

# GenPlan: Generation Vector Residential Plans Based on the Textual Description

Egor Bazhenov<sup>1</sup>, Stepan Kasai<sup>1</sup>, Viacheslav Shalamov<sup>1</sup>, Valeria Efimova<sup>1</sup>

<sup>1</sup>ITMO University, Saint-Petersburg, Russia

egorbazhenov@niuitmo.ru, kasai.stepa@gmail.com, sslavian812@gmail.com, valeryefimova@gmail.com

## Abstract

Computer graphics is fundamental in science, industry, and digital communications. It consists of vector and raster components. Raster graphics characterized by pixel-based representations is easy to obtain and edit, but cannot be scaled. Conversely, vector graphics uses mathematical structures to define shapes and lines, thus rendering them scalable without resolution degradation. However, vector graphics is often more complex to create and edit. For designers and architects, vector graphics is more versatile and convenient, although it requires more time and computational resources to create. In this paper, we present a novel method GenPlan for generating vector residential plans based on textual description. GenPlan surpasses all existing solutions in visual quality, as it works with right angles and has flexible usage settings. We also present a new residential plan vectorization algorithm, which creates vector structured images based on a given raster plan.

**Code** — <https://github.com/CTLab-ITMO/GenPlan>

## Introduction

Currently, computer graphics plays an important role in manufacturing, marketing, design, architecture, digital communications, and in many other fields, which is confirmed by a large number papers in this field. Computer graphics has two main parts: vector and raster graphics. Raster images consist of pixels and have many formats with the most famous being PNG, BMP, JPG, and JPEG. This type of graphics is easy to create and edit; moreover, it is supported by numerous software products. However, raster graphics lacks scalability, and the image weight depends quadratically on the image quality. Vector graphics consists of paths that represent various geometric shapes such as squares, circles, lines, arcs, and others. Its images are scalable with small file size. The main disadvantage of vector graphics is that such images are difficult for creating and editing, as they require to understand the structure of SVG images (Quint 2003). Nevertheless, designers and architects prefer this type of graphics for developing residential plans.

Global urbanization of population (Ichimura 2003) makes residential planning critical for real estate development.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

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 against the background of growing demand to save developers' time and resources. 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 main idea is that a generator creates a raster residential plan based on a textual description, and then a vectorizer converts a raster image to a vector one. The second approach is to generate a vector plan using a Large Language Model (LLM) (Raiaan et al. 2024). Since SVG images consist of code, this approach can be also used to generate residential plans. More information about each approach, their pros and cons, can found in the Related Works section .

In our paper, a raster-generation-vectorization approach is chosen as the baseline. Applying this approach we can customize and configure the generative algorithm. We have chosen two main models as raster generators: FLUX (Labs 2024) and AuraFlow (fal 2024). For floor plan vectorization, we have developed our own algorithm based on angle detection and analytical analysis methods.

As a result, we have developed the algorithm for generating vector residential plans based on textual description. Specifically, the main contributions are as follows:

1. We propose GenPlan — the novel method for generating vector residential plans based on textual description that provides residential plans in SVG file format without extra paths.
2. We suggest the method to adapt a pre-trained image generation model to the generation of raster images similar to vector ones for simpler vectorization.
3. We have developed the new algorithm of plan vectorization that allows obtaining right angles and straight walls in resulting vector representation, creating a structured vector image based on postprocessing analysis.
4. We make our implementation publicly available. .

For more information about existing approaches, see the Related Works section . The Methods section describes in detail the generation algorithm itself and the subtleties of its implementation. In Comparisons and Experiments , the results of comparing the proposed algorithm with existing

solutions and ablation study are presented. The Conclusion section summarizes the paper.

## Related works

The problem 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. With benefits and drawbacks these approaches have, the current open source algorithms do not seem to be effective or efficient in residential plan generation.

### Raster generation with vectorization

The first approach combines a raster generator and 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. Due to the flexibility of this approach we can configure and select the generator and vectorizer best suited for the task. This section describes various algorithms for raster generation and vectorization, as well as their features for the task of generating vector plans.

**Raster generation** For the task of raster generation based on the textual description the two main approaches are often applied: Rule-based generation models and Neural networks (Li et al. 2021). Each approach has its advantages and disadvantages in terms of the given task of raster plan generation.

**Rule-based generation models.** This approach to the task of generating raster images is deterministic and simple (Wang et al. 2023; Ávila Parra 2021). An image is selected based on the single textual template that can be customized depending on the task. As all possible variations of the images are created in advance, this approach works quickly and transparently.

