Deep Normal Estimation for Automatic Shading of Hand-Drawn Characters

Matis Hudon, Mairéad Grogan, Rafael Pagés, and Aljoša Smolić

V-SENSE, Trinity College Dublin, Dublin, Ireland
{hudonm,mgrogan,pagesscr,smolica}@scss.tcd.ie
https://v-sense.scss.tcd.ie/

Abstract. We present a new fully automatic pipeline for generating shading effects on hand-drawn characters. Our method takes as input a single digitized sketch of any resolution and outputs a dense normal map estimation suitable for rendering without requiring any human input. At the heart of our method lies a deep residual, encoder-decoder convolutional network. The input sketch is first sampled using several equally sized 3-channel windows, with each window capturing a local area of interest at 3 different scales. Each window is then passed through the previously trained network for normal estimation. Finally, network outputs are arranged together to form a full-size normal map of the input sketch. We also present an efficient and effective way to generate a rich set of training data. Resulting renders offer a rich quality without any effort from the 2D artist. We show both quantitative and qualitative results demonstrating the effectiveness and quality of our network and method.

Keywords: Cartoons · Non-photorealistic rendering · Normal estimation · Deep learning

1 Introduction

Despite the proliferation of 3D animations and artworks, 2D drawings and hand-drawn animations are still important art communication media. This is mainly because drawing in 2D is not tied to any constraining tools and brings the highest freedom of expression to artists. Artists usually work in three steps: firstly, they create the raw animation or drawing which includes finding the right scene composition, character posture, and expression. Secondly, they refine the artwork, digitalise it and clean the line-art. Finally, they add color or decorative textures, lights and shades. When working on numerous drawings some of these steps can become quite 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 [36,37], scanning [24], coloring [42,52], and by developing image registration and inbetweening techniques [41,49,50].

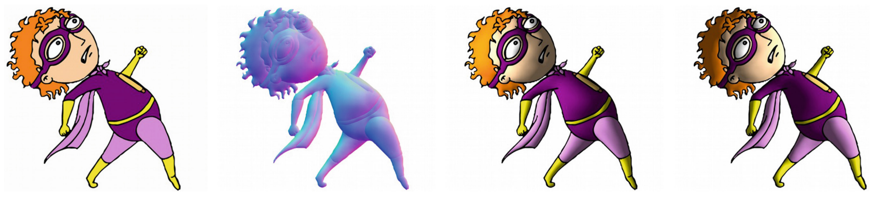

Fig. 1. Our method takes as input a drawing of any resolution and estimates a plausible normal map suitable for creating shading effects. From left to right: Input drawing and flat colors, normal estimation and two renderings with different lighting configurations.

In this paper, we consider the shading task. Besides bringing appeal and style to animations, shades and shadows provide important visual cues about depth, shape, movement and lighting [19,29,48]. Manual shading can be challenging as it requires not only a strong comprehension of the physics behind the shades but also, in the case of an animation, spatial and temporal consistency within and between the different frames. The two basic components required to calculate the correct illumination at a certain point are the light position with respect to the point and the surface normal. These normals are unknown in hand-drawn artwork. Several state-of-the-art approaches have tried to reconstruct normals and/or depth information directly from line-drawing [9,12,14,17,26,29,40,43], however, most of these works seem to be under-constrained or require too many user inputs to really be usable in a real-world artistic pipeline.

We propose a method to estimate high-quality and high-resolution normal maps suitable for adding plausible and consistent shading effects to sketches and animations (Fig. 1). Unlike state-of-the-art methods, our technique does not rely on geometric assumptions or additional user inputs but works directly, without any user interaction, on input line-drawings. To achieve this, we have built a rich dataset containing a large number of training pairs. This dataset includes different styles of characters varying from cartoon ones to anime/manga. To avoid tedious and labor intensive manual labelling we also propose a pipeline for efficiently creating a rich training database. We introduce a deep Convolutional Neural Network (CNN) inspired by Lun et al. [26] and borrow ideas from recent advances such as symmetric skipping networks [32]. The system is able to efficiently predict accurate normal maps from any resolution input line drawing. To validate the effectiveness of our system, we show qualitative results on a rich variety of challenging cases borrowed from real world artists. We also compare our results with recent state-of-the-art and present quantitative validations. Our contributions can be summarized as follows:

- We propose a novel CNN pipeline tailored for predicting high-resolution normal maps from line-drawings.
- We propose a novel tiled and multi-scale representation of input data for efficient and qualitative predictions.

- We propose a sampling strategy to generate high-quality and high resolution normal maps and compare to recent CNNs including a fully convolutional network.

2 Related Work

Before introducing our work, we first review existing methods on shape from sketches. They can be classified into two categories: geometry-based methods and learning-based methods.

