# Beyond NL2Code: A Systematic Survey of Multimodal Code Intelligence

**Anonymous authors**
**Paper under double-blind review**

## Abstract

While Large Language Models (LLMs) have revolutionized text-to-code synthesis, conventional text-centric paradigms fail to capture the dense spatial hierarchies and structural constraints inherent in real-world visual contexts, such as user interfaces and scientific plots. To bridge this gap, Multimodal Code Intelligence has emerged as a pivotal domain, empowering Vision-Language Models (VLMs) to translate visual perception into precise executable code. This paper presents a structured taxonomy of this rapidly evolving landscape, systematically categorizing the literature into four foundational domains: Graphical User Interfaces, Scientific Visualization, Structured Graphics, and Frontiers Frameworks. Within this framework, we systematically analyze tasks ranging from mainstream web and chart synthesis to complex emerging scenarios, such as programmatic visual manipulation and code-to-video generation. Through rigorous analysis of existing benchmarks and methodologies, we identify four pivotal technical shifts that may shape future research: the transition from imitation-based training to reward-driven optimization, the progression from static synthesis toward dynamic interaction, the evolution toward unified, general-purpose models, and the evolution from chat-based systems into autonomous agents. We envision this systematic survey as a foundational guide to accelerate future advancements in multimodal code intelligence. The trajectory of this field is rapidly shifting from merely extracting basic functional logic to synthesizing high-fidelity, aesthetically refined, and dynamically interactive outputs through iterative refinement. Ultimately, we posit that code constitutes the universal action space for multimodal general intelligence. By empowering AI systems to seamlessly translate complex visual intent into executable logic and autonomously navigate digital environments, visually-grounded code generation marks a definitive breakthrough toward autonomous software agents. An ongoing, dynamically updated project and resources associated with this survey have been released at anonymous repo.

# Contents

# 1 Introduction

As the crystallization of human wisdom, code serves as a fundamental interface between human abstract conceptualization and real-world computational realization, translating high-level human intent into executable instructions (Sun et al., 2024). While automated program synthesis has long been a cornerstone of software engineering research, the field is currently undergoing a paradigm shift fueled by the advent of Large Language Models (LLMs) (Chen et al., 2021; Li et al., 2023; Rozière et al., 2024). Recent released state-of-the-art (SOTA) models, such as Gemini-3-Pro (Google, 2025), Claude-4.5-Sonnet (Anthropic, 2025), and Kimi K2.5 (Kimi et al., 2026), have demonstrated remarkable capabilities in generating complex code from natural language instructions. This paradigm of synthesizing executable programs directly from textual specifications is known as Natural Language-to-Code (NL2Code) generation (Zan et al., 2023; Jiang et al., 2024; Zhu et al., 2024b; Yang et al., 2025d), which requires a sophisticated alignment between high-level linguistic intent and formal program syntax.

Driven by the expanding capabilities of LLMs, the scope of NL2Code tasks has undergone a profound evolution, transcending simple syntax translation to encompass complex, end-to-end software engineering. Traditional approaches primarily focused on function-level synthesis, where models synthesize standalone code snippets based on atomic specifications (Austin et al., 2021; Li et al., 2022b; Jain et al., 2024). However, the current frontier has shifted toward repository-level engineering, which necessitates that models navigate intricate cross-file dependencies and maintain project-wide coherence (Liu et al., 2023c; Zhang et al., 2023). Furthermore, the field now extends into system-level programming and autonomous issue resolution, where models operate as autonomous engineers capable of debugging, evolving, and patching real-world software vulnerabilities (Jimenez et al., 2023; Yang et al., 2024c; Zhang et al., 2024b). Beyond its role in traditional software engineering, the inherent executable nature of code has established it as a robust and versatile interface for performing real-world tasks within digital environments. Unlike the inherent ambiguity of natural language, generated code serves as a deterministic medium to invoke external tools, query structured databases, and orchestrate complex agentic workflows. By leveraging code to interact with software APIs and digital systems, LLMs can effectively ground their actions in functional, real-world scenarios (Schick et al., 2023; Gao et al., 2023; Wang et al., 2024d). Consequently, the advancement of these code-driven capabilities constitutes a major milestone toward realizing the practical utility of LLMs in solving intricate, open-ended tasks.

Despite these advances, most NL2Code approaches rely solely on textual descriptions. In practice, visual signals serve as a high-bandwidth and intuitive medium for communication. Unlike sequential text, a single image can efficiently encode dense spatial hierarchies and complex structural information that are challenging to articulate verbally. This modality gap becomes especially critical in visual-centric domains such as frontend development (Si et al., 2025; Laurençon et al., 2024), data visualization (Yang et al., 2024a; Zhao et al., 2025d), and computer-aided design (Wu et al., 2021; 2025c), where the generated code yields fundamentally visual outputs. In these scenarios, relying solely on text to describe intricate user interface layouts or precise geometric structures is both inefficient and prone to information loss, often leading to a misalignment between human intent and the resulting code. To bridge this gap, the recent advent of Multimodal Large Language Models (MLLMs) integrates visual perception with logical reasoning (Liu et al., 2023b; Bai et al., 2025a; Shen et al., 2025). Spurred by the need to address these real-world bottlenecks, the field of Multimodal Code Intelligence has emerged. This approach enables models to understand visual

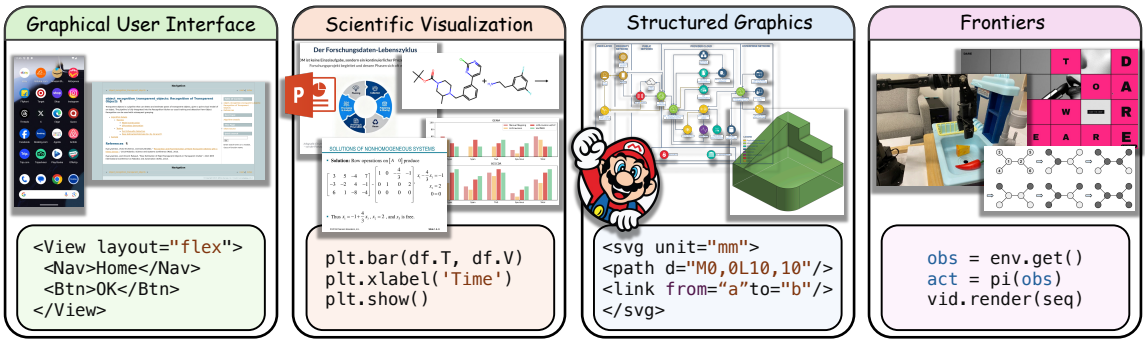


Figure 1: Overview of the Multimodal Code Intelligence landscape. The field is categorized into four primary domains: (1) Graphical User Interface, transforming visual UI designs into frontend code (e.g., React/HTML); (2) Scientific Visualization, converting charts and scientific documents into plotting scripts (e.g., Matplotlib); (3) Structured Graphics, representing vector graphics and diagrams as structured code (e.g., SVG, CAD); and (4) Frontiers, encompassing emerging applications such as vision-based programming, embodied control and video generation logic.

inputs directly, treating visual perception not as an auxiliary feature, but as a core prerequisite for automating visually-driven programming tasks.

In this survey, we provide a comprehensive overview of recent advancements in Multimodal Code Intelligence. We first establish a formal problem formulation for various multimodal code generation tasks. Then, to categorize the rapidly expanding body of literature, we organize existing research into four primary domains: (1) Section 3 reviews Graphical User Interfaces (GUIs), encompassing the generation of web and mobile applications; (2) Section 4 delves into Scientific Visualization, ranging from statistical charts and structured documents to academic presentations; (3) Section 5 focuses on Structured Graphics, covering Scalable Vector Graphics (SVG), diagrams, and Computer-Aided Design (CAD); and (4) Section 6 explores emerging Frontiers Frameworks, such as programmatic visual manipulation, video generation, and unified multimodal models. For each domain, we systematically analyze the landscape of benchmarks and methodologies, as structured in Figure 2 and 3. Looking forward, Section 7 delineates four pivotal technical paradigm shifts poised to redefine the field. These transformations encompass the transition from static code synthesis to dynamic, execution-based interaction and the paradigm evolution from Supervised Fine-Tuning (SFT) to Reinforcement Learning (RL). These advancements ultimately point toward unified model architectures and the realization of agentic workflows capable of powering multimodal code intelligence.

## 2 Task Formulation

In this section, we provide a formal taxonomy for Multimodal Code Intelligence. We systematically define the core tasks by categorizing them into visual-to-code synthesis and code-centric reasoning paradigms.

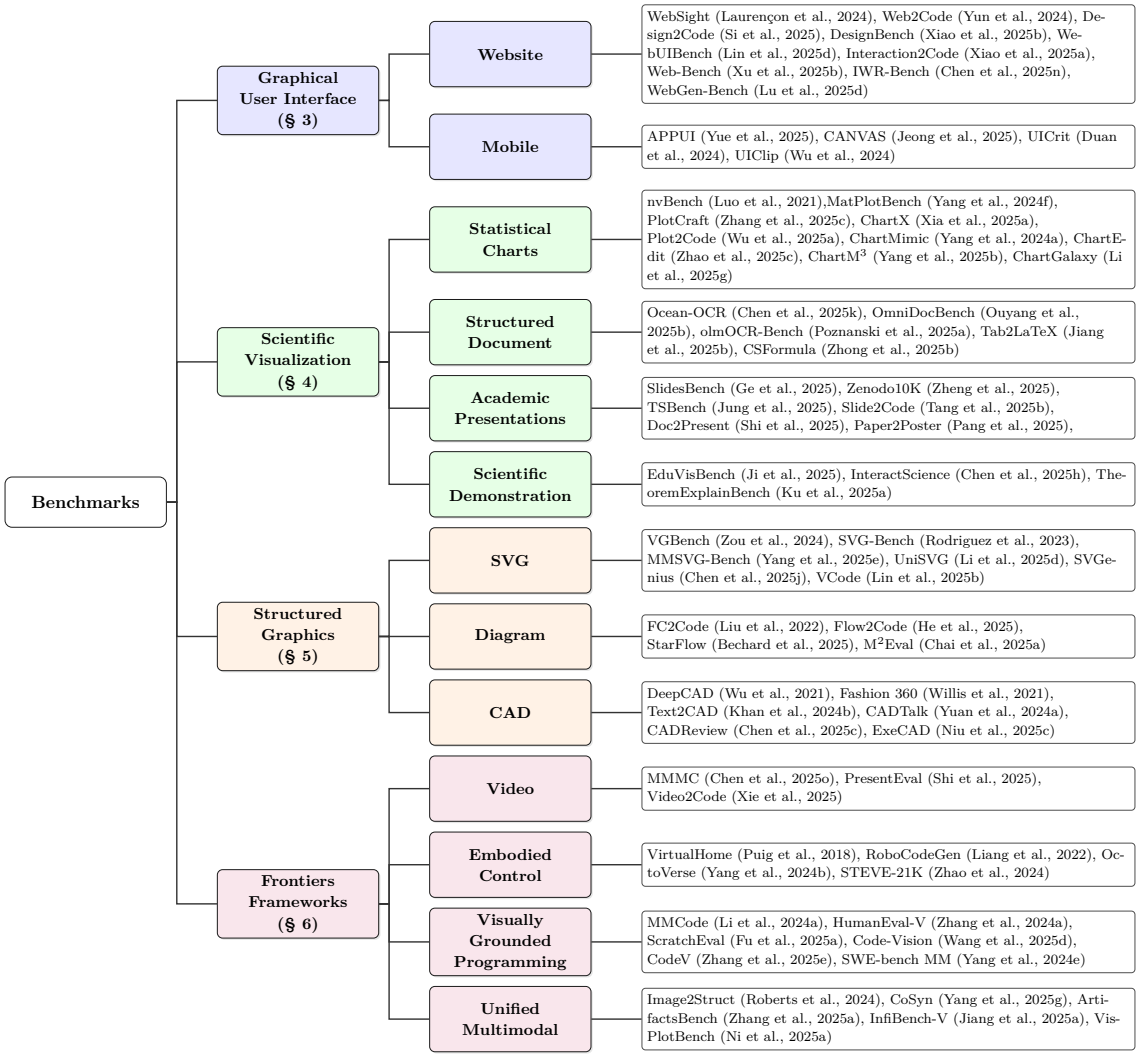


Figure 2: Taxonomy of benchmarks for multimodal code generation. We categorize existing datasets into four primary domains: Graphical User Interface (§ 3), Scientific Visualization (§ 4), Structured Graphics (§ 5), and Frontiers Frameworks (§ 6). Each domain is further divided into specific subdomains, with leaf nodes listing representative benchmarks and their citations.

### 2.1 NL2Code Preliminaries

To establish a formal baseline for multimodal extensions, the conventional NL2Code paradigm aims to synthesize an executable program $\mathcal{C}$ given a natural language description $\mathcal{T}$. Formally, this task is modeled as a mapping function

$$\mathcal{C} = \mathrm{LLM}(\mathcal{T}). \tag{1}$$

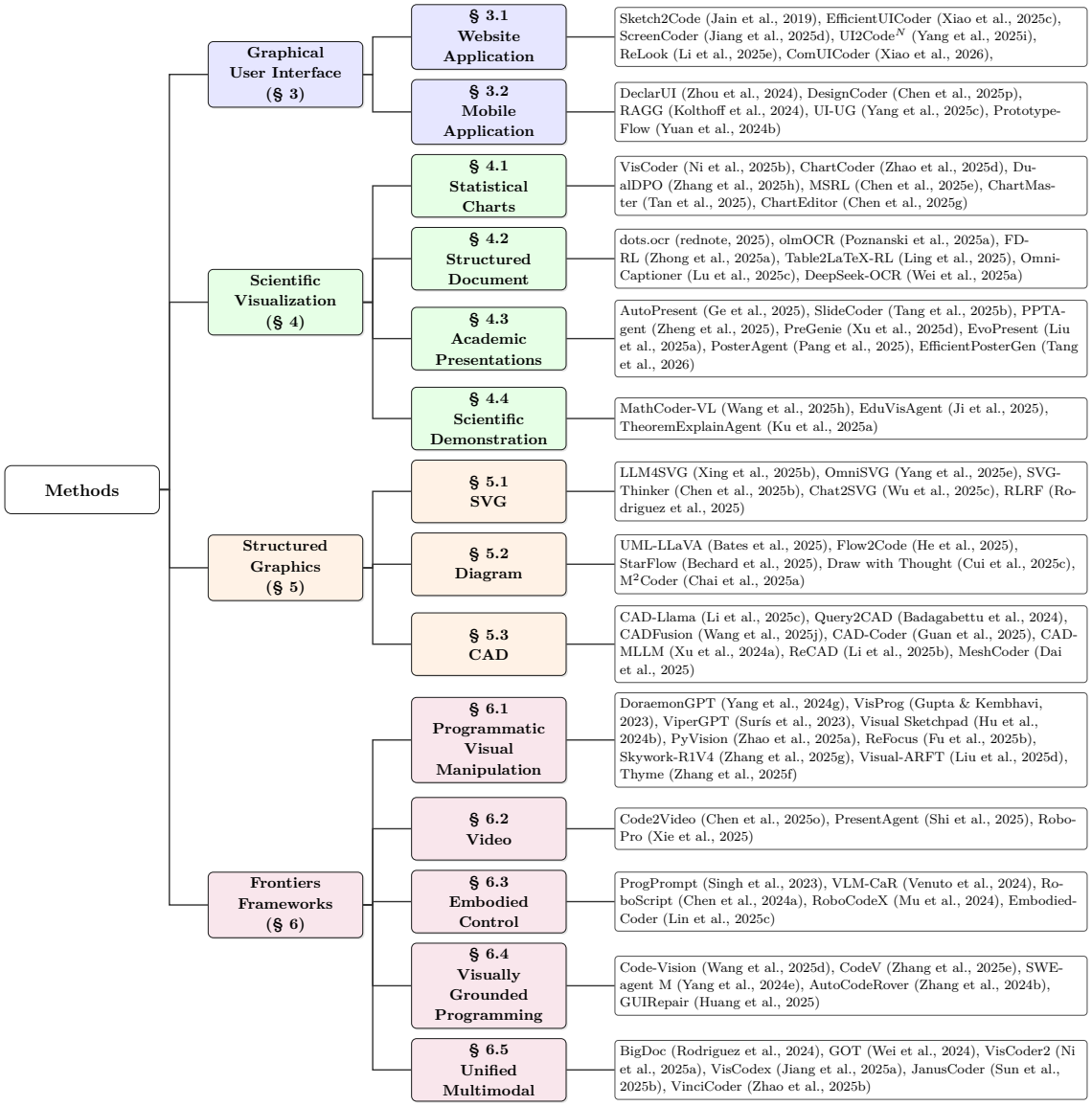


Figure 3: Taxonomy of multimodal code generation methods. The classification structure mirrors the benchmark taxonomy in Figure 2, spanning GUI (§ 3), Scientific Visualization (§ 4), Structured Graphics (§ 5), and Frontiers Frameworks (§ 6). For each subdomain, we showcase representative models and methodologies to illustrate the taxonomy.

While effective for logic-centric tasks, this unimodal formulation lacks the capacity to perceive spatial requirements, which are often essential in scenarios where intent is intrinsically tied to visual information.

### 2.2 Multimodal Code Synthesis

We define Multimodal Code Synthesis as the process of generating or modifying code where the visual context $\mathcal{I}$ is a primary input. Depending on the initial state and the underlying manipulation intent, we delineate three sub-tasks:

**Multimodal Direct Generation.** In this paradigm, the model is provided with a visual context $\mathcal{I}$ (e.g., a chart or GUI screenshot) alongside a textual prompt $\mathcal{T}_{\text{desc}}$. The objective of *Direct Generation* is to task an MLLM with synthesizing code that faithfully reconstructs the visual content of $\mathcal{I}$. The generation process is formulated as:

$$\mathcal{C}_{\text{gen}} = \text{MLLM}(\mathcal{I}, \mathcal{T}_{\text{desc}}) \tag{2}$$

where $\mathcal{T}_{\text{desc}}$ instructs the model to generate the underlying source code for the provided image. This formulation is conceptually analogous to image-to-text translation or captioning, as it focuses primarily on visual reproduction rather than handling complex user constraints or logical modifications.

**Instruction-driven Code Editing.** To further expand the task scope and leverage the instruction-following capabilities of MLLMs, recent works have explored multimodal code editing. This task requires the model to manipulate visual content based on specific user instructions. Current editing paradigms generally fall into two categories: (1) Text-guided editing, where modification intents are conveyed solely through natural language; and (2) Visual-prompt editing, where textual requirements are combined with visual prompts $\mathcal{V}$ (e.g., bounding boxes or encircling target regions) to precisely localize target elements. Formally, given an initial image $\mathcal{I}$, a textual editing instruction $\mathcal{T}_{\text{edit}}$, and an optional visual prompt $\mathcal{V}$ (where $\mathcal{V} = \emptyset$ in text-guided settings), the model must generate the target edited code $\mathcal{C}_{\text{edited}}$ that renders the desired visual state. Crucially, in this setting, the model operates without access to the source code of $\mathcal{I}$. It must implicitly infer the program logic directly from pixels and synthesize new code from scratch. The editing process is formulated as:

$$\mathcal{C}_{\text{edited}} = \text{MLLM}(\mathcal{I}, \mathcal{T}_{\text{edit}}, \mathcal{V}) \tag{3}$$

This task evaluates the model's capacity for visual reasoning, precise spatial grounding, and counterfactual code synthesis without structural guidance.

**Reference-based Code Refinement.** While editing focuses on manipulating content based on external instructions, multimodal code refinement focuses on error correction and quality improvement. Drawing upon advancements in multi-turn debugging, this task provides the model with an explicit starting point: a potentially flawed code draft $\mathcal{C}_{\text{draft}}$. The goal is to generate a refined version $\mathcal{C}_{\text{refined}}$ that aligns the draft with the visual reference $\mathcal{I}$ or satisfies specific constraints $\mathcal{T}_{\text{refine}}$. The formulation is:

$$\mathcal{C}_{\text{refined}} = \text{MLLM}(\mathcal{I}, \mathcal{T}_{\text{refine}}, \mathcal{C}_{\text{draft}}) \tag{4}$$

In contrast to the source-code-agnostic nature of editing, refinement allows the model to leverage $\mathcal{C}$draft as a structural prior, focusing its computational capacity on precise alignment and functional debugging.

### 2.3 Code-Centric Reasoning and Acting

Transcending the scope of visual synthesis, recent research has increasingly exploited executable code as a robust substrate for advanced reasoning and agentic interaction. In this subsection, we formalize the domain of code-aided reasoning, where code functions not as a visual end-product, but as a symbolic intermediary that bridges visual perception with logical deduction and environmental control. Specifically, we delineate this domain into two primary paradigms: Programmatic Tool-Use for complex reasoning, and Executable Policy for autonomous agents.

**Programmatic Tool-Use.** In this paradigm, code functions as an intermediate reasoning trace. Rather than performing end-to-end neural prediction, the model acts as a neuro-symbolic controller that decomposes a complex visual query $\mathcal{Q}$ into a modular program $\mathcal{C}_{\text{tool}}$. This program invokes external perceptual primitives (e.g., object detectors, OCR APIs) or performs precise symbolic calculations. Depending on the execution workflow, we categorize these methods into two distinct mechanisms:

- **Direct Programmatic Solving:** This approach adopts a deterministic framework where the task is resolved entirely via execution. The MLLM serves as a semantic parser to translate the natural language query into executable logic, and the execution result is treated as the final answer $\mathcal{A}$:

$$\mathcal{C}_{\text{tool}} = \text{MLLM}(\mathcal{I}, \mathcal{Q}), \quad \mathcal{A} = \text{Execute}(\mathcal{C}_{\text{tool}}, \mathcal{I}) \tag{5}$$

- **Tool-Augmented Visual Reasoning:** In this setting, the program acts as a perception enhancer to manipulate the visual input. The execution yields a processed view $\mathcal{I}' = \text{Execute}(\mathcal{C}_{\text{tool}}, \mathcal{I})$ (e.g., cropping or edge detection), which serves as an augmented context for subsequent inference:

$$\mathcal{A} = \text{MLLM}(\text{Execute}(\mathcal{C}_{\text{tool}}, \mathcal{I}), \mathcal{Q}) \tag{6}$$

By differentiating these pathways, the framework accommodates both rigorous symbolic derivation and flexible, perception-aware reasoning.

**Executable Policy.** In the realm of sequential decision-making, code functions as a high-level policy $\pi$ that maps visual observations to structured actions. This paradigm applies to a wide spectrum of interactive environments, ranging from embodied robotics to digital GUI navigation. Unlike static text generation, the model synthesizes an executable action script $\mathcal{C}_{\text{policy}}$ based on the current state observation $\mathcal{O}_t$ and a high-level goal $\mathcal{G}$. This code-based policy utilizes control loops and API calls to enable temporally extended behaviors. The process is formulated as:

$$\mathcal{O}_{t+1} = \text{Env}(\pi(\mathcal{O}_t, \mathcal{G}), \mathcal{O}_t) \tag{7}$$

where the code-based policy $\pi(\mathcal{O}_t, \mathcal{G})$ interacts with the environment to drive the transition to the next state $\mathcal{O}_{t+1}$. In this context, code empowers the agent with a structured and generalizable action space, prioritizing functional correctness and goal attainment over mere visual fidelity.