Rule-based generation models have significant disadvantages (Efimova and Filchenkov 2022). First, they lack diversity due to a limited number of ready-made raster images. Second, they lack scalability and flexibility, as completely new images cannot be created. These disadvantages make Rule-based generation models narrowly applicable. At the same time, the approach is worth considering and applying in the case of a limited number of completely identical residential plans. In this article we present a universal algorithm for generating vector plans based on a textual description, so this option is not suitable for us.

**Stable Diffusion.** Stable Diffusion (Rombach et al. 2022) is one of the prominent models for text-to-image generation based on neural networks. This model consists of three main parts: a text encoder, an image information creator and an image decoder. The first part of Stable Diffusion encodes an input text prompt to token embeddings that correspond to each word in it. CLIPText model (Qin et al. 2023) is selected as the text encoder, but other text transformers are generally suitable. The second part of Stable Diffusion is an image information creator that takes text embeddings and noise as input, and then step-by-step optimizes input data in the latent space. U-Net model (Ronneberger, Fischer, and Brox 2015) is the main part of an image information creator. The third part of Stable Diffusion is the image decoder of

an autoencoder trained on image datasets. This decoder decodes output embeddings from processed information to a final image. As a result, the algorithm generates the image corresponding to the text description.

Stable Diffusion is one of the most powerful models for image generation, but for the task of generation raster plans based on textual description this model application has some problems. One of them is poor quality of generated plans due to this model lacks specialization. This model also has problems with generation the right angles. These drawbacks demonstrate the inability of Stable Diffusion to fully solve the problem of generating raster plans.

**Stable Diffusion XL.** This model (Podell et al. 2023) is an upgraded version of Stable Diffusion with a larger model size, better prompt understanding. A larger U-Net backbone (2.6 billion parameters vs. SD 1.5's 860 Million) and a dual-text-encoder system (CLIP ViT-L + OpenCLIP ViT-bigG) allows significant improving the quality of generated plans.

SDXL generates images with high quality, which makes it suitable for plan generation. However, like Stable Diffusion, this model cannot overcome the problem with the right angles. Despite this, we chose this model for our approach as one of raster image generators due to its excellent quality of images.

**FLUX.** One more model used for text-to-image generation is FLUX (Labs 2024). This model is based on the same neural network structure and has the same architecture as Stable Diffusion, but with some differences. FLUX distinguishes itself by using flow matching (Lipman et al. 2022), positional embeddings and parallel attention layers (Medina and Kalita 2018), therefore, it operates faster and trains more efficiently compared to Stable Diffusion.

For our task, FLUX is a good option for a raster generator due the high quality of generated images and fast operation, although it also has some problems with the right angles. Nevertheless, we chose this model as one of generators for our approach.

**AuraFlow.** This model (fal 2024) is inspired by Stable Diffusion 3 (Esser et al. 2024) and for now is the largest text-to-image generation model. AuraFlow has a number of benefits compared to the original model. First, the model authors remove MMDiT Blocks from most layers and replace them with large DiT Encoder blocks, which improved the model flops utilization at 6.8B scale by 15%. Second, a new text-to-image dataset is used for training the model. The authors also use Maximal Update Parametrization (muP) (Yang et al. 2022) to find the best hyperparameters for the model. This optimization has made the model one of the powerful models for text-to-image generation.

We also chose AuraFlow as one of generators, as this model completely satisfies requirements in our vector plan generation pipeline. It is worth noting that although the quality of generated plan is acceptable, there are some problems with the right angles.

**Vectorization** Image vectorization has a lot of solutions, but every algorithm has specific characteristics (Dziuba et al. 2024). This section presents the most popular and powerful vectorization algorithms, as well as their main features.

**DiffVG.** DiffVG (Li et al. 2020) is among the most advanced and effective vectorization algorithms. It uses a process of vector image gradual optimization so that its rasterization is similar to a bitmap image. The main idea of this approach is differentiable rasterization of a vector image. The algorithm transfers the resulting bitmap image to the loss function, calls the error back propagation algorithm and calculates gradients for the parameters of the original vector image. DiffVG optimizes the final vector image step-by-step by updating the vector image parameters. The vector image is adjusted based on the specified loss function, and then optimized to obtain the bitmap image as close to the original one as possible. The bitmap image and the number of paths in a final vector image are taken as input parameters.

