A Multi-Agent MCP-Based System for Digitizing Hand-Annotated Documents

Tony Tian
Department of Computer Science
Columbia University
New York, NY, USA
jt3640@columbia.edu

Anusha Sonthalia
Department of Computer Science
Columbia University
New York, USA
anusha.sonthalia@columbia.edu

***Abstract*—The modern digital workflow no longer constrains itself to typed text alone. With the rise of tablets and stylus-enabled devices, handwriting has re-emerged as a preferred mode of expression, offering cognitive and creative benefits that typing cannot fully replicate. Studies in educational psychology suggest that handwriting engages deeper conceptual processing and memory formation than keyboard input. Yet handwritten annotations on digital documents—PDFs, contracts, or manuscripts—remain static and disconnected from the underlying document structure.**

This project proposes an agentic system that converts handwritten annotations into digitally redlined Google Doc using a Model Context Protocol (MCP)–based architecture. We study how progressively richer system designs affect image digitization performance, starting from an off-the-shelf Google Docs MCP, extending to an agentic MCP with specialized extraction tools, and finally incorporating a structured prompt playbook exposed as an MCP resource. These configurations are evaluated through controlled ablation studies measuring quality of outputs based on a test data set.

Our results demonstrate that architectural choices and prompt structuring significantly influence tool usage reliability and output correctness. Beyond text documents, this work establishes a foundational framework for extending annotation-to-digital translation to broader visual artifacts such as presentation slides, design mockups, and app wireframes.

***Index Terms*—model context protocol, agentic ai, digitization, tool-augmented llms, annotations, redlining,**

I. Introduction

The modern digital workflow no longer constrains itself to typed input. With the proliferation of stylus-enabled devices such as the Apple Pencil and Surface Pen, handwriting has re-emerged as a preferred form of annotation and feedback. Studies in educational psychology have demonstrated that handwriting engages deeper cognitive processing and improves retention compared to typing [1]. Despite these benefits, handwritten annotations for typed documents remain largely disconnected from the underlying digital structure.

In most document editing systems, handwriting is treated as a visual overlay rather than a meaningful textual layer. Consequently, handwritten notes on PDFs, contracts, or manuscripts are not-interactive and cannot easily be converted into edits on a digital document. This disconnect represents a gap between natural human input and modern digital document systems.

This project aims to close that gap by exploring whether large language models (LLMs) can translate handwritten annotations into digital document elements. Rather than focusing on the image processing itself, which is largely solved problem, we study the post-processing intelligence layer: how extracted annotation content can be aligned with document structure and converted into digital features within a Google Docs environment.

We propose and evaluate a Model Context Protocol (MCP)–based system in which an LLM acts as an orchestrator over document extraction tools and Google Docs APIs. Through a sequence of architectural ablations, we investigate how progressively adding agentic tools and structured prompt guidance affects alignment accuracy and output reliability. The contributions of this work are primarily system-level, demonstrating how architectural and prompting choices materially influence LLM tool-use performance in real-world document editing tasks.

II. Related Work

Prior work in document digitization using AI has primarily emphasized perceptual understanding, including layout-aware language modeling and visual-text alignment [12]. These approaches focus on interpreting the visual structure of documents, but typically stop at representation rather than downstream use. In contrast, we frame handwritten annotation understanding as an architectural problem, where handwritten elements are systematically converted into structured digital components to create a live document.

A. Optical Character Recognition

Optical Character Recognition (OCR) is widely regarded as a solved problem in document and image processing. Decades of research has produced many robust models capable of transcribing both printed and handwritten text with high accuracy across diverse document types [2], [4]. More recent deep learning approaches have achieved even better recognition quality, particularly for unconstrained handwriting which is particularly relevant to our use case [3]. Today, these OCR capabilities have been baked into many multi-modal large language models (LLMs) including Gemini and GPT which top the leader boards for image related tasks [13]. Given that we wouldn't be able to compete with these models given our

resources, we will use off-the-shelf model capabilities to solve this part of our problem.

