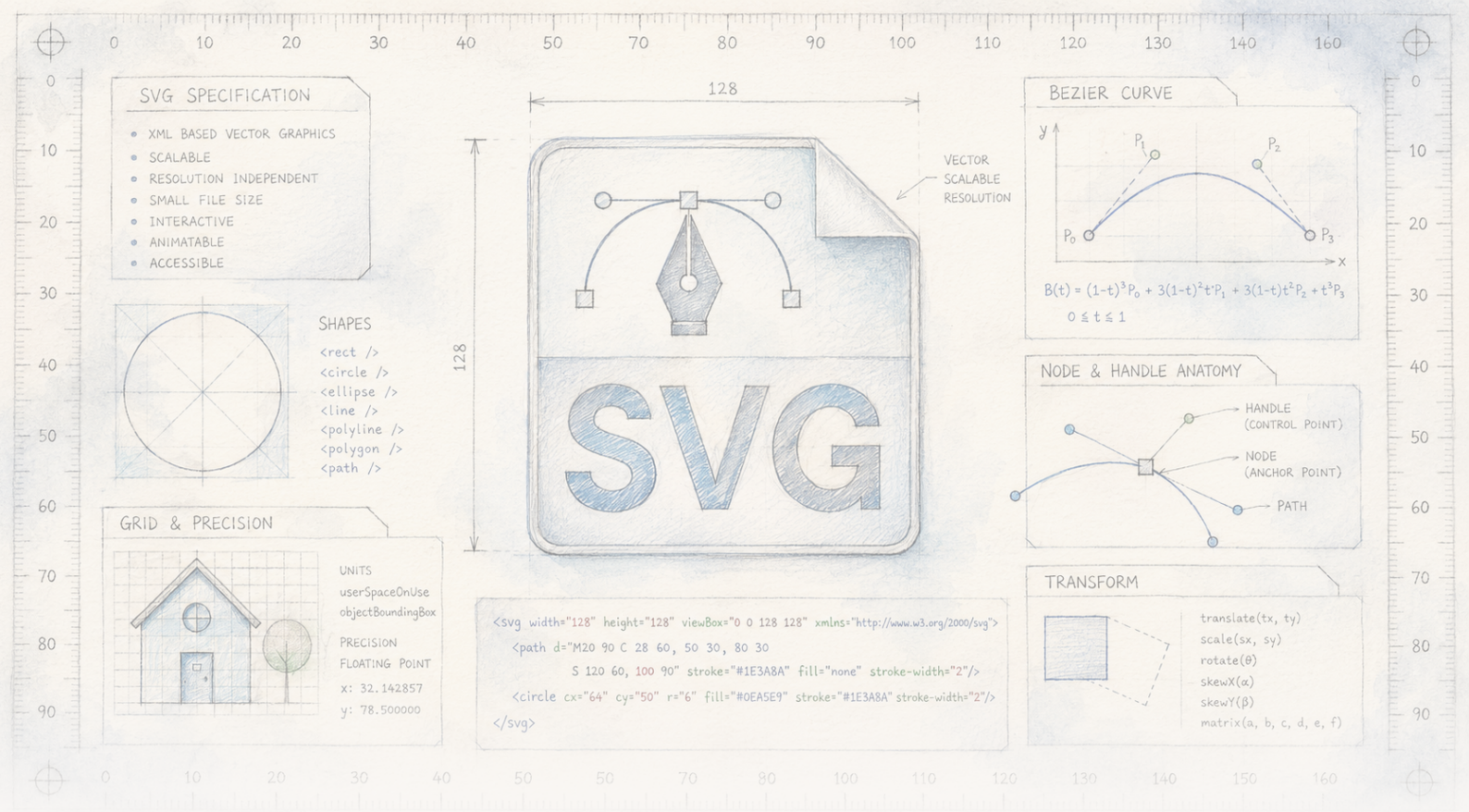


The Clean Paper

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

Kaip piešiantį DI išmokyti žiūrėti į savo drobę

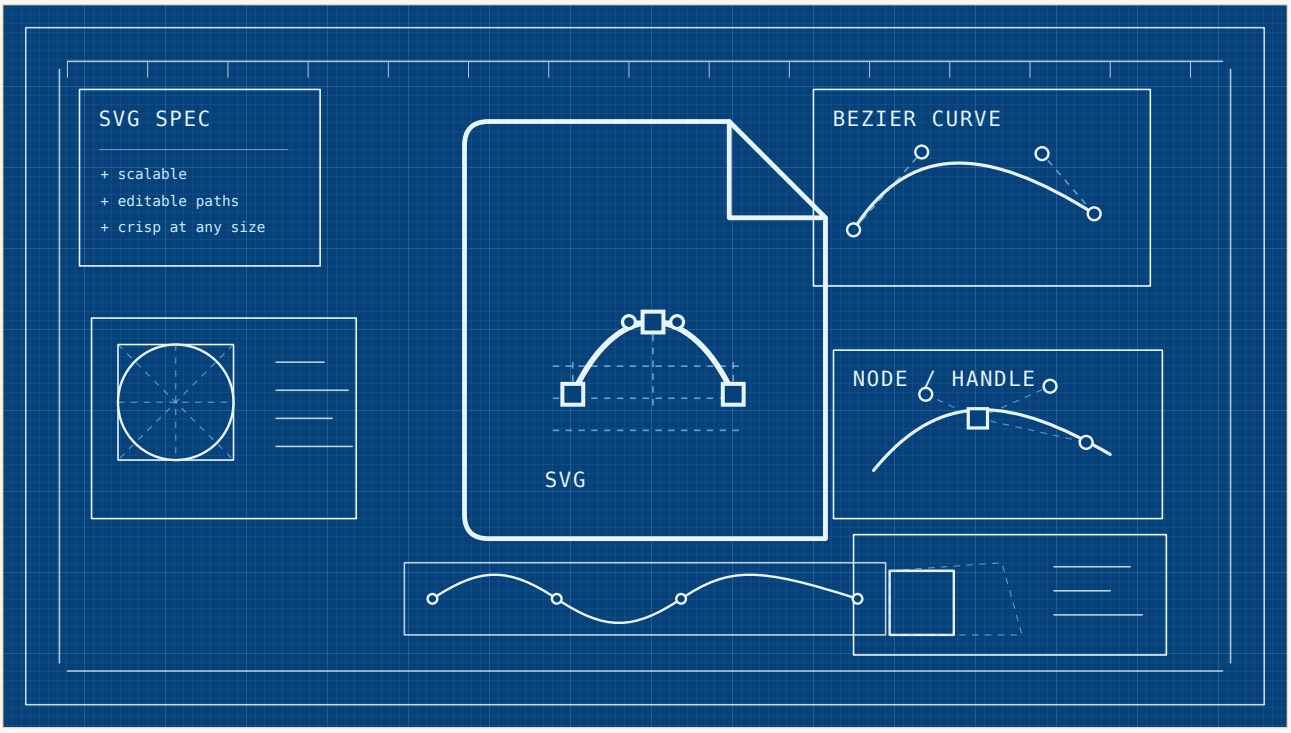
Kalbos modeliai, generuojantys vektorinę grafiką, dažnai dirba aklai – parašo visas piešimo komandas niekada nepamatę rezultato. Naujas metodas leidžia modeliui stebėti, kaip jo drobė pildosi žingsnis po žingsnio. Sąžiningas posūkis: vien suteikus „akis“ rezultatas blogėja – modelį reikia perapmokyti jomis naudotis. Tada jis viename etalone prilygsta ar truputį lenkia konkurentus, mokytus su iki dvidešimt kartų daugiau duomenų. Tai tikras ir kruopštus vieno etalono rezultatas, ne revoliucija.

