GPU-based implementation of an accelerated SR-NLUT based on N-point one-dimensional sub-principal fringe patterns in computer-generated holograms

Hee-Min Choi, Seung-Cheol Kim and Eun-Soo Kim

HoloDigilog Human Media Research Center (HoloDigilog), 3D Display Research Center (3DRC), Kwangwoon University, Korea

Abstract

An accelerated spatial redundancy-based novel-look-up-table (A-SR-NLUT) method based on a new concept of the N-point one-dimensional sub-principal fringe pattern (N-point 1-D sub-PFP) is implemented on a graphics processing unit (GPU) for fast calculation of computer-generated holograms (CGHs) of three-dimensional (3-D) objects. Since the proposed method can generate the N-point two-dimensional (2-D) PFPs for CGH calculation from the pre-stored N-point 1-D PFPs, the loading time of the N-point PFPs on the GPU can be dramatically reduced, which results in a great increase of the computational speed of the proposed method. Experimental results confirm that the average calculation time for one-object point has been reduced by 49.6% and 55.4% compared to those of the conventional 2-D SR-NLUT methods for each case of the 2-point and 3-point SR maps, respectively.

Index Terms: 3D Display

1. Introduction

Thus far, a number of approaches to generate the computer-generated holograms (CGHs) of three-dimensional (3-D) objects have been proposed [1-135]. One of them is the novel-look-up-table (NLUT) which can greatly enhance the computational speed as well as massively reduce the total number of pre-calculated interference patterns required for CGHs generation of 3-D objects [3].

In fact, the memory capacity and the calculation time have been known as two most challenging issues in the NLUT method. For reducing the memory, a new type of NLUT based on one-dimensional (1-D) sub-principal fringe pattern (1-D sub-PFPs) decomposed from the conventional 2-D PFPs, which is called 1-D NLUT, has been proposed [4]. In this method, the gigabyte (GB) memory of the conventional 2-D PFPs-based NLUT, which is called 2-D NLUT, could be dropped down to the order of megabyte (MB) memory. In addition, for enhancing the computational speed, the NLUT method employs various image compression methods for removal of both spatially and temporally redundant data of 3-D objects and 3-D moving scenes [5-8]. Among them, for removing the intra-frame redundant data, a spatial redundancy-based NLUT (SR-NLUT) was proposed [6], in which spatially-redundant object data between the adjacent pixels of the 3-D image are removed with the run-length encoding (RLE) algorithm, then the \( N \)-point PFP is applied to the NLUT for CGH generation.

Actually, for practical application of the NLUT methods mentioned above, the original NLUT and 1-D NLUT algorithms have been attempted to be implemented on field-programmable-gate-arrays (FPGAs) or graphic-processing-units (GPUs), respectively [9,10]. However, due to the limited bandwidth of the bus between the main memory directly connected to the CPU and memories in the GPU, a restoring process of the 2-D PFPs from the pre-stored 1-D sub-PFPs may deteriorate the computational performance of the GPU-based SR-NLUT system. That is, it might be practically impossible to transmit a large amount of 2-D PFPs data from the host computer to the GPU in real-time.
Therefore, in this paper, a new type of the N-point 1-D sub-PFP to greatly accelerate the CGHs calculation speed while using the small memory capacity is proposed. The proposed method is implemented on the GPU for confirming its feasibility in the practical applications. Here, the N-point 1-D sub-PFPs are generated by combined use of the 1-D sub-PFPs and the RLE algorithm. Then, in the calculation process, the N-point PFPs are generated from these pre-calculated N-point 1-D sub-PFPs, and with these the CGH patterns of 3-D objects are finally generated. A remarkable reduction of the memory capacity as well as the dramatic enhancement of the calculation speed expects to be obtained by using this new concept of the N-point 1-D sub-PFPs. Experiments with test 3-D objects are carried out and the results are comparatively analyzed with those of the conventional NLUT methods in terms of the number of calculated object points and the calculation time.

