 estructure and Design of HoloGen

June 19, 2020

Peter J. Christopher and Timothy D. Wilkinson

Centre of Molecular Materials, Photonics and Electronics, University of Cambridge, UK

Abstract

Increasing popularity of augmented and mixed reality systems has seen a similar increase of interest in 2D and 3D computer generated holography (CGH). Unlike stereoscopic approaches, CGH can fully represent a light field including depth of focus, accommodation and vergence. Along with existing telecommunications, imaging, projection, lithography, beam shaping and optical tweezing applications, CGH is an exciting technique applicable to a wide array of photonic problems including full 3D representation.

Traditionally, the primary roadblock to acceptance has been the significant numerical processing required to generate holograms requiring both significant expertise and significant computational power.

This article discusses the structure and design of HoloGen. HoloGen is an MIT licensed application that may be used to generate holograms using a wide array of algorithms without expert guidance. HoloGen uses a Cuda C and C++ backend with a C# and Windows Presentation Framework graphical user interface. The article begins by introducing HoloGen before providing an in-depth discussion of its design and structure. Particular focus is given to the communication, data transfer and algorithmic aspects.

Keywords Computer Generated Holography · HoloGen · Cuda · Fourier · Fresnel

1 Introduction

This article presents an overview of the structure of the HoloGen application as well as some of the design decisions made during development. While the majority of HoloGen is simple contextually and requires little discussion, a few areas stand out as meriting explanation for future developers. Key features for further discussion include: The parameter and command tree in Section 3; The native algorithm interface in Section 4; The serialisation architecture in Section 5 and the user interface construction in Section 6. Section 7 discusses the role of all the application libraries and their connections as well as any key classes not covered elsewhere. Finally Section 8 provides some areas of future expansion and the infrastructure put in place to handle them.

2 HoloGen

The HoloGen Suite is an application suite built on top of a custom parameter framework in C# and WPF. The package is open-source under the MIT license with sub libraries each having their own license. The source is available online. Version 2.2.1.17177 of HoloGen runs to 76,000 lines of code with dependencies on 25 external libraries.

The HoloGen application is built on a MVVMA architecture. This is a standard Model-View-ViewModel (MVVM) framework common in C# Windows Presentation Framework (WPF) applications with an additional algorithms level written in a more traditional procedural/functional style on top of an Nvidia Cuda architecture interfaced in C++ [1]. This translates into a layered application structure as shown in Figure 1. This shows the three application levels: the user interface level; application level and algorithm level. These all depend only on levels beneath them and have their own independent imported libraries. A more detailed breakdown is shown in Figures 2, 3 and 4.

The cuFFT library from NVidia is used to perform the FFT element of the algorithms due to its high performance [2]. This is built on top of a Cuda framework with Thrust wrapper. Previous researchers have also used the FFTW library [3, 4], OpenCV [5], OpenMP [6], the Computational Wave Optics (CWO) [7, 8, 9] and Intel Math Kernel (MKL) [10] libraries as well as custom implementations in Matlab [11, 12, 13, 14, 15, 16], C/C++ [17, 6, 10, 3, 4, 7, 8, 9] and Python.

HoloGen depends on a number of third-party libraries [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]. Most are licensed under the MIT license [36, 37, 38, 39, 40, 41, 42, 43, 44, 30, 45] with several licensed under the Lesser General Public License [46, 47, 48, 49] or the Apache License 2.0 [50, 51, 52, 34] and two under the three-clause BSD license [53, 54, 55]. These are all highly permissive licenses. Xamarin.Forms, C# and WPF come with appropriate usage licenses as part of Cambridge University Visual Studio package while Cuda comes with an appropriate license and EULA [56, 57].
HoloGen includes a number of novel elements. A reflection based parameter framework allows for persistent parameter models with limited code reuse. A dynamic module system interfaces with Cuda implemented algorithms allowing for real-time Cuda compilation on host machines improving performance and utilising more advanced features of newer graphics cards. Three dimensional visualisation techniques can be used while viewing the generated hologram statistics. Fourier transform functionality is incorporated directly into the viewer. A tabular batch processing framework allows for multiple operations to be scheduled for background operation. Advanced tabulation allows for comparison of different holograms. New image and file types are introduced to handle the additional information available and all results are tagged with parameter metadata used to ensure traceability. Provisional translations into French, Spanish, German and Chinese are available as well as complete help documentation. An included Chromium browser allows for integrated reporting features.

