Holo: Gestured-Controlled Generative-AI Painting Application

Final Report

CS 450

Spring 2022

Zac Cowan zcowan@bellarmine.edu

Spring 2025

Holo Repository

https://github.com/zaccowan/Holo

Executive Summary

Project Holo is a multimodal artificial intelligence (AI) system that integrates multiple AI technologies into a dynamic, hand-gesture-controlled canvas painting and image generation application. With the increasing integration of AI in consumer applications, Holo provides users with an intuitive and interactive platform for AI-driven image creation.

Holo is written in python but can be compiled down to C code for slightly increased performance. The graphical user interface (GUI) utilizes a library called “tkinter” with a custom theme wrapper called “custom tkinter.” Holo users can create a sketch using drawing tools such as a pen brush, fill tool, transform, and rectangle tool, and/or enter a text-based image generation prompt. Upon initiating the "Generate AI Image" function, Holo processes the provided inputs via an API call, using a variety of selectable AI models which convert the sketch and/or prompt into an image that appears in the designated output tab.

Holo supports standard interaction methods, including keyboard, mouse, and tablet pen input. Additionally, it enhances user engagement by incorporating a projector-interface and in-app hand tracking using MediaPipe – a Google solution suite enabling hand tracking and pose estimation in real time video feeds. The local hand position of a user in frame, derived from the MediaPipe hand solution, is mapped to screen space to control the on-screen cursor, while gesture-based controls — such as pinching the index finger and thumb to emulate a mouse press — enable a seamless, touch-free interaction experience. This innovative approach makes Holo a versatile tool for AI-assisted digital art creation, expanding the possibilities of human-computer interaction in creative applications.

I. Project / Problem Introduction

Holo seeks to explore two primary questions. Firstly, how Human Computer Interactions (HCI) be expanded to allow for more dynamic and natural means of interacting with computers? Secondly, how can Artificial Intelligence (AI) be naturally integrated into a human workflow to enhance productivity? Holo explores these two questions through the creation of a Gesture Controlled Generative-AI Painting Application.

The Gesture Controlled component of Holo pushes on the conventional keyboard, mouse, and display, means of interacting with our computers. It uses hand tracking and gesture recognition to emulate events like mouse movement, mouse click, and mouse drag. Holo also implements speech recognition to enable hands-off typing into the system.

The Generative-AI component of Holo explores how humans can be aided in image creation by AI tools to increase throughput on the creative process and bring ideas to life in an accessible manner. The pipeline created by Holo allows for a sketch and/or prompt to be used to guide image creation.

Ultimately, Holo, in its current standing, is a proof of concept. The future direction of the technology in Holo is vast, the only remaining question is how Holo can be reimplemented to become an effective tool for Human Computer Interaction to a variety of users. At the conclusion of this project, Holo currently stands a loosely creative and intriguing application that demonstrates simply promising technology that could be used by anyone who uses a computer.

II. Methods

The GUI for Holo was designed to have three major sections: Canvas, AI Image Generation, and Webcam / Gesture Control. The final graphical user interface can be seen below in Fig 1 – 3. The array of canvas tools is large and accessible on the side of the application, The AI Image Generation Section allows users to choose the AI model Replicate with use and copy the resulting image to the canvas for further iterations and refinements. The Webcam section has a variety of controls to fine turn the MediaPipe hand tracking and gestures discussed later in this section.

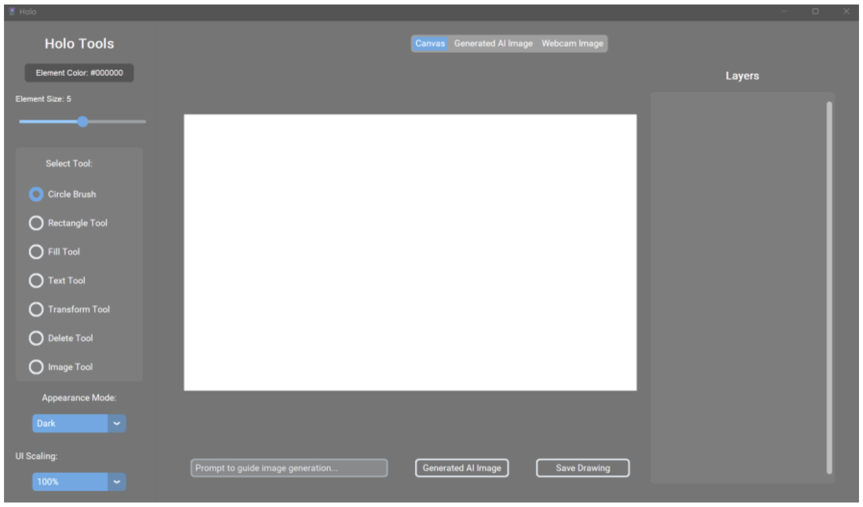


Fig 1 – Canvas screen within Holo.

