Double lookup table method for fast light propagation calculations

Petr Lobaz

Department of Computer Science and Engineering, University of West Bohemia, Czech Republic

E-mail: lobaz@kiv.zcu.cz

Abstract. A method of rapid and robust calculation of radially symmetric functions is presented. It is based on observation that $f(x, y, z) = f'(\rho, z)$, where $\rho = (x^2 + y^2)^{1/2}$. The method stores values of f' and ρ in look-up tables. It can be used e.g for fast computer generated hologram calculation. It is suitable for CPU, GPU or hardware implementation.

1. Introduction

In computer generated holography and digital holography, it is often necessary to evaluate radially symmetric functions of two variables. They include most notably convolution kernels for free space light propagation calculation, such as Rayleigh-Sommerfeld or Fresnel convolution kernels, a free space transfer function in angular spectrum propagation method or a lens phase shift function [1]. Many researchers try to accelerate their evaluation, as it takes significant time in the whole calculation of e.g. a computer generated hologram of a 3-D scene or in light propagation simulations.

There are several approaches to the acceleration of such evaluation. First of all, a moderately complicated formula, such as the Rayleigh-Sommerfeld convolution kernel

$$\begin{aligned} K_{\mathrm{RS}}(x, y; z_0) &= -\frac{1}{2\pi}\left(\mathrm{j}k - \frac{1}{r}\right)\frac{\exp(\mathrm{j}kr)}{r}\frac{z_0}{r} \\ r &= \sqrt{x^2 + y^2 + z_0^2} \end{aligned} \tag{1}$$

can be approximated by a much simpler formula, such as the Fresnel approximation [1]

$$K_{\mathrm{Fresnel}}(x, y; z_0) = \exp\left(\mathrm{j}k\frac{x^2 + y^2}{2z_0}\right)\frac{\exp(\mathrm{j}kz_0)}{\mathrm{j}\lambda z_0}. \tag{2}$$

These convolution kernels are used to calculate monochromatic light propagation from a plane $z = 0$ to a plane $z = z_0$; λ stands for a wavelength, $k = 2\pi/\lambda$ is a wave number, x and y are transverse spatial coordinates and j is the imaginary constant ($\mathrm{j}^2 = -1$). This approach not only accelerates the function evaluation, but moreover it can improve its numerical behaviour [2]. On the other hand, of course, any approximation brings an approximation error, and must be used with caution.

A completely different acceleration method just precalculates the function for every x, y and z_0 that will be used in subsequent calculations and stores the values in a 3-D look-up table (LUT); due to radial symmetry in xy coordinates, only one quadrant or even octant must be actually saved in the look-up table [3, 4]. Despite of this enhancement, memory requirements of the method are still excessive as it is often necessary to precalculate the function for many z_0 values and discretization of x and y coordinates must be usually very fine. This is due to the fact that the evaluated function usually contains high spatial frequencies; some researchers reduce the look-up table size by taking into account that the highest spatial frequency usually depends on z_0 [5]. Naturally, the method also does not specify how to precalculate the look-up table efficiently.

One way how to reduce memory requirements is based on separability of certain functions. For example, the Fresnel convolution kernel (2) can be rewritten as

$$K_{\mathrm{Fresnel}}(x, y; z_0) = \left[\exp\left(\frac{\mathrm{j}kx^2}{2z_0}\right)\sqrt{\frac{\exp(\mathrm{j}kz_0)}{\mathrm{j}\lambda z_0}}\right] \times \left[\exp\left(\frac{\mathrm{j}ky^2}{2z_0}\right)\sqrt{\frac{\exp(\mathrm{j}kz_0)}{\mathrm{j}\lambda z_0}}\right].$$

It is easy to see that one does not need a 3-D look-up table; it is sufficient to create one 2-D look-up table for each of the two factors in the equation above, as each factor depends on just two variables [6, 7, 8]. Moreover, if the extents and the sampling distances are the same for x and y coordinates, just one 2-D look-up table is necessary. Naturally, this technique can be used only if the function to be evaluated is separable.

Some researchers try to simplify function evaluation by using recurrence formulas, e.g. [9, 10], i.e. they look for simple formulas how to calculate $K(x_0+\epsilon_x, y_0+\epsilon_y; z_0)$ using value $K(x_0, y_0; z_0)$ for small ϵ_x and ϵ_y. It should be noted that influence of computer arithmetic rounding errors is usually poorly analysed and that recurrence formulas tend to produce a sequential computer code rather than a parallel one.

An original approach to evaluation of radially symmetric function K is based on computer graphics algorithm for circle rasterization [11]; this method creates a 2-D array of values $K(m\Delta_{xy}, n\Delta_{xy}; z_0)$, where Δ_{xy} is a sampling distance in transverse coordinates and integer indices m, n span the area of interest. The method calculates $K(m\Delta_{xy}, 0; z_0)$ in a common way and this value is subsequently "copied" to samples at the same distance from the point $(0, 0; z_0)$. The biggest drawback of the method is its complicated memory access, thus memory caching cannot be used efficiently. Recently, authors proposed a method that overcomes this difficulty using recurrence formulas [12].

Parts of the method we are going to analyse in following sections were independently described by other authors [12, 13, 14]. We will discuss these references after we describe the basic idea of the method in Sec. 2; in short, the method is also based on a pair of 2-D look-up tables and it can be easily used for any function that is radially symmetric in the xy plane. Main contributions of this article are introduction of a new type of a look-up table (we will call it "rhoLUT") and a detailed analysis of interpolation in look-up tables to further reduce their size; it will be discussed mainly in Sec. 3 and 4. Sec. 5 adds details about the second type of look-up table ("waveLUT") used by the method. In Sec. 6 we discuss results of the method and Sec. 7 concludes the article. Some details of the derivations, results analysis etc. were omitted in this text; they can be found in the supplementary material provided on request by the author.

