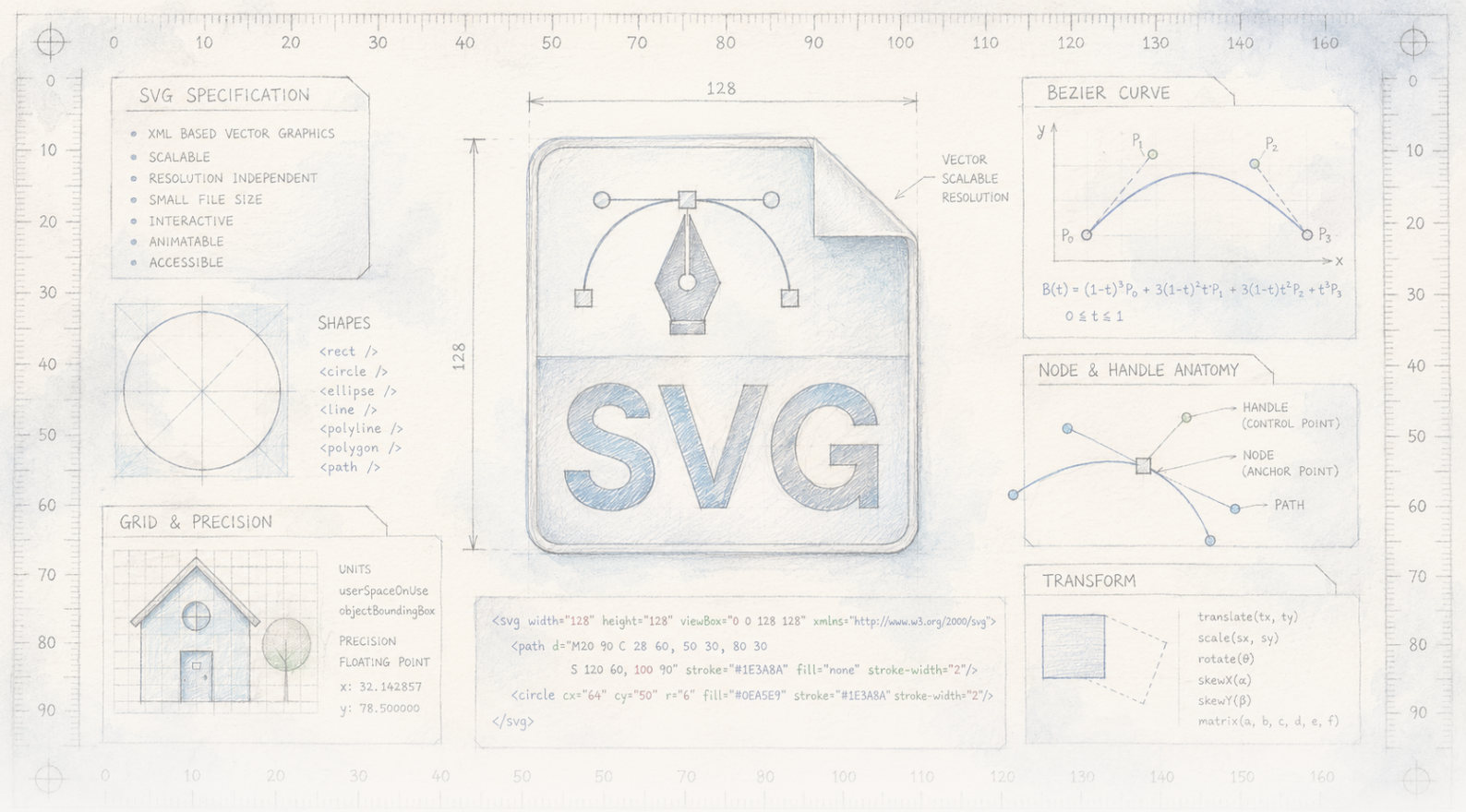


The Clean Paper

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

ללמד AI שמצייר להסתכל על הדף

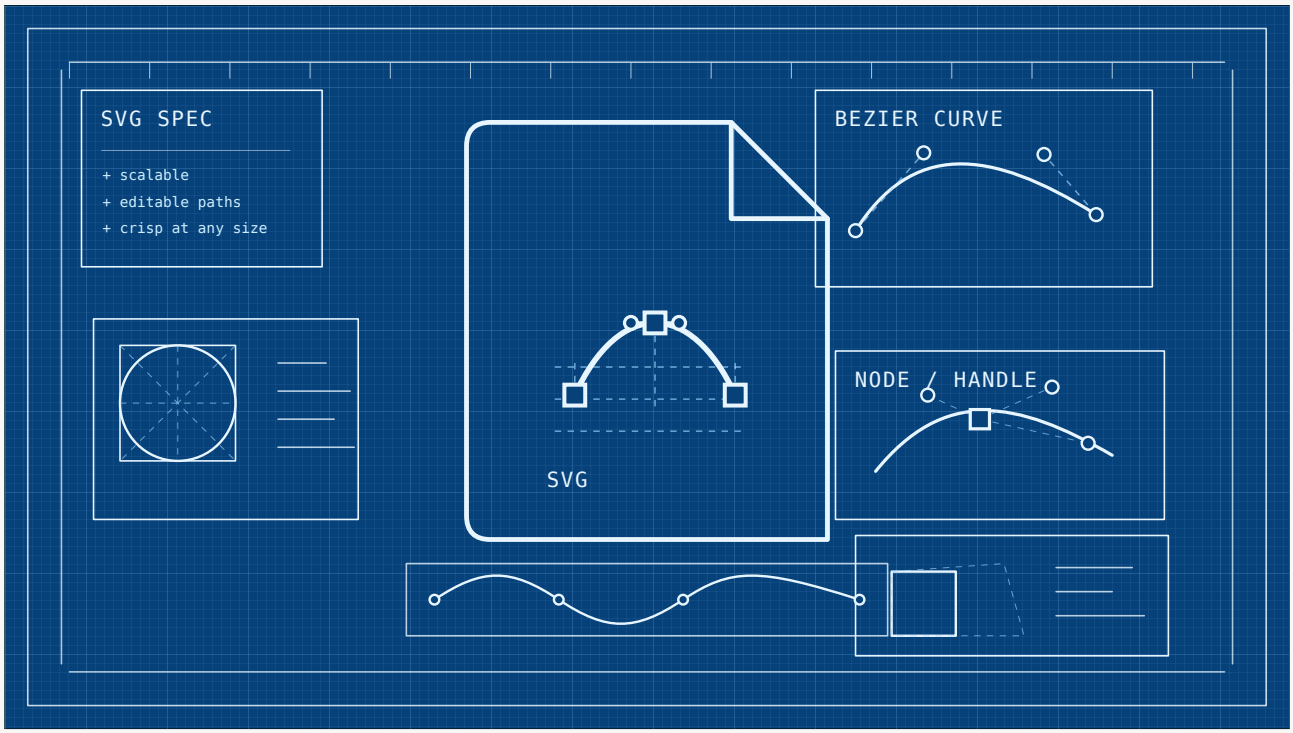
מודלי שפה שמייצרים גרפיקה וקטורית עושים זאת בעיוורון — כותבים את פקודות הציור בלי לראות את התוצאה. שיטה חדשה מאפשרת למודל לראות את הקנבס שלו מתמלא, *stroke* אחר *stroke*, והטוויסט הכנה הוא שפשוט לתת לו עיניים עושה את המצב גרוע יותר. צריך לאמן אותו מחדש להשתמש בהן. התמורה היא מודל שמשתווה או מעט עולה על יריבים שאומנו על עד פי עשרים יותר נתונים — תוצאה אמיתית וזהירה על *benchmark* אחד, לא מהפכה.

