Vol.2015-CG-161 No.13
Vol.2015-CVIM-199 No.13
2015/11/6

Evaluation of Shadow Cast by Estimated Pseudo 3D Shapes for Synthesizing Realistic Shadows

Jung-Hsuan Wu[1,a)] Suguru Saito[2,1,b)]

Abstract: Shadows play an important role in the human perception of 3D geometry in photographs, and properly synthesized shadows produce a realistic rendering result. Shadows, especially hard shadows, define powerful constraints between a light source, occluders, and the surfaces onto which shadows are cast. In our previous work, we had proposed a method to create a pseudo 3D scene in which a light produces shadows that are the same as those in the input image, and the proposed method produces realistic shadows in real-time while enabling editing of the pseudo scene. In this paper, using 3D models and synthesized images in which a 3D model casts a hard shadow on a plane, we evaluate the consistency of the shadow cast by the pseudo 3D shape created by our object shape estimation method.

1. Introduction

Shadow, as one of the most common lighting effects, is very important in computer graphics, computer vision, image processing, and many other fields. Shadow provides appropriate cues for reconstructing the 3D geometry of a scene, and there has been much research focusing on estimation with shadow.

In general, shadows can be classified into cast-shadows and self-shadows. A cast-shadow is cast by a light source, and there is always an object between the light source and the shadow. The shape of a cast-shadow is related to the shape of the object that occludes light rays from the light source, so the cast-shadow also indicates the shape of the object that casts it. Much research had focused on estimating, rendering, and removing cast-shadows in images.

Self-shadow, in contrast, is less frequently discussed than cast-shadow, although it is also important in the human perception of 3D geometry. The difference between a self-shadow and a cast-shadow is that, with a self-shadow, the object occluding light rays from a light source is part of the surface on which the shadow appears, rather than another object. Self-shadows are often observed on bumpy surfaces such as grass or folded clothes.

We had presented a method to estimate and create a pseudo 3D geometry of scenes from a single image in which there are objects casting hard shadows [1]. In this paper, we evaluated the robustness of our 3D shape estimation method against the positions of objects and light sources in images by using the synthesized images in which a 3D model is placed on a plane and is illuminated by a point light source, which means that the 3D model casts a hard shadow on the plane.

[1] Tokyo Institute of Technology
[2] Ochanomizu University
[a)] wu@img.cs.titech.ac.jp
[b)] suguru@is.ocha.ac.jp

2. Related Work

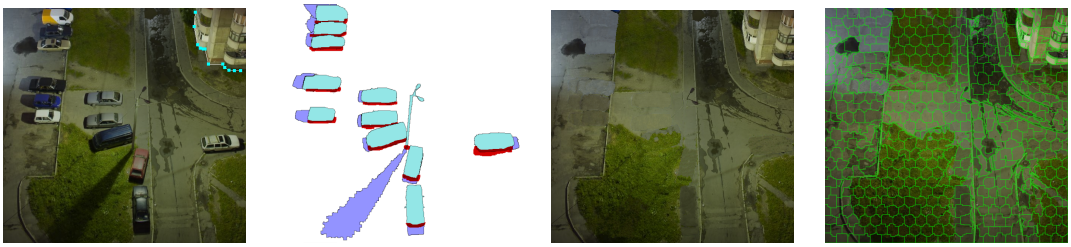
Estimation of Shadows

Wu et al.[2] modeled the cast-shadow using a Gaussian mixture model. Their method removes the cast-shadow from an image and then synthesizes it onto another image. Bousseau et al.[3] estimated the shade and self-shadows of objects and separated the image into the illumination component and the reflectance component. The user can then change the textures and a realistic result is synthesized. Both methods extract and allow limited manipulation of the shadows in an image, but the shape of the shadows is not changeable because the object that casts the shadow is not taken into account.

Kee et al.[4] proposed a method to reconstruct the relationship between shadow, object, and light source from a single image with user input. The user specifies the shadow, and the position of the object that casts the user-specified shadow is then accurately computed with the reconstructed relationship between the light source, object, and shadow. This method aims to determine the position of the object in an image from its cast-shadow. Different from this method, we want to generate a 3D model for the object with its cast-shadow, so the user is required to specify the object region in an image.

