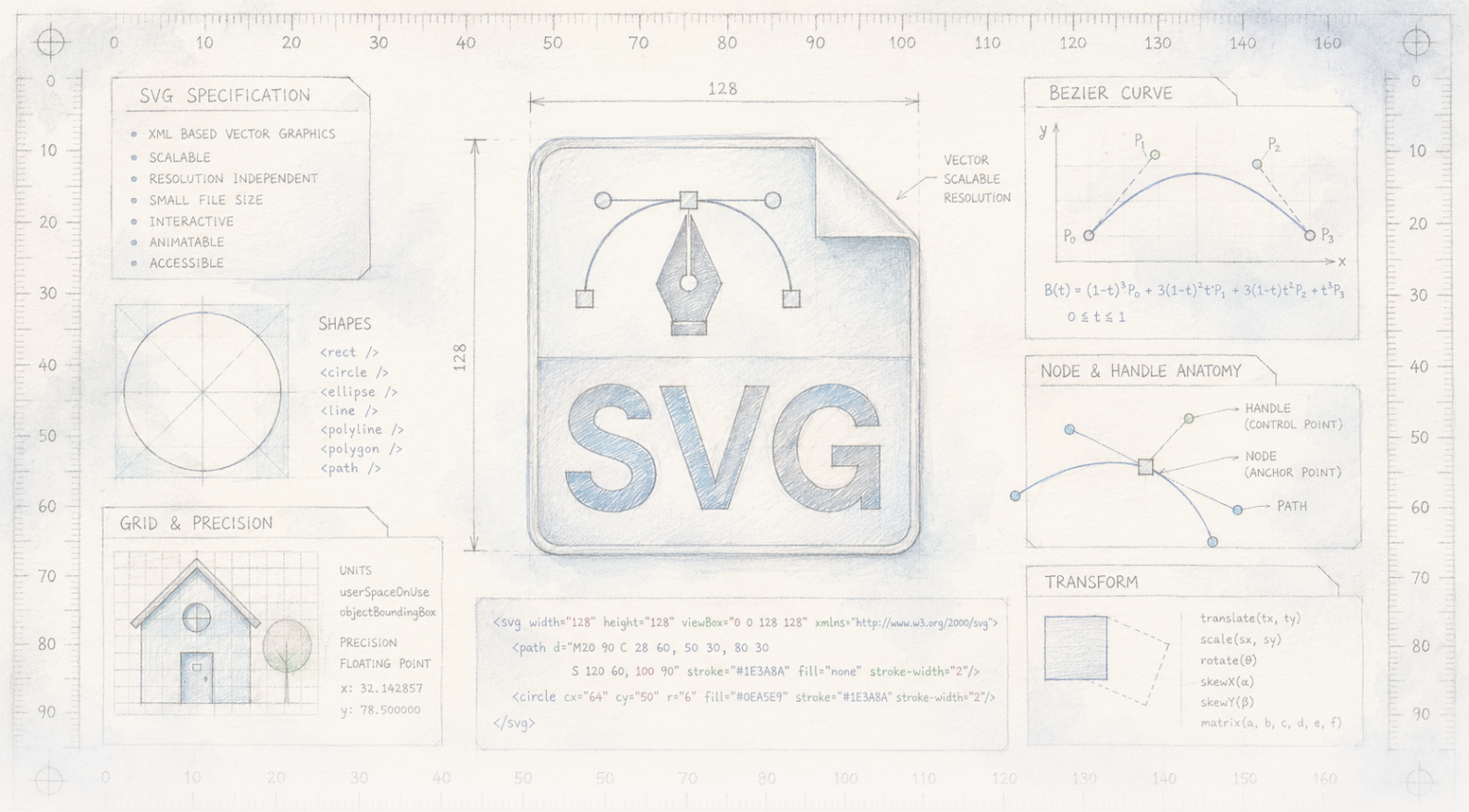


The Clean Paper

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

Einer zeichnenden KI beibringen, auf die Seite zu schauen

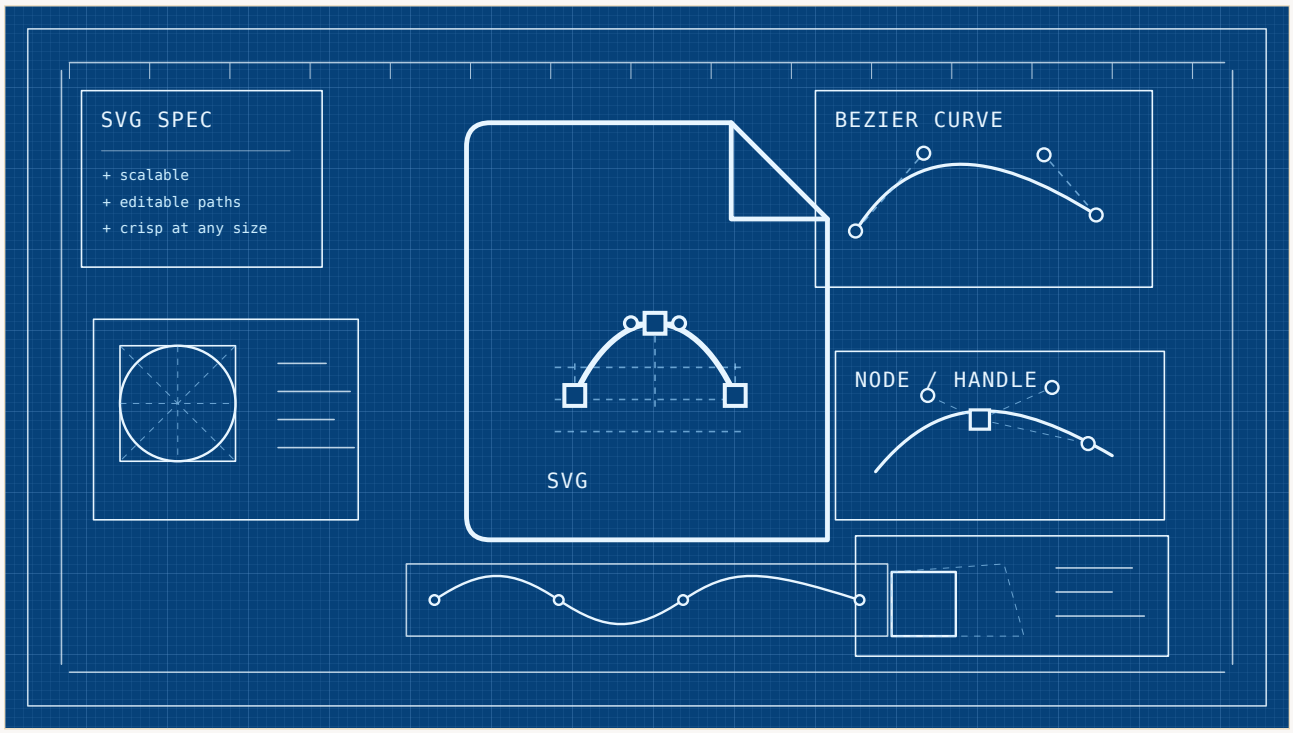
Sprachmodelle, die Vektorgrafiken erzeugen, tun es blind — sie schreiben die Zeichenbefehle aus, ohne das Ergebnis je zu sehen. Eine neue Methode lässt das Modell zusehen, wie sich seine eigene Leinwand füllt, Strich für Strich, und die ehrliche Pointe ist, dass es die Dinge verschlechtert, ihm bloß Augen zu geben: Es muss nachtrainiert werden, um sie zu benutzen. Der Lohn ist ein Modell, das Rivalen, die mit bis zu zwanzigmal mehr Daten trainiert wurden, erreicht oder knapp übertrifft — ein echtes, sorgfältiges Ergebnis auf einem Benchmark, keine Revolution.

