Capturing Multiple Illumination Conditions using Time and Color Multiplexing

Bert De Decker [1] bert.dedecker@uhasselt.be
Jan Kautz [2] j.kautz@cs.ucl.ac.uk
Tom Mertens [1] tom.mertens@uhasselt.be
Philippe Bekaert [1] philippe.bekaert@uhasselt.be

[1] Hasselt University - Expertise Centre for Digital Media - Transnationale Universiteit Limburg, Belgium *
[2] University College London, UK

Abstract

Many vision and graphics problems such as relighting, structured light scanning and photometric stereo, need images of a scene under a number of different illumination conditions. It is typically assumed that the scene is static. To extend such methods to dynamic scenes, dense optical flow can be used to register adjacent frames. This registration becomes inaccurate if the frame rate is too low with respect to the degree of movement in the scenes.

We present a general method that extends time multiplexing with color multiplexing in order to better handle dynamic scenes. Our method allows for packing more illumination information into a single frame, thereby reducing the number of required frames over which optical flow must be computed. Moreover, color-multiplexed frames lend themselves better to reliably computing optical flow. We show that our method produces better results compared to time-multiplexing alone. We demonstrate its application to relighting, structured light scanning and photometric stereo in dynamic scenes.

1. Introduction

There is a wide variety of algorithms that need images of a scene under a number of different lighting conditions. Structured light scanning computes a depth map by illuminating the scene with different coded patterns [18]. Photometric stereo [2, 23] obtains a per pixel normal of a scene by capturing the scene under a number of different point light source positions. In computer graphics, image-based relighting is performed by recording a scene under a number of basis illumination patterns [7].

In all of these methods, each illumination condition requires a separate image, and it is assumed the scene remains static. Specific methods have been developed in order to deal with dynamic scenes [25, 26, 6, 9, 22]. In this paper, we seek a general technique which is applicable to all methods requiring multiple illumination conditions.

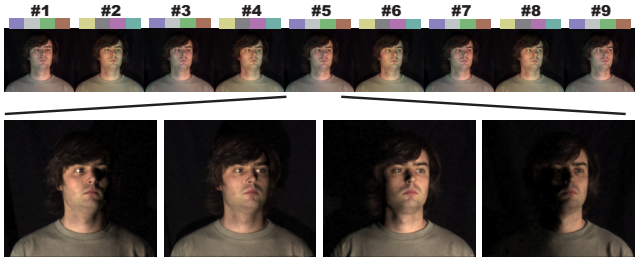


Figure 1. Demultiplexing illumination using our method. Top row: 9 input frames of a dynamic scene. A person is lit by only **2** different colored illumination patterns using 4 differently colored light sources (indicated by the squares). Bottom row: The output images show the demultiplexed lighting. I.e., each image contains the result of having illuminated the scene by each source separately.

An obvious way to extend the aforementioned methods to handle dynamic scenes, is to use a dense optical flow to register dynamic content between frames. Even for high framerates, this registration is required. This registration becomes inaccurate if scene elements move too fast, or equivalently, if the frame rate is too low. Also, as more illumination conditions are needed, optical flow must be computed over a longer span of frames. The results will therefore be more prone to occlusion errors.

In this paper, we propose to extend time multiplexing with color multiplexing. With color multiplexing we can encode more illumination conditions per frame, which reduces the number of required input images over which optical flow must be computed. This relaxes the requirement of having a sufficiently high frame rate, making it possible for using (cheaper) cameras with a lower frame rate. Estimating optical flow under varying illumination conditions is hard. Even though state-of-the art optical flow algorithms can deal with varying illumination, the aforementioned methods typically employ significantly different illumination patterns from frame to frame. This causes wildly varying intensities and even image features (e.g., due to

*http://research.edm.uhasselt.be/bdedecker/

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

moving shadow boundaries). An additional benefit of color-multiplexing is that optical flow can be computed more reliably, because all light sources are enabled in every frame and only their color varies.

Figure 1 shows a typical use of our light multiplexing approach in the context of relighting.

Our contributions are as follows. (1) We demonstrate the use of both color and time multiplexing to capture a scene under a number of lighting conditions; (2) we show how we can compensate for the motion in a dynamic scene. (3) We demonstrate that our light multiplexing scheme is generic and can be used to improve many existing algorithms.