## 3 Parameter and Command Hierarchy

HoloGen uses a custom reflection based parameter and command system. This is in contrast to the XML parameter sheet systems widely in use. Instead of the parameter types and interactions being defined in parameter sheets which are parsed at runtime, the parameter system is coded into the C# directly. This significantly reduces the runtime overhead as well as improves the error checking available at compile time. The downside is an increased architecture exposure of the parameter hierarchy. This structure is contained in the *HoloGenHierarchy* library. The decision to use the reflection based system was made to enable a side project of the author.

### 3.1 Class Inheritance Hierarchy

Two distinct areas of the *HoloGenHierarchy* library code stand out: the parameter/command types (e.g. numerical options, menu commands, etc) and the tree of elements containing them (e.g. pages, menus, etc).

The parameter types are defined using the following key classes shown in Figure 5:
• INode - Base interface for all leaf nodes in the parameter hierarchy.
• ICommand - Base command interface extended by all parameter types. Extends INode.
• IOption - Base parameter interface extended by all parameter types. Extends INode.
• Command - Abstract base command class extended by all command types. Distinct from ICommand which it implements due to C# not handling generic template references.
• Option - Abstract base parameter class extended by all parameter types. Distinct from IOption which it implements due to C# not handling generic template references. A template system allows for generic manipulation of wrapped values without exposing the internals of the class to extending objects.
• NumericOption - Abstract base parameter class extended by all numeric parameter types. Extends Option.
• IntegerOption - Integral numeric parameter type. Extends NumericOption.
• DoubleOption - Floating point numeric parameter type. Extends NumericOption.
• ILargeOption - Interface that flags to the display that implementing options require extra space on the UI. Extends IOption.
• TextOption - Text based parameter type. Extends Option.
• LargeTextOption - Larger version of TextOption. Implements ILargeOption.
• PathOption - Alternative to TextOption that handles file paths. Implements ILargeOption.
• ISelectOption - Base parameter interface extended by all selection based parameter types. Extends IOption.
• SelectOption - Abstract base parameter class extended by all selection based parameter types. Distinct from ISelectOption which it implements due to C# not handling generic template references. A template system allows for generic manipulation of wrapped values without exposing the internals of the class to extending objects. Contains a PossibilityCollection of Possibilities that can be selected as the option value. Any options owned by the selected Possibility are injected into the owning HierarchyFolder.
• ListOption - Base class for parameters representing lists of values. Extends Option. Implements ILargeOption.
Figure 3: HoloGen application level. c.f. Figure 1.

- **PathListOption** - Base class for parameters representing lists of files. Extends ListOption. Implements ILargeOption.
- **BooleanOption** - Boolean value based parameter type. Extends Option.
- **BooleanOptionWithChildren** - Boolean value based parameter type. Extends BooleanOption. When set to true, any child options are injected into the owning HierarchyFolder in a manner similar to SelectList.

The tree of elements containing the parameters or commands are shown in Figure 7:

- **ReflectiveChildrenElement** - Key class representing any non-leaf node in the parameter/command tree. When extended, this class uses the C# reflection system to find all public parameters of the same type as the specified template type and uses them to populate a list of children of that type. This allows extending classes to declare member elements without having to handle their manipulation or access.
- **ChangingChildrenElement** - Extends ReflectiveChildrenElement to handle a non-leaf node that has a changing set of children with appropriate notifications.
- **ChangingChildrenElement** - Extends ChangingChildrenElement to handle a non-leaf node that has a set of children which can be searched.
- **IHierarchyElement** - Abstract interface for all non-leaf nodes in the parameter tree.
- **HierarchyElement** - Abstract base class for all non-leaf nodes in the parameter/command tree. Distinct from IHierarchyElement which it implements due to C# not handling generic template references.
- **HierarchyRoot** - Implementation of HierarchyElement that represents the root node of a parameter/command tree. Equivalent to the tab or menu pop-out level within HoloGen.
- **HierarchyPage** - Implementation of HierarchyElement that represents a node of a parameter/command tree. Equivalent to the page or menu level within HoloGen.
- **HierarchyFolder** - Implementation of HierarchyElement that represents a node of a parameter/command tree. Equivalent to the folder or sub-menu level within HoloGen.
**OptionCollection** - Implementation of `HierarchyElement` that represents a set of `Options` within a specialisation `Option` such as `BooleanOptionWithChildren` or `SelectOption`.

**PossibilityCollection** - Implementation of `HierarchyElement` that represents a set of `Possibilities` within a `SelectOption`.

**Possibility** - Represents a possibility state for a `SelectOption`.

**HierarchyVersion** - Represents a version number for a parameter/command hierarchy.

**HierarchySaveable** - Represents any class that can be saved using the JSON serialisation.

In addition a number of function interfaces are used to mark exhibited behaviours at different levels of the hierarchy.