# 3 Graphical User Interface

## 3.1 Website Application

Bridging the semantic gap between high-level visual designs and their programmatic implementations, website-to-code (Web-to-Code) generation automates the labor-intensive translation of visual interfaces into standardized web technologies such as HTML, CSS, and JavaScript. In this paradigm, MLLMs leverage visual reasoning to extract structural and stylistic attributes from raw pixel inputs, thereby achieving faithful reconstruction (Luera et al., 2024). We begin by reviewing established evaluation benchmarks, ranging from static structural alignment to dynamic execution-based protocols. Subsequently, we trace the methodological evolution from SFT to advanced agentic workflows and RL advancements. The standard web-to-code workflow for this process is illustrated in Figure 4, with key benchmarks summarized in Table 1.

### 3.1.1 Website Code Generation Benchmarks

Leveraging the structural transparency of HTML source code, researchers have established robust evaluation frameworks to evaluate model performance. These benchmarks are principally categorized into two paradigms: static visual fidelity assessment and dynamic functional viability evaluation.

Static benchmarks primarily evaluate the structural and visual alignment between generated webpages and ground-truth designs. Pioneering efforts such as WebSight (Laurençon et al., 2024) and Web2Code (Yun et al., 2024) exemplify the early reliance on large-scale synthetic datasets to establish foundational capabilities. To bridge the gap toward real-world complexity, Design2Code (Si et al., 2025) introduces a benchmark comprising 484 manually curated samples from Common Crawl, while WebCode2M (Gui et al., 2025) leverages massive real-world websites and HTML to significantly scale training and evaluation. To achieve more granular structural analysis, Vision2UI (Gui et al., 2024) assesses the integrity of HTML DOM trees, whereas IW-Bench (Guo et al., 2025a) focuses on element-level layout accuracy. Furthermore, WebRenderBench (Lai et al., 2025) and WebGen-V Bench (Wang et al., 2025i) emphasize pixel-level visual fidelity by comparing rendered artifacts with reference images. Expanding beyond mere replication, DesignBench (Xiao et al., 2025b) proposes a holistic framework covering generation, editing, and repair tasks. Similarly, WebUIBench (Lin et al., 2025d) and FullFront (Sun et al., 2025a) incorporate assessments of aesthetic quality and perceptual comprehension.

Complementing static assessments, dynamic evaluation protocols verify whether the generated code correctly implements the intended website functionality and interactivity. Interaction2Code (Xiao et al., 2025a) pioneers the interactive web synthesis task, systematically investigating the capability of MLLMs to prototype reactive interfaces. To address architectural complexity, MRWeb (Wan et al., 2024) evaluates the synthesis of multi-page, resource-aware applications featuring internal navigation and backend routing. For complex engineering scenarios, Web-Bench (Xu et al., 2025b) designs a suite of tasks with sequential dependencies to simulate real-world development lifecycles . Recent advancements also incorporate multimodal prompts and agentic evaluation. For instance, IWR-Bench (Chen et al., 2025n) targets interactive reconstruction from video inputs, whereas WebGen-Bench (Lu et al., 2025d) leverages a GUI agent (Chen et al., 2025a) to autonomously simulate user interactions, verifying functionality through rigorous execution-based test cases.

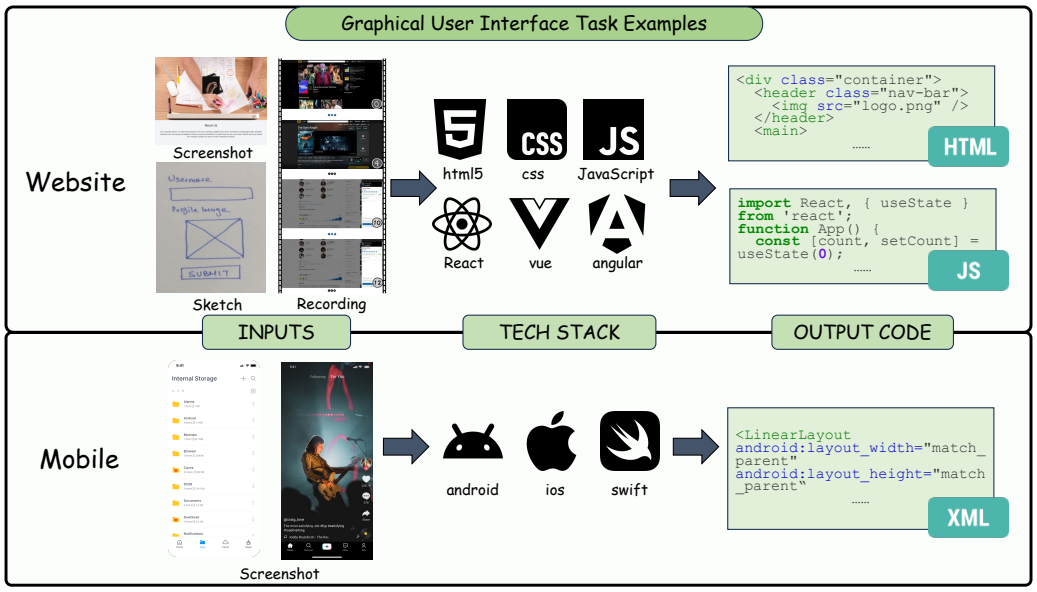


Figure 4: Examples of GUI code generation tasks for website and mobile applications.

### 3.1.2 Website Code Generation Methods

The methodological landscape of Web-to-Code has evolved from monolithic generation to sophisticated, interactive systems. Recent advancements can be broadly categorized into three evolutionary streams: SFT-based paradigms, collaborative agent-based workflows, and iterative refinement via visual feedback coupled with RL.

Early approaches, such as Sketch2Code (Jain et al., 2019), relied on hand-crafted object detection models and UI parsers to translate sketches into intermediate representations. With the advent of VLMs, the field has pivoted toward direct SFT. Pioneering efforts, such as WebSight (Laurençon et al., 2024) and Web2Code (Yun et al., 2024), establish foundational baselines by training on large-scale synthetic datasets. Design2Code (Si et al., 2025) and WebCode2M (Gui et al., 2025) further scale this approach by incorporating diverse real-world web data to enhance model robustness. To improve the computational efficiency of these SFT-based models, EfficientUICoder (Xiao et al., 2025c) introduces a bidirectional token compression framework, significantly reducing inference overhead. However, despite these advances, single-model architectures often struggle to generalize in complex, open-ended scenarios.

To address these limitations, the research frontier also pivots toward collaborative agent-based frameworks that decompose the generation process via a divide-and-conquer strategy. ScreenCoder (Jiang et al., 2025d) proposes a modular architecture comprising grounding, planning, and generation units to enhance task interpretability. Similarly, Frontend Diffusion (Ding et al., 2025) decouples the workflow into distinct design, coding, and review stages, thereby improving system modularity. For comprehensive application development, TDDev (Wan et al., 2025) introduces a multi-agent system based on Test-Driven Development (TDD), enabling the dynamic generation of full-stack web applications. In the domain of data synthesis, Instruct4Edit (Dang et al., 2025) employs LLMs to programmatically synthesize high-quality datasets specifically tailored for code

| Benchmark | Year | Data Source | Test Instances | Evaluation Metrics |
|---|---|---|---|---|
| *Static Benchmark* | | | | |
| WebSight (Laurençon et al., 2024) | 2024 | Synthetic | 823k | BLEU, TreeBLEU, SSIM |
| Web2Code (Yun et al., 2024) | 2024 | Synthetic+Real | 884.7k | Visual similarity, Code match |
| Design2Code (Si et al., 2025) | 2024 | Real | 484 | CLIP, Block match, Visual sim |
| WebCode2M (Gui et al., 2025) | 2024 | Real | 20k | TreeBLEU, SSIM, Error rate |
| Vision2UI (Gui et al., 2024) | 2024 | Real | 20k | TreeBLEU, SSIM |
| IW-Bench (Guo et al., 2025a) | 2024 | Real+Synthetic | 1.2k | Element/Layout Accuracy |
| FullFront (Sun et al., 2025a) | 2025 | Real+Synthetic | 400 | Gemini Visual Score, Code Score |
| WebRenderBench (Lai et al., 2025) | 2024 | Real | 45.1k | RDA, GDA, SDA |
| WebGen-V Bench (Wang et al., 2025i) | 2024 | Real | 647 | Layout/Style consistency |
| DesignBench (Xiao et al., 2025b) | 2025 | Real | 900 | Generation/Editing/Repair |
| *Dynamic Benchmark* | | | | |
| Interaction2Code (Xiao et al., 2025a) | 2024 | Real | 97 | Interaction success rate |
| MRWeb (Wan et al., 2024) | 2024 | Real | 500 | Functionality completeness |
| Web-Bench (Xu et al., 2025b) | 2025 | Real | 50 projects | Pass@k |
| WebGen-Bench (Lu et al., 2025d) | 2024 | Real | 101 | Accuracy, Appearance score |
| IWR-Bench (Chen et al., 2025n) | 2025 | Real | 113 | Interaction fidelity |
| *Specialized Benchmark* | | | | |
| UIClip (Wu et al., 2024) | 2024 | Synthetic+Real | 2.3M+1.2k | Design quality score |
| AUI-Gym (Lin et al., 2025a) | 2024 | Synthetic | 52 apps | Function Completeness, CUA SR |
| WebVIA (Xu et al., 2025c) | 2025 | Synthetic+Real | 11k | Task completion rate |

Table 1: Representative benchmarks in the UI-to-Code domain. Static benchmarks focus on visual similarity between generated and reference UIs, while dynamic benchmarks evaluate functional correctness through automated interaction testing.

editing tasks. ComUICoder (Xiao et al., 2026) applies semantic-aware segmentation and merging techniques for component-based UI code generation, improving code reusability and maintainability.

Recognizing the constraints of single-turn, open-loop generation, contemporary works increasingly integrate iterative visual feedback and RL to ensure functional and visual alignment. WebGen-Agent (Lu et al., 2025d) pioneers the incorporation of multi-level visual feedback, establishing a closed-loop cycle of code generation, execution, and optimization. UI2CodeN (Yang et al., 2025i) unifies generation, editing, and polishing capabilities, exploring test-time scaling strategies to leverage interactive feedback systematically. In the realm of RL optimization, ReLook (Li et al., 2025e) introduces a visual-driven framework, employing a VLM as a critic to orchestrate a diagnosis-and-optimization loop. Complementing this, WebRenderBench (Lai et al., 2025) proposes an Automated Layout and Style Inspection Agent (ALISA) strategy and formulates fine-grained reward functions based on geometric and stylistic alignment via WebDriver rendering. Furthermore, Coder-CUA (Lin et al., 2025a) advocates for a paradigm shift from human-centric to agent-centric evaluation. By leveraging a code agent to initialize and refine UIs and a Computer-Use Agent (CUA) to assess them via navigation success and task solvability, this method redefines the interface development workflow.

### 3.2 Mobile Application

As a pivotal frontier of UI-to-Code, Mobile-to-Code generation targets the automated synthesis of visual user interfaces into production-ready mobile implementations. This field has emerged as a

significant research focal point, aiming to bridge the gap between high-fidelity design mockups and functional front-end code through intelligent synthesis.

#### 3.2.1 Mobile Code Generation Benchmarks

To rigorously quantify model capabilities in mapping mobile designs to code, diverse benchmarks have been established, ranging from static view reconstruction to complex design-to-implementation workflows.

For instance, APPUI (Yue et al., 2025) establishes a comprehensive benchmark comprising 1.1k pairs of rendered images and code across 12 application categories, specifically curated for the reconstruction of static single-page applications from design mockups. Extending the scope to tool-assisted design workflows, CANVAS (Jeong et al., 2025) targets mobile UI design within the Figma ecosystem. It comprises 598 tool-driven tasks sampled from 3.3k human-crafted designs, evaluating the model's ability to interact with professional design tools.

Complementing direct generation tasks, recent research has pivoted toward granular quality assessment and reward modeling to enhance evaluation precision. UICrit (Duan et al., 2024) introduces a dataset of 3k critiques annotated with bounding boxes and design-quality ratings, serving as a critical resource for alignment via reward modeling. In parallel, UIClip (Wu et al., 2024) provides a screenshot-based scoring model. By leveraging contrastive learning, it functions as an effective reranker for mobile synthesis, surpassing simple pixel-level matching metrics.

#### 3.2.2 Mobile Code Generation Methods

Advancements in mobile-specific methodologies focus on enhancing structural coherence and visual fidelity. However, advancement in this field is notably constrained by data availability. Unlike the web domain, which benefits from the open-source nature of HTML, the mobile domain suffers from inherent data scarcity due to the encapsulated ecosystem of native applications. To navigate these challenges, recent innovations cluster into three core streams: structure-aware refinement, retrieval-augmented strategies, and unified RL-driven paradigms.

To address the intricate structural hierarchy of mobile UIs, several works incorporate hierarchical priors. DeclarUI (Zhou et al., 2024) combines component segmentation with a Page Transition Graph to capture multi-screen navigation logic, further leveraging iterative compilation checks to rectify syntax errors. Similarly, DesignCoder (Chen et al., 2025p) explicitly models UI hierarchy via a UI Grouping Chain and introduces a vision-aware autonomous repair mechanism to refine code post-rendering. To mitigate model hallucinations, RAGG (Kolthoff et al., 2024) employs a Retrieval-Augmented GUI Generation approach. By retrieving relevant references from the Rico dataset (Deka et al., 2017) and employing self-critique loops, it synthesizes more structurally grounded UIs. More recently, the field is witnessing a shift towards unified MLLM architectures. UI-UG (Yang et al., 2025c) incorporates Reinforcement Learning to jointly optimize UI understanding and generation quality within a single model.

Furthermore, emerging paradigms explore intermediate representations and interaction-centric generation. PrototypeFlow (Yuan et al., 2024b) focuses on creating high-fidelity prototypes (e.g., SVG/JSON) rather than final application code, providing editable intermediate checkpoints to facilitate a flexible mobile-oriented creation process. In a different vein, Generative Interface (Chen et al., 2025d) targets NL-to-UI generation. Unlike standard Mobile-to-Code tasks, it utilizes an

LLM to generate task-specific interactive UIs (e.g., HTML/JavaScript) from user queries through structured representations and iterative refinement, prioritizing user experience over pixel-perfect reconstruction.

## 4 Scientific Visualization

While UI-to-Code tasks primarily focus on spatial layout and aesthetic fidelity, scientific visualization necessitates a higher degree of numerical precision and semantic integrity to faithfully render data-driven insights. In this section, we provide a systematic review of the benchmarks and methodologies developed to address these domain-specific challenges.

### 4.1 Statistical Charts

As semantically rich and information-dense data representations, chart images constitute a pivotal domain in visual understanding and reasoning research (Huang et al., 2024). Moving beyond passive perception, the community has increasingly focused on the automated synthesis of high-quality, production-grade visualizations through executable plotting code. Accordingly, chart-oriented code generation has garnered significant scholarly attention. Research in this domain typically encompasses two canonical task formulations: NL-to-Chart and Chart-to-Code generation. The former focuses on synthesizing executable visualization code from natural language queries to faithfully align with user intent, while the latter aims to reverse-engineer code directly from visual chart inputs to reconstruct the underlying programmatic logic. The general task formulation for scientific visualization is depicted in Figure 5.

#### 4.1.1 Chart Code Generation Benchmarks

To rigorously evaluate model capabilities in these domains, diverse evaluation frameworks have been established, evolving from simplified synthetic scenarios to complex real-world assessments.

Benchmarks in the NL-to-Chart domain primarily focus on the fidelity and executability of code synthesized from textual instructions. Pioneering efforts, such as nvBench (Luo et al., 2021) and VisEval (Chen et al., 2024b) leverage synthetic datasets to evaluate code executability and output validity. In contrast, MatPlotBench (Yang et al., 2024f) and PandasPlotBench (Galimzyanov et al., 2025) curate evaluation samples from real-world galleries and leverage an LLM-as-a-judge mechanism for semantic assessment. More recently, the field has pivoted toward handling greater complexity (Lu et al., 2025a; Rahman et al., 2025). For example, nvBench 2.0 (Luo et al., 2025) introduces a large-scale dataset characterized by one-to-many mappings and complex reasoning traces. Simultaneously, PlotCraft (Zhang et al., 2025c) proposes a benchmark with 982 instances that incorporates multi-turn refinement tasks, moving beyond standard single-turn generation.

In parallel, the Chart-to-Code paradigm aims to reverse-engineer executable code directly from visual chart inputs. Existing benchmarks in this domain vary significantly in their data composition, ranging from synthetic to real-world charts. ChartX (Xia et al., 2025a) utilizes synthetic chart images and assesses generation quality via an LLM-based evaluation framework. Conversely, Plot2Code (Wu et al., 2025a) incorporates 132 real-world images and evaluates performance using text-matching metrics against reference code. To enable more robust assessment, ChartMimic (Yang et al., 2024a) introduces data-driven editing tasks alongside direct generation, supported by manu-

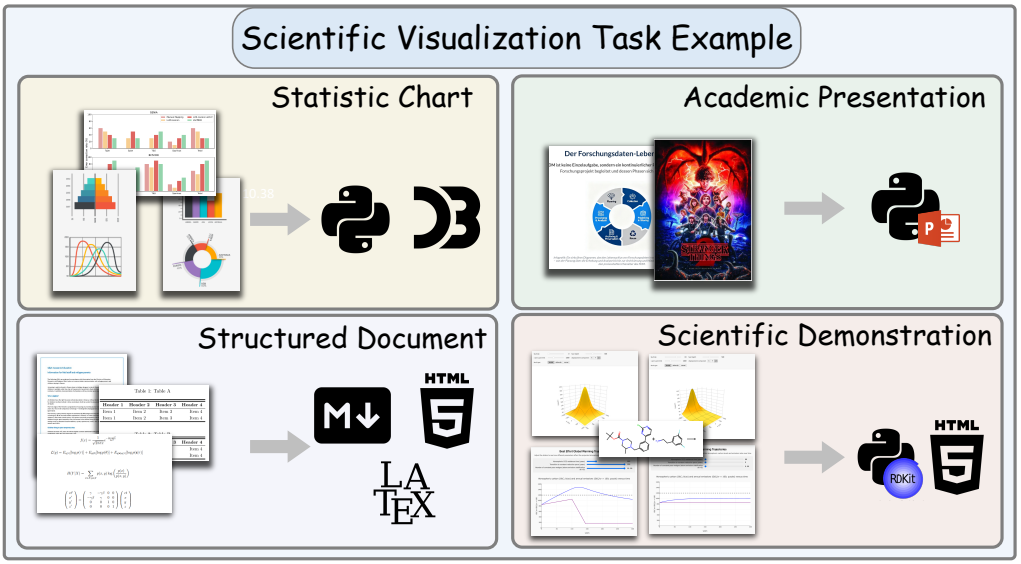


Figure 5: Examples of scientific visualization code generation tasks, including charts, documents, presentations, and demonstrations.

ally annotated ground-truth code for rigorous rule-based evaluation. Recent research expands the scope regarding task diversity and domain coverage. ChartEdit (Zhao et al., 2025c) enriches the editing landscape by introducing diverse instruction types, supported by 1.4k human-annotated instructions on 233 real-world charts. ChartM$^3$ (Yang et al., 2025b) and ChartEditVista (Chen et al., 2025g) pioneer multimodal chart editing by integrating textual instructions and visual indicator-guided mechanisms. Furthermore, ChartGalaxy (Li et al., 2025g) extends the target domain to infographic charts focusing on D3.js. Chart2Code (Tang et al., 2025a) introduces a hierarchical task structure ranging from direct generation to multi-faceted editing. Representative Chart-to-Code and NL-to-Chart benchmarks are summarized in Table 2.

#### 4.1.2 Chart Code Generation Methods

Beyond evaluation protocols, substantial research has focused on developing sophisticated methodologies and large-scale datasets to enhance code generation performance. Recent innovations in this domain are broadly categorized into three evolutionary streams: collaborative agent-based frameworks, specialized instruction tuning, and RL-enhanced optimizations.

Driven by the rapid evolution of LLMs, numerous agent-based frameworks have emerged for NL-to-Chart generation. MatPlotAgent (Yang et al., 2024f) pioneers the use of visual feedback to iteratively refine the generated Matplotlib code. To further enhance performance, VisPath (Seo et al., 2025) proposes a multi-path reasoning and feedback mechanism, while nvAgent (Ouyang et al., 2025a) and AMACE (Namgoong et al., 2025) introduce collaborative multi-agent workflows to tackle complex visualization tasks. Similarly, Doc2Chart (Jain et al., 2025) extends this to document-to-chart scenarios via an interactive protocol.

| Benchmarks | Year | Test Instances | Key Features | Evaluation Metric |
|---|---|---|---|---|
| *Chart-to-Code Benchmark* | | | | |
| Plot2Code (Wu et al., 2025a) | 2025 | 132 | Real-world Images | Model Eval |
| ChartMimic (Yang et al., 2024a) | 2024 | 4.8k | Annotated GT Code | Rule-based, Model Eval |
| ChartEdit (Zhao et al., 2025c) | 2025 | 1.4k | Diverse Editing Types | Model Eval |
| ChartM$^3$ (Yang et al., 2025b) | 2025 | 1k | Visually-guided Editing | ΔSSIM, Model Eval |
| *NL-to-Chart Benchmark* | | | | |
| MatPlotBench (Yang et al., 2024f) | 2024 | 100 | Human Verification | Model Eval |
| VisEval (Chen et al., 2024b) | 2024 | 2.3k | Large-Scale Coverage | Legality Checker, Model Eval |
| Text2Vis (Rahman et al., 2025) | 2025 | 2k | Diverse Data Science Queries | Model Eval |
| PlotCraft(Zhang et al., 2025c) | 2025 | 982 | Multi-turn Generation | Model Eval |

Table 2: Representative benchmarks in the statistical chart generation domain. We categorize these benchmarks into Chart-to-Code and NL-to-Chart settings.

Beyond inference-time frameworks, researchers have proposed specialized training paradigms. Text2Chart31 (Jain et al., 2025) leverages Proximal Policy Optimization (PPO) (Schulman et al., 2017) combined with automated feedback and cycle consistency to optimize visualization instruction-code pairs. VisCoder (Ni et al., 2025b) introduces VisCode-200K, a dataset tailored for Python-based visualization and self-correction. More recently, Step-Text2Vis (Luo et al., 2025) utilizes Step-DPO (Lai et al., 2024) on step-wise preference datasets to improve the logical granularity of the reasoning process. Similarly, PlotCraftor (Zhang et al., 2025c) synthesizes SynthVis-30K, a dataset integrating both single- and multi-turn samples, and achieves substantial performance improvements by leveraging the Qwen3-Coder-30B-A3B (Yang et al., 2025a) backbone for code synthesis.