2. Proposed method

2.1 Conventional NLUT method

A geometric structure to compute the Fresnel fringe pattern of a volumetric 3-D object is shown in Fig. 1. Here, the location coordinate of the p-th object point is specified by \((x_p, y_p, z_p)\), and each object point is assumed to have an associated real-value magnitude and phase of \(a_p, \theta_p\), respectively. The hologram pattern to be calculated is also assumed to be positioned on the depth plane of \(z=0\) [3].

![Fig. 1 Geometry for generating the Fresnel hologram pattern of a 3-D object](image)

Actually, a 3-D object can be treated as a set of image planes discretely sliced along the z-direction, and each image plane having a specific depth plane is approximated as a collection of self-luminous object points of light. In the NLUT, only the 2-D PFPs representing the fringe patterns of the object points located on the centers of each depth plane are pre-calculated and stored [3]. Here, we can define the unity-magnitude PFP for the object point \((x_p, y_p, z_p)\) positioned on the center of a depth plane of \(z_p\), \(T(x, y; z_p)\) as Eq. (1).

\[
T(x, y; z_p) = \cos \left( \frac{k}{z_p} \sqrt{(x-x_p)^2 + (y-y_p)^2 + z_p^2} \right)
\]

Where the wave number \(k\) is defined as \(k = 2\pi / \lambda\), in which \(\lambda\) means the free-space wavelength of the light. Thus, the fringe patterns for other object points on the depth plane of \(z_p\) can be obtained by simply shifting the PFP of Eq. (1). These shifted versions of PFPs are added together to get the CGH pattern for the depth plane of \(z_p\). In addition, this process is carried out for all depth planes of the 3-D object to get the final CGH pattern of that object. Therefore, in the NLUT method, the CGH pattern for an object \(I(x, y)\) can be expressed in terms of the shifted versions of pre-calculated PFPs of Eq. (1) as shown in Eq. (2).

\[
I(x, y) = \sum_{p=1}^{P} a_p T(x-x_p, y-y_p; z_p)
\]

Where \(P\) denotes the number of object points. Equation (2) shows that the NLUT may enable obtaining the CGH pattern of a 3-D object just by combination of shifting and adding operations of the PFPs on each depth plane of the 3-D object.

2.2. Proposed method

Figure 2 shows an overall block-diagram of the proposed 1-D SR-NLUT method for the accelerated computation of CGH patterns for the 3-D object, which is largely composed of four steps. In the first step, spatial redundancy of the intensity and depth data of the 3-D object is preprocessed by using the RLE method and they are re-grouped into the N-point redundancy map according to the number of the neighboring object points having the same 3-D value. In the second step, N-point sub-PFPs corresponding to the N-point redundancy maps are calculated by shifting and adding the 1-point sub-PFP of the conventional 1-D NLUT. In the third step, the CGH pattern of the 3-D object is calculated with this pre-calculated N-point sub-PFPs. In the fourth step, the 3-D object image is reconstructed from the calculated CGH pattern.

![Fig. 2 Block diagram of the proposed method for generation of the CGH pattern for the 3-D image](image)
2.2.1 Extraction of the spatial redundancy from a 3-D object

In case adjacent pixels of a 3-D object have a same value of color and depth, it is called a spatial redundancy both in intensity and depth data of the input 3-D image [6,176-19]. Figure 3 shows a concept of spatial redundancy in the 3-D input image.

Figure 3(a) shows a 3-D object with 5x5 resolutions at a depth plane and only three kinds of gray values such as 10, 150, and 255. Figure 3(b) also shows a SR map that is horizontally extracted from Fig. 3(a) using the RLE method. Here, ‘3/255’ means that there exist three adjacent image pixels having the same gray value of ‘255’ in the corresponding row.

As seen in Fig. 3, 13((=4+6+3) calculation processes are needed in the proposed method for generation of the CGH pattern contrary to the conventional NLUT method where 25((=5×5) calculation processes are normally needed. That is, in the proposed method, 13 calculation processes can be reduced in CGH generation. There exist fifteen empty spaces called ‘don’t care condition’ in Fig. 3(b), in which ‘don’t care condition’ means no need of CGH calculation.

