Automatic Illumination Effects for 2D Characters

Zhengyan Gao[*,1,2] **Taizan Yonetsuji**[2] **Tatsuya Takamura**[2]
Toru Matsuoka[2] **Jason Naradowsky**[2]
[1]Tokyo Institute of Technology, [2]Preferred Networks, Inc.
[1]`gao.z.aa@m.titech.ac.jp`, [2]`{taizan,ttakamura,tmatsuoka,narad}@preferred.jp`

Abstract

In this work we present a system to apply realistic lighting effects to 2D character illustrations. Our approach involves a cascaded set of predictions: first generating a normal map of the character's 3D structure, utilizing it (and a light source) to predict a global shadow map, and combining both to render the final effect. This allows our system to paint natural overlapping shadows, and recreate complex lighting effects, such as rim-lighting, while maintaining the ease of use associated with CGI. Results on original illustrations show the effectiveness of our method.

1 Introduction

Under the pressure of external constraints, artists are often forced to compromise on the degree to which they can realize their artistic vision. This is especially true of animators, where limited time and budget often results in fewer frames, and less detail or movement. To what extent can machine learning counteract these concessions? Such recent work has sought to streamline the time-consuming task of coloring illustrations [12]. In this work we turn to yet another of these problems: the automatic shading of 2D characters under complex and possibly dynamic lighting conditions.

The predominant paradigm in this line of research is to first reconstruct a 3D/2.5D model from 2D images, and then apply rendering engines to produce the shading result. However, these approaches often rely on additional information, either requiring a human annotator to provide hints of the intended 3D structure [3, 8, 11, 1], or multiple views of the 2D character [5]. Recent CNN-based approaches have estimated normal maps directly [10, 2]. However, such approaches do not model the rich interaction of shadows in dynamic lighting environments.

In this work we introduce a method to automatically generate rich illumination effects for 2D characters. We make a distinction between shadows which occur on the surface of a character (local illumination) and shadows which result from the primary light source (global illumination). This results in realistic shadows which are easily controllable and can be applied to illustrations or animations with minimal effort on the part of the artist.

2 Methodology

Here we briefly describe an overview of our system:

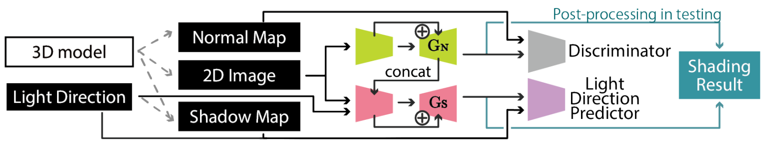

Figure 1: Architecture Overview

[*]Work completed while first author was an intern at Preferred Networks.

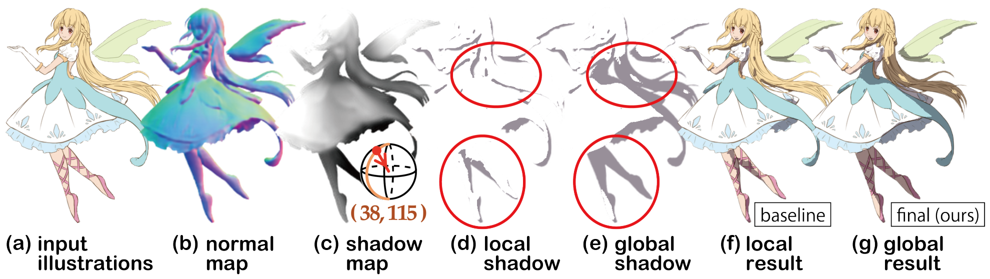

Figure 2: Results on an original illustration. Red circles highlight important differences. Result (f) uses only the normal information in (b), while our result (g) uses the global shadow information (c)

Normal Map Estimation First, a GAN architecture (here, a residual Unet [4]) is trained to predict normal maps, providing a partial reconstruction of the 3D structure implied by the 2D illustration. A 3D model is used to produce an unshaded 2D illustration and the ground truth normal maps. During training we minimize the $\mathcal{L}_1$ loss between map outputs and ground truths with the adversarial loss from SNGAN [7] and the perceptual loss from Illustration2Vec [9].

