robotics

Article

ChatGPT-Enabled daVinci Surgical Robot Prototype: Advancements and Limitations

Abhilash Pandya

Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202, USA; apandya@wayne.edu

Abstract: The daVinci Surgical Robot has revolutionized minimally invasive surgery by enabling greater accuracy and less-invasive procedures. However, the system lacks the advanced features and autonomy necessary for it to function as a true partner. To enhance its usability, we introduce the implementation of a ChatGPT-based natural language robot interface. Overall, our integration of a ChatGPT-enabled daVinci Surgical Robot has potential to expand the utility of the surgical platform by supplying a more accessible interface. Our system can listen to the operator speak and, through the ChatGPT-enabled interface, translate the sentence and context to execute specific commands to alter the robot's behavior or to activate certain features. For instance, the surgeon could say (even in Spanish) "please track my left tool" and the system will translate the sentence into a specific track command. This specific error-checked command will then be sent to the hardware, which will respond by controlling the camera of the system to continuously adjust and center the left tool in the field of view. We have implemented many commands, including "Find my tools" (tools that are not in the field of view) or start/stop recording, that can be triggered based on a natural conversational context. Here, we present the details of our prototype system, give some accuracy results, and explore its potential implications and limitations. We also discuss how artificial intelligence tools (such as ChatGPT) of the future could be leveraged by robotic surgeons to reduce errors and enhance the efficiency and safety of surgical procedures and even ask for help.

Keywords: surgical robotics; natural language processing; ChatGPT

Citation: Pandya, A. ChatGPT-Enabled daVinci Surgical Robot Prototype: Advancements and Limitations. *Robotics* **2023**, *12*, 97. https://doi.org/10.3390/robotics12040097

Academic Editor: Saïd Zeghloul

Received: 26 May 2023
Revised: 25 June 2023
Accepted: 3 July 2023
Published: 6 July 2023

1. Introduction

The daVinci Surgical Robot is a revolutionary technology that has greatly improved the field of minimally invasive surgery. It allows for greater precision and dexterity during surgical procedures. Natural language processing (NLP) is a subfield of artificial intelligence that focuses on understanding and generating natural language. Recent advancements in NLP, specifically the ChatGPT (Generative Pre-trained Transformer) language model, have enabled the creation of conversational interfaces that can understand and respond to human language. It is trained using data from the internet and can translate or even simplify language, summarize text, code, and even make robots smarter.

The integration of a natural language ChatGPT-enabled interface for the daVinci Surgical Robot has the potential to enhance the field of surgery. By creating a more intuitive and user-friendly interface, it can potentially improve the safety and efficiency of surgeries, while also reducing the cognitive load on surgeons. Can the artificial intelligence behind ChatGPT be applied to make surgeries safer and faster?

In this paper, we describe a basic implementation of ChatGPT directly interfaced with the daVinci robot. It is a low-level implementation that limits the output of ChatGPT to specific commands that can be executed by the robot. It does have the capability for domain-specific training (say on a particular type of surgery) with open dialog, but in this paper, we limit it to specific commands to control the hardware. We primarily focus on explaining the integration of AI with the daVinci system and do not include a user study to verify or showcase its effectiveness. We also discuss the potential avenues of research and development that this interface could open for the future of robotic surgery. This may be a steppingstone to a natural language robotic platform that partners with the surgeon for a more intuitive and efficient system that enhances patient outcomes. Future iterations of this GPT–robot merging of technology could allow collaboration with surgeons in more profound ways, including in surgical education [1], preplanning surgeries [2], executing specific steps of the surgery [3], and keeping the surgeon informed about any anomalies in the scene or the patient's data [4,5].

Operations in surgical robotic platforms such as the da Vinci have been facilitated by means of foot pedals, hardware buttons, and touchpads. This interface can be overwhelming as the surgeon has to manually control all aspects of the interaction. To alleviate some of this burden, we have previously implemented a natural language interface to the daVinci [6] using Vosk [7]. Vosk is a language modeling system that can convert microphone input into text. It is still used in the current implementation in this capacity. Vosk is not an extensive language model and requires keyword phrases.

The AESOP was among the earliest voice-controlled systems utilized in the operating room. It is a seven-degree-of-freedom arm employed to manipulate a laparoscopic surgery camera [8–12]. It had extremely limited movements and specific commands for moving the endoscope in increments. With this voice-controlled robot, surgeons could use either a joystick or voice control as required.