The Chart-to-Code domain is witnessing a transition from standard SFT to RL-driven paradigms, with a strategic pivot toward editing tasks. Early efforts like MatCha (Liu et al., 2023a) focus on enhancing model capabilities through specialized pre-training strategies. Building on this, ChartLlama (Han et al., 2023) and ChartVLM (Xia et al., 2025a) establish robust baselines by fine-tuning MLLMs on synthetic data derived from closed-source LLMs. ChartMOE (Xu et al., 2024b) leverages large-scale Chart-to-Code data to bridge the modality gap. To bolster Chart-to-Code generation specifically, ChartCoder (Zhao et al., 2025d) adopts a code-centric backbone (e.g., DeepSeek-Coder) and introduces a Snippet-of-Thought (SoT) reasoning strategy on its Chart2Code-160k dataset. Chart2Code53 (Niu et al., 2025d) further scales this by covering a diverse range of plotting functions and chart types. Beyond the prevalent SFT solution, some studies aim to enhance performance through multi-agent collaboration (Xu et al., 2025a; Jiang et al., 2025c). For example, METAL (Li et al., 2025a) utilizes multi-agent collaboration and test-time scaling to optimize code redrawing accuracy. Recently, the rise of RL for LLM reasoning has redefined Chart-to-Code optimization. DualDPO (Zhang et al., 2025h) proposes a dual preference-guided refinement framework to synthesize training preference data, subsequently leveraging DPO (Rafailov et al., 2023) for model optimization. Most notably, MSRL (Chen et al., 2025e) and ChartMaster (Tan et al., 2025) adopt the GRPO (Shao et al., 2024) algorithm augmented with multimodal reward feedback, targeting both code granularity and visual structural alignment. However, they diverge in their data synthesis strategies: MSRL utilizes Gemini-2.0-Flash (Team et al., 2024) for text-driven plotting code generation, whereas ChartMaster employs an image-based paradigm using Qwen2.5-VL-72B (Bai et al., 2025b) to transform visual charts directly into code. Beyond direct generation, the research scope expands to chart editing tasks. ChartReformer (Yan et al., 2024) pioneers natural language-driven

editing by manipulating JSON structures. Recently, ChartEditor (Chen et al., 2025g) introduces a rendering-aware reward signal leveraging the Matplotlib backend for fine-grained supervision.

Furthermore, many works attempt to utilize code to enhance the chart understanding and reasoning capabilities. For example, ReachQA (He et al., 2024), ECD (Yang et al., 2025h) and Chart-R1 (Chen et al., 2025f) utilize code to generate the chart images and the reasoning process. ChartReasoner (Jia et al., 2025) proposes utilizing code as thought to improve chart understanding capacity. In summary, leveraging code-based rendering for data synthesis and logical scaffolding has become a cornerstone of modern chart intelligence.

### 4.2 Structured Document

Structured documents serve as fundamental carriers Structured documents serve as essential carriers of knowledge representation, encapsulating information through the intricate interplay of natural language, tabular data, and mathematical expressions. Unlike traditional Optical Character Recognition (OCR), which primarily targets literal text extraction, structured document code synthesis prioritizes the reconstruction of underlying logical architectures, such as hierarchical schemas, complex tables, and nested formulas. To this end, research in this domain focuses on translating visual inputs into structured code representations, including Markdown, HTML, and LaTeX, thereby enabling precise, automated document parsing and semantic recovery.

#### 4.2.1 Structured Document Code Generation Benchmarks

Existing benchmarks for structured document code generation are typically categorized into three canonical task formulations, targeting distinct visual structures: Document-to-Markdown, Table-to-Code, and Formula-to-LaTeX.

The Document-to-Markdown paradigm focuses on extracting holistic structured content from document images. To assess OCR capabilities in real-world scenarios, Ocean-OCR (Chen et al., 2025k) curates an evaluation dataset with 200 samples sourced from diverse English and Chinese papers, targeting three practical tasks ranging from document understanding to handwritten text recognition. Moving towards complex full-page layouts, the field has advanced to handling systematic PDF parsing. OmniDocBench (Ouyang et al., 2025b) introduces a comprehensive benchmark comprising 1.3k PDF pages, characterized by detailed block- and span-level annotations to support flexible multi-level evaluation. In parallel, olmOCR-Bench (Poznanski et al., 2025a) consists of 1.4k distinct PDF documents and enables fine-grained assessment through 7k unique test cases, covering both general patterns and challenging extraction tasks such as tables and formulas.

The Table-to-Code paradigm aims to translate tabular data into machine-readable markup, predominantly spanning HTML and LaTeX formats. For the HTML-based tasks, pioneering benchmarks such as TableBank (Li et al., 2020a) utilize weak supervision from Word and LaTeX documents to facilitate table detection and structure recognition with 559k training samples. To enable end-to-end evaluation, PubTabNet (Zhong et al., 2020) provides detailed HTML annotations for scientific tables and introduces the Tree-Edit-Distance-based Similarity (TEDS) metric for accurate structure assessment. Conversely, FinTabNet (Zheng et al., 2021) extends the target domain to financial reports, addressing unique challenges in layout and visual style via 10k test tables with cell-level annotations. More recently, the research scope has expanded to the syntactically complex domain of Table-to-LaTeX. Tab2LaTeX (Jiang et al., 2025b) pioneers this sub-domain by providing 5k

| Benchmarks | Year | Test Instances | Key Features |
|---|---|---|---|
| *Document-to-Markdown Benchmark* | | | |
| OmniDocBench (Ouyang et al., 2025b) | 2025 | 1.3k | Multi-level Annotations |
| olmOCR-Bench (Poznanski et al., 2025a) | 2025 | 1.4k | Unit-test Evaluation |
| Ocean-OCR (Chen et al., 2025k) | 2025 | 200 | Real-world Scenarios |
| *Table-to-HTML Benchmark* | | | |
| TableBank (Li et al., 2020a) | 2025 | 3k | Structure Recognition |
| PubTabNet (Zhong et al., 2020) | 2025 | 10k | Structure and Content Recognition |
| FinTabNet (Zheng et al., 2021) | 2025 | 17k | Financial Domain |
| *Table-to-LaTeX Benchmark* | | | |
| TAB2LATEX (Jiang et al., 2025b) | 2025 | 5k | Unified Image Resolution |
| Table2LaTeX-RL (Ling et al., 2025) | 2025 | 1.2k | Complexity Stratification |
| *Formula-to-LaTeX Benchmark* | | | |
| IMG2LATEX-100K (Deng et al., 2017) | 2017 | 10k | Dual-aspect Evaluation |
| UniMER-Test (Wang et al., 2024a) | 2024 | 23k | Multi-scenario Evaluation |
| CSFormula (Zhong et al., 2025b) | 2025 | 3k | Multi-granularity Evaluation |

Table 3: Representative benchmarks in Structured Document Parsing. We categorize these approaches into distinct domains based on input type (e.g., Document, Formula, Table) and output language (e.g., Markdown, Latex).

compilable source code samples to specifically evaluate renderable LaTeX generation. To handle varying structural complexities, Table2LaTeX-RL (Ling et al., 2025) categorizes tables into simple (<100 cells), medium, and complex (>160 cells) levels, supporting fine-grained evaluation of model capabilities in processing \multirow and \multicolumn commands.

Finally, the Formula-to-LaTeX paradigm focuses on synthesizing executable LaTeX sequences from mathematical expressions. Pioneering the field, IM2LaTeX-100K (Deng et al., 2017) establishes a robust baseline by sourcing samples from academic papers, evaluating performance via image-level pixel matching and text-level BLEU scores. To address the limitations of simple scenarios, UniMER-Test (Wang et al., 2024a) enriches the landscape by introducing 23k samples covering four representative types, ranging from simple printed to complex handwritten expressions. Recently, CSFormula (Zhong et al., 2025b) serves as a challenging benchmark targeting authentic scientific literature. It encompasses multidisciplinary formulas across mathematics, physics, and chemistry, and hierarchically organizes them into line-, paragraph-, and page-level categories to evaluate context-dependent generation. Representative benchmarks are demonstrated in Table 3.

#### 4.2.2 Structured Document Code Generation Methods

Recent proposed methods on structured document code generation span diverse approaches, ranging from traditional pipeline-based frameworks to end-to-end VLM architectures and RL-driven optimizations.

The Document-to-Markdown paradigm has long focused on extracting structured content from document images. Traditional pipeline-based OCR models (Wang et al., 2024b; Cui et al., 2025b; Paruchuri, 2025) decompose the task into sequential stages, typically starting with layout analysis for region segmentation and subsequently employing region-specific parsers to arrange content in reading order. For example, MinerU (Wang et al., 2024b) integrates PDF-Extract-Kit (OpenDataLab, 2025) with refined pre- and post-processing strategies to improve extraction accuracy across

diverse document formats. Driven by the semantic capabilities of VLMs, recent works have advanced this domain. Pipeline-based VLM methods (Feng et al., 2025; Li et al., 2025f; Cui et al., 2025a) embed VLMs into multi-stage workflows. Notably, PaddleOCR-VL (Cui et al., 2025a) combines traditional layout detection with a unified VLM for holistic content extraction. In contrast, end-to-end VLMs (Liu et al., 2025c; Poznanski et al., 2025a; Wei et al., 2025a) employ unified architectures to synthesize structured outputs directly in a single step. For instance, dots.ocr (rednote, 2025) leverages native-resolution vision encoders to achieve high-fidelity extraction. More recently, the rise of RL has sparked interest in optimizing parsing logic. Infinity Parser (Wang et al., 2025a) and Logics-Parsing (Chen et al., 2025m) design verifiable rewards to capture structural consistency, while olmOCR 2 (Poznanski et al., 2025b) employs diverse binary unit tests as reward signals. Similarly, FD-RL (Zhong et al., 2025a) exploits high-entropy patterns in format-intensive content to guide RL optimization toward challenging samples.

In parallel, the Table-to-Code domain aims to translate tabular data into machine-readable markup, evolving from visual detection to unified language modeling. Early approaches for Table-to-HTML (Zhong et al., 2020; Zheng et al., 2021) introduce attention-based frameworks to perform structure recognition and cell localization jointly. To enhance parsing accuracy, Transformer-based models such as TableFormer (Nassar et al., 2022) and VAST (Huang et al., 2023) employ end-to-end architectures that integrate object detection decoders and coordinate sequence modeling. UniTable (Peng et al., 2024) further unifies the paradigm by casting structure and content extraction as a single language modeling task, while SLANet (Cui et al., 2025b) combines text detection with structure prediction to handle complex borderless tables. More recently, MLLMs (rednote, 2025; Mandalm, 2025; Niu et al., 2025a) have propelled the field forward, offering exceptional flexibility in table recognition. Beyond HTML, the community also addresses the syntactically complex domain of Table-to-LaTeX. LaTeXNet (Xia et al., 2025b) achieves unified recognition through a two-stage routing architecture that dynamically directs inputs to specialized submodules. To guide precise corrections, Latte (Jiang et al., 2025b) utilizes an iterative refine-and-correct framework supported by a novel ImageEdit algorithm. Furthermore, Table2LaTeX-RL (Ling et al., 2025) proposes a dual-reward strategy named VSGRPO, jointly optimizing structural accuracy via the TEDS metric and visual fidelity via CW-SSIM. In a broader context, OmniCaptioner (Lu et al., 2025c) establishes a unified cross-domain paradigm, generating precise LaTeX code for tables, formulas, and geometric figures alongside natural scene descriptions.

Finally, the Formula-to-LaTeX paradigm, also known as Mathematical Expression Recognition (MER), focuses on transforming mathematical images into executable LaTeX sequences. Building upon the Transformer architecture (Vaswani et al., 2017), early pioneers like Pix2tex (Blecher, 2022) and Texify (Paruchuri, 2023) establish the standard encoder-decoder paradigm. Subsequent research focuses on granular refinements to address structural challenges. Specifically, PosFormer (Guan et al., 2024) explicitly models spatial relationships via a position forest structure, while UniMERNet (Wang et al., 2024a) enhances feature extraction through detail-aware encoding. Additionally, HD-Net (Wang et al., 2025f) resolves hierarchical complexity by employing sub-formula modules. Recent advances introduce end-to-end models designed to balance accuracy with efficiency. PP-FormulaNet (Liu et al., 2025b) addresses this trade-off through dual architectures tailored for different scenarios. Conversely, DocTron-Formula (Zhong et al., 2025b) directly leverages general vision-language models without task-specific designs, effectively handling diverse granularities. Beyond specialized architectures, Docfusion (Chai et al., 2025b) bridges the gap between continuous coordinate-based detection and discrete token-based recognition, leveraging

Gaussian-Kernel Cross-Entropy Loss (GK-CEL) to enable simultaneous layout detection and content recognition within a lightweight framework.

### 4.3 Academic Presentations

The synthesis of presentation slides and academic posters represents a specialized frontier within multimodal code generation (Chen et al., 2025i). Distinct from standard code synthesis tasks where logical correctness serves as the primary objective, this domain presents a unique duality of challenges involving semantic compression to distill dense information and spatial layout reasoning to enforce aesthetic and structural integrity. Consequently, advancing this field requires models that transcend fundamental code generation capabilities to demonstrate a coherent grasp of visual hierarchy, multimodal alignment, and document-level organization.

#### 4.3.1 Presentations Generation Benchmarks

The evolution of benchmarks in the presentation domain reflects a paradigm shift from basic content extraction toward comprehensive evaluations of layout reasoning, aesthetic fidelity, and structural coherence.

Early evaluation frameworks in slide generation mainly focused on content matching. Recent benchmarks, however, target the structural complexity of real-world documents. For instance, SlidesBench (Ge et al., 2025) and Zenodo10K (Zheng et al., 2025) introduce large-scale datasets to assess design quality and coherence. Notably, Zenodo10K uses MLLM-based metrics via the PPTEval framework to better align with human preferences. Beyond generation from scratch, the capability to modify existing slides is critical for practical assistants. To this end, TSBench (Jung et al., 2025) evaluates fine-grained instruction following across text editing and visual formatting. Furthermore, as the domain expands into multimodal outputs, benchmarks like Slide2Code (Tang et al., 2025b) and Doc2Present (Shi et al., 2025) propose to assess visual reverse engineering and audio-visual alignment, respectively, pushing the evaluation boundary from static pages to dynamic presentations.

In the sub-field of poster generation, the primary evaluation challenge is assessing information density and layout rationality. Unlike slides, posters require compressing long-context papers into a single page. Recent benchmarks such as Paper2Poster (Pang et al., 2025) and P2PEval (Sun et al., 2025d) address this by introducing reader-simulation metrics like the Paper Quiz, alongside dual-track evaluations that cover both universal and fine-grained criteria. These methods ensure that the condensed layouts remain informative and readable.

#### 4.3.2 Presentations Generation Methods

Unlike broader code generation fields that predominantly focus on model training, current methodologies in this domain center on agentic workflows. Within this framework, technical approaches diverge into programmatic generation, iterative editing, and layout-focused spatial planning.

A prominent paradigm involves abstracting the creation process into programmatic calls. Approaches like AutoPresent (Ge et al., 2025) utilize modular libraries such as SlidesLib, which allows LLMs to focus on high-level content planning while delegating rendering details to specific API calls. Similarly, SlideCoder (Tang et al., 2025b) employs a hierarchical retrieval mechanism to

| Benchmarks | Year | Test Instances | Key Features | Evaluation Metric |
|---|---|---|---|---|
| *Slide Generation / Editing Benchmarks* | | | | |
| SlidesBench (Ge et al., 2025) | 2025 | 585 | Domain Diversity | Ref-based, Design Quality |
| TSBench (Jung et al., 2025) | 2025 | 379 | Fine-grained Editing | Exec., Instruction Following |
| Slide2Code (Tang et al., 2025b) | 2025 | 300 | Visual Reverse Engineering | Code Match, Visual Sim |
| Doc2Present (Shi et al., 2025) | 2025 | 30 | Audio-Visual Alignment | PresentEval (Quiz, Subjective) |
| EvoPresent (Liu et al., 2025a) | 2025 | 650 | Aesthetic Awareness | Aesthetic Score, PPL |
| *Poster Generation Benchmarks* | | | | |
| Paper2Poster (Pang et al., 2025) | 2025 | 100 | Multimodal Context | Paper Quiz, VLM Score |
| P2PEval (Sun et al., 2025d) | 2025 | 121 | Dual-track Evaluation | Fine-grained Checklist |

Table 4: Representative benchmarks for Academic Presentation Generation, classified into slide and poster generation tasks.

reverse-engineer rendering code from images, effectively bridging the gap between visual design and code representation. In contrast to these generation-first approaches, other works treat the problem as an iterative editing process, which offers finer control over user constraints. Recent works, such as PPTAgent (Zheng et al., 2025) and Talk-to-Your-Slides (Jung et al., 2025), operate by analysing existing templates to execute targeted modifications. This separation of instruction understanding from object manipulation significantly reduces the computational overhead compared to vision-based UI agents. To further enhance aesthetic quality, which remains a significant bottleneck for automated synthesis, PreGenie (Xu et al., 2025d) and EvoPresent (Liu et al., 2025a) introduce optimization feedback loops. Specifically, PreGenie employs a dual-review mechanism comprising code and page reviewers. In parallel, EvoPresent develops PresAesth, a multi-task RL model that integrates scoring, defect adjustment, and layout comparison to steer agents toward aesthetically superior results.

Transitioning from multi-page slides to the single-page constraints of poster generation, the critical technical bottleneck shifts toward spatial planning for content of variable lengths. PosterAgent (Pang et al., 2025) addresses this challenge through a binary-tree layout strategy integrated with a visual feedback mechanism termed the painter-commenter architecture, which dynamically mitigates text overflow issues. Alternatively, frameworks like P2P (Sun et al., 2025d) and PosterGen (Zhang et al., 2025i) adopt a decoupled paradigm that distinguishes content extraction from layout design. By deploying specialized agents acting in roles such as stylists for color optimization or curators for narrative structuring, these frameworks emulate professional design workflows, ensuring that the rigorous compression of scientific content preserves a coherent visual hierarchy. To further improve the poster generation efficiency, EfficientPosterGen (Tang et al., 2026) proposes the key information identification and visual-based token compression strategy to reduce the poster generation costs.

### 4.4 Scientific Demonstration

Scientific demonstration visualization involves generating visual demonstrations from complex scientific data, which presents a unique challenge for models to master domain-specific knowledge. While proficient at general code generation, LLMs must adhere to rigorous research standards including numerical precision, visual fidelity, and contextual correctness to ensure the production of scientifically valid and publication-ready figures. This fundamental demand for scientific in-

tegrity defines the nature of the task and necessitates the development of specialized evaluation methodologies.

#### 4.4.1 Scientific Demonstration Generation Benchmarks

Current benchmarks for scientific code generation are expanding beyond standard plotting (Chen et al., 2025q) and command line interaction (Sun et al., 2025c), distinguishing themselves by addressing highly specialized domain knowledge and necessitating complex multimodal reasoning. This evolution can be categorized into two emerging paradigms: domain-specific visual translation and interactive pedagogical reasoning.

The Domain-Specific Visual Translation paradigm focuses on the precise interpretation of scientific diagrams into executable scripts. ChemDraw (Zhao et al., 2025b) typifies this approach in computational chemistry by evaluating the translation of molecular diagrams into executable Python scripts. It rigorously measures the model's capability to utilize SMILES strings and chemistry-specific libraries, such as RDKit, to accurately reconstruct complex chemical structures.

In parallel, the field is witnessing a strategic shift toward Interactive Pedagogical Reasoning, aiming to generate code that not only visualizes but also demonstrates and explains scientific concepts. EduVisBench (Ji et al., 2025) introduces a multi-domain framework for assessing visual reasoning within educational settings, employing a fine-grained evaluation rubric informed by pedagogical theory to handle diverse STEM problem sets. To enable dynamic user engagement, InteractScience (Chen et al., 2025h) pioneers the benchmark for Scientific Demonstration Code Generation. It assesses the integrated ability to synthesize accurate scientific knowledge with interactive front-end code, utilizing a hybrid framework that combines Programmatic Functional Testing (PFT) and Visually-Grounded Qualitative Testing (VQT). More recently, TheoremExplainBench (Ku et al., 2025a) extends the scope to multimodal video generation, evaluating an LLM's capacity to communicate complex theorem reasoning through visually intuitive and pedagogically sound Python-generated videos.

#### 4.4.2 Scientific Demonstration Generation Methods

Recent methodologies for scientific code generation primarily target two distinct application scenarios: scientific data visualization, which focuses on plotting accuracy and data fidelity, and scientific education and explanation, which aims to teach and explain concepts through code.

In the domain of scientific data visualization, research has evolved from inference-time optimizations to large-scale data synthesis. Initial efforts focused on enhancing the reliability of LLMs through prompt engineering. Chakroborti et al. (Chakroborti et al., 2025) introduce three complementary strategies to bolster generation quality, including data-aware prompt disambiguation for resolving dataset ambiguities, retrieval-augmented prompt enhancement for contextual support, and iterative error repair for self-correcting runtime errors. To address the bottleneck of data scarcity, CoSyn (Yang et al., 2025f) proposes a scalable framework that leverages code-based rendering to synthesize high-quality multimodal data. By contributing a large-scale dataset of 400K synthetic images, incorporating samples from chemical and circuit domains, as well as 2.7M instruction-tuning samples. The use of underlying rendering code ensures precise ground truth, significantly improving the model's alignment. TinyChemVL (Zhao et al., 2025e) proposes utilizing RDKit code to render optimized molecules for the molecular property optimization task, effectively enabling

image generation capabilities. Furthermore, MathCoder-VL (Wang et al., 2025h) introduces FigCodifier, utilizing a model-in-the-loop approach to co-develop the model and dataset. Starting with DaTikZ (Belouadi et al., 2023) for initialization, it iteratively synthesizes training samples to construct ImgCode-8.6M, recognized as the largest image-to-code dataset to date.

While the aforementioned methods focus on data fidelity, a parallel stream of research addresses scientific education and explanation, progressing from interactive webpages to video generation. EduVisAgent (Ji et al., 2025) pioneers the use of multi-agent frameworks for interactive education. It coordinates five specialized agents to structure learning objectives and construct step-by-step problem-solving logic, ultimately synthesizing abstract concepts into interactive learning webpages for students. Moving from web-based interaction to video generation, TheoremExplainAgent (Ku et al., 2025a) proposes a collaborative system to explain complex theorems. It features a Planner Agent for creating narrative scripts and visualization plans, and a Coding Agent that translates these plans into executable Manim animation scripts. Furthermore, it integrates an Agentic RAG module with a router to dynamically retrieve documentation, enabling the generation of visually intuitive and pedagogically sound videos.

## 5 Structured Graphics

Different from UIs and scientific visualizations, structured graphics transition visual representation from pixel-based rasters to symbolic, executable code, enabling inherent editability and logical transparency. We systematically examine these advancements across three primary modalities: Scalable Vector Graphics (SVG) for 2D design, logic diagrams for architectural abstraction, and Computer-Aided Design (CAD) for parametric 3D modeling.

### 5.1 Scalable Vector Graphics (SVG)

Scalable Vector Graphics (SVG) serves as a fundamental format for icons and UI design due to its resolution independence and editability. The task of SVG code generation automates the conversion of high-level inputs, including text descriptions and raster images, into renderable vector code to mitigate manual design overhead. This field primarily comprises two key streams known as NL-to-SVG, which generates code from textual prompts, and Image-to-SVG, which focuses on converting raster images to vectors. We summarize representative SVG and the following CAD domain benchmarks in Table 5.

