MI-Memo: Multi-modal Input for Interactive Memo

Qinglin Chen*
University of Bern
Bern, Switzerland
qinglin.chen

Chenrui Fan*
University of Bern
Bern, Switzerland
chenrui.fan

Francesco Lam*
University of Bern
Bern, Switzerland
francesco.lam

Denis Lalanne†
University of Fribourg
Fribourg, Switzerland
denis.lalanne

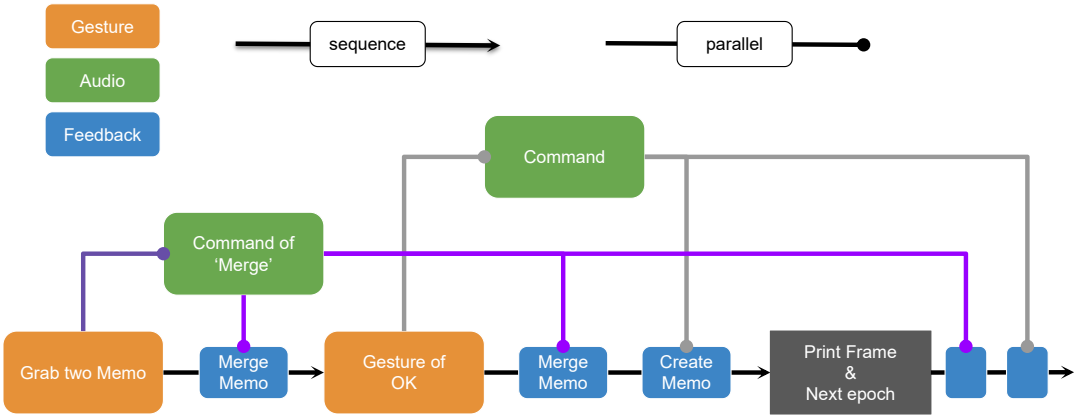

Figure 1: Parallel Strategy of MI-Memo

Abstract

It has been a long-standing phenomenon that eMemo can mostly only be interacted with via mouse and keyboard, which leads to limitations in its functionality[1]. We propose a new form of Memo interaction: a novel Memo interaction method based entirely on gestures and speech. In order to ensure the efficiency and accuracy of the system, we adopt a parallel functional architecture and use different approaches to handle situations of different complexity. Experiments and questionnaires demonstrate that our system is accurate and trustworthy, and provides a relatively more novel and convenient interaction pathway than methods that use exclusively the mouse and keyboard as input. Our implementation is available at https://github.com/Sosekie/Multi-modal-Input-for-Memo-Interaction.

1 Introduction

The electronic memo is a very common tool with many uses, such as recording to-do lists, organizing notes, or even writing a diary. However, the traditional way of interacting with memos is only mouse and keyboard, which leads to its limitation: it can't enhance user experience through multimodal interaction[2].

A significant problem is that editing a memo requires learning the mouse and keyboard counterparts. A user who wishes to merge two different memes needs to: First, open one memo 1 and copy its contents. Second, open another memo 2 and paste the content of memo 1. Finally, delete memo 1.

The above process is a common way to fulfill this requirement, it is not an intuitive representation. We would prefer to have a way to 'pick up' two memos, put them together, or say 'merge' and they will merge. This is very intuitive and easy for the user to learn.

To solve the above problems, we propose **MI-Memo**, a multimodal memo interaction method based entirely on gesture and speech recognition. The core of **MI-Memo** is to design more intuitive interactions for users. We use a combination of gestures and speech to fulfill most commands. These gestures are common and semantic, e.g. the ok gesture together with the 'create' voice means to create a new memo, which will appear at the fingertip of the 'ok' gesture and will move with the finger.

MI-Memo has many advantages. Firstly, **MI-Memo** is easy to learn because it uses gestures and voice commands that are also very commonly used in life, and users can instantly memorize these gestures and voice-corresponding commands when they see them, as opposed to mouse operations that require step-by-step reading of an instruction document. Second, **MI-Memo** provides very intuitive and rich interactive feedback. Users can freely choose where to generate the memo, where to drag the memo, and how to drag different memos together to make their contents blend, among other things. Third, **MI-Memo** provides stable operation feedback. Users can check the terminal output to know whether their commands are executed successfully or not.