ChatGPT has shown promise in robotics applications which require strategies for robot design and of designing high-level functions and libraries [13]. The main idea is that the prompts used for ChatGPT are critical and this group has created a site (PromptCraft) that explores and collectively develops/engineers appropriate queries.

ChatGPT in the Medical Field

ChatGPT has already been used in medicine for various applications, such as medical chatbots, virtual medical assistants, and medical language processing [14]. For example, ChatGPT has been employed to provide conversational assistance to patients, generate clinical reports, help physicians with diagnosis and treatment planning, etc. [15]. It has also been utilized in medical research, such as analyzing electronic medical records and predicting patient outcomes.

ChatGPT has hundreds of billions of parameters and has passed the United States Medical Licensing Examination (USMLE) at a third-year medical student level [16]. More importantly, its responses were easily interpretable with clear logic that could be explained. ChatGPT has also been suggested for clinical decision making [14]. These systems have already been used to simplify the medical jargon used in radiology reports to make it easy for patients to understand [17].

Can ChatGPT also be potentially used in surgery? Bhattacharya et al. [2] suggest using ChatGPT as a preoperative surgical planning system. The system could be used to analyze potential issues and complications, for complex procedures, and to provide checklists and options for the surgeon.

Here, we provide a novel avenue for using ChatGPT in the surgical setting as a user interface for the daVinci surgical system. In our protype system, the user can give commands with a natural language syntax and execute a basic autonomous camera [18,19] and other tasks.

2. Materials and Methods

2.1. Baseline Commands

The baseline commands that have been created and can directly be issued to the daVinci hardware include, for example, taking a picture, starting and stopping a video recording, toggling on/off an autonomous camera system to follow the tools, finding the surgeon's tools when out of the field of view, tracking the left/middle/right tool, maintaining a point in space specified by the right or left tool position within the field of view, and changing the settings associated with zoom control. There are many other features which can be programmed. These commands can be triggered via keyboard, keypad, or (as explained in this article) by natural language processing (Figure 1).

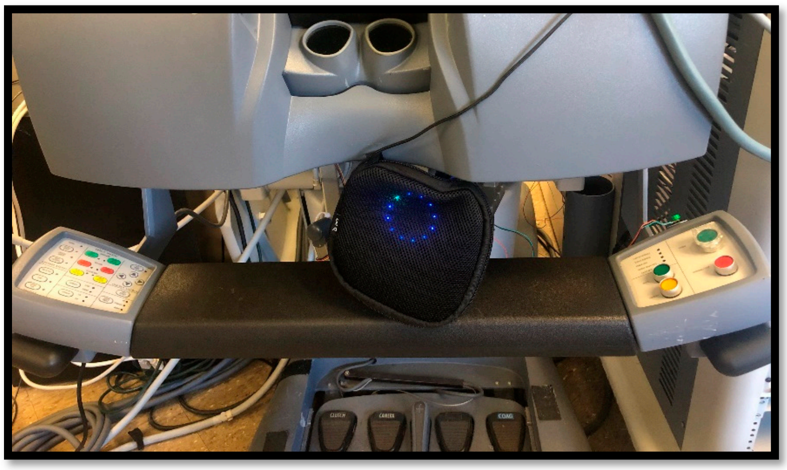


Figure 1. The daVinci Standard outfitted with a microphone. We have also modified the system to add a head sensor and buttons on the hand controllers to activate the camera and tool clutching. These buttons could also be used for voice activation.

As illustrated in Figure 2, our system receives input from a microphone near the operator, preprocesses the message, and sends it to the Chat-GPT language model. The model is trained (by giving it a few examples in the prompt) to respond specifically to only the possible commands and the output is checked to ensure this. The responses are then translated to command the hardware. The system provides a beep or buzz as feedback to the surgeon, indicating success or failure. Although our current feedback to the surgeon is only via sound and voice, we envision that augmented reality techniques could also be used as feedback for the surgeon.