- **ICanEnable** - Implemented by any class/interface that exhibits enable/disable behaviours.
- **ICanError** - Implemented by any class/interface that can be in an error state.
- **ICanImportFromString** - Implemented by any class/interface that allows for deserialisation from a string object.
- **ICanExportToString** - Implemented by any class/interface that allows for serialisation to a string object.
- **ICanFlatten** - Implemented by any `HierarchyElement` that can flatten its internal tree structure.
- **IHasName** - Implemented by any class/interface that has a name property.
- **IHasToolTip** - Implemented by any class/interface that has a tool tip property.
- **IHasWatermark** - Implemented by any class/interface that has a watermark property.
- **IHasBindingPath** - Implemented by any `HierarchyElement`, Command or `Option` that the interface can be bound to given a link to the base of the parameter or command tree.
- **ICanSearch** - Implemented by any class/interface that can be searched.
- **ICanReset** - Implemented by any class/interface that can be reset.

---

**Figure 4**: HoloGen algorithms level. c.f. Figure 1.
Figure 5: HoloGen parameter types inheritance hierarchy

- ICanRecursivelyEnable - Implemented by any class/interface that can set its own enabled state and that of its children. Extends ICanEnable.
- INotifyChanged - Implemented by any Option that notifies when the contained value changes.

Figure 6: HoloGen algorithm interface
3.2 Application

HoloGen defines a number of parameter and command hierarchies including those in the HoloGenOptions, HoloGenImageOptions, HoloGenProcessOptions, HoloGenBatchOptions, and SLMControlOptions libraries. These follow a standard structure with the following classes.

- A root element extending HierarchyRoot that provides a name and tool tip as well as containing public properties for all of the HierarchyPage objects within. Example: OptionsRoot in the HoloGenOptions library.
- Multiple elements extending HierarchyPage that provides a name, tool tip and icon as well as containing public properties for all of the HierarchyFolder objects within. Example: ProjectorPage in the HoloGenOptions library.
- Multiple elements extending HierarchyFolder that provides a name and tool tip as well as containing public properties for all of the Option objects within. Example: HologramFolder in the HoloGenOptions library.
- Multiple elements extending Option or its subclasses that provide the name, tool tip, defaults and limits for the option. Example: SLMResolutionX in the HoloGenOptions library.
  - Classes extending SelectOption contain a link to an extension of a PossibilityCollection object. Example: SLMTypeOption in the HoloGenOptions library.
  - Classes extending Possibility or BooleanOptionWithChildren are also able to contain public properties that will only be editable when selected. Example: MultiAmpSLM in the HoloGenOptions library.
- Elements extending PossibilityCollection with public properties for each of the allowable Possibilities of the SelectOption. Example: SLMPossibilities in the HoloGenOptions library.
- Elements extending Possibility with public properties for each of the allowable Possibilities of the SelectOption. Example: SLMPossibility in the HoloGenOptions library.

While the reflection based architecture requires an initial investment of time and effort to come to grips with, it has proved highly time efficient in practice during development on HoloGen. The system presents an alternative to the XML systems commonly used and should be considered for wider use.
4 Algorithm Interface

A four step process is required in order to pass data from the C# HoloGen application and user interface to the Cuda C/C++ underlying it as shown in Figure 6.

The top level is the C# application level where an AlgorithmController unpacks the Option parameter hierarchy and passes it to the Managed C++ level below. All Managed C++ wrapper classes have the suffix "*Man". Managed C++ libraries are able to use both native data types as well as the .NET data types used by C#. After the data is copied between the two types, the native C++ layer is called. This level can be exposed in a dynamic or static library but is not able to link to many native Cuda headers such as CUFFT and Thrust. All native C++ wrapper classes have the suffix "*Wrap". These classes can, in turn, pass the data onto Cuda C/C++ compiled classes which are able to communicate with the graphics card. All Cuda C/C++ classes have the suffix "*Cuda".

For example, when passing a target image to a GS algorithm, the following steps occur. AlgorithmController calls SetTargetImage() on a GSA1gStandardMan object it owns. This uses its base Convert(GSA1gStandardMan) functions inherited from AlgorithmMan to transfer the data to native types. This is then passed to the GSA1gStandardWrap and GSA1gStandardCuda objects in turn. Once in the Cuda level, the parent functions in AlgorithmCuda handle transferring the data to the graphics card and maintaining handles on its location. The actual algorithm implementations can be found in the RunIterations() functions of their respective "*Cuda" classes.

While complex, this system performs as fast as direct dll import while allowing for integrated debugging.
Figure 9: HoloGen package layout.

5 Serialisation

Figure 8 shows the serialisation architecture for HoloGen. The command level in libraries such as HoloGenUIMenu interface to the JSON serialisation and deserialisation classes in the HoloGenSerial library. These then write to and read from the file system using the classes found in the HoloGenIO library.

