Augmenting Hand-Drawn Art with Global Illumination Effects through Surface Inflation

Matis Hudon
matishudon@gmail.com
VSENSE, School of Computer Science and Statistics,
Trinity College Dublin
Dublin, Ireland

Sebastian Lutz
lutzs@scss.tcd.ie
VSENSE, School of Computer Science and Statistics,
Trinity College Dublin
Dublin, Ireland

Rafael Pagés
rafa@volograms.com
Volograms
Dublin, Ireland

Aljosa Smolic
smolica@scss.tcd.ie
VSENSE, School of Computer Science and Statistics,
Trinity College Dublin
Dublin, Ireland

ABSTRACT

We present a method for augmenting hand-drawn characters and creatures with global illumination effects. Given a single view drawing only, we use a novel CNN to predict a high-quality normal map of the same resolution. The predicted normals are then used as guide to inflate a surface into a 3D proxy mesh visually consistent and suitable to augment the input 2D art with convincing global illumination effects while keeping the hand-drawn look and feel. Along with this paper, a new high resolution dataset of line drawings with corresponding ground-truth normal and depth maps will be shared. We validate our CNN, comparing our neural predictions qualitatively and quantitatively with the recent state-of-the art, show results for various hand-drawn images and animations, and compare with alternative modeling approaches.

CCS CONCEPTS

• **Theory of computation** → **Computational geometry**; • **Applied computing** → Fine arts.

KEYWORDS

shape modeling, datasets, neural networks, 2D art, animation

ACM Reference Format:
Matis Hudon, Sebastian Lutz, Rafael Pagés, and Aljosa Smolic. 2019. Augmenting Hand-Drawn Art with Global Illumination Effects through Surface Inflation. In *European Conference on Visual Media Production (CVMP '19), December 17–18, 2019, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3359998.3369400

1 INTRODUCTION

Despite the success and proliferation of 3D graphics imagery, traditional 2D sketching remains a major art communication medium and still plays an important role in the preparation phases of 3D animation production (story-boarding, character design, animation). The key advantage of 2D drawing in animation resides in its natural, completely constraint-free environment. However, to obtain a polished drawing or animation, every frame requires a considerable amount of work and some tasks can rapidly become tedious. To help with these time-consuming and repetitive tasks, scientists have tried to automate parts of the pipeline, for example by cleaning the line-art [Simo-Serra et al. 2017, 2016], scanning [Li et al. 2017], coloring [Sỳkora et al. 2009b; Zhang et al. 2017], and by developing image registration and inbetweening techniques [Sỳkora et al. 2009a; Whited et al. 2010; Xing et al. 2015].

This work is an extension of the work presented in [Hudon et al. 2018]. We present a method to automatically obtain rich illumination effects on single drawings. These effects include but are not limited to shadowing, self shadowing, diffuse and glossy shading and inter-reflections (color bleeding). Such effects are state-of-the-art in computer graphics and can be rendered providing a 3D model.

Unlike previous methods, we reconstruct our surface models from a single image input and without any user annotation or input. Our reconstruction process relies on a neural network trained to predict normal maps from single sketch images. These maps are then inflated into qualitative surface proxy models suitable for applying global illumination effects.

2 RELATED WORK

Shape modeling from hand-drawn sketches has been a very active field of research for several decades. In the following section we briefly classify the previous works into two categories: classical approaches involving geometric priors and more recent methods relying on machine learning, which often borrow their main principles from approaches in the first category.

CVMP '19, December 17–18, 2019, London, United Kingdom

ACM ISBN 978-1-4503-7003-5/19/12…$15.00
https://doi.org/10.1145/3359998.3369400

2.1 Geometric Reconstructions

One of the first significant works in this field provided inflating-based tools to build 3D models from 2D data [Igarashi et al. 1999]. Petrovic's et al. [Petrović et al. 2000] were the first to use this technique to automate the creation of shades and shadows. They demonstrated that approximated 3D models are sufficient for generating plausible and appealing shades and shadows for cel animation. However, while reducing the labor of drawing shades and shadows by hand, the method still requires too many manual interactions for it to be applicable to real world animation pipelines. In Lumo [Johnston 2002], Johnston showed that convincing illumination could be rendered only by interpolating surface normals from the line boundaries. Later on multiple 3D reconstructions improvements were made using either different types of assumptions on the surfaces [Karpenko and Hughes 2006; Olsen et al. 2009] or different user annotations [Bui et al. 2015; Jayaraman et al. 2017; Shao et al. 2012; Tuan et al. 2017], we refer the readers to [Hudon et al. 2018] for more details. Some recent works exploit geometric constraints present in specific types of line drawings [Pan et al. 2015; Schmidt et al. 2009; Xu et al. 2014], however, these sketches are too specific to be generalized to 2D hand-drawn animation. Some works have similar goals to ours. Providing tools to artists to make shading a less labour intensive task, or simply provide new shading styles and possibilities for animations. Yet those methods are not fully automatic and requires some extensive user input. Depth layering is used in TexToons [Sỳkora et al. 2011] to enhance textured images with ambient occlusion, shading, and texture rounding effects. In our opinion the method presenting the best results is Ink and Ray [Sỳkora et al. 2014], applying smart user annotation to recover a bas-relief with approximate depth from a single sketch, which they use to illuminate 2D drawings. Reconstruction results were recently improved in [Dvorožňák et al. 2018], as they formulate the reconstruction as a single non-linear optimization problem. Considerable user effort and processing time is required to obtain high quality reconstructions from drawings. We believe that a method has to be more efficient and effortless to be used in a real production pipeline. Humans are easily able to infer depth and shapes from drawings [Belhumeur et al. 1999; Cole et al. 2009; Koenderink et al. 1992], this ability still seems unmatched in computer graphics/vision methods using geometric reconstructions.