Despite these advances, OCR-centric approaches are insufficient for digitizing hand-annotated documents. Annotated documents contain complex layouts and contextual cues which which do not map cleanly to linear text representations. Moreover, many handwritten marks are not intended as literal text, but instead encode intent edits, commentary, or emphasis with meaning derived from spatial context and visual components. Prior work in document analysis has shown that annotations function as meta-level information rather than primary document content [5]. Consequently, systems that treat handwriting solely as interpretable text fail to capture the true meaning of annotations and their relation to the underlying document, leaving a semantic gap between human feedback and usable digitized documents.

B. Tool-Augmented LLMs

Recent work on tool-augmented LLMs has demonstrated that LLMs can extend their capabilities by invoking external tools through structured interfaces. Approaches such as Toolformer enable models to learn when and how to call APIs, allowing deterministic operations to be delegated outside the model [6]. Similarly, systems like Gorilla focus on reliable API invocation across large and heterogeneous toolsets, emphasizing correct argument grounding and schema adherence [7]. These methods highlight the importance of coupling language understanding with tool execution for practical applications.

Agentic LLM frameworks build on this foundation by enabling models to perform multi-step reasoning and action selection. ReAct-style agents interleave reasoning traces with tool calls, allowing models to follow explicit reasoning paths and avoid skipping necessary steps [8]. Subsequent work has shown that reflective mechanisms further improve performance by enabling agents to revise their strategies based on observed failures [9]. While these techniques improve robustness, their effectiveness depends heavily on system design choices, including how tasks are decomposed and how tools are exposed to the model.

For our approach to annotated document digitization, orchestration is vital. We hypothesize that the best approach to correctly process documents with complex layouts and handwritten features, will be to break down processing into simpler sub-tasks. Tasks such as digitizing comments and aligning them with document structures can be done via tool usage over persistent document state. Benchmark evaluations of LLM agents show that failures often arise not from model incapability, but from poor task decomposition and ambiguous tool interface [10]. This motivates studying how architectural structure and explicit tool design affect reliability in annotation digitization pipelines.

C. Multi-Agent Systems via MCPs

Multi-agent systems create natural separation between orchestration logic and individual task execution in complex processes. Recent frameworks demonstrate that structured multi-agent communication improves scalability, modularity, and controllability in LLM-based systems [11]. However, realizing these benefits in practice requires an execution framework that enforces clear task boundaries, consistent tool interfaces, and reproducible model behavior across runs.

The Model Context Protocol (MCP) offers a particularly suitable framework for creating a multi-agent systems when agent tasks are well isolated and do not depend on evolving conversational context. Comment extraction via LLMs follows the same prompt structure every time and deviation from this or additional context can distract LLMs and reduce performance. Exposing these fixed prompt structures as MCP tools enables standardized, repeatable interactions between the orchestrating agent and specialized execution agents. MCP separates high-level reasoning and confusing and unnecessary conversational context from low-level execution that only requires specific context.

III. Methodology

A. Goal Definition

The primary goal of this system is to produce structured digital documents from hand-annotated document images. Given an input image containing a base document overlaid with handwritten annotations, the system must extract and convert these annotations into appropriate structured digital representations. These representations include typed comment content, highlighted text spans, and associations between annotations and the corresponding regions of the base document.

The secondary goal of this work focuses on the how architectural structure and tool orchestration affect the reliability and correctness of extracting structured data from complex, noisy visual inputs. The core research question is whether progressively richer system architectures can measurably improve alignment accuracy on this task.

B. Approach

In architectures that employ specialized tools, ChatGPT is used as the tooling LLM responsible for executing focused extraction tasks and producing structured outputs that are subsequently composed into a digitized Google Doc via the Google Docs MCP.

We investigate this question through a controlled architectural ablation study. Each system variant is designed to isolate the effect of architectural structure and tool orchestration on annotation-to-digital performance.

Across all configurations, Claude is used as the high-level orchestrating LLM. Claude is responsible for interpreting the input image, deciding which sub-tasks are required, selecting tools, sequencing calls, and creating content in the Google Doc. For specialized tools, ChatGPT is used as the LLM responsible for executing focused extraction tasks. This approach introduces an interesting complexity where Claude needs to decide to call tools to more effectively perform tasks it can do itself. The Claude orchester then takes extraction outputs and creates a digitized Google Doc via the Google Docs MCP.

