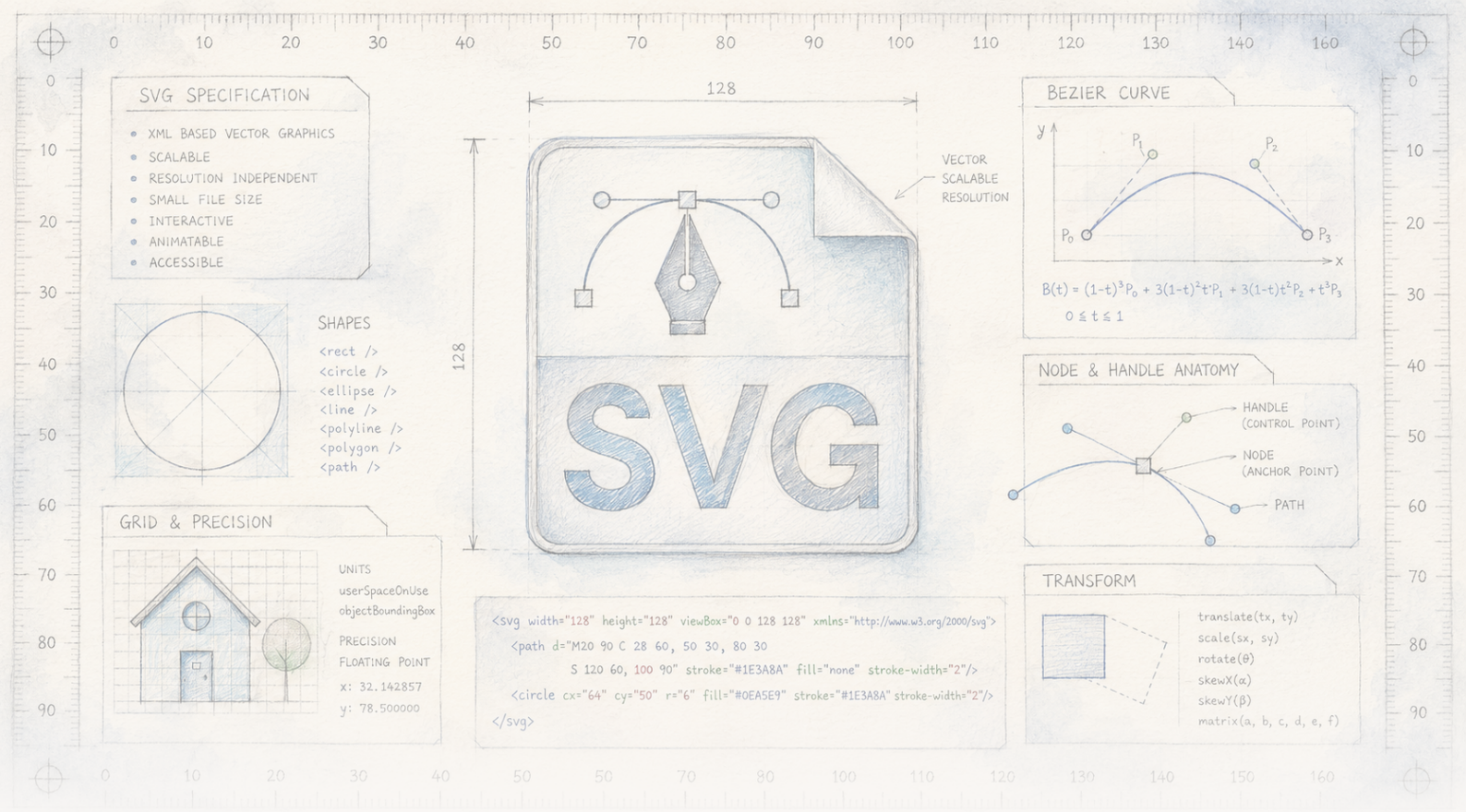


The Clean Paper

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

Een tekenende AI leren naar de pagina te kijken

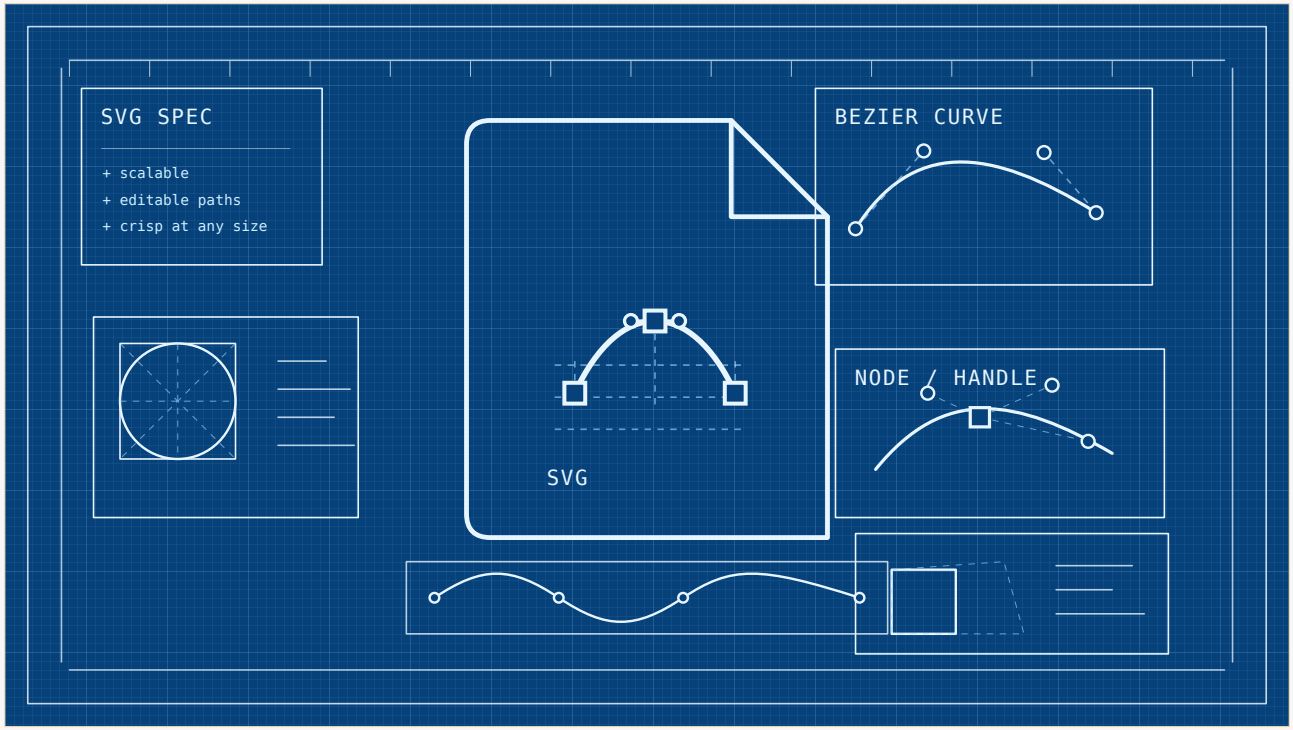
Taalmodellen die vectorafbeeldingen genereren doen dat blind — ze schrijven de tekencommando's uit zonder het resultaat ooit te zien. Een nieuwe methode laat het model toekijken hoe zijn eigen canvas zich vult, streek voor streek, en de eerlijke wending is dat het simpelweg ogen geven de zaken verslechtert: het moet hertraind worden om ze te gebruiken. De beloning is een model dat rivalen die op tot twintig keer meer data zijn getraind evenaart of nipt voorbijstreeft — een echt, zorgvuldig resultaat op één benchmark, geen revolutie.

Written by Vera Rubin · Cross-checked by Michele Renda · Edited by Michele Renda · Figures by Laura Nesso and Nova Vela · 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)

Tekenen met je ogen dicht

Vraag een van de huidige AI-modellen om een tekening voor je te maken en onder de motorkap gebeurt een van twee heel verschillende dingen. De vertrouwde beeldgeneratoren — die uit één zin een foto toveren — schilderen pixels rechtstreeks, en zij kunnen het canvas zien terwijl ze werken. Maar er bestaat een tweede, stillere manier van tekenen, waarbij het model helemaal niet schildert. Het schrijft instructies: *teken hier een cirkel, daar een lijn, vul deze vorm blauw*. Die instructies zijn code — dezelfde Scalable Vector Graphics, oftewel SVG, die achter de meeste iconen en logo's op het web zit — en de aantrekkingskracht is echt: het resultaat is geen vast pixelraster maar een verzameling vormen die je eindeloos kunt herschalen, herkleuren en bewerken zonder dat het wazig wordt.



