Text-to-Vector Conversion for Residential Plan Design

Egor Bazhenov, Stepan Kasai, Viacheslav Shalamov, Valeria Efimova
ITMO University, St. Petersburg, Russia

Implementation

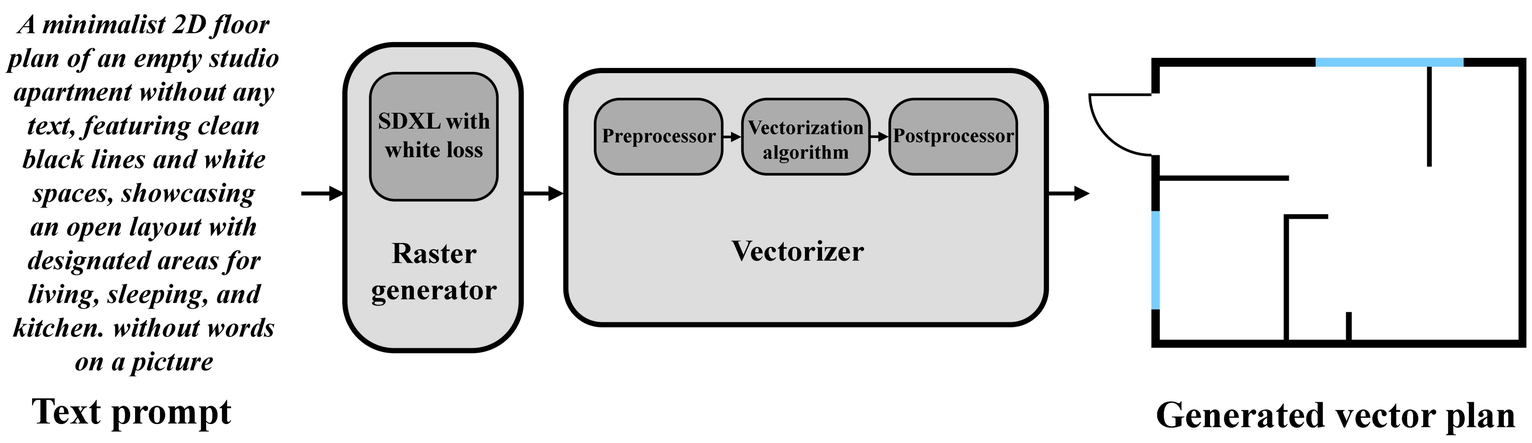

Figure 1. Example of generation vector plans according to the textual description.

We propose the novel method for generating vector residential plans based on textual description that provides plans in SVG file format without extra paths.

Abstract. Computer graphics, comprising both raster and vector components, is a fundamental part of modern science, industry, and digital communication. While raster graphics offer ease of use, its pixel-based structure limits scalability. Vector graphics, defined by mathematical primitives, provides scalability without quality loss, however, it is more complex to produce. For design and architecture, the versatility of vector graphics is paramount, despite its computational demands. This paper introduces a novel method for generating vector residential plans from textual descriptions. Our approach surpasses existing solutions by approximately 5% in CLIPScore-based visual quality, benefiting from its inherent handling of right angles and flexible settings. Additionally, we present a new algorithm for vectorizing raster plans into structured vector images. Such images have a better CLIPscore compared to others by 4%.

Introduction. Global urbanization [1] of population makes residential planning critical for real estate development. Housing construction is time consuming and investment demanding with residential plans design being only a part of it. Thus, the task to make the generation of residential plans automated is extremely relevant. Vector graphics is also preferred by designers and architects, as it is scalable and has small file size. Currently, there are two main approaches to generate vector residential plans based on textual description. The first one comprises a generator and a vectorizer. The second approach is to generate a vector plan using a Large Language Model, but the key disadvantage is the low quality of generated vector images.

Model	Stable Diffusion	SDXL	FLUX	AuraFlow	SDXL with white loss
Image example					
Image CLIPScore	0.227	0.231	0.272	0.263	0.281
CLIPScore Avg.	0.226 ± 0.012	0.23 ± 0.014	0.265 ± 0.009	0.258 ± 0.01	0.276 ± 0.006

Table 1. Comparison of plan generation models. Compared models are Stable Diffusion, Stable Diffusion XL, FLUX, AuraFlow, and SDXL with white loss.

Method. We present an approach consisting of a raster generator and a vectorizer.
The scheme of the method is shown in Figure 1.

Generation is the first part of our approach is a raster generator — SDXL with the new loss. We designed new loss function (white loss) to generate images on a white background. This is necessary so that the plans are clear for further vectorization. The function is as follows:

$$L_{white} = s \left\| D(p(x_0|x_t)) - \text{mask}(D(p(x_0|x_t))) \right\|^2, \quad (1)$$

where D — SDXL decoder, s — scale, x0 — vector without noise in latent space, xt — vector in latent space with noise corresponding to step t, mask - mask that defines the field that we would like to be white.

We also need to update xt after this by following recurrent formula:

$$x_t = x_t - \text{mask}_{\text{latent}} \left(\Delta x_t \cdot (1 - \overline{\alpha_t}) \,/\, \overline{\alpha_t} \right), \qquad (2)$$

where t — diffusion step, Δxt — gradient of xt according to Lwhite, masklatent — mask that is scaled to be used in latent space, αt — cumulative product.

We use the SDXL decoder to obtain an image from xt on early time step t, then we apply a mask to this image and calculate the MSE loss, which is high in the mask field.
Then, we update xt according to formula 2.