6 User Interface

HoloGen follows a standard View-Model-ViewModel structure for its user interface. Each UI element or View is defined graphically in a *.xaml file, e.g. SetupTabView.xaml, with a C# companion file, e.g. SetupTabView.xaml.cs. The contained UI elements bind to the data contained and manipulated by the ViewModel, e.g. SetupTabViewModel.cs, which in turn holds handles to the internal data, e.g. OptionsRoot.cs.

7 Library Descriptions

Figure 9 shows the layout of libraries within HoloGen with reference to the three application levels shown in Figure 1. This section discusses the libraries used and any key classes not discussed in earlier sections.

7.1 User Interface Libraries

All user interface libraries are prefixed with "HoloGenUI" and are written in C# using WPF.

- **HoloGenUI** - The main entry point for the user application user interface. Provides top-level implementation TabHandlers for in HoloGenUIUtils. Key classes include:
  - MainWindowViewModel - ViewModel construct underlying the main application window. Keeps an observable collection of AbstractTab objects and is responsible for matching them with appropriate display classes. Implements ITabHandler for the TabHandlerFramework. MainWindowViewModel is responsible for the application wide user settings including notifications and file IO. Drag and drop behaviours are also overseen by MainWindowViewModel using the Dragablz library.
  - AbstractTab - Tab data interface that all other tabs can inherit. This allows for templating using TabDataTemplateSelector.
7.2 Application Libraries

All application libraries are prefixed with "HoloGen".

- **HoloGenController** - Defines classes that unwrap the `Options` hierarchy defined in `HoloGenOptions` for hologram generation and communicate it with the algorithms wrapped by `HoloGenAlgBaseMan`, `HoloGenAlgGSMan`, `HoloGenAlgSAMan`, `HoloGenDSMan` and `HoloGenOSPRMan`. Key class is `AlgorithmController`.
- **HoloGenProcess** - Defines data structures for defining a hologram generation process.

The display code is written in C# and WPF and lies on top of the ViewModels. There is a one to one correspondence between ViewModel classes and XAML parameter sheets on top.

- **HoloGenUIMenus** - Defines options and commands for the main application flyout menu. Built on-top of the `HoloGenHierarchy` library discussed in Section 3.
- **HoloGenUIHamburger** - Defines a series of View/ViewModel pairs that serve to unpack and display `HoloGenHierarchy` parameter hierarchies in a Microsoft Metro "Hamburger Menu" style. Individual editors are defined in `HoloGenUIOptionEditors`.
- **HoloGenUIOptionEditors** - Defines a series of View/ViewModel pairs that serve to display the different `Option` types discussed in Section 3. Primarily used as the leaf nodes for `HoloGenUIHamburger`.
- **HoloGenUIProcessMonitor** - Defines display classes for the algorithm monitoring tab. The charting is built on top of `LiveCharts` using `HoloGenUIGraph`.
- **HoloGenUIGraph** - Defines display classes for realtime charting. `HoloGenUIGraph` is built on top of `LiveCharts`.
- **HoloGenUIResources** - Defines translation strings for the application localisation.
- **HoloGenUIBatch** - Defines display classes for the batch processing tab. Defines a number of `AbstractColumnFactory` implementations for displaying different `Option` types. Key classes include:
  - `AbstractColumnFactory` - Interface for a series of factory objects - `BooleanColumnFactory`, `DefaultColumnFactory`, `DoubleColumnFactory`, `IntegerColumnFactory`, `PathColumnFactory`, `SelectColumnFactory`, and `TextColumnFactory` - used for generating WPF `DataGridColumns`.
  - `TemplateGenerator` - Helper class for `AbstractColumnFactory` implementations that uses a given delegate to create new instances for similar `Option` types.
- **HoloGenUIImageViewer** - Defines classes for viewing complex valued images in 2D.
- **HoloGenUIMask** - Extends the functionality from `HoloGenUIImageViewer` showing masked regions on a 2D complex valued image.
- **HoloGenUI3DViewer** - Extends the functionality from `HoloGenUIMask` showing complex valued images in 3D.
- **HoloGenUIUtils** - Defines low level utility classes for the HoloGen UI as well as abstract interfaces for services offered by higher level libraries. Key classes include:
  - `ITabHandler` and `TabHandlerFramework` - Define a service framework. The `MainWindowViewModel` extends `ITabHandler` and can register itself with `TabHandlerFramework`. Lower level class libraries can then call the `TabHandlerFramework` while remaining ignorant of the implementation.
  - `CreatorsThesis` - Display theme for the application built on top of `MahApps`.
  - `ICanExport`, `ICanExportBitmap`, `ICanExportExcel`, `ICanExportMat` and `ICanSave` - Interfaces that `ITabHandler` implementations can extend to declare their IO requirements.