2. Principle of the method

We explain the method on a basic task in digital holography: monochromatic light propagation in free space. Let us calculate propagation of light from the plane $z = 0$ to the plane $z = z_0$ using the Rayleigh-Sommerfeld convolution, i.e.

$$U(x, y; z_0) = U(x, y; 0) \otimes K_{\mathrm{RS}}(x, y; z_0),$$

where $U(x, y, z)$ is a complex amplitude (phasor) of light at a point (x, y, z), $\otimes$ stands for convolution and $K_{\mathrm{RS}}(x, y; z_0)$ is the convolution kernel defined by Eq. (1). In the discrete calculation, x and y coordinates have to be sampled with sampling period Δ_{xy} in finite areas of planes $z = 0$ and $z = z_0$. Let us assume for simplicity that both areas in $z = 0$ and $z = z_0$ share the same square shape of the size $M\Delta_{xy} \times M\Delta_{xy}$, where M is an integer number of samples. If we want to employ the fast Fourier transform to calculate the discrete convolution, the kernel must be sampled by at least $(2M-1) \times (2M-1)$ samples with the same sampling distance Δ_{xy} [15]. Let us further assume that the centres of the square areas in $z = 0$ and $z = z_0$ are located at $x = 0$, $y = 0$. Then we need to evaluate $K_{\mathrm{RS}}(m\Delta_{xy}, n\Delta_{xy}; z_0)$ for all integers $m \in [1-M, M-1]$, $n \in [1-M, M-1]$.

It is easy to see that $K_{\mathrm{RS}}(x, y; z_0)$ depends in fact on r and z_0, where z_0 is constant and $r = (x^2 + y^2 + z_0^2)^{1/2}$ in this example. Moreover, we can define $\rho(x, y) = (x^2 + y^2)^{1/2}$ and write $r = (\rho^2 + z_0^2)^{1/2}$.

To calculate $K_{\mathrm{RS}}(m\Delta_{xy}, n\Delta_{xy}; z_0)$, it is then necessary to evaluate $\rho = \Delta_{xy}(m^2 + n^2)^{1/2}$ and $K_{\mathrm{RS}}(\rho; z_0)$. It is possible to calculate ρ directly or using a 2-D look-up table; we call this look-up table "rhoLUT"; its construction is simple, as x and y are sampled uniformly. In this particular case, it is even possible to store just one quarter of necessary values, i.e. just for $m \in [0, M-1]$, $n \in [0, M-1]$.

A simple look-up table cannot be constructed for $K_{\mathrm{RS}}(\rho; z_0)$, as the set of discrete values of ρ does not have uniform structure. Instead, we must rely on some interpolation scheme. If we assume that interpolation is sufficiently precise, we can indeed build a look-up table with values $K_{\mathrm{RS}}(q\Delta_w; z_0)$ where Δ_w is sufficiently small to capture the structure of K_{RS} and integer index q spans all possible values of ρ, i.e. $q \in [0, \lceil M\sqrt{2}\Delta_{xy}/\Delta_w \rceil]$, where $\lceil \cdot \rceil$ is the "round up" (ceil) operation. We denote this look-up table "waveLUT" because it actually captures the wave structure of light.

Even if we calculate $K_{\mathrm{RS}}(m\Delta_{xy}, n\Delta_{xy}; z_0)$ for single z_0 and all m, n, the above mentioned procedure is advantageous. The calculation of ρ is usually simple and must be done for all m, n anyway, i.e. for $(2M-1)^2$ samples. The calculation of $K_{\mathrm{RS}}(m\Delta_{xy}, n\Delta_{xy}; z_0)$ is much more involved, but due to waveLUT it is necessary to evaluate it just for $M\sqrt{2}\Delta_{xy}/\Delta_w$ samples of ρ, which is usually a much smaller number. We assume that look-up operations and interpolations are fast, which is usually the case.

It is also indeed possible to incorporate some interpolation to rhoLUT, i.e. to precalculate just values $\rho(r, s) = \Delta_\rho(r^2 + s^2)^{1/2}$ for r, s in $[0, \lceil M\Delta_{xy}/\Delta_\rho \rceil]$, where $\Delta_\rho \geq \Delta_{xy}$ is sufficiently small. In exchange of one more interpolation we get much smaller rhoLUT table. The scheme of calculation is depicted in Fig. 1.

The authors of [14] used a very similar method. Instead of rhoLUT, they calculated a look-up table directly for $r = (x^2 + y^2 + z_0^2)$, so they could not reuse it for any other z_0 coordinate as in the case of rhoLUT. Moreover, sampling distance of this table was set to Δ_{xy}, and thus keeping this table for every z_0 would be memory inefficient.

The authors of [13] use the table we call waveLUT here (they call it BPP_phase) and although they in fact use rhoLUT as well (due to Matlab style of matrix manipulation, see PFT_distance if Fig. 4 of [13]), they do not discuss it explicitly, nor do they investigate its influence.

The authors of [12] use waveLUT as well, but instead of rhoLUT, they use recurrence formulas to find values of ρ. They also do not discuss the influence of interpolation on calculation precision.