For vectorizing vector plans this method has some problems. DiffVG is unable to automatically add or remove shapes and segments in a vector image — it only works with an existing structure. Instead, the method optimizes shape parameters, such as point coordinates, color, and line thickness, while preserving the original image topology. Moreover, the algorithm initially has count of paths in a final vector image. Besides that, this algorithm does not handle vectorization of the right angles well. Thus, although DiffVG is a powerful vectorization algorithm, it is not suitable for the task of plan vectorization.

**LIVE.** The other method of vectorization based on machine learning is Towards Layer-wise Image Vectorization (LIVE) (Ma et al. 2022). Its main idea is an interactive addition of vector shapes to final vector image and their subsequent optimization. The algorithm takes the initial bitmap image and the number of paths in the final vector image as input. After each iteration, LIVE searches for the largest monochrome area that does not correspond to the raster representation and adds a new path to the final vector image. Next, the algorithm optimizes the added shape based on two loss functions. The first one is responsible for the similarity of the vector image to its raster counterpart. The second loss function eliminates the self-intersection of paths.

LIVE shows low productivity and manages the task of vectorizing raster plan only partially. One of the problems is an initial parameter — the number of paths in the final vector image, since this value may be unknown. Besides, this algorithm produces odd angles instead of the right ones resulting in the final image that only partially corresponds to the initial raster image. The problems above make LIVE unsuitable in plan vectorization.

**Deterministic vectorization algorithms.** There exist many deterministic vectorization algorithms based on tracing (Hettinga, Echevarria, and Kosinka 2021; Packard, Worth, and Esfahbod; Zhou, Zheng, and Seah 2009). They work quickly due to the static analysis of the original bitmap image. The main disadvantage of this approach results from the creation of redundant paths in the vector image; the structure of such an image becomes overloaded and hardly manually editable.

For the task of vectorizing plans, this approach creates an unstructured vector image that is difficult to process. Despite fast operation, these algorithms are inconvenient to use for the task.

**EvoVec.** This vectorization algorithm is based on an evolutionary algorithm (Back 1996; Bergen and Ross 2012; Bazhenov et al. 2024a). The main idea of EvoVec (Bazhenov et al. 2024b) is step-by-step optimization of the final vector image based on mutations and crossovers. The initial population is the result of a deterministic vectorization algorithm. At each step of the algorithm, mutations change coordinates, delete unnecessary paths and segments and connect paths gradiently based on color. In addition, the crossover operator changes the paths in the population between individuals. The best individuals remain in the population after applying these operations. Thus, the final optimized vector image is obtained.

EvoVec works faster than machine learning based algorithms, such as LIVE or DiffVG, and the quality is higher than the result of the work of deterministic algorithms. However, this algorithm does not fully handle the task of vectorizing the raster plan, producing the final vector image with odd angles and unclear structure.

## Large Language Models

Another approach for generating vector images is Large Language Models (LLM) (Raiaan et al. 2024). 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 (Malashenko, Jarsky, and Efimova 2025; Cai et al. 2023).

In our research we use LLMs to compare the vector image generation quality. They work extremely fast but generate plans with a primitive structure. The approach is promising for task of generation vector plans based on textual description, however, current LLMs cannot fully tackle the challenge.

## Methods

Our paper presents an approach consisting of a raster generator and a vectorizer. This method was chosen as it is customizable and scalable. FLUX, AuraFlow and SDXL with white loss were selected as models for raster generation. More information about testing and comparison is provided in section Generation . For vectorization we have developed a new approach based on the Shi-Tomasi corner detection method (Shi et al. 1994) that can work with the right angles and straight walls. Our vectorization algorithm is described in section Vectorization .

## Generation

The first part of our approach is a raster generator. We already discussed models for generating raster images in section Related works . Among the models limitations, it is the white background that is also a challenge: the models have difficulties with generating images on a white background.

We assume that the issue results from the ambiguity of a prompt for image generation. More detailed prompts can contribute to handling the issue although their development is time-consuming. However, even a highly detailed prompt does not guarantee that white background will be generated. The image background may be in shades of gray, but for the human eye the difference is significant.

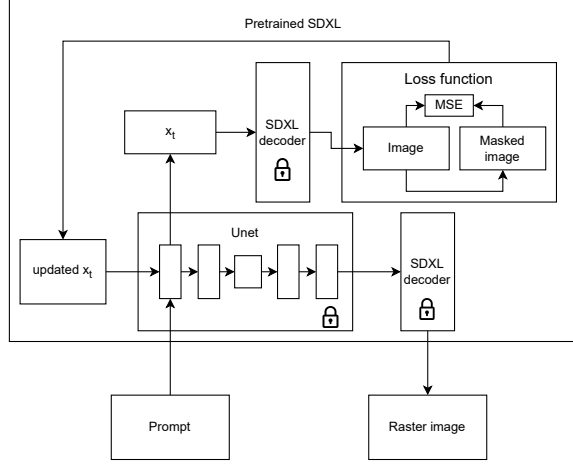


Figure 1: Image generation scheme with SDXL model using white loss.