Shadow Map Estimation Second, the illustration, normal map, and light source angle are used as input to a second residual Unet which predicts the global shading information, encoding it in the shadow map. Here the rich features of the normal map provide important structural cues, improving result quality and accelerating convergence. To make the shadow map generator better adapt to the input light direction, we proposed a novel light direction loss, and utilize it together with $\mathcal{L}_1$ loss and perceptual loss during the training process.

3 Experiments

Data We construct a training dataset consisting of 240K examples, each consisting of an image, normal map, shadow map, and light direction. Each example starts out as a freely-available 3D model, and we utilize the Unity© game engine for rendering the various 2D representations that serve as the inputs and ground truths during training.

Discussion Results of our method are shown in Fig. 2. Here we compare the result of using additional shadow map information (g) to that of using only the normal map (f), and show the tendency of the baseline system to *underspecify* shading. For example, while both characters show shadows that result from top-down illumination, our result accurately shades regions where the character's structure would obstruct this lighting, such as in the waist, hair, and legs. This is not objectively better, as there could exist a lighting situation for which (f) is correct, but we argue it is more representative of realistic lighting scenarios.

Lighting can also be adjusted continuously with little additional effort (Fig. 3).

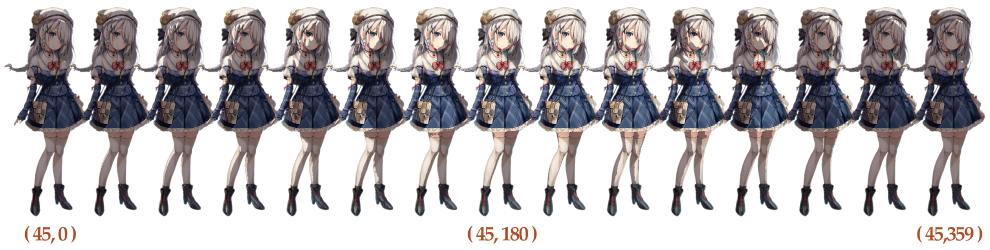

Figure 3: Shading results with continuously changing light direction.

Thus our method helps bring CGI convenience to 2D illustration, opening up opportunities for improved workflows in art studios. Future directions include support for multiple light sources, control over diffusion, and shadow projection. As an interactive design tool, by reducing shading to a single parameter, we hope to further assist artists in making stylistic lighting decisions.

References

[1] Lele Feng, Xubo Yang, and Shuangjiu Xiao. Magictoon: A 2d-to-3d creative cartoon modeling system with mobile ar. In *Virtual Reality (VR), 2017 IEEE*, pages 195–204. IEEE, 2017.

[2] Matis Hudon, Mairéad Grogan, Rafael Pagés, and Aljoša Smolic. Deep normal estimation for automatic shading of hand-drawn characters. In *The 3rd Geometry Meets Deep Learning Workshop in association*. ECCV, 2018.

[3] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM Press/Addison-Wesley Publishing Co., 1999.

[4] Chengze Li, Xueting Liu, and Tien-Tsin Wong. Deep extraction of manga structural lines. *ACM Transactions on Graphics (TOG)*, 36(4):117, 2017.

[5] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 3d shape reconstruction from sketches via multi-view convolutional networks. In *3D Vision (3DV), 2017 International Conference on*, pages 67–77. IEEE, 2017.

[6] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[7] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. ICLR, 2018.

[8] Lena Petrović, Brian Fujito, Lance Williams, and Adam Finkelstein. Shadows for cel animation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 511–516. ACM Press/Addison-Wesley Publishing Co., 2000.

[9] Masaki Saito and Yusuke Matsui. Illustration2vec: a semantic vector representation of illustrations. In *SIGGRAPH Asia 2015 Technical Briefs*, page 5. ACM, 2015.

[10] Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and Hongbo Fu. Interactive sketch-based normal map generation with deep neural networks. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):22, 2018.

[11] Daniel Sỳkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Transactions on Graphics (TOG)*, 33(2):16, 2014.

[12] Taizan Yonetsuji. Automatic colorization with chainer. https://qiita.com/taizan/items/cf77fd37ec3a0bef5d9d, 2017. Accessed: 2018-10-15.