2.1 Inferring 3D Reconstruction from Line Drawings

Works like Teddy [15] provide interactive tools for building 3D models from 2D data by "inflating" a drawing into a 3D model. Petrovic's work [29] applies this idea to create shades and shadows for cel animation. While this work reduces the labor of creating the shadow mattes compared to traditional manual drawing, it also demonstrates that a simple approximation of the 3D model is sufficient for generating appealing shades and shadows for cel animation. However, it still requires an extensive manual interaction to obtain the desired results. Instead of reconstructing a 3D model, Lumo [17] assumes that normals at the drawing outline are coplanar with the drawing plane and estimates surface normals by interpolating from the line boundaries to render convincing illumination. Olsen et al. [27] presented a very interesting survey for the reconstruction of 3D shapes from drawings representing smooth surfaces. Later, further improvements were made such as handling T-junctions and cups [18], also using user drawn hatching/wrinkle strokes [6,16] or cross section curves [35,46] to guide the reconstruction process. Recent works exploit geometric constraints present in specific types of line drawings [28,34,51], however, these sketches are too specific to be generalized to 2D hand-drawn animation. In TexToons [40], depth layering is used to enhance textured images with ambient occlusion, shading, and texture rounding effects. Recently, Sỳkora et al. [43] apply user annotation to recover a bas-relief with approximate depth from a single sketch, which they use to illuminate 2D drawings. This method clearly produces the best results, though it is still not fully automatic and still requires some user input. While these state-of-the-art methods are very interesting, we feel that the bas-relief ambiguity has not yet been solved. High-quality reconstructions require considerable user effort and time, whereas efficient methods rely on too many assumptions to be generalized to our problem. Although the human brain is still able to infer depth and shapes from drawings [5,7,21], this ability still seems unmatched in computer graphics/vision using geometry-based methods.

2.2 Learning Based Methods

As pure geometric methods fail to reconstruct high-quality 3D from sketches or images without a large number of constraints or additional user input, we are not the first to think that shape synthesis is fundamentally a learning problem. Recent works approach the shape synthesis problem by trying to predict surface depth and normals from real pictures using CNNs [8,31,47]. While these works show very promising results, their inputs provide much more information about the scene than drawn sketches, such as textures, natural shades and colors. Another considerable line of work employs parametric models (such as existing or deformed cars, bikes, containers, jewellery, trees, etc.) to guide shape reconstruction through CNNs [3,13,30]. Recently, Han et al. [11] presented a deep learning based sketching system relying on labor efficient user inputs to easily model 3D faces and caricatures. The work of Lun et al. [26], inspired by that of Tatarchenko et al. [44], is the closest to our work. They were the first to use CNNs to predict shape from line-drawings. While our method and network were inspired by their work, it differs in several ways: their approach makes use of multi-view input line-drawings whereas ours operates on a single input drawing, which allows our method to also be used for hand-drawn animations. Moreover, we present a way of predicting high-resolution normal maps directly, while they only show results for predicting 256 × 256 sized depth and normal maps and then fusing them into a 3D model. They provide a detailed comparison between view-based and voxel-based reconstruction. More recently Su et al. [39] proposed an interactive system for generating normal maps with the help of deep learning. Their method produces relatively high quality normal maps from sketch input combining a Generative Adversarial Network framework together with user inputs. However, the reconstructed models are still low resolution and lack details. The high-quality and high-resolution of our results allow us to qualitatively compete with recent work, including animation and sketch inflation, for high quality shading (see Sect. 2.1).

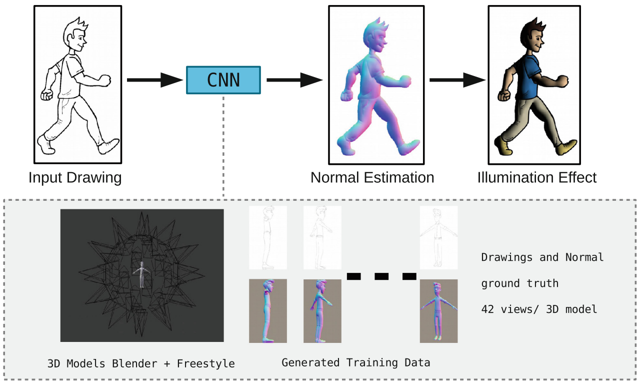

Fig. 2. System overview.

3 Proposed Technique

Figure 2 illustrates our proposed pipeline. The input to our system is a single arbitrary digital sketch in the form of a line drawing of a character. The output is an estimated normal map suitable for rendering effects, with the same resolution as the input sketch. The normals are represented as 3D vectors with values in the range $[-1, 1]$. The normal estimation relies on a CNN model trained only once offline.

The high resolution input image is split into smaller tiles/patches of size $256 \times 256 \times 3$, which are then passed through the CNN normal prediction and finally combined into a full resolution normal map. The training phase of the CNN requires a large dataset of input drawings and corresponding ground truth normal maps. As normal maps are not readily available for 2D sketches or animations, such a dataset cannot be obtained manually, and we therefore make use of 3D models and sketch-like renderings (Freestyle plugin in Blender ©) to generate the training dataset.

