Interactive Scene Generation via Scene Graphs with Attributes

Oron Ashual,[1] **Lior Wolf**[1,2]
[1]Tel Aviv University
[2]Facebook AI Research
oron.ashual@gmail.com
wolf@cs.tau.ac.il, wolf@fb.com

Abstract

We introduce a simple yet expressive image generation method. On the one hand, it does not require the user to paint the masks or define a bounding box of the various objects, since the model does it by itself. On the other hand, it supports defining a coarse location and size of each object. Based on this, we offer a simple, interactive GUI, that allows a layman user to generate diverse images effortlessly.

From a technical perspective, we introduce a dual embedding of layout and appearance. In this scheme, the location, size, and appearance of an object can change independently of each other. This way, the model is able to generate innumerable images per scene graph, to better express the intention of the user.

In comparison to previous work, we also offer better quality and higher resolution outputs. This is due to a superior architecture, which is based on a novel set of discriminators. Those discriminators better constrain the shape of the generated mask, as well as capturing the appearance encoding in a counterfactual way.

Our code is publicly available at https://www.github.com/ashual/scene_generation.

Introduction

In a recent work published at the sister conference ICCV (Ashual and Wolf 2019), we introduce an interactive method for generating realistically looking images. Our method employs scene graphs with per-object location and appearance attributes as an easy-to-manipulate way for users to express their intentions, see Fig. 1. The image specifications consist of various objects, which are defined as belonging to a certain class (horse, tree, boat, etc.) and as having certain appearance attributes. The attributes are either archetypal, and obtained by clustering previously seen attributes, or directly imported from a sample image. The relative locations of the objects are specified by a scene graph, which is a graph where the scene objects are denoted as nodes, and their relative position, such as "above" or "left of", are represented as edge types.

Each object in the image is encoded inside the neural network by a dual representation. The first part encodes the object's placement and captures a relative position and other global image features, as they relate to the specific object. It is generated based on the scene graph, by employing a graph convolution network, followed by the concatenation of a random vector z. The second encoding vector captures the appearance of the object and can be copied from the same object as it appears in another image, without directly changing the other objects in the image.

The method is implemented within a convenient user interface, which supports a dynamic placement of objects. The object locations in the user interface are interpreted as an approximated placement on a coarse grid and not as an exact location or a hard constraint. The edge relations in the scene graph are inferred automatically, given the relative position of the objects, which eliminates the need for mostly unnecessary user intervention. Rendering is done in real-time, supporting the creation of novel scenes in an interactive way.

The neural network that we employ has multiple subparts, as can be seen in Fig. 2: (i) A graph convolutional network that converts the input scene graph to a per-object embedding to their location. (ii) A CNN that converts the location embedding of each object to an object's mask. (iii) A parallel network that converts the location embedding to a bounding box location, where the object mask is placed. (iv) An appearance embedding CNN that converts image information into an embedding vector. This process is done off-line and when creating a new image, the vectors can be imported from other images, or selected from a set of archetypes. (v) A multiplexer that combines the object masks and the appearance embedding information, to create a one multidimensional tensor, where different groups of layers denote different objects. (vi) An encoder-decoder residual network that creates the output image.

Most related to our method is the one by (Johnson, Gupta, and Fei-Fei 2018). In comparison to this method, our method separates the layout embedding from the appearance embedding, in order to allow for much more control and freedom to the object selection mechanism. In addition, the architecture we employ enables better quality and higher resolution outputs and adds stochasticity before the masks are created,