We evaluate three system configurations of increasing architectural structure:

1) **Base MCP**: A minimal system in which Claude, acting alone, is prompted to interpret the annotated image and produce a digitized representation in a single step using an off-the-shelf Google Docs MCP. This configuration tests whether the orchestrator model can reliably perform complex annotation digitization without explicit task decomposition or specialized tooling.
2) **Multi-Agent MCP**: An extended architecture in which annotation digitization is decomposed into multiple specialized MCP tools. Claude is given the same simple prompt to interpret the annotated image and produce a digitized representation but has to make the decision to call specilaized tooling to improve performance.
3) **Multi-Agent MCP + Prompt**: A fully structured system that augments the multi-agent architecture with a ReAct-style prompt playbook exposed as an MCP resource. This resource provides explicit, reusable guidance for expected digitization workflows, encouraging consistent tool usage and reducing skipped or misordered extraction steps.

Importantly, this ablation does not attempt to control for differences in underlying model capabilities. Instead, it evaluates whether architectural structure and explicit tool separation enable reliable digitization in cases where a single, general-purpose model fails to perform the task end-to-end.

C. Success Criteria

System performance is evaluated by comparing extracted annotations against a manually created ground-truth dataset. We measure performance using standard information retrieval metrics:

- **Recall**: the proportion of ground-truth annotations correctly identified by the system.
- **Precision**: the proportion of extracted annotations that are correct.
- **F1 Score**: the harmonic mean of precision and recall, capturing the trade-off between coverage and correctness.

An extracted annotation is considered correct if it matches the ground truth in both semantic intent and annotation type (e.g., comment or highlight). For comments, extractions were considered correct if there were minor differences but no semantic difference between ground truth and extraction. These metrics allow us to quantify how architectural structure affects both completeness and reliability in annotation digitization. Improvements in F1 score across system variants indicate stronger alignment between visual annotations and their structured digital representations.

IV. System Design

Our final system is implemented as a layered architecture using the Model Context Protocol (MCP). We progressively augmented a baseline document digitization pipeline with agentic orchestration, prompt-engineered tooling, and explicit instructional guidance. The design goal is to understand how increasing architectural constraints and task decomposition affect the reliability of digitizing handwritten annotations.

A. Design Motivation

The primary objective of our project was to enable Claude to utilize a high-fidelity OCR model (GPT-5.2) for annotation extraction of attached images. Early experiments revealed several challenges in reliably extracting highlights and handwritten comments. Claude consistently ignored the obstruction tool and skipped highlighted or handwritten annotations, even when these were present in the document image.

B. Experiment 1 — Task Delineation Tools

To address this, we first created a tool explicitly delineating the steps necessary for successful extraction. Despite embedding these tools into a planner with strict imperative instructions, Claude largely ignored the tool. This indicated that tool-level guidance alone is insufficient, and that enforcing execution order requires code-level validation.

C. Experiment 2 — Passcode-Based Locking Mechanism

Next, we implemented a locking mechanism: the extraction tools extract_highlight and extract_comments each emitted a passcode, which was required by the createDocument tool to enforce execution order. While this approach successfully enforced the correct sequence, it exposed passcodes to the user, creating a potential security vulnerability. We concluded that a better approach was to rewrite createDocument to internally call the extraction tools while retaining their standalone usability.

D. Observations and New Objective

We initially implemented image processing by encoding images as Base64 strings. This approach encountered message length limitations, and resizing images compromised resolution and OCR accuracy. Consequently, the revised objective became to enable Claude to utilize GPT-5.2 for annotation extraction via image URLs rather than attached images. This ensures that the external OCR tool is invoked as required while avoiding payload size issues.

E. Final Design Choices for Extraction Tools

Based on these insights, we designed three conceptual extraction functions:

1) Baseline Text Extraction
2) Highlight Extraction
3) Handwritten Comment Extraction

Through testing, we found that highlights and handwritten comments depend directly on the baseline text, as all start and end indices must reference it. Therefore, we consolidated all three extraction steps into a single atomic function, which extracts the baseline text first, computes highlights and handwritten comments immediately after within the same vision-grounded call, and ensures start and end indices are correctly aligned with the extracted base text.

Although the instructions resemble a system prompt, we provided them as a user prompt. Empirically, this configuration produced more reliable extraction results when using image URLs, ensuring that the OCR, highlight, and comment extraction steps were properly grounded in the input image.