7.2 Application Libraries

All application libraries are prefixed with "HoloGen".

- **HoloGenController** - Defines classes that unwrap the `Options` hierarchy defined in `HoloGenOptions` for hologram generation and communicate it with the algorithms wrapped by `HoloGenAlgBaseMan`, `HoloGenAlgGSMan`, `HoloGenAlgSAMan`, `HoloGenDSMan` and `HoloGenOSPRMan`. Key class is `AlgorithmController`.
- **HoloGenProcess** - Defines data structures for defining a hologram generation process.
• **HoloGenImage** - Defines classes related to complex valued images. Key classes include:
  
  - **ComplexImage** - Object that holds an image in Complex format as well as generation metadata and pre-cached values. Defines specialised JSON interface commands in order to preserve disk space.
  
  - **ImageCache** - Defines a cache for different Bitmap views on a ComplexImage object. Once a particular visualisation bitmap is generated for a particular image, the result is cached to reduce future load times. Uses a GenericCache from HoloGenUtils as the underlying implementation with the TransformType, ImageViewType, ColorScheme and ImageScaleType enums as the four access keys.

• **HoloGenHierarchy** - Defines the options and Command hierarchy discussed in Section 3.

• **HoloGenImports** - Classes or constructs that have been imported from other applications in a source code format.

• **HoloGenResources** - Defines translation strings for the application localisation.

• **HoloGenOptions** - Defines an Options hierarchy for the hologram generation algorithms. Built on-top of the HoloGen-Hierarchy library discussed in Section 3.

• **HoloGenProcessOptions** - Defines an Options hierarchy for manipulating the running process display in HoloGenUIProcessMonitor. Built on-top of the HoloGenHierarchy library discussed in Section 3.

• **HoloGenBatchOptions** - Defines an Options hierarchy for the batch processing hologram generation algorithms in HoloGenUIBatch. Built on-top of the HoloGenHierarchy library discussed in Section 3.

• **HoloGenSettings** - Defines an Options hierarchy for the application settings in HoloGenUI and HoloGenUIMenus. Built on-top of the HoloGenHierarchy library discussed in Section 3.

• **HoloGenImageOptions** - Defines an Options hierarchy for visualising complex valued images in HoloGenUIImageViewer. Built on-top of the HoloGenHierarchy library discussed in Section 3.

• **HoloGenMaskOptions** - Defines an Options hierarchy for visualising complex valued images with masking data in HoloGenUIMask. Built on-top of the HoloGenHierarchy library discussed in Section 3.

• **HoloGenSerial** - Handles the serialisation of HierarchySaveable data objects. Discussed further in Section 6.

• **HoloGenIO** - Handles the file input and output of serialised of HierarchySaveable data objects. Discussed further in Section 6.

• **HoloGenUtils** - Utility classes for HoloGen at the application level. Key classes include:
  
  - **ComplexImage** - Object that holds an image in Complex format as well as generation metadata and pre-cached values. Defines specialised JSON interface commands in order to preserve disk space.
  
  - **GenericCache** - Defines a cache for computationally expensive results using up to four different keys. Lambda functions are used for the computation to increase reusability. Provides the base implementation for ImageCache.

• **HoloGen** - Command line interface for the HoloGen application.

### 7.3 Algorithm Libraries

All algorithm libraries are prefixed with "HoloGenAlg" and receive the "Man" or "Cuda" suffix depending on whether they are written in Managed C++ or Cuda C/C++.

• **HoloGenAlg** - Command line interface for the HoloGen algorithms. Used for batch processing test data in a client-server configuration.

• **HoloGenAlgBaseMan** - Managed C++ wrapper for HoloGenAlgBaseCuda.

• **HoloGenAlgCommonMan** - Managed C++ wrapper for HoloGenAlgCommonCuda.

• **HoloGenAlgGSMMan** - Managed C++ wrapper for HoloGenAlgGSCuda.

• **HoloGenAlgSAMan** - Managed C++ wrapper for HoloGenAlgSACuda.

• **HoloGenAlgDSMan** - Managed C++ wrapper for HoloGenAlgDSACuda.

• **HoloGenAlgOSPRMan** - Managed C++ wrapper for HoloGenAlgOSPRCuda.

• **HoloGenAlgBaseCuda** - C++ and Cuda C base level algorithm definitions. Discussed further in Section 4. Key classes include:
  
  - **AlgorithmCuda** - Base level algorithm definition. Stores target, illumination and starting images as well as common algorithm parameters.
  