2.2 Data-driven Methods.

Geometric methods require a large number of constraints or additional user input to reconstruct high-quality 3D models from sketches. Shape synthesis appears to be a learning problem. Many recent works have tackled the problem of estimating surface depth and normals from real pictures using CNNs [Eigen and Fergus 2015; Eigen et al. 2014; Rematas et al. 2016; Wang et al. 2015]. These works show very promising results, however natural images intrinsically contain much more information about the scene than drawn sketches, such as textures, natural shades, colors, etc. Accurate results have been shown [Bansal et al. 2016; Huang et al. 2017; Pontes et al. 2017] guiding shape reconstruction through CNNs using parametric models (such as existing or deformed cars, bikes, containers, jewellery, trees, etc.). Deep learning has also been used for modeling from sketches, Han et al. [Han et al. 2017] showed impressive modeling of 3D faces and caricatures using labor efficient user inputs. Directly related to our work, Lun et al. [Lun et al. 2017], inspired by that of Tatarchenko et al. [Tatarchenko et al. 2016] use CNNs to predict shape from line-drawings. However, they make use of multi-view input line-drawings whereas our main goal is to operate on a single input drawing. Indeed, for generating illumination effects on sketches or animations, the reconstruction of a full 3D model is not necessary. A front/camera view 3D surface is sufficient. Also [Li et al. 2018] presented a similar method aimed at reconstructing 3D surface from sketches using a robust flow guided neural reconstruction. Users can refine the results by providing additional depth values at sparse points and curvatures for strokes. Compared to Lun et al. [Lun et al. 2017], their reconstructed surfaces are higher in quality with more details. Su et al. [Su et al. 2018] proposed an interactive system for generating normal maps with the help of deep learning. Their method produces relatively high quality normal maps from sketches, combining a Generative Adversarial Network framework together with user inputs. They also outperformed Lun et al. [Lun et al. 2017] and the well-known *pix2pix* [Isola et al. 2017]. This work was later outperformed by [Hudon et al. 2018] which proposes a method to reconstruct high quality and high resolution normal maps, allowing for the creation of convincing illumination effects on 2D sketches and animations (comparable to high-end geometric methods such as TexToons [Sỳkora et al. 2011]).

This paper is an extension of the work presented in [Hudon et al. 2018]. Additional contributions are:

- A new High Resolution Dataset generated from 600+ 3D models of characters and creatures harvested from the Internet.
- An improved network design compared to [Hudon et al. 2018] leading to higher quality results.
- An inflation method based on [Nehab et al. 2005] to generate meshes from the predicted normal maps.
- Comparisons to most recent and successful state of the art.

3 PROPOSED TECHNIQUE

The presented method aims to generate a 3D proxy surface mesh from a single drawing for the accurate rendering of global illumination effects. We aim for a completely automatic method with no user action required.

3.1 Normal Map Prediction

3.1.1 New Dataset. Due to the lack of existing high resolution datasets, we created our own that will be shared freely with the research community. We generated training triples, mimicking human drawing using non-photorealistic rendering (NPR) [Grabli et al. 2010] of 3D models as described in [Hudon et al. 2018]. We chose the NPR parameters so the resulting line drawings are not overly detailed and a little closer, in our opinion, to what an artist would draw compared to [Lun et al. 2017]. Each triple comprises of a line-drawing along with corresponding normal and depth maps. We collected 606 high-quality free of use 3D models belonging to the *characters and creatures* categories of several online 3D model

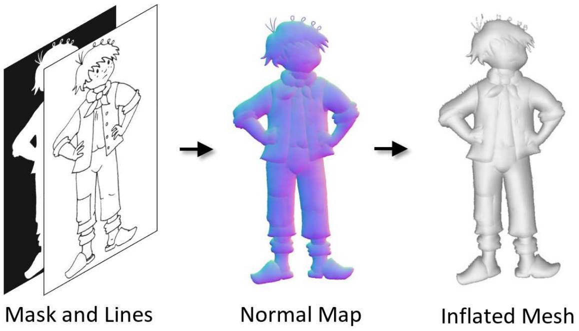


Figure 1: Our method takes as input a line drawing and a mask, then automatically predicts a high quality and high resolution normal map. The normal map is then inflated to a 3D surface mesh.

sharing websites. Noticing that already available datasets contain mainly characters in T-poses, our dataset also contains parts of or complete *characters and creatures* in more natural postures. We split the dataset into two sets of 590 models for training and 16 for testing. For each model we generated multiple triples (rendered drawing, normal and depth maps) corresponding to different viewpoints (42 per model, see Figure 2) leading to a training set of 24780 triples and a testing set of 688 triples. In this paper, we only use the normal maps, depth maps were generated for future work.

3.1.2 Inputs. In [Hudon et al. 2018] it was confirmed that counting on the fully convolutional properties of one network to process inputs of higher resolution (than the training set) was not necessarily the way to get the most qualitative results. Instead, they process the full resolution input in patches. However, to transmit more information about the surrounding area, each patch contains 3 channels capturing a local area of interest at 3 different scales. This input representation also serves as a significant data augmentation. Indeed, by randomly taking 10 patches per drawing of the dataset (instead of 100 in [Hudon et al. 2018]), our number of training pairs rises to 247800. In this paper we additionally make use of a fourth channel containing a foreground/background mask (see Figure 3).

3.1.3 Network. We propose a new U-Net [Ronneberger et al. 2015] style encoder-decoder network that significantly improves the results presented in [Hudon et al. 2018]. Our network representation can be seen in Figure 4. We added a fourth channel containing the foreground mask (equivalent to an alpha channel) to the input, passing more information to the network to achieve more accurate predictions. Instead of using 2D convolutions with a stride > 1, we make use of dilated convolutions (a.k.a. Atrous convolutions) for the five last layers of the encoder. Atrous convolutions bring a significant gain to dense prediction tasks, such as semantic segmentation [Chen et al. 2014; Yu and Koltun 2015]. Dilating convolution kernels allows the network to learn long-distance features by virtually increasing the features receptive fields without decreasing the size of the feature maps and thus preserving spatial resolution through the network layers. All convolutions are followed by Leaky *ReLUs* with a slope of 0.3, except for the final convolution of the decoder which uses *tanH* as activation function. We further multiply the output of the last convolution layer with the binary mask of the input to remove any normals in the background that might have been created by the network. To train the network we kept the same loss function:

$$\mathcal{L} = \frac{\sum_p (1 - N_e(p) \cdot N_t(p)) \times \delta_p}{\sum_p \delta_p} \tag{1}$$

where $N_e(p)$ and $N_t(p)$ are the estimated and ground truth normals at pixel p respectively, and δ_p ensures that only foreground pixels are taken into account in the loss computation, being 0 whenever p is a background pixel and 1 otherwise. With unit length, Equation 1 nicely simplifies to:

$$\mathcal{L} = \frac{\sum_p \left(\frac{1}{2} \|N_e(p) - N_t(p)\|^2\right) \times \delta_p}{\sum_p \delta_p} \tag{2}$$