#### 5.1.1 SVG Code Generation Benchmarks

The transition from optimization and sequence-based frameworks to LLM-driven code generation has rendered traditional pixel-level metrics insufficient. Unlike earlier methods that prioritized reconstruction fidelity, modern approaches necessitate the assessment of semantic alignment, code logic, and structural reasoning. Consequently, a new generation of benchmark suites has emerged to evaluate these multidimensional capabilities.

VGBench (Zou et al., 2024) serves as a key platform for evaluating LLM capabilities in vector graphics understanding and generation, noting the challenge posed by low-level formats like SVG. SVG-Bench (Rodriguez et al., 2023) establishes a unified evaluation standard by providing tailored datasets and metrics to overcome limitations in scope and reproducibility. Meanwhile, MMSVG-

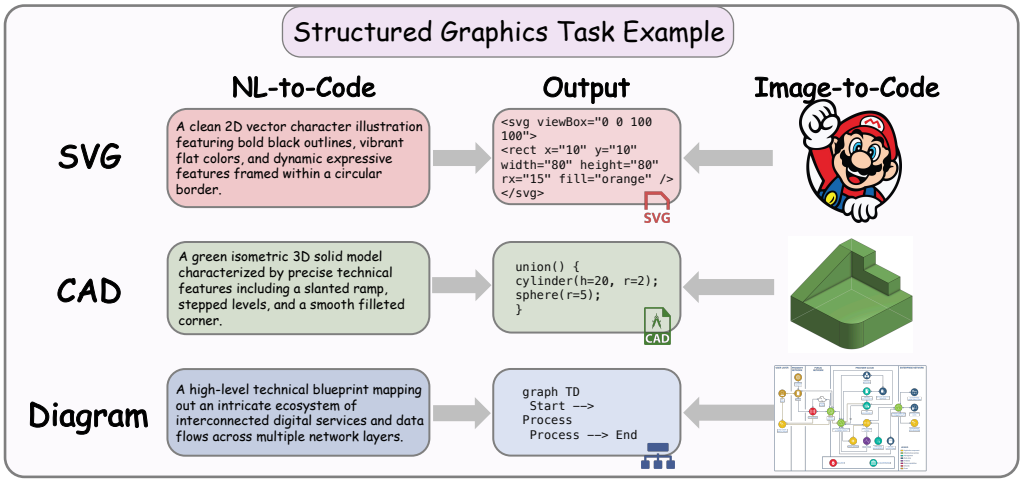


Figure 6: Examples of structured graphics generation tasks: SVG, CAD, and Diagram.

Bench (Yang et al., 2025e) focuses on evaluating multi-modal SVG generation tasks encompassing complex illustrations and anime characters. For assessing advanced LLM capabilities, the UniSVG Benchmark (Li et al., 2025d) offers multi-dimensional evaluation, while SVGenius (Chen et al., 2025j), with 2377 queries, evaluates model understanding, editing, and generation, establishing Style Transfer as the ultimate challenge. Furthermore, VCode (Lin et al., 2025b) introduces the CodeVQA protocol, reframing multimodal understanding as the generation of SVG code that preserves symbolic meaning for downstream reasoning.

Evaluation in SVG generation focuses on quantifying semantic alignment, visual quality, and structural fidelity across different tasks. For NL-to-SVG tasks, the core measures are derived from multimodal embedding spaces to ensure alignment between text and the generated image. These include the CLIP Score (Radford et al., 2021), BLIP Score (Li et al., 2022a), and SigLIP Score (Zhai et al., 2023) for semantic consistency, alongside perceptual metrics like FID (Heusel et al., 2017) and the Aesthetic/HPS (Wu et al., 2023b) scores for overall visual appeal. Structural quality is specifically assessed via the Path-Structure Similarity Score (PSS) (Chen et al., 2025j) and metrics on code efficiency. In contrast, Image-to-SVG evaluation primarily targets Reconstruction Fidelity using traditional image quality metrics such as MSE (Wang & Bovik, 2009), PSNR (Hore & Ziou, 2010), SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018), complemented by the DINOScore (Oquab et al., 2023) which offers deeper perceptual likeness. Finally, User Study metrics provide crucial Subjective Validation, encompassing human evaluations of visual quality, semantic accuracy, and overall utility.

### 5.1.2 SVG Code Generation Methods

The field of SVG generation witnesses a parallel evolution in both data resources and methodological paradigms. While dataset development traces a clear progression from foundational geometric data to complex multimodal resources, generation methods correspondingly evolve from optimization-based frameworks and sequence models to LLM-driven approaches that integrate large-scale data synthesis.

| Benchmarks | Year | Test Instances | Key Features | Evaluation Metric |
|---|---|---|---|---|
| *SVG Generate Benchmark* | | | | |
| VGBench (Zou et al., 2024) | 2024 | 10.1k | Multi-format (SVG, TikZ, Graphviz) | CLIP, FID, Accuracy |
| SVG-Bench (Rodriguez et al., 2023) | 2025 | 24.1k | Unified Evaluation (10 Datasets) | CLIP, LPIPS, SSIM |
| MMSVG-Bench (Yang et al., 2025e) | 2025 | 450 | Character-Reference Generation | Model Eval, CLIP |
| UniSVG (Li et al., 2025d) | 2025 | 2.8k | Unified Generation & Evaluation | CLIP, BERTScore, SSIM |
| SVGenius (Chen et al., 2025j) | 2025 | 2.4k | Stratified Und., Edit. & Gen. | PSS, rCLIP, rMSE |
| VCode (Lin et al., 2025b) | 2025 | 464 | CodeVQA & Symbolic Reasoning | Accuracy, SigLIP Score |
| *CAD Generate Benchmark* | | | | |
| Fashion 360 (Willis et al., 2021) | 2021 | 1.7k | Human Designed CAD Programs | IoU, Exact Recon. |
| DeepCAD (Wu et al., 2021) | 2021 | 8.9k | Large-scale CAD Operation Sequences | Chamfer Distance (CD) |
| Text2CAD (Khan et al., 2024b) | 2024 | 32k | Multi-Level Annotations | F1, CD, Model-based |
| CADReview (Chen et al., 2025c) | 2025 | 1.6k | CAD repair with reference images | ROUGE, BERTScore |

Table 5: Representative benchmarks in the structured symbolic code generation domain. Given that numerous studies introduce custom benchmarks for evaluation, we restrict our discussion to a selection of representative works.

Traditional paradigms in SVG generation primarily formulate the task as either an inverse graphics problem or a sequence modeling challenge, relying on optimization techniques and specialized neural architectures. Optimization-based methods utilize differentiable rasterizers to refine vector parameters via loss minimization. For instance, DiffVG (Li et al., 2020b) and LIVE (Ma et al., 2022) optimize parameters via loss minimization. In the realm of NL-to-SVG, VectorFusion (Jain et al., 2023) adapts Score Distillation Sampling (SDS) to optimize shape parameters. Advancing this paradigm, SVGDreamer (Xing et al., 2024b) incorporates a semantic-driven image vectorization (SIVE) process for decomposition and employs Vectorized Particle-based Score Distillation (VPSD) to enhance convergence. Conversely, Image-to-SVG approaches prioritize contour and region fidelity. SAMVG (Zhu et al., 2024a) integrates the Segment-Anything Model (SAM) (Kirillov et al., 2023) for segmentation-guided vectorization, and NeuralSVG (Polaczek et al., 2025) adopts an Implicit Neural Representation to encode SVGs with a strict hierarchical structure. In parallel, neural sequence methods learn explicit mappings for generation. Early approaches prioritize structural representation learning. DeepSVG (Carlier et al., 2020) pioneers this direction by introducing a Hierarchical VAE alongside the SVG-Icons8 dataset to facilitate non-autoregressive generation, whereas Im2Vec (Reddy et al., 2021) trains a VAE using exclusively raster supervision to generate paths without explicit vector annotations. With the adoption of Transformers, IconShop (Wu et al., 2023a) establishes the core autoregressive paradigm for NL-to-SVG conversion by converting text and paths into a unified linear token sequence. SuperSVG (Hu et al., 2024a) enhances fidelity in Image-to-SVG tasks through superpixel decomposition within a multistage pipeline.

Driven by the rapid evolution of LLMs, the field transitions toward utilizing the structural reasoning of VLMs, supported by the synthesis of large-scale datasets to bridge the modality gap. To establish robust baselines in this data-driven era, StarVector (Rodriguez et al., 2023) introduces SVG-Stack, a foundational corpus with over 2M SVG samples, and achieves superior performance by unifying NL-to-SVG and Image-to-SVG tasks through primitive-aware parameterization. Addressing specific domain limitations, LLM4SVG (Xing et al., 2025b) enhances instruction adherence by synthesizing the SVGX-SFT dataset comprising 580k samples and introducing 55 domain-specific semantic tokens. To enrich structural diversity beyond standard shapes, SVGFusion (Xing et al., 2024a) proposes the SVGX-Dataset to support complex primitives like elliptic arcs, while ColorSVG-100K (Chen & Pan, 2025) specifically addresses the scarcity of rich color information

in generative tasks. OmniSVG (Yang et al., 2025e) addresses the complexity of illustrations by curating MMSVG-2M, the largest annotated resource that enables end-to-end multi-task learning and resolves long-sequence bottlenecks in complex visualizations.

Recent advancements focus on enhancing the reasoning capabilities and interactivity of SVG generation frameworks, moving beyond simple instruction following. Reasoning-driven approaches, such as Reason-SVG (Xing et al., 2025a), integrate RL with a six-stage Drawing-with-Thought (DwT) paradigm, supported by the SVGX-DwT-10k dataset to ensure structural validity. Simultaneously, SVGThinker (Chen et al., 2025b) and SVGen (Wang et al., 2025c) utilize CoT reasoning to guide instruction-aligned generation, with SVGen further leveraging the SVG-1M dataset for optimization. Beyond static generation, the field expands toward interactive editing. RoboSVG (Wang et al., 2025g) introduces RoboDraw, a dataset of 1 million samples, to facilitate interactive generation and code completion from partial inputs. Hybrid architectures also combine the strengths of different paradigms, as seen in Chat2SVG (Wu et al., 2025c), which pairs LLM-based template generation with geometric optimization via diffusion models to facilitate iterative editing. Furthermore, to overcome the non-differentiability inherent in autoregressive models, RLRF (Rodriguez et al., 2025) applies a Rendering-Aware RL mechanism, which directly optimizes the visual fidelity of the generated vector code.

## 5.2 Diagram

Logic diagrams are a fundamental abstraction in system architecture, visualizing complex control flows and algorithmic logic. Previously, this domain prioritizes the Diagram-to-Code paradigm, focusing on translating hand-drawn flowcharts into structured representations. Recently, leveraging the advanced comprehension capabilities of LLMs, NL-to-Diagram synthesis has also been explored.

### 5.2.1 Diagram Code Generation Benchmarks

Regarding the evaluation of diagram code generation, previous exploratory studies rely on qualitative analysis using limited-scale datasets. Existing benchmarks are broadly classified into text-centric NL-to-Diagram generation and image-centric Diagram-to-Code synthesis.

In the NL-to-Diagram domain, pioneering works such as FC2Code (Liu et al., 2022) employ a structure recognition model to transcribe flowcharts into pseudo-code, subsequently evaluating the efficacy of transforming this intermediate representation into executable code. Recently, VisPlotBench (Ni et al., 2025a) incorporates Mermaid code generation tasks to assess model proficiency in translating natural language into executable diagrammatic code. In contrast, contemporary Diagram-to-Code benchmarks evaluate visual structural interpretation across diverse domains. Flow2Code (He et al., 2025) leverages rendering engines to synthesize diagram images from established codebases, curating a multilingual test set of 1.6k samples across 15 languages evaluated via Pass@k. StarFlow (Bechard et al., 2025) establishes a rigorous benchmark for translating sketch images into structured JSON workflows. This dataset comprises 2.7k samples covering five distinct visual styles and incorporates a comprehensive metric suite. In the UML domain, UML-LLaVA (Bates et al., 2025) establishes dual benchmarks comprising a large-scale synthetic dataset and a curated real-world set of 57 samples, facilitating evaluation across both in-domain and out-of-domain scenarios. $M^2$Eval (Chai et al., 2025a) explores the task of code generation based on code diagrams and constructs 300 samples across 10 programming languages. Extending beyond

software, MMVG (Chang et al., 2024) applies this visual-structural paradigm to hardware design by benchmarking the generation of Verilog code from block diagrams.

#### 5.2.2 Diagram Code Generation Methods.

Current methods have largely converged on specialized SFT paradigms, focusing on optimizing training data composition and scale for specific generation tasks.

In the diagram-to-code domain, approaches prioritize the alignment of visual structure with specific code language. To address domain-specific requirements, UML-LLaVA (Bates et al., 2025) synthesizes a large-scale corpus of activity and sequence diagrams based on randomized textual descriptions to fine-tune the LLaVA-1.5 (Liu et al., 2023b) architecture. Flow2Code (He et al., 2025) validates the Flowchart-to-Code paradigm by leveraging 15k training samples derived from established codebases, effectively bridging the semantic gap between visual flowcharts and executable syntax. Similarly, StarFlow (Bechard et al., 2025) constructs a composite training set of 18k samples that blends synthetic graphs, hand-drawn sketches, and UI renders, enabling models to generate structured JSON workflows. Draw with Thought (Cui et al., 2025c) proposes generating mxGraph code from scientific diagrams via cognitively inspired CoT prompting. More recently $M^2$Coder (Chai et al., 2025a) introduces $M^2$C-Instruct, a massive corpus spanning 50 programming languages with over 13.1M samples, thereby improving code generation accuracy and alignment with architectural intent. Beyond direct code generation, works such as FlowVQA (Singh et al., 2024) leverage flowchart code to synthesize Visual VQA tasks, thereby extending the utility of diagrammatic code representations.

### 5.3 Computer-Aided Design (CAD)

In the realm of Computer-Aided Design (CAD), generative tasks are undergoing a paradigm shift from geometry synthesis to procedural synthesis. While traditional approaches focus on reconstructing static representations like B-reps or meshes, these formats often lack editability and semantic depth. In contrast, representing CAD models as code, which ranges from serialized command sequences to high-level programming scripts such as CadQuery, enables the capture of the underlying construction logic and parametric dependencies. Consequently, methodologies in this domain are generally categorized into two paradigms: Natural Language-to-CAD (NL-to-CAD) and CAD-to-Code. NL-to-CAD approaches focus on synthesizing CAD models directly from textual descriptions. In contrast, CAD-to-Code methods aim to reverse-engineer procedural code from visual modalities such as images or point clouds.

#### 5.3.1 CAD Code Generation Benchmarks

To evaluate CAD generation performance, it is common practice to construct custom datasets and split them into training and testing sets. Since most of these datasets are unnamed and not utilized in other works, this section focuses on benchmarks that are widely adopted or designed for specific task settings.

Pioneering works such as DeepCAD (Wu et al., 2021) and Fashion 360 (Willis et al., 2021) serve as widely adopted benchmarks in this category, providing approximately 8k and 1.7k samples for testing and validation, respectively. Subsequent research typically follows these established protocols, training on the designated training sets and employing the corresponding test sets for evaluation.

With the integration of Large Language Models (LLMs), recent methodologies leverage these models for automated data annotation and filtering. For instance, Text2CAD (Khan et al., 2024b) employs an LLM-driven pipeline for instruction generation and filtering to construct a dataset that maps abstract CAD descriptions to detailed specifications. Focusing on CAD code understanding, CADTalk (Yuan et al., 2024a) presents a benchmark for code commenting comprising over 5.3k instances, which include 5.2k machine-generated samples and 45 human-authored programs. Similarly, SGP-Bench (Qiu et al., 2024) utilizes SVG and CAD code to assess the semantic consistency of symbolic graphics programs. Although originally proposed for understanding tasks, it is increasingly adopted to evaluate code generation capabilities. Finally, addressing the domain of code correction, CADReview (Chen et al., 2025c) constructs a specialized benchmark for editing tasks containing 1.6k samples, pairing erroneous CAD programs with their corresponding correct reference images.

Furthermore, several benchmarks are introduced alongside specific training methodologies to address specialized settings. Notable examples include ExeCAD (Niu et al., 2025c) and CADExpert (Niu et al., 2025b), which target CADQuery code generation by incorporating natural language prompts and three-view images across diverse task scenarios.

### 5.3.2 CAD Code Generation Methods

Previously, a significant body of research focused on generating Boundary Representations (B-rep). However, to align with the scope of multimodal code generation, we exclude direct B-rep synthesis methods and exclusively introduce approaches utilizing command sequence representations and programmatic code generation.

DeepCAD (Wu et al., 2021) pioneers the formulation of CAD model generation as the synthesis of sequential CAD commands. It utilizes a Transformer-based autoencoder to embed CAD models first and generate executable command sequences, contributing a comprehensive dataset of 178k designs to support this sequence-based generation. Building upon this paradigm, many subsequent works have adopted command sequence representations for CAD generation. In the NL-to-CAD domain, early approaches such as Text2CAD (Khan et al., 2024b) and CAD-Translator (Li et al., 2024b) employ encoder-decoder architectures to translate textual descriptions into command sequences. Specifically, Text2CAD utilizes an LLM-based pipeline for data annotation, while CAD-Translator aligns texts and parametric CAD sequences via a Cascading Contrastive Strategy. Driven by the rapid evolution of LLMs, recent research transitions toward utilizing general-purpose models for CAD synthesis. CAD-Llama (Li et al., 2025c) proposes a hierarchical annotation pipeline that captures both structured information and detailed textual descriptions of 3D shapes, fine-tuning the model through pre-training and SFT stages. Similarly, Query2CAD (Badagabettu et al., 2024) harnesses proprietary LLMs, such as GPT-4 Turbo, to leverage CadQuery for generating and refining executable CAD macros. Furthermore, leveraging RL to enhance generation accuracy has garnered widespread attention. CADFusion (Wang et al., 2025j) optimizes NL-to-CAD generation by integrating textual and visual signals simultaneously, employing SFT on ground-truth parametric sequences and DPO on curated visual preference data. More recently, CAD-Coder (Guan et al., 2025) implements RL via GRPO, incorporating a strategic planning process to enforce both the syntactic correctness and geometric plausibility of the output CadQuery code. In parallel, to enhance the efficiency of data construction and VLM-based evaluation, CAD-Judge (Zhou et al., 2025) introduces a Compiler-as-a-Judge (CJM) strategy for constructing binary preference data

and a Compiler-as-a-Reviewer (CRM) strategy for test-time self-debugging. Following an two stage SFT and then KTO (Ethayarajh et al., 2024) training pipeline, it achieves superior performance on NL-to-CAD tasks while maintaining computational efficiency.

Expanding the scope to CAD-to-Code, approaches generate CAD commands from multimodal inputs like images and point clouds. Traditional methods generally utilize Transformer-based models for multimodal encoding and command decoding. For instance, CAD-SIGNet (Khan et al., 2024a) proposes Sketch instance Guided Attention (SGA) for command generation given input point clouds, while Img2CAD (Chen et al., 2025l) focuses on synthesizing commands from single images or sketches. Benefiting from the development of MLLMs, the field is evolving toward utilizing the powerful code generation capacity of MLLMs. Pioneeringly, CAD-MLLM (Xu et al., 2024a) introduces Omni-CAD, a large-scale dataset with 453k samples comprising constructive modeling sequences, textual descriptions, multi-view images, and point clouds. Building on this, it achieves superior performance in multimodal scenarios. CAD-Recode (Rukhovich et al., 2025) proposes to generate CAD sketch-extrude sequences in Python code using CadQuery rather than command sequences. With training a linear layer to encode point clouds, it constructs an MLLM specifically tailored for CAD reverse engineering. CAD-RL (Niu et al., 2025c) proposes a multimodal CoT-guided RL framework, employing CoT-based SFT for the cold-start phase and RL for post-training optimization. The study further demonstrates that incorporating reference images and CoT reasoning significantly enhances the fidelity of the generated CadQuery code. Similarly, ReCAD (Li et al., 2025b) adopts a comparable SFT-RL training paradigm, introducing Hierarchical Primitive Learning (HPL) as a curriculum strategy specifically tailored to the complexities of CAD generation. Moving beyond direct generation, CAD-Assistant (Mallis et al., 2025) pioneers a tool-augmented framework for generic task solving. By ingesting multimodal inputs such as hand-drawn sketches and 3D scans, it leverages CAD-specific tools to synthesize the corresponding code iteratively. Also, addressing the need for automated correction, CADReview (Chen et al., 2025c) curates a dataset comprising over 20k program-image pairs and proposes ReCAD, a program repair framework designed to rectify erroneous CAD scripts. Specifically, ReCAD utilizes SFT and DPO to train a feedback generator that discerns discrepancies between rendered and reference images, subsequently guiding a code editor for precise code refinement.

Transcending the rigid definitions of traditional CAD, a growing body of research has focused on shape program generation. Pioneering works, such as DreamCoder (Ellis et al., 2021) and ShapeCoder (Jones et al., 2023), propose transforming visual structures into Domain-Specific Languages (DSLs) to represent 3D geometry in a structured and interpretable manner. Recently, Real2Code (Mandi et al., 2024) formulates the reconstruction and joint prediction of articulated objects as a code generation task. The pipeline first integrates a pre-trained DUSt3R model with a fine-tuned SAM to convert multi-view RGB images into segmented 3D point clouds. Subsequently, by employing a learned shape-completion network and a fine-tuned CodeLlama, these partial geometric inputs are translated into comprehensive code descriptions of the object. Similarly, MeshCoder (Dai et al., 2025) targets the synthesis of Blender Python scripts directly from point cloud inputs. It introduces a large-scale object-code dataset constructed via custom Blender APIs and trains a model to generate scripts that faithfully reconstruct the target 3D meshes.

# 6 Frontiers Frameworks

This section investigates frontier scenarios where code acts as a dynamic engine for active reasoning and interaction. We explore programmatic visual manipulation, video generation, and embodied control as evolving spatiotemporal and physical paradigms and further analyze visually grounded programming for real-world logic alignment and conclude with unified models pursuing general-purpose intelligence.

## 6.1 Programmatic Visual Manipulation

In the rapidly evolving domain of Thinking with Image, the integration of executable code into VLMs establishes a transformative paradigm known as programmatic image manipulation (Su et al., 2025). Unlike passive perception, this approach empowers models to actively interact with visual content through a symbolic intermediary. By treating code as an executable thought process, models perform precise geometric measurements and rigorous logical deductions, effectively mitigating hallucinations via deterministic execution. Since this paradigm utilizes code primarily as a reasoning engine for general Visual Question Answering (VQA) rather than as a final product, it lacks dedicated code generation benchmarks. Consequently, our review focuses exclusively on methodologies, categorizing the landscape into utilizing pre-defined tools and generative code reasoning.

### 6.1.1 Programmatic Visual Manipulation Methods

The integration of executable code into VLMs has fundamentally reshaped the landscape of multimodal research. Recent methodologies in this domain broadly fall into two primary paradigms: the use of predefined code tools, where models act as API controllers, and generative code reasoning to synthesise custom programs for open-ended problems.

