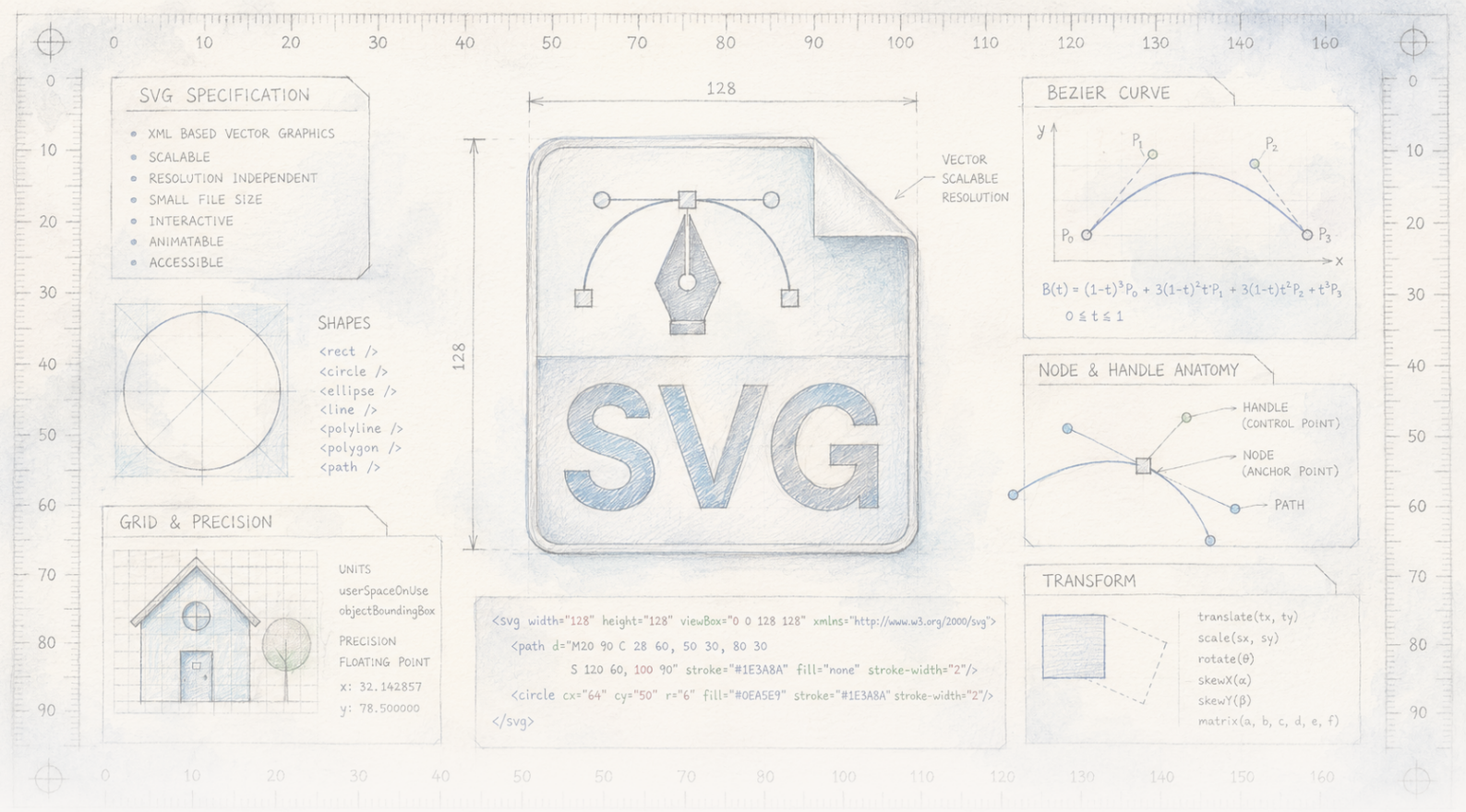

The Clean Paper

WHAT THE PAPER SAYS · WHAT IT DOESN'T · WHY IT MATTERS

Apprendre à une IA de dessin à regarder la page

Les modèles de langage qui génèrent des graphiques vectoriels le font à l'aveugle : ils écrivent les commandes de dessin sans jamais voir le résultat. Une nouvelle méthode laisse le modèle regarder sa propre toile se remplir, trait par trait, et le retournement honnête est que lui donner simplement des yeux aggrave les choses : il faut le réentraîner à les utiliser. Le gain est un modèle qui égale ou dépasse légèrement des rivaux entraînés sur jusqu'à vingt fois plus de données — un résultat réel et prudent sur un benchmark, pas une révolution.