3. Introduction of the rhoLUT

The aim of the rhoLUT is to replace the calculation of $\rho(x, y) = (x^2 + y^2)^{1/2}$, $x \in [0, M\Delta_{xy}]$, $y \in [0, M\Delta_{xy}]$, by a look-up operation, optionally followed by interpolations. Let us define it as a 2-D array

$$\mathrm{rhoLUT}[m, n] = \Delta_\rho \sqrt{m^2 + n^2}$$

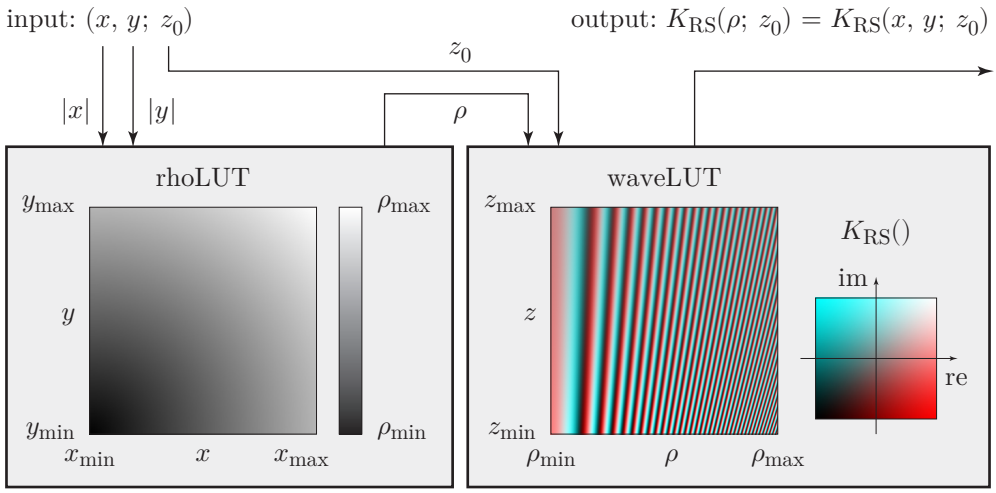

Figure 1. Calculation of radially symmetric function $K_{\mathrm{RS}}(x, y; z_0)$ using 2-D look-up tables rhoLUT and waveLUT. Complex value $(a + b\mathrm{j})$ in the waveLUT is displayed as a RGB color $[a, b, b]$, where a and b are scaled to fit 0–255 range.

for integer indices m, n in $[0, \lceil M\sqrt{2}\Delta_{xy}/\Delta_\rho \rceil]$. It is naturally possible to generalize the rhoLUT for other index or xy range, but we want to keep the analysis simple.

First of all, we should ask if such a look-up table is practical. In a general computational environment, such as a modern CPU, it depends on other circumstances. Floating point operations are provided here with sufficient precision (usually IEEE 754 double precision or better) and are quite fast; a large look-up table requiring random access to the main memory may suffer from high traffic on the bus. On the other hand, clever caching and small look-up table can easily outperform direct calculation, as especially square root operation is significantly slower than memory access.

Once we move towards more special architectures, such as GPUs or even specially designed FPGAs, reasons to use look-up table become stronger. Architecutures designed for massively parallell computing tend to keep computational cores as simple as possible. This leads to both limited instruction set and limited precision. Direct calculation forces every core to have addition, multiplication and square root operations; look-up table approach requires just memory access plus addition and multiplication for interpolations. Limited precision is even more severe – it is easy to see that calculation of $\sqrt{x^2}$ leads to loss of about half of mantissa bits. Single precision calculation (mantissa length 24 bits) is thus at the edge of applicability for wave optics calculations [2]; lower precision arithmetic such as half precision is out of the question. Contrary to direct ρ calculation, a look-up table precalculated in a high precision environment can be easily used in limited precision enviromnemt.

Finally, as rhoLUT implementation is very easy, it is often possible to implement both direct calculation and look-up table and to choose the faster approach either in advance or at runtime. In conclusion, it is advantageous to implement rhoLUT.

Let us now examine the rhoLUT details – its sampling (or size) and the interpolation of values.

In the simplest case, sampling distance of the rhoLUT equals sampling distance in x, y coordinates, i.e. $\Delta_\rho = \Delta_{xy}$, and ρ calculation for $x = m\Delta_{xy}$, $y = n\Delta_{xy}$ is trivial:

$$\begin{aligned}\rho_C(x, y) &= \mathrm{rhoLUT}\big[\,\mathrm{round}(|x|/\Delta_\rho), \mathrm{round}(|y|/\Delta_\rho)\big] \\ &= \mathrm{rhoLUT}\big[|m|, |n|\big].\end{aligned}$$

We can indeed use the same equation even if $\Delta_\rho \neq \Delta_{xy}$; now, $\rho(x, y)$ is approximated by a piecewise constant ("staircase") function $\rho_C(x, y)$. It is easy to see that the approximation error is at most $\Delta_\rho\sqrt{2}/2$ and the structure of the error is the same in the whole xy plane. This observation is important: in the next step, we are going to use ρ as an index to the waveLUT. However, local frequency of the function sampled by the waveLUT, e.g. $K_{\mathrm{RS}}(\rho; z_0)$, usually grows as $\rho \to \infty$; a small error in ρ will be thus magnified for big x, y. Piecewise constant approximation must be therefore used with the utmost caution.

As $\rho(x, 0) = |x|$, it follows that the linear interpolation between rhoLUT values gives exact results for $y = 0$. We should then expect that in a general case, bilinear interpolation should give quite precise results:

$$\begin{aligned}
\rho_B(x, y) &= (1 - i_x)(1 - i_y)\rho_{00} + (1 - i_x)i_y\rho_{01} + i_x(1 - i_y)\rho_{10} + i_x i_y \rho_{11}, \\
i_x &= \frac{|x|}{\Delta_\rho} - \left\lfloor \frac{|x|}{\Delta_\rho} \right\rfloor, \qquad i_y = \frac{|y|}{\Delta_\rho} - \left\lfloor \frac{|y|}{\Delta_\rho} \right\rfloor, \\
\rho_{st} &= \mathrm{rhoLUT}\big[\lfloor |x|/\Delta_\rho \rfloor + s, \lfloor |y|/\Delta_\rho \rfloor + t\big].
\end{aligned}$$

It can be easily found that the approximation error $\rho_B(x, y) - \rho(x, y)$ vanishes for $x \to \infty$, $y \to \infty$, and its maximum is $\Delta_\rho(2 - \sqrt{2})/4 \approx 0.146\Delta_\rho$; the maximum error is located at $(x, y) = (\Delta_\rho/2, \Delta_\rho/2)$, see Fig. 2.