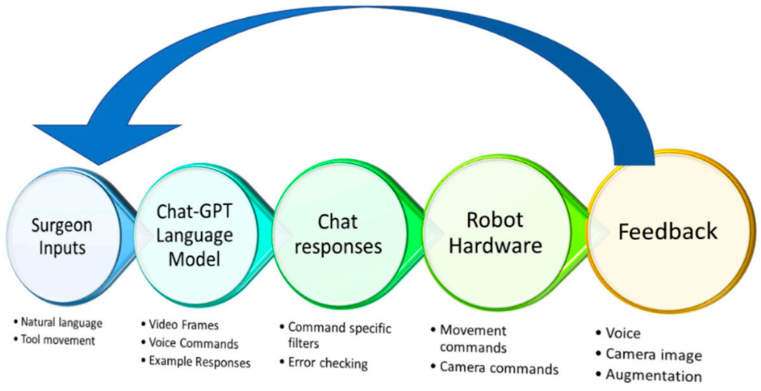


Figure 2. Overview of the system and ChatGPT integration.

2.2. The DVRK/Robot Operating System Interface

Our research laboratory has access to the da Vinci Standard Surgical System, which has been modified to work with the da Vinci Research Kit (DVRK) [20,21]. The DVRK allows the use of open-source software and hardware control boxes to command and receive feedback from the robotic system.

The research employs this equipment in conjunction with the Robot Operating System (ROS) [22] software framework. The Robot Operating System (ROS) is an open-source middleware used for building robotics applications. It provides a set of tools and libraries for building complex robotic systems, including drivers, communication protocols, and algorithms. ROS is designed as a distributed system, allowing the communication and collaboration of multiple nodes running on different computers. Nodes can publish and subscribe to messages on topics, allowing for easy exchange of information between different components of the system.

In our voice assistant applications for the da Vinci Research Kit (DVRK), we utilize the ROS middleware for direct access to the state information and control capabilities of the robotic system. This allows us to integrate our voice commands with the robot's control system, enabling natural language control of its functions. The voice assistant application consists of multiple ROS nodes that communicate with each other through ROS topics. One node is responsible for processing the voice commands and translating them into ROS messages that are sent to the DVRK control node. The control node then executes the appropriate action based on the received message. Overall, the use of ROS in our voice assistant applications enables seamless integration with the DVRK and provides a powerful toolset for building complex robotics systems. More details of the base natural language processing system are provided here [6].

2.3. Capturing and Preprocessing the Voice Inputs

ReSpeaker Mic Array v2.0, developed by Seeed Studios Inc. in Shenzhen, China, was utilized for testing purposes due to its built-in voice recognition capabilities. The device features a circular array with four microphones to determine the location of acoustic sources and is equipped with hardware and onboard algorithms for far-field voice detection and vocal isolation. The device functions as a USB microphone and tested very well in both noisy and quiet environments. The microphone provides six channels, including processed and raw captures from the onboard microphones and playback of the input audio device through an auxiliary cord connected to a speaker. After we received inputs from the microphone, we pieced the words together until a natural pause in the sentence was heard. This pause indicates a full sentence or command to the system. We then used this fully formed sentence as input to ChatGPT.