Shadow Carving

Savarese et al.[5] built a device to automatically reconstruct the geometry of an object by self-shadows. In their method, an object is placed on a rotatable plate and a camera surrounded by several lights is aimed at the object. By alternately turning on different lights and capturing pictures of the object from different views, a 3D model of the object is created by the self-shadows observed in the images. However, since this method requires images from multiple views and lighting from different directions, it is difficult to apply it to an ordinary single photograph. In contrast, our

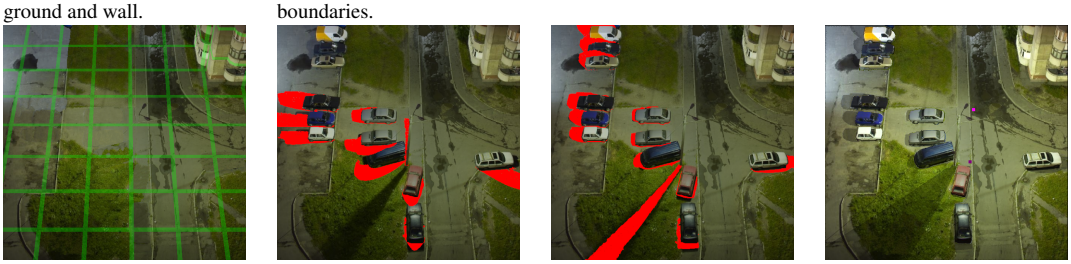


(a) Input image and user-marked edges between the ground and wall.

(b) Segmentation of objects cast-shadows, and contacting boundaries.

(c) The shadow-free texture, **T**, obtained by PatchMatch.

(d) Superpixels established by VCells.



(e) 3D geometry of ground and walls.

(f) Shadows cast by 3D model of objects before optimization.

(g) Shadows cast by optimized 3D model of objects.

(h) Scene synthesized with recovered 3D models.

Fig. 1 System overview. Our method requires an input image (a) and segmentation of the objects and the cast-shadows (b). A shadow-free texture (c) is generated by removing cast-shadows and objects using the PatchMatch algorithm. User annotations, self-shadows, and superpixels (d) are used to reconstruct the 3D geometry of the ground and walls (e), and the shape of the objects are recovered with their cast-shadows (f)(g). The final result (h) is synthesized with the shadow-free texture and reconstructed 3D models and the user is able to edit the image by moving the objects and the light source.

method estimates the shape of an object with a single input image and a small number of user annotations that can be specified in just minutes.

User-Assisted Image-Based Modeling

Horry et al.[6] proposed an efficient method to create a simplified 3D scene with one input image and a small amount of user input. Although this method allows rendering from different viewpoints in a scene, the scene itself is not editable. Karsch et al.[7] presented a method to estimate the scene and illumination condition in an image, and a realistic result of inserting synthetic objects into the image is synthesized with the estimated 3D model and illumination condition. However, this method also does not support editing of the scene and the lighting.

The method proposed by Zheng et al.[8] allows users to interactively edit the scene in an image. It effectively extracts the object by minimizing the error, which is computed according to the user annotations, and provides an interface for the user to modify the objects. However, their assumption that the objects are combinations of cuboids means their method can not be applied to objects with curved surfaces such as cars or trash cans. Our method, in contrast, assumes that objects have a continuous surface but still allows sharp edges, so it can generate a 3D model for objects of different shapes.

Kholgade et al.[9] proposed a method to manipulate the 3D object in a single image. This method supports full 3D operations, including translation, rotation, and scaling, but it assumes that the 3D model and the texture of the object in the image are known. This assumption limits the application of their method to objects whose 3D model is available. With our method, we aim to create pseudo 3D models in one image with constraints provided by the user and do not require the original 3D model of the object.

3. Algorithm

In this section, we briefly explain our method proposed in [1] and modify it by introducing weight to the terms in the objective function. The appearance of the scene in an image **I** is determined by its geometry **X**, texture **T**, and light source **L**.

$$\mathbf{I} = v(\mathbf{L}, \mathbf{X})\ \mathrm{lam}(\mathbf{L}, \mathbf{X})\ \mathbf{T} \tag{1}$$

where $v(\cdot)$ denotes the visibility of the light source **L** at the scene **X** and $\mathrm{lam}(\cdot)$ is the Lambertian cosine function. We ask the user to specify the light source **L** and to label the background, the objects, its cast-shadow pixels, and the boundaries where the objects come into contact with the ground or walls, as shown in Fig.1(b), in which pixels labeled as the objects are cyan, shadow-labeled pixels are blue, pixels on contacting boundaries are red, and the background is white, because **I** is the only known variable in Eq.1, and estimating **X**, **T**, and **L** from **I** is highly ill-posed. One object region must have one shadow region. This user input provides powerful cues to solve **X** and **T**. The texture of the scene, **T**, is obtained by inpainting the user-labeled object regions and shadow regions with the PatchMatch algorithm proposed by Barnes et al. [10]. Fig.1(c) shows the scene texture **T**, in which the objects and their cast-shadows are removed. Typical shadow removal methods, e.g., [2], can be applied to generate **T**, too, but these may require a greater number of user annotations.

The 3D geometries of the scene are divided into the ground and walls, $\mathbf{X}_S$, and the other objects, $\mathbf{X}_i$. The 3D model $\mathbf{X}_i$ corresponds to the i-th object region, $\mathbb{O}_i$, labeled by the user. Different approaches are used to create $\mathbf{X}_S$ and $\mathbf{X}_i$.