Supplementary Materials

A Dataset Construction

Our training dataset consists of color images, and their corresponding normal maps, shadow maps and light directions, created by rendering freely-available 3D anime characters. We collected these from 3D model sharing sites, such as Niconi Solid[2]. Fig. 4 shows some examples of our training data. We collected 219 3D anime character figures in total. When creating the dataset, we rotate character models around the y-axis and apply 3 types of basic character motions (relaxing, walking and running) randomly. To compute gold shadow maps, we choose an arbitrary lighting direction, constrained to be pointing towards the center of the character, and thus this direction is specified as the angle between the light source and the center of the character. Here, we use $l_\alpha \in [0, 180]$ and $l_\beta \in [0, 359]$ to indicate the vertical angle and the horizontal angle respectively, as shown in Fig. A (a).

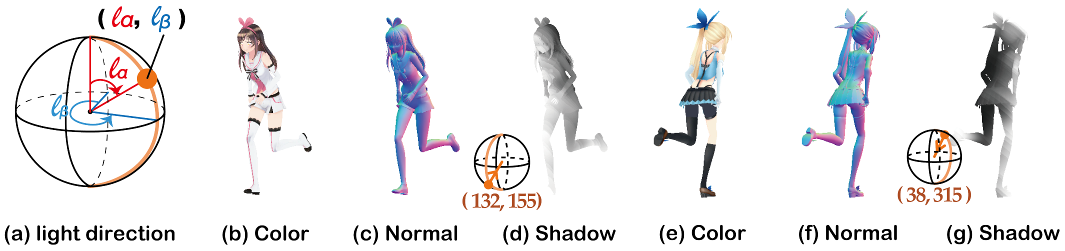

Figure 4: Examples for our dataset. (Source 3D model © Kizuna AI, © Mirai Akari Project. All rights reserved.)

B Model Details

B.1 Network Architecture

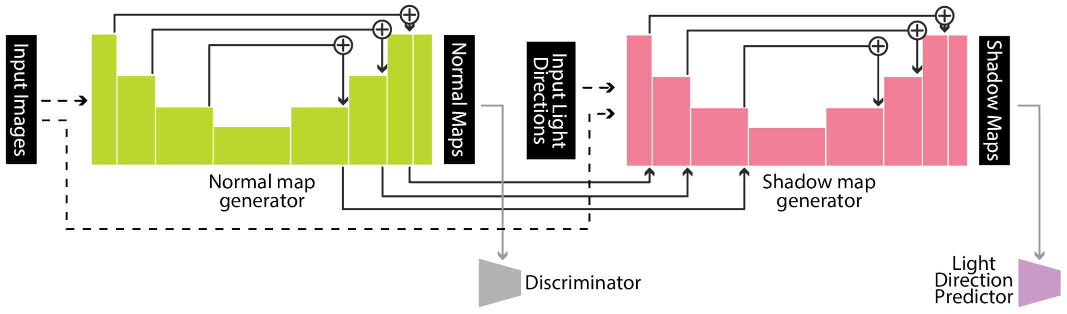

Figure 5: Overview architecture.

Our proposed networks are represented in Fig. 5, the normal map generator is on the left (depicted in green) and the shadow map generator is on the right (depicted in red), which inspired by the network proposed in [4] for extracting structural lines of pattern-rich manga. Each block of generators in Fig. 5 consists of several residual blocks, and channel-wise addition is used as the skip connections between the encoder and the decoder. Features extracted by the normal map generator are full of detailed local geometric information, so we also input those features to the shadow map generator, together with the input images and light directions. Using those features helps the shadow map generator get detailed structure information, and we observe faster convergence during training when using this technique.

[2]`https://3d.nicovideo.jp/`

B.2 Loss Function

Besides the $\mathcal{L}_1$ loss, we also use a conditional-GAN [6] loss and a perceptual loss to get high-quality results. We also add spectral normalization [7] to the discriminator for stable training. The perceptual loss is calculated as the distance of the extracted features of generated normal maps and ground truths, and we use a pretrained Illustration2Vec network [9] as the extractor. With $\widetilde{\boldsymbol{y}_N} := G_N(\boldsymbol{x})$ indicates the generated normal maps, the loss function for normal map generator is described as following:

$$\begin{aligned}
\mathcal{L}_N &= \mathcal{L}_1 + \lambda_1 \mathcal{L}_{adv} + \lambda_2 \mathcal{L}_{per} && (1)\\
\mathcal{L}_1 &= \|\widetilde{\boldsymbol{y}_N} - \boldsymbol{y}_N\|_1 && (2)\\
\mathcal{L}_{adv} &= -\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[\log(\widetilde{\boldsymbol{y}_N})] && (3)\\
\mathcal{L}_{per} &= \sum_{i=1}^{n} \frac{1}{M_i}[\|I^{(i)}(\widetilde{\boldsymbol{y}_N}) - I^{(i)}(\boldsymbol{y}_N)\|_1] && (4)
\end{aligned}$$

where $\boldsymbol{y}_N$ is the ground truth normal map with respect to the input color image $\boldsymbol{x}$, and $I^{(i)}$ denotes the i-th layer with M_i elements of the Illustration2Vec network.

Furthermore, we propose to use a pretrained classification network, indicated as the light direction predictor. Besides loss functions used in training normal map generator, we also make shadow map generator minimize the $\mathcal{L}_2$ distance between the predicted light direction of generated shadow maps and ground truths. With $F(G_N(\boldsymbol{x}))$ and $\widetilde{\boldsymbol{y}_S} := G_S[\boldsymbol{x}, F(G_N(\boldsymbol{x}))]$ indicate the extracted features from G_N and the generated shadow maps, the loss function for shadow map generator then becomes:

$$\begin{aligned}
\mathcal{L}_S &= \mathcal{L}_1 + \lambda_1 \mathcal{L}_{per} + \lambda_2 \mathcal{L}_{light} && (5)\\
\mathcal{L}_1 &= \|\widetilde{\boldsymbol{y}_S} - \boldsymbol{y}_S\|_1 && (6)\\
\mathcal{L}_{per} &= \sum_{i=1}^{n} \frac{1}{M_i}[\|I^{(i)}(\widetilde{\boldsymbol{y}_S}) - I^{(i)}(\boldsymbol{y}_S)\|_1] && (7)\\
\mathcal{L}_{light} &= \frac{1}{181}\|L_\alpha(\widetilde{\boldsymbol{y}_S}) - l_\alpha\|_2 + \frac{1}{360}\|(L_\beta(\widetilde{\boldsymbol{y}_S}) + 180 - l_\beta)\%360 - 180\|_2 && (8)
\end{aligned}$$

where $\boldsymbol{y}_S$ is the ground truth shadow map with respect to the input color image $\boldsymbol{x}$ and input labels of light direction ($l_\alpha \in [0, 180], l_\beta \in [0, 359]$). L_α and L_β are predicted light direction and $\%$ indicates the remainder operator. The horizontal angle of light direction starts from 0 and ends at 359, but 0 comes after 359. To prevent discontinuous light loss from this gap, we propose to use a mapping of the angle as shown in the second item of equation (8) .

B.3 Training Details

Line-drawing Augmentation Because artists have a high-level freedom when creating hand-drawn illustrations, real-world drawings may vary significantly from synthetic images in stylistic aspects. Thus it is important to hinder the model from over-fitting to superficial characteristics of the synthetic data generator, and to better generalize to the stylistic diversity of real-world illustrations. Specifically, line-drawings significantly influence normal map generator, which could result in unnecessarily complicated output. Therefore, we add a line-drawing-based data augmentation technique which randomly adjusts the color and thickness of line art input images, as shown in Fig. 6. We extract line-drawings from the original by the technique introduced in [12]. Effects of the line-drawing data augmentation are shown in Fig. 7.

Continuity of Input Light Direction There are 360 degrees of possible light directions along a single axis, but because of the cyclic nature of this rotation, the integer difference between 0 and 359 is large, while conceptually these correspond to successive angles. In order to create an input value that behaves continuously between the endpoints of the input interval, we divide the horizontal angle parameter $l_\beta \in [0, 359]$ to $l_{\beta_1} \in [0, 180]$ and $l_{\beta_2} \in [0, 180]$ to indicate the angle from back to front, and the angle from right to left, respectively. We then use these two parameters together as the horizontal light direction input to the shadow map generator. When the horizontal angle is 0, the input to the generator is $(0, 90)$. When it is 359, the input becomes $(1, 89)$, which preserves the continuity of the input.