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)

לצייר בעיניים עצומות

אם תבקשו מאחד ממודלי ה-AI של היום לצייר תמונה, מאחורי הקלעים יקרה אחד משני דברים שונים מאוד. מחוללי התמונות המוכרים — אלה שמייצרים צילום ממשפט — צובעים פיקסלים ישירות, ויכולים לראות את הקנבס תוך כדי עבודה. אבל יש סוג שני ושקט יותר של ציור, שבו המודל כלל אינו "צובע". הוא כותב הוראות: צייר עיגול כאן, קו שם, מלא את הצורה הזאת בכחול. ההוראות האלה הן קוד — אותו Graphics, Vector Scalable או SVG, שנמצא מאחורי רוב האייקונים והלוגואים ברשת — ויש לכך יתרון אמיתי: התוצאה אינה רשת קבועה של פיקסלים אלא אוסף צורות שאפשר לשנות להן קנה מידה, צבע ועריכה בלי סוף ובלי טשטוש.



יצירת blueprint מקורית ב-SVG: התמונה עצמה היא קובץ וקטורי ניתן לעריכה, הבנוי מנתיבים, קווי רשת, nodes ו-handles ולא מפיקסלים קבועים.

Laura Nesso / The Clean Paper Permission on file

הבעיה היא שעד לאחרונה, מודל שכתב את קוד הציור עשה זאת בעיוורון. הוא פלט את כל רצף ההוראות במעבר אחד — עיגול, קו, מילוי — בלי לרנדר אותן כדי לראות מה יצר. דמיינו שאתם מציירים פנים בעיניים עצומות: אולי תצליחו לשים את העיניים בערך במקום הנכון, אבל אין לכם דרך לשים לב שהעין השנייה נחתה על הלחי, או שהצורה שציירתם לשיער יושבת עכשיו על האף. בערך כך המודלים האלה עבדו, ולכן הם הפיקו לעיתים קרובות קוד תקף לחלוטין שמבחינה חזותית היה בלגן.

צוות בראשות Liang Guotao עשה עכשיו את הדבר הכמעט קומית-מובן מאליו: הוא נתן למודל לפתוח את העיניים. השיטה שלהם, *Render-in-the-Loop*, מרנדרת את הציור החלקי אחרי כל שלב ומחזירה את התמונה למודל לפני שהוא מוסיף את המשיכה הבאה. החלק המעניין באמת אינו שזה עוזר — אלא מה היה צריך לעשות כדי שזה יעזור בכלל.

איך נותנים ציון לציור?

תמונה לשפוט מודדים? ואיך במה, מנצח לשאול: הוגן אחר, מודל "מנצח" שלו שהמודל אומר כשמאמר ציונים כמה על נשען התחום ולכן — נייד" מחשב ל"צייר נכונה יחידה תשובה אין — קשה באמת הוא שנוצרה התוצאה. על שמסתכל לאדם לא-מושלם תחליף הוא מהם אחד שכל אוטומטיים,
כמה מהם מופיעים במאמר הזה. FID משווה את האופי הסטטיסטי הכולל של אוסף תמונות שנוצרו לזה של תמונות אמיתיות; נמוך יותר טוב יותר, אבל הוא מתאר אוסף ולא תמונה בודדת. score CLIP שואל AI נפרד אם התמונה מתאימה למילות ה-prompt — שימושי, אך חד רק כמו השופט הזה. DINO, SSIM ו-LPIPS משווים תמונה משוחזרת ליעד, ברמות שנעות מפיקסלים גולמיים ועד תכונות שנלמדו.
אף אחד מהם אינו אמת. אלה ,proxies והם נוטים לזוז בצעדים קטנים. כשקוראים שמודל אחד מקבל 127.6 ואחר 128.8, זה הבדל אמיתי בכיוון הרצוי — אבל זו דחיפה קטנה, לא מפולת, ובמספר שעוקב רק באופן רופף אחרי מה שהעין שלכם הייתה אומרת. כדאי לזכור זאת כשהכותרת היא "מנצח מודל שאומן על פי עשרים יותר נתונים".