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)

Zeichnen mit geschlossenen Augen

Bittet man eines der heutigen KI-Modelle, ein Bild zu zeichnen, passiert unter der Haube eines von zwei sehr unterschiedlichen Dingen. Die vertrauten Bildgeneratoren — jene, die aus einem Satz ein Foto zaubern — malen Pixel direkt, und sie können die Leinwand sehen, während sie arbeiten. Aber es gibt eine zweite, stillere Art des Zeichnens, bei der das Modell gar nicht malt. Es schreibt Anweisungen: *zeichne hier einen Kreis, dort eine Linie, fülle diese Form blau*. Diese Anweisungen sind Code — dieselben Scalable Vector Graphics, kurz SVG, die hinter den meisten Icons und Logos im Web stecken — und der Reiz ist real: Das Ergebnis ist kein festes Pixelraster, sondern ein Satz von Formen, die sich endlos skalieren, umfärben und bearbeiten lassen, ohne unscharf zu werden.



Originales SVG-Blaupausen-Kunstwerk: Das Bild ist selbst eine bearbeitbare Vektordatei, aufgebaut aus Pfaden, Rasterlinien, Knoten und Anfassern statt aus festen Pixeln.

Laura Nesso / The Clean Paper Permission on file

Der Haken ist, dass ein Modell, das diesen Zeichencode schreibt, es bis vor Kurzem blind tat. Es gab die gesamte Abfolge von Anweisungen in einem einzigen Durchgang aus — Kreis, Linie, Füllung —, ohne sie jemals zu rendern, um zu sehen, was es geschaffen hatte. Man stelle sich vor, mit geschlossenen Augen ein Gesicht zu skizzieren: Die Augen landen vielleicht ungefähr dort, wo Augen hingehören, aber man hätte keine Möglichkeit zu bemerken, dass das zweite auf der Wange gelandet ist oder dass die Form, die man für die Haare gezeichnet hat, nun über der Nase sitzt. So ungefähr arbeiteten diese Modelle, weshalb sie so oft Code produzierten, der vollkommen gültig und dennoch visuell ein Durcheinander war.

Ein Team um Guotao Liang hat nun das fast schon komisch Naheliegende getan: Es lässt das Modell die Augen öffnen. Ihre Methode, *Render-in-the-Loop*, rendert die halbfertige Zeichnung nach jedem

Schritt und reicht dieses Bild an das Modell zurück, bevor es den nächsten Strich zeichnet. Der wirklich interessante Teil ist nicht, dass das hilft — sondern was sie tun mussten, damit es überhaupt hilft.

Wie benotet man eine Zeichnung?

Wenn ein Paper behauptet, sein Modell "schlage" ein anderes, ist die Frage berechtigt: schlägt es worin, gemessen wie? Ein generiertes Bild zu beurteilen ist wirklich schwer — es gibt keine einzig richtige Antwort auf "zeichne einen Laptop" —, weshalb sich das Feld auf eine Handvoll automatischer Bewertungen stützt, jede ein unvollkommener Ersatz für einen Menschen, der sich das Ergebnis ansieht.
Einige davon tauchen in diesem Paper auf. **FID** vergleicht den statistischen Gesamtcharakter eines ganzen Stapels generierter Bilder mit echten; niedriger ist besser, aber es beschreibt den Stapel, nicht ein einzelnes Bild. **CLIP score** fragt eine separate KI, ob ein Bild zu den Worten des Prompts passt — nützlich, aber nur so scharf wie dieser Richter. **DINO**, **SSIM** und **LPIPS** vergleichen ein rekonstruiertes Bild mit einem Ziel, auf Ebenen von rohen Pixeln bis zu gelernten Merkmalen.
Keine dieser Zahlen ist die Wahrheit. Sie sind Näherungswerte, und sie bewegen sich meist in kleinen Schritten. Wenn man liest, dass ein Modell 127.6 erreicht, wo ein anderes 128.8 erreicht, ist das ein echter Unterschied in der beabsichtigten Richtung — aber ein Schubs, kein Erdrutsch, und noch dazu bei einer Zahl, die nur lose dem folgt, was das Auge sagen würde. Das sollte man im Hinterkopf behalten, wenn die Schlagzeile lautet: "schlägt ein Modell, das mit zwanzigmal so vielen Daten trainiert wurde."