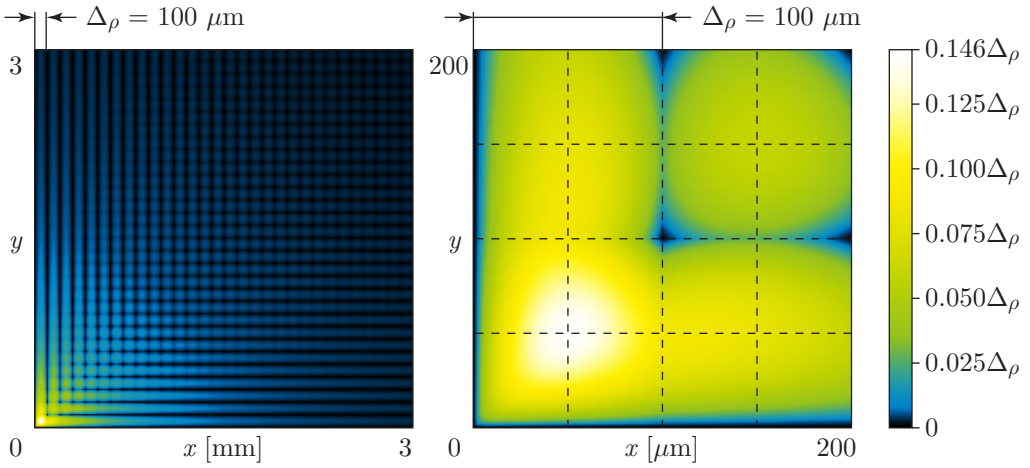

Figure 2. Visualization of $\rho_B(x, y) - \rho(x, y)$, i.e. the error introduced by the rhoLUT with bilinear interpolation. Left graph shows error vanishing, right graph shows that maximum error is located at $(x, y) = (\Delta_\rho/2, \Delta_\rho/2)$. Notice there is indeed no error for x, y being integer multiples of Δ_ρ.

4. Sampling of the rhoLUT

To select a small enough sampling distance Δ_ρ, it is not sufficient to examine error $\rho_B(x, y) - \rho(x, y)$ alone – we need to take into account that approximate value ρ_B is used in subsequent calculations. Let us assume that we are going to calculate $K_{\mathrm{RS}}(\rho_B; z_0)$ directly, without waveLUT. Effects of waveLUT approximation error will be discussed afterwards.

There are at least two ways how to analyze effect of error in ρ approximation. The first one assumes that the error of ρ is multiplied by the local frequency of $K_{\mathrm{RS}}(\rho; z_0)$; let us explain why.

Local frequency $\mathrm{lf}_{\mathrm{RS}}(\rho; z_0)$ is defined as a first derivative of the argument of the high frequency component of $K_{\mathrm{RS}}(\rho; z_0)$ [1]:

$$\mathrm{lf}_{\mathrm{RS}}(\rho_0; z_0) = \frac{\partial}{\partial \rho}\frac{r}{\lambda}\bigg|_{\rho=\rho_0} = \frac{\partial}{\partial \rho}\frac{\sqrt{\rho^2+z_0^2}}{\lambda}\bigg|_{\rho=\rho_0} = \frac{\rho_0}{\lambda\sqrt{\rho_0^2+z_0^2}}. \tag{3}$$

It tells us how many cycles per unit length can be found in the vicinity of ρ_0. Thus, unit error in ρ results in phase change $\mathrm{lf}_{\mathrm{RS}}(\rho; z_0)$ of $K_{\mathrm{RS}}(\rho; z_0)$. It is therefore necessary to analyze the function $[\rho_B(x,y) - \rho(x,y)] \times \mathrm{lf}_{\mathrm{RS}}(\rho(x,y); z_0)$.

The second way leads to the same results and is perhaps more intuitive; let us derive optimal Δ_ρ in this way. Here, only the main steps of the derivation are presented; please contact the author for supplementary material with full derivation and reasoning.

The evaluation of K_{RS} as defined in Eq. (1) depends on $r(\rho, z_0) = (\rho^2 + z_0^2)^{1/2}$. If we compare r evaluated with exact ρ and with its approximation ρ_B (using bilinear approximation), we can observe following facts (see Fig. 3):

- The difference $r\big(\rho_B(x,y), z_0\big) - r\big(\rho(x,y), z_0\big)$ approaches 0 as $\rho(x,y) \to \infty$. However, the error decreases quite slowly for $\rho \approx 0$. As the most sensitive term of K_{RS} is $\exp(\mathrm{j}2\pi r/\lambda)$, we can conclude that the overall error of K_{RS} also vanishes.
- The biggest error can be found in the vicinity of $(x,y) = (0,0)$. Moreover, maximum error for $0 < x < \Delta_\rho$, $0 < y < \Delta_\rho$ is either the global maximum of the error, or it is very close to the maximum.
- Although the error of ρ is maximal at $(x,y) = (\Delta_\rho/2, \Delta_\rho/2)$, the error of r is maximal at a different point. However, the error of r at $(x,y) = (\Delta_\rho/2, \Delta_\rho/2)$ is a rough approximate ($\approx 85\%$) of the maximum error.