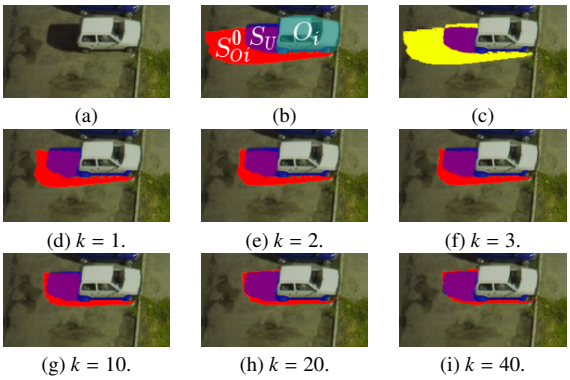


Fig. 2 (a) Input image. (b) $\mathbb{O}_i$ (cyan), the i-th object region labeled by user, $\mathbb{S}^0_{Oi}$ (red), the shadow cast by initial 3D model $\mathbf{X}^0_i$, and $\mathbb{S}_U$ (blue), the shadow region labeled by user. (c) The symmetric difference (yellow) of $\mathbb{S}^0_{Oi}$ and $\mathbb{S}_U$, i.e., $\mathbb{S}^0_{Oi} \triangle \mathbb{S}_U \equiv (\mathbb{S}^0_{Oi} \bigcup \mathbb{S}_U) - (\mathbb{S}^0_{Oi} \bigcap \mathbb{S}_U)$. (d) to (i) The shadow $\mathbb{S}^k_{Oi}$ (red), which is cast by the 3D model $\mathbf{X}^k_i$ obtained after k iterations, becomes more consistent with the user-specified shadow region $\mathbb{S}_U$ when the number of iterations, k, grows.

Here, two assumptions are introduced that the walls are perpendicular to the horizontal ground and that the camera is at the origin and has no yaw or roll rotation. We first generate a rough 3D geometry of the ground and walls, $\mathbf{X}^0_S$, using the approach proposed by Iizuka et al. [11]. This method automatically generates 3D geometry for the ground and walls as well as camera parameters with edges between the ground and walls marked by the user, as illustrated in Fig.1(a). The rough 3D geometry $\mathbf{X}^0_S$ is refined to $\mathbf{X}_S$ by using self-shadows.

For $\mathbf{X}_i$, our method processes one object at a time. We first generate a rough 3D model for the current object and then refine the 3D model by the relationships between the light source, the object, and its cast-shadow using the method presented in Section 3.1.

After deriving $\mathbf{X}_S$ and all $\mathbf{X}_i$, the final 3D geometry $\mathbf{X}$ is constructed by inserting all $\mathbf{X}_i$ into $\mathbf{X}_S$. The user is allowed to modify $\mathbf{X}$ and $\mathbf{L}$ by moving the foreground objects and the light source, and the resulting image is then synthesized by Eq.1. In our implementation, the shadow mapping algorithm proposed by [12] is used to generate the visibility term $v(\cdot)$ in Eq.1.

3.1 Object Model Creation

The 3D model $\mathbf{X}_i$ is composed of vertices that correspond to the pixels inside the i-th object region $\mathbb{O}_i$ labeled by the user, i.e., $\forall$ pixel $p \in \mathbb{O}_i$, p corresponds to one $\mathbf{x}_p$, where $\mathbf{x}_p$ is a vertex of $\mathbf{X}_i$, and the inflation method introduced by [13] is applied to determine the initial position of each vertex $\mathbf{x}_p$. Let $\mathbf{X}^0_i$ be the 3D model created by the inflation method. It casts a shadow $\mathbb{S}^0_{Oi}$, which is calculated by a common shadow mapping technique with the light specified by the user. The pixels that are labeled as 'object' are excluded from the cast-shadow since we do not know what the shadow looks like behind the object. Fig.2(b) shows the synthesized shadow region $\mathbb{S}^0_{Oi}$ (red) and the shadow region labeled by the user (blue), denoted as $\mathbb{S}_U$. The symmetric difference of $\mathbb{S}^0_{Oi}$ and $\mathbb{S}_U$, i.e., $\mathbb{S}^0_{Oi} \triangle \mathbb{S}_U \equiv (\mathbb{S}^0_{Oi} \bigcup \mathbb{S}_U) - (\mathbb{S}^0_{Oi} \bigcap \mathbb{S}_U)$ (yellow region in Fig.2(c)), indicates the difference between $\mathbf{X}^0_i$ and the real 3D model of the object. Note that $\triangle$ denotes the symmetric difference operator. Therefore, minimizing the symmetric difference, $\mathbb{S}_{Oi} \triangle \mathbb{S}_U$, also minimizes the difference between $\mathbf{X}^0_i$ and the real 3D model.