  - **FFTHandlerCuda** - Wraps the CUFFT FFT library with Thrust friendly functions.
  
  - **FFTUpdaterCuda** - Updates the replay field after the change of a single refraction field pixel.
  
  - **Normaliser** - Provides a fast vector normalisation feature.


7.4 Imported Libraries

HoloGen also uses a number of imported libraries. A brief description of their function is presented here.

- **MahApps** - Custom controls for WPF apps as well as a material design skin based on Microsoft’s Metro UI.
- **LiveCharts** - Customisable and bindable real-time charting library.
- **Dragablz** - Draggable tabs for WPF.
- **ControlzEx** - Custom controls for WPF apps.
- **HelixToolkit** - 3D viewer for WPF apps.
- **Xamarin.Forms** - Mobile/tablet compatibility.
- **MaterialDesign** - Material Design compatible skin for WPF apps.
- **MaterialSkin** - Alternative Material Design compatible skin for WPF apps.
- **GongSolutions.WPF.DragDrop** - WPF drag and drop capability.
- **FastMember.Signed** - Fast reflection for .NET.
- **DocumentFormat.OpenXML** - Microsoft Office file format interoperability.
- **ExcelNumberFormat** - Advanced Excel number formatting.
- **SharpDX** - WPF DirectX compatibility.
- **ClosedXML** - Microsoft Excel integration.
- **CefSharp** - WPF compatibly wrapper for the Chromium browser.
- **Cuda** - Programmable interface to NVidia graphics cards.
- **PdfiumViewer** - PDF file viewer.
- **Xceed** - DataGrid controls for WPF apps.
- **NHotKey** - Global hot keys for WPF apps.
- **Newtonsoft.Json** - JSON serialisation for .NET languages.
- **MathNet** - Mathematics package for .NET languages.
- **AForge** - Mathematics package for .NET languages.
- **Accord** - Mathematics package for .NET languages.
- **NUnit** - unit test framework for .NET languages.

8 Further Research and Conclusion

This article has presented a brief summary of the structure and design of HoloGen, an open-source hologram generation package. While HoloGen is feature rich, it is by no means a complete package. In particular, the number of algorithms available is limited and additional algorithms should be implemented.

Development of HoloGen is ongoing and the authors welcome any feedback, advice and assistance offered during this process.
REFERENCES

[1] Cuda, www.developer.nvidia.com/cuda-zone (2018-01-01).
URL https://developer.nvidia.com/cuda-zone/

[2] P. Steinbach, M. Werner, gearshift– the fft benchmark suite for heterogeneous platforms, in: International Supercomputing Conference, Springer, 2017, pp. 199–216.

[3] T. Shimobaba, N. Masuda, T. Ito, Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane, Optics Letters 34 (20) (2009) 3133. doi:10.1364/ol.34.003133.

[4] T. Shimobaba, J. Weng, T. Sakurai, N. Okada, T. Nishitsuji, N. Takada, A. Shiraki, N. Masuda, T. Ito, Computational wave optics library for C++: CWO++ library, Computer Physics Communications 183 (5) (2012) 1124–1138. doi:10.1016/j.cpc.2011.12.027.
URL http://dx.doi.org/10.1016/j.cpc.2011.12.027

[5] Y. Wang, X. Sang, Z. Chen, H. Li, L. Zhao, Real-time photorealistic computer-generated holograms based on backward ray tracing and wavefront recording planes, Optics Communications 429 (July) (2018) 12–17. doi:10.1016/j.optcom.2018.07.068.
URL https://doi.org/10.1016/j.optcom.2018.07.068

[6] N. Takada, T. Shimobaba, H. Nakayama, A. Shiraki, N. Okada, M. Oikawa, N. Masuda, T. Ito, Fast high-resolution computer-generated hologram computation using multiple graphics processing unit cluster system, Applied Optics 51 (30) (2012) 7303. doi:10.1364/ao.51.007303.

[7] T. Shimobaba, T. Ito, Fast generation of computer-generated holograms using wavelet shrinkage, Optics Express 25 (1) (2017) 77. doi:10.1016/j.optexp.2016.05.077.

[8] T. Shimobaba, K. Matsushima, T. Takahashi, Y. Nagahama, S. Hasegawa, M. Sano, R. Hirayama, T. Kakue, T. Ito, Fast, large-scale hologram calculation in wavelet domain, Optics Communications 412 (August 2017) 80–84. doi:10.1016/j.optcom.2017.11.066.
URL https://doi.org/10.1016/j.optcom.2017.11.066

[9] T. Shimobaba, S. Yamada, T. Kakue, T. Ito, Fast hologram calculation using wavelet transform, in: Proc. SPIE 10964, Tenth International Conference on Information Optics and Photonics, Vol. 10964, SPIE, 2018, p. 116. doi:10.1117/12.2505810.