2.2.2 Generation of N-point PFPs and N-point sub-PFPs

In the SR-NLUT method, the 1-point PFP for one object point is defined as Eq. (1) mentioned above in the conventional NLUT. And, the 2-point PFP for two adjacent object points with unity magnitude and depth of \( z_p \) can be expressed by Eq. (3).

\[
T_2(x, y, z_p) = T(x, y; z_p) + T(x - d, y; z_p)
\]

Where \( d \) represents a discretization step of adjacent points [3,6]. Likewise, the N-point PFP for \( N \) adjacent object points with unity magnitude and depth of \( z_p \), \( T_n(x, y; z_p) \) can be expressed by Eq. (4).

\[
T_n(x, y; z_p) = \sum_{k=1}^{n} T(x - (k - 1)d, y; z_p)
\]

Therefore, the N-point PFP of Eq. (5) can be derived by using a set of 1-D sub-PFPs.

\[
T_n(x, y; z_p) = \sum_{k=1}^{n} T(x - (k - 1)d, y; z_p)
\]

Where, \( S_{x,1}, S_{y,1}, S_{z,1} \) and \( S_{z,1} \) mean the 1-point 1-D cosine sub-PFP, 1-point 1-D sine sub-PFP, N-point 1-D cosine sub-PFP and N-point 1-D sine sub-PFP, respectively. Therefore, the N-point cosine and sine sub-PFPs can be expressed be Eq. (6).

\[
S_{x,n}(x; z_p) = \sum_{k=1}^{n} S_{x}(x - (k - 1)d; z_p)
\]

Figure 4 shows a flowchart to generate the N-point sub-PFPs for an arbitrary depth plane of Eq. (6) using the proposed method. Here, we consider three adjacent points located on a depth plane of \( z_i; A(0, 0, z_i), B(d, 0, z_i) \) and \( C(2d, 0, z_i) \) as shown in Fig. 4(a). Fig. 4(b) also shows the 1-point sub-PFP for an arbitrary depth plane of \( z_i \) in the conventional 1-D NLUT. Thus, the 3-point sub-PFP for three adjacent object points can be calculated by simple shifting and adding operations of the 1-point sub-PFP.

![Fig. 4 A generation process of the 3-point sub-PFP from the 1-D sub-PFP](http://dz.doi.org/10.7840/ictx.2014.000004)
GPU-based implementation of an accelerated SR-NLUT based on N-point one-dimensional sub-principal fringe patterns in computer-generated holograms

3. Experiments and the results

In the experiment, three types of 3-D objects: ‘Dice’, ‘Car’ and ‘House and Car’, are used as the test objects, and their intensity and depth images are shown in Fig. 6. Here, the resolutions of each test 3-D object and the CGH pattern are assumed to be 300×300×256 pixels and 1,920×1,080 pixels, respectively in which each pixel size is given by 10μm×10μm. The horizontal and vertical discretization steps of less than 30μm (100μm×0.003 = 30μm) are chosen since the viewing-distance is assumed to be 100mm. Accordingly, to fully display the fringe patterns, the 2-D PFP must be shifted by 900 pixels (300×3 pixels = 900 pixels) horizontally and vertically. Thus, the total resolution of the 2-D PFP becomes 2,820 (1,920 + 900) × 1,980 (1,080 + 900) pixels. For the 1-D NLUT and proposed methods, only two sets of N-point 1-D sub-PFPs are needed. Therefore, the total resolution of the 1-D PFP becomes 2,820 (1,920 + 900) × 1 pixels.

![Fig. 6 3-D test object images: (a)-(c) Intensity images and (d)-(f) Depth images](http://www.ictexpress.org)

In the experiment, a PC system employing an Intel Pentium i7-3770 operating at 3.4 GHz, 8 GB RAM and a Linux CentOS as well as the Nvidia GTX titan are used for hardware implementation of the proposed method.