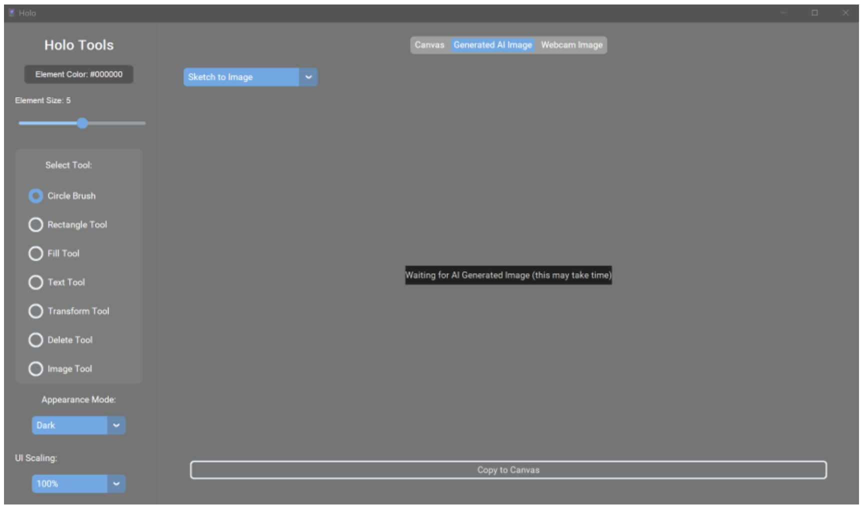


Fig 2 – AI Generation Screen in Holo

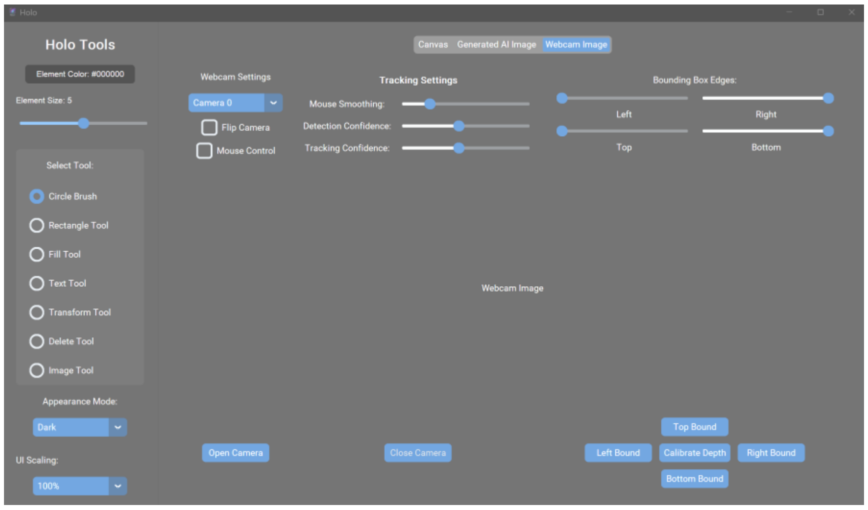


Fig 3 – Webcam and Controls in Holo.

Holo, written in Python 3.12, is dependent on a variety of external libraries. For the graphical interface I opted to use Tkinter, python bindings to the tk gui library. More specifically I use a custom theme built on top of the Tkinter binding, called CustomTkinter, that essentially just styles components like buttons and text to allow me to give Holo a modern appearance without having to spend time with styling myself. Tkinter has a few major features that made it a good fit for building Holo: grid layout management allowed a logic position of Graphical components in the interface, canvas elements were used to hold brush strokes, images, and to display the webcam feed, user interaction on buttons can call a specified function and provided event state like mouse position and more. In terms of user interface, Tkinter provided all the tools I need to build Holo.

For AI Image Generation, I opted for a cloud-based approach via Replicate over running models locally on a user's machine. There were two primary reasons for this choice. Firstly, Replicate manages the machines used to run the AI Models and provides computing power beyond what most user machines can supply. Secondly, with Replicate, Holo has a very modular design in terms of what generation model to use. Since the input parameters are a base64 encoded image and/or a text prompt, Holo can utilize any of the current or future models that Replicate may support. A new model can be made available for Holo to use by a simple change to one line of the API call that occurs when the Generate image button is pressed. There are often small fees per API call, all of which Replicate allows a user to manage in one place, regardless of model. Image generation is tailored to the users sketch and prompt, see the flowchart below for a visual of this process.