We can thus define estimate of the maximum error of r when using the rhoLUT with bilinear interpolation as

$$\mathrm{maxRErr}(\Delta_\rho, z_0) = 1.2\left\{r\left[\rho_B\left(\frac{\Delta_\rho}{2}, \frac{\Delta_\rho}{2}\right), z_0\right] - r\left[\rho\left(\frac{\Delta_\rho}{2}, \frac{\Delta_\rho}{2}\right), z_0\right]\right\}, \tag{4}$$

where factor 1.2 reflects the observation that the error for $(x,y) = (\Delta_\rho/2, \Delta_\rho/2)$ is approximately 85% of the maximum. Although it is possible to analyze the error more rigorously, in numerical testing we have found that this approximation works well (see Sec. 6) and the factor 1.2 being slightly higher than 1/0.85 leads to a slightly pessimistic error estimate.

It can be easily seen in Fig. 4 that in the log-log graph, maxRErr can be approximated by a line for a wide range of useful errors and Δ_ρ. Please note that r is divided by λ in Eq. (1), and thus an acceptable error must be much lower than λ. For convenience, there is a thick dashed horizontal line in Fig. 4 at maxRErr = 1 μm, i.e. any acceptable error for visible light calculations must be well below this line.

The linear approximation in a log-log graph is defined as $\log(\mathrm{maxRErr}) \approx \kappa \log(\Delta_\rho) + \xi$, where κ is a slope and ξ is an intercept. It is easy to measure that $\kappa \approx 1.9998$ for a wide range of common z_0 and acceptable maxRErr. It is also clear that ξ depends on z_0 and this dependence is also linear in the log-log graph: $\xi = \xi_0 + \xi_1 \log(z_0)$, where ξ_0 is a value for $z_0 = 1$ and ξ_1 is the dependency factor. It can be easily measured from the graph that $\xi_0 \approx -1.9886$ and $\xi_1 \approx -0.9999$.

The linear approximation allows us to find optimal $\Delta_{\rho\ \mathrm{opt}}$ for a chosen maxRErr:

$$\Delta_{\rho\ \mathrm{opt}}(\mathrm{maxRErr}, z_0) \approx \exp\left(\frac{1}{\kappa}\log\left[\frac{\mathrm{maxRErr}}{(z_0)^{\xi_1}\exp(\xi_0)}\right]\right) \quad \text{for} \quad \begin{aligned} \kappa &\approx +1.9998 \\ \xi_0 &\approx -1.9886 \\ \xi_1 &\approx -0.9999 \end{aligned} \tag{5}$$

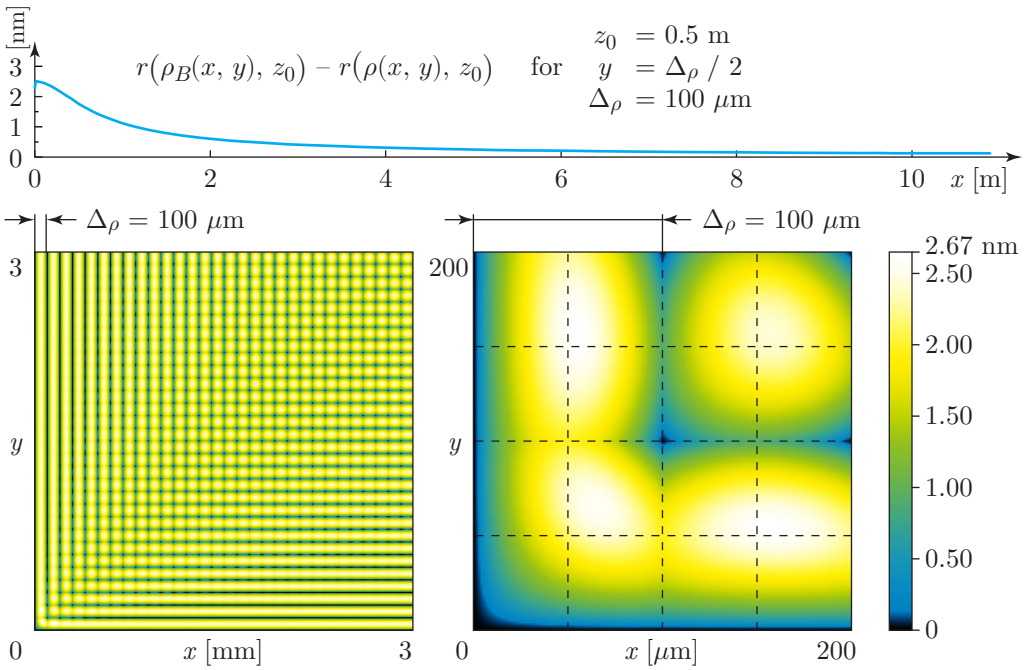

Figure 3. Visualization of $r(\rho_B(x,y),z_0) - r(\rho(x,y),z_0)$, i.e. the error introduced in evaluation of $r = (\rho^2 + z_0^2)^{1/2}$ by a rhoLUT with bilinear interpolation. The top graph shows error vanishing, the bottom left shows the error structure, the bottom right shows that error at $(x,y) = (\Delta_\rho/2, \Delta_\rho/2)$ is a rough approximation of the maximum error. Notice that each graph has different scale .