2. Related Work

There is a wide range of techniques that need images of a scene under varying lighting conditions. In this section we will give a short overview of these algorithms. Many of them only use time multiplexing to obtain the needed information. Using the algorithm described in this paper, it becomes possible to alter these techniques so they use both time and color multiplexing.

Relighting In relighting applications, the goal is to change the lighting of a scene after it is captured [22]. Debevec et al. [7] demonstrate a first system that is able to relight static scenes. Images of a scene are captured under a large number of lighting conditions, and by blending the images, the scene can be relit. This system is extended to dynamic scenes by Wenger et al. [22]. High-speed cameras and LEDs are used to capture the scene under different lighting conditions at a very high framerate. They can handle dynamic scenes by compensating for the motion using the optical flow between tracking frames. These are special frames for which the scene is uniformly lit with white light. Another extension of this idea is presented by Wang et al. [21]. Here, a person captured under ambient light and time multiplexed IR light patterns can be relit. To this end, information from the IR domain is transferred to the visible domain. All of the above techniques only use time multiplexing to obtain the scene under a number of lighting conditions, in this paper we present a technique to use both time and color multiplexing to obtain this information.

Hsu et al. [11] can relight a scene from a single image using their white balancing technique. They are able to change the colors of the light sources in the scene under the assumption that there are only two light sources present in the scene. Our method can work with any number of light sources.

Photometric Stereo Photometric stereo techniques capture a per pixel normal for a scene lit by a number of point light sources [23, 2]. Woodham et al. [23] proposed to use color multiplexing to obtain the required information, but they did not show in practice how to do this for dynamic scenes. Hernandez et al. [9] also perform photometric stereo with colored lights, but assume the materials of all the objects in the scene are the same, while we do not constrain the number of materials in the scene. They are able to demultiplex three light sources with different colors since the materials in the scene are known. Due to the use of only three light sources, the photometric stereo problem becomes underdetermined in shadow regions. This problem was addressed by Hernandez et al. [10].

There are a number of algorithms that combine ideas from structure from motion, photometric stereo and multi-view stereo [24, 13, 12] to obtain the shape of a moving object under static illumination. These methods assume a single rigid object, while our method can handle deforming objects. Our ability to deal with non-rigid scenes will be demonstrated in the results section.

Depth from Structured Light Depth from structured light techniques are able to obtain a dense depth map of a scene by illuminating it by a number of well chosen black and white patterns [18, 17, 4, 25, 5, 14]. Spacetime Stereo [26, 6] in combination with structured light can be used to obtain a depth map for dynamic scenes. Caspi et al. [4] project a number of colored stripe patterns onto a static scene, and for each pattern an image is taken. This way each pixel is given a unique label over time and from this a depth map can be calculated. Zhang et al. [25] also project one or more colored stripe patterns onto the scene, but they use the color transitions between stripes to label the space. Hall-Holt et al. [8] also use color transitions and are able to obtain a depth map of dynamic scenes. Some problems arise when using colored stripes to obtain a depth map from a scene. The color of the objects in the scene interfere with the color of the stripes and artifacts occur near depth discontinuities. These problems are addressed by Lee et al. [14]. Chen et al. [5] use colored structured light patterns to diminish the ambiguity when calculating depth from stereo. In contrast to previous methods for dynamic scenes, when using our method, scene colors do not interfere in the process and we can capture the scene colors in addition to the depth map.

General Light Multiplexing Schechner et al. [20, 19] present a general technique to multiplex light without the use of color multiplexing. The lights are turned on/off using Hadamard-based patterns. For each frame they turn on multiple lights at once. This way they can capture low intensity lighting, without the need for long exposure times. Also the technique of Wenger et al. [22] can be seen as a general light multiplexing technique. They compare multiple light bases, among which the Hadamard-based patterns.

Discussion While there are specific extensions to some of the basic structured light and photometric stereo algorithms that use both color and time multiplexed light [23, 14, 5], these extensions are not as general and have their own advantages and disadvantages. Existing photo-

metric stereo algorithms [23, 9, 10] suppose the scene is static or that all the objects in the scene have the same albedo. Our method on the other hand is able to handle dynamic scenes without constraining the number of albedos. When our method is applied to depth from structured light, we obtain the depth map and the scene colors simultaneously. Existing depth from structured light algorithms for dynamic scenes are unable to do this, they only obtain a depth map [4, 25, 8, 14, 5].

3. Time and Color Multiplexing

In this section, we present our light multiplexing algorithm for static scenes. The extension to dynamic scenes will be described in the next section 4.

