Real Time Relighting with Dynamic Light Environment Using an RGB-D Camera

Takuya Ikeda, Francois de Sorbier and Hideo Saito
Graduate School of Science and Technology
Keio University, 3–14–1, Hiyoshi, Kouhoku–ku, Yokohama–shi, Kanagawa–ken, 223–0061, Japan

***Abstract*—Relighting techniques often require knowledge about the shape and the reflectance property of a target object and the light environment. The quality of the relighting is highly dependent on the normals accuracy of the object since they are used for the computation of the illumination. We propose a new relighting approach for arbitrary shaped objects with dynamic light environment using an RGB-D camera such as the Microsoft's Kinect. The generated depth map is useful to estimate the normals of the object, but it's inaccurate because of the noise such as discrete depth values or missing data. We focus on the depth map modification to segment the accurate object region, and normal estimation for relighting. Our implementation achieves a relighting at 10fps with GPU. Therefore, our relighting system can apply for a moving object and a dynamic light environment in real time.**

Keywords*—*Relighting, GPU, RGB-D camera.

I. INTRODUCTION

Relighting is a currently active research topic in computer vision. It can be applied not only to videos but also to matching technique that use SIFT or SURF which are weak on change of illumination conditions. However, getting an accurate relighting result remains a complicated task. Generally, relighting technique needs the object shape, the reflectance property and light environments. The accuracy of the results depends on accuracy of each of these factors.

In previous works, there are some researches focusing on relighting for fixed or moving objects with off-line processing because of the difficulty of obtaining the exact shape and the reflectance property in real time.

Zhen *et al*. [1] and Oswald *et al*. [2] proposed relighting techniques applied on human face images by using compute vision techniques. The human face geometry (normal map) can be estimated with a morphable model of 3D faces. Zhen *et al*. proposed a rate image technique for relighting. Processing time of this method is fast because they only considered the diffuse component and approximated the radiance environment map with Spherical Harmonics (SH) [3]. In our case, we also use this technique in our proposed relighting flow. Oswald *et al*. also estimated the face geometry and relighted it by considering not only the diffuse component but also the specular one. Considering specular component, more accurate and natural relighting results can be obtained. However, as mentioned before, these works only treated a stationary human face image.

Debevec *et al*. proposed acquiring the reflectance field of the object by using Light Stage [4]. They captured the object under 2048 light directions for estimating the reflectance functions of the object, so that the high accuracy relighting results can be obtained by using these functions. This method does not need to estimate the object normal because the reflectance functions include not only the object's diffuse and specular reflectance but also visibility from all light directions. Wenger *et al*. obtained relighting results for moving object such as human movements using newer Light Stage and high speed camera [5]. Their system lights up in high-speed and high speed camera captures a large number of images in each object pose. After that, they constructed the reflectance functions in image sequence. In contrast their works, we propose the real time relighting system without such a lot of images.

In this paper, we proposed real time relighting with dynamic light environment using an RGB-D camera. Depth sensor can obtain a scene geometry in real time (more than 30 fps). As far as we have investigated, such approaches using RGB-D camera have not been proposed yet. For arranging the object on to another scene background, segmentation of the correct object region is important. And, the object normals significantly affect the accuracy of the relighting result. However, the boundaries in the depth map does not always match color's ones and the depth map has noisy data such as discrete depth values or missing data. Because of these, it is not easy to accurately segment and to estimate the normals from a single depth map. We focus on these two processes. Firstly, we modify the depth map using color and depth information. Thus, we obtain improved depth map especially in the boundary between the object and background and the object region is segmented with a simple depth thresholding. After that, we estimate the normal map from the modified depth map. We use normal estimation method proposed by Stefan *et al*. [6]. Their method uses integral images and smooth normal map can be obtained.

In our flow, we use some processing which are integral image, Kmeans clustering and SH rotation. Recently, parallel algorithms of these processing with GPU are proposed [7], [8], [9]. We apply these methods and all processing are implemented by GPU or GPU + CPU. In Sec.V, we discuss about the experimental results and processing time.