We designed new loss function 1 to generate images on a white background:

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

where  $D$  — SDXL decoder,  $s$  — scale,  $x_0$  — vector without noise in latent space,  $x_t$  — vector in latent space with noise corresponding to step  $t$ ,  $\text{mask}$  — mask that defines the field that we would like to be white.

We also need to update  $x_t$  after this by following recurrent formula 2:

$$x_t = x_t - \text{mask}_{\text{latent}} \left( \Delta x_t \frac{1 - \bar{\alpha}_t}{\bar{\alpha}_t} \right), \quad (2)$$

where  $t$  — diffusion step,  $\Delta x_t$  — gradient of  $x_t$  according to  $L_{white}$ ,  $\text{mask}_{\text{latent}}$  — mask that is scaled to be used in latent space,  $\bar{\alpha}_t$  — cumulative product (Rombach et al. 2022).

Figure 1 demonstrates the scheme for this approach. We use the SDXL decoder to obtain an image from  $x_t$  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  $x_t$  according to formula 2.

## Vectorization

This section describes our algorithm for vectorizing a raster plan. It consists of three main parts — preprocessing, vectorization, and postprocessing. Each part performs its own function to obtain the final vector image.

---

### Algorithm 1: Vectorization algorithm

---

**Input:** Raster image

**Output:** Vector image

```

1:  $corners \leftarrow \emptyset, \text{valid\_rectangles} \leftarrow \emptyset.$ 
2:  $corners \leftarrow \text{find\_corners}(\text{image}).$ 
3:  $corners \leftarrow \text{average\_corners}(\text{corners})$ 
4: for  $\text{iteration} = 1, 2, \dots, N$  do
5:    $\text{rectangles} \leftarrow \emptyset.$ 
6:    $corners \leftarrow \text{average\_corners}(\text{corners}).$ 
7:   for  $\text{rectangle} \in \text{rectangles}$  do
8:     if  $\text{is\_valid\_rectangle}(\text{rectangle})$  then
9:        $corners \leftarrow \text{corners} + \text{rectangle.corners}.$ 
10:       $\text{valid\_rectangles} \leftarrow \text{valid\_rectangles} + \text{rectangle}.$ 
11:    end if
12:  end for
13: end for
14:  $\text{final\_svg} \leftarrow \text{create\_svg\_from\_rectangles}(\text{valid\_rectangles}).$ 
15: return  $\text{final\_svg}$ 

```

---

**Preprocessing** The first part of vectorization algorithm is preprocessing. The main idea of this part is to prepare the image for subsequent vectorization. Preprocessing is necessary to ensure that the final image has no unnecessary parts, such as interior items.

The algorithm of preprocessing converts each pixel of a black and white raster RGB image to monochrome. For this operation we use function 3:

$$f(p) = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B, \quad (3)$$

where  $p$  — pixel of image,  $R$  — red channel value,  $G$  — green channel value,  $B$  — blue channel value.

Then algorithm filters pixels by a threshold value. This value is one of our algorithm hyperparameters. It was chosen based on the experiments presented in section Ablation study.

In the final part of the preprocessing algorithm, the reverse conversion from monochrome to RGB occurs. As result of the preprocessing algorithm, the image becomes a clean and black and white plan.

**Vectorization Algorithm** The next part of vectorization is the vectorization algorithm. It converts the black and white raster preprocessed plan to a vector image using the Shi-Tomasi corner detection method (Shi et al. 1994). The structure of the algorithm ?? is presented below.

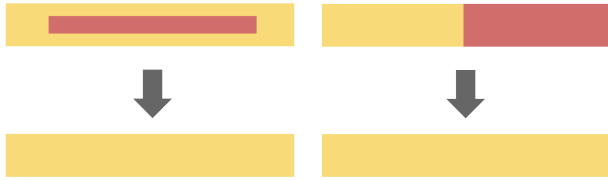
First, there is search for corners in the image. Shi-Tomasi corner detection method searches for angles based on the color difference. This method provides a list of wall corners. We average the corners coordinates so that they are on the same straight lines, also solving the problem of crooked walls.

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. Caching and preprocessing of the original bitmap image is implemented for fast filtering.