[10] S.-C. Kim, E.-S. Kim, Effective generation of digital holograms of three-dimensional objects using a novel look-up table method, Applied Optics 47 (19) (2008) D55. doi:10.1364/ao.47.000d55.

[11] S.-C. Kim, E.-S. Kim, Fast computation of hologram patterns of a 3D object using run-length encoding and novel look-up table methods, Applied Optics 48 (6) (2009) 1030. doi:10.1364/ao.48.001030.

[12] J. Jia, Y. Wang, J. Liu, X. Li, Y. Pan, Z. Sun, B. Zhang, Q. Zhao, W. Jiang, Reducing the memory usage for effective computer-generated hologram calculation using compressed look-up table in full-color holographic display, Applied Optics 52 (7) (2013) 1404. doi:10.1364/ao.52.001404.

[13] G. Makey, M. S. El-Daher, K. Al-Shufl, Accelerating the calculations of binary detour phase method by integrating both CUDA and Matlab programming for GPU’s parallel computations, Optik 124 (22) (2013) 5486–5488. doi:10.1016/j.ijleo.2013.03.170.
URL http://dx.doi.org/10.1016/j.ijleo.2013.03.170

[14] P. Memmolo, L. Miccio, F. Merola, A. Paciello, V. Embrione, S. Fusco, P. Ferraro, P. Antonio Netti, Investigation on specific solutions of Gerchberg-Saxton algorithm, Optics and Lasers in Engineering 52 (1) (2014) 206–211. doi:10.1016/j.optlaseng.2013.06.008.
URL http://dx.doi.org/10.1016/j.optlaseng.2013.06.008

[15] Z. Wang, G. Lv, Q. Feng, A. Wang, H. Ming, Highly efficient calculation method for computer-generated holographic stereogram using a lookup table, Applied Optics 58 (5) (2019) A41–A47. doi:10.1364/ao.58.000a41.

[16] J. Carpenter, T. D. Wilkinson, Graphics processing unit–accelerated holography by simulated annealing, Optical Engineering 49 (9) (2010) 095801. doi:10.1117/1.3484950.

[17] MaterialSkin Library, www.github.com/IgnaceMaes/MaterialSkin (2018-01-01).
URL https://github.com/IgnaceMaes/MaterialSkin

[18] LiveCharts Library, www.lvcharts.net (2018-01-01).
URL https://lvcharts.net/

[19] SharpDX Library, www.sharpdx.org (2018-01-01).
URL http://sharpdx.org/
[21] Helix Toolkit Library, www.github.com/helix-toolkit/helix-toolkit (2018-01-01).
URL https://github.com/helix-toolkit/helix-toolkit
[22] ControlzEx Library, www.github.com/ControlzEx/ControlzEx (2018-01-01).
URL https://github.com/ControlzEx/ControlzEx
[23] NHotkey Library, www.github.com/thomaslevesque/NHotkey (2018-01-01).
URL https://github.com/thomaslevesque/NHotkey
[24] Math.NET Library, www.mathdotnet.com (2018-01-01).
URL https://www.mathdotnet.com/
[25] Newtonsoft.Json Library, www.newtonsoft.com/json (2018-01-01).
URL https://www.newtonsoft.com/json
[26] NUnit Library, www.nunit.org (2018-01-01).
URL http://nunit.org/
[27] AForge.NET Library, www.aforgenet.com/framework (2018-01-01).
URL http://www.aforgenet.com/framework/
[28] ManagedCuda Library, www.kunzmi.github.io/managedCuda (2018-01-01).
URL https://kunzmi.github.io/managedCuda/
[29] Accord.NET Library, accord-framework.net/ (2018-01-01).
URL http://accord-framework.net/
[30] csmatio Library, sourceforge.net/projects/csmatio/ (2018-01-01).
URL https://sourceforge.net/projects/csmatio/
[31] PdfiumViewer Library, github.com/pvginkel/PdfiumViewer (2018-01-01).
URL https://github.com/pvginkel/PdfiumViewer
[32] CefSharp Library, github.com/cefsharp/CefSharp (2018-01-01).
URL https://github.com/cefsharp/CefSharp
[33] ClosedXML Library, github.com/closedxml/closedxml (2018-01-01).
URL https://github.com/closedxml/closedxml
[34] MoreLinq Library, github.com/morelinq/MoreLINQ (2018-01-01).
URL https://github.com/morelinq/MoreLINQ
[35] GongSolutions.WPF.DragDrop Library, github.com/punker76/gong-wpf-dragdrop (2018-01-01).
URL https://github.com/punker76/gong-wpf-dragdrop
[36] MaterialSkin License, www.github.com/IgnaceMaes/MaterialSkin/blob/master/LICENSE (2018-01-01).
URL https://github.com/IgnaceMaes/MaterialSkin/blob/master/LICENSE
[37] LiveCharts License, www.github.com/beto-rodriguez/Live-Charts/blob/master/LICENSE.TXT (2018-01-01).
URL https://github.com/beto-rodriguez/Live-Charts/blob/master/LICENSE.TXT
[38] SharpDX License, www.github.com/sharpdx/SharpDX/blob/master/LICENSE.txt (2018-01-01).
URL https://github.com/sharpdx/SharpDX/blob/master/LICENSE.txt
[39] Helix Toolkit License, www.github.com/helix-toolkit/helix-toolkit/blob/develop/LICENSE (2018-01-01).
URL https://github.com/helix-toolkit/helix-toolkit/blob/develop/LICENSE
[40] ControlzEx License, www.github.com/ControlzEx/ControlzEx/blob/develop/LICENSE (2018-01-01).
URL https://github.com/ControlzEx/ControlzEx/blob/develop/LICENSE
[41] Math.NET License, www.github.com/mathnet/mathnet-numerics/blob/master/LICENSE.md (2018-01-01).
URL https://github.com/mathnet/mathnet-numerics/blob/master/LICENSE.md
[42] Newtonsoft.Json License, www.github.com/JamesNK/Newtonsoft.Json/blob/master/LICENSE.md (2018-01-01).
URL https://github.com/JamesNK/Newtonsoft.Json/blob/master/LICENSE.md
[43] NUnit License, www.nunit.org/nuget/nunit3-license.txt (2018-01-01).
URL http://nunit.org/nuget/nunit3-license.txt
[44] MIT License, www.opensource.org/licenses/MIT (2018-01-01).
URL https://opensource.org/licenses/MIT
Structure and Design of HoloGen