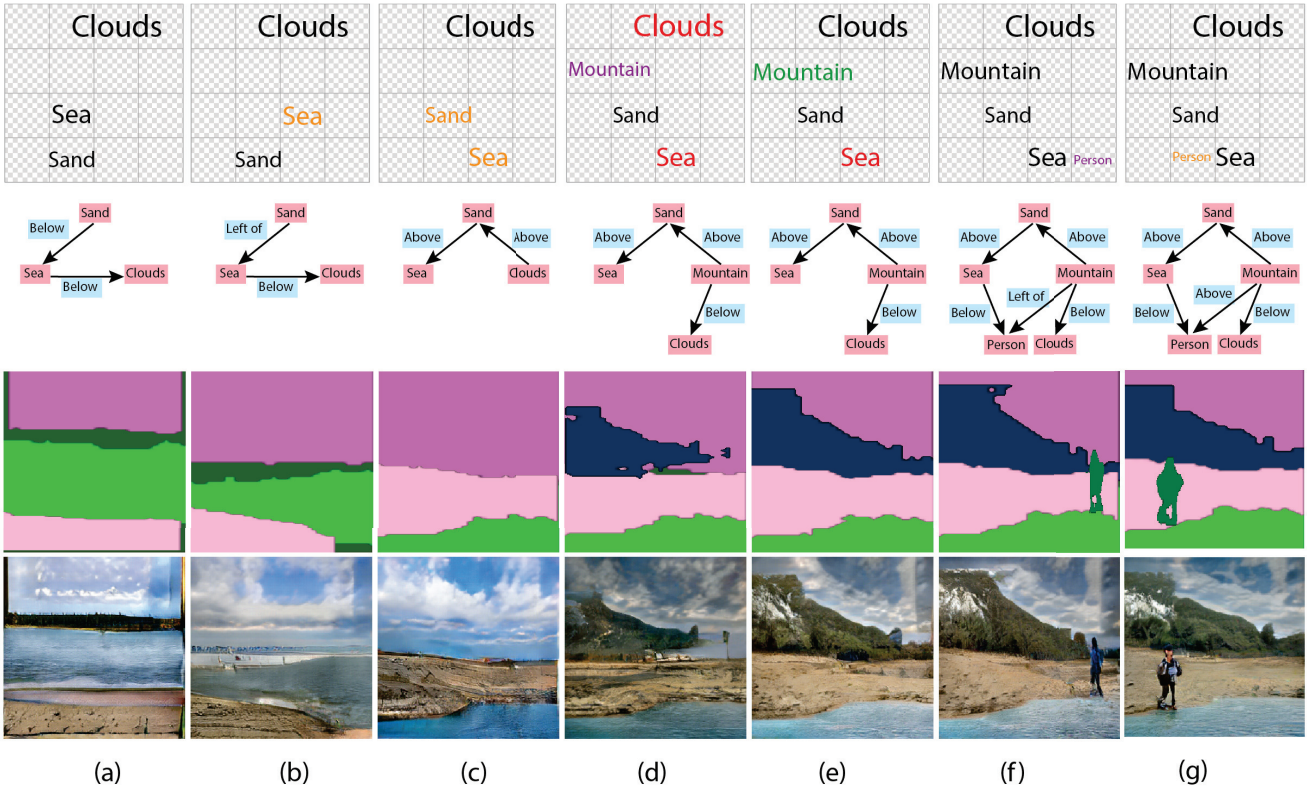

Figure 1: An example of the image creation process. (top row) the schematic illustration panel of the user interface, in which the user arranges the desired objects. (2nd row) the scene graph that is inferred automatically based on this layout. (3rd row) the layout that is created from the scene graph. (bottom row) the generated image. Legend for the GUI colors in the top row: purple – adding an object, green – resizing it, orange – moving it, red – replacing its appearance. (a) A simple layout with a sea, clouds and a sand objects. All object appearances are initialized to a random archetype appearance. (b) The sea is moved to the right. The appearance of the sea is changed to a different archetype in b-e. (c) The sand and the sea are moved, causing the perspective to change. (d) A mountain object is added. The appearance of the clouds is changed. (e) The mountain object is enlarged. (f) A person is added (g) The person is moved to the left.