This consolidated approach proved to be the most efficient and robust, preserving modularity while significantly improving correctness guarantees in GPT-5.2.

F. Final Layered MCP Architecture

The final system integrates the consolidated extraction function within a layered MCP pipeline, combining agentic orchestration to coordinate tool execution, prompt-engineered instructions to enforce JSON output format and index accuracy, and grounded OCR with index computation performed in a single coherent operation.

This architecture balances modularity, robustness, and strict adherence to extraction rules, ensuring reliable extraction of printed text, highlighted segments, and handwritten annotations. By integrating lessons from experiments and observations, this design achieves maximal efficiency and reliability for multimodal document annotation extraction.

G. System Overview

The full system consists of three conceptual layers, coordinated by Claude acting as the orchestrator:

1) **Google Docs MCP:** a baseline integration that allows a language model to materialize extracted content into a Google Doc [14].
2) **Multi-agent MCP:** an MCP where tools call specialized modules to achieve image processing tasks. These include base text extraction, highlighted span detection, handwritten annotation extraction, and annotation context detection.
3) **Instructional Prompt:** a structured instructional playbook exposed as an MCP resource that constrains orchestrator behavior and enforces systematic tool usage.

Each layer builds on the previous one, enabling controlled ablation of architectural complexity and making it possible to isolate the impact of agentic decomposition and prompt structure on system performance.

H. Google Docs MCP

The Google Docs MCP provides an off-the-shelf interface for creating and populating Google Docs using language model outputs. In the baseline configuration, the orchestrating model (Claude) is prompted to interpret the annotated document image and directly generate a digitized Google Doc in a single step.

This configuration serves as a lower bound for performance and furthers the case for task specific prompt engineering. While the model is capable of basic OCR and document formatting, it struggles with complex layouts and handwritten annotations. In practice, this approach frequently results in missing annotations, misinterpreted highlights, and poorly structured documents. These failures illustrate that multimodal LLMs alone are insufficient for reliably digitizing messy, hand-annotated documents. The baseline clearly motivates the introduction of explicit task decomposition.

I. Multi-Agent MCP

To address the limitations of the baseline, we introduce additional tooling within the MCP architecture that break up image processing into multiple specialized tools. Claude still acts as the orchestrator but with access to more capable tooling. ChatGPT is used as the tooling LLM responsible for executing narrowly scoped extraction tasks. The following tools are exposed via MCP:

- `extract_base_text`
- `extract_comments`
- `extract_highlights`
- `associate_comments`

Each tool is designed to return structured outputs following a strict schema, allowing the orchestrator to compose results deterministically into a digitized Google Doc via the Google Docs tools.

Designing reliable tooling required significant prompt engineering effort. The most challenging tasks being comment extraction and highlight extraction. A recurring challenge with comments were incorporation of typed text and correct separation of separate annotations. Extracted comments often contained hallucinations relating to the base text but not correctly reflecting hte comment content. Comments were also often incorrectly merged or split, especially when marginal notes were spatially close which was often the case for crowded documents. This was mitigated by explicitly restricting content inferences and defining what constitutes a single comment using spatial proximity, semantic coherence, and visual cues.

Another major challenge was false highlight detection. Models frequently misclassified circles, underlines, or colored pen strokes as highlights. To reduce these errors, we enforced a narrow definition of highlights as transparent colored markers applied directly over letters in a continuous horizontal stroke. We also introduced strong negative rules and instructed the tool to return no highlights when evidence was ambiguous. This bias toward precision significantly reduced hallucinated highlights.

While the multi-agent MCP significantly improved extraction quality, orchestration remained unreliable. The orchestrator often attempted to interpret the image directly rather than invoking the specialized tools, leading to inconsistent behavior across runs.

J. Instructional Prompt

To address unreliable orchestration, we introduce an instructional prompt playbook exposed as an MCP resource. This ReAct-style prompt defines an explicit digitization workflow, specifying which tools must be called, in what order, and how their outputs should be included in the final Google Doc.

The playbook serves two purposes. First, it constrains the orchestrator by making tool usage mandatory, explicitly

discouraging direct image interpretation outside of tool calls. Second, it centralizes complex prompting logic into a reusable resource, reducing prompt drift and improving reproducibility across executions.