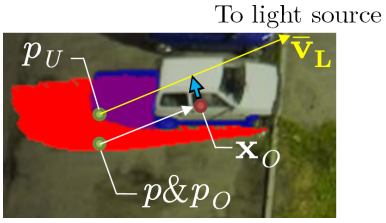


Fig. 3 For a pixel $p \in \mathbb{S}^k_{Oi} \triangle \mathbb{S}_U$, our method finds the nearest pixel p_U in $\mathbb{S}_U$, and the nearest pixel p_O in $\mathbb{S}^k_{Oi}$. Note that if $p \in \mathbb{S}^k_{Oi}$, $p \equiv p_O$, and if $p \in \mathbb{S}_U$, $p \equiv p_U$. The vertex $\mathbf{x}_O \in \mathbf{X}^k_i$ that casts its shadow on p_O is found by back-tracking from shadow mapping. The ray $\bar{\mathbf{v}}_\mathbf{L}$ from pixel p_U to light source $\mathbf{L}$ indicates where the vertex $\mathbf{x}_O$ should be located. E_{shadow} preserves the shadow consistency by minimizing the distance between $\mathbf{x}_O$ and the ray $\bar{\mathbf{v}}_\mathbf{L}$.

The initial 3D model $\mathbf{X}^0_i$ is refined by taking the relationship between the differences in shadow region and 3D model as well as the other two constraints into consideration. First, the 3D model should cast a shadow that has the same shape as the shadow region labeled by the user (the blue region in Fig.2(b)). Second, the projection of the 3D model on the image plane should fit the region of the object labeled by the user (the cyan region in Fig.2(b)). Third, it is assumed that the object does not float in the air, i.e., that it is located either on the ground or on the wall.

Our objective function f is designed in accordance with these constraints. It consists of a local smoothing term E_{smooth}, a gradient term $E_{gradient}$, an anchor term E_{anchor}, a projection constraint $E_{project}$, and a shadow constraint E_{shadow}:

$$\begin{aligned} f(\mathbf{X}_i, \mathbf{X}^k_i, \mathbb{S}^k_{Oi}) = {} & \lambda_{smooth}\, E_{smooth}(\mathbf{X}_i) + \lambda_{gradient}\, E_{gradient}(\mathbf{X}_i, \mathbf{X}^k_i) + \\ & \lambda_{anchor}\, E_{anchor}(\mathbf{X}_i) + \lambda_{project}\, E_{project}(\mathbf{X}_i) + \\ & \lambda_{shadow}\, E_{shadow}(\mathbf{X}_i, \mathbb{S}^k_{Oi}), \end{aligned} \tag{2}$$

where λ_{smooth}, $\lambda_{gradient}$, λ_{anchor}, $\lambda_{project}$, and λ_{shadow} are non-negative scalars representing the weight of each term. The 3D model is iteratively refined by minimizing Eq.2

$$\mathbf{X}^{k+1}_i = \arg\min_{\mathbf{X}_i} f(\mathbf{X}_i, \mathbf{X}^k_i, \mathbb{S}^k_{Oi}) \tag{3}$$

The local smoothing term E_{smooth} is defined as

$$E_{smooth}(\mathbf{X}_i) = \sum_{p \in \mathbb{O}_i} \sum_{p_j \in \mathbb{N}(p)} \|\mathbf{X}_i(p) - \mathbf{X}_i(p_j)\|^2, \tag{4}$$

where $\mathbb{N}(\cdot)$ represents the adjacent neighbors of a given pixel and $\mathbf{X}_i(p)$ is the corresponding vertex in $\mathbf{X}_i$ to pixel p.

E_{smooth} keeps the local continuity of $\mathbf{X}_i$ by restricting the position of vertices in $\mathbf{X}_i$ to be as close to their neighbors as possible.

The gradient term $E_{gradient}$ maintains the similarity of local gradient between $\mathbf{X}_i$ and $\mathbf{X}^k_i$ by comparing their gradients.

$$E_{gradient}(\mathbf{X}_i, \mathbf{X}^k_i) = w(p, p_j)\|(\mathbf{X}_i(p) - \mathbf{X}_i(p_j)) - (\mathbf{X}^k_i(p) - \mathbf{X}^k_i(p_j))\|^2, \tag{5}$$

where $w(\cdot)$ is the weight that returns a large value if pixels p and p_j have similar colors and a small value otherwise. It means that large differences across the edge of different textures are allowed by $w(\cdot)$, which is defined by the inverse of color distance between p and p_j.