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)

Piešti užsimerkus

Paprašius dabartinio DI modelio nupiešti paveikslą, viduje gali nutikti vienas iš dviejų labai skirtingų dalykų. Įprasti vaizdų generatoriai – tie, kurie iš sakinio sukuria fotografiją – tiesiogiai piešia pikselius ir darbo metu „mato“ drobę. Tačiau yra ir tylesnė piešimo rūšis, kai modelis visai netapo. Jis rašo instrukcijas: *čia nubrėžk apskritimą, ten liniją, šią figūrą užpildyk mėlynai*. Tos instrukcijos yra kodas – tas pats Scalable Vector Graphics, arba SVG, formatas, slypintis už daugelio žiniatinklio piktogramų ir logotipų. Jo pranašumas realus: gaunamas ne fiksuotas pikselių tinklelis, o figūrų rinkinys, kurį galima be galo didinti, perdažyti ir redaguoti neprarandant ryškumo.



Originalus SVG techninio brėžinio kūrinys: pats vaizdas yra redaguojamas vektorinis failas, sudarytas iš kreivių, tinklelio linijų, mazgų ir valdymo rankenėlių, o ne fiksuotų pikselių.

Laura Nesso / The Clean Paper Permission on file

Problema ta, kad iki šiol tokį piešimo kodą modeliai dažniausiai rašė aklai. Visa instrukcijų seka – apskritimas, linija, užpildymas – sugeneruojama vienu srautu niekada neperteikus tarpinių žingsnių į vaizdą. Įsivaizduokite piešiantį veidą žmogų užmerktomis akimis: pirmą akį gal dar pataikytumėte į vietą, bet nepastebėtumėte, kad antroji atsidūrė ant skruosto, o plaukų forma uždengė nosį. Maždaug taip dirbo šie modeliai, todėl jų kodas galėjo būti visiškai taisyklingas, o rezultatas vizualiai – chaotiškas.

Guotao Liang vadovaujama komanda padarė beveik komiškai akivaizdų dalyką: leido modeliui atsimerkti. Jų metodas *Render-in-the-Loop* po kiekvieno žingsnio atvaizduoja nebaigtą piešinį ir šį vaizdą grąžina modeliui prieš kitą potėpį. Tačiau įdomiausia ne tai, kad vaizdinis grįžtamasis ryšys gali padėti. Įdomiausia – ko reikėjo, kad jis išvis pradėtų padėti.

Kaip įvertinti piešinį?

Kai straipsnis sako, kad jo modelis „pranoksta“ kitą, verta paklausti: kuo tiksliai ir kaip tai išmatuota? Sugeneruotą vaizdą vertinti iš tiesų sunku – nėra vieno teisingo atsakymo į užduotį „nupiešk nešiojamąjį kompiuterį“. Todėl sritis remiasi keliomis automatinėmis metrikomis, kurios tėra netobuli žmogaus žvilgsnio pakaitalai.

Keletas jų naudojama ir šiame darbe. **FID** lygina visos sugeneruotų vaizdų grupės statistinį pasiskirstymą su realiais vaizdais; mažiau yra geriau, tačiau rodiklis apibūdina rinkinį, ne konkretų piešinį. **CLIP score** klausia kito DI, ar vaizdas atitinka užklausos tekstą – naudinga, bet tik tiek, kiek geras pats vertintojas. **DINO**, **SSIM** ir **LPIPS** lygina atkurtą vaizdą su taikiniu nuo tiesioginių pikselių iki išmoktų požymių lygmens.

Nė viena metrika nėra tiesa. Tai tarpiniai rodikliai, o jų skirtumai dažnai maži. Jei vienas modelis gauna 127,6, o kitas 128,8, tai tikras skirtumas numatyta kryptimi – bet pastūmėjimas, ne nuošliauža, ir dar skaičiuje, kuris tik apytikriai atitinka tai, ką pasakytų žmogaus akis. Tai verta prisiminti skaitant antraštę „pranoko modelį, mokytą su dvidešimt kartų daugiau duomenų“.