II. RELIGHTING THEORY

A. Spherical Harmonics Lighting

In this section, we explain about a relighting theory. Image based lighting is very useful for rendering synthetic objects into real scenes such as in Augmented Reality and Mixed Reality. This technique is also useful for the relighting. We

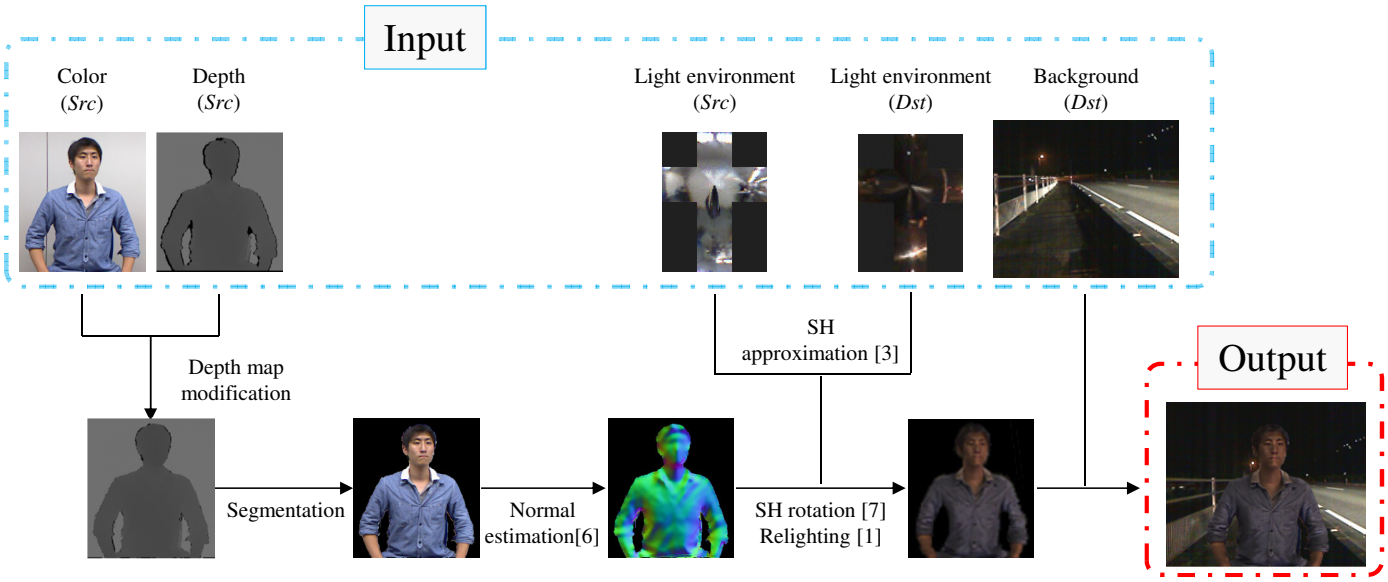


Fig. 1. Flow of proposed our relighting.

assumed that a light source come from infinity distance. For considering a point light source in all directions, we assume that light environment is distributed on a unit sphere. The irradiance $E(\boldsymbol{x})$ observed at a point $\boldsymbol{x} = (x, y, z)$ is given by an integral over the sphere Ω.

$$E(\boldsymbol{x}) = \int_{\Omega} L(\omega) max((\omega \cdot \boldsymbol{n}), 0) d\omega \tag{1}$$

Note that $L(\omega)$ represents the light intensity along the direction vector ω. $(\omega \cdot \boldsymbol{n})$ is the cosine term and $\boldsymbol{n}$ is a normal vector at a point $\boldsymbol{x}$. This model considers only shading model (does not consider shadow).

We obtain a light environment from captured light probe [10] and approximate the light environment in SH [3]. Using the notation of [3], the irradiance of point $\boldsymbol{x}$ can be represented as a linear combination of SH basis functions,

$$E(\boldsymbol{x}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} A_l(\theta) L_{lm} Y_{lm}(\omega) \tag{2}$$

For computing $E(\boldsymbol{x})$ with arbitrary normal vector $\boldsymbol{n}$, we compute the SH rotation [7] for a term of $A_l(\theta)$. Preliminarily, we compute the standard cosine term A_l^{std} is equal to $A_l(0)$. After normal estimation, we compute SH rotation from A_l^{std} to $A_l(\theta)$. Note that, θ is a elevation angle corresponding to the normal direction $\boldsymbol{n}$.

Ramamoorthi and Hanrahan [3] showed that for diffuse reflectance, only 9 coefficients are needed to approximate the irradiance function. Assuming Lambertian surface reflectance, the pixel value of $i(\boldsymbol{u})$ is written as

$$i(\boldsymbol{u}) = R_d \sum_{l=0}^{2} \sum_{m=-l}^{l} A_l(\theta) L_{lm} Y_{lm}(\omega) \tag{3}$$

where $\boldsymbol{u} = (u, v)$ represents pixel position corresponding to point $\boldsymbol{x}$. R_d is a diffuse reflection parameter of the object surface. Since lambertian surface uniformly reflects the light to all directions, R_d is set to a constant value.