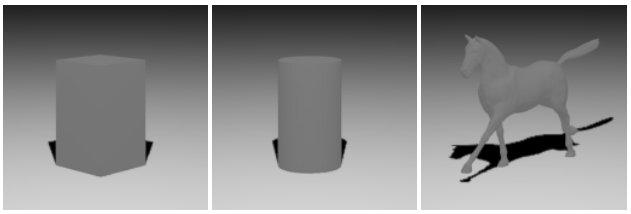


Fig. 4 The objects used in our user test: a cuboid, a cylinder, and a horse.

$$w(p, p_j) = \frac{\epsilon}{\|\mathbf{I}(p) - \mathbf{I}(p_j)\| + \epsilon} \tag{6}$$

where $\mathbf{I}$ denotes the input image represented in RGB color space, and each color channel is from zero to one. ϵ is a small value (1/255 in our implementation) to prevent from division by zero.

The anchor term E_{anchor} literally anchors the object by indicating where it should be placed in the scene. E_{anchor} is defined as

$$E_{anchor}(\mathbf{X}_i) = \sum_{p \in \mathbb{U}_G} \|\mathbf{X}_i(p) - \mathbf{X}_S(p)\|^2, \tag{7}$$

where $\mathbb{U}_G$ is the user-specified contacting boundaries.

$E_{project}$ forces the projection of the 3D model $\mathbf{X}_i$ onto the image plane to be as similar to the object region $\mathbb{O}_i$ labeled by user as possible. The projection consistency is evaluated with the distance between vertex $\mathbf{X}_i(p)$ and $\mathbf{v}_p$, which denotes the ray emitted from the camera and passing pixel p on image plane.

$$E_{project}(\mathbf{X}_i) = \sum_{p \in \mathbb{O}_i} \|\mathbf{X}_i(p) - \frac{\mathbf{X}_i(p) \cdot \mathbf{v}_p}{\|\mathbf{v}_p\|^2} \mathbf{v}_p\|^2 \tag{8}$$

where $\mathbf{v}_p = \mathbf{X}_S(p)$ since the camera is assumed to be located at the origin.

The shadow constraint E_{shadow} evaluates the consistency between $\mathbb{S}^k_{Oi}$ and $\mathbb{S}_U$. For a pixel $p \in \mathbb{S}^k_{Oi} \triangle \mathbb{S}_U$, our method finds the nearest pixel p_U in $\mathbb{S}_U$and the nearest pixel p_O in $\mathbb{S}^k_{Oi}$. Pixel p_U indicates where the object's cast-shadow should appear, so we achieve the consistency between $\mathbb{S}^k_{Oi}$ and $\mathbb{S}_U$by modifying the shape of the 3D model $\mathbf{X}^k_i$ to move the shadow from p_O to p_U. Let $\mathbf{x}_O$ be the vertex in $\mathbf{X}^k_i$ that casts a shadow on $\mathbf{X}_S(p_O)$. The ray $\overline{\mathbf{v}}_\mathbf{L}$ from $\mathbf{X}_S(p_U)$ to light source $\mathbf{L}$ indicates where $\mathbf{x}_O$ should be located. The shadow consistency is kept by minimizing the distance between vertex $\mathbf{x}_O$ and the ray $\overline{\mathbf{v}}_\mathbf{L}$.

$$E_{shadow}(\mathbf{X}_i, \mathbb{S}^k_{Oi}) = \sum_{p \in \mathbb{S}^k_{Oi} \triangle \mathbb{S}_U} \|\mathbf{x}'_O - \frac{\mathbf{x}'_O \cdot \overline{\mathbf{v}}_\mathbf{L}}{\|\overline{\mathbf{v}}_\mathbf{L}\|^2} \overline{\mathbf{v}}_\mathbf{L}\|^2 \tag{9}$$

where $\overline{\mathbf{v}}_\mathbf{L} = \mathbf{L} - \mathbf{X}_S(p_U)$, and $\mathbf{x}'_O$ is the vector from $\mathbf{X}_S(p_U)$ to $\mathbf{x}_O$, i.e., $\mathbf{x}'_O = \mathbf{x}_O - \mathbf{X}_S(p_U)$.

Fig.3 visualizes the relationship between p, p_U, p_O, $\mathbf{x}_O$, and $\overline{\mathbf{v}}_\mathbf{L}$ used in Equation9. The vertex $\mathbf{x}_O$ is found by back-tracking from shadow mapping [12]. If there are multiple vertices casting a shadow on the same pixel, only the vertex that is closest to the light source is reserved for back-tracking.

Since Eq.3 is a quadratic function of $\mathbf{X}_i$, it can be solved by deriving the first-order derivative with respect to $\mathbf{X}_i$ and then applying a linear solver (Sorkine and Alexa [14]. The 3D model $\mathbf{X}_i$ is refined iteratively until the difference of shadow, which is evaluated by $|\mathbb{S}^k_{Oi} \triangle \mathbb{S}_U|$, is less than a threshold. In our implementation, we selected $0.05 \times |\mathbb{S}_U|$ as the threshold. Fig.2(d)-(i) shows the results after different numbers of iterations.

