SketchyDepth: from Scene Sketches to RGB-D Images

Gianluca Berardi, Samuele Salti, Luigi Di Stefano
Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
{gianluca.berardi3, samuele.salti , luigi.distefano}@unibo.it

Abstract

Sketch-based content generation is a creative and fun activity suited to casual and professional users that has many different applications. Today it is possible to generate the geometry and appearance of a single object by sketching it. Yet, only the appearance can be synthesized from a sketch of a whole scene. In this paper we propose the first method to generate both the depth map and image of a whole scene from a sketch. We demonstrate how generating geometrical information as a depth map is beneficial from a twofold perspective. On one hand, it improves the quality of the image synthesized from the sketch. On the other, it unlocks depth-enabled creative effects like Bokeh, fog, light variation, 3D photos and many others, which help enhancing the final output in a controlled way. We validate our method showing how generating depth maps directly from sketches produces better qualitative results with respect to alternative methods, i.e. running MiDaS after image generation. Finally we introduce depth sketching, a depth manipulation technique to further condition image generation without the need of additional annotation or training.

1. Introduction

Sketching is a universal form of communication that has been used by humans since ancient times. A sketch conveys information to represent objects, scenes, actions or concepts in a concise though expressive form. There exist a variety of reasons to create sketches for professional, e.g. manga, CAD or portraits, as well as casual applications, like free hand drawing, games (like Pictionary or Quik, Draw! [7]) or doodling. Nowadays, the widespread diffusion of touch-screen devices and the impressive advances of machine learning can boost the potential of sketch-based applications. We can draw a sketch to search for images, videos or 3D models, we can define games based on sketch recognition [7] or take part in collaborative drawing by interacting with an AI [5]. One of the most exciting area of research in the realm of AI applied to sketches concerns generating

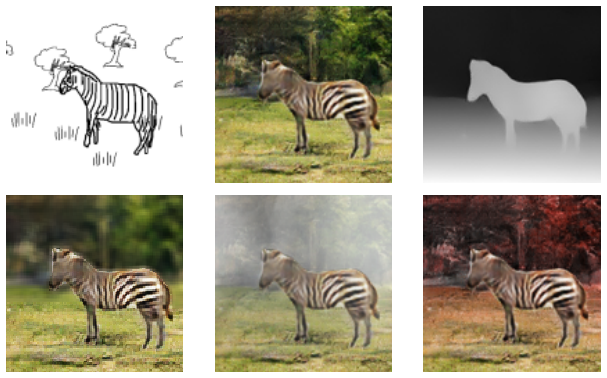
Figure 1. SketchyDepth generates images and depth maps from sketches (first row), enabling depth based creative effects (second row).

an RGB image that may effectively depict the concepts the user is willing to convey through a sketch, *i.e.* synthesizing a plausible RGB image conditioned on a sketch [2][4]. A further leap in this direction would enable to pop the sketch out of its 2D world so as to figure out also a plausible 3D geometry for the sketched content. Generating geometrical information alongside appearance from sketches holds the potential to unlock creative image manipulation applications like climatic effects (e.g. fog, rain), light variation and shadows, shallow depth of field or Bokeh, background substitution, autostereograms and 3D photos. Furthermore, synthesizing geometry from sketches may also contribute to 3D models generation, opening up a host of new possibilities in the space of creative design. In literature there exist some papers concerned with synthesizing 3D information from a sketch, but these works are focused on single object sketches [8][16][23]. Very recently, with the advance of the research over single object sketches, also the topic of scene sketches is starting to attract more attention, with two recent published contributions addressing sketch to image generation [4] and image retrieval from sketches [15].

In this paper we propose the first approach to generate geometrical information from scene sketches. In particular, our deep learning method can synthesize both an RGB image as well as its associated depth map conditioned on an input scene sketch. We empirically demonstrate how leveraging geometrical information helps to generate higher quality RGB images and that we are able to obtain depth maps that are qualitatively better with respect to alternative methods. We show the potential of leveraging depth maps in creative applications, as displayed in Fig. 1, manipulating the generated images by, e.g. Bokeh effect, fog, alteration of the illumination, 3D photos and others. Finally, our framework enables depth map sketching, a novel method to condition image generation by directly manipulating depth maps so as to refine the background content without requiring further training or supervision.

2. Related Works

In the last few years, with the development of deep learning, many papers have tried to apply it to sketches. Classic research themes like sketch recognition [28], sketch-based image retrieval [27] [20] and sketch-based 3d shape retrieval [22] have seen relevant performance improvements while new topics have been proposed like sketch-based model generation [10], sketch generation/synthesis [7] or object-based sketch-to-image synthesis [2]. Also interesting works with human-AI sketch-based interaction have been published [7] [5]. A broader overview of the sketch literature can be found in the recent survey [26].

Despite the growing interest and development of applications in many sketch related areas, geometry generation from sketches, that represents a hard task, is still far to be fully explored. The main investigations concern the generation of 3D shapes from the sketch of a single object [11] [8][16][23][3]. Multi-view Convolutional Networks [16] considers a maximum of three sketches of the same object (front, side and top views) and leverages multi view depth and normal maps prediction to fuse them into a 3D point cloud solving an optimization problem. A different work [23] proposes to synthesize training sketches simulating free hand human style and to normalize them before directly predicting the corresponding 3D shape. A recent work [3] instead introduces a casual sketch to 3D shape interactive tool where it is possible to create a good 3D model of a monster and define simple animations for it.