3.1. Model

The aforementioned applications all require that an object be captured under varying illumination. We address the common scenario in which illumination is varied by changing the color of otherwise fixed light sources (Figure 2).

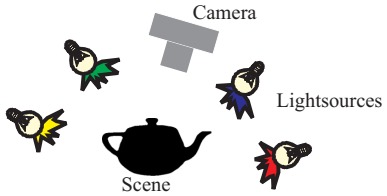


Figure 2. Our setup consists of the scene to be captured, filmed by one camera and lit by a number of light sources with changing colors.

We make the following assumptions:

- Only the color of the light sources varies from captured frame to frame. Position and directional emission distribution remain constant during the capture.
- Surfaces are Lambertian;
- Indirect illumination is negligible;
- Negligible motion blur;
- Object motion relative to light sources affects only reflected light intensity, not its color.

It will be shown in the results section, that our time and color multiplexing scheme still yields acceptable results, even if these assumptions are violated to some (practical) extent.

For now we also assume that both the camera and the objects do not move during the capture process. In the next section we will show how we can drop these additional assumptions to handle dynamic scenes.

Now we will describe our image formation model for a single light source, later we will show how this is generalized to multiple light sources. If only a single light source illuminates the object, the RGB color $\mathbf{C}^i = (C^i_r, C^i_g, C^i_b)$ of a pixel in any captured image $i, i = 1 \dots n$ with n the number of illumination scenarios, will be the product

$$\mathbf{C}^i = \mathbf{A} \odot \mathbf{L}^i I \tag{1}$$

of three factors:

- The RGB material color $\mathbf{A} = (A_r, A_g, A_b)$ of the object seen in the pixel. $\mathbf{A}$ is independent of illumination;
- The RGB light source color $\mathbf{L}^i = (L^i_r, L^i_g, L^i_b)$ in frame i;
- A scalar form factor I that indicates the intensity of the reflected light. I depends on object and light source relative position, orientation and visibility. Since neither move (in this section), it is constant. Note, it is allowed to vary in the next section.

The symbol $\odot$ stands for component-wise multiplication of RGB color vectors. For clarity, we do not index pixels: We use the same symbol $\mathbf{C}$ for the color of any single pixel and for a complete RGB image. We use the same symbol I for the form factor of a single pixel, and for the set of form factors corresponding to all pixels in an image. We will refer to the latter as "intensity images" in the following.

Since only the product result counts, any of these factors can be scaled to wish, provided the scaling is compensated in the other factors. In our multiplexing scheme, we normalize material colors $\mathbf{A}$ such that the sum of square components is unity. Our choice of Euclidean metric is convenient, but otherwise arbitrary: any normalization metric would do. This normalization is compensated in the intensity value I.

When m light sources $j, j = 1 \dots m$, with colors $\mathbf{L}^i_j$ are used, pixel colors $\mathbf{C}^i$ will satisfy the following matrix equation:

$$\mathbf{C} = \mathbf{M}\,\mathbf{I}, \text{ with} \tag{2}$$

$$\begin{bmatrix} C^1_r \\ C^1_g \\ C^1_b \\ C^2_r \\ C^2_g \\ \vdots \\ C^n_b \end{bmatrix} = \begin{bmatrix} A_r L^1_{1r} & A_r L^1_{2r} & \dots A_r L^1_{mr} \\ A_g L^1_{1g} & A_g L^1_{2g} & \dots A_g L^1_{mg} \\ A_b L^1_{1b} & A_b L^1_{2b} & \dots A_b L^1_{mb} \\ A_r L^2_{1r} & A_r L^2_{2r} & \dots A_r L^2_{mr} \\ A_g L^2_{1g} & A_g L^2_{2g} & \dots A_g L^2_{mg} \\ \vdots & \vdots & \vdots \\ A_b L^n_{1b} & A_b L^n_{2b} & \dots A_b L^n_{mb} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_m \end{bmatrix}$$

3.2. Algorithm outline

Capturing an object under n illumination scenarios $i, i = 1 \dots n$, realized with m light sources $j, j = 1 \dots m$, as explained above, comes down to measuring $\mathbf{C}^i$ as a function of $\mathbf{L}^i_j$. If the normalized material colors $\mathbf{A}$ are known, and provided the rank of $\mathbf{M}$ is greater or equal than the number of light sources m, the intensity image I_j of each

light source j can be computed by solving the linear system of equations (2). Since this system is in general overdetermined, least squares techniques are used. The normalized material colors are usually not a priori known, however. We explain in 3.3 how to obtain them.