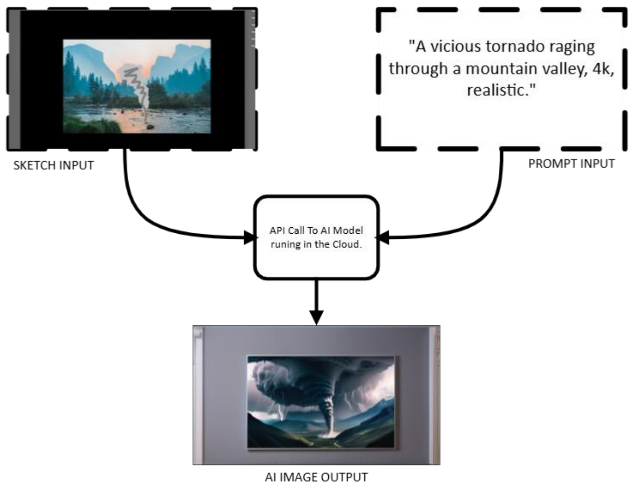

"A vicious tornado raging through a mountain valley, 4k, realistic."
SKETCH INPUT
PROMPT INPUT
API Call To AI Model runing in the Cloud.
AI IMAGE OUTPUT

Fig 4 – Input pipeline into AI Image Generation Model.

For Speech Recognition, a local, neural-network free model is used called Vosk. All of the files needed for Vosk model to operate locally and offline are found in a set of files consuming less than 100MB of memory. PyAudio is used alongside to help process the audio for use in speech-to-text with Vosk.

For Gesture Control OpenCV, PyAutoGUI, and MediaPipe are used. Open Computer Vision (OpenCV) is a necessary tool for managing the video feed from the webcam. While OpenCV is useful for computer vision tasks, all of the heavy lifting for gesture control is performed by MediaPipe. MediaPipe is a Google solution that enables hand tracking in real time. Mediapipe detects and tracks regions of the hand, maps the hand to the corresponding bone structure seen in Fig 5, and allows Holo to map hand position and gestures to various functions within the computer. For Holo, pinching index finger and thumb together emulates a mouse click, via PyAutoGUI, while moving the hand around the video frame moves the mouse correspondingly, also via PyAutoGUI. MediaPipe do provide a solution for recognizing various gestures – see the Results section of this paper for future directions on this component of MediaPipe.

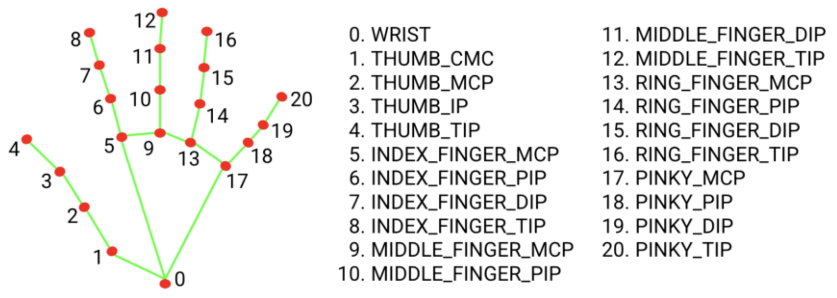


Fig 5 – Hand bone structure via MediaPipe documentation.

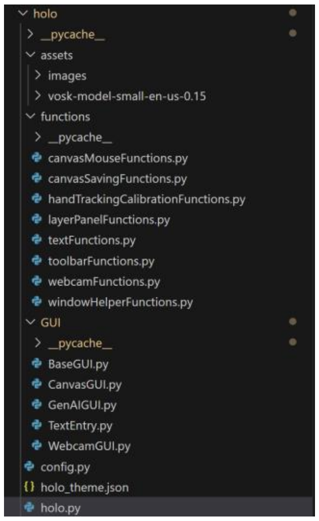
Various Python libraries are necessary for Holo’s function, of which are not specific to any one feature. A brief overview of these tools can be seen below:

- Threading – Used to run the camera loop and speech recognition in separate threads to avoid blocking the main GUI thread.
- NumPy – Used for efficient array operations and interpolation of coordinates.
- Pillow – Python Imaging Library used for image preparation and simple manipulation.