Figure 7 shows the SR maps extracted by horizontal scanning of the test objects of Fig. 6 using the RLE algorithm. In the extracted redundancy maps, the gray color means that there are no adjacent object points having the same intensity and depth values, while the green and blue colors mean that two and three adjacent object-points have the same intensity and depth values, respectively. In addition, the white color means the object points of ‘don’t care condition’.
Table 1 shows a distribution of the spatially redundant data of the test 3-D objects along the horizontal direction. As shown in Table 1, the spatially redundant data of the ‘Dice’ object are estimated to be 15,317, 11,295, 8,507, respectively for each case of the conventional and proposed methods (2- and 3-point cases). Thus, the numbers of object point to be calculated of the proposed method have been reduced by 32.8% and 44.5%, respectively for 2-point and 3-points cases, respectively by using the spatial redundancy when they are compared to that of the conventional 1-D NLUT method.

<table>
<thead>
<tr>
<th>Number of spatially redundant data of the test 3-D object</th>
<th>Conventional 1-D NLUT (1-point)</th>
<th>Proposed method (2-point)</th>
<th>Proposed method (3-point)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dice</td>
<td>15,317 (100%)</td>
<td>10,295 (67.2%)</td>
<td>8,507 (55.5%)</td>
</tr>
<tr>
<td>Car</td>
<td>15,627 (100%)</td>
<td>12,906 (82.6%)</td>
<td>11,923 (76.3%)</td>
</tr>
<tr>
<td>House &amp; Car</td>
<td>17,705 (100%)</td>
<td>13,114 (74.1%)</td>
<td>11,684 (66.0%)</td>
</tr>
</tbody>
</table>

Likewise, for the case of ‘Car’ the numbers of object points to be calculated have been reduced by 17.4% (2-point case) and 23.7% (3-point case) compared to that of the conventional method. Furthermore, for the case of the ‘House & Car’, the numbers of object points to be calculated have been also reduced by 25.9% (2-point case) and 34.0% (3-point case) compared to that of the conventional method, respectively.

Figure 8 shows three types of object images reconstructed from the CGH patterns generated with the conventional and proposed methods. In case of ‘Dice’ Fig. 8(a) shows the focused images of the front ‘Die’ reconstructed at the distance of 684mm and the rear ‘Die’ reconstructed at the distance of 720mm from the CGH pattern generated by using the conventional and proposed methods, respectively.

As seen in Fig. 8(a), in all cases objects images have been successfully reconstructed. That is, object images of the front ‘Die’ are clearly focused but object images of the rear ‘Die’ are blurred at the reconstruction distance of 720mm. On the other hand, at the reconstruction distance of 684mm, object images of the rear ‘Die’ are focused, but the object images of the rear ‘Die’ are out of focused.

Table 2 shows the CGH calculation times for each case of the conventional 2-D NLUT, 2-D SR-NLUT, 1-D NLUT and proposed methods. Table 2 also shows the detailed calculation times for each step of the CGH generation process such as hologram generation, pre-processing and loading time of the PFP. As seen in Table 2, in case of ‘Dice’, 422.76ms, 307.10ms and 269.82ms are needed to generate the CGH pattern for each case of N=1, 2, 3 in the 2-D NLUT and 2-D SR-NLUT methods. That is, the CGH calculation time gets decreased by removing the spatially redundant data from the 3-D object. However, the total calculation time is given by 794.75ms, 771.31ms and 796.94ms for each case of the 2-D NLUT and 2-D SR-NLUT.
NLUT methods. That is, the total calculation time decreases for the case of $N=2$, while it increases for the case of $N=3$. By comparing the cases of $N=2$ and $N=3$ in the 2-D SR-NLUT method, the loading time of the PFP is increased by 62.90 ms, even though the hologram generation time is decreased by 37.28 ms. That is, the loading time for the $N$-point PFPs gets larger than the hologram calculation time for the reduced object points by removing the redundant data of 3-D object. That is, the loading time of the $N$-point 2-D PFP may increase if the $N$ number gets increased because the number of $N$-point 2-D PFPs to be loaded on the GPU is increased if the $N$ number gets increased. In the conventional 2-D NLUT and 2-D SR-NLUT methods, the loading time of the PFP on the GPU is composed of a large portion of the total calculation time. That is, the loading time of the $N$-point 2-D PFP occupies 57.5% of the total calculation time. On the other hand, the pre-processing time required for extraction of the spatial redundancy of the 3-D object, possess an extremely small portion on the total calculation time. That is, the pre-processing time occupies only 0.05% in the total calculation time.