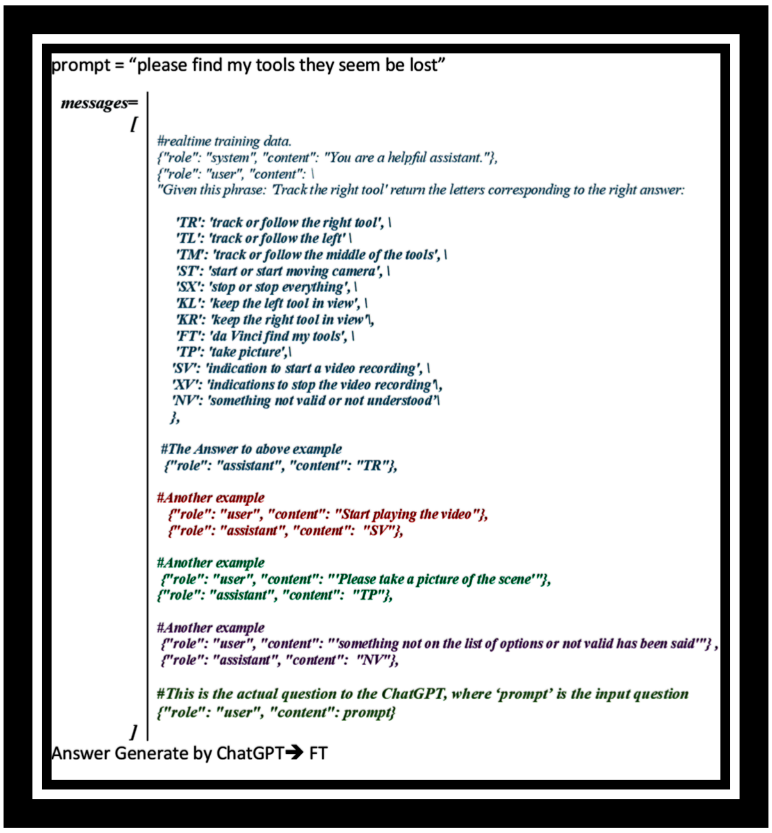
2.4. Asking for Input from ChatGPT

We have created an AskGPT() function which provides a way to interact with the daVinci Surgical Robot using natural language commands. By leveraging the power of the ChatGPT model, it can generate responses to a wide variety of commands. The AskGPT() function takes a prompt as input and generates a response using the OpenAI ChatGPT model. The prompt represents the command that a user wants to execute on the daVinci Surgical Robot, such as "track the right tool". The openai.ChatCompletion.create() method is used to generate a response to the prompt. It takes several parameters, including the model to use (in this case, "gpt-3.5-turbo"), the temperature value to use for generating responses, and a set of messages that provide real-time training data for the model.

The temperature value in a ChatGPT API call represents a parameter that controls the creativity or variability of the responses generated by the model. In the context of the OpenAI GPT-3 language model, the temperature value is used to scale the logits (output of the model) before applying the softmax function to obtain the probability distribution over the vocabulary of possible next tokens. A higher temperature value results in a probability distribution with higher entropy, meaning that the model is more likely to produce more diverse and surprising responses. Conversely, a lower temperature value results in a probability distribution with lower entropy, meaning that the model is more likely to produce more conservative and predictable responses.

In general, this parameter can be dynamically set and could be useful when exploring the space of possible responses, generating creative and diverse text, and encouraging the model to take risks and try new things. Lower temperature values are useful when generating more coherent and consistent text that is closely aligned with the training data and has a more predictable structure.

In Figure 3, the interface message to ChatGPT is shown. The messages parameter (programmatically sent to the ChatGPT interface) is an array of JSON objects that represents a conversation between a user and an assistant. Each message has a role and content field. The role field specifies whether the message is from the user or the assistant, and the content field contains the text of the message. Through this message protocol, we must provide ChatGPT with clear examples of what the expected responses are. In this example, if we do not provide a specific set of outputs, the system can become difficult to control (See Figure 3).



```
prompt = "please find my tools they seem be lost"

messages=
        [
          #realtime training data.
          {"role": "system", "content": "You are a helpful assistant."},
          {"role": "user", "content": \
          "Given this phrase:  Track the right tool' return the letters corresponding to the right answer:

              'TR': 'track or follow the right tool', \
              'TL': 'track or follow the left' \
              'TM': 'track or follow the middle of the tools', \
              'ST': 'start or start moving camera', \
              'SX': 'stop or stop everything', \
              'KL': 'keep the left tool in view', \
              'KR': 'keep the right tool in view'\,
              'FT': 'da Vinci find my tools', \
              'TP': 'take picture',\
             'SV': 'indication to start a video recording', \
             'XV': 'indications to stop the video recording'\,
             'NV': 'something not valid or not understood'\
            },

           #The Answer to above example
            {"role": "assistant", "content": "TR"},

          #Another example
             {"role": "user", "content": "Start playing the video"},
             {"role": "assistant", "content":  "SV"},

          #Another example
           {"role": "user", "content": "'Please take a picture of the scene'"},
          {"role": "assistant", "content":  "TP"},

          #Another example
           {"role": "user", "content": "'something not on the list of options or not valid has been said'"} ,
           {"role": "assistant", "content":  "NV"},

          #This is the actual question to the ChatGPT, where 'prompt' is the input question
          {"role": "user", "content": prompt}
        ]
Answer Generate by ChatGPT➔ FT
```

Figure 3. The message structure that is sent to ChatGPT. Note that several examples are necessary to prompt the specific style of responses needed from ChatGPT.