Geometry generation from scene sketches is instead an unexplored area of research. The scene sketch research [1] has seen recent advances on image generation [4] and image retrieval [15]. In SketchyCOCO [4] a proposed EdgeGAN generates foreground objects (cows, zebras, etc.) one by one following sketch details while a conditional generative network (`pix2pix`) [12] produces the final image, following loosely background indications (grass, sky, forest). SceneSketcher [15] proposes instead a fine-grained image retrieval solution from scene sketches exploiting graph convolutional networks [14].

Thus, in this paper we investigate further along the pathway of scene sketch research and propose the first approach to generate geometrical and appearance information, *i.e.* an RGB-D image, from a scene sketch.

3. Method

Our framework, dubbed SketchyDepth and illustrated in Fig. 2, takes a scene sketch as input and generates the corresponding RGB image and geometric information, the latter represented as a depth map. Akin to SketchyCOCO [4], the objects in the sketch are subdivided into foreground and background, where foreground sketches express precisely user intentions, *i.e.* scale, position and details of an object, while background sketches roughly indicate how the rest of the image should be completed, e.g. grass or trees in certain regions.

To train our framework we start from a dataset consisting of {*scene_sketch*, *scene_image*} pairs. As every object in a *scene_sketch* can be localized and classified by a sketch segmentation method [31], it is possible to crop the regions corresponding to foreground objects from each *scene_image*. Pasting these regions onto the corresponding *scene_sketch* yields *partial_images*, denoted as i_p in Fig. 2. Then, we complement the dataset with the *depth_map* associated with each *scene_image*, which is obtained by an off-the-shelf monocular depth estimation model trained by the authors on a mix of diverse datasets [18]. Thus, to train our generative model that returns the final image, i_{gen} and its associated depth map d_{gen} we rely on a dataset consisting of {*partial_image*, *depth_map*, *scene_image*} triplets.

As depicted within the orange dashed line in Fig. 2, the partial images required by our solution at inference time are obtained by applying the EdgeGAN model [4] to the foreground objects detected by a sketch segmentation method [31]. EdgeGAN is trained separately based on a dataset consisting of {*object_sketch*, *object_image*} pairs, as described in [4].

3.1. Depth Map and Final Image Generation

To produce the depth map and final image corresponding to the input scene sketch, we design the conditional generative adversarial network [6][12] shown within the solid blue line in Fig. 2. Having obtained a partial image, i_p, for each *scene_sketch* in the training dataset as explained above, to train our generative model we create triplets $\{i_p, d_{real}, i_{real}\}$, where d_{real} and i_{real} are the *depth_map* and *scene_image* corresponding to each *scene_sketch*. Then, a depth generator, G_d, takes as input a partial image, i_p, and produces a corresponding depth map, d_{gen}:

$$d_{gen} = G_d(i_p) \tag{1}$$

To guide the generator a depth discriminator, D_d, is defined. D_d is trained to discriminate between fake $\{i_p, d_{gen}\}$ pairs

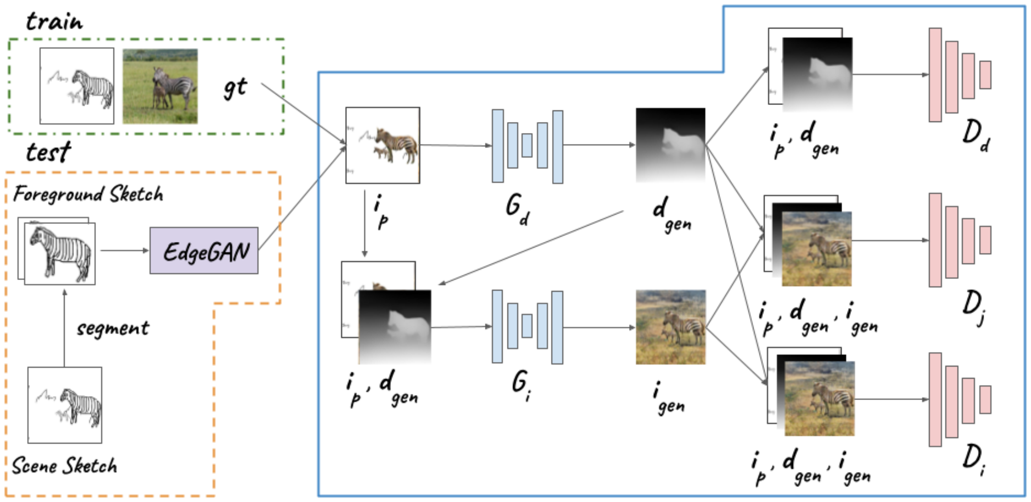

Figure 2. Overview of the SketchyDepth framework.

and real ones, $\{i_p,\ d_{real}\}$. We leverage a conditional GAN loss [12] to train the generator G_d and the discriminator D_d:

$$\mathcal{L}_{gd} = \mathbb{E}_{i_p,d_{real}}[\log D_d(i_p, d_{real})] + \mathbb{E}_{i_p}[\log(1 - D_d(i_p, d_{gen})] \tag{2}$$

where $\mathbb{E}_x[\cdot]$ denotes expectation over x and the discriminator D_d is trained to maximize $\mathcal{L}_{gd}$, while the generator G_d is trained to minimize $\mathcal{L}_{gd}$. Similarly to `pix2pix` [12], the previous loss is complemented by an L1 term that encourages G_d to generate depth maps, d_{gen}, that are similar to the corresponding real ones, d_{real}:

$$\mathcal{L}_{rd} = \mathbb{E}_{i_p,d_{real}}[\|d_{real} - d_{gen}\|_1] \tag{3}$$

The complete loss for G_d and D_d is thus given by:

$$\mathcal{L}_d = \arg\min_{G_d}\max_{D_d}\mathcal{L}_{gd} + \alpha\mathcal{L}_{rd} \tag{4}$$

where α is a scale factor that weighs the contribution of the L1 term.