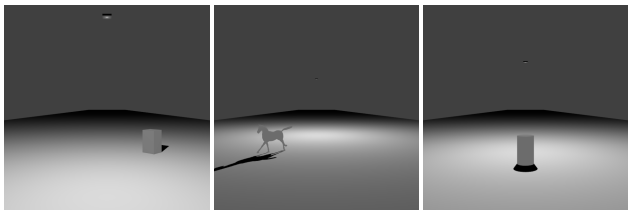


Fig. 5 The object and light source locate at different positions in the test images.

4. User Test with Equivalent Weights

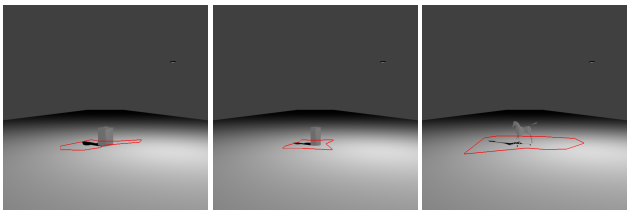
We performed a user test to evaluate the visual plausibility of the shadows cast by the pseudo 3D models created by our method with equivalent weights to all terms in Eq.2, that is, we set all λs to one.

We tested three different objects: a cuboid, a cylinder, and a horse as shown in Fig.4. The cuboid is used to evaluate the visual plausibility for objects with sharp corners, the cylinder is for objects whose normal vector of the surface changes smoothly near the profile, the horse is for the objects that has a more complex shape which has concaves.

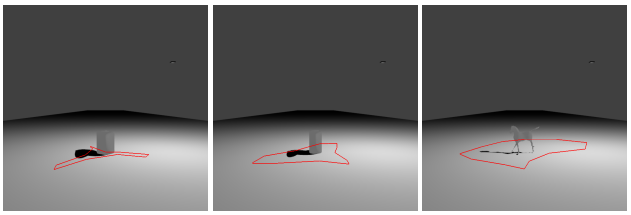
In order to examine the visual plausibility of the synthetic shadow, the regions in which the created pseudo 3D models can move with a visual consistent cast shadow were answered by participants and recorded. The scene was consisted of a plane, a point light source, and one of the three objects placed on the plane. No textures were assigned to the plane and object so that the participants see the contour of the shadow and object more clearly. In order to evaluate the robustness of our system against the positions of the objects and point light sources in an image, we prepared twenty test images in which one object located at different places and was illuminated by point light source from different directions in the image for each object as shown in Fig.5. The test images were rendered by "Blender"[15] with default rendering settings.

Three participants were involved in this user test. In this test, the participant saw one scene at a time and one participant answered one third of the questions. One question was to ask the participant to answer a limit point that the shadow of an object or the object itself seems strange while moving the object along one direction in the scene by dragging a slidebar under the scene image. The scene was reconstructed by our method from a test image and was rendered with the same configurations that was used to render the test images. For each test image, twelve moving directions of the object were prepared. Since there were three objects and twenty test images for each object, there were totally 720 questions in our user test. The limit points answered by the participants were locally weighted averaged in direction and then were used to circle the region in which the pseudo 3D models created by our method can provide visually plausible images.

Fig.6 shows the results obtained from the user test. The red lines indicate the region in which the object whose 3D shape was estimated from the test image by our previous method casts visually plausible shadow in the image. Table 1 shows the overall area percentage of visually plausible regions in the image. In this table, each row shows the results of one of the three objects un-



(a) 1.1% (b) 0.88% (c) 3.34%



(d) 1.1% (e) 2.7% (f) 4.9%

Fig. 6 The participants tended to answer a larger region for horse (c)(f) than for cuboid (a)(d) and cylinder (b)(e). The percentage showing below each image indicates the area percentage of the visually plausible region, which is answered by the participants, to the image. The three objects in one row are placed at the same position and illuminated by the same light.

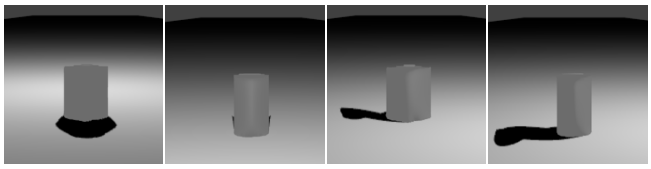
der different lighting conditions, each column shows the results of the three objects under the same lighting conditions, and each element shows the area percentage to the input image of the region answered by participants, in which the pseudo object created by our method produced visually plausible cast shadows.

Table 1 The area percentage of visually plausible regions to the image answered by the participants.