3.1.4 Final Normal Map Reconstruction. As the network was trained on $256 \times 256 \times 4$ input elements, high-resolution drawings have to be sampled into $256 \times 256 \times 4$ tiles for better results as shown in [Hudon et al. 2018]. These tiles are then passed through the network and outputs have to be combined together to form the expected high-resolution normal map. As shown in [Hudon et al. 2018], direct naive tile reconstructions can lead to inconsistent, blocky normal maps, which are not suitable for adding high-quality shading effects to sketches. In order to overcome this issue, we use a multi-grid diagonal sampling strategy as shown in Fig. 5(e). Rather than processing only one grid of tiles, we process multiple overlapping grids, as shown in Fig. 5(e). Each new grid is created by shifting the original grid (Fig. 5(d)) diagonally, each time by a different offset. Then at every pixel location, the predicted normals are averaged together to form the final normal map, as shown in Fig. 5(c). The use of diagonal shifting is an appropriate way to wipe away the blocky (mainly horizontal and vertical) sampling artifacts seen in Fig. 5(b) when computing the final normal map. Increasing the number of grids also improves the accuracy of the normal estimation, however, the computational cost also increases with the number of grids.

3.2 Inflation

For the inflation we were greatly inspired by [Nehab et al. 2005]. Under single view and orthographic assumptions, a 3D point cloud $\mathbf{P}$ can be created using predicted depth values $\mathbf{y}_{u,v}$, where each 3D point $\mathbf{P}_{u,v}$ corresponds to a pixel position (u, v) as follows:

$$\mathbf{P}_{u,v} = [u, v, \mathbf{y}_{u,v}]^T \tag{3}$$

Each 3D point $\mathbf{P}_{u,v}$ can also be associated with a predicted normal vector $\mathbf{n}_{u,v}$. To turn the predicted normal map into a surface mesh, we inflate a depth map $\mathbf{y}$ (initialized to zeros) such that the first order depth derivatives yield surface tangents as close as possible to the predicted normals $\mathbf{n}_{u,v}$.

Given a pixel at position (u, v) and its depth $y_{u,v}$, two surface tangents can be estimated based on first order depth derivatives:

$$\mathbf{t}^{(u)}_{u,v} = \begin{bmatrix} 1 & 0 & \gamma \frac{\partial y_{u,v}}{\partial u} \end{bmatrix}, \quad \mathbf{t}^{(v)}_{u,v} = \begin{bmatrix} 0 & 1 & \gamma \frac{\partial y_{u,v}}{\partial v} \end{bmatrix}, \tag{4}$$

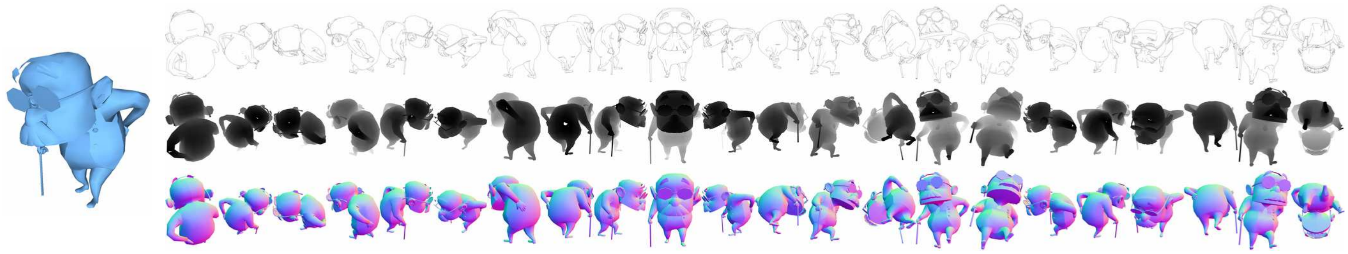


Figure 2: Example of triples corresponding to half the views generated for one model. (left) Input 3D model, (top) NPR line-drawings, (middle) depth maps, and (bottom) normal maps.

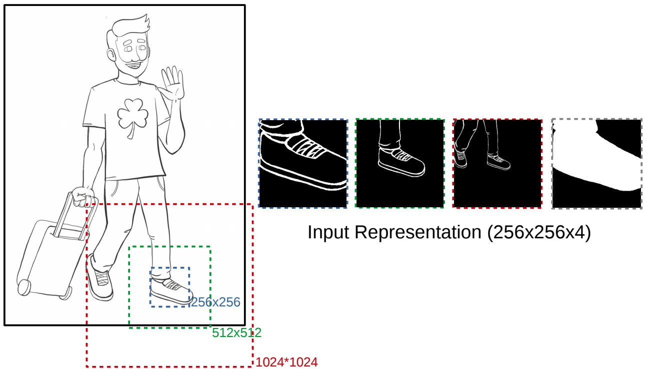


Figure 3: Structure of the input data. Here the target normal reconstruction scale is the blue channel. The two other channels provide additional multi-scale representation of the local area, the last dimension is the foreground/background mask. Figure partly taken from [Hudon et al. 2018].