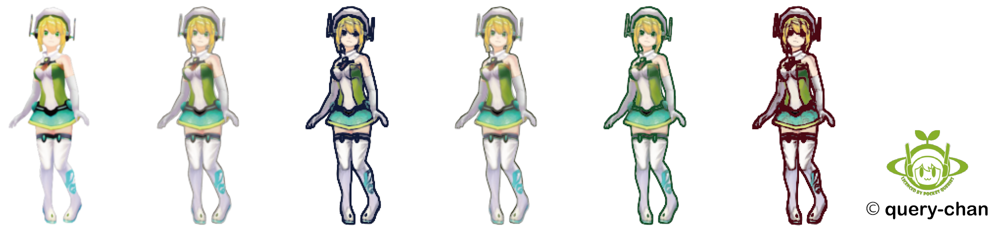

Figure 6: An example of line-drawing data augmentation. The 1st image on the left is the original, and the others are augmented images. (Source 3D model ©query-chan. All rights reserved.)

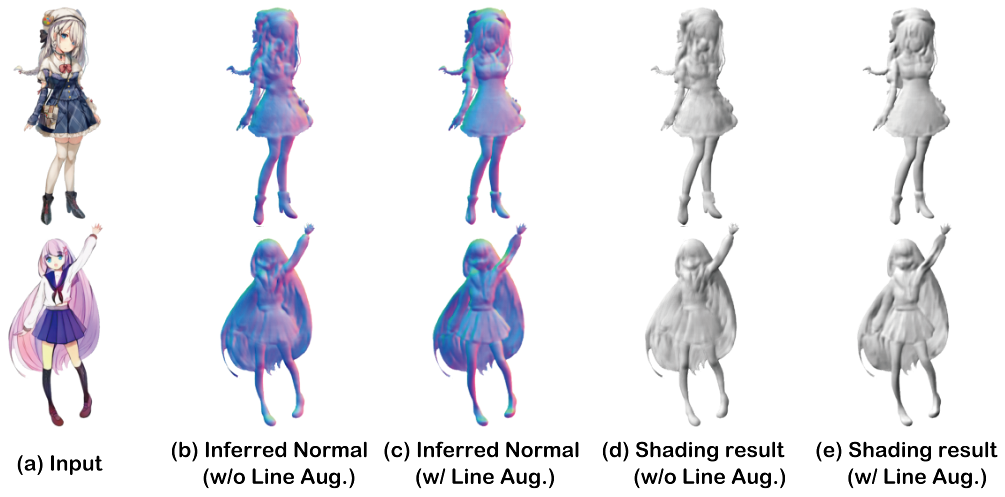
(a) Input
(b) Inferred Normal (w/o Line Aug.)
(c) Inferred Normal (w/ Line Aug.)
(d) Shading result (w/o Line Aug.)
(e) Shading result (w/ Line Aug.)

Figure 7: Comparison of inferred normal maps w/ or w/o line-drawing augmentation.

C Additional Results

Real-world artwork varies significantly from our synthetically generated training examples. Fig. 8 and Fig. 9 show our shading results with various light direction and art styles. Although the characters' cloths and poses are quite different from our synthetic data, our proposed method can generate plausible results.