B. Relighting in different light environment

In relighting part, we adapt the rate image technique [1]. We need to know two light environments. The first is a scene captured the object and the other is a scene which light environment is different from the first one. In this paper, we call these two scenes *Src* and *Dst* respectively. If *Src* and *Dst* light environments and the object normal are known, the pixel intensity of relighting image is obtained by the ratio of Eq. (2),

$$i^{dst}(\boldsymbol{u}) = i^{src}(\boldsymbol{u}) \frac{E^{dst}(\boldsymbol{x})}{E^{src(\boldsymbol{x})}} \tag{4}$$

Note that $i^{src}(\boldsymbol{u})$ is an input pixel value and E^{src} is calculated from the *Src* light environment, $i^{dst}(\boldsymbol{u})$ and E^{dst} are same representation in the *Dst* light environment. By computing the rate of two irradiance functions, the constant value R_d can be removed. This means that we can obtain the relighting results without estimating the reflectance property.

III. RELIGHTING USING AN RGB-D CAMERA

In this section, we explain about the proposed relighting approach. Our relighting flow is shown in Fig. 1. Input images are *Src* and *Dst* light environments, the object color image, depth map and *Dst* background image. Light environments are represented in cube map. Note that, *Dst* light environment and back ground are image sequence in our relighting system. For relighting, segmentation of the object region and normal estimation is very important. In our strategy, the object region is segmented using a depth thresholding. However, the boundaries in depth map does not always match color's ones because of the noisy depth map. Therefore, before segmentation, we modify the depth map around the object boundary. After that, normal map is estimated using the modified depth map and we compute the SH rotation for the term of $A_l(\theta)$. Finally,

the relighting result is obtained by computing Eq. (4) and it is arranged on *Dst* background.

A. Depth Map Modification for Exact Segmentation

The strategy of depth map modification is similar to the one from Chen et al. [11]. In our case, we only modify the boundary between the object and background. In the first step, we set the invalid depth region around the object boundary. In the second step, the invalid depth is filled by using color and depth information of the neighbor pixels, as depicted in Fig. 2.

Firstly, we segment the object region by using simple depth thresholding. Large segmented region which is shown in Fig. 2 (a) is obtained by applying a dilatation operator. After that, we compute the invalid depth region by using color and depth edge information. Both image edges are obtained by Canny Operator. The depth edge region mask M_{der} is obtained by expanding the edge pixels with a square window. Same operation is adapted in the color image and we obtain the color edge region mask M_{cer}. Then, final mask of the invalid depth region M_{idr} is obtained as follows:

$$M_{idr} = M_{der} AND M_{cer} \tag{5}$$

The color edge includes not only the object boundary but also texture edge in the object or background. Compared with the color edge, most of the depth edge is detected near the object boundary because the distance between the object boundary and background is large. The window size of depth edge should be larger than color one. By doing this operation, M_{idr} shown in Fig. 2 (b) includes around the boundary in color image. Note that the mask region of M_{idr} is replaced color information. Invalid depth map D_i is obtained using the raw depth map with M_{idr}.

The second step is filling the invalid depth. The invalid depth filling equation is the same as [11]. The estimated depth value $D_i(\boldsymbol{u})$ of invalid pixel $\boldsymbol{u}$ is calculated as follows:

$$D_i^t(\boldsymbol{u}) = \frac{1}{k(\boldsymbol{u})} \sum_{\boldsymbol{v}\in\Omega_s, D_i^{t-1}\notin 0} g_s(\boldsymbol{u}-\boldsymbol{v}) g_c(i(\boldsymbol{u})-i(\boldsymbol{v})) D_i^{t-1}(\boldsymbol{v}) \tag{6}$$

where g_s is spatial weight and g_c is the color similarity weight. They are Gaussian function and σ_s, σ_c are standard deviation respectively. Ω_s is a square window around an invalid pixel $\boldsymbol{u}$ and $k_(\boldsymbol{u})$ is a normalizing factor. $D_i(\boldsymbol{v})$ is a valid depth value of pixel $\boldsymbol{v}$. $\boldsymbol{u}-\boldsymbol{v}$ and $i(\boldsymbol{u})-i(\boldsymbol{v})$ represent Euclidean distance in image space and color space respectively. D_i^t is obtained by computing Eq. (6) t times. By repeating this process with updating D_i, we obtain the modified depth map D_m.