	Back-side	Fore-side	Left- and right-side	Top
Cuboid	6.47%	5.73%	1.32%	1.16%
Cylinder	7.90%	2.31%	2.00%	1.37%
Horse	4.54%	6.05%	4.01%	4.07%

From this table, we see that the visually plausible area of the horse, which has the most complex shape, tends to be larger than that of the cuboid and the cylinder, which have a simple 3D shape, while the light comes from the left- and right-side and the top of the object. This is probably by the reason that it is much more difficult for the participant to correctly perceive the 3D shape of the horse than that of the cuboid and the cylinder. Fig.6 shows the regions answered by participants for different objects placed at the same location under the same lighting conditions. The visually plausible regions of the horse (Fig.6(e)(f)) are larger than that of the cuboid (Fig.6(a)(b)) and the cylinder (Fig.6(c)(d)).

By this user test, we found that while the light comes from the back-side or fore-side of the object, that is, if the point light source does not locate at the left- and right-side of the object, the shadow is more visually plausible as shown in Fig.7. We think that there are two reasons. The first reason is that the projection term $E_{project}$ provides strong and explicit constraints to the x- and y-coordinate of the vertices but there is no direct constraints for the z-coordinate, that is, the depth, of the vertices, so the z-coordinate of vertices of the created 3D models may not be correctly estimated. While the light comes from the back- or fore-side of object, the depth of the object does not influence the shape of the shadow so much, therefore the shadow looks plausible. On



(a) (b) (c) (d)

Fig. 7 The shadow of the object whose 3D shape is recovered by our method seems fine if the light comes from the back-side (a) or fore-side (b) but the shadow seems distorted while the light comes from the right-side (c)(d).

the other hand, while the light comes from the left- or right-side of object, the incorrect depth of object causes noticeable distortion to the shape of the shadow. The second reason is that our 3D shape estimation method does not recover the hidden part of object, so the shadow-constraint term (Eq.9) tended to stretch the 3D model, which makes the shadow look strange while the light comes from the left- or right-side as shown in Fig.7(c)(d).

5. User Test with Different Weights

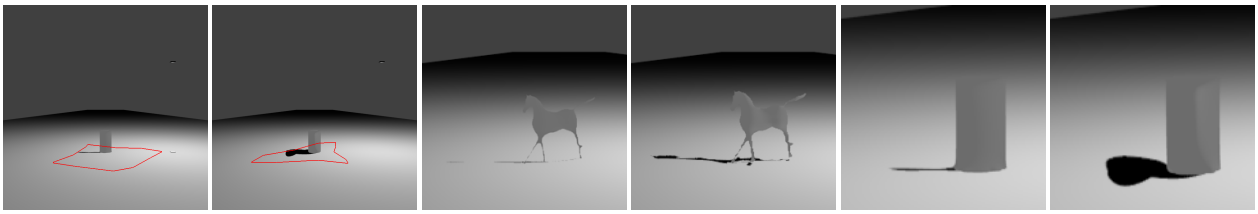
We tried to solve the problem of stretched 3D models by adjusting the weight of the terms in Eq.2 to $\lambda_{smooth} = 10$, $\lambda_{gradient} = 1$, $\lambda_{anchor} = 5$, $\lambda_{project} = 5$, and $\lambda_{shadow} = 1$. We created 3D models for the cylinder and horse again with this new setting of weights and asked the participants to do the user test again with the same process and settings as described in Section 4 but using the new reconstructed 3D models. The result of this additional experiment is shown in Table 2.

Table 2 The area percentage of visually plausible regions of the new 3D models to the image answered by the participants.

	Back-side	Fore-side	Left- and right-side	Top
Cylinder	8.87%	5.15%	2.07%	0.61%
Horse	11.74%	8.89%	3.55%	3.26%

Table 2 shows the result of the user test with 3D models created with different weights and we found that the participants answered a larger visually plausible region for these new 3D models while the light comes from the back- and fore-side but a smaller or almost equal region while the light comes from the left-side, right-side, and top. This is because λ_{smooth} is much larger than λ_{shadow}, so the created 3D shapes were not stretched too much. However, large λ_{smooth} makes the 3D shape flattened. While the light comes from the right-side of the object, we can see that the shadow of the 3D models created with different weights (Fig.8(c)) is obviously thinner than that created with equivalent weights (Fig.8(d)), and thus the participants answered a smaller visually plausible region for the horse in Table 2 than that in Table 1. The fact that the visually plausible region for the cylinder in Table 2 is slightly larger than that in Table 1 indicates that the participants thought that a thin but straight shadow (Fig.8(f)) looks more plausible than a stretched shadow (Fig.8(e)).

From the experiments, we found that the 3D model created by our method can produce visually plausible shadows while it is placed in 2% to 4% area of the whole image around its original position, and that the 3D model created with equivalent weights has better performance than a flattened 3D model (e.g., a billboard) while the light comes from side of the object.