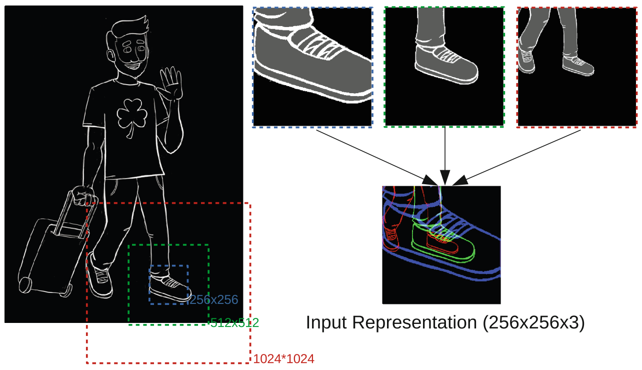

Fig. 3. Structure of the input data. Here the target normal reconstruction scale is the blue channel. The two other channels provide additional multi-scale representation of the local area

3.1 Input Preparation

One key factor that can improve the success of a CNN-based method is the preparation of input data. To take advantage of this, we propose a new data representation to feed to our network. Instead of simply feeding tiles of our input line drawing to the encoder, we feed in many multi-scale tile representations to the CNN, with each tile capturing a local area of interest of the sketch. This multi-scale representation prevents our network from missing important higher scale details. A multi-scale tile example is shown in Fig. 3. Each channel of the tile is used to represent the lines of the sketch in a local area at one of 3 different scales. In each channel, pixels on a line are represented by a value of 1. The first channel in each tile (blue channel in Fig. 3) represents the scale at which the normal vectors will be estimated.

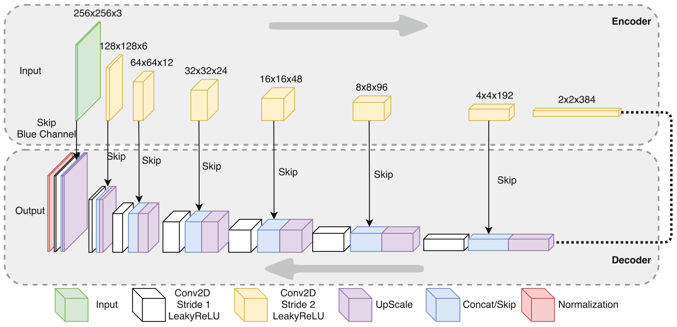

Fig. 4. Overall structure of the proposed network.

3.2 CNN Model

Our main requirements for the CNN is that the output has to be of the same size as the input and that extracted normals have to be aligned pixel-wise with the input line drawing. This is why a U-net based CNN is an appropriate choice. A typical pixel-wise CNN model is composed of two parts: an encoding branch and a decoding branch. Our proposed network is represented in Fig. 4, the top branch being the encoding network and the bottom branch the decoding network.

Encoder. The encoder, inspired by Lun et al. [26], compresses the input data into feature vectors which are then passed to the decoder. In the proposed network, this feature extraction is composed of a series of 2D convolutions, down-scaling and activation layers. The size of every filter is shown in Fig. 4. Rather than using max-pooling for down-scaling, we make use of convolution layers with a stride of 2 as presented by Springenberg et al. [38]. As our aim is to predict the normals, which take values in the range $[-1, 1]$, our activation functions also need to let negative values pass through the network. Therefore, each group of operations (convolution/down-scale), is followed by leaky ReLUs (slope 0.3), avoiding the dying ReLUs issue when training.

Decoder. The decoder is used to up-sample the feature vector so that it has the original input resolution. Except for the last block, the decoder network is composed of a series of block operations composed of up-sampling layers (factor of 2), 2D convolutions (stride 1, kernel 4) and activation layers (Leaky ReLUs with slope = 0.3). Similarly to U-Net [32] we make use of symmetric skipping between layers of the same scale (between the encoder and the decoder) to reduce the information loss caused by successive dimension reduction. To do so, each level of the decoder is merged (channel-wise) with the block of the corresponding encoder level. Input details can then be preserved in the up-scaling blocks. The last block of operations only receives the first channel (Blue channel in Fig. 3) for merging as the other two channels would be irrelevant at this final level of reconstruction. Our output layer uses a sigmoidal hyperbolic tangent activation function since normals lie in the range $[-1, 1]$. The last layer is an L_2 normalization to ensure unit length of each normal.

3.3 Learning

Training Data. Training a CNN requires a very considerable dataset. In our case, our network requires a dataset of line drawing sketches with corresponding normal maps. Asking human subjects to provide 2D line drawings from a 3D model would be too labour-intensive and time consuming. Instead we apply an approach similar to Lun et al. [26], automatically generating line drawings from 3D models using different properties such as silhouette, border, contour, ridge and valley or material boundary. We use the Freestyle software implemented for Blender©[10] for this process. Using Blender©scripts we were able to automatically create high-resolution drawings from 3D models (42 viewpoints per 3D model), along with their corresponding normal maps. As we generate our normal maps from 3D models alone, all background pixels have the same background normal value of $[-1, 0, 0]$. This will be used in the loss function computation (see Sect. 3.3). Tiling the input full resolution image into 256×256 tiles also provides a tremendous data augmentation. We extract 200 randomly chosen tiles from every drawing, making sure that every chosen tile contains sufficient drawing information. Therefore our relatively small dataset of 420 images (extracted from ten 3D models) is augmented to an 84000 elements training dataset.

Loss Function. During the training, as we are only interested in measuring the similarity of the foreground pixels, we use a specific loss function:

$$L = \sum_{p} (1 - N_e(p) \cdot N_t(p)) \times \delta_p, \quad (1)$$

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 (i.e. whenever $N_t = [-1, 0, 0]$) and 1 otherwise.

We train our model with the ADAM solver [20] (learning rate = 0.001) against our defined loss function. Finally, the learning process ends when the loss function converges.