After depth generation, an image generator, G_i, takes as input the partial image, i_p, and the generated depth map, d_{gen}, in order to generate the final scene image, i_{gen}:

$$i_{gen} = G_i(i_p, d_{gen}) \tag{5}$$

An image discriminator, D_i, is defined as the critic of G_i. D_i is trained to discriminate between fake triplets, $\{i_p,\ d_{gen},\ i_{gen}\}$, and real ones, $\{i_p,\ d_{real},\ i_{real}\}$. Akin to depth generation, a conditional generative loss combined with an L1 term defines the loss for G_i and D_i:

$$\mathcal{L}_{gi} = \mathbb{E}_{i_p,d_{real},i_{real}}[\log D_i(i_p, d_{real}, i_{real})] + \mathbb{E}_{i_p}[\log(1 - D_i(i_p, d_{gen}, i_{gen})] \tag{6}$$

$$\mathcal{L}_{ri} = \mathbb{E}_{i_p,i_{real}}[\|i_{real} - i_{gen}\|_1] \tag{7}$$

$$\mathcal{L}_i = \arg\min_{G_i}\max_{D_i}\mathcal{L}_{gi} + \alpha\mathcal{L}_{ri} \tag{8}$$

Furthermore, to encourage alignment between d_{gen} and i_{gen}, we train a joint discriminator, D_j, to tell apart fake $\{i_p,\ d_{gen},\ i_{gen}\}$ triplets and real ones, $\{i_p,\ d_{real},\ i_{real}\}$. The loss for D_j is defined as a conditional generative loss where both generators are optimized:

$$\mathcal{L}_{gj} = \mathbb{E}_{i_p,d_{real},i_{real}}[\log D_j(i_p, d_{real}, i_{real})] + \mathbb{E}_{i_p}[\log(1 - D_j(i_p, d_{gen}, i_{gen})] \tag{9}$$

$$\mathcal{L}_j = \arg\min_{G_i,G_d}\max_{D_j}\mathcal{L}_{gj} \tag{10}$$

The complete training procedure alternates the update of every discriminator with an update of both generators with respect to $\mathcal{L}_i$, $\mathcal{L}_d$ and $\mathcal{L}_j$.

4. Experiments

4.1. Dataset

In our experiments we use the SketchyCOCO dataset [4], a dataset consisting of $\{$*scene_sketch*, *scene_image*$\}$ pairs that contains 14 foreground classes (like horse, giraffe or elephant) and 3 background classes (trees, grass and sky). Indeed, although image generation from sketches is a generative task, and therefore open-ended, the dataset features for each sketch a ”ground-truth” real image which was paired by the authors with the sketch and can be used as an example of a valid result.

To evaluate our proposed conditional generative network, we start from pairs $\{$*partial_image*, *scene_image*$\}$. Partial images are obtained as detailed in Section 3. These are the same pairs used in SketchyCOCO to train and test the background generation. We then add depth supervision for ground-truth images (and, in turn, the associated sketch) by running a pre-trained version of MiDaS [18], a recent and popular monocular depth estimator trained for generalization. The final datasets contain 11265 training triplets and 2816 test triplets. To validate our experiments we used a split of the training dataset with 9576 training triplets (85%) and 1689 validation triplets (15%).

4.2. Baseline and Implementation

As baseline for scene image generation we use the SketchyCOCO proposal [4]. With this method, in the background generation phase a `pix2pix` network [12] is trained to learn to transform foreground images (i.e. partial images) into scene images. The configuration for the training and evaluation differ from the `pix2pix` default ones as follows. A pre-processing operation is added so as to crop images centrally to obtain the largest squared image and then resize it to 128×128; the `pix2pix` resize and crop pre-processing operations are set to size 128×128 so that they do not modify the images further; as for the generator, the UNet128 [19] architecture proposed in `pix2pix` is selected, the output size of the generated images is 128×128, and the system is trained for 110 epochs. All the other hyper-parameters are kept aligned with the `pix2pix` code.

We trained the baseline to replicate SketchyCOCO results but we were not able to exactly replicate them, even if we used the same official code for `pix2pix` and we had the kind support of the authors in several private communications. This is probably due to different versions of the official `pix2pix` pytorch code[1] [12] [30] that got many fixes and updates over time. Since we were not able to recover the exact version of `pix2pix` code used in SketchyCOCO, we selected a recent version of `pix2pix` pytorch code to have an implementation of reference for all our experiments[2]. Thus, in the experiments reported in Tab. 1 and discussed in the next subsection, we report the SketchyCOCO paper results (first row) as well as the results we have replicated by using the selected `pix2pix` version (second row), which we consider our baseline.