After suitable rectangles, which correspond to the walls in the initial raster plan, are selected, they are converted into vector paths. This algorithm part results in a vector plan corresponding to the initial raster plan.

**Postprocessing** The final part of vectorization is a post-processing algorithm. This part is critical as it allows obtaining the final vector plan with a correct SVG structure and without redundant paths.

The postprocessing algorithm contains two parts. The first part removes useless paths inside other paths (Fig. 2a). The second part combines paths that share a common part and form a rectangle together (Fig. 2b). These parts are necessary to preserve the structure of the image with each path responsible for a separate wall of the plan.



(a) Example where one path includes another (b) Example where two paths form a new rectangle

Figure 2: Examples of problems that the postprocessing algorithm solves.

The postprocessing algorithm results in a clean structured vector plan. Each path in the vector plan corresponds to a wall in the original bitmap image.

## Comparisons and Experiments

This section compares our approach with the existing ones. It includes two parts of comparison with the complex generator and vectorizer approach and LLM approach. Besides that, we performed a number of experiments to identify the best parameters of the vectorization algorithm.

### Experimental setup

The main hyperparameters and thresholds of our algorithm are shown below. The raster generator is SDXL with white loss. The filtering value for preprocessing is 10. The deviation value of 3 was chosen to average the coordinates of the angles. The following parameters were selected for the validation algorithm: the maximum value of dissimilar pixels is 2000, the maximum percentage of pixel differences is 15%, and the minimum width of a vector square is 3. These parameters were derived empirically based on a large number of experiments.

All of the experiments were launched on GPU — NVIDIA GeForce GTX 1080 Ti 11GB. GPU is necessary for correct operation of raster generation models and some vectorization algorithms, such as LIVE and DiffVG.

### Comparison with generator and vectorizer approach

We compare our algorithm with the existing approaches based on a generator and a vectorizer. The comparison consists of two parts: a comparison of plan generation models and a comparison of vectorization methods.

The first part demonstrates the quality of generating raster plans. 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. Table 3 shows the examples of how different models manage the task of generating a raster plan and averaged metrics of generation quality. To evaluate the visual correspondence between the generated image and the text prompt, we employ the CLIPScore metric (Hessel et al. 2021) and Image Reward metric (Xu et al. 2023). However, the Image Reward metric proved to be ineffective in capturing comparative quality, its results are irrelevant and unlikely, therefore, we do not present them in Table 3.

Based on the visual consistency of the prompt and averaged CLIPScore, it can be concluded which models manage to task of generating bitmap plans more effective. Our testing demonstrated that SDXL with white loss is the most suitable for generating raster plans.






Table 1: Quality comparison of basic SDXL and SDXL with white loss by FID score.

Model	FID ↓
Stable Diffusion XL	106.8
SDXL with white loss	99.9

Table 2: Parameters for raster image generation for quality comparison between basic SDXL against SDXL with white loss.

Parameter	Stable Diffusion XL	SDXL with white loss
Sampling steps	50	50
Guidance scale	7.5	7.5
Batch size	4	1
Resolution	512×512	512×512
GPU memory (T4, in GB)	8	14

Table 3: Comparison of plan generation models. Compared models are Stable Diffusion (Rombach et al. 2022), Stable Diffusion XL (Podell et al. 2023), FLUX (Labs 2024), AuraFlow (fal 2024), and SDXL with white loss. Text prompt for an example of image generation is "A minimalist 2D floor plan of an empty studio apartment, featuring clean black lines and white spaces, showcasing an open layout with designated areas for living, sleeping, and kitchen".

Model	Image ex-ample	Image CLIPScore	Average CLIPScore
Stable Dif-fusion		0.227	0.226 $\pm$ 0.012
SDXL		0.231	0.23 $\pm$ 0.014
FLUX		0.272	0.265 $\pm$ 0.009
AuraFlow		0.263	0.258 $\pm$ 0.01
SDXL with white loss		0.281	0.276 $\pm$ 0.006

We also compared SDXL with white loss and basic SDXL in Table 2. We used Frechet distance (FID) (Heusel et al. 2017) to measure realism and diversity of generated images. 550 images were generated from different prompts, all containing "on a white background".

In terms of quality, our approach, SDXL with white loss, is more effective than basic SDXL according to Table 1 by 6.9 (6.15%) FID.

The following is a comparison of vectorization algorithms. The same image is provided to the input of each vectorization algorithm for objective comparison. 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. We compare parameters such as the number of paths, operation time and matching the prompt based on CLIPScore and visual representation.