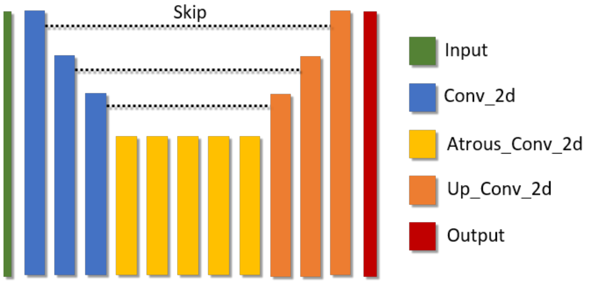


Figure 4: Our presented, U-Net style encoder-decoder network.

where γ controls the inflation scale. The term $E_n(\tilde{y})$ penalizes the deviation from orthogonality between surface tangents and the predicted surface normals $\mathbf{n}_{u,v}$:

$$E_n(y) = \sum_{u,v} [(\mathbf{t}^{(u)}_{u,v} \cdot \mathbf{n}_{u,v})^2 + (\mathbf{t}^{(v)}_{u,v} \cdot \mathbf{n}_{u,v})^2] \tag{5}$$

To inflate our depth values, we minimize $E_n(y)$ using gradient descent [ADAM]. To compute the depth gradients we make use of two convolutions with the following kernels:

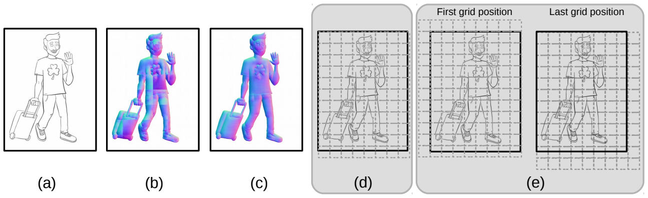


Figure 5: Normal map reconstruction of the input sketch (a) using a direct naive sampling (b) and our multi-grid diagonal sampling (c). Sampling grids used in direct naive sampling (d) and multi-grid diagonal sampling (e). Figure taken from [Hudon et al. 2018].

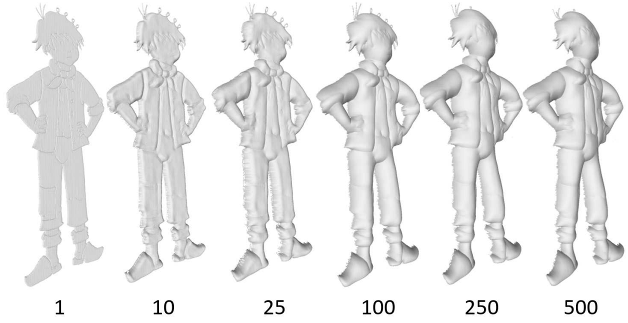


Figure 6: Different states of the inflated surface through the gradient descent iterations. 500 iterations are enough for the inflation process to fully converge.

$$\mathcal{K}_x = \frac{1}{12}\begin{bmatrix} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{bmatrix}, \mathcal{K}_y = \frac{1}{12}\begin{bmatrix} 1 & 4 & 1 \\ 0 & 0 & 0 \\ -1 & -4 & -1 \end{bmatrix} \tag{6}$$

Different states of the reconstructed surface through the gradient descent iterations can be seen Figure 6. The gradient descent process sometimes leaves the surfaces noisy, therefore we clean the result using fast guided filtering [He and Sun 2015].

3.3 Rendering

Given a reconstructed surface proxy mesh, the input drawing can be augmented with global illumination effects. Any rendering engine

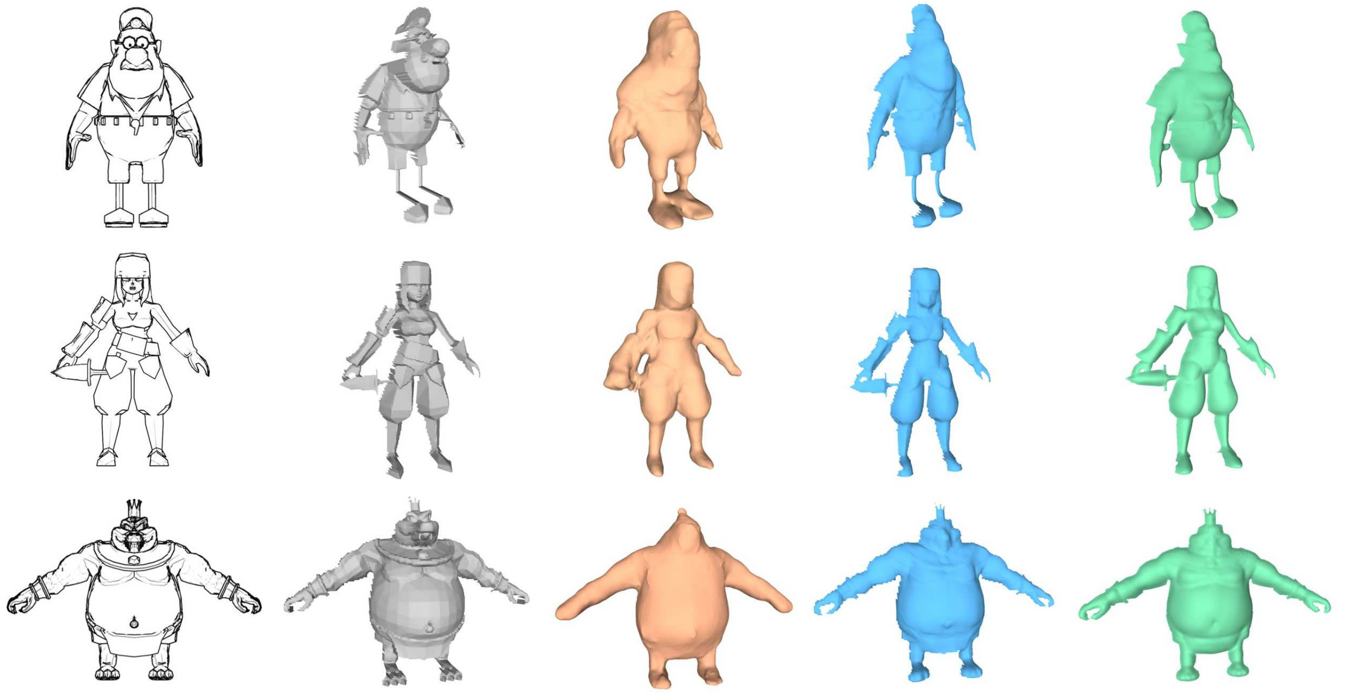


Figure 7: Left to right: Input drawing from the characters dataset publish by Lun et al. [Lun et al. 2017], ground truth model, Lun et al. [Lun et al. 2017], Li et al [Li et al. 2018], proposed.

can be used at this point. For the examples presented in this paper, we make use of Blender Cycles©rendering engine. For our testing needs, we created a simple interface between blender and the open source animation software Pencil2D. In our implementation the user can easily assign different reflectance properties to flat colors such as diffuse, glossy or both. The user is able to control the rendering parameters directly in Pencil2D including: the light position, color and power (a surface area light in our implementation), the amount of ambient light, the glossy exponent. We also let the user choose the final quality of the rendering. The default quality is low, which allows real-time rendering to efficiently set the rendering parameters. The quality can be increased for the final render.