We used the same PyTorch [17] implementation for all the operations in common between the baseline and our proposal, like pre-processing and data augmentation. In our adversarial training, we used `pix2pix` default PatchGAN [12] architecture as discriminators. The depth map follows the same pre-processing of the partial images but processed as single channel. The α L1 balancing term is equal to 100 and we use the Adam optimizer [13] with 0.0002 as learning rate, as done for the baseline. All other hyper-parameters share the same value in the baseline and our method. We used a 1080ti and a 1060 nvidia gpus for our experiments. We will make our code publicly available in case of acceptance.

[1]github: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
[2]commit: f13aab8148bd5f15b9eb47b690496df8dadbab0c

Table 1. Quantitative scene image generation results. We report FID (lower is better) as image quality metric and SSIM (higher is better) as faithfulness metric among generated images and ground-truth images.

Method	**FID** $\downarrow$	**SSIM** $\uparrow$
SketchyCOCO (original) [4]	164.80	**0.2880**
SketchyCOCO (replicated)	130.48	0.2766
SketchyDepth (w/o G_d and D_d)	130.38	0.2820
SketchyDepth (w/o D_j)	136.13	0.2783
SketchyDepth	**116.87**	0.2693

4.3. Scene Image Evaluation

We evaluate the images generated from the sketches belonging to the test set by comparing them with the corresponding ground-truth images according to Fréchet Inception Distance (FID) [9] and SSIM [24] metrics. FID is a measure of similarity between two sets of images that correlates well with human judgement of visual quality and it is often used to evaluate the realism of generated images. Thus, we use it to compare the set of generated images and the set of ground-truth images in order to evaluate the image generation quality. The SSIM metric measures the similarity between a pair of images and it is normally used as a quality measure when the perfect image one wishes to attain is known (*e.g.* in super resolution applications). In the context of scene generation from sketches, the SSIM has been proposed as a measure of faithfulness to the sketch of the generated image by using the ground-truth image as a proxy for the sketch. However, we argue the SSIM to be less meaningful than the FID for a generative task like scene from sketches, because the "ground-truth" image is just one of the possible high-quality results and not the only correct one, and slightly higher SSIM with respect to it may not really measure a lower faithfulness to the sketch, especially for background regions which are coarsely indicated in the sketch and may instead vary wildly between generated images. Nonetheless, we report SSIM scores as done in [4].

Tab. 1 collects the experimental results. The first row reports the original SketchyCOCO paper results, while the second row those obtained trying to replicate it, which we use as baseline to compare the results provided by our framework, shown in row 5.

Our framework yields better FID with respect to the baseline, this highlighting that the images generated by our method are usually of higher quality with respect to those generated by the baseline. The SSIM score, instead, is slightly worse: as discussed above, we believe this result mainly indicates that the content generated by our method for a specific sketch differs more from the corresponding ground-truth image. Yet, it may still represent well the input sketch, as shown by the qualitative results in Fig. 3.

As reported in Tab. 1, beside testing the SketchyCOCO

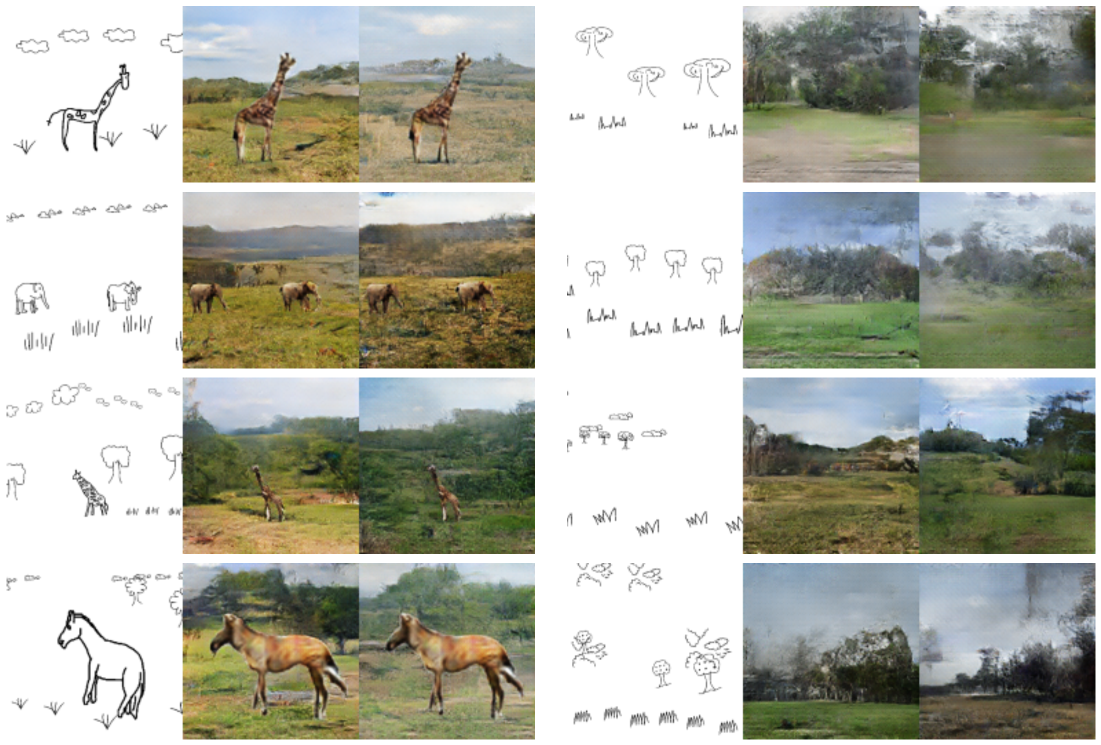

Figure 3. Scene image generation from sketch. Results are visualized in two columns: each column shows from left to right the input sketch, our result, and the baseline. The first column deal with sketches featuring foreground and background content while in the second the sketches contain only background classes.