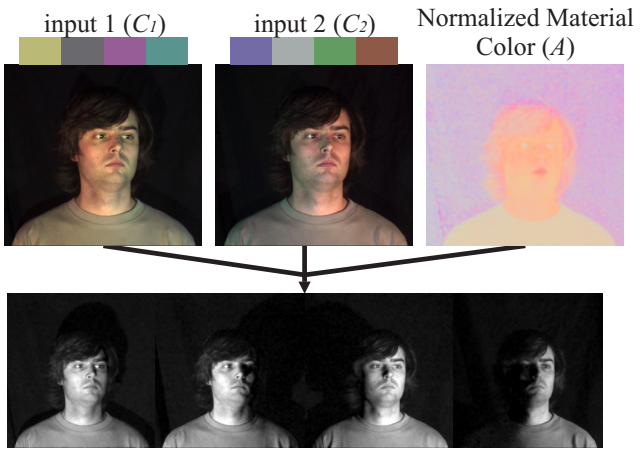


Figure 3. Given images $\mathbf{C}^1$ and $\mathbf{C}^2$ of an object captured under varying illumination, and assuming the normalized material color image $\mathbf{A}$ is known, our algorithm calculates intensity images I_j for each light source $j = 1 \ldots 4$. The colored squares on top of the input images indicate the (user chosen) color of the (otherwise fixed) light sources during capture.

Example Application: Relighting Given the normalized material colors $\mathbf{A}$ and the intensity images I_j for all light sources j, a relighted image C' corresponding to freely chosen light source colors $\mathbf{L}'_j$ can be obtained directly from (2) as

$$\mathbf{C}' = \sum_{j=1}^{m} \mathbf{A} \odot \mathbf{L}'_j I_j. \tag{3}$$

Selection of Light Source Colors The light source colors $\mathbf{L}$ are chosen as to ensure good conditioning of the system (2). We enumerate a number of candidate light source colors using a simple grid search. For each candidate we calculate the condition number of the matrix M (see equation (2)). The candidate with the smallest condition number is selected. Note that the ideal light source colors depend on the normalized material colors $\mathbf{A}$. In our examples, we assume the normalized material colors to be white when calculating the light source colors. Better light source colors could be obtained by using the normalized material colors of the previous frame.

3.3. Acquisition of Normalized Material Colors

The normalized material color image $\mathbf{A}$ corresponds to the normalized RGB image that would be captured if all light sources are set to unit strength white color. Colors are normalized by scaling them such that the sum of squared components is unity in our implementation, but any other normalization metric would do as well. We now present two ways to obtain $\mathbf{A}$.

Dedicated Image Method One way to obtain the normalized material colors, is to actually capture, and normalize, an image $\mathbf{W}$ with all light source colors set to full intensity white. For instance, given $m = 3$ coloured light sources, a single image ($n = 1$) yields $\mathbf{M}$ as a 3×3 matrix, and it is trivial to see that it is possible to choose the colors of our light sources in such a way that this matrix becomes rank 3 as needed to solve (2). The disadvantage of this approach is the need to capture an extra image only to calculate $\mathbf{A}$.

Complementary Colors Method Another way of calculating the normalized material color is by choosing the colors of each light source for each frame in such a way that these colors add up to white. So for example if we only take 2 images, the color of one light source for the first frame is the complement of the color of this light source for the second frame. Or in general $\forall j$: $\sum_i (L^i_{jr}, L^i_{jg}, L^i_{jb}) = (1, 1, 1)$ Normalizing the average of these images also yields $\mathbf{A}$.

Given n captured images, we show in the appendix that is it possible to demultiplex $m = (3n - 2)$ light sources. The complementary colors method allows to demultiplex one light source more than the *Dedicated Image Method*, at the cost of some extra noise.

3.3.1 Dealing with Sensor Noise

By using color multiplexing, each pixel of our input images contains more information than when using only time multiplexing. So our algorithm trades color depth for fewer input frames. When using a normal digital video camera, the input images are only low dynamic range images and also contain some noise. If we used our input images directly, the noise in the input images would be amplified, which would result in noisy output images as shown in Figure 4a.

(a) Without noise reduction (b) Post-process bilateral blur (c) Our noise reduction method

Figure 4. Noise reduction is needed to generate high quality results. Blurring the noisy result during a post processing step generates less good results than our noise reduction method.