4 EVALUATION

The whole pipeline from the drawing to the surface inflation was implemented using Python and Tensorflow. The following table shows the processing times for a 1000*x*1000 image:

Table 1: Timings

Total Normal Prediction - 20 Grids	4 sec
Inflation	1.3 sec

In general a user can obtain a surface mesh from a drawing in approximately 5 seconds, decreasing the number of grids can also reduce the processing time at the cost of quality.

Table 2: Improvements

	[Hudon et al. 2018]	[Hudon et al. 2018] (re-trained)	Proposed
L1	0.254	0.223	**0.209**
L2	0.300	0.264	**0.247**
Angular	29.799	26.375	**24.163**

4.1 CNN

4.1.1 Improvements. As an extension of Hudon et al. [Hudon et al. 2018], we present a full comparison showing the improvements made. Note that [Hudon et al. 2018] already presented a full comparison with [Su et al. 2018]. For a meaningful comparison with [Hudon et al. 2018], we also used a fourth input channel containing the foreground/background mask to their network for all presented comparisons. We show that not only does our new dataset allow us to train more qualitative networks but also that our improvement of the CNN leads to finer predictions.

The previous table 2 shows numerical errors higher than the ones presented in [Hudon et al. 2018], this is mainly due to our more challenging test dataset.

4.1.2 Additional Comparisons. We compare our CNN performance with the approaches presented in [Lun et al. 2017] and [Li et al. 2018], two of the most recent and interesting systems in the field. In their paper, Li et al. [Li et al. 2018] compared to [Lun et al. 2017], and presented a procedure for a meaningful comparison. The authors of [Li et al. 2018] kindly provided the reconstructed models used for comparison in their paper, (see [Li et al. 2018] for complete details). We use the characters datasets (front-view) published by [Lun et al. 2017] for re-training and testing the proposed network. In this dataset input drawings contain more geometrical information and seem less natural, however reconstructions are capturing more details. For uniformity we downscaled our resulting normal maps before computing the angular normal errors on the test dataset to compare them to the results presented in [Li et al. 2018]:

Table 3: Comparison with recent SOA

	Lun et al. [Lun et al. 2017]	Li et al. [Li et al. 2018]	Proposed
Angular	22.4	18.6	**13.8**

A visual comparison of the model reconstruction from [Lun et al. 2017] and [Li et al. 2018] with our inflated models can be seen in Figure 7 (Low poly ground truth normal maps were smoothed for training). Without any user input, our method seems to perform as well as [Li et al. 2018].

4.2 Visual comparisons

We show qualitative comparisons with the most relevant state of the art aiming at augmenting 2D art or animation with shading or global illumination effects: Ink and Ray [Sỳkora et al. 2014], which can be seen in Figure 8. The Ink and Ray pipeline [Sỳkora et al. 2014] produces very qualitative results at the cost of extensive user inputs (Figure 8(b-d)) whereas our method is fully automatic (Note that the mask in Figure 8(g) can be easily and automatically computed from a color image). As Ink and Ray is based on geometric inflation only, features such as cloth folds or buttons are not present in the final mesh, whereas the inflation step in our method allows us to capture more details. However, the more detailed input that is required by Ink and Ray makes it possible to model real discontinuities in the final reconstruction, which are important for global illumination effects such as self shadowing. The fully automatic reconstruction in our method doesn't model these, but is much faster and therefore closer to real world animation production needs.

To demonstrate the versatility of our method we show some additional visual results in Figure 10 for various drawing styles.

4.3 PENCIL 2D© Integration

For demonstration purposes only, we implemented our own version of pencil 2D able to generate the surfaces and render global illuminations effects on 2D hand-drawn animations. Screen-shots of the application can be seen on Figure 9. This implementation is just an example to showcase what could be done with our method. We have tried to remain as close as possible to the 2D framework. In this case the user can assign reflectance properties to plain colors (diffuse, glossy or both) (See Figure 9(b)). We also give the possibility to place a horizontal shadow catcher (Figure 9(c)) with a right-click on the canvas (when our tool is selected). Once the surfaces have been generated, ©Blender is launched as a back process and allows us to generate the rendered frames. The quality of the renders are low by default so the user can set the light properties with real time feedback (Figure 9(e-f)). Once the user is satisfied with the lighting she or he can increase the rendering quality and render all frames (Figure 9(h)).

5 DISCUSSION AND CONCLUSION

In this paper we presented an improvement of [Hudon et al. 2018]: a new method to apply high quality illumination effects on line drawings, using accurate 3D reconstruction of the object. We are convinced that reconstructing 3D shapes from single sketches is fundamentally a learning problem and that recent advancements in deep learning will and already have a large impact on this field. Therefore, our first contribution is a new dataset consisting of over 25000 line drawings with corresponding ground-truth normal and depth maps. We also present an improvement of the neural network presented in [Hudon et al. 2018] able to accurately predict normal maps from line drawings and a fusion process to generate a detailed 3D reconstruction suitable for adding global illumination effects on a 2D drawing.

The presented method is significantly lighter and simpler to use than all presented state of the art. Yet we show that the final quality is equivalent. We show quantitative results regarding the accuracy of our normal map predictions, where we outperform the current state of the art. Additionally, we show and discuss the quality of our results by comparing them qualitatively with results from recent state of the art.

The main strength of the presented method is that no user input is required while most of previous works depends more or less heavily on such inputs. The high-quality of our reconstructions leads to convincing shading effects in a fully automatic manner, and therefore could be easily integrated in real animation production pipelines, reducing drastically the human labour. For our testing needs, we integrated the presented pipeline into a open source software of animation Pencil2D ©. While this integration is mainly for research purposes, this still demonstrates the usability of our tool. As an example, all results, drawings and animations, presented in Figure 10 were generated without leaving Pencil 2D.