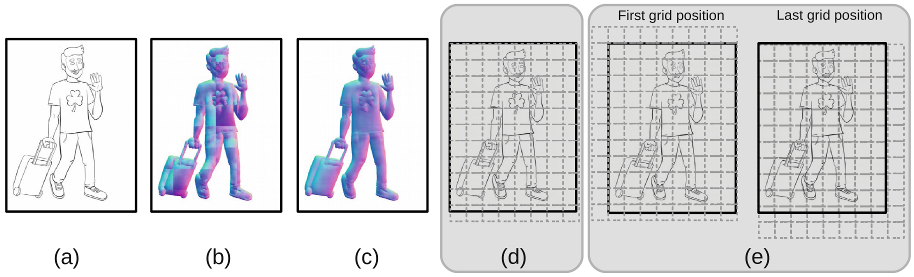

Fig. 5. Normal map reconstruction of the input sketch (a) using a direct naive sampling (b) and our multi-grid diagonal sampling (c). Sampling grids used in direct naive sampling (d) and multi-grid diagonal sampling (e).

3.4 Tile Reconstruction

As the network is designed to manage $256 \times 256 \times 3$ input elements, high-resolution drawings have to be sampled into $256 \times 256 \times 3$ tiles. These tiles are then passed through the network and outputs have to be combined together to form the expected high-resolution normal map. However, the deep reconstruction of normal maps is not always consistent across tile borders, and may be inaccurate when a tile misses important local strokes, as can be seen in Fig. 5(b). Direct naive tile reconstructions can therefore 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 propose 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. We therefore measured the root mean squared error (RMSE) of the output normal map versus ground truth depending on the number of grids and found that using 10 grids is a good compromise between quality of estimation and efficiency (<1 s for a $1K \times 2K$ px input image).

3.5 Texturing and Rendering

Since only the normal maps are predicted, and the viewing angle remains the same, properties from the input image such as flat-colors or textures can be directly used for rendering. Once textured or colorized, any image based rendering can be applied to the sketch, as in Fig. 6(c), such as diffuse lighting, specular

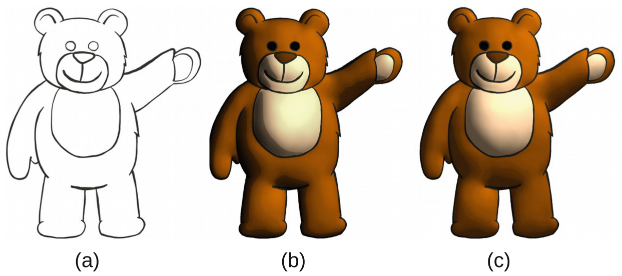

Fig. 6. Different types of shading: One can also employ a stylized toon shader such as [2,4,23,45], or classic non-photorealistic technique imitating global illumination such as diffuse and/or specular, Fresnel effect etc.

lighting or Fresnel effect (as presented in [33]). For a more stylized effect one can also apply a toon shader, as in Fig. 6(b).

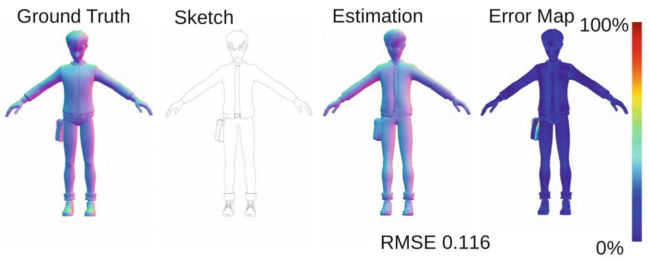

Fig. 7. Comparison of our normal estimation versus ground truth normal of a 3D model. From left to right: the ground truth normal map, the estimated sketch from the 3D model, our estimated normal map and finally the error map. This 3D model was not part of the training database.

4 Results and Discussion

In this section, we evaluate the full system using both qualitative and quantitative analysis. All the experiments are performed using line-drawings that are not included in our training database. Furthermore, as the training database only contains drawings with a line thickness of one pixel, we pre-process our input drawings using the thinning method proposed by Kwot [22].

We have implemented our deep normal estimation network using Python and Tensorflow [1]. The whole process takes less than 2.5 s for a 1220×2048 image running on an Intel Core i7 with 32 GB RAM and an Nvidia TITAN Xp. See Table 1 for a timing breakdown of each individual stage.

Table 1. Timings

Pre-processing	0.114 s
Single grid creation	0.012 s
Single grid prediction	0.054 s
Assembling grids	<0.1 s
Total time 1220 × 2048 - 10 Grids	2.328 s

Table 2. Tests results: Average error with L1, L2 and Angular metrics using our test database of 126 1K × 2K images, (FullyConv) Fully convolutional networks, (S2N) sketch to normals [39] with no user input on 256 × 256 images, (OursNoMS) our method without multi-scale input, (OursMS) our method.

Metric	FullyConv	S2N (256 × 256)	OursNoMS	OursMS
L1	0.244	0.227	0.208	**0.199**
L2	0.282	0.266	0.241	**0.231**
Angular	28.770	27.236	24.40	**23.468**

4.1 Quantitative Evaluation

We measure the accuracy of our method using a test database composed of 126 1K × 2K images, created with 3D models which are converted into sketches with the same non-photorealistic rendering technique we used to create our training database (see Sect. 3.3). This way, we can compare the result of our technique with the original normal ground truth of the 3D model which, of course, was not included in the training database.