A key challenge addressed by the playbook was the orchestrator's tendency to bypass tools and attempt image processing. By encoding step-by-step expectations the prompt resource significantly reduced skipped steps and catastrophic failures. This final layer proved essential for achieving consistent, high-quality digitization and for realizing the full benefits of the multi-agent MCP architecture.

V. System Performance

A. Protocol

We evaluate system performance on a manually curated dataset of hand-annotated document images. We specifically use literary analysis documents which include poems or prose passages with handwritten comments and highlighted text. For each document, system outputs were manually compared against the original annotated image to determine which extractions were correct. Due to the large manual effort required to evaluate performance and the scarcity of appropriate samples, the test set is on the smaller side.

We focus evaluation on handwritten comments and highlighted text spans, as these annotation types exhibited the highest variability in system performance during development. In contrast, base text extraction was largely stable across configurations due to the preference towards typed content, making comments and highlights the most informative signals for comparing architectural choices. Furthermore, comment associate was difficult to evaluate deterministically as many comments were simply in margins and not explicitly linked to parts of the base text.

Evaluation was performed at the annotation level using manual comparison against the original document images. For highlights, predicted highlight spans were compared against reference spans. For handwritten comments, extracted annotations were evaluated based on semantic fidelity rather than exact string match. An extracted comment was considered correct if it preserved the original meaning and intent of the handwritten annotation, even when minor wording changes or formatting differences were present. The reason for this decision was that despite the significant prompt engineering effort, ChatGPT still sometimes missed minor punctuation marks or changed words to improve the grammar of annotations. These issues, though frustrating, retained semantic meaning of the comments and did not detract from the ultimate goal of the system. As such they were considered as correct extractions.

We report precision, recall, and F1 score for both highlights and comments. When no highlights were present in the ground truth for a given document, highlight metrics are reported as N/A.

B. Results

1) Baseline: Google Docs MCP Only: Table I reports per-document performance for the baseline configuration, where Claude directly produces a digitized Google Doc via the Google Docs MCP using a single end-to-end prompt.

The baseline system exhibits highly inconsistent behavior. Highlight detection frequently produces false positives, while comment extraction suffers from missed content, hallucinated content, and poor segmentation. Especially with regards to comment extraction, failures are often catastrophic with nonsensical content.

One notable failure occurs for Doc 6 in the baseline configuration. In this case, the orchestrator model Claude refused to perform extraction, likely due to the volume and density of handwritten annotations present in the document image. As a result, no comments or highlights were produced. This behavior highlights a practical limitation of relying on a single general-purpose model for end-to-end digitization when annotation complexity exceeds implicit model tolerance thresholds.

2) Multi-Agent MCP Tools: We attempted to evaluate an intermediate configuration consisting of the multi-agent MCP tooling without the instructional prompt resource. In this setting, Claude was responsible for orchestration and ChatGPT executed the extraction tools. Unfortunately, this configuration was unable to provide meaninful data in practice.

Across repeated runs, the orchestrator almost always bypassed the extraction tools and attempted to interpret the annotated images directly. This behavior led to outputs too close to the baseline thus not providing truly comparable data. As a consequence, we did not collect and compare per-document metrics for this configuration.

This failure case highlights an important insight which has been validated by previous research: introducing specialized tools alone is insufficient to guarantee reliable agent behavior. Explicit instructional constraints are required to ensure consistent tool usage and to realize the benefits of task decomposition.

3) Multi-Agent MCP with Instructional Prompt: Table II reports per-document performance for the fully structured system, combining multi-agent MCP tooling with a ReAct-style instructional prompt resource.

The fully structured system achieves strong and consistent performance across all documents. Highlight detection shows near-perfect precision and recall on applicable examples and correctly ignores cases with emphasis annotation which are not transparent marker highlights. Furthermore, comment extraction benefits from improved segmentation and reduced hallucination. Most importantly, the system avoids the catastrophic failure cases observed in the baseline configuration.