[45] ClosedXML Library License, github.com/ClosedXML/ClosedXML/blob/develop/LICENSE (2018-01-01).
  URL https://github.com/ClosedXML/ClosedXML/blob/develop/LICENSE

[46] AForge.NET Library License, www.aforgenet.com/framework/license.html (2018-01-01).
  URL http://www.aforgenet.com/framework/license.html

[47] ManagedCuda Library License, www.github.com/kunzmi/managedCuda/blob/master/LICENSE.txt (2018-01-01).
  URL https://github.com/kunzmi/managedCuda/blob/master/LICENSE.txt

[48] Accord.NET Library License, github.com/accord-net/framework/blob/development/LICENSE (2018-01-01).
  URL https://github.com/accord-net/framework/blob/development/LICENSE

[49] Lesser General Public License, www.gnu.org/licenses/lgpl-3.0.en.html (2018-01-01).
  URL https://www.gnu.org/licenses/lgpl-3.0.en.html

[50] NHotkey Library License, www.github.com/thomaslevesque/NHotkey/blob/master/LICENSE.md (2018-01-01).
  URL https://github.com/thomaslevesque/NHotkey/blob/master/LICENSE.md

[51] PdfiumViewer Library License, github.com/pvginkel/PdfiumViewer/blob/master/LICENSE (2018-01-01).
  URL https://github.com/pvginkel/PdfiumViewer/blob/master/LICENSE

[52] Apache License 2.0, www.apache.org/licenses/LICENSE-2.0 (2018-01-01).
  URL https://www.apache.org/licenses/LICENSE-2.0

[53] CefSharp Library License, github.com/cefsharp/CefSharp/blob/master/LICENSE (2018-01-01).
  URL https://github.com/cefsharp/CefSharp/blob/master/LICENSE

[54] The 3-Clause BSD License, opencores.org/licenses/BSD-3-Clause (2018-01-01).
  URL https://opencores.org/licenses/BSD-3-Clause

[55] GongSolutions.WPF.DragDrop Library License, github.com/punker76/gong-wpf-dragdrop/blob/dev/LICENSE (2018-01-01).
  URL https://github.com/punker76/gong-wpf-dragdrop/blob/dev/LICENSE

[56] Cuda EULA, docs.nvidia.com/cuda/eula/index.html (2018-01-01).
  URL http://docs.nvidia.com/cuda/eula/index.html

[57] Xamarin.Forms, www.xamarin.com/forms (2018-01-01).
  URL https://www.xamarin.com/forms

[58] D. Blinder, A. Ahar, S. Bettens, T. Birnbaum, A. Symeonidou, H. Otevaere, C. Schretter, P. Schelkens, Signal processing challenges for digital holographic video display systems, Signal Processing: Image Communication 70 (2019) 114–130.