In order to obtain better segmentation results, the depth value from an invalid pixel in the object region should be estimated using the valid depth value in the same region. To satisfy this situation, we compute color segmented image using Kmeans clustering that is shown in Fig. 2 (c). After that, each pixel in the color segmented result has a label. The depth value of $\boldsymbol{u}$ is estimated using neighbor pixels which label is same with $\boldsymbol{u}$. The depth modified result is shown in Fig. 2 (d). After depth modification, we resegment the object region using modified depth map. Compared with segmented results applying depth threshold to raw depth and modified depth map shown in Fig. 2 (e) and (f), the object boundary

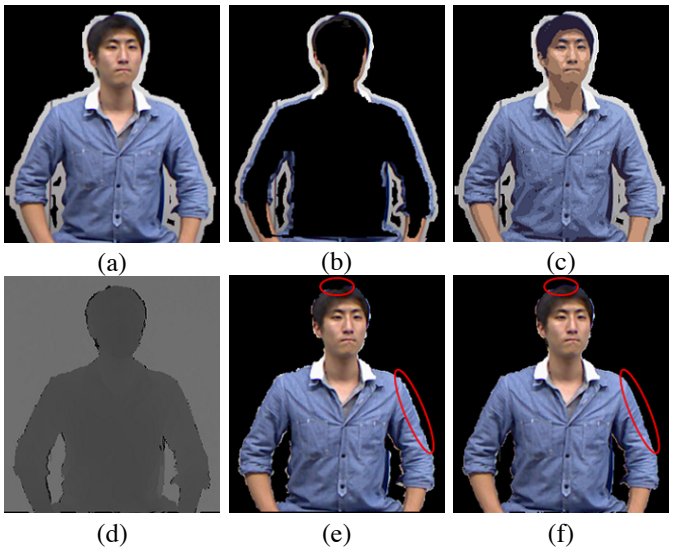


Fig. 2. Depth modification. (a)large segmented image, (b) invalid depth mask with color information, (c) Kmeans clustering image, (d) modified depth map, (e) and (f) segmented results by applying depth threshold to the raw depth map and the modified depth map respectively. The red circles show the area that which segmentation is improved notably.

is improved by using this process. Especially in red circle area, the human head and the line of left arm are correctly segmented. However, sometimes Kmeans clustering result has *minor label pixel*. It means that the label of interest pixel and neighbor pixels are different. In such a situation, if the *minor label pixel* is invalid depth, our method cannot estimate its depth value. This behavior depends on Kmeans cluster number and iteration number. In our implementation, we set the appropriate parameters about window size and them.

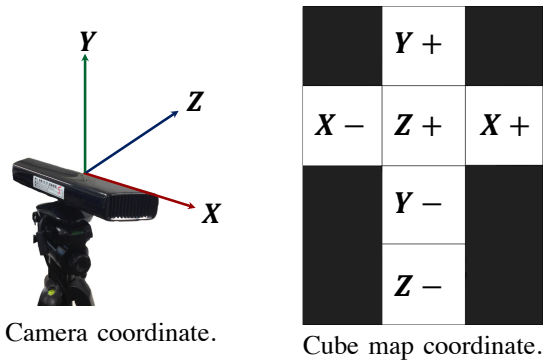
B. Normal Estimation from Modified Depth Map

After depth map modification and segmentation, we obtain ideal depth map only the object region. However, the depth map has still discrete value and contains noise. To reduce them and obtain smoothing depth map, we apply a bilateral filter [12] to the modified depth map before computing vertex map. The new depth map with reduced noise D_b as follows:

$$D_b(\boldsymbol{u}) = \frac{1}{k'(\boldsymbol{u})} \sum_{\boldsymbol{v}\in\Omega_{s'}} g_{s'}(\boldsymbol{u}-\boldsymbol{v}) g_d(D_m(\boldsymbol{u})-D_m(\boldsymbol{v})) D_m(\boldsymbol{v}) \tag{7}$$