מה עשו המחברים

הבעיה שהמחברים ניסו לפתור היא הציור העיוור עצמו. מודלים קיימים מתייחסים לכתיבת SVG כאל משימת טקסט טהורה: לנבא את קטע הקוד הבא מתוך הקוד שנכתב עד כה, ולעולם לא לרנדר אותו. כך נשארות מובטלות "העיניים" החזקות — מקודד הראייה — שכבר קיימות במודלים מולטימודליים מודרניים. Render-in-the-Loop מארגנת את המשימה מחדש כתהליך חזותי של צעד-אחר-צעד. אחרי כל קטע קוד ציור, ה-SVG החלקי מרונדר לתמונה ומוחזר למודל כתמונה, כך שהקטע הבא נבחר בזמן שהמודל באמת מסתכל על הקנבס כפי שהוא נראה כרגע.

הממצא הראשון שלהם הוא אזהרה, ולזכותם הם מדווחים עליו ישירות: פשוט לחבר את הלולאה הזאת למודל מדף קיים לא עובד. כאשר הם הזינו renders ביניים למודלים כלליים חזקים בלי אימון מיוחד, האיכות לא השתפרה — היא הידרדרה בכל המדדים. מודל שמעולם לא לימדו אותו להשתמש בעיניים שלו למשימה הזאת לא יודע פתאום איך לעשות זאת.

לכן רוב העבודה נמצאת בלימוד. הם בנו מחדש את נתוני האימון כך שכל ציור מפורק להרבה צעדים קטנים ובעלי משמעות חזותית — מפצלים צורות מורכבות לחלקים פשוטים יותר, כדי שבכל שלב באמת יהיה משהו חדש לראות — ואז ביצעו fine-tuning למודל פתוח בעל שמונה מיליארד פרמטרים (המבוסס על Qwen3-VL) על הרצפים צעד-אחר-צעד. הם קוראים לכך Self-Feedback. Visual בזמן הציור הם מוסיפים מנגנון שני, :Render-and-Verify לפני קבלת כל stroke חדש, המודל מרנדר אותו ובודק אם הוא באמת שינה את התמונה או רק חזר על הקודם. strokes שלא מוסיפים דבר נזרקים, וכאשר שום דבר נוסף אינו עוזר, המודל מקבל הנחיה לעצור. ראוי לציון שכל זה פועל על מאגר קטן יחסית — כ-850,000 דוגמאות, פחות ממחצית ממה שמתחרה אחד השתמש בו וחלק קטן ממאגר של אחר.

מה הם מצאו

- לאחר אימון כזה, המודל מצייר טוב יותר מן המקבילה העיוורת שלו — במיוחד במקרי כישלון ברורים, שבהם מודל עיוור שם עין על לחי או מדלג על תרשים העמודות שהתבקש ומוציא מסך גנרי במקום.
- ב-benchmark הסטנדרטי ,MMSVGBench השיטה תחרותית מול יריבים חזקים ובכמה מדדים מעט לפניהם —

כולל OmniSVG, שאומן על יותר מפי שניים נתונים, ו-InternSVG, שאומן על בערך פי עשרים.
- הפערים קטנים. בסט האייקונים, למשל, ציון איכות התמונה הראשי הוא 127.6 לעומת 128.8 של היריב הטוב ביותר, וציון ההתאמה ל-prompt הוא 0.293 לעומת 0.291 — אמיתי ועקבי, אך צר.
- שני המרכיבים הנוספים מצדיקים את עצמם: מכבים את האימון המיוחד או את הבדיקה בזמן הציור והמספרים יורדים באופן מדיד. שלב האימות במיוחד מונע מן המודל להיתקע ולצייר שוב ושוב את אותו דבר.
- התוצאה שהמחברים עצמם מדגישים היא יעילות — להגיע עד כאן עם הרבה פחות נתוני אימון מאלה שהמובילים השתמשו בהם.