Table 4: Comparison of our algorithm with existing algorithms of vectorization. The same image is provided to the input. The following vectorization algorithms are selected : DiffVG, LIVE, EvoVec, SvgTracer and our algorithm. The table shows algorithms metrics (vector image, number of paths, operation time and CLIPScore).






Vectorizer	Vector image	Paths	Time, s.	CLIPScore
DiffVG		512	2338.8	0.274
LIVE		16	1953.6	0.271
EvoVec		37	760.8	0.282
SvgTracer		94	1.4	0.279
Ours		12	23.3	0.288

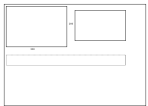

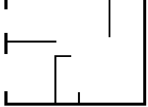
Table 4 shows that our vectorization algorithm produces final vector images that demonstrate the best correspondence to the prompts they are based on. Beside, its operation time is one of the shortest compared to others. The number of paths in our algorithm corresponds to the number of walls. For LIVE and DiffVG the number of paths in the final vector image is one the input parameters. EvoVec and SvgTracer generate extra paths making the final vector image unstructured.

### Comparison with LLM approach

The following is a comparison of GenPlan with such LLMs-based approaches as GPT-4 (Achiam et al. 2023) and DeepSeek-v3 (Liu et al. 2024). The main parameters for comparison are the number of paths, operation time and correspondence to the text prompt based on CLIPScore and visual representation.

Table 5 shows a low visual quality of vector plan generation by LLMs. Although the operation time of LLMs is short and the number of paths is small, low compliance with prompt makes such generation inapplicable for our task at

Table 5: Comparison of our algorithm with existing algorithms based on LLMs GPT-4 (Achiam et al. 2023) and DeepSeek-v3 (Liu et al. 2024). The table shows metrics of algorithms: number of paths, operation time and CLIPScore.

Approach	Vector image	Paths	Time, s.	CLIPScore
GPT-4		9	6.7	0.247
DeepSeek-v3		12	8.5	0.232
GenPlan		12	23.3	0.288

the moment.

### Ablation study

In this section, we discuss hyperparameters of our algorithm and their impact on the final result. The main parts are filtering pixels in preprocessing, obtaining the averaged coordinates of corners, and determining the validity of rectangles. Each of these parts has a specific list of parameters that can be configured for a specific task. We have selected the most optimal parameters for the task of generating plans.

First, the method of filtering pixel for cleaning image has one parameter — a threshold, which is the minimum value of black. If the pixel color is less than the threshold value, the pixel is white.

Table 6: An effect of the color filtering parameter on the final raster result.





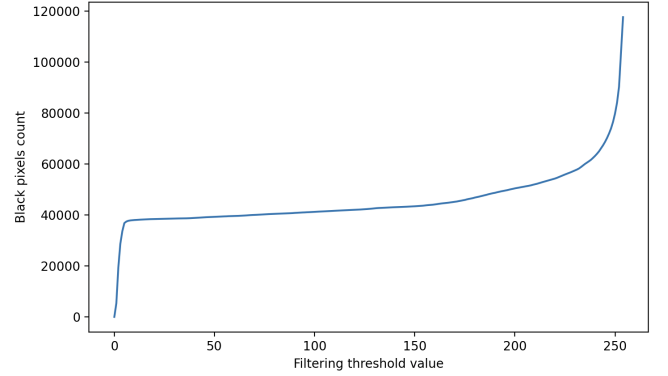
	1	10	50	150
Image				
Black pixels count	5301	37990	39237	43381

Table 6 and Figure 3 show that the minimum value of black change the final preprocessed raster image. It is essential to select an optimal filtering threshold value that maintains the integrity of the fundamental image information while simultaneously minimizing the influence of noise. The most appropriate value is 10 based on our experiments.

Second, the method of averaging coordinates of corners has one value — a deviation in pixels from the original

Figure 3: Graph of the dependence of the black pixels number on the filtering threshold value.



straight line. This parameter is imperative for obtaining straight walls in the final vector plan.

Figure 4: An example of a problem that requires coordinate averaging. The influence of the deviation value ( $D$ ) for the final result.

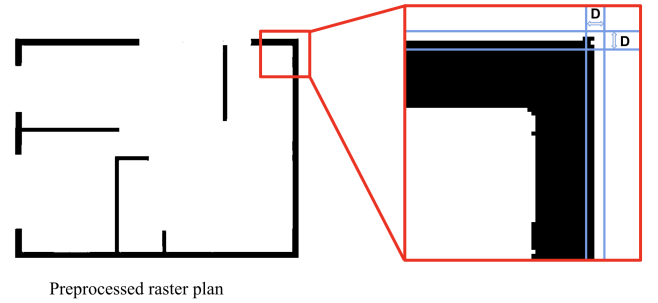


Figure 4 shows that the deviation value critically influences on the final vector image. The parameter should be chosen to mitigate issues related to irregular corner detection while preserving walls. Value 3 was chosen as the most suitable parameter for this averaging method.