The first paradigm leverages predefined code tools, in which VLMs invoke a closed toolset $\mathcal{T}$ to augment their visual reasoning beyond internal parametric knowledge. This methodology evolves from training-free mechanisms to fine-tuned, reinforcement-driven policies. Initial strategies predominantly rely on training-free mechanisms powered by In-Context Learning (ICL), where the model acts as a prompt-conditioned policy guided by system prompts. MM-React (Yang et al., 2023) pioneers this direction by integrating LLMs with vision experts (e.g., OCR, detection) via textual prompts to dispatch domain-specific modules. VipAct (Zhang et al., 2024c) advances this by employing a coordination policy to manage vision and captioning agents, achieving fine-grained perception through collaborative planning. Hydra (Ke et al., 2024) further optimizes this paradigm via an RL-based controller that dynamically selects optimal reasoning paths to rectify potential planning errors. To address the complexity of dynamic visual streams, DoraemonGPT (Yang et al., 2024g) incorporates a planner based on Monte Carlo Tree Search (MCTS) to strategically schedule spatial-temporal reasoning tools. Sharing this agentic focus, TraveLER (Shang et al., 2024) adopts a modular multi-agent framework where a planner iteratively directs agents to traverse, locate, and evaluate information across video frames. Distinctively, MoReVQA (Min et al., 2024) decouples the process by using external memory to separate video understanding into distinct parsing and reasoning stages. Two concurrent works, both named Video Agent (Fan et al., 2024; Wang et al., 2024c), also explore this agentic approach. The former (Fan et al., 2024) proposes a unified memory mechanism queried by tools, while the latter (Wang et al., 2024c) simulates human cognitive processes by iteratively identifying and compiling crucial video information using VLMs. Furthermore,

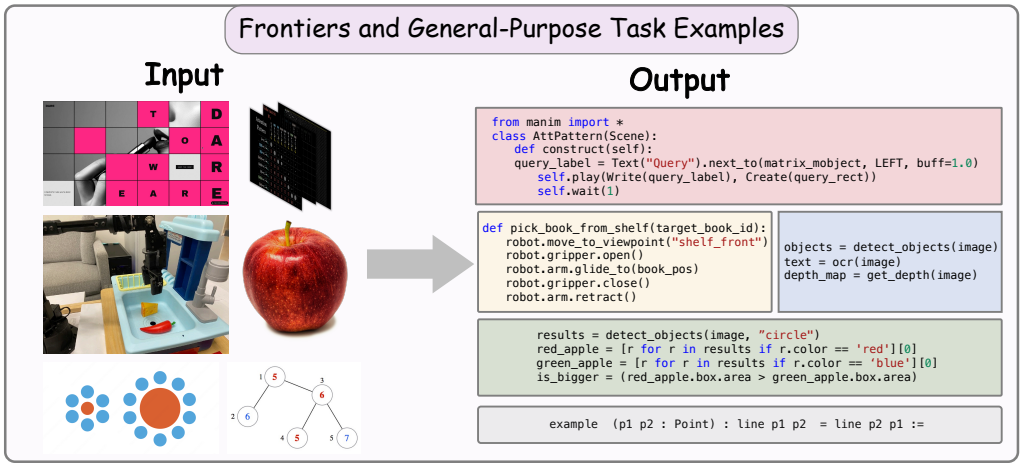


Figure 7: Tasks in Frontiers Frameworks section, including video scenarios, programmatic visual manipulation, embodied control, visually grounded programming and unified frameworks.

to enhance temporal precision, VTimeCoT (Zhang et al., 2025d) introduces visual tools for generating progress bars and highlighting key moments, allowing the VLM to visually perceive and reason about temporal progression. To improve reliability, subsequent research shifts focus towards SFT and agentic RL, explicitly training models to decompose reasoning processes into executable tool calls. This paradigm typically entails a multi-stage pipeline fine-tuned on explicit tool-use trajectories, equipping models with diverse utilities ranging from visual manipulations (e.g., cropping) to symbolic calculators. MLLM-Tool (Wang et al., 2025b) exemplifies this by incorporating instruction tuning to heighten sensitivity to multi-modal instructions, ensuring precise selection from extensive API pools. Moving beyond external calls, CogCoM (Qi et al., 2025) trains general VLMs to elicit intrinsic manipulations effectively internalizing the tool-use process without relying on external APIs. Adopting the RL, Pixel Reasoner (Wang et al., 2025e) employs curiosity-driven rewards to incentivize pixel-space operations. To scale the agentic capabilities, VISTA-R1 (Lu et al., 2025b) utilizes standardized environments to robustly interleave reasoning and tool execution, effectively mitigating the high reliance on costly synthetic or manual annotation.

The second paradigm of generative code reasoning represents a more advanced approach, treating the model as a Code-as-Policy generator. Instead of selecting from fixed tools, the model synthesizes custom executable programs to perform visual reasoning. Recent works in this field are witnessing a transition from inference-time composition to end-to-end verifiable training via RL. Pioneering this domain, VisProg (Gupta & Kembhavi, 2023) interprets textual instructions into executable sequences of visual modules (e.g., face detection, segmentation), facilitating compositional reasoning without task-specific training. ViperGPT (Surís et al., 2023) leverages code-generation models (e.g., Codex) to compose vision-and-language APIs into Python subroutines, uniquely enabling the execution of complex logic and arithmetic operations within visual queries. Similarly, CodeVQA (Subramanian et al., 2023) explicitly formulates visual question answering as a program synthesis problem, generating code that orchestrates visual primitives (like image-text matching) with conditional logic to derive answers. Moving beyond simple execution, previous frameworks leverage code to actively synthesize visual artifacts to aid reasoning. Visual Sketchpad (Hu et al.,

2024b) empowers VLMs to generate Python code that renders auxiliary sketches onto a dedicated visual pad to externalize intermediate reasoning steps. Extending this concept to active manipulation, ViLaSR (Wu et al., 2025b) introduces a drawing to reason paradigm optimized via reflective rejection sampling, where the model generates spatial indicators like bounding boxes to supplement visual details. Expanding into general-purpose settings, PyVision (Zhao et al., 2025a) empowers VLMs to autonomously synthesize and execute code utilizing diverse libraries for dynamic multi-turn analysis, circumventing the rigidity of fixed toolsets. ReFocus (Fu et al., 2025b) employs code to actively edit input images through highlighting or masking operations, thereby steering the model's attention during complex multi-hop reasoning.

Recently, the research frontier prioritizes internalizing agentic capabilities via SFT and RL from Execution Feedback (RLEF). SFT aligns models with code-based trajectories, while RLEF utilizes environmental signals (e.g., compilation success) to incentivize verifiable reasoning. Demonstrating the efficacy of SFT, Skywork-R1V4 (Zhang et al., 2025g) achieves robust visual intelligence by interleaving reasoning with active image manipulation and deep search, trained solely on high-quality planning-execution trajectories. Incorporating RL for further refinement, Visual-ARFT (Liu et al., 2025d) utilizes verifiable rewards to train agents in precise tasks like image manipulation and web searching. Advancing this, Thyme (Zhang et al., 2025f) employs a two-stage SFT-RL strategy to optimize decision-making for high-resolution perception, autonomously generating code for diverse image processing operations. Meanwhile, CodeVision (Guo et al., 2025c) proposes a code-as-tool framework utilizing dense process rewards to overcome the brittleness of fixed toolsets, enabling the invocation of arbitrary image operations. Tackling the issue of unfaithful tool usage, CodeV (Hou et al., 2025) introduces Tool-Aware Policy Optimization (TAPO), a process-level RL framework that explicitly rewards verifiable and evidence-consistent operations. In mathematical domains, CodePlot-CoT (Duan et al., 2025) trains an adaptive policy to generate executable plotting code for visual aids, dynamically switching between code-based and direct analysis. Similarly, for geometric reasoning, Geoint-R1 (Wei et al., 2025b) formalizes the process by constructing auxiliary elements via Lean4 code, utilizing a pipeline guided by verification reward models. Unlike fixed-tool approaches, this paradigm affords maximum versatility. The ability to generate executable code allows MLLMs to fabricate custom tools in real-time for unanticipated scenarios, effectively removing the ceiling on generalization imposed by static vocabularies.

### 6.2 Video Generation

The intersection of video and code has evolved from early efforts in extracting program logic from visual demonstrations (Sun et al., 2018) to a robust, bidirectional paradigm. This domain now encompasses two canonical tasks: Code-centric Video Generation and Video-to-Code Synthesis. The former leverages the deterministic nature of code to achieve precise spatiotemporal control, mitigating the inherent consistency challenges in pixel-space generation. The latter focuses on extracting executable policies or programmatic logic from dynamic visual streams, effectively bridging the gap between visual perception and physical execution. Recently, Kimi K2.5 (Kimi et al., 2026) has shown strong capacity in Video-to-Code, attracting widespread attention.

#### 6.2.1 Video Code Generation Benchmarks

To rigorously evaluate capabilities in these evolving domains, recent benchmarks have been established to target both educational synthesis and robotic policy extraction.

| Benchmarks | Domain | Test Instances | Key Features | Evaluation Metric |
|---|---|---|---|---|
| MMMC (Chen et al., 2025o) | Code-to-Video | 456 | Educational Content | TeachQuiz / Model-based |
| PresentEval (Shi et al., 2025) | Code-to-Video | 30 | Document-to-Presentation | Content Fidelity / User Study |
| Video2Code (Xie et al., 2025) | Video-to-Code | 115k | Robotic Policy Extraction | Success Rate / Model-based |

Table 6: Comparison with existing benchmarks in the video code generation domain.

Benchmarks in the Code-to-Video domain emphasize the utility and fidelity of synthesized content. MMMC (Chen et al., 2025o) serves as a pioneering benchmark for code-driven educational video generation, explicitly evaluating the educational efficacy of synthesized content via metrics such as TeachQuiz. In the domain of presentation generation, PresentEval (Shi et al., 2025) assesses the quality of document-to-video conversion, focusing on content fidelity and audience comprehension through Python and pre-defined HTML template evaluation.

Conversely, the Video-to-Code domain focuses on policy extraction from visual data. Video2Code (Xie et al., 2025) provides a large-scale dataset comprising 115k video-code-observation triplets. This benchmark is designed to advance the synthesis of robotic manipulation policies from uncontrolled video data, evaluating the model's ability to recover executable actions from visual observations. We provide details about current video code generation benchmarks in Table 6.

### 6.2.2 Video Code Generation Methods

Beyond the scope of evaluation protocols, substantial research focuses on developing methodologies to bridge the gap between static code and dynamic visual streams. Recent innovations in this domain primarily categorize into agent-based generative frameworks and reverse synthesis pipelines for policy extraction.

To mitigate the stochasticity inherent in pixel-level video generation, numerous agent-based frameworks have emerged to govern video structure through executable code. Code2Video (Chen et al., 2025o) introduces a collaborative multi-agent framework that synthesizes Python scripts to precisely control the logical flow and layout of educational videos. Similarly, PresentAgent (Shi et al., 2025) employs a modular pipeline to segment documents and synchronize generated visual assets with narration. Furthermore, Theorem ExplainAgent (Ku et al., 2025b) automates the retrieval and generation of explanatory animations for scientific theorems, validating the efficacy of code in visualizing complex scientific logic.

In parallel, the field is witnessing a strategic pivot toward extracting executable strategies from visual data and developing unified models. RoboPro (Xie et al., 2025) establishes a video-to-code pipeline, employing VLMs and Code LLMs to synthesize robotic manipulation policies. By leveraging large-scale video data, this approach effectively circumvents high data acquisition costs while enabling zero-shot instruction following. More recently, open-source initiatives are emerging to challenge the dominance of proprietary models. JanusCoder (Sun et al., 2025b) establishes a versatile visual-programmatic interface designed to bridge the gap between static vision-centric tasks and dynamic code-driven videos (*e.g.*, Manim animations). This signals a trend toward unified models capable of handling both generation and understanding tasks within a single framework.

### 6.3 Embodied Control

Embodied control aims to map high-level semantic instructions, conditioned on visual observations, into precise physical execution logic. Distinct from passive data analysis, these tasks necessitate active interaction with dynamic environments, requiring agents to interpret visual feedback and ground abstract linguistic concepts into concrete sensorimotor actions. Recent research paradigms have shifted from manually defined modular architectures toward programmatic planning and specialized model post-training, leveraging the reasoning and code-generation capabilities of VLMs.

#### 6.3.1 Embodied Control Benchmarks

The open-ended nature of embodied tasks necessitates standardized evaluation protocols, which have evolved from foundational simulation environments to rigorous code-centric control benchmarks.

Pioneering this domain, VirtualHome (Puig et al., 2018) establishes critical infrastructure by modeling complex household activities as executable programs, accompanied by a large-scale database designed for long-horizon interaction. Transitioning toward code-centric evaluation, Code as Policies (CaP) (Liang et al., 2022) introduces the RoboCodeGen benchmark. Comprising 37 tasks, it quantitatively assesses model proficiency in spatial-geometric reasoning and hierarchical control-flow synthesis. Further expanding the evaluation scope to multi-environment scenarios, OctoVerse (Yang et al., 2024b) provides a comprehensive benchmark spanning photorealistic interiors, Minecraft, and GTA-V, specifically tailored for vision-dependent function calls. Complementing these environments, the STEVE-21K (Zhao et al., 2024) offers 21k multimodal pairs, encompassing vision-environment data and skill-code triplets, to facilitate the development of agents capable of long-horizon interaction in open worlds. Collectively, these benchmarks serve as a vital bridge, connecting high-level language understanding with situated robotic performance.

#### 6.3.2 Embodied Control Methods

Beyond the scope of evaluation protocols, substantial research focuses on developing methods to bridge the semantic gap between high-level intent and low-level control. Recent innovations in this domain broadly categorize into two core streams: inference-time agent frameworks and learning-based optimization paradigms.

Driven by the zero-shot capabilities of LLMs, prominent works focus on building robust agent systems through programmatic planning without training the model. Firstly, ProgPrompt (Singh et al., 2023) pioneers this direction by introducing a programmatic prompt structure that incorporates API specifications and logic assertions, thereby enhancing the model's situated awareness and error-recovery capabilities. Complementing planning with reward engineering, VLM-CaR (Venuto et al., 2024) leverages VLMs to generate executable dense reward functions, automating the labor-intensive process of reward shaping. Integrating such logic-driven paradigms, STEVE (Zhao et al., 2024) represents a comprehensive embodied agent that decomposes high-level intent into granular guidelines to master complex sandbox environments. Most recently, to achieve finer physical control beyond high-level planning, EmbodiedCoder (Lin et al., 2025c) proposes a training-free framework that utilizes geometric parameterization to translate object point clouds into functional abstractions, directly synthesizing parameterized trajectories. Further extending the programmatic paradigm to complex manipulation, RoboScript (Chen et al., 2024a) generates structured Python

scripts for free-form robot tasks across simulation and real-world platforms, utilizing a hierarchical reasoning structure to explicitly handle object-centric spatial relationships.

While inference-time methods leverage the model's pre-trained capabilities, specialized post-training on robotic data has become indispensable for capturing physical dynamics. Initial efforts like PACT (Wei et al., 2023) explore representational learning through a perception-action causal transformer architecture, employing self-supervised training on sensorimotor sequences to implicitly capture robot dynamics. In contrast, RoboCodeX (Mu et al., 2024) focuses on multimodal post-training by curating a specialized reasoning dataset and employing an iterative self-updating SFT strategy. This approach empowers the model to predict physical constraints and motion preferences, facilitating a more effective translation of high-level intent into robot-specific behaviours compared to general-purpose multimodal models. Recently, the field has witnessed a transition toward RL for code refinement. For instance, Octopus (Yang et al., 2024b) introduces RL with Environmental Feedback (RLEF) strategy to elevate generation precision. By training the VLM on binary success-failure signals directly from the simulator using PPO, it allows the model to iteratively align its programmatic planning with the physical realities and constraints of the environment.

### 6.4 Visually Grounded Programming

Distinct from strictly converting visual layouts into code, visually grounded code generation involves synthesizing code conditioned on visual contexts that are inherently difficult to articulate through text. In such scenarios, visual inputs, encompassing elements from algorithmic diagrams to software execution screenshots, serve as critical grounding information that necessitates aligning code logic with visual constraints.

#### 6.4.1 Visually Grounded Programming Benchmarks

Current evaluation benchmarks can be broadly classified into two types based on task domain: code generation tasks, which focus on logical reasoning, and software engineering tasks derived from real-world repositories.

Code generation tasks focusing on logical reasoning primarily aim to solve algorithmic challenges grounded in visual contexts. MMCode (Li et al., 2024a) pioneers this direction by curating 3.5k competition problems from online judges, incorporating 6.6k images to illustrate problem states. However, recognizing that visual cues in MMCode are often supplementary, HumanEval-V (Zhang et al., 2024a) introduces 253 tasks where visual information is indispensable. By deliberately minimizing textual descriptions, it rigorously evaluates the model's dependency on visual reasoning. Beyond standard algorithms, specialized benchmarks broaden the scope of visual logic. ScratchEval (Fu et al., 2025a) leverages the block-based scratch language to assess logical and spatial perception, while TurtleBench (Rismanchian et al., 2025) tasks VLMs with utilizing the Python turtle library for graphic reproduction. Furthermore, Code-Vision (Wang et al., 2025d) introduces a reverse-engineering paradigm, tasking models with synthesizing executable programs directly from algorithmic and mathematical flowcharts.

The Software Engineering category consists of tasks derived from real-world repositories, evaluating agents in authentic development scenarios involving visual elements. While the seminal SWE-bench (Jimenez et al., 2024) establishes the standard for repository-based evaluation, it contains limited visual instances, accounting for approximately 5.6% of the dataset. Consequently, recent

| Benchmark | Test Instance | Domain | Language | Visual Assets |
|---|---|---|---|---|
| MMCode (Li et al., 2024a) | 3.5k | Code Generation | Python | 6.6k (Image) |
| HumanEval-V (Zhang et al., 2024a) | 253 | Code Generation | Python | 253 (Image) |
| Code-Vision (Wang et al., 2025d) | 338 | Code Generation | Python | 338 (Image) |
| ScratchEval (Fu et al., 2025a) | 305 | Multiple Choice | Scratch | 305 (Image) |
| TurtleBench (Rismanchian et al., 2025) | 260 | Visual Execution | Python | 260 (Image) |
| OmniGIRL (Guo et al., 2025b) | 959 | SWE | Multi | 34 (Image) |
| Visual SWE-bench (Zhang et al., 2025e) | 133 | SWE | Python | 217 (Image) + 2 (Video) |
| SWE-bench M (Yang et al., 2024e) | 617 | SWE | JavaScript | 221 (Image) + 70 (Video) |

Table 7: Summary of Benchmarks for Visually Grounded Programming Tasks. We feature them according to their task domains, output code language, and input visual assets. SWE is the abbreviation of software engineering.

extensions have focused on enriching the visual dimension of real-world issues. OmniGIRL (Guo et al., 2025b) broadens the scope with multi-lingual and multi-modal support, featuring tasks that utilize buggy code and runtime screenshots directly from actual projects. To guarantee visual dependency, CodeV (Zhang et al., 2025e) strictly filters repository issues, curating 133 tasks where visual cues are explicitly necessitated for resolution. Specializing in frontend development, SWE-bench MM (Yang et al., 2024e) compiles 617 tasks from 17 JavaScript repositories, covering diverse visual scenarios such as interactive mapping and web framework rendering. A comprehensive comparison of these benchmarks in Table 7.

### 6.4.2 Visually Grounded Programming Methods

Whether in code generation or software engineering, methodologies for enhancing vision-augmented programming prioritize the effective processing of visual information. To mitigate the prevalence of syntax errors and unexecutability in code directly generated by VLMs, a predominant strategy involves an intermediate modality conversion: transforming images into textual descriptions or captions before feeding them into the model alongside the problem statement. For instance, Code-Vision (Wang et al., 2025d) demonstrates that converting flowcharts into Mermaid code before target code generation effectively suppresses compilation errors. Similarly, constrained by the disparate coding proficiencies of VLMs, HumanEval-V (Zhang et al., 2024a) adopts a two-stage strategy: leveraging a VLM for initial textual description generation, followed by a specialized Code LLM for final synthesis, thereby significantly boosting performance. CodeV (Zhang et al., 2025e) refines this by converting visual inputs into fine-grained descriptions and structured summaries, effectively reducing complex multimodal challenges to text-based tasks where LLMs excel.

Most advancements in software engineering now focus on equipping agents with visual perception, such as SWE-agent (Yang et al., 2024c), Agentless (Xia et al., 2024), and AutoCodeRover (Zhang et al., 2024b). While SWE-agent (Yang et al., 2024d) encapsulates Shell commands to enable terminal operations, SWE-agent M (Yang et al., 2024e) extends this interface with web browsing, screenshotting, and image viewing capabilities, empowering the system to visually reproduce issues and verify fixes. Agentless (Xia et al., 2024) employs a hierarchical localization strategy, allowing the LLM to sequentially narrow down the scope from files to classes, methods, and finally specific lines of code, passing this context to a secondary LLM for patch generation. Methods for enhancing visually grounded programming primarily focus on bridging the gap between visual understanding and code generation, encompassing and the other to iterative repair attempts. Additionally, GUIRepair (Huang et al., 2025) proposes a closed-loop repair process driven by visual feedback, comprising

an Image2Code module for generating reproduction scripts from issues and a Code2Image module for capturing screenshots of executed fixes. By contrasting generated screenshots with original issue images, it provides critical feedback to guide the repair process.

In summary, the primary bottleneck in vision-augmented code generation stems from the significant performance asymmetry between the visual comprehension and code synthesis capabilities of VLMs. Consequently, SOTA methods favor an indirect strategy: either leveraging VLMs to translate visual inputs into textual representations for specialized Code LLMs, or incorporating visual feedback loops within agentic workflows to mitigate the limitations of standalone models.

### 6.5 Unified Multimodal Code Generation

Although multimodal code generation has been widely explored across various domains, previous studies generally treat each task type and domain as an isolated problem, an approach inconsistent with the comprehensive capabilities inherent in LLMs. Aiming to achieve generalist capabilities, recent studies are exploring unified multimodal code generation by consolidating multiple tasks and domains within a single LLM/VLM. While previous sections introduced works focusing on multiple tasks within individual domains, this section centers on approaches that unify these diverse domains.

#### 6.5.1 Unified Multimodal Code Generation Benchmarks

Beyond previous works in single-domain multimodal code generation, unified benchmarks have evolved to cover diverse modalities and task formulations. These benchmarks evaluate capabilities ranging from NL-to-code to image-to-code and editing tasks. Pioneering the realm of image-to-code tasks, Image2Struct (Roberts et al., 2024) assesses the capability of VLMs to extract structured code, such as LaTeX and HTML, from visual images across diverse domains like webpages, mathematical formulas, and musical scores. To ensure robust evaluation of visual fidelity, Image2Struct introduces novel metrics including Cosine Inception Similarity (CIS) and Earth Mover Similarity (EMS). Similarly, CoSyn (Yang et al., 2025g) adopts a code-first paradigm, leveraging LLMs to synthesize rendering code across 9 image categories prior to constructing QA pairs. Beyond the domain of image understanding, many works utilize its evaluation set to assess multimodal code generation capabilities. To address the gap between code generation and visual interaction, ArtifactsBench (Zhang et al., 2025a) assesses models on dynamic, visual-interactive artifacts. It comprises 1.8k queries across nine domains. Furthermore, the work proposes a multimodal evaluation pipeline utilizing a checklist-guided MLLM-as-Judge to strictly verify the executability and interactive logic of the generated applications. Recently, InfiBench-V (Jiang et al., 2025a) targets real-world applicability with a set of 322 visually rich questions where images are indispensable for reasoning. Covering 13 programming languages, the benchmark maps tasks to five distinct domains: Front-end, Back-end, Data Science & Machine Learning, Mobile & Desktop Development, and IT Operations. In the domain of NL-to-Visualization generation, VisPlotBench (Ni et al., 2025a) establishes a unified benchmark for visualization coding agents, comprising 888 tasks across eight programming languages. Furthermore, the framework incorporates a multi-round self-debug protocol to evaluate the capability for iterative code refinement.