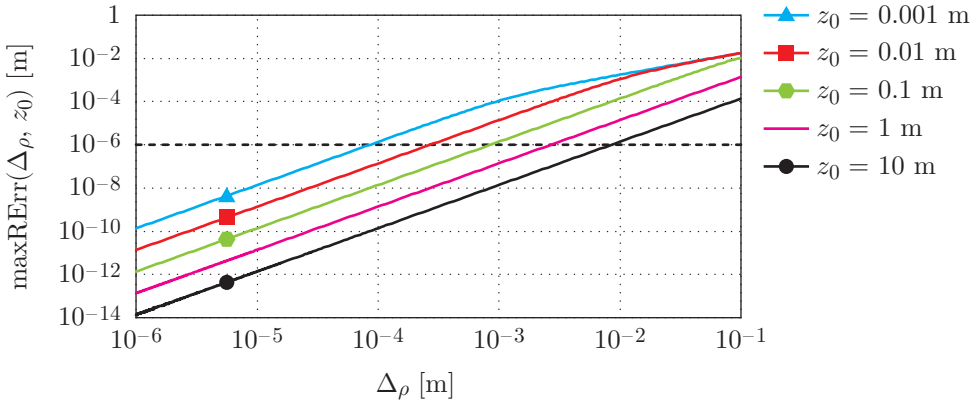

Figure 4. Approximate value of maximum error in calculation of r using rhoLUT with bilinear approximation. Useful part of the curves is below the thick horizontal line for visible light. Please note it is a log-log graph and both axes are in meters.

Example values of $\Delta_{\rho\ \mathrm{opt}}$ for $\lambda = 500$ nm can be found in Fig. 5. For example, propagation to $z_0 = 1$ m can be calculated with rhoLUT prepared with $\Delta_{\rho\ \mathrm{opt}} = 0.19$ mm if we allow error in r calculation $\lambda/100$, which is usually acceptable. If the sizes of the areas in $z = 0$ and $z = z_0$ are 50×50 mm, the rhoLUT size should be 264×264 samples. For maximum error $\lambda/10$, we set $\Delta_{\rho\ \mathrm{opt}} = 0.6$ mm and the rhoLUT size is just 84×84 samples.

We can approximate further to get some insight to Eq. (5). As $\kappa \approx 2$, $\xi_0 \approx -2$ and $\xi_1 \approx -1$,

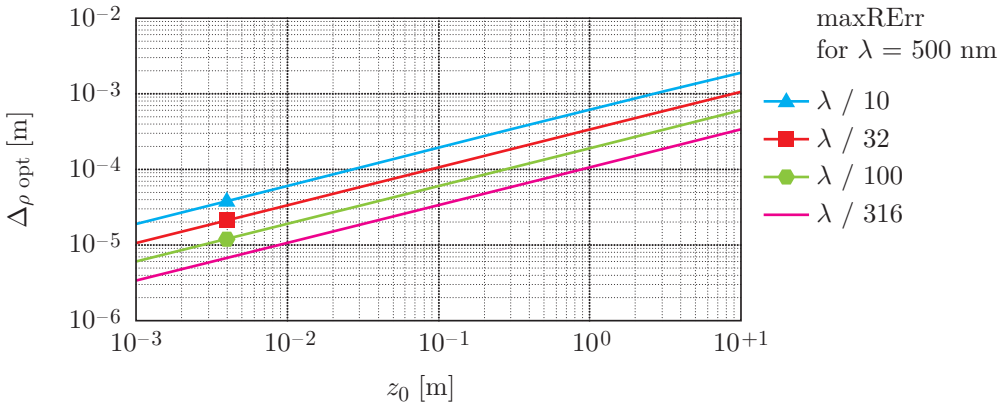

Figure 5. Optimal sampling distance $\Delta_{\rho\ \mathrm{opt}}$ for the rhoLUT calculated using Eq. (5) for $\lambda = 500$ nm. Please note it is a log-log graph and both axes are in meters.

we can approximate Eq. (5) by

$$\Delta_{\rho\ \mathrm{opt}}(\mathrm{maxRErr}, z_0) \approx 2.72\sqrt{z_0\,\mathrm{maxRErr}}. \tag{6}$$

We know that for the fixed extent of ρ, the rhoLUT size is proportional to $1/\Delta_\rho^2$ (it is a 2-D look-up table). Therefore Eq. (5) actually tells that the rhoLUT size is inversely related to maxRErr and z_0. It is also worth noting that Eq. (6) can be easily derived from the Taylor expansion of Eq. (4) with respect to Δ_ρ; in this case, the constant factor changes to ≈ 2.70.

5. The waveLUT

Once we have the approximate ρ value, we can calculate $K_{\mathrm{RS}}(\rho; z_0)$ using the waveLUT. We should set small enough Δ_w and define

$$\mathrm{waveLUT}[q; z_0] = K_{\mathrm{RS}}(q\Delta_w; z_0).$$

where q is an integer index. Again, for a particular value ρ_0 we can estimate the value of $K_{\mathrm{RS}}(\rho_0; z_0)$ using piecewise constant approximation

$$K_{\mathrm{RS\ C}}(\rho_0; z_0) = \mathrm{waveLUT}[\mathrm{round}(\rho_0/\Delta_w); z_0] \tag{7}$$

or e.g. piecewise linear approximation

$$\begin{aligned} K_{\mathrm{RS\ L}}(\rho_0; z_0) &= (1-i)\mathrm{waveLUT}\big[\lfloor \rho_0/\Delta_w \rfloor; z_0\big] + i\,\mathrm{waveLUT}\big[\lfloor \rho_0/\Delta_w \rfloor + 1; z_0\big], \\ i &= \frac{\rho_0}{\Delta_w} - \left\lfloor \frac{\rho_0}{\Delta_w} \right\rfloor. \end{aligned} \tag{8}$$

We should discuss following topics: how to deal with interpolation of z_0, how to set Δ_w, and what interpolation to use in the ρ direction.