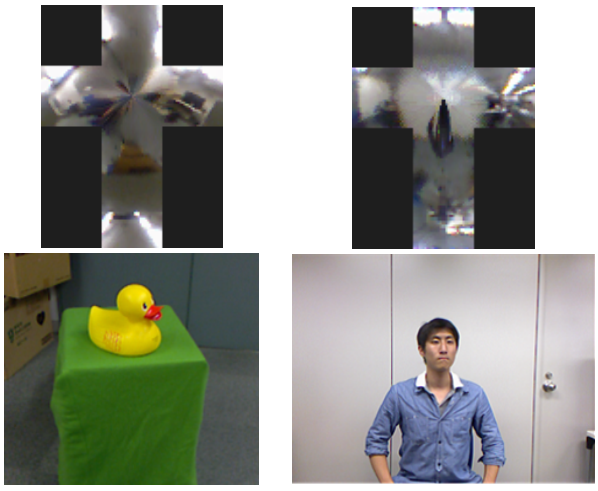
$k'(\boldsymbol{u})$ are normalizing factor, and g_d is the depth similarity weight. $g_{s'}$ and g_d are Gaussian function with $\sigma_{s'}$ and σ_d respectively. $\Omega_{s'}$ needs appropriate value for smoothing. If $\Omega_{s'}$ is too large, very smoothed depth map is obtained and the detail of normal information is lost. On the other hand, if $\Omega_{s'}$ is too small, the normal map is roughness shape.

Since depth map and the camera's intrinsic calibration parameter are known, it can be converted $D_b(\boldsymbol{u})$ into 3D vertex map $\boldsymbol{V}$ in the camera's coordinate space. After that, we estimate the normal map by applying the method proposed by Stefan *et al.* [6] to V. This method creates nine it integral images to compute the normal for specific point from the covariance matrix of its local neighborhood. The advantages of it are to obtain smooth normal map and fast computation time.



Camera coordinate. Cube map coordinate.

Fig. 3. Camera (Kinect) and cube map coordinate system.



Duck **Human**

Fig. 4. Objects using our experiments. The upper side is light environment and the bottom side is input color image.

However, this method cannot estimate around the boundary if it has a large difference of depth value. Therefore, we estimate the normal in such an invalid pixel $\boldsymbol{u}$ by using other simple method. It is computed by using adjacent pixel vertices as follows: $\boldsymbol{n}(\boldsymbol{u}) = (V(u+1,v) - V(u,v)) \times (V(u,v+1) - V(u,v))$ after normalization, we get normal map $N = \boldsymbol{n}/||\boldsymbol{n}||$. Combining the first and second methods, we get the object normal map completely. Then, the relighting result is obtained by computing Eq. (4) with normal vector and *Src* and *Dst* light environments.

IV. IMPLEMENTATION

In our experiments, we use Microsoft's Kinect as the RGB-D camera. Before the relighting, we captured the *Src* and *Dst* light probes [10] and *Dst* background. In this paper, we show the light environment image as a cube map [10]. The Camera and cube map coordinate are shown in Fig. 3. We used two objects **Duck** and **Human** which are static and dynamic shape respectively. Each *Src* light environment and captured input color image are shown in the upper side of Fig. 4. In case of **Duck**, we relight it to synthetic and dynamic light environment. Synthetic white light illuminates the object with moving its surroundings. From this experiment, we evaluate the relighting results depending on any light directions. On the other hand, in case of **Human**, we relight the moving object to real dynamic light environment captured in night road. In this environment, some cars come from behind the camera and light environment is dynamically changed. We evaluate whether the relighting results can correctly represent the brightness changes of light environment and synthesize *Dst* scene without discomfort. We prepared the three hundred image sequence for each *Dst* environment.

Implementation parameters of each object are shown in Table.I. The resolution of input is 640x480 image, and all processes are computed on a PC with Intel Core i7-3940XM 3.00GHz, 32.0GB memory and NVIDIA GeForce GTX 680M. In the next section, we discuss about each relighting result and processing time.

TABLE I. PARAMETERS ABOUT EXPERIMENT OF **Duck** AND **Human**

Processing	parameter	**Duck**	**Human**
Invalid depth mask	Window size of M_{der}	10×10	10×10
	Window size of M_{cer}	8×8	8×8
Invalid depth filling [11]	σ_s	1.4	1.4
	σ_c	1.2	1.2
	Ω_s	5×5	5×5
	t	40	40
Kmeans clustering	k	32	20
	Iteration number	20	20
Bilateral filter [12]	$\sigma_{s'}$	150	150
	σ_d	60	60
	$\Omega_{s'}$	9×9	13×13