### 6.5.2 Unified Multimodal Code Generation Methods

Driven by advancing model capabilities, research is shifting from task-specific solutions to general-purpose approaches. To address the absence of foundation models for multimodal code and tackle complex, multimodal software environments (Comanici et al., 2025), the development of general-purpose multimodal code models has attracted widespread attention.

Early foundational works establish the groundwork for cross-domain capabilities. For instance, GOT (Wei et al., 2024) expands the traditional OCR scope to encompass chemical, chart, and geometric scenarios through a three-stage training paradigm. Complementing these training efforts, BigDoc (Rodriguez et al., 2024) introduces a large-scale open-source dataset specifically curated for multimodal document and code-centric tasks. Recent advancements focus on broadening task coverage and input flexibility. VisCoder2 (Ni et al., 2025a) introduces the VisCode-Multi-679K dataset, which comprises 613k NL-to-visualisation and 66k code-correction samples to optimize both code generation and correction capabilities. However, it only considers textual input, which remains constrained to unimodal scenarios. For multimodal input scenarios, VisCodex (Jiang et al., 2025a) proposes a model-merging strategy that integrates a Code LLM with a VLM backbone. To facilitate training, it further introduces MCD-598k, a comprehensive dataset that covers web-to-code, chart-to-code, image-augmented code QA, and algorithmic code tasks. JanusCoder (Sun et al., 2025b) further advances unified multimodal code generation by integrating both text- and vision-centric tasks. To support this, it introduces the JanusCode-800K dataset, which spans 11 distinct task categories. The frontier of this field centres on RL for further optimization, VinciCoder (Zhao et al., 2025b) introduces a coarse-to-fine mechanism for image-to-code generation, formulating a reward signal based on the visual similarity between input images and rendered outputs. Through the integration of large-scale SFT with this specific RL strategy, it demonstrates superior performance across various image-to-code benchmarks. Furthermore, OCRVerse (Zhong et al., 2026) unifies OCR and programmatic tasks within end-to-end VLMs by employing a holistic OCR framework, utilizing decoupled textual and visual reward signals for precise RL optimization.

However, the performance of current unified methods lags behind leading commercial models, primarily due to an over-reliance on high-quality data and limited generalizability across tasks. Future research may prioritize the development of data-efficient and generalizable training paradigms to bridge this gap.

## 7 Future Directions

### 7.1 The Paradigm Shift: From SFT to RL

A pivotal evolution in the Multimodal Code Intelligence landscape is the strategic transition from SFT to RL. Although SFT establishes robust baselines using large-scale datasets like Web2Code (Yun et al., 2024), ChartCoder (Zhao et al., 2025d), and DeepCAD (Wu et al., 2021), it inevitably encounters performance limitations. Specifically, SFT alone is often insufficient for optimizing the execution correctness and visual fidelity of the generated code. To mitigate these limitations, various RL approaches have emerged. Early methods, such as DualDPO (Zhang et al., 2025h) and CADFusion (Chai et al., 2025b), utilize DPO to align the model more precisely with ground-truth code distributions. More recently, leveraging advancements like GRPO, works such as MSRL (Chen et al., 2025e) and RLRF (Rodriguez et al., 2025) have designed fine-grained reward

mechanisms that incorporate visual signals to optimize visual fidelity in the chart and SVG domains. Through these advanced RL strategies, these approaches significantly improve the model's ability to generate code that is both executable and visually accurate. However, unlike conventional code generation, which can rely on deterministic unit testing for rewards, multimodal code generation lacks precise automated metrics for evaluating visual output. Consequently, designing reliable reward functions and robust training strategies for RL remains a critical direction for future research.

### 7.2 Expanding the Scope: From Task-Specific to Unified

In the preceding sections of this survey, we review multimodal code generation based on specific tasks and domains. However, these task-specific approaches generally focus on individual task improvements, which contrasts with the generalist capabilities of LLMs/VLMs. Driven by the evolution of foundation models and the growth of open-source data, the field is shifting toward unified multimodal code generation. Despite progress in specialized areas, achieving true unification remains challenging, as current methods often treat modalities in isolation and rely heavily on large-scale SFT data. Recent works such as JanusCoder (Sun et al., 2025b) and VisCoder2 (Ni et al., 2025a) represent initial steps toward unification, yet they still rely predominantly on SFT. While VinciCoder (Zhao et al., 2025b) attempts to integrate vision-based RL, its strategy is currently limited to simple image-to-code tasks. We argue that future research should explore more robust and generalizable RL paradigms capable of enhancing diverse multimodal code tasks, extending from image editing to the full spectrum of multimodal code generation.

### 7.3 Expanding the Horizon: From Static Synthesis to Dynamic Interaction

Beyond the evolution of training paradigms, the field of Multimodal Code Intelligence experiences a fundamental transition in problem complexity. The focus shifts from the synthesis of isolated static fragments to the construction of complex dynamic systems rooted in real-world scenarios. For example, in the Web-to-Code domain, early research primarily focuses on static inputs. These methods translate single screenshots into markup code to replicate visual appearance. Benchmarks such as WebSight (Laurençon et al., 2024) and Design2Code (Si et al., 2025) exemplify this task format. However, real-world software engineering necessitates much higher structural complexity. Code must govern user interactions, state management, and multi-page routing. Consequently, recent initiatives move toward dynamic and comprehensive complexity. For instance, Interaction2Code (Xiao et al., 2025a) moves beyond static rendering to evaluate functional interactivity. Similarly, MRWeb (Wan et al., 2024) challenges models to synthesize resource-aware multi-page applications. The emergence of Video-to-Code generation further accelerates this expansion. This field extends the input dimension from spatial snapshots to complex spatiotemporal streams. As demonstrated by datasets such as Video2Code (Xie et al., 2025), the focus shifts toward extracting executable policies and physical manipulation logic directly from continuous video data. Notably, recent commercial models like Kimi K2.5 (Kimi et al., 2026) demonstrate strong capabilities in this domain. This evolution marks a decisive departure from passive visual translation toward active grounded reasoning in complex environments. These advancements are essential for developing embodied agents capable of mastering time-dependent and structurally intricate workflows.

### 7.4 Empowering Agency: From Chat Models to Autonomous Agents

Looking ahead, by synthesizing the future directions discussed above, we argue that the optimization of Multimodal Code Intelligence will extend beyond single chat generation and standard RL toward advanced multi-turn interaction and Agentic RL (Zhang et al., 2025b). As visual inputs and instructional tasks become increasingly complex, they often involve video data and multiple editing instructions. In these scenarios, single-turn generation is insufficient to handle the requirements. The future of this field likely lies in developing models capable of iterative debugging, refinement, and utilizing diverse code tools (Qu et al., 2025) to address practical real-world applications. By integrating these agentic workflows with RL, future systems are poised to evolve from simple code generators into autonomous visual software engineers capable of reasoning, planning, and adapting in open-ended environments.

## 8 Conclusion

In this paper, we conduct a comprehensive survey of Multimodal Code Intelligence, systematically organizing the landscape into four primary domains: Graphical User Interfaces, Scientific Visualization, Structured Graphics, and Frontiers Frameworks. We provide an extensive review of existing benchmarks and methodologies, analyzing how models translate visual perception into diverse executable representations. Specifically, our survey covers a wide range of multimodal code generation tasks, such as web layouts, scientific plots, industrial designs, and videos. Crucially, this work addresses a significant gap in the current literature regarding the utilization of code as a universal interface for general-purpose visual tasks. In this paradigm, executable code serves as a versatile intermediate medium that enables models to ground visual reasoning, invoke external tools, and precisely solve open-ended problems in dynamic environments. While text-based program synthesis is mature, this approach of leveraging code for visual problem-solving remains fragmented. Our analysis highlights the transformative potential of moving beyond specialized, task-specific models to general-purpose agents that utilize code for comprehensive multimodal interaction.

We believe that this detailed consolidation of resources and paradigms facilitates the advancement of this field. By establishing a clear taxonomy and standardizing the evaluation landscape, this work serves as a foundational roadmap, accelerating the community's transition toward developing more robust and capable multimodal coding agents.

## References

Anthropic. Claude 4.5 Sonnet System Card. `https://www.anthropic.com/research/claude-4-5-sonnet-system-card`, 2025. Accessed: 2026-02-09.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Akshay Badagabettu, Sai Sravan Yarlagadda, and Amir Barati Farimani. Query2cad: Generating cad models using natural language queries. *arXiv preprint arXiv:2406.00144*, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang,

Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025a. URL `https://arxiv.org/abs/2511.21631`.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Averi Bates, Ryan Vavricka, Shane Carleton, Ruosi Shao, and Chongle Pan. Unified modeling language code generation from diagram images using multimodal large language models. *Machine Learning with Applications*, pp. 100660, 2025.

Patrice Bechard, Chao Wang, Amirhossein Abaskohi, Juan Rodriguez, Christopher Pal, David Vazquez, Spandana Gella, Sai Rajeswar, and Perouz Taslakian. Starflow: Generating structured workflow outputs from sketch images. *arXiv preprint arXiv:2503.21889*, 2025.

Jonas Belouadi, Anne Lauscher, and Steffen Eger. Automatikz: Text-guided synthesis of scientific vector graphics with tikz. *arXiv preprint arXiv:2310.00367*, 2023.

Lukas Blecher. pix2tex - latex ocr. `https://github.com/lukas-blecher/LaTeX-OCR`, 2022. Accessed: 2025-12-10.

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361, 2020.

Linzheng Chai, Jian Yang, Shukai Liu, Wei Zhang, Liran Wang, Ke Jin, Tao Sun, Congnan Liu, Chenchen Zhang, Hualei Zhu, et al. Multilingual multimodal software developer for code generation. *arXiv preprint arXiv:2507.08719*, 2025a.

Mingxu Chai, Ziyu Shen, Chong Zhang, Yue Zhang, Xiao Wang, Shihan Dou, Jihua Kang, Jiazheng Zhang, and Qi Zhang. Docfusion: a unified framework for document parsing tasks. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7584–7599, 2025b.

Apu Kumar Chakroborti, Yi Ding, and Lipeng Wan. Toward automated and trustworthy scientific analysis and visualization with llm-generated code. *arXiv preprint arXiv:2511.21920*, 2025.

Kaiyan Chang, Zhirong Chen, Yunhao Zhou, Wenlong Zhu, Kun Wang, Haobo Xu, Cangyuan Li, Mengdi Wang, Shengwen Liang, Huawei Li, et al. Natural language is not enough: Benchmarking multi-modal generative ai for verilog generation. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2024.

Ada Chen, Yongjiang Wu, Junyuan Zhang, Jingyu Xiao, Shu Yang, Jen-tse Huang, Kun Wang, Wenxuan Wang, and Shuai Wang. A survey on the safety and security threats of computer-using agents: Jarvis or ultron? *arXiv preprint arXiv:2505.10924*, 2025a.

Hanqi Chen, Zhongyin Zhao, Ye Chen, Zhujin Liang, and Bingbing Ni. Svgthinker: Instruction-aligned and reasoning-driven text-to-svg generation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 11004–11012, 2025b.

Jiali Chen, Xusen Hei, HongFei Liu, Yuancheng Wei, Zikun Deng, Jiayuan Xie, Yi Cai, and Li Qing. Cadreview: Automatically reviewing cad programs with error detection and correction. *arXiv preprint arXiv:2505.22304*, 2025c.

Jiaqi Chen, Yanzhe Zhang, Yutong Zhang, Yijia Shao, and Diyi Yang. Generative interfaces for language models. *arXiv preprint arXiv:2508.19227*, 2025d.

Junting Chen, Yao Mu, Qiaojun Yu, Tianming Wei, Silang Wu, Zhecheng Yuan, Zhixuan Liang, Chao Yang, Kaipeng Zhang, Wenqi Shao, et al. Roboscript: Code generation for free-form manipulation tasks across real and simulation. *arXiv preprint arXiv:2402.14623*, 2024a.

Lei Chen, Xuanle Zhao, Zhixiong Zeng, Jing Huang, Liming Zheng, Yufeng Zhong, and Lin Ma. Breaking the sft plateau: Multimodal structured reinforcement learning for chart-to-code generation. *arXiv preprint arXiv:2508.13587*, 2025e.

Lei Chen, Xuanle Zhao, Zhixiong Zeng, Jing Huang, Yufeng Zhong, and Lin Ma. Chart-r1: Chain-of-thought supervision and reinforcement for advanced chart reasoner. *arXiv preprint arXiv:2507.15509*, 2025f.

Liangyu Chen, Yichen Xu, Jianzhe Ma, Yuqi Liu, Donglu Yang, Liang Zhang, Wenxuan Wang, and Qin Jin. Charteditor: A reinforcement learning framework for robust chart editing. *arXiv preprint arXiv:2511.15266*, 2025g.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL `https://arxiv.org/abs/2107.03374`.

Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. Viseval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*, 2024b.

Qiaosheng Chen, Yang Liu, Lei Li, Kai Chen, Qipeng Guo, Gong Cheng, and Fei Yuan. Interactscience: Programmatic and visually-grounded evaluation of interactive scientific demonstration code generation. *arXiv preprint arXiv:2510.09724*, 2025h.

Qiguang Chen, Mingda Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, et al. Ai4research: A survey of artificial intelligence for scientific research. *arXiv preprint arXiv:2507.01903*, 2025i.

Siqi Chen, Xinyu Dong, Haolei Xu, Xingyu Wu, Fei Tang, Hang Zhang, Yuchen Yan, Linjuan Wu, Wenqi Zhang, Guiyang Hou, et al. Svgenius: Benchmarking llms in svg understanding, editing and generation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 13289–13296, 2025j.

Song Chen, Xinyu Guo, Yadong Li, Tao Zhang, Mingan Lin, Dongdong Kuang, Youwei Zhang, Lingfeng Ming, Fengyu Zhang, Yuran Wang, et al. Ocean-ocr: Towards general ocr application via a vision-language model. *arXiv preprint arXiv:2501.15558*, 2025k.

Tianrun Chen, Chunan Yu, Yuanqi Hu, Jing Li, Tao Xu, Runlong Cao, Lanyun Zhu, Ying Zang, Yong Zhang, Zejian Li, et al. Img2cad: Conditioned 3-d cad model generation from single image with structured visual geometry. *IEEE Transactions on Industrial Informatics*, 2025l.

Xiangyang Chen, Shuzhao Li, Xiuwen Zhu, Yongfan Chen, Fan Yang, Cheng Fang, Lin Qu, Xiaoxiao Xu, Hu Wei, and Minggang Wu. Logics-parsing technical report. *arXiv preprint arXiv:2509.19760*, 2025m.

Yang Chen, Minghao Liu, Yufan Shen, Yunwen Li, Tianyuan Huang, Xinyu Fang, Tianyu Zheng, Wenxuan Huang, Cheng Yang, Daocheng Fu, et al. Iwr-bench: Can lvlms reconstruct interactive webpage from a user interaction video? *arXiv preprint arXiv:2509.24709*, 2025n.

Yanzhe Chen, Kevin Qinghong Lin, and Mike Zheng Shou. Code2video: A code-centric paradigm for educational video generation, 2025o. URL `https://arxiv.org/abs/2510.01174`.

Yunnong Chen, Shixian Ding, YingYing Zhang, Wenkai Chen, Jinzhou Du, Lingyun Sun, and Liuqing Chen. Designcoder: Hierarchy-aware and self-correcting ui code generation with large language models. *arXiv preprint arXiv:2506.13663*, 2025p.

Zehao Chen and Rong Pan. Svgbuilder: Component-based colored svg generation with text-guided autoregressive transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 2358–2366, 2025.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery, 2025q. URL `https://arxiv.org/abs/2410.05080`.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, et al. Paddleocr-vl: Boosting multilingual document parsing via a 0.9 b ultra-compact vision-language model. *arXiv preprint arXiv:2510.14528*, 2025a.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, et al. Paddleocr 3.0 technical report. *arXiv preprint arXiv:2507.05595*, 2025b.

Zhiqing Cui, Jiahao Yuan, Hanqing Wang, Yanshu Li, Chenxu Du, and Zhenglong Ding. Draw with thought: Unleashing multimodal reasoning for scientific diagram generation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 5050–5059, 2025c.

Bingquan Dai, Li Ray Luo, Qihong Tang, Jie Wang, Xinyu Lian, Hao Xu, Minghan Qin, Xudong Xu, Bo Dai, Haoqian Wang, et al. Meshcoder: Llm-powered structured mesh code generation from point clouds. *arXiv preprint arXiv:2508.14879*, 2025.

Truong Hai Dang, Jingyu Xiao, and Yintong Huo. Envisioning future interactive web development: Editing webpage with natural language. In *2025 2nd IEEE/ACM International Conference on AI-powered Software (AIware)*, pp. 61–66. IEEE, 2025.

Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pp. 845–854, 2017.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. Image-to-markup generation with coarse-to-fine attention. In *International Conference on Machine Learning*, pp. 980–989. PMLR, 2017.

Zijian Ding, Qinshi Zhang, Mohan Chi, and Ziyi Wang. Frontend diffusion: Empowering self-representation of junior researchers and designers through agentic workflows. *arXiv preprint arXiv:2502.03788*, 2025.

Chengqi Duan, Kaiyue Sun, Rongyao Fang, Manyuan Zhang, Yan Feng, Ying Luo, Yufang Liu, Ke Wang, Peng Pei, Xunliang Cai, Hongsheng Li, Yi Ma, and Xihui Liu. Codeplot-cot: Mathematical visual reasoning by thinking with code-driven images, 2025. URL `https://arxiv.org/abs/2510.11718`.

Peitong Duan, Chin-Yi Cheng, Gang Li, Bjoern Hartmann, and Yang Li. Uicrit: Enhancing automated design evaluation with a ui critique dataset. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–17, 2024.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, pp. 835–850, 2021.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. Videoagent: A memory-augmented multimodal agent for video understanding. In *European Conference on Computer Vision*, pp. 75–92. Springer, 2024.

Hao Feng, Shu Wei, Xiang Fei, Wei Shi, Yingdong Han, Lei Liao, Jinghui Lu, Binghong Wu, Qi Liu, Chunhui Lin, et al. Dolphin: Document image parsing via heterogeneous anchor prompting. *arXiv preprint arXiv:2505.14059*, 2025.

Rao Fu, Ziyang Luo, Hongzhan Lin, Zhen Ye, and Jing Ma. Scratcheval: Are gpt-4o smarter than my child? evaluating large multimodal models with visual programming challenges. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 689–699, 2025a.

Xingyu Fu, Minqian Liu, Zhengyuan Yang, John Corring, Yijuan Lu, Jianwei Yang, Dan Roth, Dinei Florencio, and Cha Zhang. Refocus: Visual editing as a chain of thought for structured image understanding. *arXiv preprint arXiv:2501.05452*, 2025b.

Timur Galimzyanov, Sergey Titov, Yaroslav Golubev, and Egor Bogomolov. Drawing pandas: A benchmark for llms in generating plotting code. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*, pp. 503–507. IEEE, 2025.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Jiaxin Ge, Zora Zhiruo Wang, Xuhui Zhou, Yi-Hao Peng, Sanjay Subramanian, Qinyue Tan, Maarten Sap, Alane Suhr, Daniel Fried, Graham Neubig, et al. Autopresent: Designing structured visuals from scratch. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 2902–2911, 2025.

Google. Gemini 3 Pro Technical Report. `https://deepmind.google/models/gemini/pro/`, 2025. Accessed: 2026-02-09.

Tongkun Guan, Chengyu Lin, Wei Shen, and Xiaokang Yang. Posformer: recognizing complex handwritten mathematical expression with position forest transformer. In *European Conference on Computer Vision*, pp. 130–147. Springer, 2024.

Yandong Guan, Xilin Wang, Ximing Xing, Jing Zhang, Dong Xu, and Qian Yu. Cad-coder: Text-to-cad generation with chain-of-thought and geometric reward. *arXiv preprint arXiv:2505.19713*, 2025.

Yi Gui, Zhen Li, Yao Wan, Yemin Shi, Hongyu Zhang, Yi Su, Shaoling Dong, Xing Zhou, and Wenbin Jiang. Vision2ui: A real-world dataset with layout for code generation from ui designs. *CoRR*, 2024.

Yi Gui, Zhen Li, Yao Wan, Yemin Shi, Hongyu Zhang, Bohua Chen, Yi Su, Dongping Chen, Siyuan Wu, Xing Zhou, et al. Webcode2m: A real-world dataset for code generation from webpage designs. In *Proceedings of the ACM on Web Conference (WWW 2025)*, pp. 1834–1845, 2025.

Hongcheng Guo, Wei Zhang, Junhao Chen, Yaonan Gu, Jian Yang, Junjia Du, Shaosheng Cao, Binyuan Hui, Tianyu Liu, Jianxin Ma, et al. Iw-bench: Evaluating large multimodal models for converting image-to-web. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6449–6466, 2025a.

Lianghong Guo, Wei Tao, Runhan Jiang, Yanlin Wang, Jiachi Chen, Xilin Liu, Yuchi Ma, Mingzhi Mao, Hongyu Zhang, and Zibin Zheng. Omnigirl: A multilingual and multimodal benchmark for github issue resolution. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):24–46, 2025b.

Zirun Guo, Minjie Hong, Feng Zhang, Kai Jia, and Tao Jin. Thinking with programming vision: Towards a unified view for thinking with images. *arXiv preprint arXiv:2512.03746*, 2025c.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14953–14962, 2023.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*, 2023.

Mengliang He, Jiayi Zeng, Yankai Jiang, Wei Zhang, Zeming Liu, Xiaoming Shi, and Aimin Zhou. Flow2code: Evaluating large language models for flowchart-based code generation capability. *arXiv preprint arXiv:2506.02073*, 2025.

Wei He, Zhiheng Xi, Wanxu Zhao, Xiaoran Fan, Yiwen Ding, Zifei Shan, Tao Gui, Qi Zhang, and Xuanjing Huang. Distill visual chart reasoning ability from llms to mllms. *arXiv preprint arXiv:2410.18798*, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pp. 2366–2369. IEEE, 2010.

Xinhai Hou, Shaoyuan Xu, Manan Biyani, Mayan Li, Jia Liu, Todd C Hollon, and Bryan Wang. Codev: Code with images for faithful visual reasoning via tool-aware policy optimization. *arXiv preprint arXiv:2511.19661*, 2025.

Teng Hu, Ran Yi, Baihong Qian, Jiangning Zhang, Paul L Rosin, and Yu-Kun Lai. Supersvg: Superpixel-based scalable vector graphics synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24892–24901, 2024a.

Yushi Hu, Otilia Stretcu, Chun-Ta Lu, Krishnamurthy Viswanathan, Kenji Hata, Enming Luo, Ranjay Krishna, and Ariel Fuxman. Visual program distillation: Distilling tools and programmatic reasoning into vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9590–9601, 2024b.