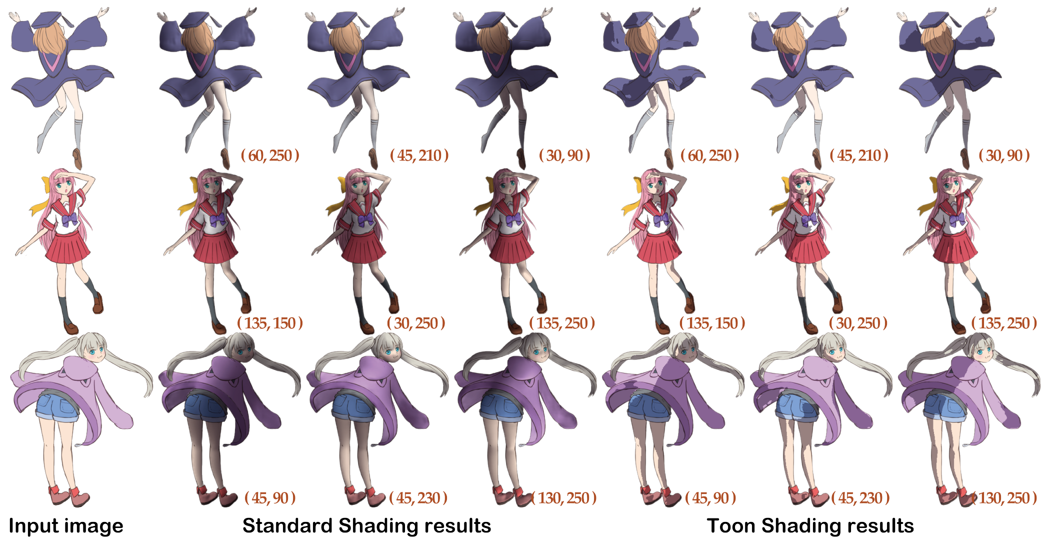

Figure 8: Shading results on original illustrations with various light direction.