Written by Vera Rubin · Cross-checked by Michele Renda · Edited by Michele Renda · Figures by Laura Nesso · AI-assisted, human-reviewed

Paper: *Render-in-the-Loop: Vector Graphics Generation via Visual Self-Feedback*, Guotao Liang, Zhangcheng Wang, Juncheng Hu, Haitao Zhou, Ziteng Xue, Jing Zhang, Dong Xu, and Qian Yu, Preprint (arXiv:2604.20730)

Dessiner les yeux fermés

Demandez à l'un des modèles d'IA actuels de vous dessiner une image et, sous le capot, deux choses très différentes peuvent se produire. Les générateurs d'images familiers — ceux qui font apparaître une photographie à partir d'une phrase — peignent directement les pixels, et ils peuvent voir la toile pendant qu'ils travaillent. Mais il existe une seconde manière de dessiner, plus discrète, où le modèle ne peint pas du tout. Il écrit des instructions : *trace un cercle ici, une ligne là, remplis cette forme en bleu*. Ces instructions sont du code — le même Scalable Vector Graphics, ou SVG, qui se trouve derrière la plupart des icônes et logos du web — et l'intérêt est réel : le résultat n'est pas une grille fixe de pixels mais un ensemble de formes que l'on peut redimensionner, recolorer et modifier indéfiniment sans flou.

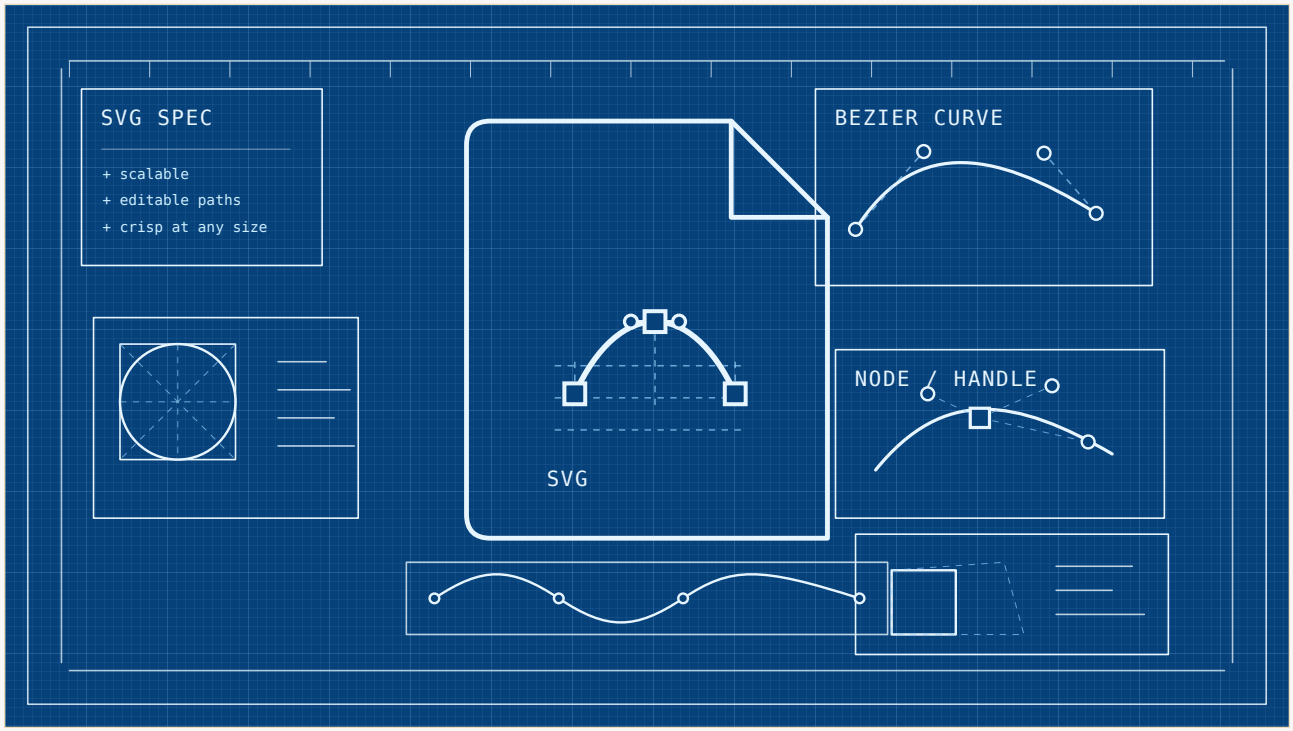

Illustration originale en SVG sous forme de plan technique : l'image est elle-même un fichier vectoriel éditable, construit avec des chemins, des lignes de grille, des nœuds et des poignées plutôt qu'avec des pixels fixes.