Kai Huang, Jian Zhang, Xiaofei Xie, and Chunyang Chen. Seeing is fixing: Cross-modal reasoning with multimodal llms for visual software issue fixing. *arXiv preprint arXiv:2506.16136*, 2025.

Kung-Hsiang Huang, Hou Pong Chan, May Fung, Haoyi Qiu, Mingyang Zhou, Shafiq Joty, Shih-Fu Chang, and Heng Ji. From pixels to insights: A survey on automatic chart understanding in the era of large foundation models. *IEEE Transactions on Knowledge and Data Engineering*, 37(5): 2550–2568, 2024.

Yongshuai Huang, Ning Lu, Dapeng Chen, Yibo Li, Zecheng Xie, Shenggao Zhu, Liangcai Gao, and Wei Peng. Improving table structure recognition with visual-alignment sequential coordinate modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11134–11143, 2023.

Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1911–1920, 2023.

Akriti Jain, Pritika Ramu, Aparna Garimella, and Apoorv Saxena. Doc2chart: Intent-driven zero-shot chart generation from documents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 34936–34951, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Vanita Jain, Piyush Agrawal, Subham Banga, Rishabh Kapoor, and Shashwat Gulyani. Sketch2code: transformation of sketches to ui in real-time using deep neural network. *arXiv preprint arXiv:1910.08930*, 2019.

Daeheon Jeong, Seoyeon Byun, Kihoon Son, Dae Hyun Kim, and Juho Kim. Canvas: A benchmark for vision-language models on tool-based user interface design. *arXiv preprint arXiv:2511.20737*, 2025.

Haonian Ji, Shi Qiu, Siyang Xin, Siwei Han, Zhaorun Chen, Dake Zhang, Hongyi Wang, and Huaxiu Yao. From eduvisbench to eduvisagent: A benchmark and multi-agent framework for reasoning-driven pedagogical visualization. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025.

Caijun Jia, Nan Xu, Jingxuan Wei, Qingli Wang, Lei Wang, Bihui Yu, and Junnan Zhu. Chartreasoner: Code-driven modality bridging for long-chain reasoning in chart question answering. *arXiv preprint arXiv:2506.10116*, 2025.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 2024.

Lingjie Jiang, Shaohan Huang, Xun Wu, Yixia Li, Dongdong Zhang, and Furu Wei. Viscodex: Unified multimodal code generation via merging vision and coding models. *arXiv preprint arXiv:2508.09945*, 2025a.

Nan Jiang, Shanchao Liang, Chengxiao Wang, Jiannan Wang, and Lin Tan. Latte: Improving latex recognition for tables and formulae with iterative refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 4030–4038, 2025b.

Rui Jiang, Shenrong Wu, Zhehao Wu, Zhenjie Han, and Jiang Zhong. Chartgen-agent: A three-stage framework for automated high-quality chart generation. In *International Conference on Advanced Data Mining and Applications*, pp. 19–32. Springer, 2025c.

Yilei Jiang, Yaozhi Zheng, Yuxuan Wan, Jiaming Han, Qunzhong Wang, Michael R Lyu, and Xiangyu Yue. Screencoder: Advancing visual-to-code generation for front-end automation via modular multimodal agents. *arXiv preprint arXiv:2507.22827*, 2025d.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

R Kenny Jones, Paul Guerrero, Niloy J Mitra, and Daniel Ritchie. Shapecoder: Discovering abstractions for visual programs from unstructured primitives. *ACM Transactions on Graphics (TOG)*, 42(4):1–17, 2023.

Kyudan Jung, Hojun Cho, Jooyeol Yun, Soyoung Yang, Jaehyeok Jang, and Jaegul Choo. Talk to your slides: Language-driven agents for efficient slide editing. *arXiv preprint arXiv:2505.11604*, 2025.

Fucai Ke, Zhixi Cai, Simindokht Jahangard, Weiqing Wang, Pari Delir Haghighi, and Hamid Rezatofighi. Hydra: A hyper agent for dynamic compositional visual reasoning. In *European Conference on Computer Vision*, pp. 132–149. Springer, 2024.

Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-signet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4713–4722, 2024a.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential cad models from beginner-to-expert level text prompts, 2024b. URL https://arxiv.org/abs/2409.17106.

Kimi, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Kristian Kolthoff, Felix Kretzer, Lennart Fiebig, Christian Bartelt, Alexander Maedche, and Simone Paolo Ponzetto. Zero-shot prompting approaches for llm-based graphical user interface generation. *arXiv preprint arXiv:2412.11328*, 2024.

Max Ku, Cheuk Hei Chong, Jonathan Leung, Krish Shah, Alvin Yu, and Wenhu Chen. Theoremexplainagent: Towards video-based multimodal explanations for LLM theorem understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025*, pp. 6663–6684. Association for Computational Linguistics, 2025a. URL https://aclanthology.org/2025.acl-long.332/.

Max Ku, Cheuk Hei Chong, Jonathan Leung, Krish Shah, Alvin Yu, and Wenhu Chen. Theoremexplainagent: Towards video-based multimodal explanations for llm theorem understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6663–6684, 2025b.

Peichao Lai, Jinhui Zhuang, Kexuan Zhang, Ningchang Xiong, Shengjie Wang, Yanwei Xu, Chong Chen, Yilei Wang, and Bin Cui. Webrenderbench: Enhancing web interface generation through layout-style consistency and reinforcement learning. *arXiv preprint arXiv:2510.04097*, 2025.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.

Hugo Laurençon, Léo Tronchon, and Victor Sanh. Unlocking the conversion of web screenshots into html code with the websight dataset. *arXiv preprint arXiv:2403.09029*, 2024.

Bingxuan Li, Yiwei Wang, Jiuxiang Gu, Kai-Wei Chang, and Nanyun Peng. Metal: A multi-agent framework for chart generation with test-time scaling. *arXiv preprint arXiv:2502.17651*, 2025a.

Jiahao Li, Yusheng Luo, Yunzhong Lou, and Xiangdong Zhou. Recad: Reinforcement learning enhanced parametric cad model generation with vision-language models. *arXiv preprint arXiv:2512.06328*, 2025b.

Jiahao Li, Weijian Ma, Xueyang Li, Yunzhong Lou, Guichun Zhou, and Xiangdong Zhou. Cad-llama: leveraging large language models for computer-aided design parametric 3d model generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18563–18573, 2025c.

Jinke Li, Jiarui Yu, Chenxing Wei, Hande Dong, Qiang Lin, Liangjing Yang, Zhicai Wang, and Yanbin Hao. Unisvg: A unified dataset for vector graphic understanding and generation with multimodal large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 13156–13163, 2025d.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022a.

Kaixin Li, Yuchen Tian, Qisheng Hu, Ziyang Luo, Zhiyong Huang, and Jing Ma. Mmcode: Benchmarking multimodal large language models for code generation with visually rich programming problems. *arXiv preprint arXiv:2404.09486*, 2024a.

Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. Tablebank: Table benchmark for image-based table detection and recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 1918–1925, 2020a.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15, 2020b.

Xueyang Li, Yu Song, Yunzhong Lou, and Xiangdong Zhou. Cad translator: An effective drive for text to 3d parametric computer-aided design generative modeling. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 8461–8470, 2024b.

Yuhang Li, Chenchen Zhang, Ruilin Lv, Ao Liu, Ken Deng, Yuanxing Zhang, Jiaheng Liu, Wiggin Zhou, and Bo Zhou. Relook: Vision-grounded rl with a multimodal llm critic for agentic web coding. *arXiv preprint arXiv:2510.11498*, 2025e.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022b.

Zhang Li, Yuliang Liu, Qiang Liu, Zhiyin Ma, Ziyang Zhang, Shuo Zhang, Zidun Guo, Jiarui Zhang, Xinyu Wang, and Xiang Bai. Monkeyocr: Document parsing with a structure-recognition-relation triplet paradigm. *arXiv preprint arXiv:2506.05218*, 2025f.

Zhen Li, Duan Li, Yukai Guo, Xinyuan Guo, Bowen Li, Lanxi Xiao, Shenyu Qiao, Jiashu Chen, Zijian Wu, Hui Zhang, et al. Chartgalaxy: A dataset for infographic chart understanding and generation. *arXiv preprint arXiv:2505.18668*, 2025g.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

Kevin Qinghong Lin, Siyuan Hu, Linjie Li, Zhengyuan Yang, Lijuan Wang, Philip Torr, and Mike Zheng Shou. Computer-use agents as judges for generative user interface. *arXiv preprint arXiv:2511.15567*, 2025a.

Kevin Qinghong Lin, Yuhao Zheng, Hangyu Ran, Dantong Zhu, Dongxing Mao, Linjie Li, Philip Torr, and Alex Jinpeng Wang. Vcode: a multimodal coding benchmark with svg as symbolic visual representation. *arXiv preprint arXiv:2511.02778*, 2025b.

Zefu Lin, Rongxu Cui, Chen Hanning, Xiangyu Wang, Junjia Xu, Xiaojuan Jin, Chen Wenbo, Hui Zhou, Lue Fan, Wenling Li, et al. Embodiedcoder: Parameterized embodied mobile manipulation via modern coding model. *arXiv preprint arXiv:2510.06207*, 2025c.

Zhiyu Lin, Zhengda Zhou, Zhiyuan Zhao, Tianrui Wan, Yilun Ma, Junyu Gao, and Xuelong Li. Webuibench: A comprehensive benchmark for evaluating multimodal large language models in webui-to-code. *arXiv preprint arXiv:2506.07818*, 2025d.

Jun Ling, Yao Qi, Tao Huang, Shibo Zhou, Yanqin Huang, Jiang Yang, Ziqi Song, Ying Zhou, Yang Yang, Heng Tao Shen, et al. Table2latex-rl: High-fidelity latex code generation from table images via reinforced multimodal language models. *arXiv preprint arXiv:2509.17589*, 2025.

Chengzhi Liu, Yuzhe Yang, Kaiwen Zhou, Zhen Zhang, Yue Fan, Yanan Xie, Peng Qi, and Xin Eric Wang. Presenting a paper is an art: Self-improvement aesthetic agents for academic presentations. *arXiv preprint arXiv:2510.05571*, 2025a.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12756–12770, 2023a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023b.

Hongen Liu, Cheng Cui, Yuning Du, Yi Liu, and Gang Pan. Pp-formulanet: Bridging accuracy and efficiency in advanced formula recognition. *arXiv preprint arXiv:2503.18382*, 2025b.

Tianyang Liu, Canwen Xu, and Julian McAuley. Repobench: Benchmarking repository-level code auto-completion systems. *arXiv preprint arXiv:2306.03091*, 2023c.

Yuan Liu, Zhongyin Zhao, Le Tian, Haicheng Wang, Xubing Ye, Yangxiu You, Zilin Yu, Chuhan Wu, Zhou Xiao, Yang Yu, et al. Points-reader: Distillation-free adaptation of vision-language models for document conversion. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1576–1601, 2025c.

Zejie Liu, Xiaoyu Hu, Deyu Zhou, Lin Li, Xu Zhang, and Yanzheng Xiang. Code generation from flowcharts with texts: A benchmark dataset and an approach. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6069–6077, 2022.

Ziyu Liu, Yuhang Zang, Yushan Zou, Zijian Liang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual agentic reinforcement fine-tuning. *arXiv preprint arXiv:2505.14246*, 2025d.

Jinwei Lu, Yuanfeng Song, Haodi Zhang, Chen Jason Zhang, Kaishun Wu, and Raymond Chi-Wing Wong. Towards robustness of text-to-visualization translation against lexical and phrasal variability. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 793–806. IEEE, 2025a.

Meng Lu, Ran Xu, Yi Fang, Wenxuan Zhang, Yue Yu, Gaurav Srivastava, Yuchen Zhuang, Mohamed Elhoseiny, Charles Fleming, Carl Yang, et al. Scaling agentic reinforcement learning for tool-integrated reasoning in vlms. *arXiv preprint arXiv:2511.19773*, 2025b.

Yiting Lu, Jiakang Yuan, Zhen Li, Shitian Zhao, Qi Qin, Xinyue Li, Le Zhuo, Licheng Wen, Dongyang Liu, Yuewen Cao, et al. Omnicaptioner: One captioner to rule them all. *arXiv preprint arXiv:2504.07089*, 2025c.

Zimu Lu, Yunqiao Yang, Houxing Ren, Haotian Hou, Han Xiao, Ke Wang, Weikang Shi, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Webgen-bench: Evaluating llms on generating interactive and functional websites from scratch. *arXiv preprint arXiv:2505.03733*, 2025d.

Reuben Luera, Ryan A. Rossi, Alexa Siu, Franck Dernoncourt, Tong Yu, Sungchul Kim, Ruiyi Zhang, Xiang Chen, Hanieh Salehy, Jian Zhao, Samyadeep Basu, Puneet Mathur, and Nedim Lipka. Survey of user interface design and interaction techniques in generative ai applications, 2024. URL `https://arxiv.org/abs/2410.22370`.

Tianqi Luo, Chuhan Huang, Leixian Shen, Boyan Li, Shuyu Shen, Wei Zeng, Nan Tang, and Yuyu Luo. nvbench 2.0: Resolving ambiguity in text-to-visualization through stepwise reasoning. *arXiv preprint arXiv:2503.12880*, 2025.

Yuyu Luo, Jiawei Tang, and Guoliang Li. nvbench: A large-scale synthesized dataset for cross-domain natural language to visualization task. *arXiv preprint arXiv:2112.12926*, 2021.

Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16314–16323, 2022.

Dimitrios Mallis, Ahmet Serda Karadeniz, Sebastian Cavada, Danila Rukhovich, Niki Foteinopoulou, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-assistant: tool-augmented vllms as generic cad task solvers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7284–7294, 2025.

Souvik Mandalm. Nanonets-OCR-s. `https://nanonets.com/research/nanonets-ocr-s/`, 2025. Accessed: 2025-12-12.

Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024.

Juhong Min, Shyamal Buch, Arsha Nagrani, Minsu Cho, and Cordelia Schmid. Morevqa: Exploring modular reasoning models for video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13235–13245, 2024.

Yao Mu, Junting Chen, Qinglong Zhang, Shoufa Chen, Qiaojun Yu, Chongjian Ge, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, et al. Robocodex: Multimodal code generation for robotic behavior synthesis. *arXiv preprint arXiv:2402.16117*, 2024.

Hyuk Namgoong, Jeesu Jung, Hyeonseok Kang, Yohan Lee, and Sangkeun Jung. Amace: Automatic multi-agent chart evolution for iteratively tailored chart generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21483–21498, 2025.

Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. Tableformer: Table structure understanding with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4614–4623, 2022.

Yuansheng Ni, Songcheng Cai, Xiangchao Chen, Jiarong Liang, Zhiheng Lyu, Jiaqi Deng, Kai Zou, Ping Nie, Fei Yuan, Xiang Yue, et al. Viscoder2: Building multi-language visualization coding agents. *arXiv preprint arXiv:2510.23642*, 2025a.

Yuansheng Ni, Ping Nie, Kai Zou, Xiang Yue, and Wenhu Chen. Viscoder: Fine-tuning llms for executable python visualization code generation. *arXiv preprint arXiv:2506.03930*, 2025b.

Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, et al. Mineru2. 5: A decoupled vision-language model for efficient high-resolution document parsing. *arXiv preprint arXiv:2509.22186*, 2025a.

Ke Niu, Haiyang Yu, Zhuofan Chen, Zhengtao Yao, Weitao Jia, Xiaodong Ge, Jingqun Tang, Benlei Cui, Bin Li, and Xiangyang Xue. Cme-cad: Heterogeneous collaborative multi-expert reinforcement learning for cad code generation. *arXiv preprint arXiv:2512.23333*, 2025b.

Ke Niu, Haiyang Yu, Zhuofan Chen, Mengyang Zhao, Teng Fu, Bin Li, and Xiangyang Xue. From intent to execution: Multimodal chain-of-thought reinforcement learning for precise cad code generation. *arXiv preprint arXiv:2508.10118*, 2025c.

Tianhao Niu, Yiming Cui, Baoxin Wang, Xiao Xu, Xin Yao, Qingfu Zhu, Dayong Wu, Shijin Wang, and Wanxiang Che. Chart2code53: A large-scale diverse and complex dataset for enhancing chart-to-code generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15839–15855, 2025d.

OpenDataLab. Pdf-extract-kit. `https://github.com/opendatalab/PDF-Extract-Kit`, 2025. Accessed: 2025-12-12.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Geliang Ouyang, Jingyao Chen, Zhihe Nie, Yi Gui, Yao Wan, Hongyu Zhang, and Dongping Chen. nvagent: Automated data visualization from natural language via collaborative agent workflow. *arXiv preprint arXiv:2502.05036*, 2025a.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, et al. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 24838–24848, 2025b.

Wei Pang, Kevin Qinghong Lin, Xiangru Jian, Xi He, and Philip Torr. Paper2poster: Towards multimodal poster automation from scientific papers. *arXiv preprint arXiv:2505.21497*, 2025.

Vik Paruchuri. Texify. `https://github.com/VikParuchuri/texify`, 2023. Accessed: 2025-12-10.

Vik Paruchuri. Marker: Fast and accurate pdf to markdown converter. `https://github.com/datalab-to/marker`, 2025. Accessed: 2025-12-12.

ShengYun Peng, Aishwarya Chakravarthy, Seongmin Lee, Xiaojing Wang, Rajarajeswari Balasubramaniyan, and Duen Horng Chau. Unitable: Towards a unified framework for table recognition via self-supervised pretraining. *arXiv preprint arXiv:2403.04822*, 2024.

Sagi Polaczek, Yuval Alaluf, Elad Richardson, Yael Vinker, and Daniel Cohen-Or. Neuralsvg: An implicit representation for text-to-vector generation. *arXiv preprint arXiv:2501.03992*, 2025.

Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. olmocr: Unlocking trillions of tokens in pdfs with vision language models. *arXiv preprint arXiv:2502.18443*, 2025a.

Jake Poznanski, Luca Soldaini, and Kyle Lo. olmocr 2: Unit test rewards for document ocr. *arXiv preprint arXiv:2510.19817*, 2025b.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8494–8502, 2018.

Ji Qi, Ming Ding, Weihan Wang, Yushi Bai, Qingsong Lv, Wenyi Hong, Bin Xu, Lei Hou, Juanzi Li, Yuxiao Dong, and Jie Tang. Cogcom: A visual language model with chain-of-manipulations reasoning, 2025. URL `https://arxiv.org/abs/2402.04236`.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Zhen Liu, Tim Z Xiao, Katherine M Collins, Joshua B Tenenbaum, Adrian Weller, Michael J Black, and Bernhard Schölkopf. Can large language models understand symbolic graphics programs? *arXiv preprint arXiv:2408.08313*, 2024.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Mizanur Rahman, Md Tahmid Rahman Laskar, Shafiq Joty, and Enamul Hoque. Text2vis: A challenging and diverse benchmark for generating multimodal visualizations from text. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31837–31862, 2025.

Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7342–7351, 2021.

rednote. dots.ocr: Multilingual document layout parsing in a single vision-language model. `https://github.com/rednote-hilab/dots.ocr`, 2025. Accessed: 2025-12-12.

Sina Rismanchian, Yasaman Razeghi, Sameer Singh, and Shayan Doroudi. Turtlebench: A visual programming benchmark in turtle geometry. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12170–12188, 2025.

Josselin S Roberts, Tony Lee, Chi H Wong, Michihiro Yasunaga, Yifan Mai, and Percy Liang. Image2struct: Benchmarking structure extraction for vision-language models. *Advances in Neural Information Processing Systems*, 37:115058–115097, 2024.

Juan Rodriguez, Xiangru Jian, Siba Smarak Panigrahi, Tianyu Zhang, Aarash Feizi, Abhay Puri, Akshay Kalkunte, François Savard, Ahmed Masry, Shravan Nayak, et al. Bigdocs: An open dataset for training multimodal models on document and code tasks. *arXiv preprint arXiv:2412.04626*, 2024.

Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images. *arXiv preprint arXiv:2312.11556*, 2023.

Juan A Rodriguez, Haotian Zhang, Abhay Puri, Aarash Feizi, Rishav Pramanik, Pascal Wichmann, Arnab Mondal, Mohammad Reza Samsami, Rabiul Awal, Perouz Taslakian, et al. Rendering-aware reinforcement learning for vector graphics generation. *arXiv preprint arXiv:2505.20793*, 2025.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL `https://arxiv.org/abs/2308.12950`.

Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-recode: Reverse engineering cad code from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9801–9811, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Wonduk Seo, Seungyong Lee, Daye Kang, Hyunjin An, Zonghao Yuan, and Seunghyun Lee. Automated visualization code synthesis via multi-path reasoning and feedback-driven optimization. *arXiv preprint arXiv:2502.11140*, 2025.

Chuyi Shang, Amos You, Sanjay Subramanian, Trevor Darrell, and Roei Herzig. Traveler: A modular multi-lmm agent framework for video question-answering. *arXiv preprint arXiv:2404.01476*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

Jingwei Shi, Zeyu Zhang, Biao Wu, Yanjie Liang, Meng Fang, Ling Chen, and Yang Zhao. Presentagent: Multimodal agent for presentation video generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 760–773, 2025.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 3956–3974, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: program generation for situated robot task planning using large language models. *Autonomous Robots*, 47(8):999–1012, 2023.

Shubhankar Singh, Purvi Chaurasia, Yerram Varun, Pranshu Pandya, Vatsal Gupta, Vivek Gupta, and Dan Roth. Flowvqa: Mapping multimodal logic in visual question answering with flowcharts. *arXiv preprint arXiv:2406.19237*, 2024.

Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, et al. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers. *arXiv preprint arXiv:2506.23918*, 2025.

Sanjay Subramanian, Medhini Narasimhan, Kushal Khangaonkar, Kevin Yang, Arsha Nagrani, Cordelia Schmid, Andy Zeng, Trevor Darrell, and Dan Klein. Modular visual question answering via code generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Toronto, Canada, 2023. Association for Computational Linguistics.

Haoyu Sun, Huichen Will Wang, Jiawei Gu, Linjie Li, and Yu Cheng. Fullfront: Benchmarking mllms across the full front-end engineering workflow. *arXiv preprint arXiv:2505.17399*, 2025a.

Qiushi Sun, Zhirui Chen, Fangzhi Xu, Kanzhi Cheng, Chang Ma, Zhangyue Yin, Jianing Wang, Chengcheng Han, Renyu Zhu, Shuai Yuan, et al. A survey of neural code intelligence: Paradigms, advances and beyond. *arXiv preprint arXiv:2403.14734*, 2024.

Qiushi Sun, Jingyang Gong, Yang Liu, Qiaosheng Chen, Lei Li, Kai Chen, Qipeng Guo, Ben Kao, and Fei Yuan. Januscoder: Towards a foundational visual-programmatic interface for code intelligence. *arXiv preprint arXiv:2510.23538*, 2025b.

Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, et al. Scienceboard: Evaluating multimodal autonomous agents in realistic scientific workflows. *arXiv preprint arXiv:2505.19897*, 2025c.

Shao-Hua Sun, Hyeonwoo Noh, Sriram Somasundaram, and Joseph Lim. Neural program synthesis from diverse demonstration videos. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4790–4799. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/sun18a.html`.