While the results presented in this paper are really promising, there are still some issues that we wish to point out. First, in the current implementation, the CNN cannot handle concave surfaces, although they are present in the datatset. Perhaps a specific loss penalizing false non-concave results could be added for training. Second, while the reconstructed meshes show some details, we feel that the discontinuities between different elements of a character are often too small, probably due to the bas-relief ambiguity. The lack of sharpness and gaps in discontinuities are penalizing for self shadowing effects to really take place. While discontinuities could be easily emphasized by inputting sparse depth inequalities such as in [Sỳkora et al. 2011], we feel that searching for a solution without any user input is more appealing. Finally, this cannot be seen on Figure 10, but a small flickering can be observed when processing animations. We think that spatio-temporal filtering could be used to solve this issue.

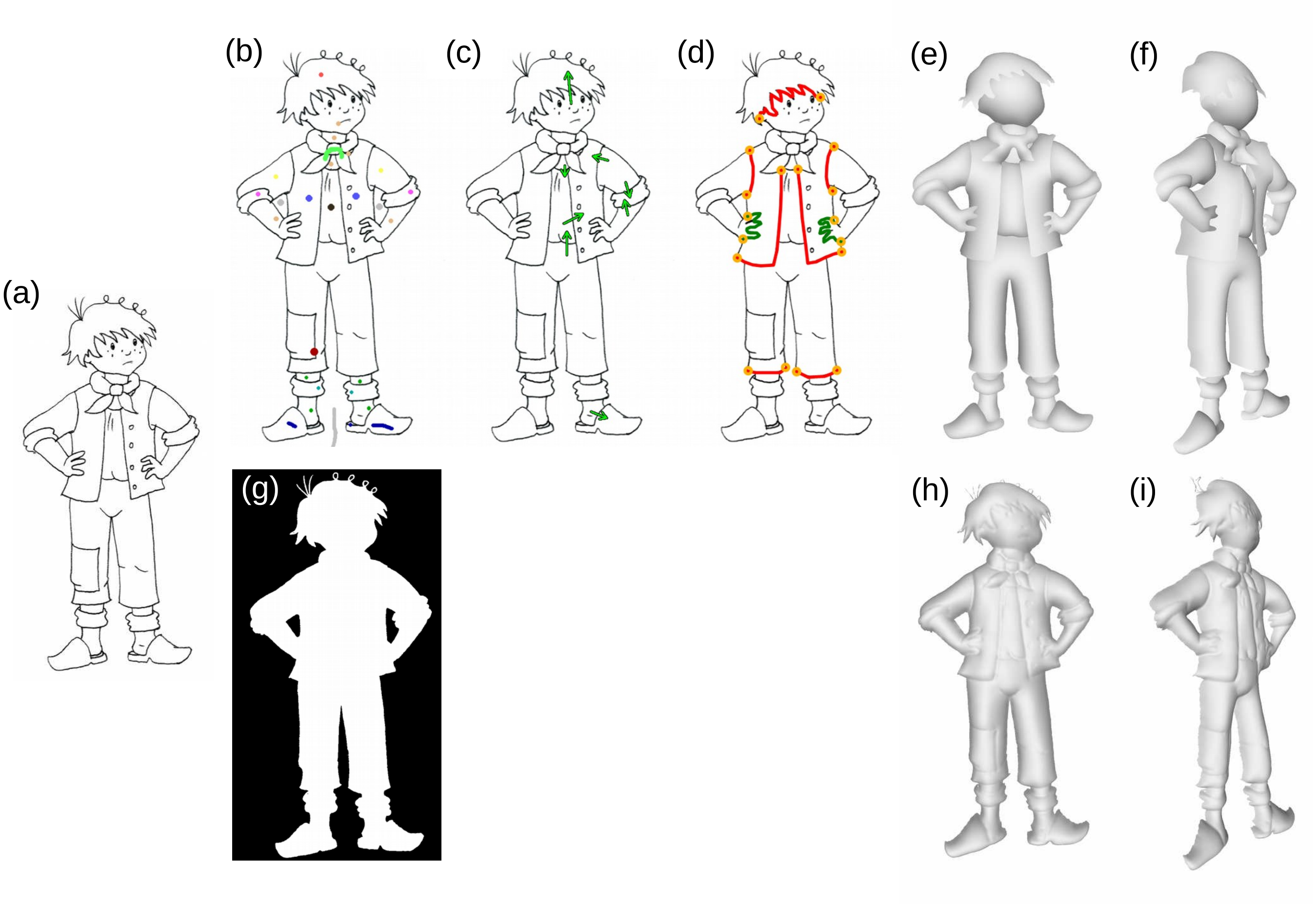


Figure 8: Comparison of reconstructed meshes with Ink and Ray [Sỳkora et al. 2014]. (a) input, (b-d) additional user input for [Sỳkora et al. 2014] (images taken from [Sỳkora et al. 2014]) , (e,f) result from [Sỳkora et al. 2014] (provided by the authors), (g) mask needed for our method (note that in our implementation the mask is automatically obtained from the color image), (h,i) our result.

ACKNOWLEDGMENTS

The authors would like to thank David Revoy and Ester Huete, for sharing their original creations. This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant Number 15/RP/2776. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU used for this research.

REFERENCES

Aayush Bansal, Bryan Russell, and Abhinav Gupta. 2016. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 5965–5974.

Peter N Belhumeur, David J Kriegman, and Alan L Yuille. 1999. The bas-relief ambiguity. *International journal of computer vision* 35, 1 (1999), 33–44.

Minh Tuan Bui, Junho Kim, and Yunjin Lee. 2015. 3D-look shading from contours and hatching strokes. *Computers & Graphics* 51 (2015), 167–176.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2014. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062* (2014).

Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. 2009. How well do line drawings depict shape?. In *ACM Transactions on Graphics (ToG)*, Vol. 28. ACM, 28.

Marek Dvorožňák, Saman Sepehri Nejad, Ondřej Jamriška, Alec Jacobson, Ladislav Kavan, and Daniel Sỳkora. 2018. Seamless reconstruction of part-based high-relief models from hand-drawn images. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering.* ACM, 5.