Next, we present the validation method of rectangles as a part of our vectorization algorithm. This method aims to evaluate the similarity of a vector rectangle with a wall in the original raster plan. The similarity is calculated based on matching pixels between vector and bitmap images. There are three main values to correctly calculate the number of similar pixels. The first threshold is the maximum value of dissimilar pixels, the second threshold is the maximum percentage of pixel differences, and the third threshold is the minimum thickness of a vector square. Each parameter has its own logic for influencing vector validation. If the number of different pixels between the vector representation and the raster representation is greater than the maximum difference value, then this representation is not added to the



final vector result. The same logic applies to the maximum percentage of pixel differences. The minimum thickness of a vector square filters out thin vector paths based on width and height of square.

Table 7: Dependence of the final vector result on the maximum pixel differences percentage and the maximum value of dissimilar pixels.










	Maximum pixel differences percentage				
	5%	15%	25%		
Image				500	Maximum value of dissimilar pixels
Paths	6	11	12		
Image				2000	
Paths	6	12	13		
Image				5000	
Paths	6	13	13		

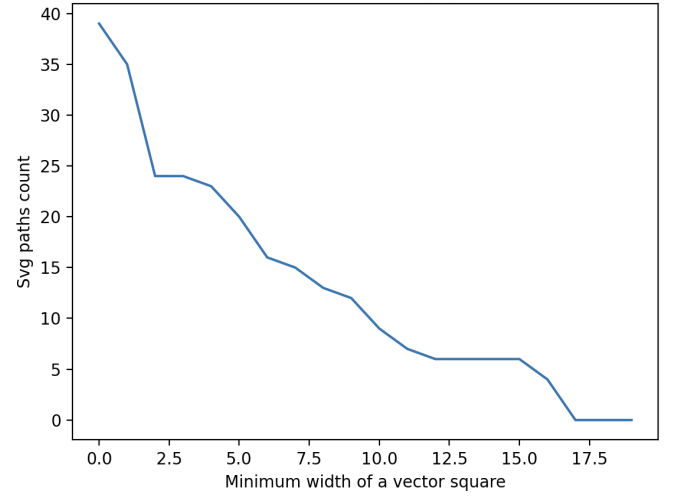
Table 7 shows that the maximum value of dissimilar pixels and the maximum percentage of pixel differences affect the number of paths in the final vector image. Higher values of these parameters generate more paths and vice versa. Thus, to obtain a more stable and correct final vector result, the best parameter for the maximum pixel differences percentage is 15% and for the maximum value of dissimilar pixels is 2000.

Figure 5 shows the dependence of the final vectorization image on the lines thickness. Due to this parameter, a clean image without unnecessary lines is obtained. Analyzing the graph, we can conclude that the best thickness value is 3, since in this case the final vector image contains no unnecessary paths and basic information is not lost.

## Conclusion

In this paper we have presented GenPlan, a new approach to the generation of vector residential plans based on textual description. We designed a new loss function to generate raster images on a white background that also contributes to create better residential plans, as they have clean lines and correspond well with text description. Additionally, we have developed a new vectorization algorithm based on the Shi-Tomasi corner detection method. The final vector plan features a clean SVG structure without unnecessary paths, with each element corresponding to a part of plan. Our algorithm generates better quality plans compared with all existing approaches. Further research directions include the generation three-dimensional architectural plans based on the

Figure 5: Graph of the dependence of the vector paths number on the minimum path thickness.



textual descriptions. We also intend to extend the scope of the current approach by specializing it for the creation of detailed floor plans, thereby enhancing its applicability in architectural design and urban planning.

## References

- Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Ávila Parra, R. 2021. *House generation using procedural modeling with rules*. Master's thesis, Universitat Politècnica de Catalunya.
- Back, T. 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.
- Bazhenov, E.; Jarsky, I.; Efimova, V.; and Muravyov, S. 2024a. Evolutionary image vectorization with variable curve number. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2083–2086.
- Bazhenov, E.; Jarsky, I.; Efimova, V.; and Muravyov, S. 2024b. EvoVec: Evolutionary Image Vectorization with Adaptive Curve Number and Color Gradients. In *International Conference on Parallel Problem Solving from Nature*, 383–397. Springer.
- Bergen, S.; and Ross, B. J. 2012. Automatic and interactive evolution of vector graphics images with genetic algorithms. *The Visual Computer*, 28(1): 35–45.
- Cai, M.; Huang, Z.; Li, Y.; Ojha, U.; Wang, H.; and Lee, Y. J. 2023. Leveraging large language models for scalable vector graphics-driven image understanding. *arXiv preprint arXiv:2306.06094*.