Table 2 shows the numerical results of our different quantitative tests including our network with (OursMS) and without (OursNOMS) the multi-scale input, a fully convolutional network (FullyConv) trained on our database with patch-wise training [25]. We also trained the network from [39] with our database, however with the code provided we were only able to process images at low resolution. The results shown in Table 2 for this method (S2N) are for 256 × 256 images, hence input images and ground truth normal maps had to be re-sized beforehand. For every metric, our method (with multi-scale input) was the most accurate. Also note that using our network in a fully convolutional way to process images of high resolution is not as accurate as using our tiling technique.

Figure 7 shows a test sketch along with the ground truth normal map, our reconstructed normal map with 10 grids and a visual colored error map. While the overall root mean square error measured on the normal map is relatively low, the main inaccuracies are located on the feet and details of the face and the bag (which might be because there are no similar objects to the bag in our database). Also, as the reconstruction depends highly on the sketch strokes, minor variations and/or artistic choices in the input drawings can lead to different levels of accuracy. For example, artists commonly draw very simplified face details with very few strokes rather than realistic ones, which is especially visible in the nose, ears and eyes. In fact, the non-photorealistic method used to generate the sketches is already an estimation of what a sketch based on a 3D model could look like. We see this effect, for example, looking at the difference between the ground truth and predicted normals at the bottom of the jacket in Fig. 7, as some lines are missing on the sketch estimation; an artist might make different drawing choices.

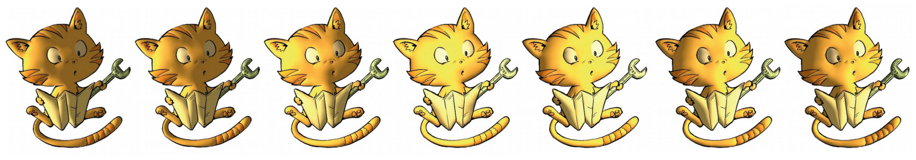
Fig. 8. Shading effects with different light configurations.

4.2 Visual Results

To show the versatility of our method we test it with a set of sketches with very different styles. Figure 12 presents results for four of these sketches: column (a) shows the original sketch, (b) the estimated normal map, (c) the flat-shaded sketch, and (d) shows the sketch after applying shading effects. As shown in column (b), normal maps present extremely fine details, which are correctly predicted in difficult areas such as fingers, cloth folds or faces; even the wrench, in the third row, is highly detailed. Column (d) shows the quality of shading that can be obtained using our system: final renders produce believable illumination and shading despite the absence of depth information. Once the normals are predicted, the user is able to render plausible shading effects consistently from any direction. This can be seen in Fig. 8, where we show several shading results using the same input drawing, by moving a virtual light around the cat.

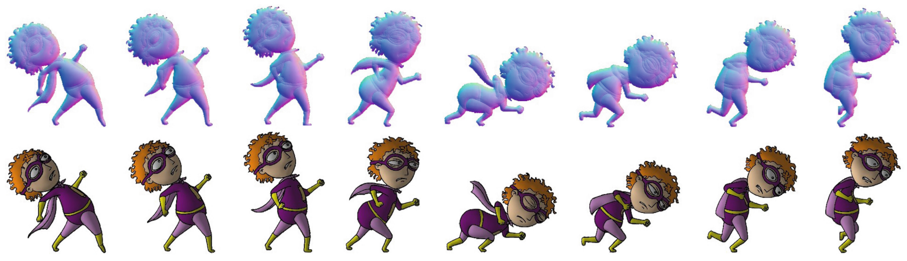
Fig. 9. Normals obtained with our method (top) and possible shading (bottom).

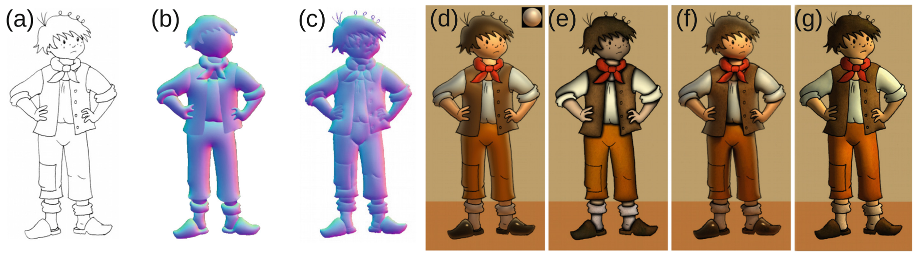

Fig. 10. Comparison with state-of-the-art methods. Input drawing (a), normals from the 3D reconstruction by [43] (b), our predicted normals (c), normal map based shading using Lumo [17] (d), 3D-like shading used in TexToons [40] (e), 3D reconstruction with global illumination effects [43] (e), Our approach (f). (3D model, line drawing and shading results from [43] kindly provided by the authors. Source drawing ©Anifilm. All rights reserved.)