David Eigen and Rob Fergus. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision.* 2650–2658.

David Eigen, Christian Puhrsch, and Rob Fergus. 2014. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems.* 2366–2374.

Stéphane Grabli, Emmanuel Turquin, Frédo Durand, and François X Sillion. 2010. Programmable rendering of line drawing from 3D scenes. *ACM Transactions on Graphics (TOG)* 29, 2 (2010), 18.

Xiaoguang Han, Chang Gao, and Yizhou Yu. 2017. DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling. *arXiv preprint arXiv:1706.02042* (2017).

Kaiming He and Jian Sun. 2015. Fast guided filter. *arXiv preprint arXiv:1505.00996* (2015).

Haibin Huang, Evangelos Kalogerakis, Ersin Yumer, and Radomir Mech. 2017. Shape synthesis from sketches via procedural models and convolutional networks. *IEEE transactions on visualization and computer graphics* 23, 8 (2017), 2003–2013.

Matis Hudon, Mairéad Grogan, Rafael Pagés, and Aljoša Smolić. 2018. Deep Normal Estimation for Automatic Shading of Hand-Drawn Characters. In *European*

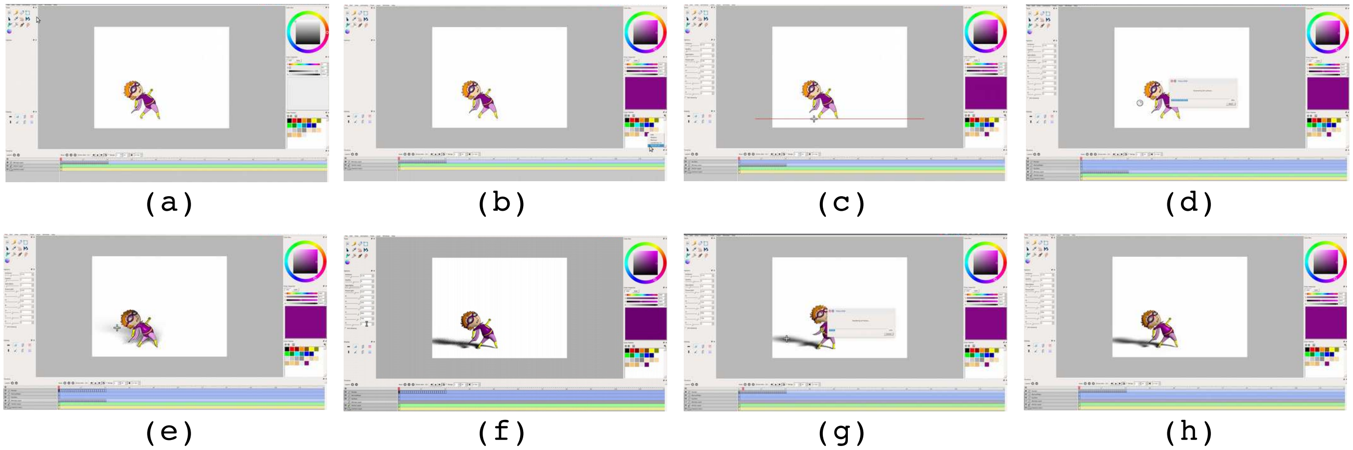


Figure 9: Our implementation interface. (a) Interface with loaded animation, (b) user can set for each color diffuse property or glossy property or both, here the user sets the purple color to glossy and diffuse, (c) optional floor (shadow catcher can be set directly), (d) generating surfaces and rendering all frames with low quality, (e,f) thanks to the low quality of rendering the user can adjust the lighting properties with real-time feedback, (g) final high quality rendering of all frames, (h) final first frame.

Conference on Computer Vision. Springer, 246–262.

T Igarashi, S Matsuoka, and H Tanaka. 1999. Teddy: A Sketching Interface for 3D Freeform Design, SIGGRAPH âĂŸ99. In *Conference Proceedings), ACM.*

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Pradeep Kumar Jayaraman, Chi-Wing Fu, Jianmin Zheng, Xueting Liu, and Tien-Tsin Wong. 2017. Globally Consistent Wrinkle-Aware Shading of Line Drawings. *IEEE Transactions on Visualization and Computer Graphics* (2017).

Scott F Johnston. 2002. Lumo: illumination for cel animation. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*. ACM, 45–ff.

Olga A Karpenko and John F Hughes. 2006. SmoothSketch: 3D free-form shapes from complex sketches. In *ACM Transactions on Graphics (TOG)*, Vol. 25. ACM, 589–598.

Jan J Koenderink, Andrea J Van Doorn, and Astrid ML Kappers. 1992. Surface perception in pictures. *Attention, Perception, & Psychophysics* 52, 5 (1992), 487–496.

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

Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. 2018. Robust flow-guided neural prediction for sketch-based freeform surface modeling. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 238.

Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 2017. 3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks. *arXiv preprint arXiv:1707.06375* (2017).

Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. 2005. Efficiently combining positions and normals for precise 3D geometry. *ACM transactions on graphics (TOG)* 24, 3 (2005), 536–543.

Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. 2009. Sketch-based modeling: A survey. *Computers & Graphics* 33, 1 (2009), 85–103.

Hao Pan, Yang Liu, Alla Sheffer, Nicholas Vining, Chang-Jian Li, and Wenping Wang. 2015. Flow aligned surfacing of curve networks. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 127.

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

Jhony K Pontes, Chen Kong, Sridha Sridharan, Simon Lucey, Anders Eriksson, and Clinton Fookes. 2017. Image2Mesh: A Learning Framework for Single Image 3D Reconstruction. *arXiv preprint arXiv:1711.10669* (2017).

Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Efstratios Gavves, and Tinne Tuytelaars. 2016. Deep reflectance maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4508–4516.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 234–241.

Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009. Analytic drawing of 3D scaffolds. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 149.

Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. 2012. CrossShade: Shading Concept Sketches Using Cross-Section Curves. *ACM Transactions on Graphics* 31, 4 (2012). https://doi.org/10.1145/2185520.2185541

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2017. Mastering Sketching: Adversarial Augmentation for Structured Prediction. *arXiv preprint arXiv:1703.08966* (2017).