In many applications, only a few values of z_0 are necessary or the value of z_0 is not critical. For example, in computer generated holography, it is often necessary to calculate the propagation of light from a point light source located at (x_0, y_0, z_0) to a plane $z = 0$, and it is acceptable to quantize z_0 quite coarsely. In that case, it is possible to choose a suitable quantization step Δ_z and to precalculate the waveLUT for every integer multiple of Δ_z in a given range $[z_{\min}, z_{\max}]$.

It is also possible to set Δ_z as an integer multiple of λ. Now, the waveLUT looks like its visualization in Fig. 1 – notice that the structure in the vertical (z) direction is very simple.

It should be possible to introduce some form of interpolation in this direction, but we will not discuss it here.

The extent of the waveLUT in the ρ direction is easy to set. Maximum and minimum values of ρ are given by the rhoLUT (or the geometry of the problem). For a given maximum value $\rho_{\max}$, we can calculate the local frequency $\mathrm{lf}_{\mathrm{RS}}(\rho_{\max}; z_0)$ using Eq. (3) and set $\Delta_w < 1/[2\,\mathrm{lf}_{\mathrm{RS}}(\rho_{\max}; z_0)]$ to have at least two samples per cycle of $K_{\mathrm{RS}}(\rho; z_0)$.

Our experiments show that good results are obtained with 8 samples per cycle, i.e.

$$\Delta_{w\ \mathrm{opt}} = 1/[8\,\mathrm{lf}_{\mathrm{RS}}(\rho_{\max}; z_0)] \tag{9}$$

and linear interpolation according to Eq. (8). Piecewise constant approximation (7) leads to much higher phase quantization and subsequently increases noise in the optical field. However, provided that ρ is precise enough, phase quantization itself does not destroy properties of the optical field [2]. We have also tested other interpolation schemes (cubic and various windowed sinc), but experiments show that for our purposes (computer generated holography), slightly higher accuracy of the result does not justify slower calculation.

6. Results

We have implemented several tests to measure look-up tables performance, both in speed and in precision. Here we present results of CPU tests; details on GPU implementation are in preparation and will be published elsewhere. Please note that this section presents just summary of the results; full details are available on request.

As shown in [2], direct calculation of highly oscillatory functions such as K_{RS} is prone to numerical error in single precision calculations. In short, problems appear in calculation of $\cos[2\pi(x^2 + y^2 + z_0^2)^{1/2}/\lambda]$ for visible light and $z_0 \approx 1$ m or bigger. No problems appeared when using a rhoLUT and a waveLUT precalculated in double precision. Direct calculation of ρ (i.e. omission of a rhoLUT) is possible for on-axis propagation calculations; in off-axis cases, calculation of $\rho = (x^2 + y^2)^{1/2}$ in single precision is prone to numerical errors as well. See Fig. 6 for an example of a correct and incorrect kernel calculation.

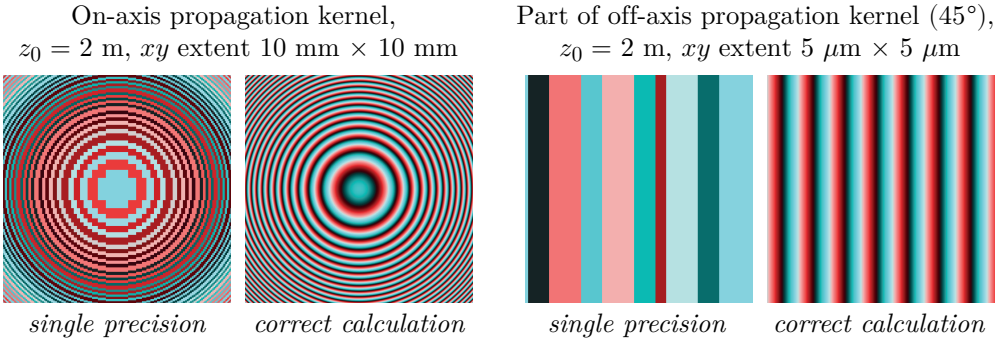

Figure 6. Examples of problems in calculation of K_{RS}. "Single precision" shows influence of rounding errors in IEEE 754 single precision arithmetic, "correct calculation" shows a correct result calculated in double precision. Calculation using look-up tables leads to the same correct results. Color coding of complex values is the same as in Fig. 1.

Tests of Δ_ρ and Δ_w selection according to Eqs. (5) and (9) confirmed theoretical analysis. For example, if we decide to accept error maxRErr = $\lambda/100$ in calculation of r, Δ_ρ calculated using Eq. (5) and bilinear interpolation in the rhoLUT leads to an actual error at most $\lambda/103$, i.e. 97% of the desired value. Other values of maxRErr lead to similar numbers.

We have also measured actual error of calculation of K_{RS}. It was calculated as the maximum or the average value of $|K - K'|/K_{\max}$, where K is the value of K_{RS} calculated precisely, K' is the value of K_{RS} calculated using look-up tables and $K_{\max}$ is the maximum of K_{RS} in the whole calculated area; the error was evaluated for both real and imaginary parts and the higher (worse) value is presented here.

Choosing 8 samples per fringe in selection of Δ_w according to Eq. (9) and linear interpolation in the waveLUT leads to maximum error 7.0% (1.1% on average) in the final propagation kernel; 16 samples per fringe to 1.9% (0.3% on average); 32 samples per fringe to $\approx 0.46\%$ (0.07% on average). While these numbers may seem high, it should be noted that the maximum error appears in the finest fringes and does not affect their frequency; thus, this error has a negligible impact on the optical field properties. Combination of the rhoLUT and the waveLUT naturally further increases the error; typical values of maxRhoErr $= \lambda/100$ and 8 samples per fringe lead to the maximum error 8.8% (2.8% on average), which is perfectly acceptable for our purposes.