Origineel SVG-blauwdrukkunstwerk: het beeld is zelf een bewerkbaar vectorbestand, opgebouwd uit paden, rasterlijnen, knooppunten en handvatten in plaats van vaste pixels.

Laura Nesso / The Clean Paper Permission on file

Het addertje is dat een model dat deze tekencode schrijft, dat tot voor kort blind deed. Het schreef de hele reeks instructies in één keer uit — cirkel, lijn, vulling — zonder ze ooit te renderen om te zien wat het had gemaakt. Stel je voor dat je met je ogen dicht een gezicht schetst: je krijgt de ogen misschien ongeveer waar ogen horen, maar je zou niet kunnen merken dat het tweede op de wang is beland, of dat de vorm die je voor het haar tekende nu over de neus ligt. Zo werkten deze modellen min of meer, en daarom produceerden ze zo vaak code die volkomen geldig was en toch visueel een rommeltje.

Een team onder leiding van Guotao Liang heeft nu het bijna komisch voor de hand liggende gedaan: ze laten het model zijn ogen openen. Hun methode, *Render-in-the-Loop*, rendert de halfafgemaakte tekening na elke stap en geeft dat beeld terug aan het model voordat het de volgende streek zet. Het werkelijk interessante is niet dát dit helpt — het is wat ze moesten doen om het überhaupt te laten

helpen.

> **Hoe geef je een tekening een cijfer?**
>
> Wanneer een paper zegt dat zijn model een ander “verslaat”, is het een eerlijke vraag: verslaat het waarin, en hoe gemeten? Een gegenereerd beeld beoordelen is oprecht moeilijk — er is geen enkel juist antwoord op “teken een laptop” — dus leunt het veld op een handvol automatische scores, elk een onvolmaakte plaatsvervanger voor een mens die naar het resultaat kijkt.
> Een paar ervan komen in dit paper voor. **FID** vergelijkt de statistische smaak van een hele batch gegenereerde beelden met echte; lager is beter, maar het beschrijft de batch, niet één afzonderlijk beeld. **CLIP score** vraagt een aparte AI of een beeld overeenkomt met de woorden van de prompt — nuttig, maar slechts zo scherp als die rechter. **DINO**, **SSIM** en **LPIPS** vergelijken een gereconstrueerd beeld met een doel, op niveaus van rauwe pixels tot geleerde kenmerken.
> Geen van deze is de waarheid. Het zijn benaderingen, en ze bewegen doorgaans in kleine stapjes. Als je leest dat het ene model 127.6 scoort waar een ander 128.8 scoort, is dat een echt verschil in de bedoelde richting — maar het is een duwtje, geen aardverschuiving, en dan nog in een getal dat maar losjes volgt wat je oog zou zeggen. Goed om vast te houden wanneer de kop luidt: “verslaat een model dat op twintig keer zoveel data is getraind.”

Wat de auteurs deden

Het probleem dat de auteurs wilden verhelpen is het blinde tekenen zelf. Bestaande modellen behandelen het schrijven van SVG als een pure teksttaak: voorspel het volgende stuk code uit de code tot dusver, en render het nooit. Daarmee blijven de krachtige “ogen” — de vision-encoder — die moderne multimodale modellen toch al bezitten, ongebruikt. Render-in-the-Loop herstructureert de klus als een stapsgewijs visueel proces. Na elk fragment tekencode wordt de gedeeltelijke SVG naar een beeld gerenderd en als plaatje aan het model teruggegeven, zodat het volgende fragment wordt gekozen terwijl het model daadwerkelijk naar het canvas tot dusver kijkt.

Hun eerste bevinding is een waarschuwing, en het siert hen dat ze die onomwonden rapporteren: deze lus simpelweg vastschroeven aan een bestaand kant-en-klaar model werkt niet. Toen ze tussentijdse renders aan sterke algemene modellen voerden zonder enige speciale training, verbeterde de kwaliteit niet — ze ging over de hele linie *achteruit*. Een model dat nooit is geleerd zijn ogen hiervoor te gebruiken, kan dat niet opeens.