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 121.

Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and FU Hongbo. 2018. Interactive Sketch-Based Normal Map Generation with Deep Neural Networks. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2018)*. ACM.

Daniel Sỳkora, Mirela Ben-Chen, Martin Čadík, Brian Whited, and Maryann Simmons. 2011. TexToons: practical texture mapping for hand-drawn cartoon animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*. ACM, 75–84.

Daniel Sỳkora, John Dingliana, and Steven Collins. 2009a. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*. ACM, 25–33.

Daniel Sỳkora, John Dingliana, and Steven Collins. 2009b. LazyBrush: Flexible Painting Tool for Hand-drawn Cartoons. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 599–608.

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

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2016. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*. Springer, 322–337.

Bui Minh Tuan, Junho Kim, and Yunjin Lee. 2017. Height-field Construction using Cross Contours. *Computers & Graphics* (2017).

Xiaolong Wang, David Fouhey, and Abhinav Gupta. 2015. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 539–547.

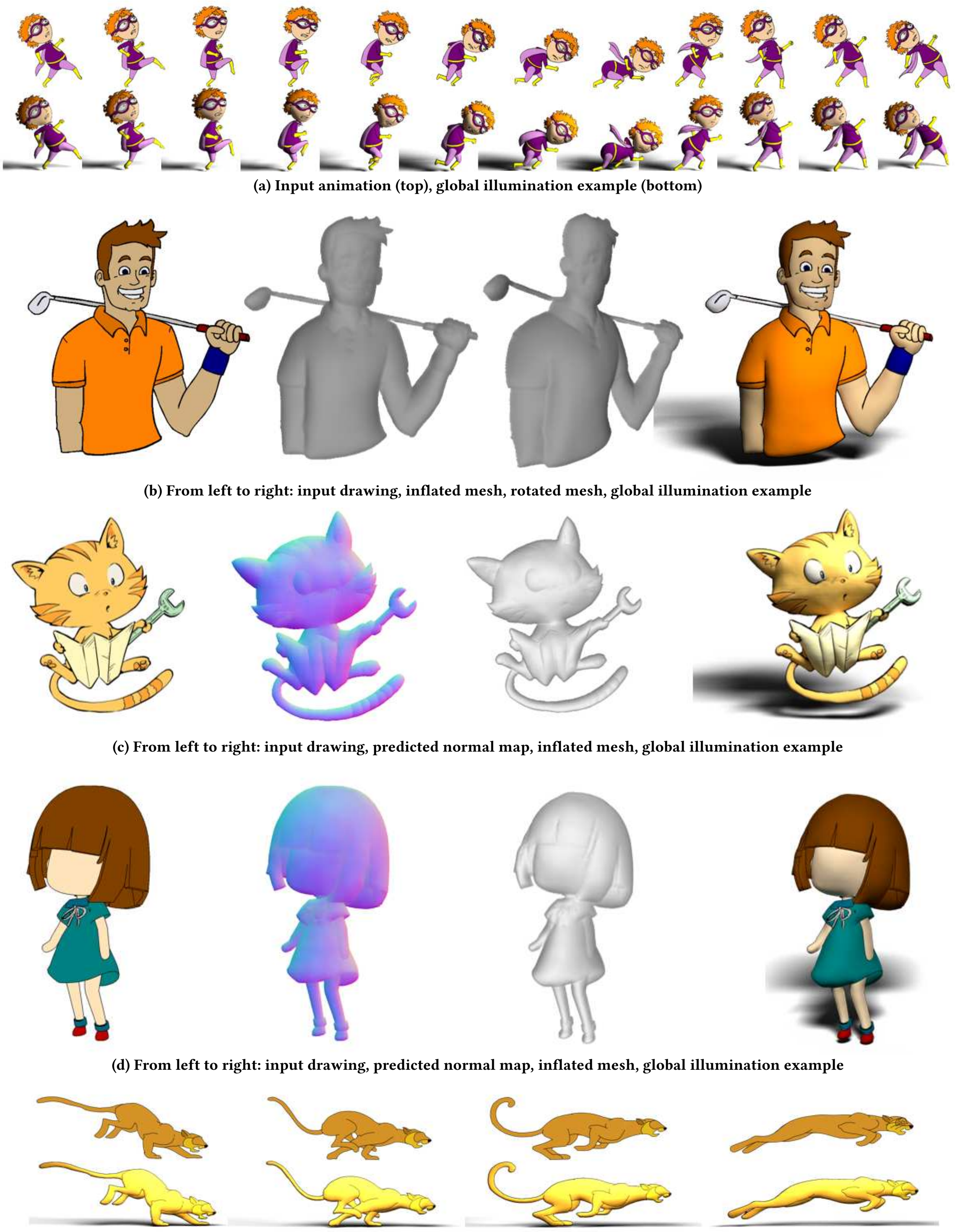
Brian Whited, Gioacchino Noris, Maryann Simmons, Robert W Sumner, Markus Gross, and Jarek Rossignac. 2010. Betweenit: An interactive tool for tight inbetweening. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 605–614.

Jun Xing, Li-Yi Wei, Takaaki Shiratori, and Koji Yatani. 2015. Autocomplete hand-drawn animations. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 169.

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Transactions on Graphics* 33, 4 (2014).

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).

Lvmin Zhang, Yi Ji, and Xin Lin. 2017. Style Transfer for Anime Sketches with Enhanced Residual U-net and Auxiliary Classifier GAN. *arXiv preprint arXiv:1706.03319* (2017).



(a) Input animation (top), global illumination example (bottom)

(b) From left to right: input drawing, inflated mesh, rotated mesh, global illumination example

(c) From left to right: input drawing, predicted normal map, inflated mesh, global illumination example

(d) From left to right: input drawing, predicted normal map, inflated mesh, global illumination example

(e) Input animation (top), global illumination example (bottom)

Figure 10: Results of artworks and animations augmented using our method. Note that we made used of physically realistic area lights therefore shadows are not hard-shadows. The shadow catcher was implemented for demonstration purposes and is placed horizontally, only the vertical position can be customized.