In order to reduce noise, we can blur the input images when calculating the intensity images $\mathbf{I}$; or in other words, we trade resolution for dynamic range. The disadvantage of

this approach is that we remove the high frequency components of the image. However, it is possible to add these high frequency components back to the final result as follows:

$$I_{\text{res}} = I_{\text{blur}} W/W_{\text{blur}}, \quad (4)$$

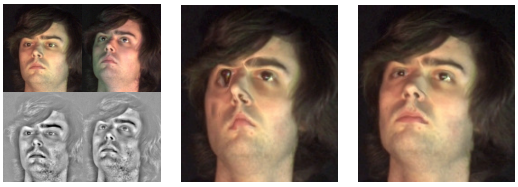
where I_{blur} is the blurred light intensity image using a usual Gaussian blur kernel, I_{res} is the resulting light intensity image with the high frequencies put back in, W is the lightness of the RGB image $\mathbf{W}$ of the object under white light, introduced in the previous section. W_{blur} is W blurred with the same kernel as I_{blur}. The main approximation here, is that the high frequencies of W, which contains little noise as it is fully lit (W is derived from $\mathbf{W}$), should also be present in the resulting intensity image I_{res}. Multiplication with W/W_{blur} puts some of these frequencies back into I_{blur}.

This way at least the high frequency components of the texture of the input images are preserved, even though the high frequency components of the lighting are lost. As a consequence we cannot reproduce very hard shadows. We demonstrate in the results section that this effect is not very significant.

4. Dynamic Scenes

So far we have shown how the time and color multiplexing works in the case of a static scene. In this section, we show how we can extend this method to dynamic scenes.

In dynamic scenes, the objects in the scene will have moved in two consecutive images and we cannot directly apply our proposed algorithm. To compensate for this motion, we calculate the optical flow between the images and warp the images to a reference image, for which we use the optical flow algorithm by Brox et al. [3] with improvements by Sand et al. [16].



(a) (b) (c)

Figure 5. Calculating the optical flow *with* and *without* removing lighting from the image. (a) Input images and filtered input images. (b) Result when calculating the optical flow from the input images directly. (c) Result when calculating the optical flow from the filtered input images. When comparing the left eye and the nose in images b and c, it is apparent that calculating the optical flow from the input images directly does not work very well.

Since the colors of the light sources are not the same for each image, naïvely applying optical flow on the images does not work very well as shown in Figure 5. Instead we first apply a filter to the images that removes the lighting, but keeps the texture [15]. For each pixel we calculate the local brightness and the local contrast of a small neighborhood, and the filtered pixel value is the original pixel value subtracted by the local brightness and divided by the local contrast. We perform the optical flow on these filtered images. We also choose the color of the light sources in such a way that no dark shadows are created in any of the color channels. By doing this, all the texture information of the scene is visible in every image, and nothing gets fully obscured by shadows. From our experiments, we found that the use of the filter does not degrade the quality of the optical flow, even when we calculate the optical flow between two images that are lit in the same way.

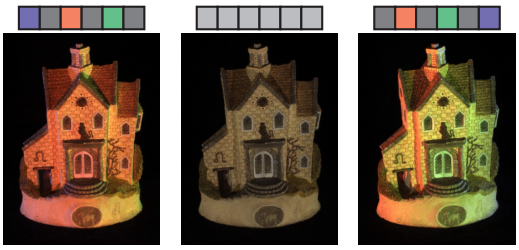
If we want to demultiplex the light for a given frame, we warp the required number of nearby frames to this frame. This way, we can assume we have a static scene for the other steps of the algorithm.

5. Results

In our experimental setup we use projectors as light sources as they are easy to control and it is possible to switch their colors rapidly. In a practical setup, using LEDs as lightsources might be a better option, since they are cheap and can switch on and off rapidly. PointGrey flea and grasshopper cameras have been used to capture video sequences at 20 fps.

Computation times are in the order of 25 minutes for the calculation of the optical flow between two images and approximately 5 minutes for the rest of the calculations using our Matlab implementation on a 3 GHz CPU.