### Table 2 Calculation time for each of the conventional 2-D NLUT, 2-D SR-NLUT, 1-D NLUT and proposed methods

<table>
<thead>
<tr>
<th></th>
<th>2-D NLUT</th>
<th>2-D SR-NLUT</th>
<th>1-D NLUT</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N=1$</td>
<td>$N=2$</td>
<td>$N=3$</td>
<td>$N=1$</td>
</tr>
<tr>
<td>CGH generation time (ms)</td>
<td>422.76</td>
<td>307.10</td>
<td>269.82</td>
<td>427.65</td>
</tr>
<tr>
<td>%</td>
<td>(53.19)</td>
<td>(39.82)</td>
<td>(33.86)</td>
<td>(98.5)</td>
</tr>
<tr>
<td>Pre-processing time (ms)</td>
<td>0.34</td>
<td>0.40</td>
<td>0.41</td>
<td>0.34</td>
</tr>
<tr>
<td>%</td>
<td>(0.04)</td>
<td>(0.05)</td>
<td>(0.05)</td>
<td>(0.08)</td>
</tr>
<tr>
<td>PFP loading time (ms)</td>
<td>371.65</td>
<td>463.81</td>
<td>526.71</td>
<td>618.00</td>
</tr>
<tr>
<td>%</td>
<td>(46.76)</td>
<td>(60.13)</td>
<td>(66.09)</td>
<td>(1.42)</td>
</tr>
<tr>
<td>Total calculation time (ms)</td>
<td>794.75</td>
<td>771.31</td>
<td>796.94</td>
<td>434.17</td>
</tr>
<tr>
<td>%</td>
<td>(100%)</td>
<td>(100%)</td>
<td>(100%)</td>
<td>(100%)</td>
</tr>
<tr>
<td>CGH generation time (ms)</td>
<td>431.54</td>
<td>378.43</td>
<td>366.10</td>
<td>430.57</td>
</tr>
<tr>
<td>%</td>
<td>(54.48)</td>
<td>(42.93)</td>
<td>(37.08)</td>
<td>(98.52)</td>
</tr>
<tr>
<td>Pre-processing time (ms)</td>
<td>0.33</td>
<td>0.44</td>
<td>0.44</td>
<td>0.32</td>
</tr>
<tr>
<td>%</td>
<td>(0.04)</td>
<td>(0.05)</td>
<td>(0.05)</td>
<td>(0.07)</td>
</tr>
<tr>
<td>PFP loading time (ms)</td>
<td>369.5</td>
<td>502.73</td>
<td>620.79</td>
<td>617.00</td>
</tr>
<tr>
<td>%</td>
<td>(45.48)</td>
<td>(57.02)</td>
<td>(62.88)</td>
<td>(1.41)</td>
</tr>
<tr>
<td>Total calculation time (ms)</td>
<td>792.17</td>
<td>881.60</td>
<td>987.33</td>
<td>437.06</td>
</tr>
<tr>
<td>%</td>
<td>(100%)</td>
<td>(100%)</td>
<td>(100%)</td>
<td>(100%)</td>
</tr>
</tbody>
</table>

That is, the numbers of PFPs in the 1-D methods are same with those of the 2-D methods, but the memory sizes of each $N$-point PFP in the 1-D methods are extremely small compared to those of the 2-D methods. Therefore, the loading time of the $N$-point 1-D sub-PFPs occupied only 2.2% in the total calculation time. And then, the total calculation times becomes 434.17 ms, 333.59 ms and 303.83 ms for each case of the 1-D NLUT and proposed methods (2 and 3-point cases), respectively. By comparing the cases of $N=2$ and $N=3$ in the proposed methods, the loading time of the PFP is increased by only 1.11 ms even though the CGH generation time is decreased by 30.88 ms. That is, the total calculation time has been decreased just by removing the spatially redundant data of the 3-D object. On the other hand, the pre-processing time occupies 0.11% in the total calculation time in the 1-D NLUT methods.