Tao Sun, Enhao Pan, Zhengkai Yang, Kaixin Sui, Jiajun Shi, Xianfu Cheng, Tongliang Li, Wenhao Huang, Ge Zhang, Jian Yang, et al. P2p: Automated paper-to-poster generation and fine-grained benchmark. *arXiv preprint arXiv:2505.17104*, 2025d.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning, 2023. URL `https://arxiv.org/abs/2303.08128`.

Wentao Tan, Qiong Cao, Chao Xue, Yibing Zhan, Changxing Ding, and Xiaodong He. Chartmaster: Advancing chart-to-code generation with real-world charts and chart similarity reinforcement learning. *arXiv preprint arXiv:2508.17608*, 2025.

Jiahao Tang, Henry Hengyuan Zhao, Lijian Wu, Yifei Tao, Dongxing Mao, Yang Wan, Jingru Tan, Min Zeng, Min Li, and Alex Jinpeng Wang. From charts to code: A hierarchical benchmark for multimodal models. *arXiv preprint arXiv:2510.17932*, 2025a.

Wenxin Tang, Jingyu Xiao, Wenxuan Jiang, Xi Xiao, Yuhang Wang, Xuxin Tang, Qing Li, Yuehe Ma, Junliang Liu, Shisong Tang, et al. Slidecoder: Layout-aware rag-enhanced hierarchical slide generation from design. *arXiv preprint arXiv:2506.07964*, 2025b.

Wenxin Tang, Jingyu Xiao, Yanpei Gong, Fengyuan Ran, Tongchuan Xia, Junliang Liu, Man Ho Lam, Wenxuan Wang, and Michael R Lyu. Efficientpostergen: Semantic-aware efficient poster generation via token compression and accurate violation detection. *arXiv preprint arXiv:2603.00155*, 2026.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

David Venuto, Sami Nur Islam, Martin Klissarov, Doina Precup, Sherry Yang, and Ankit Anand. Code as reward: Empowering reinforcement learning with vlms. *arXiv preprint arXiv:2402.04764*, 2024.

Yuxuan Wan, Yi Dong, Jingyu Xiao, Yintong Huo, Wenxuan Wang, and Michael R Lyu. Mrweb: An exploration of generating multi-page resource-aware web code from ui designs. *arXiv preprint arXiv:2412.15310*, 2024.

Yuxuan Wan, Tingshuo Liang, Jiakai Xu, Jingyu Xiao, Yintong Huo, and Michael R Lyu. Automatically generating web applications from requirements via multi-agent test-driven development. *arXiv preprint arXiv:2509.25297*, 2025.

Baode Wang, Biao Wu, Weizhen Li, Meng Fang, Yanjie Liang, Zuming Huang, Haozhe Wang, Jun Huang, Ling Chen, Wei Chu, et al. Infinity parser: Layout aware reinforcement learning for scanned document parsing. *arXiv preprint arXiv:2506.03197*, 2025a.

Bin Wang, Zhuangcheng Gu, Guang Liang, Chao Xu, Bo Zhang, Botian Shi, and Conghui He. Unimernet: A universal network for real-world mathematical expression recognition. *arXiv preprint arXiv:2404.15254*, 2024a.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, et al. Mineru: An open-source solution for precise document content extraction. *arXiv preprint arXiv:2409.18839*, 2024b.

Chenyu Wang, Weixin Luo, Sixun Dong, Xiaohua Xuan, Zhengxin Li, Lin Ma, and Shenghua Gao. Mllm-tool: A multimodal large language model for tool agent learning. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6678–6687. IEEE, 2025b.

Feiyu Wang, Zhiyuan Zhao, Yuandong Liu, Da Zhang, Junyu Gao, Hao Sun, and Xuelong Li. Svgen: Interpretable vector graphics generation with large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 9608–9617, 2025c.

Hanbin Wang, Xiaoxuan Zhou, Zhipeng Xu, Keyuan Cheng, Yuxin Zuo, Kai Tian, Jingwei Song, Junting Lu, Wenhui Hu, and Xueyang Liu. Code-vision: Evaluating multimodal llms logic understanding and code generation capabilities. *arXiv preprint arXiv:2502.11829*, 2025d.

Haozhe Wang, Alex Su, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*, 2025e.

Jiale Wang, Junhui Yu, Huanyong Liu, and Chenanran Kong. Enhancing complex formula recognition with hierarchical detail-focused network. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025f.

Jiuniu Wang, Gongjie Zhang, Quanhao Qian, Junlong Gao, Deli Zhao, and Ran Xu. Robosvg: A unified framework for interactive svg generation with multi-modal guidance. *arXiv preprint arXiv:2510.22684*, 2025g.

Ke Wang, Junting Pan, Linda Wei, Aojun Zhou, Weikang Shi, Zimu Lu, Han Xiao, Yunqiao Yang, Houxing Ren, Mingjie Zhan, and Hongsheng Li. Mathcoder-VL: Bridging vision and code for enhanced multimodal mathematical reasoning. In *The 63rd Annual Meeting of the Association for Computational Linguistics*, 2025h. URL https://openreview.net/forum?id=nuvtX1imAb.

Kuang-Da Wang, Zhao Wang, Yotaro Shimose, Wei-Yao Wang, and Shingo Takamatsu. Webgen-v bench: Structured representation for enhancing visual design in llm-based web generation and evaluation. *arXiv preprint arXiv:2510.15306*, 2025i.

Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. Text-to-cad generation through infusing visual feedback in large language models. *arXiv preprint arXiv:2501.19054*, 2025j.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision*, pp. 58–76. Springer, 2024c.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024d.

Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, et al. General ocr theory: Towards ocr-2.0 via a unified end-to-end model. *arXiv preprint arXiv:2409.01704*, 2024.

Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025a.

Jingxuan Wei, Caijun Jia, Qi Chen, Honghao He, Linzhuang Sun, Conghui He, Lijun Wu, Bihui Yu, and Cheng Tan. Geoint-r1: Formalizing multimodal geometric reasoning with dynamic auxiliary constructions, 2025b. URL https://arxiv.org/abs/2508.03173.

Yao Wei, Yanchao Sun, Ruijie Zheng, Sai Vemprala, Rogerio Bonatti, Shuhang Chen, Ratnesh Madaan, Zhongjie Ba, Ashish Kapoor, and Shuang Ma. Is imitation all you need? generalized decision-making with dual-phase training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16221–16231, 2023.

Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4): 1–24, 2021.

Chengyue Wu, Zhixuan Liang, Yixiao Ge, Qiushan Guo, Zeyu Lu, Jiahao Wang, Ying Shan, and Ping Luo. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3006–3028, 2025a.

Jason Wu, Yi-Hao Peng, Xin Yue Amanda Li, Amanda Swearngin, Jeffrey P Bigham, and Jeffrey Nichols. Uiclip: a data-driven model for assessing user interface design. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–16, 2024.

Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing. *arXiv preprint arXiv:2506.09965*, 2025b.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Iconshop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics (TOG)*, 42(6):1–14, 2023a.

Ronghuan Wu, Wanchao Su, and Jing Liao. Chat2svg: Vector graphics generation with large language models and image diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 23690–23700, 2025c.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6772–6782, 2021.

Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score: Better aligning text-to-image models with human preference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2096–2105, 2023b.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

Renqiu Xia, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Botian Shi, Junchi Yan, and Bo Zhang. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *IEEE Transactions on Image Processing*, 2025a.

Renqiu Xia, Hongbin Zhou, Ziming Feng, Huanxi Liu, Boan Chen, Bo Zhang, and Junchi Yan. Latexnet: A specialized model for converting visual tables and equations to latex code. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025b.

Jingyu Xiao, Yuxuan Wan, Yintong Huo, Zixin Wang, Xinyi Xu, Wenxuan Wang, Zhiyao Xu, Yuhang Wang, and Michael R Lyu. Interaction2code: Benchmarking mllm-based interactive webpage code generation from interactive prototyping. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 241–253. IEEE, 2025a.

Jingyu Xiao, Ming Wang, Man Ho Lam, Yuxuan Wan, Junliang Liu, Yintong Huo, and Michael R Lyu. Designbench: A comprehensive benchmark for mllm-based front-end code generation. *arXiv preprint arXiv:2506.06251*, 2025b.

Jingyu Xiao, Zhongyi Zhang, Yuxuan Wan, Yintong Huo, Yang Liu, and Michael R Lyu. Efficient-tuicoder: Efficient mllm-based ui code generation via input and output token compression. *arXiv preprint arXiv:2509.12159*, 2025c.

Jingyu Xiao, Jiantong Qin, Shuoqi Li, Man Ho Lam, Yuxuan Wan, Jen-tse Huang, Yintong Huo, and Michael R Lyu. Comuicoder: Component-based reusable ui code generation for complex websites via semantic segmentation and element-wise feedback. *arXiv preprint arXiv:2602.19276*, 2026.

Senwei Xie, Hongyu Wang, Zhanqi Xiao, Ruiping Wang, and Xilin Chen. Robotic programmer: Video instructed policy code generation for robotic manipulation, 2025. URL `https://arxiv.org/abs/2501.04268`.

Ximing Xing, Juncheng Hu, Jing Zhang, Dong Xu, and Qian Yu. Svgfusion: Scalable text-to-svg generation via vector space diffusion. *arXiv preprint arXiv:2412.10437*, 2024a.

Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. Svgdreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4546–4555, 2024b.

Ximing Xing, Yandong Guan, Jing Zhang, Dong Xu, and Qian Yu. Reason-svg: Hybrid reward rl for aha-moments in vector graphics generation. *arXiv preprint arXiv:2505.24499*, 2025a.

Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering llms to understand and generate complex vector graphics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19487–19497, 2025b.

Chengzhi Xu, Yuyang Wang, Lai Wei, Lichao Sun, and Weiran Huang. Improved iterative refinement for chart-to-code generation via structured instruction. *arXiv preprint arXiv:2506.14837*, 2025a.

Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm. *arXiv preprint arXiv:2411.04954*, 2024a.

Kai Xu, YiWei Mao, XinYi Guan, and ZiLong Feng. Web-bench: A llm code benchmark based on web standards and frameworks. *arXiv preprint arXiv:2505.07473*, 2025b.

Mingde Xu, Zhen Yang, Wenyi Hong, Lihang Pan, Xinyue Fan, Yan Wang, Xiaotao Gu, Bin Xu, and Jie Tang. Webvia: A web-based vision-language agentic framework for interactive and verifiable ui-to-code generation. *arXiv preprint arXiv:2511.06251*, 2025c.

Xiaojie Xu, Xinli Xu, Sirui Chen, Haoyu Chen, Fan Zhang, and Ying-Cong Chen. Pregenie: An agentic framework for high-quality visual presentation generation. *arXiv preprint arXiv:2505.21660*, 2025d.

Zhengzhuo Xu, Bowen Qu, Yiyan Qi, Sinan Du, Chengjin Xu, Chun Yuan, and Jian Guo. Chartmoe: Mixture of diversely aligned expert connector for chart understanding. *arXiv preprint arXiv:2409.03277*, 2024b.

Pengyu Yan, Mahesh Bhosale, Jay Lal, Bikhyat Adhikari, and David Doermann. Chartreformer: Natural language-driven chart image editing. In *International Conference on Document Analysis and Recognition*, pp. 453–469. Springer, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, et al. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. *arXiv preprint arXiv:2406.09961*, 2024a.

Donglu Yang, Liang Zhang, Zihao Yue, Liangyu Chen, Yichen Xu, Wenxuan Wang, and Qin Jin. Chartm3: Benchmarking chart editing with multimodal instructions. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 5001–5009, 2025b.

Hao Yang, Weijie Qiu, Ru Zhang, Zhou Fang, Ruichao Mao, Xiaoyu Lin, Maji Huang, Zhaosong Huang, Teng Guo, Shuoyang Liu, et al. Ui-ug: A unified mllm for ui understanding and generation. *arXiv preprint arXiv:2509.24361*, 2025c.

Jian Yang, Xianglong Liu, Weifeng Lv, Ken Deng, Shawn Guo, Lin Jing, Yizhi Li, Shark Liu, Xianzhen Luo, Yuyu Luo, et al. From code foundation models to agents and applications: A comprehensive survey and practical guide to code intelligence. *arXiv preprint arXiv:2511.18538*, 2025d.

Jingkang Yang, Yuhao Dong, Shuai Liu, Bo Li, Ziyue Wang, Haoran Tan, Chencheng Jiang, Jiamu Kang, Yuanhan Zhang, Kaiyang Zhou, et al. Octopus: Embodied vision-language programmer from environmental feedback. In *European conference on computer vision*, pp. 20–38. Springer, 2024b.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024c.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024d. URL `https://arxiv.org/abs/2405.15793`.

John Yang, Carlos E Jimenez, Alex L Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R Narasimhan, et al. Swe-bench multimodal: Do ai systems generalize to visual software domains? *arXiv preprint arXiv:2410.03859*, 2024e.

Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Fukun Yin, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. Omnisvg: A unified scalable vector graphics generation model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025e.

Yue Yang, Ajay Patel, Matt Deitke, Tanmay Gupta, Luca Weihs, Andrew Head, Mark Yatskar, Chris Callison-Burch, Ranjay Krishna, Aniruddha Kembhavi, and Christopher Clark. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. In *ACL 2025*, pp. 17486–17505. Association for Computational Linguistics, 2025f. URL `https://aclanthology.org/2025.acl-long.855/`.

Yue Yang, Ajay Patel, Matt Deitke, Tanmay Gupta, Luca Weihs, Andrew Head, Mark Yatskar, Chris Callison-Burch, Ranjay Krishna, Aniruddha Kembhavi, et al. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. *arXiv preprint arXiv:2502.14846*, 2025g.

Yuwei Yang, Zeyu Zhang, Yunzhong Hou, Zhuowan Li, Gaowen Liu, Ali Payani, Yuan-Sen Ting, and Liang Zheng. Effective training data synthesis for improving mllm chart understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2653–2663, 2025h.

Zhen Yang, Wenyi Hong, Mingde Xu, Xinyue Fan, Weihan Wang, Jiele Cheng, Xiaotao Gu, and Jie Tang. Ui2code$^n$: A visual language model for test-time scalable interactive ui-to-code generation, 2025i. URL `https://arxiv.org/abs/2511.08195`.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Zhiyu Yang, Zihan Zhou, Shuo Wang, Xin Cong, Xu Han, Yukun Yan, Zhenghao Liu, Zhixing Tan, Pengyuan Liu, Dong Yu, et al. Matplotagent: Method and evaluation for llm-based agentic scientific data visualization. *arXiv preprint arXiv:2402.11453*, 2024f.

Zongxin Yang, Guikun Chen, Xiaodi Li, Wenguan Wang, and Yi Yang. Doraemongpt: Toward understanding dynamic scenes with large language models (exemplified as a video agent). *arXiv preprint arXiv:2401.08392*, 2024g.

Haocheng Yuan, Jing Xu, Hao Pan, Adrien Bousseau, Niloy J Mitra, and Changjian Li. Cadtalk: An algorithm and benchmark for semantic commenting of cad programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3753–3762, 2024a.

Mingyue Yuan, Jieshan Chen, Yongquan Hu, Sidong Feng, Mulong Xie, Gelareh Mohammadi, Zhenchang Xing, and Aaron Quigley. Towards human-ai synergy in ui design: Enhancing multi-agent based ui generation with intent clarification and alignment. *arXiv preprint arXiv:2412.20071*, 2024b.

Chuhuai Yue, Jiajun Chai, Yufei Zhang, Zixiang Ding, Xihao Liang, Peixin Wang, Shihai Chen, Wang Yixuan, Guojun Yin, Wei Lin, et al. Uiorchestra: Generating high-fidelity code from ui designs with a multi-agent system. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 2769–2782, 2025.

Sukmin Yun, Haokun Lin, Rusiru Thushara, Mohammad Qazim Bhat, Yongxin Wang, Zutao Jiang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, Haonan Li, Preslav Nakov, Timothy Baldwin, Zhengzhong Liu, Eric P. Xing, Xiaodan Liang, and Zhiqiang Shen. Web2code: A large-scale webpage-to-code dataset and evaluation framework for multimodal llms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 112134–112157. Curran Associates, Inc., 2024.

Daoguang Zan, Bei Chen, Fengji Zhang, Dianjie Lu, Bingchao Wu, Bei Guan, Wang Yongji, and Jian-Guang Lou. Large language models meet nl2code: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7443–7464, 2023.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

Chenchen Zhang, Yuhang Li, Can Xu, Jiaheng Liu, Ao Liu, Changzhi Zhou, Ken Deng, Dengpeng Wu, Guanhua Huang, Kejiao Li, et al. Artifactsbench: Bridging the visual-interactive gap in llm code generation evaluation. *arXiv preprint arXiv:2507.04952*, 2025a.

Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. Repocoder: Repository-level code completion through iterative retrieval and generation. *arXiv preprint arXiv:2303.12570*, 2023.

Fengji Zhang, Linquan Wu, Guancheng Lin, Xiao Li, Xiao Yu, Yue Wang, Bei Chen, Jacky Keung, et al. Humaneval-v: Evaluating visual understanding and reasoning abilities of large multimodal models through coding tasks. 2024a.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025b.

Jiajun Zhang, Jianke Zhang, Zeyu Cui, Jiaxi Yang, Lei Zhang, Binyuan Hui, Qiang Liu, Zilei Wang, Liang Wang, and Junyang Lin. Plotcraft: Pushing the limits of llms for complex and interactive data visualization. *arXiv preprint arXiv:2511.00010*, 2025c.

Jinglei Zhang, Yuanfan Guo, Rolandos Alexandros Potamias, Jiankang Deng, Hang Xu, and Chao Ma. Vtimecot: Thinking by drawing for video temporal grounding and reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 24203–24213, 2025d.

Linhao Zhang, Daoguang Zan, Quanshun Yang, Zhirong Huang, Dong Chen, Bo Shen, Tianyu Liu, Yongshun Gong, Huang Pengjie, Xudong Lu, et al. Codev: Issue resolving with visual data. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7350–7361, 2025e.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, et al. Thyme: Think beyond images. *arXiv preprint arXiv:2508.11630*, 2025f.

Yifan Zhang, Liang Hu, Haofeng Sun, Peiyu Wang, Yichen Wei, Shukang Yin, Jiangbo Pei, Wei Shen, Peng Xia, Yi Peng, et al. Skywork-r1v4: Toward agentic multimodal intelligence through interleaved thinking with images and deepresearch. *arXiv preprint arXiv:2512.02395*, 2025g.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1592–1604, 2024b.

Zhehao Zhang, Ryan Rossi, Tong Yu, Franck Dernoncourt, Ruiyi Zhang, Jiuxiang Gu, Sungchul Kim, Xiang Chen, Zichao Wang, and Nedim Lipka. Vipact: Visual-perception enhancement via specialized vlm agent collaboration and tool-use. *arXiv preprint arXiv:2410.16400*, 2024c.

Zhihan Zhang, Yixin Cao, and Lizi Liao. Enhancing chart-to-code generation in multimodal large language models via iterative dual preference learning. *arXiv preprint arXiv:2504.02906*, 2025h.

Zhilin Zhang, Xiang Zhang, Jiaqi Wei, Yiwei Xu, and Chenyu You. Postergen: Aesthetic-aware paper-to-poster generation via multi-agent llms. *arXiv preprint arXiv:2508.17188*, 2025i.

Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling. *arXiv preprint arXiv:2507.07998*, 2025a.

Xuanle Zhao, Deyang Jiang, Zhixiong Zeng, Lei Chen, Haibo Qiu, Jing Huang, Yufeng Zhong, Liming Zheng, Yilin Cao, and Lin Ma. Vincicoder: Unifying multimodal code generation via coarse-to-fine visual reinforcement learning. *arXiv preprint arXiv:2511.00391*, 2025b.

Xuanle Zhao, Xuexin Liu, Haoyue Yang, Xianzhen Luo, Fanhu Zeng, Jianling Li, Qi Shi, and Chi Chen. Chartedit: How far are mllms from automating chart analysis? evaluating mllms' capability via chart editing. *arXiv preprint arXiv:2505.11935*, 2025c.

Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Zhiyuan Liu, and Maosong Sun. Chartcoder: Advancing multimodal large language model for chart-to-code generation. *arXiv preprint arXiv:2501.06598*, 2025d.

Xuanle Zhao, Shuxin Zeng, Xinyuan Cai, Xiang Cheng, Duzhen Zhang, Xiuyi Chen, and Bo Xu. Tinychemvl: Advancing chemical vision-language models via efficient visual token reduction and complex reaction tasks. *arXiv preprint arXiv:2511.06283*, 2025e.

Zhonghan Zhao, Wenhao Chai, Xuan Wang, Boyi Li, Shengyu Hao, Shidong Cao, Tian Ye, and Gaoang Wang. See and think: Embodied agent in virtual environment. In *European Conference on Computer Vision*, pp. 187–204. Springer, 2024.

Hao Zheng, Xinyan Guan, Hao Kong, Wenkai Zhang, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. Pptagent: Generating and evaluating presentations beyond text-to-slides. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14413–14429, 2025.

Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 697–706, 2021.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: data, model, and evaluation. In *European conference on computer vision*, pp. 564–580. Springer, 2020.

Yufeng Zhong, Lei Chen, Zhixiong Zeng, Xuanle Zhao, Deyang Jiang, Liming Zheng, Jing Huang, Haibo Qiu, Peng Shi, Siqi Yang, et al. Reading or reasoning? format decoupled reinforcement learning for document ocr. *arXiv preprint arXiv:2601.08834*, 2025a.

Yufeng Zhong, Zhixiong Zeng, Lei Chen, Longrong Yang, Liming Zheng, Jing Huang, Siqi Yang, and Lin Ma. Doctron-formula: Generalized formula recognition in complex and structured scenarios. *arXiv preprint arXiv:2508.00311*, 2025b.

Yufeng Zhong, Lei Chen, Xuanle Zhao, Wenkang Han, Liming Zheng, Jing Huang, Deyang Jiang, Yilin Cao, Lin Ma, and Zhixiong Zeng. Ocrverse: Towards holistic ocr in end-to-end vision-language models. *arXiv preprint arXiv:2601.21639*, 2026.

Ting Zhou, Yanjie Zhao, Xinyi Hou, Xiaoyu Sun, Kai Chen, and Haoyu Wang. Bridging design and development with automated declarative ui code generation. *arXiv preprint arXiv:2409.11667*, 2024.

Zheyuan Zhou, Jiayi Han, Liang Du, Naiyu Fang, Lemiao Qiu, and Shuyou Zhang. Cad-judge: Toward efficient morphological grading and verification for text-to-cad generation. *arXiv preprint arXiv:2508.04002*, 2025.

Haokun Zhu, Juang Ian Chong, Teng Hu, Ran Yi, Yu-Kun Lai, and Paul L Rosin. Samvg: A multi-stage image vectorization model with the segment-anything model. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4350–4354. IEEE, 2024a.

Qingfu Zhu, Xianzhen Luo, Fang Liu, Cuiyun Gao, and Wanxiang Che. A survey on natural language processing for programming. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 1690–1704, 2024b.

Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. Vgbench: Evaluating large language models on vector graphics understanding and generation. *arXiv preprint arXiv:2407.10972*, 2024.