In the first detailed example, an example prompt is given with an expected answer. The first message in the messages array tells the user that they are interacting with a helpful assistant. The second message is the simulated prompt to the system—"Track the right tool". The next message provides a set of options that ChatGPT can choose from to execute their command, along with corresponding return codes. Then, a message indicating the correct response, "TR", is given. The remaining messages in the messages array are examples of diverse types of prompts that the user might provide, along with the expected response from the ChatGPT model. These are all used as just-in-time training for ChatGPT. Finally, the last message in the array is the prompt that the user provided (from the microphone) for which an answer is expected. Note that the examples given are not an exhaustive list of commands, just a few indicating the general type of answer desired. The input can even be in another language or more elaborately specified with a nuanced context.

2.5. Processing the ChatGPT Responses

Once the openai.ChatCompletion.create() method is called, it generates a response to the provided prompt using the ChatGPT model. The response is returned as a completions object, which contains the generated text as well as some metadata about the response. The function returns one of the options from the list of choices, which corresponds to the action that the calling program should take.

To finetune the responses to specific commands that can be issued with confidence, our code limits the responses of ChatGPT to those that are valid commands. The code defines a dictionary called "choices" which maps a two-letter code to a specific command that ChatGPT can respond with. The commands include actions such as tracking left or right, starting and stopping video recording, finding tools, and taking pictures. As an added check, the script also defines a string variable called "listofpossiblecommands" which contains a space-separated list of all the valid two-letter codes. These codes are used to check if the response from ChatGPT is a valid command. If the response is not a valid command, then the script returns the "NV" index, which stands for "not valid".

2.6. Triggering Hardware Commands

Using the ROS node structure, the two letters returned by ChatGPT represent a possible command that can be executed on the daVinci hardware. When a command is triggered, a sequence of actions is initiated through the assistant_bridge node to activate the hardware. For instance, if ChatGPT is prompted with "Can you please track my right tool", the system will return the "TR" index, which corresponds to the very specific "daVinci track right" command. This command is sent to the "assistant/autocamera/track" node, which in turn sends a message to the/assistant_bridge node. Finally, the/assistant_bridge node sends a message to the dVRK nodes that control the hardware in a loop, resulting in the camera arm being continually adjusted to keep the left tool in the center of the field of view. Commands to find tools that may not be in the field of view, take a picture of the current scene, start and stop taking a movie of the scene, etc., are initiated similarly. What ChatGPT adds to this basic framework is the ability to speak naturally, without a key word or phrase in a specific order. The system is also able to operate the commands even if the phrase being uttered is in a different language that is known to ChatGPT (Figure 4).

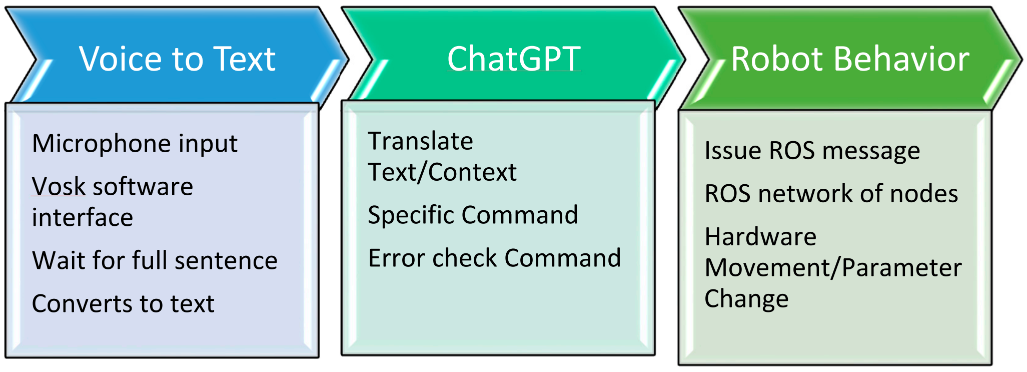


Figure 4. There is a ROS node structure for triggering hardware commands to the robot. The main item to note is that the output of ChatGPT is filtered and then commands are triggered within the ROS node tree that change the behavior of the hardware.