In the same way, in case of 'Car' the loading time of the PFP gets decreased from 454.06 ms to 7.42 ms by applying the $N$-point 1-D sub-PFPs. Therefore, the total calculation time is given by 792.17 ms, 881.60 ms and 987.33 ms in the 2-D NLUT and 2-D SR-NLUT (2- and 3-points cases) methods, respectively. That is, the total calculation time is increased despite the spatially redundant data of the 3-D object gets removed. However, the total calculation time is given by 437.06 ms, 397.40 ms and 391.30 ms in the 1-D NLUT and proposed (2- and 3-points cases) methods, respectively. That is, the total calculation time has been decreased just by removing the spatially redundant data of the 3-D object.

Likewise, in case of ‘House and car’ object, the loading time of the PFP is decreased from 125.26 ms to 4.20 ms by applying the $N$-point 1-D sub-PFP. The loading time of this case is very small compare to other cases because only 49, 77 and 101 PFPs are loaded for the case of $N=1$, 2 and 3, respectively, because the ‘House and Car’ object has only 49 depth layers in all 3-D object. The total calculation time is given by 518.01 ms, 488.44 ms and 495.48 ms in the 2-D NLUT and 2-D SR-NLUT (2- and 3-points cases) methods, respectively. However, the total calculation time is given by
450.06 ms, 361.93 ms and 336.55 ms in the 1-D NLUT and proposed (2- and 3-points cases) methods, respectively.

Table 3 shows the average calculation time for one-object point in the conventional 2-D NLUT, 2-D SR-NLUT, 1-D NLUT and proposed methods, respectively. As seen in Table 3, in case of ‘dice’ object, the average calculation time for one-object point is given by 51.89 μs, 50.36 μs and 52.03 μs in the conventional 2-D NLUT and 2-D SR-NLUT (2- and 3-points cases) methods, respectively. That is, the average calculation time for one-object point for the case of N=2 is reduced by 2.9% compared to that of the 2-D NLUT method. However, the average calculation time for one-object point for the case of N=3 is increased by 0.3% and 3.2% compared to those of the 2-D NLUT and 2-D SR-NLUT (N=2) methods. That is, as mentioned above, the loading time for the N-point PFPs gets larger than the CGH calculation time for the reduced object point by removing the redundant data of the 3-D object.

Table 3 Average calculation time and required memory space for one-object point in each case of the conventional 2-D NLUT, 2-D SR-NLUT, 1-D NLUT and proposed methods

<table>
<thead>
<tr>
<th>Method</th>
<th>2-D NLUT</th>
<th>2-D SR-NLUT</th>
<th>1-D NLUT</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N=1</td>
<td>N=2</td>
<td>N=3</td>
<td>N=1</td>
</tr>
<tr>
<td>Dice</td>
<td>51.89 μs</td>
<td>50.36 μs</td>
<td>52.03 μs</td>
<td>28.35 μs</td>
</tr>
<tr>
<td></td>
<td>(100%)</td>
<td>(97.1%)</td>
<td>(100.3%)</td>
<td>(54.6%)</td>
</tr>
<tr>
<td>Car</td>
<td>50.69 μs</td>
<td>56.42 μs</td>
<td>63.18 μs</td>
<td>27.97 μs</td>
</tr>
<tr>
<td></td>
<td>(100%)</td>
<td>(111.3%)</td>
<td>(124.6%)</td>
<td>(55.2%)</td>
</tr>
<tr>
<td>Average</td>
<td>43.95 μs</td>
<td>44.79 μs</td>
<td>47.73 μs</td>
<td>27.24 μs</td>
</tr>
<tr>
<td></td>
<td>(100%)</td>
<td>(101.9%)</td>
<td>(108.6%)</td>
<td>(62.0%)</td>
</tr>
<tr>
<td>Memory</td>
<td>1.33 GB</td>
<td>2.66 GB</td>
<td>3.99 GB</td>
<td>1.38 MB</td>
</tr>
<tr>
<td></td>
<td>(100%)</td>
<td>(200%)</td>
<td>(300%)</td>
<td>(0.1%)</td>
</tr>
</tbody>
</table>

By comparison with the 2-D NLUT and 2-D SR-NLUT methods, the conventional 1-D NLUT and 1-D SR-NLUT methods need smaller loading times. As shown in Table 3, in case of ‘dice’, the average calculation time for one-object point is given by 28.35 μs, 21.78 μs and 19.84 μs in the conventional 1-D NLUT and 1-D SR-NLUT methods. That is, the average calculation time for one-object point is reduced by 45.4%, 58.0% and 61.8% compared to those of the 2-D NLUT method, respectively.