VI. Evaluation

A. Result Analysis

To assess whether the observed performance improvements are meaningful rather than incidental, we compare per-document F1 scores between the baseline system and the fully structured system. Across documents, the full system consistently outperforms the baseline for both highlight detection and

TABLE I
PER-DOCUMENT RESULTS: GOOGLE DOCS MCP ONLY

	Highlights						Comments					
Document	Ref	Pred	Corr	Recall	Prec	F1	Ref	Pred	Corr	Recall	Prec	F1
Doc 1	0	4	0	0	0	0	32	17	4	0.13	0.24	0.16
Doc 2	15	17	15	1.00	0.88	0.94	8	7	1	0.13	0.14	0.13
Doc 3	0	20	0	0	0	0	18	20	14	0.78	0.70	0.74
Doc 4	9	9	6	0.67	0.67	0.67	18	18	17	0.94	0.94	0.94
Doc 5	0	7	0	0	0	0	28	25	15	0.54	0.60	0.57
Doc 6	12	0	0	0	0	0	19	0	0	0	0	0
Doc 7	2	5	1	0.50	0.20	0.29	23	23	14	0.61	0.61	0.61
Doc 8	0	28	0	0	0	0	23	14	14	0.61	1.00	0.76
Average				0.54	0.44	0.47				0.47	0.53	0.49

TABLE II
PER-DOCUMENT RESULTS: MULTI-AGENT MCP WITH INSTRUCTIONAL PROMPT

	Highlights						Comments (Words)					
Document	Ref	Pred	Corr	Recall	Prec	F1	Ref	Pred	Corr	Recall	Prec	F1
Doc 1	0	0	0	N/A	N/A	N/A	32	29	25	0.78	0.86	0.82
Doc 2	15	16	15	1.00	0.94	0.97	8	8	8	1.00	1.00	1.00
Doc 3	0	2	0	N/A	0	N/A	18	17	15	0.83	0.88	0.86
Doc 4	9	7	7	0.78	1.00	0.88	18	18	18	1.00	1.00	1.00
Doc 5	0	0	0	N/A	N/A	N/A	28	26	24	0.86	0.92	0.89
Doc 6	12	12	12	1.00	1.00	1.00	19	18	14	0.74	0.78	0.76
Doc 7	2	2	2	1.00	1.00	1.00	23	21	19	0.83	0.90	0.86
Doc 8	0	0	0	N/A	N/A	N/A	23	24	21	0.91	0.88	0.89
Average				0.94	0.98	0.96				0.87	0.90	0.88

comment digitization, with improvements observed on nearly every document where annotations are present.

While the dataset size is limited, the improvements are large in magnitude and consistent in direction. In particular, highlight F1 increases from approximately 0.47 in the baseline to 0.96 in the full system, and comment F1 improves from approximately 0.49 to 0.88. Given the paired nature of the evaluation and the absence of regressions on any document with non-trivial annotations, these results provide strong evidence that architectural structure materially improves performance. The effect size dominates potential noise introduced by manual evaluation or document variability.

B. Key Learnings

From the results we can see that end-to-end multi-modal generation is brittle for complex tasks. Even capable models frequently hallucinate, misinterpret structures, or refuse to process dense annotations. Introducing specialized extraction tools improves performance only when the orchestrator is explicitly constrained to use them. The failed intermediate configuration demonstrates that tool availability alone is insufficient; reliable behavior requires explicit instructional guidance.

Most importantly, the results show that a system design which supports task decomposition can transform an incompetent system into a capable one. The ReAct-style prompt resource plays a critical role by enforcing step-by-step workflows, validating empty cases, and preventing the orchestrator from bypassing tools. These findings support the central hypothesis of this work: alignment and reliability in image-to-document translation are driven primarily by system design rather than model capability.

C. Implementation Constraints

During system development and evaluation, we encountered several implementation-level constraints that influenced system design but did not materially affect the comparative validity of our results.

First, not all public web platforms preserve image fidelity when hosting user-uploaded files. While services such as Google Drive retain original image resolution and compression characteristics, other platforms (e.g., Imgur) apply lossy compression and resampling. This degradation can negatively impact OCR and visual annotation extraction performance, particularly for fine-grained handwritten marks. Nevertheless, because all evaluated models were exposed to identical inputs under the same hosting conditions, relative performance comparisons remain valid. The observed performance gap between GPT-based and Claude-based approaches persists despite these fidelity constraints.