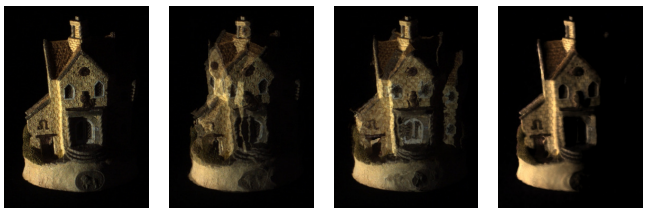
Relighting We first apply our method to relighting, see Figure 6. The scene consists of a house on a rotating platform and it is lit by six light sources. The house rotates 5 degrees between two consecutive images. We used stop motion to capture this sequence in order to capture additional ground truth images and to compare our results with those of Wenger at al. [22]. From this large set of captured frames, we generate image sequences that are given to the algorithms we are comparing. To these algorithms, the input looks as if it was captured in real time using a video camera. Images (a)-(c) in Figure 6 show the input images for our algorithm. The squares on top of each input image depict the colors of each of the six light sources for the frame. Images (d)-(f) show the output of our algorithm, the scene is shown under a number of new lighting conditions. In this example we used the *Dedicated Image Method* to obtain the normalized material color A, for which we use image (b). To demultiplex the illumination, we run our algorithm twice. First we demultiplex the three lightsources that were not grey in image (a). The image (a) is influenced by all 6 light sources, the 3 colored ones and the 3 grey ones. We remove the influence of the three grey lightsources by subtracting a scaled version of (b) from it. The resulting image



(a) Input (b) Input (c) Input

(d) Output (e) Output (f) Output

Figure 6. Top row: input images of a dynamic (rotating) scene lit by six light sources with changing colors, the squares on top of each image depict the colors of each of the six light sources for this frame. Bottom row: the output images show the scene lit under new lighting conditions.



(a) Reference (b) Wenger et al. (c) Wenger et al. (d) Our method

Figure 7. From left to right: ground truth, output of Wenger et al. with a tracking frame every 7 frames, output of Wenger et al. with a tracking frame every 3 frames, and our result.

now only depends on the three colored lightsources we are trying to demultiplex. Using this image and the normalized material color A we are now able to demultiplex the three lightsources. Since we need to solve for 3 lightsources, $\mathbf{M}$ is a full rank 3×3 matrix, so equation (2) is easy to solve. To demultiplex the other three lightsources, the ones that are not grey in image (c), we use a similar approach. We also could have demultiplexed the six lightsources at once, but by doing it this way, the results are more robust against errors in the optical flow. To relight the scene, we use all six light sources. In the accompanying video, some small artifacts are visible, this is because of occlusion problems when calculating the optical flow.

In Figure 7 we show a comparison between our method and the relighting method of Wenger et al. [22]. Note that the method of Wenger et al. is a high quality relighting system that generates very good results but only uses time multiplexing. I.e., one light source is turned on in each frame

(a) Ground truth (b) Our result

Figure 8. Comparing ground truth and our method when relighting a static scene. Our approach generates convincing results, even though the assumption that surfaces are Lambertian, is slightly broken. The results might not be physically correct, yet they are very convincing.

and optical flow is used to merge the information from several frames. In this particular example we push their method and our algorithm to the limit by running it on a scene that has a lot of movement between two successive frames. Image (a) shows the ground truth, only one of the light sources is turned on in this image; (b) and (c) show the same image, but obtained by the method of Wenger et al.; (d) shows the same image obtained by our method. To deal with dynamic scenes, Wenger et al. inserts tracking frames. These are frames for which the scene is uniformly lit with white light. For image (b) a tracking frame was inserted every seven and for (c) every three frames. In (b) it is clear that the optical flow breaks down. This is because the house rotated 35 degrees between two consecutive tracking frames, which results in occlusion problems. For very fast moving scenes our algorithm suffers from the same problem. In (c) the house rotates only 15 degrees between two tracking frames, so the optical flow works quite well, but the lighting information still needs to be warped over a distance of 8 frames (6 light sources and 2 tracking frames). So shadows will not move over the object as they should. Our algorithm has the same problem, but since the lighting information only has to be warped over 2 frames, this problem is far less pronounced. These artifacts are most visible when comparing the door of the house.

Since we use the three color channels, one might expect that our system has a speedup of 3 compared to the method of Wenger et al. [22], but it is more nuanced than that. For example, if we have 10 lights, we need 4 frames to capture all the information. Wenger et al. needs 15 frames, if a tracking frame has to be inserted every 2 lighting frames. In this case, our method has a speedup of 3.75.