baseline and our proposal, we also consider two additional configurations as an ablation study. In particular, on one hand, we wish to sift out the actual contribution of the joint depth map generation to the image generation performance of our framework, while on the other we wish to validate the impact of the joint discriminator D_j. If we remove all components strictly needed to produce and discriminate the depth map in our proposal, *i.e.* G_d and D_d, we are left with the generator G_i and the discriminators D_i and D_j, modified so as to not take a depth map as input. This can also be thought of as the baseline with two discriminators, *i.e.* with the additional training signal of D_j for G_i. Row 3 shows the results of this experiment, while row 4 ablates the contribution of the joint discriminator D_j. Ablating out the joint depth map generation, SketchyDepth obtains the same FID score as the baseline, showing how the additional training signal for the image generator, considering only the RGB image information, does not help to reach higher image quality generation. This validates that the FID gain of our method is related to the introduction of depth information, and shows how depth information can impact significantly the sketch to image task, in particular increasing image generation quality. Interestingly, the SSIM of this configuration increases with respect to the baseline: more supervision for the generator within the same budget for training time forces it to produce images more similar to the ground-truth ones, which increases SSIM but does not increases realism as measured by the FID, another hint of the biased nature of the SSIM and the limited relevance of small differences in its value to assess the performance of the sketch to scene image generation task. Similarly, our solution without D_j cannot improve performances over the baseline. This result show how, beside introducing the additional depth information as studied in the previous experiment, it is also important to have the correct amount of training signal and alignment between d_{gen} and i_{gen}, as introduced by D_j, to be able to improve image quality generation. This validates that all components of our framework are needed to deliver improved performance with respect to the baseline in the sketch to scene image generation task.

In Fig. 3, we show images generated from test sketches. The first column concerns sketches with foreground and background content, the second one sketches with background content only. Visual observation suggests that depth information contributes to fill effectively the empty regions of the input sketch, see in particular column 2 rows 3 and 4, upper right portion of images. We investigated on the FID subdividing test examples in foreground-background (fb) and background-only (bo), obtaining 96.70 in fb and 184.73 in bo for the baseline and 92.91 in fb and 162.24 in bo for the proposed method. This quantitative measurements seem to support our observation concerning the ability of our so-

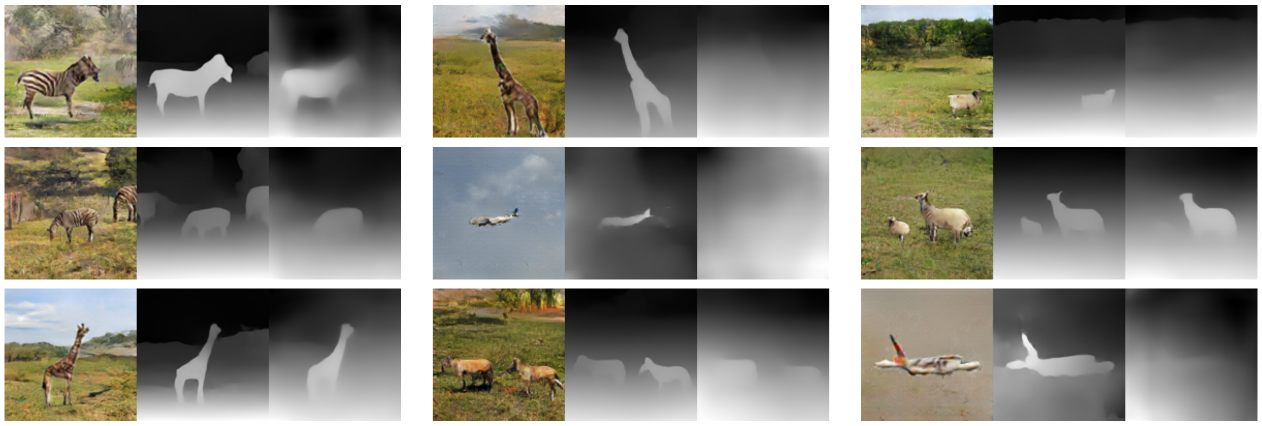
Figure 4. Depth map generation from sketch. Results are visualized in three columns. We show from left to right: our generated scene images, our generated depth maps and depth maps generated by MiDaS from our generated images.

lution to contribute more in terms of background generation.