III. Results

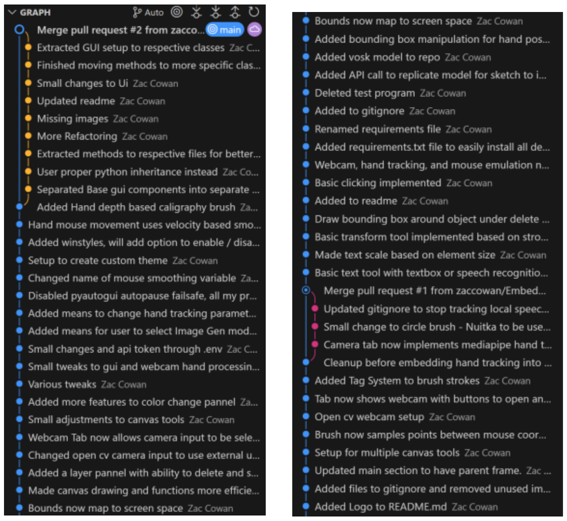
Holo was successful in its designed task of enabling Gesture-Controlled Generative-AI Painting for users. Though as will be discussed in more detail, Holo in its current state is not nearly as useful as the future direction it could be taken. The largest two limitations with Holo are the learning curve for gesture control and the quality of outputs by current Image Generation models.

In my development, I stayed mostly ahead of schedule. Development was much more rapid than expected, largely due to the use of assistive tools like GitHub Copilot which helped me perform menial and repetitive tasks nearly instantly. GitHub Copilot did help introduce some optimizations to code, but, still, the bulk of the function of Holo had to be manually written. There were some complications that slowed development. At one point I thought the MediaPipe hand tracking and system control were laggy due to lack of system resources during a computational heavy task. After reading documentation for PyAutoGUI I learned that there was an intentionally designed failsafe that introduced a 0.1 second delay after a moveTo function was called. Disabling this failsafe alleviated all the lag previously present and left me with some additional optimizations that were made in attempt to fix the unknown issue. The second biggest setback was the need to refactor the code as the Holo program grew to well over two thousand lines of code. During this refactoring I moved code segments to files that corresponded to their functions. The new refactored structure can be seen below as well as a list of all the GitHub commits to give insight into development flow. As future improvement to my development lifecycle, I could better plan the software structure to maintain readability as the program grows in size.



```
holo
  __pycache__
  assets
    images
    vosk-model-small-en-us-0.15
  functions
    __pycache__
    canvasMouseFunctions.py
    canvasSavingFunctions.py
    handTrackingCalibrationFunctions.py
    layerPanelFunctions.py
    textFunctions.py
    toolbarFunctions.py
    webcamFunctions.py
    windowHelperFunctions.py
  GUI
    __pycache__
    BaseGUI.py
    CanvasGUI.py
    GenAIGUI.py
    TextEntry.py
    WebcamGUI.py
  config.py
  holo_theme.json
  holo.py
```