Moreover, our method can be applied to single drawings as well as animations, without the need of any additional tools, as our method can directly and automatically generate normal maps for each individual frame of an animation. Our normal prediction remains consistent across all frames without the need to explicitly address temporal consistency or add any spatiotemporal filters. Figure 9 shows how shading is consistent and convincing along the animated sequence. As our normal vectors are estimated using training data, errors can occur in areas of sketches that are not similar to any object found in the database, such as the folded manual in the third row of Fig. 12. Such artifacts can be minimised by increasing the variety of objects captured by the training data. Furthermore, when boundary conditions are only suggested rather than drawn in the input drawing, it can result in unwanted smooth surface linking elements. An example of this effect can be seen between the cat's head and neck (Fig. 12, third row). The Ink-and-Ray approach presented in [43] handles such C^0 and C^1 boundary conditions (sparse depth inequalities, grafting, etc.) at the cost of additional user input. Finally, while most strokes in the sketch enhance the normal vector estimation, such as those around cloth folds, others only represent texture, and *texture copying* artifacts may appear when these are considered for normal estimation. However, an artist using our tool could easily avoid such unnecessary texture copying artifacts by drawing texture on a sepa-

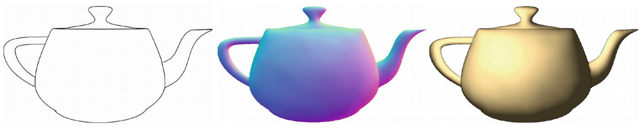

Fig. 11. Line-drawing of the Utah teapot (left), normal obtained with our method (middle), shading effect (right).

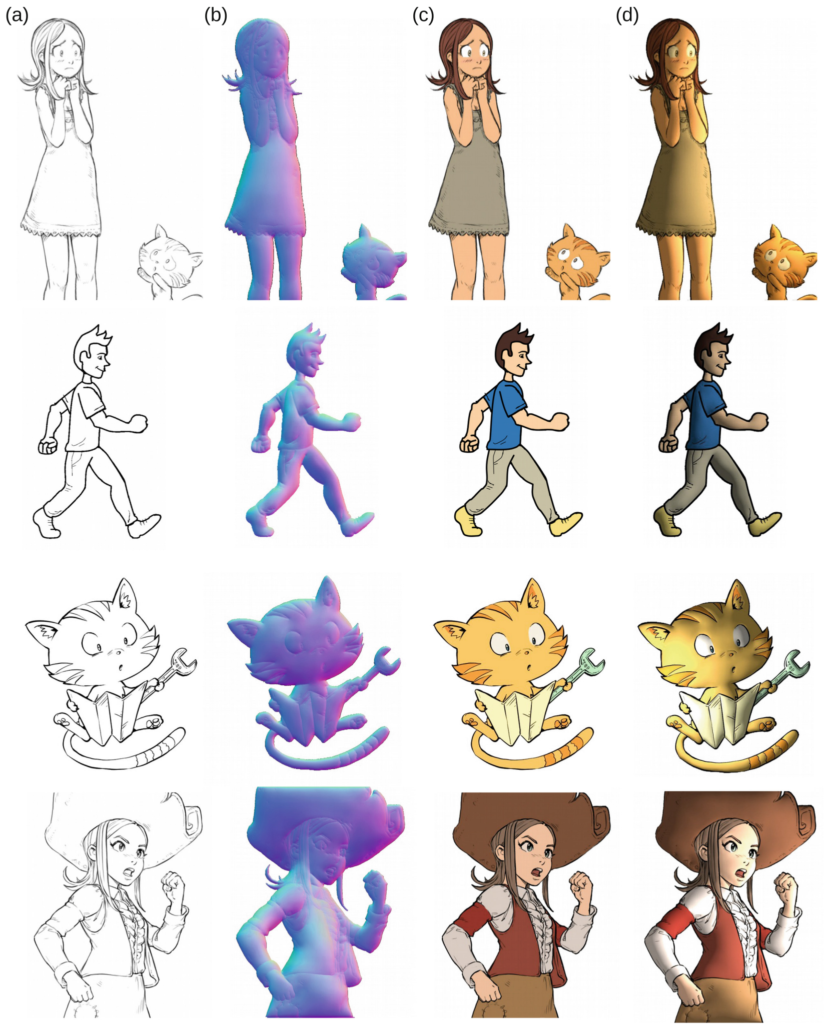

Fig. 12. Our algorithm takes as input the artist line-art (a) and estimates a high-resolution and quality normal map (b). Flat colours (c) can now be augmented with shading effects (d).

rate layer. In Fig. 10, we compare our method to other state-of-the-art geometry based approaches: Lumo [17], TexToons [40] and the Ink-and-Ray pipeline [43]. We created shading effects using our own image-based render engine and tried to match our result as closely as possible to the lighting from [43] for an accurate qualitative comparison. While both Lumo [17] and our technique create 2D normal maps to generate shades, Lumo requires significant user interaction to generate their results. As neither our technique nor Lumo create a full 3D model, neither can add effects such as self-shadowing or inter-reflections. TexToons and

Ink-and-Ray are capable of producing such complex lighting effects, however, again at the cost of significant user interaction. Relative depth ordering added in TexToons via user input, allows for the simulation of ambient occlusion, while Ink-and-Ray requires a lot of user input to reconstruct a sufficient 3D model to allow for the addition of global illumination effects. However, in Fig. 10 we can observe that even without a 3D model, our technique can create high quality shading results without any user interaction, while also being very fast. Furthermore, also shown in Fig. 10, the normal map estimated by Ink-and-Ray is missing many of the finer details of the sketch. Our normal estimation in comparison is more accurate in areas such as folds, facial features, fingers, and hair.