Second, comments created programmatically via the Google Drive API v3 `addComment` endpoint are not visually anchored to specific text ranges in the Google Docs user interface. Although such comments appear correctly in the document's global comment list, they are displayed as referring to "original content deleted" rather than highlighting the intended text span. This behavior reflects a limitation of the Google Docs API rather than an error in the agentic system logic.

Finally, while our system design includes structured prompt playbooks intended to be exposed as MCP resources, limitations in Claude's orchestration abilities left these MCP resources inaccessible. As a result, prompt logic was manually embedded within the agent workflow. Although this reduced modularity and observability, it does not affect the functional

behavior of the evaluated system variants and therefore does not invalidate the ablation results presented in this section.

Overall, these constraints primarily affect system presentation and implementation convenience rather than the core research objective of this work: evaluating how architectural design and prompt structuring influence LLM tool-use reliability and document editing correctness.

a) Future Extensions.: The current system focuses on comments and highlights, which exhibit the base case for handwritten annotation tasks. A natural extension is to support more complex editorial marks commonly found in copy-edited documents, such as deletion symbols, insertion carets, and other proofreading shorthand. These annotations encode richer intent and would require additional tooling and association logic.

Beyond text documents, the proposed architecture generalizes naturally to other visual mediums. Presentation slides, graphic design mockups, and app wireframes all contain base digital content that is often hand-annotated by editors. By extending the tooling layer and adapting the instructional prompt playbook, the same MCP-based framework could support annotation-to-action pipelines across these domains. This suggests that the system described here represents a foundational approach for translating human visual intent into structured digital artifacts across a wide range of modern workflows.

VII. Conclusion

This work investigates how architectural structure influences the reliability of digitizing handwritten annotations into digital documents. Rather than improving visual recognition, we focus on the post-perceptual problem of translating noisy, ambiguous handwritten feedback into structured digital artifacts. Through a sequence of controlled architectural ablations, we show that end-to-end multi-modal generation is brittle for this task, while agentic decomposition significantly improves alignment accuracy.

Our results demonstrate that introducing specialized extraction tools alone is insufficient. Reliable performance emerges only when tool usage is explicitly enforced through instructional constraints. By combining a multi-agent MCP architecture with a ReAct-style prompt playbook, the system achieves substantial and consistent gains in both highlight detection and comment digitization, avoiding common failure modes such as hallucinated annotations, skipped steps, and model refusals.

Beyond the specific task studied, this work highlights a broader systems insight: for complex image-to-structure translation problems, reliability is driven more by orchestration and constraint than by raw model capability. The proposed MCP-based framework provides a general foundation for translating human visual intent into structured digital representations, with potential extensions to copy-edited documents, design artifacts, and other annotated visual media.

In sum, this project demonstrates that architecture is all you need to make complex visual-to-digital translations work in practice.

References

[1] P. A. Mueller and D. M. Oppenheimer, "The pen is mightier than the keyboard: Advantages of longhand over laptop note taking," *Psychological Science*, vol. 25, no. 6, pp. 1159–1168, 2014.

[2] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63–84, Jan. 2000.

[3] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2009.

[4] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. Int. Conf. Document Analysis and Recognition (ICDAR)*, Curitiba, Brazil, 2007, pp. 629–633.

[5] S. Marinai and H. Fujisawa, *Document Analysis Systems*. Berlin, Germany: Springer, 2008.

[6] T. Schick *et al.*, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[7] S. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive APIs," *arXiv preprint arXiv:2305.15334*, 2023.

[8] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2023.

[9] N. Shinn, J. Labash, and L. Zettlemoyer, "Reflexion: Language agents with verbal reinforcement learning," *arXiv preprint arXiv:2303.11366*, 2023.

[10] Z. Liu *et al.*, "AgentBench: Evaluating LLMs as agents," *arXiv preprint arXiv:2308.03688*, 2024.

[11] Q. Wu *et al.*, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023.

[12] Y. Xu *et al.*, "LayoutLM: Pre-training of text and layout for document image understanding," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2020, pp. 1192–1200.

[13] LMArena, "Image Edit Arena," LMArena.ai. [Online]. Available: https://lmarena.ai/leaderboard/vision. Accessed: Dec. 17, 2025.

[14] A. Bonus, "google-docs-mcp," GitHub repository, 2025. [Online]. Available: https://github.com/a-bonus/google-docs-mcp.