. A point cloud hologram is calculated either directly or using a single wavefront recording plane. Hidden surface elimination is not implemented, as well as using multiple wavefront recording planes. Another missing feature is a demonstration of look-up tables to accelerate the calculation. Scene decomposition to a stack of layers should be enhanced to a stack of nonparallel layers, which leads to a polygon-based hologram calculation. The most important missing feature in stereogram calculation is using other primitives than points, such as triangle meshes. There is no demonstration of analytic methods, i.e. methods that utilize an analytic representation of a wave produced by a triangle, a rectangle or other non-trivial geometrical primitive.

Any improvements of CGDH Tools are welcome. CGDH Tools can be downloaded at http://holo.zcu.cz or at https://gitlab.com/petr.lobaz/CGDH-Tools.

ACKNOWLEDGEMENTS

This work was supported by University spec. research – 1311, and by the University specific research project SGS-2016-013 Advanced Graphical and Computing Systems.

REFERENCES

[1] Goodman, J. W., [Introduction to Fourier Optics (3 ed.)], Roberts & Company, Englewood (2004).
[2] Poon, T.-C. and Liu, J.-P., [Introduction To Modern Digital Holography: With MATLAB], Cambridge University Press, New York (2014).
[3] Vijayakumar, A, Bhattacharya, S., [Design and Fabrication of Diffractive Optical Elements with MATLAB], SP IE Press, Bellingham (2017).
[4] Voelz, D., [Computational Fourier Optics: A MATLAB Tutorial], SPIE Press, Bellingham (2011).
[5] Schmidt, J. D., [Numerical Simulation of Optical Wave Propagation: With examples in MATLAB], SPIE, Bellingham (2010).
[6] Picart, P., Li, J., [Digital holography], ISTE, London (2012).
[7] Poon, T.-C., Banerjee, P. P., [Contemporary Optical Image Processing with MATLAB], Elsevier Science, Kidlington (2001).
[8] Shimobaba, T., Weng, J., Sakurai, T., Okada, N., Nishitsuji, T., Takada, N., Shiraki, A., Masuda, N., Ito, T., "Computational wave optics library for C++: CWO++ library", Comput. Phys. Commun. 183(5), 1124–1138 (2012).
[9] Matsushima, K., "A Took-kit for Simulation in Wave-Optics: WaveField Tools", Optics & Photonics Japan, 9aC2 (2010).

[10] Object-based Media Group at MIT Media Lab, [Holovideo-mit (Software)], https://github.com/itsermo/holovideo-mit/ (2017).
[11] Müller, P., [nrefocus: Python algorithms for numerical focusing (version 0.1.8) (Software)]. https://pypi.python.org/pypi/nrefocus/ (2017).
[12] Lobaz, P., "Computer generated display holography", EG 2017 – Tutorials (2017).
[13] MathWorks, [MATLAB R2017b (software)], https://www.mathworks.com/products/matlab.html (2017).
[14] Eaton, J. W., Bateman, D., Hauberg, S. and Wehbring, R., [GNU Octave version 4.2.1 manual: a high-level interactive language for numerical computations], https://www.gnu.org/software/octave/doc/v4.2.1 (2017).
[15] Scilab Enterprises, [Scilab: Free and Open Source software for numerical computation (Windows, Version 6.0.0) (Software)], http://www.scilab.org (2017).
[16] Poynton, C., [Digital Video and HD: Algorithms and Interfaces (2 ed.)], Morgan Kaufmann, San Francisco, 315–334 (2012).
[17] Matsushima, K. and Shimobaba, T., "Band-limited angular spectrum method for numerical simulation of free-space propagation in far and near fields", Opt. Express 17(22), 19662–19673 (2009).
[18] Lucente, M., [Diffraction-Specific Fringe Computation For Electro-Holography (Ph.D. thesis)], Massachusetts Institute of Technology, Cambridge (1994).
[19] Lesem, L. B., Hirsch, P. M., Jordan, J. A., "The Kinoform: A New Wavefront Reconstruction Device", IBM J. Res. Dev. 13(2), 150–155 (1969).
[20] Lohmann, A. W., "Three-dimensional properties of wave-fields", Optik 51(2), 105–117 (1978).
[21] Symeonidou, A., Blinder, D., Munteanu, A., Schelkens, P., "Computer-generated holograms by multiple wavefront recording plane method with occlusion culling", Opt. Express 23(8), 22149–22161 (2015).
[22] Zhang, H., Cao, L., Jin, G., "Computer-generated hologram with occlusion effect using layer-based processing", Appl. Opt. 56(13), F138-F143 (2017).
[23] Lesem, L. B., Hirsch, P. M., Jordan, Jr., J. A., "Holographic Display of Digital Images", Proc. AFIPS '67 (Fall) 41–47 (1967).
[24] Shimobaba, T., Masuda, N., Ito, T., "Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane", Opt. Lett. 34(10), 3133–3135 (2009).
[25] Yatagai, T., "Stereoscopic approach to 3-D display using computer-generated holograms", Appl. Opt. 15(11), 2722–2729 (1976).
[26] Yamaguchi, M., Hoshino, H., Honda, T., Ohyama, N., "Phase-added stereogram: calculation of hologram using computer graphics technique", Proc. SPIE 1914, 25–31 (1993).
[27] Kang, H., Fujii, T., Yamaguchi, T., Yoshikawa, H., "Compensated phase-added stereogram for real-time holographic display", Opt. Eng. 46(9), 095802-095802-11 (2007).
[28] Kang, H., Yamaguchi, T., Yoshikawa, H., "Accurate phase-added stereogram to improve the coherent stereogram", Appl. Opt. 47(19), D44–D54 (2008).
[29] Kang, H., Yaraş, F., Onural, L., Yoshikawa, H., "Real-Time Fringe Pattern Generation with High Quality", Proc. Digital Holography and Three-Dimensional Imaging, paper DTuB7 (2009).