Was die Autoren getan haben

Das Problem, das die Autoren beheben wollten, ist das blinde Zeichnen selbst. Bestehende Modelle behandeln das Schreiben von SVG als reine Textaufgabe: den nächsten Codeabschnitt aus dem bisherigen Code vorhersagen und ihn niemals rendern. Damit liegen die mächtigen "Augen" — der Vision-Encoder —, die moderne multimodale Modelle ohnehin mitbringen, brach. Render-in-the-Loop strukturiert die Aufgabe als schrittweisen visuellen Prozess um. Nach jedem Fragment des Zeichencodes wird das partielle SVG zu einem Bild gerendert und dem Modell als Bild zurückgereicht, sodass das nächste Fragment gewählt wird, während das Modell tatsächlich auf die bisherige Leinwand blickt.

Ihr erster Befund ist eine Warnung, und es spricht für sie, dass sie ihn offen berichten: Diese Schleife einfach an ein bestehendes Modell von der Stange anzubauen, funktioniert nicht. Als sie starken Allzweckmodellen ohne jedes spezielle Training Zwischenrenderings zuführten, verbesserte sich die Qualität nicht — sie *verschlechterte* sich auf ganzer Linie. Ein Modell, dem nie beigebracht wurde, seine Augen dafür zu benutzen, kann es nicht plötzlich.

Der Großteil der Arbeit steckt also im Beibringen. Sie bauen die Trainingsdaten so um, dass jede Zeichnung in viele kleine, visuell bedeutsame Schritte zerlegt wird — komplexe Formen werden in einfachere Teile aufgespalten, damit es in jeder Phase etwas Neues zu sehen gibt —, und feintunen

dann ein offenes Modell mit acht Milliarden Parametern (aufgebaut auf Qwen3-VL) auf diesen Schritt-für-Schritt-Sequenzen. Sie nennen das Visual Self-Feedback. Beim Zeichnen selbst fügen sie einen zweiten Mechanismus hinzu, Render-and-Verify: Bevor jeder neue Strich akzeptiert wird, rendert das Modell ihn und prüft, ob er das Bild tatsächlich verändert oder bloß den letzten wiederholt hat. Striche, die nichts beitragen, werden verworfen, und wenn nichts mehr hilft, wird das Modell zum Aufhören angehalten. Bemerkenswerterweise läuft all das mit einem ziemlich kleinen Datensatz — rund 850.000 Beispiele, weniger als die Hälfte dessen, was ein Konkurrent nutzte, und ein kleiner Bruchteil dessen, was ein anderer verwendete.

Was sie herausfanden

- So trainiert, zeichnet das Modell besser als sein blindes Gegenstück — am sichtbarsten bei den Fehlerfällen, in denen ein blindes Modell ein Auge auf eine Wange setzt oder ein angefordertes Balkendiagramm überspringt und stattdessen einen generischen Monitor ausgibt.
- Auf dem Standard-Benchmark (MMSVGBench) erweist sich die Methode als konkurrenzfähig mit starken Rivalen — und liegt in mehreren Messwerten leicht vorn —, darunter OmniSVG, trainiert mit mehr als doppelt so vielen Daten, und InternSVG, trainiert mit ungefähr zwanzigmal so vielen.
- Die Abstände sind klein. Auf dem Icon-Set etwa liegt sein wichtigster Bildqualitätswert bei 127.6 gegenüber 128.8 des besten Rivalen und sein Prompt-Übereinstimmungswert bei 0.293 gegenüber 0.291 — real und konsistent, aber knapp.
- Beide hinzugefügten Bausteine verdienen ihren Platz: Schaltet man das spezielle Training oder die Verifikation beim Zeichnen ab, sinken die Zahlen messbar. Insbesondere der Verifikationsschritt verhindert, dass das Modell hängen bleibt und immer wieder dasselbe zeichnet.
- Das Ergebnis, das die Autoren selbst betonen, ist Effizienz — so weit zu kommen mit weit weniger Trainingsdaten, als die Spitzenreiter nutzten.