4.4. Depth Evaluation

There are no previous methods that can generate a depth map from a scene sketch. However, it is possible to obtain depth maps from images by running depth-from-mono models. Thus, to asses the quality of the depth maps generated by our method, we compare them with those generated by a pre-trained monocular depth estimator, i.e. MiDaS [18], applied to the generated RGB images. We apply MiDaS on generated images because real RGB images are not available at test time. Furthermore, even if they were available, the comparison would not be meaningful because, for the same sketch, the real image and the generated one exhibit different structures and therefore different depth maps. Fig. 4 shows examples of depth map generation. In each of the three columns we show, from left to right, generated images, the corresponding depth maps generated jointly by our method and the depth maps obtained by MiDaS on generated images. Our method produces good depth maps most of the times, while MiDaS depth maps are often less detailed, in particular on foreground objects, and sometimes grossly wrong. This is somewhat surprising since MiDaS is also used to create depth supervision at training time for our framework, and it shows how monocular depth estimation algorithms are influenced by subtle changes in the image texture [25], which occurs in our generated images. This also vouches for the importance of simultaneous generation of RGB and depth information, as proposed in our framework, to avoid such depth artifacts or errors.

To further assess our depth map generation results we conducted a human evaluation. We selected 100 random sketches containing foreground and background from the test dataset and generated the corresponding image and depth map with our method. From the generated images we also obtained depth maps by MiDaS. Given the generated image and both depth maps, the test proposed to the participant consists in choosing which of the depth maps best corresponds to the displayed image. We collected data from 23 people with previous experience with computer vision and depth maps. Over $23 \times 100 = 2300$ answers the 23 participants preferred our generated depth maps compared to those yielded by MiDaS 2153 times, *i.e.* about $\mathbf{93.60}\%$ of the times. Human evaluation results are therefore aligned with the previous qualitative considerations.

4.5. Depth Based Creative Applications

One of the interesting applications unlocked by generating depth maps together with images is the possibility to apply effects over images which vary according to depth values. Given an effect, it is possible to choose the depth value where the effect will start to affect the image, the direction of the effect, i.e. if the intensity of the effect increases or decreases alongside depth value, and the maximum intensity of the effect. We experimented with three different effects. The first one is a blur filter whose intensity increases with the estimated depth of the central pixel, in particular the weight of the central pixel is scaled accordingly. This solution can be used to reproduce a shallow depth of field or a Bokeh effect[3]. Light, hue shifts, and colour to grayscale transitions, where the intensity of the effect increases along with depth, are also possible and can be realized transforming the image to the HSV color-space and scaling a specific channel of pixels according to depth value. Finally, climatic fog and the transition-to-cartoon effects can be realized by using an overlay image of the desired effect, *i.e.* a fog image and a cartooned version of the input image. The output image is produced as a weighted average of the input image and the overlay image where the weight varies according to the depth value at every pixel. Further details about how we

[3]`https://en.wikipedia.org/wiki/Bokeh`

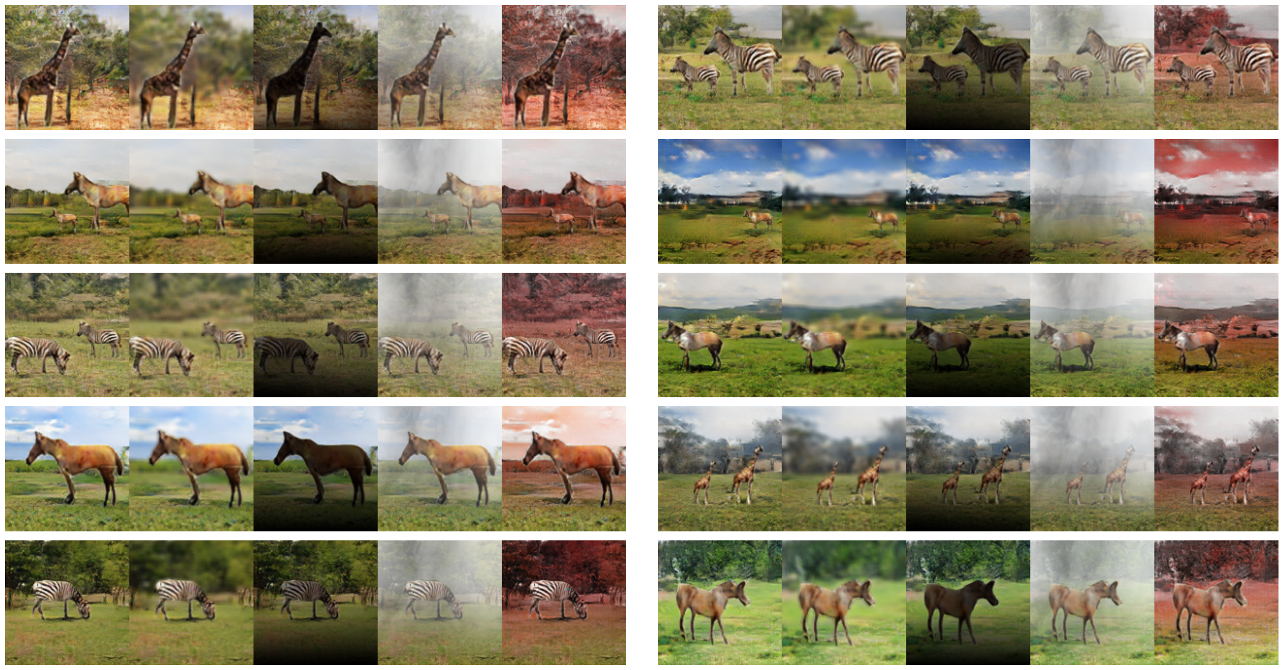
Figure 5. Leveraging depth map information to apply creative effects on generated images. In both columns from left to right we show generated images, bokeh effect, light variation, fog and hue shift.