(a) 4.16% (b) 2.7% (c) 4.32% (d) 4.9% (e) 1.73% (f) 1.06%

Fig. 8 The 3D models created with different weights (a) casts visually plausible shadow in a larger region than that created with equivalent weights (b). However, the shadow cast by the 3D models created with different weights (c)(e) is much thinner than that created with equivalent weights (d)(f) while the light comes from the right-side. The percentage showing below each image indicates the area percentage of the visually plausible region to the image.

Although we had thought that the pseudo 3D model created by our method could produce a visually plausible shadow while it moved along the direction parallel to its cast shadow, the experimental results show that visually plausible shadows were also generated while the created pseudo 3D model moved in more various directions as shown in Fig.6, which means that our method is suitable for the applications that need to move the objects with cast shadows in an image.

6. Conclusion

We evaluated the visual plausibility of shadows cast by objects which are reconstructed with our object shape estimation method [1] by the user test and found three conclusions.

(1) Our method works well while the light source is at the back- or fore-side of the shadow-casting object.

(2) Placing the shadow-casting object further away from the light source usually leads to more visually plausible results.

(3) Our method is more suitable to the shadow-casting objects whose 3D shape is complex.

We think that adopting objects with more different shapes (e.g., sharp edges with concaves) helps the conclusions above more clearly. The improvement of the visual plausibility of the estimation result, redesign the gradient and shadow terms in our objective function, and the recovery of the hidden part of an object are also three important future works.

References

[1] Wu, J.-H. and Saito, S.: Synthesizing Realistic Shadows with Bumpy Surface and Object Model Estimated by Shadows in Single Image, *Proceedings of the 10th International Conference on Computer Graphics Theory and Applications*, pp. 35–45 (2015).

[2] Wu, T.-P., Tang, C.-K., Brown, M. S. and Shum, H.-Y.: Natural shadow matting, *ACM Trans. Graph.*, Vol. 26, No. 2 (online), DOI: 10.1145/1243980.1243982 (2007).

[3] Bousseau, A., Paris, S. and Durand, F.: User-assisted intrinsic images, *ACM Trans. Graph.*, Vol. 28, No. 5, pp. 130:1–130:10 (2009).

[4] Kee, E., O'Brien, J. F. and Farid, H.: Exposing Photo Manipulation with Inconsistent Shadows, *ACM Trans. Graph.*, Vol. 32, No. 3, pp. 28:1–28:12 (2013).

[5] Savarese, S., Andreetto, M., Rushmeier, H. E., Bernardini, F. and Perona, P.: 3D Reconstruction by Shadow Carving: Theory and Practical Evaluation., *International Journal of Computer Vision*, Vol. 71, No. 3, pp. 305–336 (2007).

[6] Horry, Y., Anjyo, K.-I. and Arai, K.: Tour into the Picture: Using a Spidery Mesh Interface to Make Animation from a Single Image, *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co., pp. 225–232 (online), DOI: 10.1145/258734.258854 (1997).

[7] Karsch, K., Hedau, V., Forsyth, D. and Hoiem, D.: Rendering synthetic objects into legacy photographs, *ACM SIGGRAPH Asia '11*, pp. 157:1–157:12 (2011).

[8] Zheng, Y., Chen, X., Cheng, M.-M., Zhou, K., Hu, S.-M. and Mitra, N. J.: Interactive images: cuboid proxies for smart image manipulation, *ACM Trans. Graph.*, Vol. 31, No. 4, pp. 99:1–99:11 (2012).

[9] Kholgade, N., Simon, T., Efros, A. and Sheikh, Y.: 3D Object Manipulation in a Single Photograph Using Stock 3D Models, *ACM Trans. Graph.*, Vol. 33, No. 4, pp. 127:1–127:12 (online), DOI: 10.1145/2601097.2601209 (2014).

[10] Barnes, C., Shechtman, E., Finkelstein, A. and Goldman, D. B.: PatchMatch: a randomized correspondence algorithm for structural image editing, *ACM SIGGRAPH '09*, pp. 24:1–24:11 (2009).

[11] Iizuka, S., Kanamori, Y., Mitani, J. and Fukui, Y.: Efficiently Modeling 3D Scenes from a Single Image, *IEEE CG&A*, Vol. 32, pp. 18–25 (2011).

[12] Williams, L.: Casting Curved Shadows on Curved Surfaces, *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78, New York, NY, USA, ACM, pp. 270–274 (online), DOI: 10.1145/800248.807402 (1978).

[13] Johnston, S. F.: Lumo: Illumination for Cel Animation, *Proceedings of the 2Nd International Symposium on Non-photorealistic Animation and Rendering*, NPAR '02, New York, NY, USA, ACM, pp. 45–ff (online), DOI: 10.1145/508530.508538 (2002).

[14] Sorkine, O. and Alexa, M.: As-Rigid-As-Possible Surface Modeling, *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pp. 109–116 (2007).

[15] Blender: 2.72 Release. https://www.blender.org/ (2015).