2.7. Testing the System

As a way of testing the usability of the system, we wanted to see how accurately the entire system works end-to-end (voice input, text conversion, sentence formation, ChatGPT send, ChatGPT responses, and robot behavior). To do this, we used the following paragraph as a continuous input to the system:

"Good morning. I would like you to start recording a video. Now, please start the autocamera system. At this point, please track my left tool. Now track my right tool. Now, just track the middle of my tools. Keep the point of my left tool in view. I seem to have lost the view of my tools; can you please find my tools? Okay, stop recording the video. Please keep the point of the right tool in view. Can you please take a picture now? I need to concentrate on this region, please stop moving the camera."

After every sentence, we would wait for confirmation before saying the next sentence. In addition, we also wanted to know the effect of the temperature parameter on the Chat-

GPT engine. So, we tested this paragraph (5 repetitions each) for each of the 5 temperature settings (0.1, 0.2, 0.5, 0.75, 1.0). For these settings, we collected how many times the system did not produce the correct response.

3. Results/Discussion

The results of the testing described above are shown in Table 1. Although this was a quick usability test, the results of 275 sentences processed show 94.2% correctness. The errors included not being able to understand what was said and even giving the wrong command. The most common error was "NV", not valid; however, there were errors where the system responded with a keep left or keep right when a track left or track right command was given. The errors were also due to transcription errors from voice to text. These results are preliminary and need a full-blown user study. Before we perform a user study, our group is interested in solving the issue of the time delay of the system.

Table 1. Table of incorrect responses by the system. The results are the end-to-end result of voice input, text conversion, sentence formation, ChatGPT send, ChatGPT responses, and robot behavior. The end-to-end accuracy for 275 commands to the system was 94.2%.

ChatGPT Temperature	**Number Incorrect**	**Total Sentences**
0.1	5	55
0.2	2	55
0.5	2	55
0.75	3	55
1.0	4	55
Total	16	275
Percentage	5.8%	94.2%

For the contingency table above, we find that the chi-squared statistic *p* value is 2.280 and the corresponding *p*-value is approximately 0.681. Since the obtained *p*-value (0.681) is greater than the significance level (0.05), we do not reject the null hypothesis (that there is no difference between the groups). This shows that there is no significant difference between the conditions based on the provided data. Hence, temperature in this dataset had no effect.

An issue with the current implementation is that there is a delay of about 1–2 s from signal capture and send to when ChatGPT generates a response. This delay is due to the nature of the GPT-3.5 model, which is a large deep learning model that requires significant computation time to generate responses. In addition, there is network transmission delay. There are several ways in which this delay could be mitigated. One approach is to use a smaller, faster model, such as GPT-2 or a custom-trained language model that is optimized for the specific task. With new tools now available like CustomGPT [23], which allow you to input and create a system that works only on your own data, this will likely be possible.

It is, in theory, possible to have a local copy of a large language model which can alleviate the network lag. This is known as on-premises or on-device deployment, where the model is stored locally on a computer or server, rather than accessed through a cloud-based API. There are several benefits to on-premises deployment, including improved response times due to reduced network latency, improved privacy and security, and greater control over the system's configuration and resources. However, on-premises deployment also requires more resources and expertise to set up and maintain the system. Other options for on-premises deployment include training your own language model using open-source tools like Hugging Face Transformers or Google's TensorFlow. Integrating a custom TensorFlow model into ChatGPT requires significant programming expertise, but it can offer more control over the model's behavior and potentially better performance than using a pre-built model from a third-party service.

We ran an experiment to test the speed of execution between a smaller LLM (70.8 million parameters) running on a an RTX 2070 GPU with 8GB of VRAM [24] and the full ChatGPT (version 3.5) LLM running from the web. The model we used locally executed, on average, a command in 300 milliseconds, whereas a command using the full ChatGPT model on the web took, on average, 580 milliseconds. Running on a local machine with a more focused model is almost twice as fast as running on the web. More research on the trade-off between the privacy, accuracy, and speed of LLMs is needed.

4. Conclusions

We believe that a robotic embodiment is crucial as GPT technology matures. An artificial general intelligence, which we may be approaching, without actuation is comparable to cornflakes without milk. The development of chatbots using language models such as ChatGPT has opened exciting possibilities for creating conversational interfaces that can be integrated with various robotics-related applications. Our implementation is the first and most basic step in this evolution.