V. EXPERIMENTAL RESULTS

A. *Result of* **Duck**

First we discuss about the results of **Duck**. In the *Src* light environment of **Duck** shown in the Fig. 4, two fluorescent lights are placed at the left and right sides of the ceiling. Therefore, the object shading is bright overall without the shadow observed in the neck and back of **Duck** and the green table. *Dst* light environments and the relighting results are shown in Fig. 5. This figure shows the four relighting results from the image sequence and the object normal map. In the 18^{th} frame, the white light comes from $z-$ axis (around camera direction) and the object shading is bright overall in relighting result. On the other hand, in the 28^{th} frame, 36^{th} frame and 81^{th} frame, white light comes from another direction. In the case of light comes from $x-$ axis (28^{th} frame), the left side of the object is bright and right side is dark. In the case of light comes from $z+$ axis (from object back side), the object shading is dark overall shown in 36^{th} frame. The fifth column of Fig. 5 shows the object normal map in color space. Smooth normal map is obtained by our estimation method. Thus, out relighting method can apply to arbitrary light directions. However, input color image of **Duck** shown in Fig. 4 can be seen not only a diffuse but also a specular reflection. We only consider the diffuse component and thus these results do not take into account the specular reflection and shadow. To obtain more accurate results, we need to consider the specular reflection model and the visibility between the object and the light positions.

B. *Result of* **Human**

Next, we discuss about the results of **Human**. *Src* light environment of **Human** is different from **Duck** and some fluorescent lights are placed at the the forward side of the ceiling

that is seen in the right column of Fig. 4. Four relighting results from the image sequence are shown in Fig. 6. Since the object shape and *Dst* light environment is dynamically changed, this figure shows each frame *Dst* light environment, color image, relighting result and normal map. For easier viewing detail of the results, the brightness and contrast are changed in light environments and relighting images. From these results, the object region is exactly segmented by using our proposed depth map modification. Thus, the object is naturally arranged on *Dst* background. Our method can be process in real time, we can estimate the object normal map for the arbitrary pose as shown in the bottom row of Fig. 6, Especially, the shape of hand and wrinkles and collars of clothes are estimated accurately. Each relighting results represent the shadings of these complex shape. In *Dst* environment of this case, some cars pass through from the back and right side of the camera to the forward side. When the car position is the back side, the object right side is bright shown in 1^{st} frame and 49^{th} frame because the car head light illuminates the object. It is seen at the $z-$ are of the cube map in these frame. After 110 frame, the car comes the camera neighborhood from the back side, the light environment is a little darker than previous frame. Compared with the light environments of 159^{th} frame and previous frame, the position of the car head light is changed from $z-$ area to $x+$ area in these cube maps. Thus, the dark shading region is increased because of change of the light source. More time passed and the car passed through the camera. Finally, the light environment is dark and the car head light disappeared as seen in cube map of 222^{th} frame. The human shading becomes close to almost black.

C. Computation time

Finally, we discuss about computation time of experimental results. Computation time of **Duck** and **Human** is shown in Table.II. This table shows depth map modification, segmentation, normal map estimation, SH rotation and relighting. Processing speed of **Duck** and **Human** is 8.92fps and 10.20fps. As seen in this table, most time taking part is depth map modification and normal map estimation. In our implementation, the online processing is computed by GPU or GPU+CPU. In process of depth map modification, we apply Kmeans clustering with GPU and CPU hybrid proposed by Sirotkovic *et al*. [8]. Thus, this process need to take many time. Additionally, computation time depends on cluster number. Because the cluster number of **Duck** is larger than **Human** (seen in Table.I), process time is longer in **Duck**. It is also possible to mention same thing in normal map estimation. This method need to compute the distance transform map mentioned in [6]. Since we compute it by only using CPU, this process takes a large time. Our method can be more faster, to change the algorithm or relighting flow without using CPU.

TABLE II. COMPUTATION TIME OF **Duck** AND **Human**

Processing	**Duck** (msec)	**Human** (msec)
Depth map modification	45.80	36.51
Segmentation	0.027	0.026
Normal map estimation [6]	42.72	42.69
SH rotation [7]	0.0065	0.0061
Relighting [1]	0.020	0.021

D. Limitation

From these two experiments, our relighting method can be apply for arbitrary shape objects and arbitrary light environments. However, we assume that surface reflectance is Lambertian model and only consider diffuse component. It is difficult to apply our method to specular object. To solve this situation, we need to estimate the object BRDF in real time. If the target object shape is complex such as a tree, it is also difficult to relight because obtaining its correct shape from Kinect is quite difficult.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed real time relighting with dynamic light environment using an RGB-D camera. The difficulty of relighting from the raw depth map is mainly caused by depth data. To solve this problem, we proposed the depth map modification around the object boundary for exact segmentation. After that we compute the normal map and relight to arbitrary light environments. In our experiments, we showed the relighting results for two objects which are static and dynamic shape. Our method is able to relight at 10fps and apply for dynamic objects and dynamic light environments. We discussed the accuracy of results, computation time and limitation of results. We think that our method can be more accurate to consider specular model and faster to change the algorithm or relighting flow.