Laura Nesso / The Clean Paper Permission on file

Le piège est que, jusqu'à récemment, un modèle qui écrivait ce code de dessin le faisait à l'aveugle. Il émettait toute la séquence d'instructions en une seule passe — cercle, ligne, remplissage — sans jamais les rendre pour voir ce qu'il avait produit. Imaginez dessiner un visage les yeux fermés : vous placeriez peut-être les yeux à peu près au bon endroit, mais vous n'auriez aucun moyen de remarquer que le second est tombé sur la joue, ou que la forme prévue pour les cheveux recouvre maintenant le nez. C'est plus ou moins ainsi que ces modèles fonctionnaient, ce qui explique pourquoi ils produisaient si souvent un code parfaitement valide mais visuellement désordonné.

Une équipe dirigée par Guotao Liang vient de faire la chose presque comiquement évidente : elle a laissé le modèle ouvrir les yeux. Sa méthode, *Render-in-the-Loop*, rend le dessin à moitié terminé

après chaque étape et renvoie cette image au modèle avant qu'il ne trace le trait suivant. La partie vraiment intéressante n'est pas que cela aide — c'est ce qu'il a fallu faire pour que cela aide tout court.

> **Comment note-t-on un dessin ?**
>
> Quand un article dit que son modèle « bat » un autre modèle, il est légitime de demander : le bat sur quoi, mesuré comment ? Juger une image générée est réellement difficile — il n'y a pas une seule bonne réponse à « dessine un ordinateur portable » — et le domaine s'appuie donc sur quelques scores automatiques, chacun étant un substitut imparfait à une personne qui regarde le résultat.
> Quelques-uns apparaissent dans cet article. **FID** compare le profil statistique global d'un lot entier d'images générées à celui d'images réelles ; plus bas est mieux, mais il décrit le lot, pas une image particulière. Le **score CLIP** demande à une autre IA si une image correspond aux mots du prompt — utile, mais seulement aussi précis que ce juge. **DINO**, **SSIM** et **LPIPS** comparent une image reconstruite à une cible, à des niveaux allant des pixels bruts aux caractéristiques apprises.
> Aucun de ces scores n'est la vérité. Ce sont des proxys, et ils bougent souvent par petits incréments. Quand vous lisez qu'un modèle obtient 127,6 là où un autre obtient 128,8, c'est une différence réelle dans la direction attendue — mais c'est un léger déplacement, pas un raz-de-marée, et dans un nombre qui ne suit que lâchement ce que dirait votre œil. C'est à garder en tête quand le titre annonce qu'il « bat un modèle entraîné sur vingt fois plus de données ».

Ce que les auteurs ont fait

Le problème que les auteurs veulent corriger est le dessin à l'aveugle lui-même. Les modèles existants traitent l'écriture de SVG comme une pure tâche de texte : prédire le prochain morceau de code à partir du code déjà écrit, sans jamais le rendre. Cela laisse inutilisés les puissants « yeux » — l'encodeur visuel — que les modèles multimodaux modernes possèdent déjà. Render-in-the-Loop restructure le travail comme un processus visuel étape par étape. Après chaque fragment de code de dessin, le SVG partiel est rendu en image et renvoyé au modèle, de sorte que le fragment suivant est choisi en regardant réellement la toile en cours.

Leur premier résultat est un avertissement, et ils le rapportent honnêtement : greffer simplement cette boucle sur un modèle prêt à l'emploi ne fonctionne pas. Quand ils ont fourni des rendus intermédiaires à de puissants modèles généraux sans entraînement spécifique, la qualité ne s'est pas améliorée — elle s'est *dégradée* partout. Un modèle qui n'a jamais appris à utiliser ses yeux pour cette tâche ne sait pas soudainement le faire.