מה זה לא מוכיח

- הוא **לא** מראה שלתת למודל "לראות" הוא רווח חינם. להפך: הניסוי של המאמר עצמו מראה שמשוב חזותי בלי אימון מחדש הופך את התוצאה לגרועה יותר. השיפור מגיע מהאימון, לא מן העיניים לבדן.
- הוא **לא** מבסס יתרון גדול או מכריע. ברוב הציונים השיטה כמעט צמודה ליריבים; "מנצחת מודל שאומן על פי 20 יותר נתונים" הוא משפט נכון, אבל בהפרשים קטנים במדדי proxy ועל benchmark אחד.
- הוא **לא** מדגים יכולת אמנותית כללית. זהו מודל מחקר של שמונה מיליארד פרמטרים שמצייר אייקונים ואיורים פשוטים ברזולוציה קבועה קטנה (224×224 פיקסלים), לא מעצב כללי.
- ההשוואה מול מודל כללי גדול (GPT-5) **אינה apples-to-apples**: המודל הזה לא נבנה או כוונן למשימת ציור-בקוד הצרה הזאת, ולכן לנצח אותו כאן אומר מעט על שני המודלים באופן כללי.
- זה **לא** מגיע בחינם. רינדור וקריאה מחדש של הקנבס בכל צעד הופכים את היצירה לאיטית יותר מהפקת הקוד במעבר עיוור אחד — עלות שהמחברים מכירים בה.

עד כמה הראיות חזקות?

- **מוצקות** במקום שבו מדובר בהשוואה מבוקרת בתנאים של המאמר. ה-ablations נקיים: מסירים את האימון או את האימות והמספרים יורדים — לכן שני המרכיבים באמת עושים את העבודה שהמחברים מייחסים להם.
- **כנות לגבי ההפתעה של עצמו.** הממצא שמשוב חזותי נאיבי דווקא פוגע מדווח ולא נקבר, והוא אולי הדבר המעניין ביותר במאמר — תיקון שימושי לאינטואיציה שיותר קלט תמיד טוב יותר.
- **דקות** במקום שבו מדובר בטענת leaderboard. הניצחונות על יריבים עם יותר נתונים קטנים וחיים על benchmark יחיד שנבנה בידי מחברי אחד מאותם יריבים. פערים קטנים במדדי proxy על סט בדיקה אחד הם רמז, לא הכרעה.
- **לא נבדק בקנה מידה ובשטח.** הכול כאן הוא אייקונים ואיורים פשוטים ברזולוציה נמוכה. האם אותו רעיון יחזיק בגרפיקה מורכבת, ברזולוציה גבוהה או בעיצוב אמיתי נשאר לעבודה עתידית — והמחברים אומרים זאת.

למה זה חשוב

הרעיון במרכז המאמר כמעט מביך בפשטותו, והוא חורג הרבה מעבר לציור. אם תכנית עומדת לייצר משהו באמצעות כתיבת קוד — דף אינטרנט, תרשים, דיאגרמה, סצנה תלת-ממדית — היא יכולה לכתוב הכול בעיוורון ולקוות, או לרנדר תוך כדי ולתקן כיוון. סגירת הלולאה הזאת טבעית לאדם; אנחנו מציצים בדף כל הזמן. במודלים האלה זה מהלך חדש באופן מפתיע.

מה שהופך את המאמר לראוי לקריאה הוא הכוכבית שהוא מצמיד לרעיון. לתת למודל דרך לראות אינו אותו דבר כמו ללמד אותו להסתכל. צריך לאמן את העיניים, ורק אז הלולאה משתלמת — וגם אז התמורה אמיתית אך מדודה: שרטט יציב יותר, לא סוג אחר של אמן. לכל מי שבונה כלים שהופכים תיאור לגרפיקה ניתנת לעריכה — מהסוג שמגיע לאייקונים ולאיורים של אפליקציה — הלקח השקט והמעשי הוא השימושי. הניצחון כאן הגיע מאימון זול וחכם יותר, לא מעוד נתונים. זה דבר טוב יותר ללמוד מאשר עוד נקודה ב-leaderboard.