In the same way, in case of ‘Car’ object, average calculation time for one-object point is given by 50.69 μs, 56.42 μs and 63.18 μs in the conventional 2-D NLUT and 2-D SR-NLUT methods, respectively. That is, the average calculation time for one-object point is increased in spite of removing the spatially redundant data of 3-D object. However, the average calculation time for one-object point is given by 27.97 μs, 25.43 μs and 25.04 μs in the conventional 1-D NLUT and proposed methods, respectively. That is, the average calculation time for one-object point is reduced by 44.8%, 54.9% and 60.4% compared to each of the 2-D methods, respectively, by replacing the N-point 2-D PFPs into the N-point 1-D sub-PFPs.

Likewise, in case of ‘House and car’ object, the average calculation time for one-object point is given by 29.26 μs, 27.59 μs and 27.99 μs in the conventional 2-D NLUT and 2-D SR-NLUT methods, respectively. However, the average calculation time for one-object point is given by 25.42 μs, 20.44 μs and 19.01 μs in the conventional 1-D NLUT and proposed methods, respectively. That is, the average calculation time for one-object point is reduced by 13.1%, 32.1% and 61.8% compared to each of the 2-D methods, respectively, by replacing the N-point 2-D PFPs into the N-point 1-D sub-PFPs.

Thus, the average calculation time for one-object point for all three cases is given by 43.95 μs, 44.79 μs, 47.73 μs, 27.24 μs, 22.55 μs and 21.29 μs in the conventional 2-D NLUT, 2-D SR-NLUT (N=2, 3), 1-D NLUT and proposed methods (N=2, 3), respectively. That is, the average calculation time for one-object point is reduced by 38.0%, 49.6% and 55.4% compared to each of the 2-D methods, respectively, by replacing the N-point 2-D PFPs into the N-point 1-D sub-PFPs.

In addition, the memory capacities required in the conventional 2-D NLUT, 2-D SR-NLUT and 1-D NLUT methods as well as the proposed method are also calculated. As seen in Table 3, the total memory size required for storing all N-point PFPs of the 3-D image volume of 300 × 300 × 256 pixels in the conventional 2-D NLUT and 2-D SR-NLUT methods are calculated to be 1.33 GB, 2.66 GB and 3.99 GB, respectively, in which image data for one PFP is assumed to be 5.32 MB (= 2820×1980×8bit). For 1-D NLUT and proposed methods, only two sets of N-point 1-D sub-PFPs are needed, therefore, the total memory size required for storing all N-point sub-PFPs are calculated to be 1.38 MB, 2.75 MB and 4.13 MB for the 1-D NLUT and proposed methods, respectively. In other words, the proposed method only uses 0.2%, 0.3% of the memory volume of the conventional 2-D NLUT method for each.
case of the 2, 3-point SR maps, respectively. Therefore, the loading time of the N-point PFPs can be dramatically reduced compared to those of the conventional 2-D NLUT and 2-D SR-NLUT methods.

3. Experiments and the results

In this paper, a novel approach to massively reduce the memory capacity as well as to dramatically reduce the calculation time of the conventional NLUT method has been proposed by combined use of the 1-D sub-PFP and the spatial redundancy of the 3-D object for its GPU-based implementation. Experiments with three types of test 3-D objects confirm that the average calculation times for one-object point of the proposed method have been reduced by 49.6% and 55.4% compared to those of the 2-D SR-NLUT methods for each case of the 2, 3-point SR maps. In addition, the proposed method has been found to use only 0.2% and 0.3% of the memory volume of the conventional 2-D NLUT method for each case of the 2-point and 3-point SR maps.

References