REFERENCES

[1] Zhen Wen, Zicheng Liu and Thomas S Huang, "Face Relighting with Radiance Environment Maps", in CVPR, pp. 158–165, 2003.

[2] Oswald Aldrian and William A. P. Smith, "Inverse rendering with a morphable model: A multilinear approach", in BMVA, pp. 88.1–88.10, 2011

[3] Ravi Ramamoorthi, Pat Hanrahan, "An efficient representation for irradiance environment maps", in SIGGRAPH, pp. 497–500, 2001.

[4] Paul Debevec, Tim Hawkins, Chris Tchou and Haarm-Pieter Duiker, Westley Sarokin and Mark Sagar, "Acquiring the reflectance field of a human face", in SIGGRAPH, pp. 145–156, 2000.

[5] Andreas Wenger, Andrew Gardner, Chris Tchou, Jonas Unger, Tim Hawkins and Paul Debevec, "Performance relighting and reflectance transformation with time-multiplexed illumination" ACM Transactions on Graphics 24, 3, pp.756–764, 2005.

[6] Stefan Holzer, Radu Bogdan Rusu, M. Dixon, Suat Gedikli, and Nassir Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images", in IROS, pp. 2684–2689, 2012.

[7] Derek Nowrouzezahrai, Patricio Simari, Eugene Fiume, "Sparse Zonal Harmonic Factorization for Efficient SH Rotation", in ACM Transactions on Graphic, 31(3): 23, 2012.

[8] Jadran Sirotkovic, Hrvoje Dujmic, Vladan Papic, "K-Means Image Segmentation on Massively Parallel GPU Architecture", in MIPRO, pp. 489–494, 2012.

[9] Berkin Bilgic, Berthold K. P. Horn, Ichiro Masaki, "Efficient Integral Image Computation on the GPU", in IEEE Intelligent Vehicles Symposium, pp. 528–533, 2010.

[10] Paul Debevec, "Rendering synthetic objects into real scenes: bridging traditional and image-ased graphics with global illumination and high dynamic range photography", in SIGGRAPH, pp. 189–198, 1998.

[11] Li Chen, Hui Lin, Shutao Li, "Depth image enhancement for Kinect using region growing and bilateral filter", in ICPR, pp. 3070–3073, 2012.

[12] Carlo Tomasi, Roberto Manduchi, "Bilateral filtering for gray and color images", in ICCV, pp. 839–846, 1998.

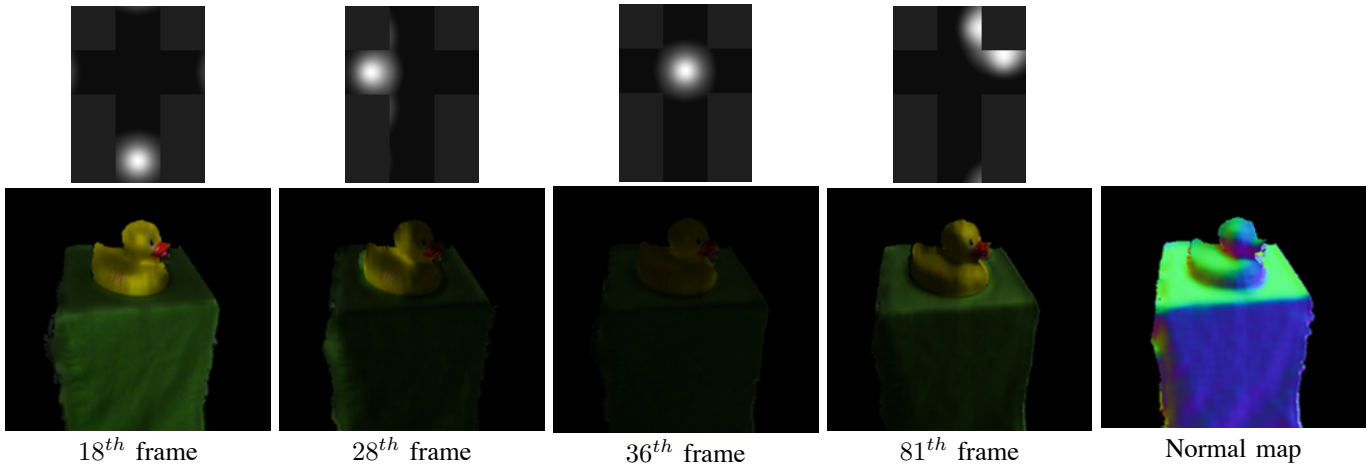


Fig. 5. Relighting results of **Duck**. From the first column to fourth column, the upper row is *Dst* light environment and the lower side is relighting result. Each caption shows frame number. The *Dst* light is synthetic white light and the direction of it is dynamically changed. The fifth column shows the the object normal map in color space.