Ką padarė autoriai

Autoriai siekė išspręsti būtent aklo piešimo problemą. Ankstesni modeliai SVG generavimą traktuoja kaip grynai tekstinę užduotį: iš jau parašyto kodo numato kitą kodo dalį ir niekada jos neatvaizduoja. Taip nenaudojamos galingos „akys“ – vaizdo koduotuvas – kurias šiuolaikiniai multimodaliniai modeliai jau turi. *Render-in-the-Loop* užduotį pertvarko į žingsnis po žingsnio vykstantį vizualinį procesą. Po kiekvieno kodo fragmento dalinis SVG atvaizduojamas į paveikslą ir grąžinamas modeliui, todėl kitą fragmentą jis pasirenka jau matydamas dabartinę drobę.

Pirmasis rezultatas yra perspėjimas, ir autoriai jį pateikia tiesiai: paprasčiausiai pridėti šį ciklą prie jau egzistuojančio modelio neveikia. Kai stipriems bendrosios paskirties modeliams be papildomo mokymo buvo duodami tarpiniai atvaizdai, kokybė negerėjo – ji *visais atvejais blogėjo*. Modelis, kuris niekada nebuvo mokytas tokiu būdu naudotis savo vaizdo sistema, staiga pats to neišmoksta.

Todėl didžioji darbo dalis yra mokymas. Autoriai pertvarkė mokymo duomenis taip, kad kiekvienas piešinys būtų suskaidytas į daug mažų vizualiai prasmingų etapų – sudėtingos figūros padalytos į paprastesnes, kad kiekviename žingsnyje būtų kažkas naujo pamatyti – ir šiomis sekomis papildomai apmokė atvirą aštuonių milijardų parametrų modelį, paremtą Qwen3-VL. Šį procesą jie vadina Visual Self-Feedback. Generavimo metu pridėtas ir antras mechanizmas – Render-and-Verify: prieš priimdamas naują potėpį modelis jį atvaizduoja ir tikrina, ar šis iš tiesų pakeitė vaizdą, ar tik pakartojo ankstesnį. Nieko nepridedantys potėpiai atmetami, o kai daugiau niekas nepadeda, modelis skatinamas sustoti. Svarbu, kad visa sistema apmokyta su palyginti mažu duomenų rinkiniu – apie 850 000 pavyzdžių, mažiau nei puse vieno konkurento ir tik maža dalimi kito naudotų duomenų.

Ką jie nustatė

- Taip apmokytas modelis piešia geriau už savo „aklą“ variantą. Skirtumas ypač matomas nesėkmėse: aklas modelis gali nupiešti akį ant skruosto arba vietoj prašytos stulpelinės diagramos pateikti bendrą monitoriaus piktogramą.
- Standartiniame MMSVGBench etalone metodas konkuruoja su stipriais varžovais ir pagal kelis rodiklius juos truputį lenkia – tarp jų OmniSVG, mokytą su daugiau nei dvigubai daugiau duomenų, ir InternSVG, mokytą su maždaug dvidešimt kartų daugiau.
- Skirtumai maži. Pavyzdžiui, piktogramų rinkinyje pagrindinis vaizdo kokybės rodiklis yra 127,6, kai geriausias konkurentas turi 128,8, o užklausos atitikimo rodiklis – 0,293 prieš 0,291. Skirtumai nuoseklūs, bet ploni.
- Abu papildomi komponentai realiai prisideda: išjungus specialų mokymą arba generavimo metu atliekamą patikrą, rezultatai pastebimai blogėja. Patikros žingsnis ypač gerai apsaugo modelį nuo užstrigimo, kai tas pats elementas piešiamas vėl ir vėl.
- Patys autoriai labiausiai pabrėžia efektyvumą – panašų lygį pasiekti su gerokai mažiau mokymo duomenų nei pirmaujantys modeliai.