Het meeste werk zit dus in het aanleren. Ze bouwen de trainingsdata om zodat elke tekening wordt opgedeeld in vele kleine, visueel betekenisvolle stappen — complexe vormen worden gesplitst in eenvoudiger stukken zodat er in elke fase iets nieuws te zien is — en finetunen vervolgens een open model van acht miljard parameters (gebouwd op Qwen3-VL) op deze stapsgewijze reeksen. Ze noemen dit Visual Self-Feedback. Tijdens het tekenen voegen ze een tweede mechanisme toe, Render-and-Verify: voordat elke nieuwe streek wordt geaccepteerd, rendert het model die en controleert het of hij het

beeld werkelijk heeft veranderd, of enkel de vorige herhaalde. Streken die niets toevoegen worden weggegooid, en wanneer niets meer helpt, wordt het model aangespoord te stoppen. Opvallend genoeg draait dit alles op een vrij kleine dataset — ongeveer 850.000 voorbeelden, minder dan de helft van wat één rivaal gebruikte en een kleine fractie van die van een ander.

Wat ze vonden

- Zo getraind tekent het model beter dan zijn blinde tegenhanger — het zichtbaarst bij de faalgevallen, waar een blind model een oog op een wang zet, of een gevraagd staafdiagram overslaat en in plaats daarvan een generieke monitor uitschrijft.
- Op de standaardbenchmark (MMSVGBench) komt de methode als concurrerend uit de bus met — en op verschillende maten net iets vóór — sterke rivalen, waaronder OmniSVG, getraind op ruim twee keer zoveel data, en InternSVG, getraind op grofweg twintig keer zoveel.
- De marges zijn klein. Op de iconenset bijvoorbeeld staat zijn belangrijkste beeldkwaliteitsscore op 127.6 tegen 128.8 voor de beste rivaal, en zijn prompt-matchscore op 0.293 tegen 0.291 — echt en consistent, maar mager.
- Beide toegevoegde onderdelen verdienen hun plek: schakel de speciale training uit, of de verificatie tijdens het tekenen, en de cijfers zakken meetbaar. Vooral de verificatiestap voorkomt dat het model blijft hangen en steeds hetzelfde opnieuw tekent.
- Het resultaat dat de auteurs zelf benadrukken is efficiëntie — zo ver komen met veel minder trainingsdata dan de koplopers gebruikten.

Wat dit niet bewijst

- Het toont **niet** aan dat een model laten “zien” een gratis winst is. Integendeel: het eigen experiment van het paper laat zien dat visuele feedback *zonder* hertraining de zaken verslechtert. De winst komt van de hertraining, niet van de ogen alleen.
- Het vestigt **geen** grote of beslissende voorsprong. Op de meeste scores gaat de methode nek-aan-nek met haar rivalen; “verslaat een model getraind op 20× de data” is waar, maar met duwtjes op proxymetrieken, op één benchmark.
- Het demonstreert **geen** algemene artistieke vaardigheid. Dit is een onderzoeksmodel van acht miljard parameters dat iconen en eenvoudige illustraties tekent op een kleine vaste resolutie (224×224 pixels), geen designer voor algemeen gebruik.
- De vergelijking met een groot algemeen model (GPT-5) is **geen** vergelijking van gelijken: dat model is niet gebouwd of afgesteld voor deze smalle codetekentaak, dus het hier verslaan zegt weinig over beide modellen in het algemeen.
- Het komt **niet** gratis. Bij elke stap het canvas renderen en opnieuw inlezen maakt het genereren trager dan de code in één blinde run uitschrijven — een kostenpost die de auteurs erkennen.

Hoe sterk is het bewijs

- **Solide** waar het een gecontroleerde vergelijking op eigen voorwaarden is. De ablaties zijn schoon: haal de training weg, of de verificatie, en de cijfers zakken — de twee ingrediënten doen dus echt het werk dat de auteurs claimen.
- **Eerlijk over de eigen verrassing.** De bevinding dat naïeve visuele feedback *schaadt* wordt gerapporteerd, niet weggemoffeld, en het is het interessantste van het paper — een nuttige correctie op de intuïtie dat meer input altijd beter is.
- **Dun waar het een ranglijstclaim is.** De overwinningen op rivalen met meer data zijn klein en leven op één enkele benchmark, gebouwd door de auteurs van een van die rivalen. Kleine marges op proxymetrieken op één testset zijn suggestief, niet beklonken.
- **Onbeproefd op schaal en in het wild.** Alles hier is iconen en eenvoudige illustraties op lage resolutie. Of hetzelfde idee standhoudt voor complex, hoge-resolutie- of echt ontwerpwerk wordt overgelaten aan toekomstig werk — dat zeggen de auteurs zelf.