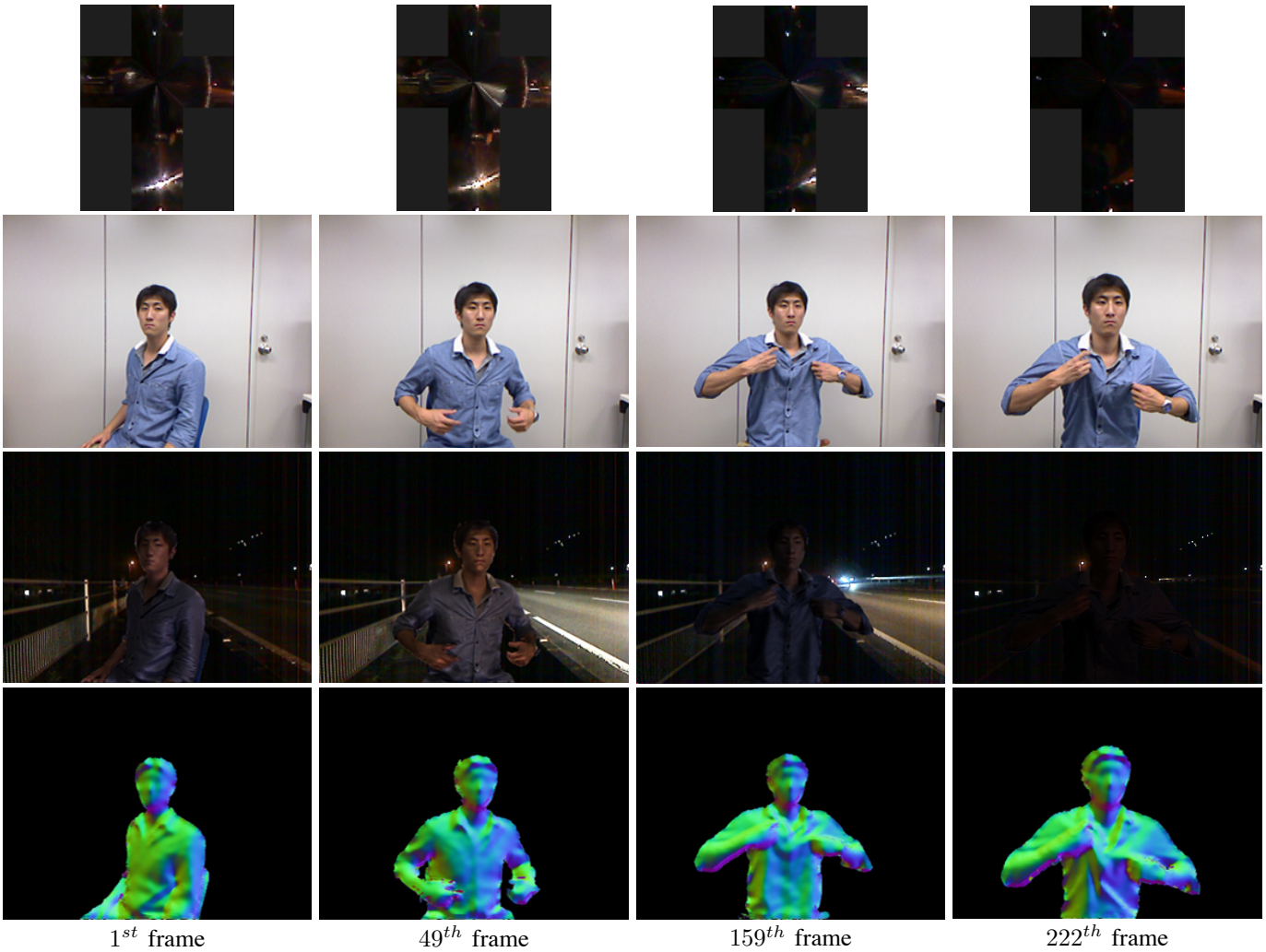


Fig. 6. Relighting results of **Human**. From first row to the lower row show captured light environment, input color image, relighting result and normal map. Each caption shows frame number.The *Dst* light environment and back ground are captured in night road. Light environment is changed by the car's head light which comes from behind the camera.