Ko tai neįrodo

- Tyrimas **neparodo**, kad suteikti modeliui „regėjimą“ savaime yra nemokama pergalė. Priešingai: be papildomo mokymo vaizdinis grįžtamasis ryšys rezultatą blogina. Nauda atsiranda dėl mokymo, ne vien dėl papildomos įvesties.
- Jis **neparodo didelės ar lemiamos persvaros**. Pagal daugumą rodiklių metodas eina beveik lygiai su konkurentais. Teiginys „pranoksta modelį, mokytą su 20× daugiau duomenų“ teisingas, bet skirtumai yra maži, matuojami tarpinėmis metrikomis viename etalone.
- Jis **neparodo bendrų meninių gebėjimų**. Tai aštuonių milijardų parametrų tyrimų modelis, piešiantis piktogramas ir paprastas iliustracijas maža fiksuota 224×224 pikselių raiška, o ne universalus dizaineris.
- Palyginimas su dideliu bendrosios paskirties modeliu GPT-5 **nėra lygiavertis**: GPT-5 nėra specialiai sukurtas ar derintas siaurai kodo piešimo užduočiai, todėl rezultatas mažai ką sako apie bendrą abiejų modelių pajėgumą.
- Metodas **nėra nemokamas skaičiavimo požiūriu**. Kiekviename žingsnyje atvaizduoti ir iš naujo perskaityti drobę yra lėčiau nei vienu srautu aklai sugeneruoti visą kodą; autoriai šią kainą pripažįsta.

Kiek tvirti yra įrodymai

- **Tvirti**, kai kalbama apie kontroliuojamą palyginimą jo paties sąlygomis. Ablacijos aiškios: pašalinus specialų mokymą arba verifikaciją, rodikliai krenta. Vadinasi, abu komponentai iš tiesų atlieka tą darbą, kurį jiems priskiria autoriai.

- **Sąžiningai pateikiama netikėta nesėkmė.** Faktas, kad naivus vaizdinis grįžtamasis ryšys *kenkia*, nėra paslėptas. Tai bene įdomiausia darbo dalis – naudinga korekcija intuicijai, kad daugiau įvesties visada reiškia geriau.
- **Plonesni**, kai kalbama apie reitingo persvarą. Pergalės prieš modelius, mokytus su daugiau duomenų, mažos ir parodytos viename etalone, kurį sukūrė vieno konkurento autoriai. Maži tarpinių metrikų skirtumai viename testų rinkinyje yra signalas, o ne galutinė išvada.
- **Neišbandyta mastu ir realiame dizaine.** Čia vertinamos piktogramos ir paprastos mažos raiškos iliustracijos. Ar ta pati idėja veiks sudėtingiems, aukštos raiškos ar realiems dizaino projektams, paliekama būsimam darbui.

Kodėl tai svarbu

Pagrindinė idėja beveik gėdingai paprasta ir siekia toliau nei piešimas. Jei programa kuria objektą rašydama kodą – tinklalapį, grafiką, diagramą ar 3D sceną – ji gali parašyti viską aklai ir tikėtis geriausio arba nuolat atvaizduoti rezultatą ir koreguoti kursą. Žmogui toks uždaras ciklas savaime suprantamas: į lapą žvilgčiojame nuolat. Modeliams tai stebėtinai naujas žingsnis.

Straipsnį vertą skaityti padaro žvaigždutė prie šios idėjos. Duoti modeliui galimybę matyti nėra tas pats, kas išmokyti jį žiūrėti. „Akis“ reikia treniruoti, ir tik tada ciklas duoda naudą. Net tada nauda reali, bet saikinga: stabilesnis braižytojas, ne naujos rūšies menininkas. Žmonėms, kuriantiems įrankius, kurie tekstinį aprašymą verčia redaguojama grafika – tokia, kokia vėliau tampa programėlių piktogramomis ar iliustracijomis – tylesnė praktinė pamoka yra naudingesnė už dar vieną reitingo tašką. Pergalė čia atėjo iš išmanesnio ir pigesnio mokymo, ne iš didesnio duomenų kalno.