Figure 8 shows that our algorithm delivers convincing results, even when scene assumptions are violated slightly. Artefacts do appear if the assumptions are violated more severly as can be seen in figure 9. A scene with severe color bleeding and non-lambertian objects is depticed. Our method also breaks down when applied to a scene with considerable motion blur.

(a) Input (b) Ground truth (c) Our result

Figure 9. A test case where our scene assumptions are violated. False color bleeding is visible (e.g., on the wall) and artifacts appear near specularities.

(a) Input: three MSB (b) Input: material colors (c) Input: three LSB

(d) Extracted structured light patterns

(e) Extracted depth map and novel viewpoints

Figure 10. Depth from structured light using time and color multiplexing.

Depth from Structured Light In Figure 10 we calculate depth from structured light [18, 17] using time and color multiplexing. To acquire a depth map with 6-bit precision, we project 6 structured light patterns. Image (a) shows the scene lit by three light sources, the red one is the most significant structured light pattern, the green one the second and the blue one the third most significant pattern. Image (b) is used to obtain the normalized material color A. And image (c) shows the scene lit by the three least significant patterns. Extracted structured light patterns are shown in (d) and the resulting depth map along with images of the scene under novel viewpoints is shown in (e). In contrast to previous methods for dynamic scenes [4, 25, 8, 14, 5], scene colors do not interfere in the process and we can capture the scene colors and depth map simultaneously.

Photometric Stereo In figure 11 we apply our light multiplexing method to photometric stereo [2, 23]. (a) shows two input images of a scene lit by three light sources. (b) shows the demultiplexed lights, these images are the input of the photometric stereo algorithm. (c) shows the calculated normal map of the scene. Normals can then be used for instance to add specular highlights using an environment map, as seen in (d). (e) shows some more results where

(a) Input (b) Demultiplexed lights

(c) Normal map (d) Output (e) Specularities (f) 3D Surface

Figure 11. A normal map can be calculated using photometric stereo, which allows us to add additional specular highlights or compute a 3D surface.

specularities were added to a diffuse scene. (f) shows the 3D surface computed from the normal maps of the scenes depicted in (e). To obtain this 3D surface we used the method by Agrawal et al. [1].

Previous photometric stereo algorithms for dynamic scenes are restricted to scenes in which all objects have the same albedo [9, 10] or to scenes containing a single rigid object [24, 13, 12]. As shown in Figure 11, our method handles multi-coloured non-rigid objects. Apart from the normal map, our method also recovers material colors.

6. Conclusion and Future Work

We proposed a generic approach for capturing dynamic scenes under both color and time multiplexed varying illumination. Our approach is well suited for a broad range of applications, and was demonstrated in the context of relighting, structured light depth capturing and photometric stereo. Compared to previous light multiplexing methods, our method allows to extract more information from fewer images. In contrast to previous photometric stereo methods, our technique is able to handle multi-coloured dynamic scenes. Applied to structured light scanning, our method allows to simultaneously acquire depth and scene colors.

At present, the computation time with our implementation is rather high. We believe there is plenty of room for improvement, perhaps up to near real time speed.

Some avenues for future research include the noise filtering from section 3.3.1, investigating the relation between noise in the input images and noise in the output images. The core idea of this paper may also be applicable to other multiplexing techniques.

7. Acknowledgements

We would like to thank Eugene Hsu and Frédo Durand for their input. The authors acknowledge financial support on a structural basis from the Flemish institute for broadband communication (IBBT) and impulse financing from the transnationale Universiteit Limburg.

Appendix: Max. Number of Light Sources

When we use 2 images and the *Complementary Colors Method*, we can only demultiplex 4 light sources, even though **M** in equation (2) is a 6×6-matrix. In this case the rank of **M** is only 4 instead of 6 for the following reason. For simplicity assume that $A = (1, 1, 1)$. Define V_1 the first row of **M**, V_2 the second row and V_3 the third row. Because the colors of the first frame are the complement of the colors of the second frame, we have that the fourth row is $V_4 = k - V_1$, with k some number. The fifth row is $V_5 = k - V_2$, and the sixth row is $V_6 = k - V_3$. Now the fifth row is a linear combination of the other rows: $V_5 = k - V_2 = (k - V_1) + (V_1) - (V_2) = V_4 + V_1 - V_2$. And in the same way is V_6 also a linear combination of the first 4 rows, so the matrix **M** is rank 4.