- Dziuba, M.; Jarsky, I.; Efimova, V.; and Filchenkov, A. 2024. Image vectorization: a review. *Journal of Mathematical Sciences*, 1–14.
- Efimova, V.; and Filchenkov, A. 2022. Text-based sequential image generation. In *Fourteenth International Conference on Machine Vision (ICMV 2021)*, volume 12084, 125–132. SPIE.
- Esser, P.; Kulal, S.; Blattmann, A.; Entezari, R.; Müller, J.; Saini, H.; Levi, Y.; Lorenz, D.; Sauer, A.; Boesel, F.; et al. 2024. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*.
- fal, T. 2024. AuraFlow. <https://blog.fal.ai/auraflow>.
- 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*.
- Hettinga, G.; Echevarria, J.; and Kosinka, J. 2021. Efficient image vectorisation using mesh colours. In *Italian Chapter Conference 2021: Smart Tools and Apps in Graphics*. Eurographics Association.
- Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.
- Ichimura, M. 2003. Urbanization, urban environment and land use: challenges and opportunities. In *Asia-Pacific Forum for Environment and Development, Expert Meeting*, volume 23, 1–14. Citeseer.
- Labs, B. F. 2024. FLUX. <https://github.com/black-forest-labs/flux>.
- Li, T.-M.; Lukáč, M.; Gharbi, M.; and Ragan-Kelley, J. 2020. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15.
- Li, Z.; Liu, F.; Yang, W.; Peng, S.; and Zhou, J. 2021. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12): 6999–7019.
- Lipman, Y.; Chen, R. T.; Ben-Hamu, H.; Nickel, M.; and Le, M. 2022. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.
- Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Ma, X.; Zhou, Y.; Xu, X.; Sun, B.; Filev, V.; Orlov, N.; Fu, Y.; and Shi, H. 2022. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16314–16323.
- Malashenko, B.; Jarsky, I.; and Efimova, V. 2025. Leveraging Large Language Models For Scalable Vector Graphics Processing: A Review. *arXiv preprint arXiv:2503.04983*.
- Medina, J. R.; and Kalita, J. 2018. Parallel attention mechanisms in neural machine translation. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, 547–552. IEEE.
- Packard, K.; Worth, C.; and Esfahbod, B. ??? Cairo.
- Podell, D.; English, Z.; Lacey, K.; Blattmann, A.; Dockhorn, T.; Müller, J.; Penna, J.; and Rombach, R. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.
- Qin, L.; Wang, W.; Chen, Q.; and Che, W. 2023. Cliptext: A new paradigm for zero-shot text classification. In *Findings of the Association for Computational Linguistics: ACL 2023*, 1077–1088.
- Quint, A. 2003. Scalable vector graphics. *IEEE MultiMedia*, 10(3): 99–102.
- Raiaan, M. A. K.; Mukta, M. S. H.; Fatema, K.; Fahad, N. M.; Sakib, S.; Mim, M. M. J.; Ahmad, J.; Ali, M. E.; and Azam, S. 2024. A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE access*, 12: 26839–26874.
- Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.
- Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III*, 234–241. Springer.
- Shi, J.; et al. 1994. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, 593–600. IEEE.
- Wang, J.; Fan, W.; Zhao, B.; Yang, Y.; and Zhang, Z. 2023. A Rule-Based Design Approach to Generate Mass Housing in Rural Areas of the North China Plain. *Buildings*, 13(10): 2539.
- Xu, J.; Liu, X.; Wu, Y.; Tong, Y.; Li, Q.; Ding, M.; Tang, J.; and Dong, Y. 2023. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36: 15903–15935.
- Yang, G.; Hu, E. J.; Babuschkin, I.; Sidor, S.; Liu, X.; Farhi, D.; Ryder, N.; Pachocki, J.; Chen, W.; and Gao, J. 2022. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*.
- Zhou, H.; Zheng, J.; and Seah, H. S. 2009. Converting Bitmap Images into Scalable Vector Graphics. In *Proceedings of the Korean Society of Broadcast Engineers Conference*, 435–440. The Korean Institute of Broadcast and Media Engineers.