Trumpa santrauka

Kalbos modeliai, generuojantys vektorinę grafiką – redaguojamą ir be kokybės praradimo keičiamo dydžio kodą, slypintį už daugelio žiniatinklio piktogramų ir logotipų – tradiciškai dirbo „aklai“: parašydavo visas piešimo instrukcijas niekada neatvaizdavę tarpinio rezultato. Guotao Liang vadovaujama komanda siūlo Render-in-the-Loop: po kiekvieno žingsnio atvaizduoti nebaigtą piešinį ir grąžinti jį modeliui, kad kitą potėpį jis pieštų matydamas drobę. Svarbiausias ir sąžiningiausias rezultatas – paprasčiausiai pridėjus tokį grįžtamąjį ryšį prie esamo modelio kokybė *blogėja*. Nauda atsiranda tik iš naujo apmokius modelį naudotis vaizdu ir pridėjus generavimo metu veikiančią patikrą, kuri atmeta nieko nekeičiančius potėpius. Taip apmokytas 8 mlrd. parametrų modelis viename standartiniame etalone prilygsta arba truputį lenkia konkurentus, mokytus su iki dvidešimt kartų daugiau duomenų. Tai tikras rezultatas, bet maži skirtumai tarpinių metrikų skalėje, piktogramoms ir paprastoms mažos raiškos iliustracijoms. Svarbiausia autorių išvada – efektyvumas: gerai išmokti žiūrėti į savo darbą gali būti naudingiau nei vien aklai didinti duomenų kiekį.

Be pagražinimų

Ką rodo tyrimas: perapmokius vektorinės grafikos modelį po kiekvieno žingsnio atvaizduoti ir perskaityti savo nebaigtą piešinį, rezultatai tampa geresni ir išsamesni nei piešiant aklai; viename standartiniame etalone metodas konkuruoja su modeliais, mokytais su gerokai daugiau duomenų, ir kai kur juos truputį lenkia.

Kas tikėtina, bet neįrodyta: kad „žiūrėti į savo darbą“ apskritai yra geresnė strategija nei didinti duomenų kiekį; kad tas pats metodas padės sudėtingai, aukštos raiškos ar realaus pasaulio grafikai; kad maži etalono skirtumai būtų aiškiai matomi žmogui.

Ko tyrimas nerodo: kad vaizdinis grįžtamasis ryšys savaime padeda – be perapmokymo jis kenkia; kad persvara prieš konkurentus didelė ar lemiama; bendrosios paskirties piešimo gebėjimų; sąžiningo tiesioginio palyginimo su bendrais modeliais, tokiais kaip GPT-5, kurie šiai užduočiai nekurti.

Pagrindiniai ribotumai: vienas etalonas, iš dalies sukurtas konkurento autorių; maži skirtumai tarpinių metrikų skalėje; 8B modelis, piktogramos ir paprastos iliustracijos 224×224 raiška; lėtesnis generavimas dėl atvaizdavimo kiekviename žingsnyje.

Kiek pasitikėti turėtų bendrasis skaitytojas? Tvirtai – kad uždaryti ciklą, atvaizduojant ir iš naujo „pažiūrint“ piešiant, iš tiesų padeda, bet šį gebėjimą reikia išmokyti, o ne tiesiog įjungti. Mažai arba vidutiniškai – kad šis konkretus modelis lemiamai pranoksta konkurentus; frazę „pranoko su 20× mažiau duomenų“ verta skaityti kaip tikrą, bet nedidelį vieno etalono rezultatą. Tvirtai – dėl vienos idėjos, kurią verta prisiminti: kodui, kuris piešia, geriau žiūrėti darbo eigoje nei piešti aklai.

Šaltiniai

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/