Was das nicht beweist

- Es zeigt **nicht**, dass es ein Gratisgewinn ist, ein Modell "sehen" zu lassen. Im Gegenteil: Das eigene Experiment des Papers zeigt, dass visuelles Feedback *ohne* Nachtraining die Dinge verschlechtert. Der Gewinn kommt vom Nachtraining, nicht von den Augen allein.
- Es belegt **keinen** großen oder entscheidenden Vorsprung. In den meisten Werten liegt die Methode Kopf an Kopf mit ihren Rivalen; "schlägt ein Modell, das mit 20× so vielen Daten trainiert wurde" stimmt, aber nur um Schubser auf Näherungsmetriken, auf einem Benchmark.
- Es demonstriert **keine** allgemeine künstlerische Fähigkeit. Dies ist ein Forschungsmodell mit acht Milliarden Parametern, das Icons und einfache Illustrationen bei einer kleinen festen Auflösung (224×224 Pixel) zeichnet, kein Allzweck-Designer.
- Der Vergleich mit einem großen Allzweckmodell (GPT-5) ist **kein** Vergleich von Gleichem mit Gleichem: Jenes Modell ist für diese enge Code-Zeichenaufgabe weder gebaut noch abgestimmt, sodass ein Sieg hier wenig über eines der beiden Modelle im Allgemeinen aussagt.

- Es kommt **nicht** umsonst. Die Leinwand bei jedem Schritt zu rendern und neu einzulesen macht die Generierung langsamer, als den Code in einem blinden Durchgang auszugeben — ein Preis, den die Autoren einräumen.

Wie belastbar sind die Belege?

- **Solide**, wo es ein kontrollierter Vergleich zu eigenen Bedingungen ist. Die Ablationen sind sauber: Entfernt man das Training oder die Verifikation, fallen die Zahlen — die beiden Zutaten leisten also wirklich die Arbeit, die die Autoren behaupten.
- **Ehrlich gegenüber der eigenen Überraschung.** Der Befund, dass naives visuelles Feedback *schadet*, wird berichtet, nicht vergraben, und er ist das Interessanteste an dem Paper — eine nützliche Korrektur der Intuition, dass mehr Input immer besser sei.
- **Dünn, wo es eine Bestenlisten-Behauptung ist.** Die Siege über Rivalen mit mehr Daten sind klein und leben auf einem einzigen Benchmark, der von den Autoren eines dieser Rivalen gebaut wurde. Kleine Abstände auf Näherungsmetriken auf einem Testset sind ein Hinweis, keine Gewissheit.
- **Ungetestet im großen Maßstab und in freier Wildbahn.** Alles hier sind Icons und einfache Illustrationen bei niedriger Auflösung. Ob dieselbe Idee für komplexe, hochauflösende oder reale Designarbeit trägt, bleibt künftiger Arbeit überlassen — das sagen die Autoren selbst.

Warum es wichtig ist

Die Idee im Zentrum dieses Papers ist fast schon peinlich einfach, und sie reicht weit über das Zeichnen hinaus. Wenn ein Programm etwas erzeugen soll, indem es Code schreibt — eine Webseite, ein Diagramm, ein Schaubild, eine 3D-Szene —, kann es entweder das Ganze blind schreiben und hoffen, oder unterwegs rendern und den Kurs korrigieren. Diese Schleife zu schließen ist für einen Menschen selbstverständlich; wir werfen ständig einen Blick auf die Seite. Für diese Modelle ist es ein überraschend junger Schritt.

Was das Paper lesenswert macht, ist das Sternchen, das es anfügt. Einem Modell eine Möglichkeit zum Sehen zu geben ist nicht dasselbe, wie ihm das Hinsehen beizubringen. Die Augen müssen trainiert werden, und erst dann zahlt sich die Schleife aus — und selbst dann ist der Ertrag real, aber maßvoll: ein ruhigerer Zeichner, kein anderer Typ von Künstler. Für alle, die Werkzeuge bauen, die eine Beschreibung in eine bearbeitbare Grafik verwandeln — die Art von Dingen, die am Ende hinter den Icons und Illustrationen einer App stecken —, ist die stille, praktische Lehre die nützliche. Der Gewinn kam hier von billigerem, klügerem Training, nicht von mehr Daten. Das ist eine bessere Erkenntnis als ein weiterer Punkt auf einer Bestenliste.