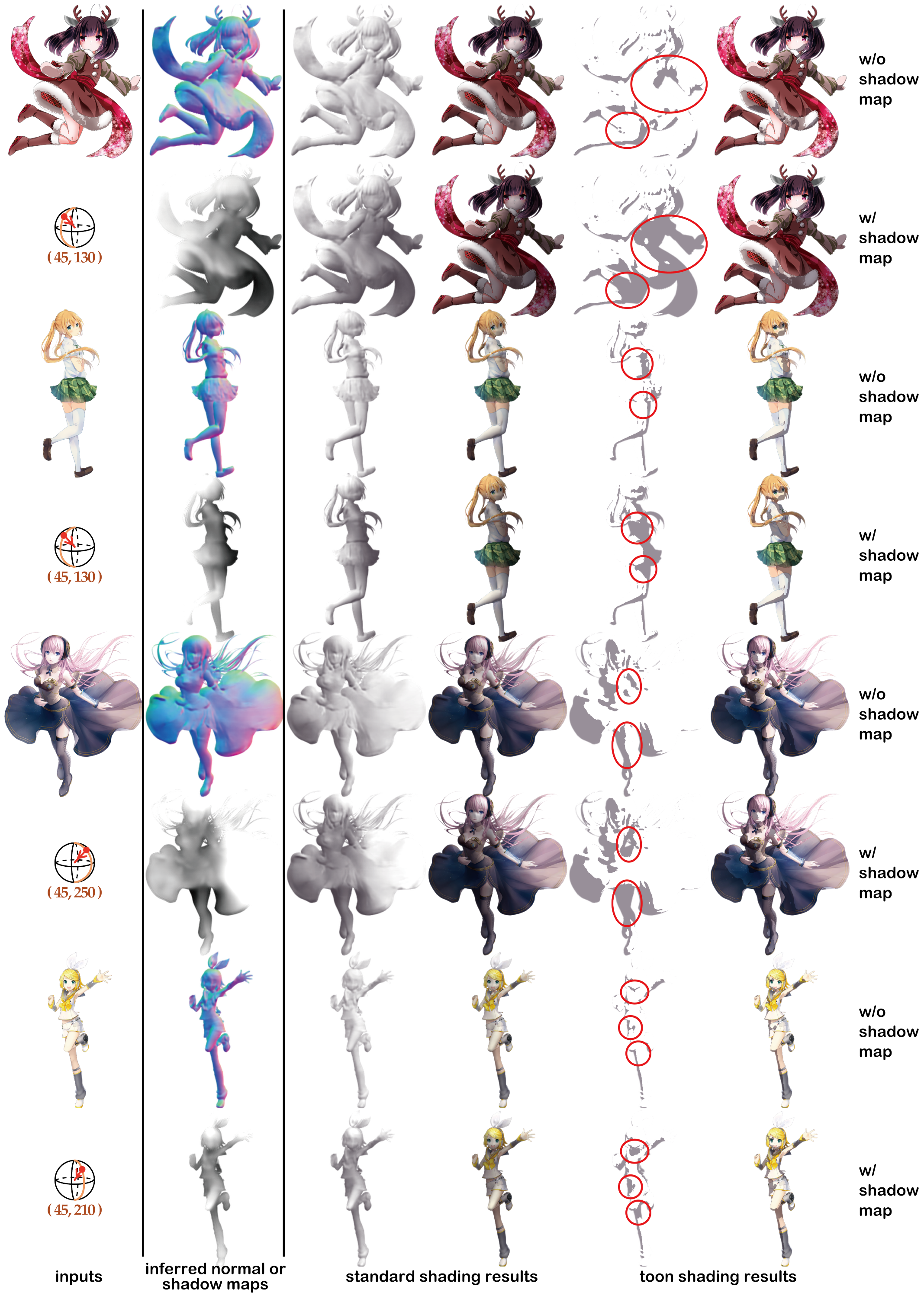

Figure 9: Additional comparison on real-world illustrations. For each sample, we show the results without using the shadow map in the 1st row, and the results with the shadow map in the 2nd row. Red ovals highlight the difference. (Source drawings © SSS LLC., © 2018 Pronama LLC., © Crypton Future Media, INC. www.piapro.net. All rights reserved.)

Rim Lighting Effect

Rim lighting effect is a popular technique in photography and in 3D rendering, and is the effect where the character is backlit, highlighting only the edge contours of the character. We can add a natural rim lighting effect to 2D characters as an application of our proposed method as shown in Fig. 10. In the real world, rim lighting effect is caused by subsurface scattering from the back side of the object. Although we do not have information about the back side of the 2D character, we can assume that the boundary of the back side is similar to the boundary of the front side. From this assumption, we invert the horizontal angle parameter and get the boundary of the shading result of the inverted light direction (Fig. 10 c), and blend it with one of the input light directions (Fig. 10 b) to calculate the rim lighting effect (Fig. 10 d). Here, we get the boundary of the character from the input image's alpha channel, and the final shading results with rim lighting effect are shown in Fig. 10 f.

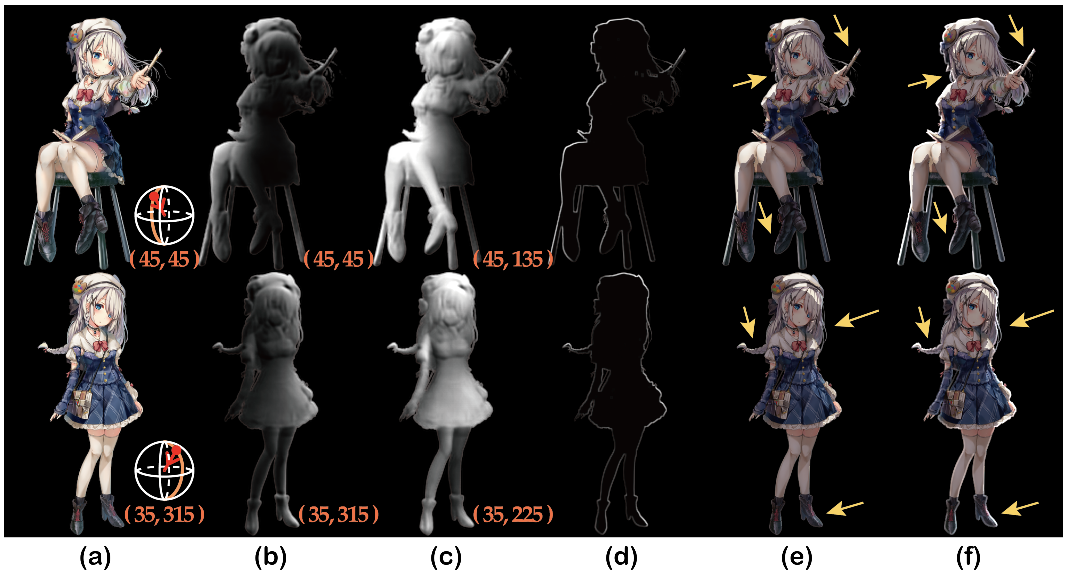

Figure 10: Rim lighting results using inferred shadow maps. (a) inputs, (b) shading results of input light direction, (c) shading results of inverted light direction, (d) rim lighting effect, (e) toon shading results, (f) toon shading results w/ rim lighting effect. Yellow arrows point out some major differences.