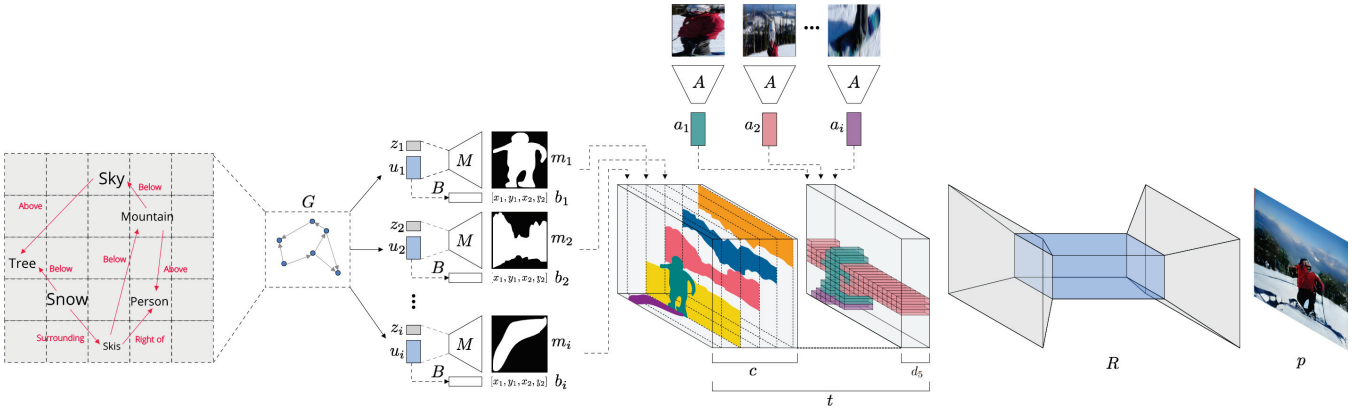

Figure 2: The architecture of our composite network, including the subnetworks G, M, B, A, R, and the process of creating the layout tensor t. The scene graph is passed to the network G to create the layout embedding u_i of each object. The bounding box b_i is created from this embedding, using network B. A random vector z_i is concatenated to u_i, and the network M computes the mask m_i. The appearance information, as encoded by the network A, is then added to create the tensor t with $c + d_5$ channels, c being the number of classes. The autoencoder R generates the final image p from this tensor.

Figure 3: Image generation based on a given scene graph. Each row is a different example. (a) the scene graph, (b) the ground truth image, from which the layout was extracted, (c) our results when we used the ground truth layout of the image, similar to (Zhao et al. 2018), (d) our method's results, where the appearance attributes present a random archetype and the location attributes coarsely describe the ground truth bounding box, (e) our results when we use the ground truth image to generate the appearance attributes, and the location attributes are zeroed $l_i = 0$, (f) our results where $l_i = 0$, and the appearance attributes are sampled from the archetypes, and (g) the results of (Johnson, Gupta, and Fei-Fei 2018).

Figure 4: The diversity obtained when keeping the location attributes l_i fixed at zero and sampling different appearance archetypes. (a) the scene graph, (b) the ground truth image, from which the layout was extracted, (c–g) generated images.

Figure 5: Duplicating an object's appearance in the generated image. Images are created based on the scene graph, such that the appearance is taken from one of five unrelated images. In this example, the bus's appearance is generated from the reference image, while all other objects use the same random appearance archetype.

leading to the generation of multiple results per scene graph. We also introduce a mask discriminator, which plays a crucial role in generating plausible masks, as well as another novel discriminator that captures the appearance encoding in a counterfactual way. The appearance of the object is better captured by introducing feature matching losses based on the discriminator network as well as a perceptual loss term.

Results

In (Ashual and Wolf 2019), we provide a very detailed empirical comparison to the state of the art methods of (Johnson, Gupta, and Fei-Fei 2018) and (Zhao et al. 2018), presenting a sizable gap in performance in favor of our method. Here, we focus on sharing the qualitative results obtained.

Sample results of our 256x256 model are shown in Fig. 3, using test images from the COCO-stuff datasets. Each row presents the scene layout, the ground truth image from which the layout was extracted, our method's results, where the object attributes present a random archetype and the location attributes are zeroed ($l_i = 0$), our results when using the ground truth layout of the image (including masks and bounding boxes), our results where the appearance attributes of each object are copied from the ground truth image and the location vectors are zero, and our results where the location attributes coarsely describe the objects' locations and the appearance attributes are randomly selected from the archetypes. In addition, we present the result of the baseline method of (Johnson, Gupta, and Fei-Fei 2018) at the 64x64 resolution for which a model was published.

As can be seen, our model produces realistic results across all settings, which are more pleasing than the baseline method. Using ground truth location and appearance attributes, the resulting image better matches the test image.

Fig. 4 presents samples obtained when sampling the appearance attributes. In each case, for all i, $l_i = 0$ and the object's appearance embedding a_i is sampled between the archetypes. This results in a considerable visual diversity.

The ability of our method to copy the appearance of an existing image object is demonstrated in Fig. 5. In this example, we generate the same test scene graph, while varying a single object in accordance with five different options extracted from images unseen during training. Despite the variability of the appearance that is presented in the five sources, the generated images mostly maintain their visual quality.

Conclusion

In the presented image generation tool, the input consists of a scene graph with the addition of location information. Each object is associated both with a location embedding as well as with an appearance embedding, which can be extracted from another image. This allows the duplication of objects in other images such that their layout drastically changes. The method also introduces a new architecture and new loss terms, which improve the quality of the generated images in comparison to the literature baselines.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the EU Horizon 2020 research and innovation programme (grant ERC CoG 725974).

References

Ashual, O., and Wolf, L. 2019. Specifying object attributes and relations in interactive scene generation. In *The IEEE International Conference on Computer Vision (ICCV)*.

Johnson, J.; Gupta, A.; and Fei-Fei, L. 2018. Image generation from scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhao, B.; Meng, L.; Yin, W.; and Sigal, L. 2018. Image generation from layout. *CoRR* abs/1811.11389.