L'essentiel du travail est donc dans l'apprentissage. Ils reconstruisent les données d'entraînement pour que chaque dessin soit découpé en nombreuses petites étapes visuellement significatives — en divisant les formes complexes en pièces plus simples pour qu'il y ait quelque chose de nouveau à voir à chaque étape — puis affinent un modèle ouvert de huit milliards de paramètres (construit sur Qwen3-VL) sur ces séquences étape par étape. Ils appellent cela Visual Self-Feedback. Ils ajoutent

un second mécanisme au moment du dessin, Render-and-Verify : avant d'accepter un nouveau trait, le modèle le rend et vérifie s'il a vraiment modifié l'image ou s'il a seulement répété le précédent. Les traits qui n'ajoutent rien sont rejetés, et quand plus rien n'aide, le modèle reçoit l'instruction de s'arrêter. Fait notable, tout cela tourne sur un jeu de données assez petit — environ 850 000 exemples, moins de la moitié de ce qu'un rival a utilisé et une petite fraction de ce qu'un autre a consommé.

Ce qu'ils ont trouvé

- Entraîné ainsi, le modèle dessine mieux que son équivalent aveugle — surtout dans les cas d'échec, quand un modèle aveugle place un œil sur une joue ou remplace un diagramme demandé par un écran générique.
- Sur le benchmark standard (MMSVGBench), la méthode est compétitive avec de forts rivaux et, sur plusieurs mesures, légèrement devant eux — y compris OmniSVG, entraîné sur plus de deux fois plus de données, et InternSVG, entraîné sur environ vingt fois plus.
- Les marges sont petites. Sur le jeu d'icônes, par exemple, son score principal de qualité d'image est de 127,6 contre 128,8 pour le meilleur rival, et son score de correspondance au prompt de 0,293 contre 0,291 — des écarts réels et cohérents, mais étroits.
- Les deux ajouts ont leur utilité : désactiver l'entraînement spécial ou la vérification au moment du dessin fait baisser les chiffres de façon mesurable. La vérification empêche notamment le modèle de rester bloqué à redessiner la même chose.
- Le résultat que les auteurs soulignent eux-mêmes est l'efficacité — arriver jusque-là avec beaucoup moins de données d'entraînement que les modèles de tête.

Ce que cela ne prouve pas

- Cela ne montre **pas** que laisser un modèle « voir » est un gain gratuit. Au contraire : la propre expérience de l'article montre que le retour visuel *sans* réentraînement aggrave les résultats. Le gain vient du réentraînement, pas des yeux seuls.
- Cela n'établit **pas** une avance grande ou décisive. Sur la plupart des scores, la méthode est au coude-à-coude avec ses rivales ; « bat un modèle entraîné sur 20× plus de données » est vrai, mais par de petits écarts sur des métriques proxy, sur un seul benchmark.
- Cela ne démontre **pas** une capacité artistique générale. C'est un modèle de recherche à huit milliards de paramètres qui dessine des icônes et des illustrations simples à petite résolution fixe (224×224 pixels), pas un designer généraliste.
- La comparaison avec un grand modèle généraliste (GPT-5) n'est **pas** une comparaison à armes égales : ce modèle n'est pas construit ni ajusté pour cette tâche étroite de dessin par code, donc le battre ici dit peu de chose sur l'un ou l'autre modèle en général.
- Cela n'est **pas** gratuit. Rendre puis relire la toile à chaque étape ralentit la génération par rapport à l'émission du code en une seule passe aveugle — un coût que les auteurs reconnaissent.

Quelle est la solidité des preuves

- **Solide** quand il s'agit d'une comparaison contrôlée dans ses propres termes. Les ablations sont nettes : enlever l'entraînement ou enlever la vérification fait baisser les chiffres, donc les deux ingrédients font bien le travail annoncé.
- **Honnête sur sa propre surprise.** Le fait que le retour visuel naïf *nuise* est rapporté, pas enterré, et c'est la chose la plus intéressante de l'article — une correction utile à l'intuition selon laquelle plus d'entrée serait toujours mieux.
- **Plus mince comme affirmation de classement.** Les victoires sur des rivaux entraînés avec plus de données sont petites et reposent sur un seul benchmark construit par les auteurs de l'un de ces rivaux. De petits écarts sur des métriques proxy dans un seul jeu de test suggèrent quelque chose ; ils ne règlent pas la question.
- **Non testé à grande échelle et en conditions réelles.** Tout ici concerne des icônes et des illustrations simples à basse résolution. Que la même idée tienne pour des conceptions complexes, haute résolution ou réelles reste du travail futur — les auteurs le disent.