In general, if we have m frames, **M** is rank $(3 * m - 2)$. Or if we have n light sources, we need $(n + 2)/3$ images.

References

[1] A. Agrawal, R. Raskar, and R. Chellappa. What is the range of surface reconstructions from a gradient field ? In *European Conference on Computer Vision (ECCV)*, 2006.

[2] S. Barsky and M. Petrou. The 4-source photometric stereo technique for three-dimensional surfaces in the presence of highlights and shadows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1239–1252, 2003.

[3] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In T. Pajdla and J. Matas, editors, *European Conference on Computer Vision*, volume 3024, pages 25–36, May 2004.

[4] D. Caspi, N. Kiryati, and J. Shamir. Range imaging with adaptive color structured light. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):470–480, 1998.

[5] C.-S. Chen, Y.-P. Hung, C.-C. Chiang, and J.-L. Wu. Range data acquisition using color structured lighting and stereo vision. *Image and Vision Computing*, 15(6):445–456, 1997.

[6] J. Davis, R. Ramamoorthi, and S. Rusinkiewicz. Spacetime stereo: A unifying framework for depth from triangulation. In *CVPR*, pages II: 359–366, 2003.

[7] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In *SIGGRAPH '00*, pages 145–156, 2000.

[8] O. Hall-Holt and S. Rusinkiewicz. Stripe boundary codes for real-time structured-light range scanning of moving objects. In *IEEE International Conference on Computer Vision*, pages II: 359–366, 2001.

[9] C. Hernandez, G. Vogiatzis, G. J. Brostow, B. Stenger, and R. Cipolla. Non-rigid photometric stereo with colored lights. In *IEEE Int. Conf. on Computer Vision*, pages 1–8, 2007.

[10] C. Hernandez, G. Vogiatzis, and R. Cipolla. Shadows in three-source photometric stereo. In *European Conference on Computer Vision*, volume 5302, pages 290–303, 2008.

[11] E. Hsu, T. Mertens, S. Paris, S. Avidan, and F. Durand. Light Mixture Estimation for Spatially Varying White Balance. *ACM Trans. Graph.*, 27(3):70:1–70:7, 2008.

[12] N. Joshi and D. Kriegman. Shape from varying illumination and viewpoint. In *ICCV*, pages 1–7. IEEE, 2007.

[13] A. Lakdawalla and A. Hertzmann. Shape from video: Dense shape, texture, motion and lighting from monocular image streams. In *IEEE Workshop on Photometric Analysis for Computer Vision, in conjunction with ICCV*, 2007.

[14] K. Lee, C. Je, and S. Lee. Color-stripe structured light robust to surface color and discontinuity. In *Asian Conference on Computer Vision*, pages II: 507–516, 2007.

[15] P. Sand and S. Teller. Video matching. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):592–599, 2004.

[16] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2195–2202, 2006.

[17] K. Sato and S. Inokuchi. Three-dimensional surface measurement by space encoding range imaging. *Journal of Robotic Systems*, 2:27–39, 1985.

[18] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *IEEE International Conference on Computer Vision*, pages 195–202, 2003.

[19] Y. Y. Schechner, S. K. Nayar, and P. N. Belhumeur. A theory of multiplexed illumination. In *IEEE International Conference on Computer Vision*, pages 808–815, 2003.

[20] Y. Y. Schechner, S. K. Nayar, and P. N. Belhumeur. Multiplexing for optimal lighting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(8):1339–1354, 2007.

[21] O. Wang, J. Davis, E. Chuang, I. Rickard, K. de Mesa, and C. Dave. Video relighting using infrared illumination. *Computer Graphics Forum (Proc. Eurogr.)*, 27(2):271–279, 2008.

[22] A. Wenger, A. Gardner, C. Tchou, J. Unger, T. Hawkins, and P. Debevec. Performance relighting and reflectance transformation with time-multiplexed illumination. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24(3):756–764, 2005.

[23] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[24] L. Zhang, B. Curless, A. Hertzmann, and S. M. Seitz. Shape and motion under varying illumination: Unifying structure from motion, photometric stereo, and multi-view stereo. In *The 9th IEEE International Conference on Computer Vision*, pages 618–625, Oct. 2003.

[25] L. Zhang, B. Curless, and S. M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *IEEE Int. Symposium on 3D Data Processing, Visualization, and Transmission*, pages 24–36, June 2002.

[26] L. Zhang, B. Curless, and S. M. Seitz. Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 367–374, June 2003.