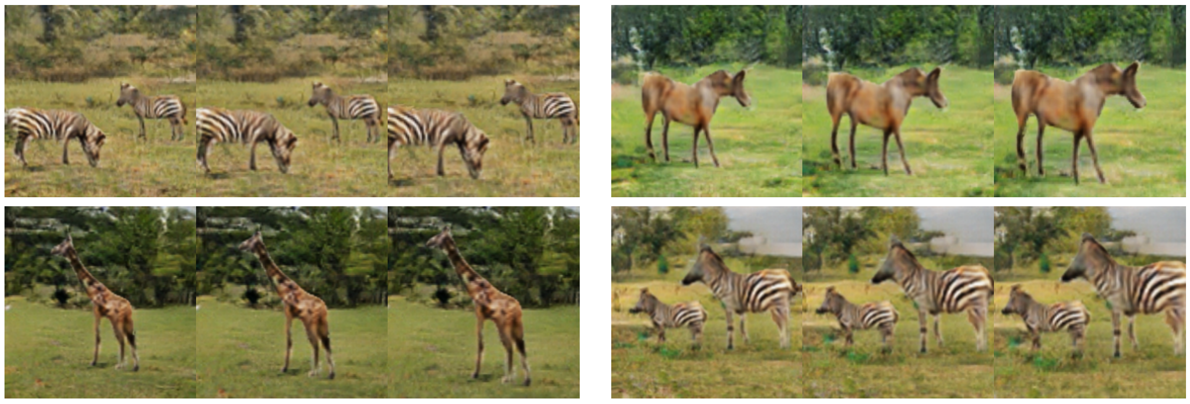
Figure 6. 3D photo example: simulation of the movement of the camera toward the foreground of the scene, i.e. a zoom effect is simulated.

obtained overlay images can be found in the supplementary materials. We note that in the literature also more sophisticated depth-based algorithms [29] to introduce simulated fog into images do exist, which could be beneficially leveraged starting from our generated depth map. Fig. 5 reports examples of the above mentioned effects applied to images generated by our method based on the associated generated depth maps. In particular Bokeh, light variation, fog and hue shift effects are shown in the figure, while colour to grayscale and transition-to-cartoon effects are visible in the supplementary material. We also experimented with the creation of so called 3D photos, *i.e.* a video where a 3D effect is applied. We used a pre-trained model [21] to produce 3D photos using our generated depth maps and images. Fig. 6 shows some sequences of frames sampled from the generated 3D photos. Animated versions are provided in the supplementary material.

Overall, these qualitative results show how obtaining geometrical information from a sketch enables different and engaging effects that exceed the limitations of the 2D image plane. The proposed effects are just an example of the possibilities unleashed by the availability of depth maps, and more creative uses can be explored.

4.6. Can You Sketch a Depth ?

The design of our pipeline enables the exploration of another creative use of depth maps. Once our system is trained, we can get the partial image of a sketch from a

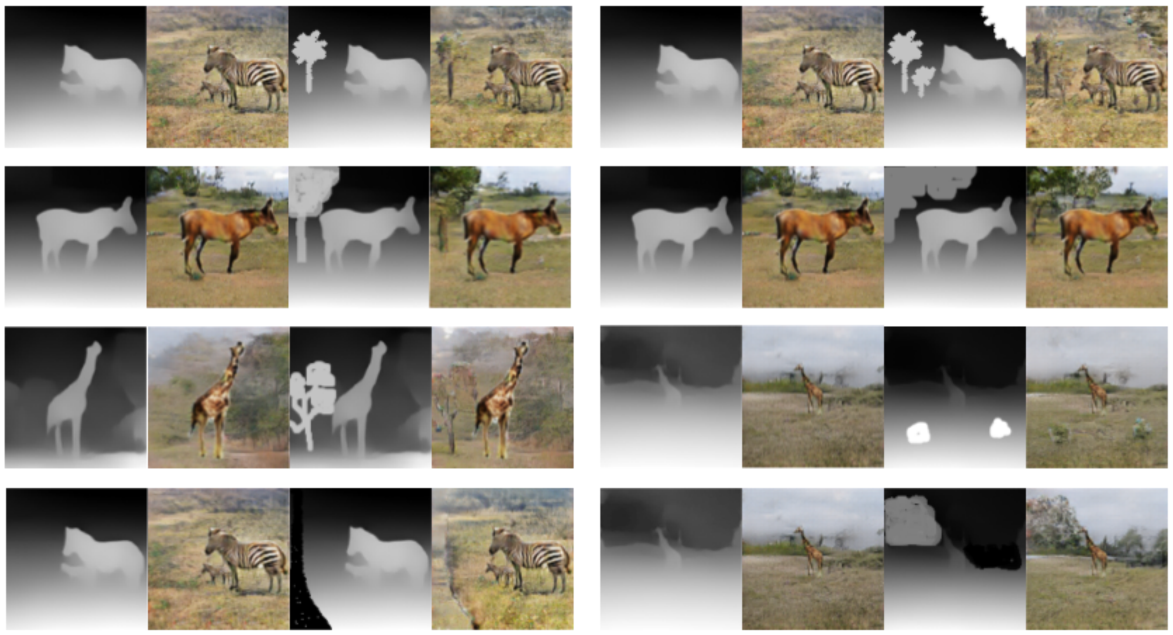
Figure 7. Sketching generated depth maps to acquire finer control over image generation. In both columns, from left to right: generated depth map, generated image, sketched depth map and newly generated image.

pre-trained EdgeGAN and use G_d to get the corresponding depth map. Since the image generator G_i is then conditioned on it, if we manually modify the depth map with an image editor before feeding it to G_i, we can gain additional control over image generation. For instance, we can add shapes of objects that belong to background classes or modify the structure of the image, e.g. the height of the horizon. In our experiments, we treat the depth map as a grayscale image and edit for every new object/area a single fixed grayscale value, which defines its depth. After the depth sketching phase, we can use the resulting depth map and the partial image as input to G_i to obtain the corresponding generated image.