Kurz zusammengefasst

Sprachmodelle, die Vektorgrafiken erzeugen — den bearbeitbaren, skalierbaren Code hinter den meisten Web-Icons und Logos —, haben das traditionell "blind" getan: Sie schrieben alle Zeichenbefehle aus, ohne sie jemals zu rendern und das Ergebnis zu sehen. Ein Team um Guotao Liang schlägt Render-in-the-Loop vor: Die halbfertige Zeichnung wird nach jedem Schritt gerendert und zurückgereicht, sodass das Modell den nächsten Strich zeichnet, während es auf die Leinwand blickt. Ihr zentraler, ehrlicher Befund ist, dass ein bestehendes Modell dadurch allein *schlechter* wird; der Gewinn stellt sich erst ein, nachdem das Modell darauf nachtrainiert wurde, das visuelle Feedback zu nutzen, unterstützt von einer Prüfung beim Zeichnen, die Striche verwirft, die nichts verändern. Das nachtrainierte Modell mit acht Milliarden Parametern erreicht auf einem Standard-Benchmark das Niveau von Rivalen, die mit bis zu zwanzigmal mehr Daten trainiert wurden, oder schlägt sie leicht — ein echtes Ergebnis, aber mit kleinen Abständen auf Näherungswerten, für Icons und einfache Illustrationen bei niedriger Auflösung. Die Lehre, die die Autoren betonen und die es zu behalten lohnt, betrifft Effizienz: Die eigene Arbeit gut zu sehen kann schiere Datenmenge ersetzen.

Nüchtern geprüft

Was das Paper zeigt: Ein Vektorgrafik-Modell darauf nachzutrainieren, seine eigene halbfertige Zeichnung Schritt für Schritt zu rendern und anzusehen, liefert bessere und vollständigere Ergebnisse als blindes Zeichnen — konkurrenzfähig mit Rivalen mit mehr Daten und auf einem Standard-Benchmark leicht vor ihnen, aus weniger Trainingsdaten.

Was plausibel, aber nicht bewiesen ist: Dass "die eigene Arbeit sehen" generell ein besseres Rezept ist als das Skalieren von Daten; dass dieselbe Idee bei komplexen, hochauflösenden oder realen Grafiken hilft; dass die kleinen Benchmark-Abstände einen Unterschied widerspiegeln, den ein Mensch tatsächlich bemerken würde.

Was es nicht zeigt: Dass visuelles Feedback für sich allein hilft (ohne Nachtraining schadet es); dass der Vorsprung vor den Rivalen groß oder entscheidend ist; allgemeine Zeichenfähigkeit; einen fairen direkten Vergleich mit Allzweckmodellen wie GPT-5, die für diese Aufgabe nicht gebaut sind.

Wesentliche Einschränkungen: Ein Benchmark, teilweise von den Autoren eines Rivalen gebaut; kleine Abstände auf Näherungsmetriken; ein 8B-Modell, Icons und einfache Illustrationen bei 224×224; langsamere Generierung wegen der Render-bei-jedem-Schritt-Schleife.

Wie viel Vertrauen sollte eine allgemeine Leserschaft haben? Hoch, dass das Schließen der Schleife — rendern und neu einlesen beim Zeichnen — wirklich hilft und dass es antrainiert werden muss, nicht bloß eingeschaltet. Niedrig bis mäßig, dass dieses spezielle Modell seine Rivalen entscheidend schlägt; die Zeile "schlägt 20× so viele Daten" sollte man als reales, aber bescheidenes Ergebnis auf einem einzigen Benchmark nehmen. Hoch bei der einen Idee, die es sich zu merken lohnt: Für Code, der zeichnet, schlägt Hinsehen beim Zeichnen das blinde Zeichnen.

Quellen

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/