The tiling method used to train the CNN has the effect that we learn normals of primitives rather than of full character shapes. Therefore we can also estimate normals for more generalised input data. Examples of this are shown in Fig. 12 (third row, wrench) and in Fig. 11: even though these objects are not represented in the training database, normals are correctly estimated, allowing us to render high-quality shading effects.

5 Conclusion and Future Work

In this paper we presented a CNN-based method for predicting high-quality and high-resolution normal maps from single line drawing images of characters. We demonstrated the effectiveness of our method for creating plausible shading effects for hand-drawn characters and animations with different types of rendering techniques. As opposed to recent state-of-the-art works, our method does not require any user annotation or interaction, which drastically reduces the labor that is drawing shades by hand. Our tool could be easily incorporated into the animation pipelines used nowadays, to increase efficiency of high quality production. We also showed that using a network in a fully convolutional way does not necessarily produce the most accurate results even when using patch-wise training. We believe and have proven that CNNs further push the boundaries of 3D reconstruction, and remove the need for laborious human interaction in the reconstruction process. While this work only focuses on reconstructing high fidelity normal maps, the CNN could be further extended to also reconstruct depth as in [26] and therefore full 3D models as in [43]. While we created a substantial training database we strongly believe that the predictions could be further improved and applicability extended by simply extending the training database. Other types of drawings could be added to the database such as everyday life objects. Dedicated CNNs could be pre-trained and made available for different types of objects (such as characters in our example here) if better customization is required.

Acknowledgements. 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 Titan Xp GPU used for this research.

References

1. Abadi, M., et al.: Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
2. Anjyo, K.i., Wemler, S., Baxter, W.: Tweakable light and shade for cartoon animation. In: Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering, pp. 133–139. ACM (2006)
3. Bansal, A., Russell, B., Gupta, A.: Marr revisited: 2D–3D alignment via surface normal prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5965–5974 (2016)
4. Barla, P., Thollot, J., Markosian, L.: X-Toon: an extended toon shader. In: Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering, pp. 127–132. ACM (2006)
5. Belhumeur, P.N., Kriegman, D.J., Yuille, A.L.: The bas-relief ambiguity. Int. J. Comput. Vis. **35**(1), 33–44 (1999)
6. Bui, M.T., Kim, J., Lee, Y.: 3D-look shading from contours and hatching strokes. Comput. Graph. **51**, 167–176 (2015)
7. Cole, F., et al.: How well do line drawings depict shape? In: ACM Transactions on Graphics (ToG), vol. 28, p. 28. ACM (2009)
8. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2650–2658 (2015)
9. Feng, L., Yang, X., Xiao, S., Jiang, F.: An interactive 2D-to-3D cartoon modeling system. In: El Rhalibi, A., Tian, F., Pan, Z., Liu, B. (eds.) Edutainment 2016. LNCS, vol. 9654, pp. 193–204. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40259-8_17
10. Grabli, S., Turquin, E., Durand, F., Sillion, F.X.: Programmable rendering of line drawing from 3D scenes. ACM Trans. Graph. (TOG) **29**(2), 18 (2010)
11. Han, X., Gao, C., Yu, Y.: DeepSketch2Face: a deep learning based sketching system for 3D face and caricature modeling. arXiv preprint arXiv:1706.02042 (2017)
12. Henz, B., Oliveira, M.M.: Artistic relighting of paintings and drawings. Vis. Comput. **33**(1), 33–46 (2017)
13. Huang, H., Kalogerakis, E., Yumer, E., Mech, R.: Shape synthesis from sketches via procedural models and convolutional networks. IEEE Trans. Vis. Comput. Graph. **23**(8), 2003–2013 (2017)
14. Hudon, M., Pagés, R., Grogan, M., Ondřej, J., Smolić, A.: 2D shading for cel animation. In: Expressive The Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-photorealistic Animation and Rendering (2018)
15. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3D freeform design. In: SIGGRAPH 1999 Conference Proceedings. ACM (1999)
16. Jayaraman, P.K., Fu, C.W., Zheng, J., Liu, X., Wong, T.T.: Globally consistent wrinkle-aware shading of line drawings. IEEE Trans. Vis. Comput. Graph. **24**(7), 2103–2117 (2017)
17. Johnston, S.F.: Lumo: illumination for cel animation. In: Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering, pp. 45–ff. ACM (2002)
18. Karpenko, O.A., Hughes, J.F.: Smoothsketch: 3D free-form shapes from complex sketches. In: ACM Transactions on Graphics (TOG), vol. 25, pp. 589–598. ACM (2006)