Calculation time was tested in realistic geometric scenarios for propagated areas sampled by 64×64 samples to 2048×2048 samples. We have prepared two test cases – one for complicated filtered propagation kernels (see [16]) where look-up tables should clearly win over direct calculation, and one for simplified Rayleigh-Sommerfeld kernel (without $-1/r$ term) where even direct calculation is quite fast. Naturally, larger kernels benefit from look-up tables as their preparation takes comparatively smaller part of calculation time.

Complicated filtered propagation kernel calculation is accelerated mainly by utilizing waveLUT; 10× or 100× faster calculation is easily achieved, depending on the complexity of the kernel. Using waveLUT in simple kernel calculation leads to about 1.7× faster calculation. These numbers include the waveLUT calculation, which takes about 1% of the overall time.

Introducing rhoLUT enhances numerical behaviour in single precision environment; in double precision environment (CPU), this advantage is not important, as ρ can be easily evaluated directly. Unoptimized implementation of the rhoLUT can actually double calculation time compared to direct ρ calculation and the waveLUT. A slightly optimized rhoLUT implementation is approximately as fast as direct ρ calculation – this optimization exploits the fact that some values used in the rhoLUT interpolation are constant within a single row (or column) of the rhoLUT. On the other hand, careful rhoLUT implementation leads to further 20% to 40% speed-up compared to to direct ρ calculation and the waveLUT. This optimization sets Δ_ρ to an integer multiple of Δ_{xy} and uses integer arithmetic whenever possible.

7. Conclusion

We have introduced a method of calculation of arbitrary radially symmetric functions using a pair of look-up tables, a rhoLUT and a waveLUT. While using a waveLUT is always advantageous, using a rhoLUT has its pros and cons. The rhoLUT enhances numerical behaviour in a limited precision environment (such as GPU); in a high precision environment, it must be carefully implemented to improve the speed of calculation. We have also analyzed selection of look-up tables parameters and their influence on the calculation precision.

Acknowledgments

This work was supported by the European Regional Development Fund (ERDF), project NTIS New Technologies for the Information Society, European Centre of Excellence, CZ.1.05/1.1.00/02.0090, and by Ministry of Education, Youth, and Sport of Czech Republic University spec. research 1311. I would like to thank Petr Vaněček for collaboration on implementation and testing and Libor Váša for valuable comments and help with manuscript preparation.

References

[1] J. W. Goodman, *Introduction to Fourier Optics* (Roberts & Company Publishers, 2004), 3rd ed.

[2] P. Lobaz and P. Vaněček, "Safe range of free space light propagation calculation in single precision," Opt. Express **23**, 3260–3269 (2015).

[3] S.-C. Kim and E.-S. Kim, "Effective generation of digital holograms of three-dimensional objects using a novel look-up table method," Appl. Opt. **47**, D55–D62 (2008).

[4] S. Cho, B.-K. Ju, N.-Y. Kim, and M.-C. Park, "One-eighth look-up table method for effectively generating computer-generated hologram patterns," Optical Engineering **53**, 054108 (2014).

[5] T. Shimobaba, H. Nakayama, N. Masuda, and T. Ito, "Rapid calculation algorithm of Fresnel computer-generated-hologram using look-up table and wavefront-recording plane methods for three-dimensional display," Opt. Express **18**, 19504–19509 (2010).

[6] Y. Pan, X. Xu, S. Solanki, X. Liang, R. B. A. Tanjung, C. Tan, and T.-C. Chong, "Fast CGH computation using S-LUT on GPU," Opt. Express **17**, 18543–18555 (2009).

[7] G. B. Esmer, "Fast computation of Fresnel diffraction field of a three-dimensional object for a pixelated optical device," Appl. Opt. **52**, A18–A25 (2013).

[8] S.-C. Kim, J.-M. Kim, and E.-S. Kim, "Effective memory reduction of the novel look-up table with one-dimensional sub-principle fringe patterns in computer-generated holograms," Opt. Express **20**, 12021–12034 (2012).

[9] H. Yoshikawa, S. Iwase, and T. Oneda, "Fast computation of Fresnel holograms employing difference," in "Proc. SPIE," vol. 3956 (2000), pp. 48–55.

[10] K. Matsushima and M. Takai, "Recurrence formulas for fast creation of synthetic three-dimensional holograms," Appl. Opt. **39**, 6587–6594 (2000).

[11] T. Nishitsuji, T. Shimobaba, T. Kakue, N. Masuda, and T. Ito, "Fast calculation of computer-generated hologram using the circular symmetry of zone plates," Opt. Express **20**, 27496–27502 (2012).

[12] T. Nishitsuji, T. Shimobaba, T. Kakue, and T. Ito, "Fast calculation of computer-generated hologram using run-length encoding based recurrence relation," Opt. Express **23**, 9852–9857 (2015).

[13] Y. Huang, K. Zhao, X. Yan, C. Pei, and X. Jiang, "Phase-searching look-up table method for computer-generated holograms," Journal of Electronic Imaging **23**, 043013 (2014).

[14] S. Lee, H. C. Wey, D. K. Nam, D. S. Park, and C. Y. Kim, "Fast hologram pattern generation by radial symmetric interpolation," in "Proc. SPIE," vol. 8498 (2012), pp. 84980O–84980O–7.

[15] P. Lobaz, "Reference calculation of light propagation between parallel planes of different sizes and sampling rates," Opt. Express **19**, 32–39 (2011).

[16] P. Lobaz, "Memory-efficient reference calculation of light propagation using the convolution method," Opt. Express **21**, 2795–2806 (2013).