The surgical robotics space has extremely stringent safety and operational restrictions. There are challenges that need to be addressed, such as privacy, network latency, errors, and the limited control over the responses of the model. One way to overcome these issues is by training a local copy of the GPT model using TensorFlow and integrating it into the chatbot. Tools for customizable datasets are now available [23]. Such systems will allow for faster response times and domain-specific control over the responses generated by the model.

For clinicians, no level of error in the translation of intent is acceptable. There are many improvements needed in this technology before it is validated for clinical use. Speech-to-text conversion remains a critical issue. Additionally, it is important to carefully design the conversation flow and supply specific instructions and limitations to the model, as seen in the prompt example shown here. This is a new field of "prompt engineering" which helps to ensure that the responses generated by the model are valid for the given context and the type and modality of interaction are appropriate for surgical tasks.

The implementation of a natural language ChatGPT-enabled interface for the daVinci Surgical Robot has the potential to significantly improve surgical outcomes and increase accessibility to the technology. Our study provides implementation details along with a basic prototype highlighting the usability of the natural language interface. This is only a scratch on the surface of this field. Further research and development are needed to evaluate the long-term implications of the natural language interface and its potential impact on surgical outcomes.

The use of ChatGPT in aiding surgeons with complex cases through warnings, suggestions and alternatives, patient monitoring, fatigue monitoring, and even surgical tool manipulation could be possible in the near future. With advances in image and video analysis in the ChatGPT framework, this avenue of research development could lead to higher-functioning and more intelligent surgical systems that truly partner with the surgical team. Future studies should evaluate the effectiveness and usability of the natural language interface in surgical settings. We have begun implementing a "talk to me" functionality that will be trained on specific surgical interventions and allow surgeons-in-training to ask questions and receive information from the dataset that the system will be trained on.

Funding: This research received no external funding.

Data Availability Statement: Any data and developed software are available upon request. Please contact the author.