19. Kersten, D., Mamassian, P., Knill, D.C.: Moving cast shadows induce apparent motion in depth. Perception **26**(2), 171–192 (1997)
20. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Koenderink, J.J., Van Doorn, A.J., Kappers, A.M.: Surface perception in pictures. Atten. Percept. Psycho. **52**(5), 487–496 (1992)
22. Kwok, P.: A thinning algorithm by contour generation. Commun. ACM **31**(11), 1314–1324 (1988)
23. Lee, Y., Markosian, L., Lee, S., Hughes, J.F.: Line drawings via abstracted shading. In: ACM Transactions on Graphics (TOG), vol. 26, p. 18. ACM (2007)
24. Li, C., Liu, X., Wong, T.T.: Deep extraction of manga structural lines. ACM Trans. Graph. (TOG) **36**(4), 117 (2017)
25. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
26. Lun, Z., Gadelha, M., Kalogerakis, E., Maji, S., Wang, R.: 3D shape reconstruction from sketches via multi-view convolutional networks. arXiv preprint arXiv:1707.06375 (2017)
27. Olsen, L., Samavati, F.F., Sousa, M.C., Jorge, J.A.: Sketch-based modeling: a survey. Comput. Graph. **33**(1), 85–103 (2009)
28. Pan, H., Liu, Y., Sheffer, A., Vining, N., Li, C.J., Wang, W.: Flow aligned surfacing of curve networks. ACM Trans. Graph. (TOG) **34**(4), 127 (2015)
29. Petrović, L., Fujito, B., Williams, L., Finkelstein, A.: Shadows for cel animation. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 511–516. ACM Press/Addison-Wesley Publishing Co. (2000)
30. Pontes, J.K., Kong, C., Sridharan, S., Lucey, S., Eriksson, A., Fookes, C.: Image2Mesh: a learning framework for single image 3D reconstruction. arXiv preprint arXiv:1711.10669 (2017)
31. Rematas, K., Ritschel, T., Fritz, M., Gavves, E., Tuytelaars, T.: Deep reflectance maps. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4508–4516 (2016)
32. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
33. Schlick, C.: An inexpensive BRDF model for physically-based rendering. In: Computer Graphics Forum, vol. 13, pp. 233–246. Wiley Online Library (1994)
34. Schmidt, R., Khan, A., Singh, K., Kurtenbach, G.: Analytic drawing of 3D scaffolds. In: ACM Transactions on Graphics (TOG), vol. 28, p. 149. ACM (2009)
35. Shao, C., Bousseau, A., Sheffer, A., Singh, K.: CrossShade: shading concept sketches using cross-section curves. ACM Trans. Graph. **31**(4) (2012). https://doi.org/10.1145/2185520.2185541. https://hal.inria.fr/hal-00703202
36. Simo-Serra, E., Iizuka, S., Ishikawa, H.: Mastering sketching: adversarial augmentation for structured prediction. arXiv preprint arXiv:1703.08966 (2017)
37. Simo-Serra, E., Iizuka, S., Sasaki, K., Ishikawa, H.: Learning to simplify: fully convolutional networks for rough sketch cleanup. ACM Trans. Graph. (TOG) **35**(4), 121 (2016)
38. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: the all convolutional net. arXiv preprint arXiv:1412.6806 (2014)

39. Su, W., Du, D., Yang, X., Zhou, S., Hongbo, F.: Interactive sketch-based normal map generation with deep neural networks. In: ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2018). ACM (2018)
40. Sỳkora, D., Ben-Chen, M., Čadík, M., Whited, B., Simmons, M.: Textoons: practical texture mapping for hand-drawn cartoon animations. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-photorealistic Animation and Rendering, pp. 75–84. ACM (2011)
41. Sỳkora, D., Dingliana, J., Collins, S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In: Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering, pp. 25–33. ACM (2009)
42. Sỳkora, D., Dingliana, J., Collins, S.: Lazybrush: flexible painting tool for hand-drawn cartoons. In: Computer Graphics Forum, vol. 28, pp. 599–608. Wiley Online Library (2009)
43. Sỳkora, D., et al.: Ink-and-ray: bas-relief meshes for adding global illumination effects to hand-drawn characters. ACM Trans. Graph. (TOG) **33**(2), 16 (2014)
44. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Multi-view 3D models from single images with a convolutional network. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 322–337. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_20
45. Todo, H., Anjyo, K.I., Baxter, W., Igarashi, T.: Locally controllable stylized shading. ACM Trans. Graph. (TOG) **26**(3), 17 (2007)
46. Tuan, B.M., Kim, J., Lee, Y.: Height-field construction using cross contours. Comput. Graph. **66**, 53–63 (2017)
47. Wang, X., Fouhey, D., Gupta, A.: Designing deep networks for surface normal estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 539–547 (2015)
48. Wanger, L.R., Ferwerda, J.A., Greenberg, D.P.: Perceiving spatial relationships in computer-generated images. IEEE Comput. Graph. Appl. **12**(3), 44–58 (1992)
49. Whited, B., Noris, G., Simmons, M., Sumner, R.W., Gross, M., Rossignac, J.: BetweenIT: an interactive tool for tight inbetweening. In: Computer Graphics Forum, vol. 29, pp. 605–614. Wiley Online Library (2010)
50. Xing, J., Wei, L.Y., Shiratori, T., Yatani, K.: Autocomplete hand-drawn animations. ACM Trans. Graph. (TOG) **34**(6), 169 (2015)
51. Xu, B., Chang, W., Sheffer, A., Bousseau, A., McCrae, J., Singh, K.: True2Form: 3D curve networks from 2D sketches via selective regularization. ACM Trans. Graph. **33**(4), 131 (2014)
52. Zhang, L., Ji, Y., Lin, X.: Style transfer for anime sketches with enhanced residual U-Net and auxiliary classifier GAN. arXiv preprint arXiv:1706.03319 (2017)