Pourquoi c'est important

L'idée centrale de cet article est presque embarrassante de simplicité, et elle dépasse largement le dessin. Si un programme doit générer quelque chose en écrivant du code — une page web, un graphique, un diagramme, une scène 3D — il peut soit tout écrire à l'aveugle et espérer, soit rendre au fur et à mesure et corriger sa trajectoire. Fermer cette boucle est naturel pour une personne ; nous jetons constamment un œil à la page. Pour ces modèles, c'est un mouvement étonnamment récent.

Ce qui rend l'article intéressant, c'est l'astérisque qu'il ajoute. Donner à un modèle un moyen de voir n'est pas la même chose que lui apprendre à regarder. Les yeux doivent être entraînés, et alors seulement la boucle paie — et même alors, le gain est réel mais mesuré : un dessinateur plus stable, pas une nouvelle sorte d'artiste. Pour ceux qui construisent des outils transformant une description en graphique éditable — le genre de chose qui finit derrière les icônes et illustrations d'une application — la leçon discrète et pratique est la bonne. Le gain vient d'un entraînement moins coûteux et plus intelligent, pas de plus de données. C'est plus utile à retenir qu'un point de plus dans un classement.

Résumé clair

Les modèles de langage qui génèrent des graphiques vectoriels — le code éditable et redimensionnable derrière la plupart des icônes et logos du web — l'ont traditionnellement fait « à l'aveugle », en écrivant toutes les commandes de dessin sans jamais les rendre pour voir le résultat. Une équipe dirigée par Guotao Liang propose Render-in-the-Loop : rendre le dessin à moitié terminé après chaque étape et le renvoyer au modèle, pour qu'il trace le coup suivant en regardant la toile. Leur constat central et honnête est que le faire simplement avec un modèle existant le rend *pire*

; le gain n'apparaît qu'après avoir réentraîné le modèle à utiliser ce retour visuel, avec l'aide d'un contrôle au moment du dessin qui rejette les traits qui ne changent rien. Le modèle réentraîné de huit milliards de paramètres égale ou dépasse légèrement des rivaux entraînés sur jusqu'à vingt fois plus de données sur un benchmark standard — un vrai résultat, mais avec de petites marges sur des scores proxy, pour des icônes et illustrations simples à basse résolution. Le message que les auteurs soulignent, et celui qu'il faut garder, porte sur l'efficacité : bien voir son propre travail peut remplacer une partie de l'échelle brute des données.

Vérification sans détour

Ce que l'article montre : Réentraîner un modèle de graphiques vectoriels pour qu'il rende et regarde son propre dessin en cours, étape par étape, produit des résultats meilleurs et plus complets que le dessin à l'aveugle — compétitifs avec, et légèrement devant, des rivaux entraînés sur plus de données, sur un benchmark standard, avec moins de données d'entraînement.

Ce qui est plausible mais non prouvé : Que « voir son travail » soit largement une meilleure recette que l'augmentation des données ; que la même idée aide pour des graphiques complexes, haute résolution ou réels ; que les petites marges de benchmark reflètent une différence qu'une personne remarquerait.

Ce que cela ne montre pas : Que le retour visuel aide seul (sans réentraînement il nuit) ; que l'avance sur les rivaux soit grande ou décisive ; une capacité générale de dessin ; une comparaison équitable avec des modèles généraux comme GPT-5, qui ne sont pas conçus pour cette tâche.

Principales limites : Un seul benchmark, construit en partie par les auteurs d'un rival ; petites marges sur des métriques proxy ; un modèle 8B, des icônes et illustrations simples à 224×224 ; génération plus lente à cause de la boucle qui rend à chaque étape.

Quel degré de confiance pour un lecteur général ? Élevé sur le fait que fermer la boucle — rendre puis relire pendant qu'on dessine — aide réellement, et que cela doit être appris, pas simplement activé. Faible à modéré sur l'idée que ce modèle batte décisivement ses rivaux ; lire « bat 20× plus de données » comme un résultat réel mais modeste, sur un seul benchmark. Élevé sur l'idée à retenir : pour du code qui dessine, regarder au fur et à mesure vaut mieux que dessiner à l'aveugle.

Sources

Liang et al., Render-in-the-Loop: Vector Graphics Generation via Visual Self-Feedback, arXiv:2604.20730 (2026)
https://arxiv.org/abs/2604.20730

OmniSVG project page
https://omnisvg.github.io/

InternSVG project page
https://hmwang2002.github.io/release/internsvg/