Fig 6 – Holo director structure



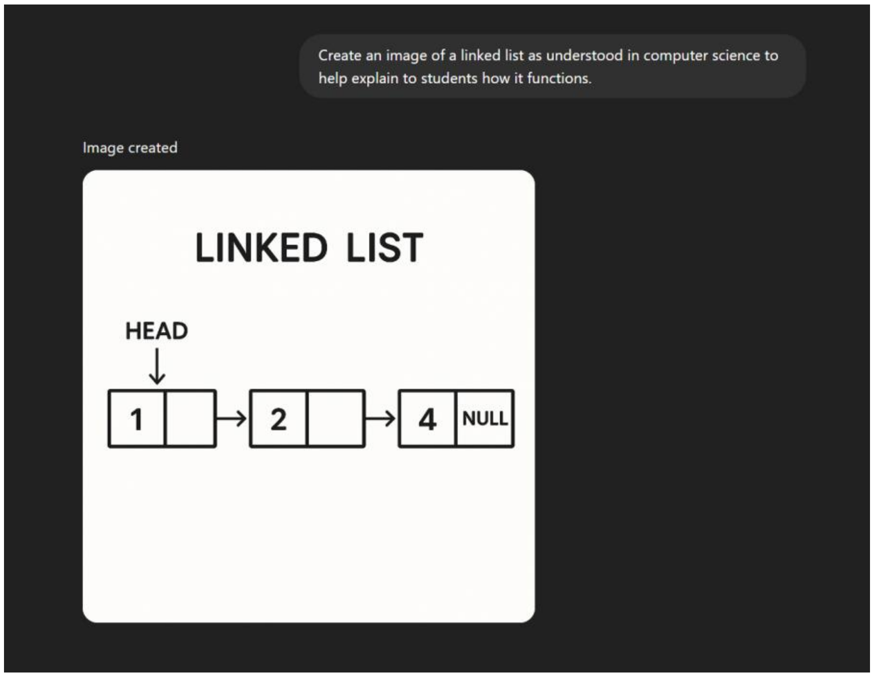
```
Merge pull request #2 from zacco...
Extracted GUI setup to respective classes  Zac C...
Finished moving methods to more specific clas...
Small changes to Ui  Zac Cowan
Updated readme  Zac Cowan
Missing images  Zac Cowan
More Refactoring  Zac Cowan
Extracted methods to respective files for better...
User proper python inheritance instead  Zac Co...
Separated Base gui components into separate ...
Added Hand depth based caligraphy brush  Za...
Hand mouse movement uses velocity based smo...
Added winstyles, will add option to enable / disa...
Setup to create custom theme  Zac Cowan
Changed name of mouse smoothing variable  Za...
Disabled pyautogui autopause failsafe, all my pr...
Added means to change hand tracking paramet...
Added means for user to select Image Gen mod...
Small changes and api token through .env  Zac C...
Small tweaks to gui and webcam hand processin...
Various tweaks  Zac Cowan
Added more features to color change pannel  Za...
Small adjustments to canvas tools  Zac Cowan
Webcam Tab now allows camera input to be sele...
Changed open cv camera input to use external u...
Added a layer pannel with ability to delete and s...
Made canvas drawing and functions more efficie...
Bounds now map to screen space  Zac Cowan
```

```
Bounds now map to screen space  Zac Cowan
Added bounding box manipulation for hand pos...
Added vosk model to repo  Zac Cowan
Added API call to replicate model for sketch to i...
Deleted test program  Zac Cowan
Added to gitignore  Zac Cowan
Renamed requirements file  Zac Cowan
Added requirements.txt file to easily install all de...
Webcam, hand tracking, and mouse emulation n...
Basic clicking implemented  Zac Cowan
Added to readme  Zac Cowan
Draw bounding box around object under delete ...
Basic transform tool implemented based on stro...
Made text scale based on element size  Zac Cowan
Basic text tool with textbox or speech recognitio...
Merge pull request #1 from zaccowan/Embed...
Updated gitignore to stop tracking local speec...
Small change to circle brush - Nuitka to be use...
Camera tab now implements mediapipe hand t...
Cleanup before embedding hand tracking into ...
Added Tag System to brush strokes  Zac Cowan
Tab now shows webcam with buttons to open an...
Open cv webcam setup  Zac Cowan
Brush now samples points between mouse coor...
Setup for multiple canvas tools  Zac Cowan
Updated main section to have parent frame.  Zac ...
Added files to gitignore and removed unused im...
Added Logo to README.md  Zac Cowan
```

Fig 7 – Commits to the Holo repository (top left is most recent)

As previously mentioned, the technology in Holo has promising future directions. The Hand Gesture Control component of Holo could be isolated into its own application tailored to teachers, presenters, and many other professionals allowing hands off interaction with their computers. This means you could gain hands off control of any system: PowerPoint slides, brainstorming applications, time management tools, creative applications. Holo could be specially tuned to map certain hand gestures to complex key binds, functions, or other macros in a system.

Additionally, as image generation models grow, Holo in its current state could be better suited for generating technical images for teachers or creatives that correspond better to the mental images within one's head. Even recently ChatGPT has gained the ability to generate images like a linked list. Since Holo is modular in terms of AI Image Generation model, a newer task specific model could be utilized by Holo to create better images.



Create an image of a linked list as understood in computer science to help explain to students how it functions.

Image created

Fig 8 – A prompt and response by ChatGPT image generation model on April 25th, 2025.

IV. Conclusion

Human Computer Interaction (HCI)

With any new means of interacting with a system, learning can take some time. During a Bellarmine Maker Fair in March 2025, many high school students were able to use Holo. Although the interface is intuitive, becoming proficient in drawing with hand gesture control can take time.

Artificial Intelligence (AI) Integration

Holo allows users to generate images using their sketches and prompts, choosing the model they desire for a given scenario. Holo also allows users to iterate on previously generated images: users may sketch over any given image (AI generated included) to better guide the models to get their desired output.

With AI systems like Holo, the most effective users learn to predict the style of a given AI model and in turn can edit their prompts for the best results.

Sketch and Text to Image Generative Model

As with many current AI systems, the model training data and architecture strongly correlates to the desirability of the output. The ability to prompt and guide Holo’s supported AI models with images and text does play some part in the final output although choosing the right model for image generation makes a significant difference.

References

Mediapipe

https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker

Custom Tkinter

https://customtkinter.tomschimansky.com

Replicate

https://replicate.com/docs

Vosk

https://alphacephei.com/vosk

ChatGPT

https://chatgpt.com