Waarom het ertoe doet

Het idee in het hart van dit paper is bijna gênant simpel, en het reikt ver voorbij het tekenen. Als een programma iets gaat genereren door code te schrijven — een webpagina, een grafiek, een diagram, een 3D-scène — kan het óf het geheel blind schrijven en hopen, óf gaandeweg renderen en bijsturen. Die lus sluiten is voor een mens een tweede natuur; we werpen voortdurend een blik op de pagina. Voor deze modellen is het een verrassend recente zet.

Wat het paper het lezen waard maakt, is de asterisk die het eraan hangt. Een model een manier geven om te zien is niet hetzelfde als het leren kijken. De ogen moeten getraind worden, en pas dan betaalt de lus zich uit — en zelfs dan is de opbrengst echt maar gematigd: een vastere tekenhand, geen ander soort kunstenaar. Voor wie gereedschap bouwt dat een beschrijving omzet in een bewerkbare afbeelding — het soort dat uiteindelijk achter de iconen en illustraties van een app zit — is de stille, praktische les de nuttige. De winst kwam hier van goedkopere, slimmere training, niet van meer data. Dat is iets beters om te leren dan nóg een punt op een ranglijst.

Heldere samenvatting

Taalmodellen die vectorafbeeldingen genereren — de bewerkbare, herschaalbare code achter de meeste webiconen en logo's — deden dat van oudsher "blind": ze schreven alle tekencommando's uit zonder ze ooit te renderen om het resultaat te zien. Een team onder leiding van Guotao Liang stelt Render-in-the-Loop voor: render de halfafgemaakte tekening na elke stap en geef die terug, zodat het model de volgende streek zet terwijl het naar het canvas kijkt. Hun centrale, eerlijke bevinding is dat dit simpelweg toepassen op een bestaand model het *slechter* maakt; de winst verschijnt pas nadat het model is hertraind om de visuele feedback te gebruiken, geholpen door een controle tijdens het tekenen die streken weggooit die niets veranderen. Het hertrainde model van acht miljard

parameters evenaart of verslaat nipt rivalen die op tot twintig keer meer data zijn getraind, op een standaardbenchmark — een echt resultaat, maar met kleine marges op proxyscores, voor iconen en eenvoudige illustraties op lage resolutie. De les die de auteurs benadrukken, en die het bewaren waard is, gaat over efficiëntie: je werk goed zien kan pure dataschaal vervangen.

Zonder omwegen

Wat het paper laat zien: Een vectorgrafiekmodel hertrainen om zijn eigen halfafgemaakte tekening stap voor stap te renderen en te bekijken, levert betere en volledigere resultaten op dan blind tekenen — concurrerend met, en net iets vóór, rivalen met meer data op één standaardbenchmark, met minder trainingsdata.

Wat plausibel maar niet bewezen is: Dat "je werk zien" in brede zin een beter recept is dan data opschalen; dat hetzelfde idee zal helpen bij complexe, hoge-resolutie- of echte grafische toepassingen; dat de kleine benchmarkmarges een verschil weerspiegelen dat een mens werkelijk zou opmerken.

Wat het niet laat zien: Dat visuele feedback op zichzelf helpt (zonder hertraining schaadt ze); dat de voorsprong op rivalen groot of beslissend is; algemene tekenvaardigheid; een eerlijke directe confrontatie met algemene modellen zoals GPT-5, die niet voor deze taak zijn gebouwd.

Belangrijkste beperkingen: Eén benchmark, deels gebouwd door de auteurs van een rivaal; kleine marges op proxymetrieken; een 8B-model, iconen en eenvoudige illustraties op 224×224; tragere generatie door de render-bij-elke-stap-lus.

Hoeveel vertrouwen zou een algemene lezer moeten hebben? Hoog dat het sluiten van de lus — renderen en opnieuw inlezen terwijl je tekent — werkelijk helpt, en dat het erin getraind moet worden, niet zomaar aangezet. Laag tot matig dat dit specifieke model zijn rivalen beslissend verslaat; behandel de regel "verslaat 20× de data" als een echt maar bescheiden resultaat op één benchmark. Hoog voor het ene idee dat het onthouden waard is: voor code die tekent, wint kijken terwijl je tekent van blind tekenen.

Bronnen

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/`