Vectorization consists of three main parts — preprocessing, vectorization, and postprocessing. *Preprocessing* prepares the image for subsequent vectorization. It is necessary to ensure that the final image has no unnecessary parts, such as interior items. The preprocessing algorithm converts each pixel of a black and white raster RGB image to monochrome, filter them, and converts them back to RGB. *Vectorization Algorithm* converts the black and white raster preprocessed plan to a vector image using the Shi-Tomasi [3] corner detection method []. First, there is search for corners in the image, average their coordinates so that they are on the same straight lines, also solving the problem of crooked walls. After that, a cycle begins and at each iteration it creates rectangles that can match the walls based on the corners list. The rectangles are selected based on their similarity to the original image. After suitable rectangles are selected, they are converted into vector paths. *Postprocessing* is the final part of vectorization. It allows obtaining the final vector plan with a correct SVG structure and without redundant paths. The postprocessing algorithm removes useless paths inside other paths and combines paths that share a common part and form a rectangle together.

References. [1] Ichimura, M. 2003. Urbanization, urban environment and land use: challenges and opportunities. In Asia-Pacific Fokum for Environment and Development, Expert Meeting, volume 23, 1–14. Citeseer.

[2] Podell, D.; English, Z.; Lacey, K.; Blattmann, A.; Dockhorn, T.; M¨uller, J.; Penna, J.; and Rombach, R. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. arXiv preprint arXiv:2307.01952.

[3] Shi, J.; et al. 1994. Good features to track. In 1994 Proceedings of IEEE conference on computer vision and pattern recognition, 593–600. IEEE.

[4] Hessel, J.; Holtzman, A.; Forbes, M.; Bras, R. L.; and Choi, Y. 2021. Clipscore: A reference-free evaluation metric for image captioning. arXiv preprint arXiv:2104.08718.

Related works. The task of generating a vector image based on the textual description can be solved by combining a raster generator with a vectorizer or end-to-end vector generator.

Raster generator with a vectorizer. The raster generator creates a bitmap plan based on a textual description, and then the vectorization algorithm converts the bitmap image to a vector format. Due to the flexibility of this approach, we can configure and select the generator and vectorizer best suited for the task. For raster generation the two main approaches are often applied: Rule-based generation models and neural networks. *Rule-based generation* is deterministic approach. An image is selected based on the single textual template that can be customized. As all possible variations of the images are created in advance, this approach works quickly and transparently. *Text-to-image generation* models based on neural networks, such as Stable Diffusion, SDXL [2], FLUX and AuraFlow, use diffusion processes for creating images. This approach is more flexible, and generated images have high quality, but has problems with generation the right angles (Examples in Table 1). The second part is vectorizer. *Image vectorization* task has two main approaches: deterministic and based on machine learning. Deterministic algorithms are based on tracing. Algorithms based on machine learning, such as DiffVG and LIVE, use differentiable rasterization and error back propagation. The other example is EvoVec, this approach is based on evolutionary algorithm.

Large Language Models (LLM) are an another approach for generating vector images. LLMs process the text and create a new one based on it. Since vector images consist of SVG code, which is XML markup, LLMs generate a vector image in one iteration, therefore, this process is extremely fast. Unfortunately, the key disadvantage of existing LLMs is the low quality of generated vector images. SVG images have a complex structure, so current LLMs cannot efficiently manage the vector image generation task. Numerous papers report on the poor quality of vector image generation using LLMs.

Comprasion. The experiments include two parts — comparison with the complex generator and vectorizer approach and LLM approach. We conducted about 30 experiments with different prompts and got a similar distribution of metrics. To evaluate the visual correspondence between the generated image and the text prompt, we employ the CLIPScore [4] metric.

We compare our algorithm with the existing approaches based on a generator and a vectorizer. About 20 prompts describing apartment plans were generated for the model comparison and 10 plans were generated based on each prompt to obtain average metrics using models: Stable Diffusion, FLUX, AuraFlow, SDXL and SDXL with white loss. Then an image is provided to the input of vectorization algorithm. Compared vectorization algorithms are DiffVG (N=512), LIVE (N=16), EvoVec, SvgTracer and our algorithm, where N is the number of paths in the final vector image. Table 2 shows that our algorithm produces final vector images that demonstrate the best correspondence to the prompts they are based on. Beside, its operation time is one of the shortest compared to others. In our algorithm, paths number corresponds to the number of walls.

The following is a comparison of our method with such LLMs-based approaches as GPT-4 and DeepSeek-v3. Table 2 shows a low visual quality of vector plan generation by LLMs. Although the operation time of LLMs is short and the paths number is small, low compliance with prompt makes such generation inapplicable for our task.

Model	SD + DiffVG	AuraFlow + LIVE	SDXL + EvoVec	FLUX + SvgTrace	DeepSeek-v3	GPT-4	Our approach
Paths	512	16	37	94	12	9	12
Time, s.	2356.6	1971.4	778.6	19.2	8.5	6.7	23.3
CLIPScore	0.241	0.243	0.268	0.261	0.232	0.247	**0.288**

Table 2. Comparison of generation with one text prompt of our algorithm

Conclusion. We have presented the novel method for generating vector residential plans based on textual description. We designed a new loss function to generate raster images on a white background. Additionally, we have developed a new vectorization algorithm based on the Shi-Tomasi corner detection method. Table 2 shows that our algorithm works fast, generates better quality plans compared with all existing approaches with clean SVG structure without unnecessary paths.

Acknowledgments. This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No139-15-2025-010.