Depth sketching examples are visible in Fig. 7. Qualitative results show how our system can generate new objects and partially modify the image structure when conditioned on a sketched version of the generated depth map, without further training or supervision. The system learns to do so only from the edited depth shapes, in particular G_i gains knowledge about the shapes of objects and image structures that occur often in the training dataset. Depth sketching gives us more control over position, shape, and scale of objects which are normally part of the background. Indeed, this information cannot be specified directly in the sketch, where background classes are meant as coarse selectors of the background texture, not as precise indicators of the appearance of the objects forming the background. As in our setting the background objects that appear more frequently are grass, sky and trees, we can get additional and finer control mainly over different types of trees, bushes, vegetation. Although depth sketching is an interesting additional way to gain control over the generated image, it is enabled by our framework almost as a side effect and it comes with its own limitations: in particular, sometimes more than one attempt is needed to obtain the desired result. We believe that a training protocol specifically designed with this use case in mind, *e.g.* with data augmentation simulating the test conditions, could easily enhance its robustness.

5. Conclusion and Future Work

We have presented SketchyDepth, the first framework to generate a depth map and an image from a scene sketch. We demonstrate how leveraging geometrical information allows for improving scene sketch to image generation quality and how our framework can generate depth maps that are consistently better compared to an alternative method. Moreover, generated depth maps can be used to obtain many depth-based effects over generated images which offer a variety of tuning nobs to creative users. Our framework also enables depth sketching, another creative depth manipulation technique that gives finer control over the generated background. We hope that our findings may foster further investigations dealing with depth maps generation. For instance, one might conjecture that leveraging depth information could improve the results in generative tasks beyond sketches, *e.g.* in a classic RGB generation task with state of the art generative models.

References

[1] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: Internet image montage. *ACM transactions on graphics (TOG)*, 28(5):1–10, 2009.

[2] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9416–9425, 2018.

[3] Marek Dvorožňák, Daniel Sỳkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. Monster mash: a single-view approach to casual 3d modeling and animation. *ACM Transactions on Graphics (TOG)*, 39(6):1–12, 2020.

[4] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou. Sketchycoco: image generation from freehand scene sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5174–5183, 2020.

[5] Songwei Ge, Vedanuj Goswami, C Lawrence Zitnick, and Devi Parikh. Creative sketch generation. *arXiv preprint arXiv:2011.10039*, 2020.

[6] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *In Advances in Neural Information Processing Systems (NIPS)*, 2014.

[7] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

[8] Xiaoguang Han, Chang Gao, and Yizhou Yu. Deepsketch2face: A deep learning based sketching system for 3d face and caricature modeling. *ACM Transactions on graphics (TOG)*, 36(4):1–12, 2017.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

[10] Conghui Hu, Da Li, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Sketch-a-classifier: Sketch-based photo classifier generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2018.

[11] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH 2006 Courses*, pages 11–es. 2006.

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

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Fang Liu, Changqing Zou, Xiaoming Deng, Ran Zuo, Yu-Kun Lai, Cuixia Ma, Yong-Jin Liu, and Hongan Wang. Scenesketcher: Fine-grained image retrieval with scene sketches. In *European Conference on Computer Vision*, pages 718–734. Springer, 2020.

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

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] Ranftl René, Lasinger Katrin, Hafner David, Schindler Konrad, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[20] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

[21] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8028–8038, 2020.

[22] Fang Wang, Le Kang, and Yi Li. Sketch-based 3d shape retrieval using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1883, 2015.

[23] Jiayun Wang, Jierui Lin, Qian Yu, Runtao Liu, Yubei Chen, and Stella X Yu. 3d shape reconstruction from free-hand sketches. *arXiv preprint arXiv:2006.09694*, 2020.

[24] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[25] Alex Wong, Safa Cicek, and Stefano Soatto. Targeted adversarial perturbations for monocular depth prediction. In *Advances in neural information processing systems*, 2020.

[26] Peng Xu, Timothy M Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. Deep learning for free-hand sketch: A survey and a toolbox. *arXiv e-prints*, pages arXiv–2001, 2020.

[27] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. Sketch me that shoe.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 799–807, 2016.

[28] Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Sketch-a-net: A deep neural network that beats humans. *International journal of computer vision*, 122(3):411–425, 2017.

[29] Ning Zhang, Lin Zhang, and Zaixi Cheng. Towards simulating foggy and hazy images and evaluating their authenticity. In *International Conference on Neural Information Processing*, pages 405–415. Springer, 2017.

[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[31] Changqing Zou, Haoran Mo, Chengying Gao, Ruofei Du, and Hongbo Fu. Language-based colorization of scene sketches. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.