בקצרה

מודלי שפה שמייצרים גרפיקה וקטורית — הקוד הניתן לעריכה ולשינוי קנה מידה שמאחורי רוב האייקונים והלוגואים ברשת — עשו זאת באופן מסורתי "בעיוורון", כשהם כותבים את כל פקודות הציור בלי לראות את התוצאה. צוות בראשות Liang Guotao מציע Render-in-the-Loop: לרנדר את הציור החצי-גמור אחרי כל שלב ולהחזיר אותו למודל, כך שה-stroke הבא נבחר בזמן שהמודל מסתכל על הקנבס. הממצא המרכזי והכנה הוא שפשוט לעשות זאת למודל קיים הופך אותו לגרוע יותר; השיפור מופיע רק לאחר אימון מחדש לשימוש במשוב החזותי, בעזרת בדיקה בזמן הציור שזורקת strokes שלא משנים דבר. המודל המאומן מחדש, בעל שמונה מיליארד פרמטרים, משתווה או מעט עולה על יריבים שאומנו על עד פי עשרים יותר נתונים ב-benchmark סטנדרטי — תוצאה אמיתית, אך בפערים קטנים במדדי proxy, לאייקונים ואיורים פשוטים ברזולוציה נמוכה. המסר שהמחברים מדגישים, וזה שכדאי לשמור, הוא יעילות: לראות היטב את העבודה שלך יכול לפצות על גידול עיוור בכמות הנתונים.

בדיקה מפוכחת

מה המאמר מראה: אימון מחדש של מודל גרפיקה וקטורית לרנדר ולהסתכל על הציור החצי-גמור שלו, צעד אחר צעד, מייצר תוצאות טובות ושלמות יותר מציור עיוור — תחרותיות ואף מעט טובות יותר מיריבים עם יותר נתונים ב-benchmark סטנדרטי אחד, תוך שימוש בפחות נתוני אימון.

מה סביר אך לא הוכח: ש"לראות את העבודה שלך" הוא מתכון טוב באופן רחב יותר מהגדלת כמות הנתונים; שאותו רעיון יעזור בגרפיקה מורכבת, ברזולוציה גבוהה או בעבודה אמיתית; שהפערים הקטנים ב-benchmark משקפים הבדל שאדם באמת יבחין בו.

מה הוא לא מראה: שמשוב חזותי עוזר בפני עצמו (בלי אימון מחדש הוא פוגע); שהיתרון על יריבים גדול או מכריע; יכולת ציור כללית; השוואה הוגנת מול מודלים כלליים כמו GPT-5, שלא נבנו למשימה הזאת.

המגבלות העיקריות: benchmark אחד, שנבנה בחלקו בידי מחברי יריב; פערים קטנים במדדי proxy; מודל 8B, אייקונים ואיורים פשוטים ב-224×224; יצירה איטית יותר בגלל לולאת רינדור בכל צעד.

כמה ביטחון צריך להיות לקורא כללי? גבוה בכך שסגירת הלולאה — לרנדר ולהסתכל מחדש בזמן הציור — באמת עוזרת, ושצריך לאמן אותה ולא רק להפעיל. נמוך-בינוני בכך שהמודל המסוים הזה מנצח באופן מכריע את היריבים; יש לקרוא את "מנצח פי 20 נתונים" כתוצאה אמיתית אך צנועה מ-benchmark יחיד. גבוה ברעיון האחד שכדאי לזכור: בקוד שמצייר, להסתכל תוך כדי עדיף על לצייר בעיוורון.

מקורות

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

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

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