Acknowledgments: We would like to acknowledge Abhishek Shankar, Luay Jawad, Xingyu Zhou, Mohammad Alawad, and David Edelman, for valuable discussions. We would also like to thank Arshdeep Singh Chudda for help with the timing studies of local LLMs.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Long, Y.; Cao, J.; Deguet, A.; Taylor, R.H.; Dou, Q. Integrating artificial intelligence and augmented reality in robotic surgery: An initial dvrk study using a surgical education scenario. In Proceedings of the 2022 International Symposium on Medical Robotics (ISMR), Atlanta, GA, USA, 13–15 April 2022; pp. 1–8.
2. Bhattacharya, K.; Bhattacharya, A.S.; Bhattacharya, N.; Yagnik, V.D.; Garg, P.; Kumar, S. ChatGPT in surgical practice—A New Kid on the Block. *Indian J. Surg.* **2023**, 1–4. [CrossRef]
3. Richter, F.; Shen, S.; Liu, F.; Huang, J.; Funk, E.K.; Orosco, R.K.; Yip, M.C. Autonomous robotic suction to clear the surgical field for hemostasis using image-based blood flow detection. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1383–1390. [CrossRef]
4. Rahbar, M.D.; Ying, H.; Pandya, A. Visual Intelligence: Prediction of Unintentional Surgical-Tool-Induced Bleeding during Robotic and Laparoscopic Surgery. *Robotics* **2021**, *10*, 37. [CrossRef]
5. Rahbar, M.D.; Reisner, L.; Ying, H.; Pandya, A. An entropy-based approach to detect and localize intraoperative bleeding during minimally invasive surgery. *Int. J. Med. Robot. Comput. Assist. Surg.* **2020**, *16*, 1–9. [CrossRef] [PubMed]
6. Elazzazi, M.; Jawad, L.; Hilfi, M.; Pandya, A. A Natural Language Interface for an Autonomous Camera Control System on the da Vinci Surgical Robot. *Robotics* **2022**, *11*, 40. [CrossRef]
7. Vosk Offline Speech Recognition API. Available online: https://alphacephei.com/vosk/ (accessed on 1 May 2022).
8. Mettler, L.; Ibrahim, M.; Jonat, W. One year of experience working with the aid of a robotic assistant (the voice-controlled optic holder AESOP) in gynaecological endoscopic surgery. *Hum. Reprod.* **1998**, *13*, 2748–2750. (In English) [CrossRef] [PubMed]
9. Unger, S.; Unger, H.; Bass, R. AESOP robotic arm. *Surg. Endosc.* **1994**, *8*, 1131. [CrossRef] [PubMed]
10. Allaf, M.E.; Jackman, S.V.; Schulam, P.G.; Cadeddu, J.A.; Lee, B.R.; Moore, R.G.; Kavoussi, L.R. Laparoscopic visual field: Voice vs. foot pedal interfaces for control of the AESOP robot. *Surg. Endosc.* **1998**, *12*, 1415–1418. (In English) [CrossRef]
11. Nathan, C.O.; Chakradeo, V.; Malhotra, K.; D'Agostino, H.; Patwardhan, R. The voice-controlled robotic assist scope holder AESOP for the endoscopic approach to the sella. *Skull Base* **2006**, *16*, 123–131. [CrossRef] [PubMed]
12. Kraft, B.M.; Jäger, C.; Kraft, K.; Leibl, B.J.; Bittner, R. The AESOP robot system in laparoscopic surgery: Increased risk or advantage for surgeon and patient? *Surg. Endosc.* **2004**, *18*, 1216–1223. (In English) [CrossRef] [PubMed]
13. Vemprala, S.; Bonatti, R.; Bucker, A.; Kapoor, A. Chatgpt for robotics: Design principles and model abilities. *Microsoft Auton. Syst. Robot. Res.* **2023**, *2*, 20.
14. Sallam, M. ChatGPT utility in healthcare education, research, and practice: Systematic review on the promising perspectives and valid concerns. *Healthcare* **2023**, *11*, 887. [CrossRef] [PubMed]
15. Khan, R.A.; Jawaid, M.; Khan, A.R.; Sajjad, M. ChatGPT-Reshaping medical education and clinical management. *Pak. J. Med. Sci.* **2023**, *39*, 605. [CrossRef] [PubMed]
16. Gilson, A.; Safranek, C.; Huang, T.; Socrates, V.; Chi, L.; Taylor, R.A.; Chartash, D. How Well Does ChatGPT Do When Taking the Medical Licensing Exams? The Implications of Large Language Models for Medical Education and Knowledge Assessment. *JMIR Med. Educ.* **2023**, *9*, e45312. [CrossRef] [PubMed]
17. Lyu, Q.; Tan, J.; Zapadka, M.E.; Ponnatapura, J.; Niu, C.; Myers, K.J.; Whitlow, C.T. Translating radiology reports into plain language using ChatGPT and GPT-4 with prompt learning: Results, limitations, and potential. *Vis. Comput. Ind. Biomed. Art* **2023**, *6*, 9. [CrossRef] [PubMed]
18. Eslamian, S.; Reisner, L.A.; Pandya, A.K. Development and evaluation of an autonomous camera control algorithm on the da Vinci Surgical System. *Int. J. Med. Robot. Comput. Assist. Surg.* **2020**, *16*, e2036. [CrossRef] [PubMed]
19. Da Col, T.; Mariani, A.; Deguet, A.; Menciassi, A.; Kazanzides, P.; De Momi, E. Scan: System for camera autonomous navigation in robotic-assisted surgery. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 2996–3002.
20. Chen, Z.; Deguet, A.; Taylor, R.; DiMaio, S.; Fischer, G.; Kazanzides, P. An open-source hardware and software platform for telesurgical robotics research. In Proceedings of the MICCAI Workshop on Systems and Architecture for Computer Assisted Interventions, Nagoya, Japan, 22–26 September 2013; Volume 2226.
21. D'Ettorre, C.; Mariani, A.; Stilli, A.; Baena, F.R.; Valdastri, P.; Deguet, A.; Kazanzides, P.; Taylor, R.H.; Fischer, G.S.; DiMaio, S.P.; et al. Accelerating surgical robotics research: A review of 10 years with the da vinci research kit. *IEEE Robot. Autom. Mag.* **2021**, *28*, 56–78. [CrossRef]
22. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
23. Rosario, A.D. Available online: https://customgpt.ai/ (accessed on 15 May 2023).
24. Laurer, M.; Atteveldt, W.V.; Casas, A.; Welbers, K. Less Annotating, More Classifying–Addressing the Data Scarcity Issue of Supervised Machine Learning with Deep Transfer Learning and Bert-Nli. Preprint. 2022. Available online: https://osf.io/wqc86/ (accessed on 1 May 2023).