Table 1: Operation correspondence table. Specific multimodal input states and memo states are combined to control the corresponding operations.

Operation	Gesture	Speech	Memo State		
			is_pinched	is_opened	is_catched
Create memo	Pinch	Create	FALSE	FALSE	FALSE
Open memo	Pinch	Open	TRUE	FALSE	FALSE
Add content	Pinch	Add	TRUE	TRUE	FALSE
Close memo	Pinch	Close	TRUE	TRUE	FALSE
Merge memos	Touch both	Merge	FALSE	T/F	TRUE
	Grab and drop.	None	FALSE	T/F	TRUE

Algorithm 1 MI-Memo parallel function

Input: Gesture, audio, and memo state values
Output: Boolean value for whether to execute

for each round R_i in loop **do**
  $GestureType \leftarrow M_{gesture}(Gesture)$
  $MemoStatus \leftarrow Memo.status$
  if $MemoStatus$ and $GestureType$ **then**
    Start a new thread
    $AudioTrigger \leftarrow M_{audio}(Audio)$
    if $AudioTrigger$ **then**
      Execute the corresponding operation
    end if
  end if
  $Frame \leftarrow NewFrame$
  Render $Frame$
end for

2 MI-Memo

This section describes the framework of our MI-Memo. We will first introduce the specific modal inputs we use, then explain how we handle each modal input, and finally the overall architecture.

MI-Memo uses the video content from the front camera of the user's computer and the input from the microphone as the raw data, which the pre-trained model processes to get the gesture information[3] and voice content information[4]. Since most computers come with cameras and microphones, MI-Memo can ensure that the modal input is continuous and uninterrupted. Based on this, MI-Memo combines the two modalities to derive a series of operations.

2.1 Multimodalities

The modalities used in MI-Memo include **gesture** and **speech**. Among them, voice is divided into 'trigger' voice, which is used in combination with gestures to control operations, and 'content' voice, which is used for detailed text input. The difference is that the former is processed by converting it to MFCC[5] and then calculating the cosine similarity[6] with standard speech, resulting in a Boolean value (do or do not do) that takes less than 0.03 seconds. The latter is a direct end-to-end processing of the speech file into text using a pre-trained model, which takes longer and is not conducive to interacting with gestures as the system needs to wait for a long time.

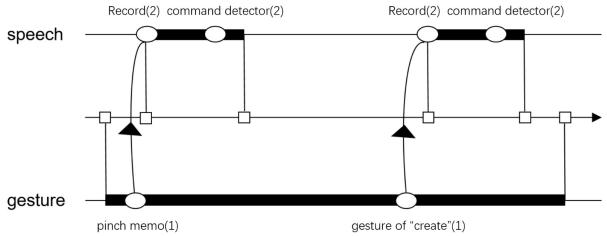

Figure 2: In this system, different modalities operate in parallel.

2.2 MI-Memo overview

The structure of MI-Memo is shown in Figure 1. We use a parallel structure to handle different functional operations, each of which corresponds to a different modal input. We use the state of the memo and the combination of the two modal inputs to precisely control the corresponding commands. The specific operation correspondences are shown in Table 1.

Speech recognition takes time, so we designed all the functions related to voice as parallel places[7]. We require that 1. each function cannot be run more than once at the same time, and 2. the system runs at most 3 functions in parallel at the same time. This ensures that the system performance can support parallel operations and that the user does not have to wait for a long time for the current operation to complete. Specifically, whenever the gesture and memo state satisfy the corresponding operation, a thread is created and audio input detection begins. When the thread finishes, it detects the result and decides whether to execute the corresponding operation. During this time, the user can continue to complete other operations. The result of the audio thread execution will come at any future moment and will be shown in the final rendering section of each round. The implementation is shown in Algorithm 1.

2.3 CASE/CARE

The concepts of CARE and CASE, introduced by Nigay and Coutaz, define two distinct design spaces for multimodal human-machine interactions. The aim is to for-

Table 2: Survey

	Q1	Q2	Q3	Q4	Q5	Q6
User 1	4	3	5	7	Using only gesture	5
User 2	7	5	3	7	Using both gesture and speech	5
User 3	5	4	5	7	Using both gesture and speech	4
User 4	6	5	5	7	Using both gesture and speech	6
User 5	4	4	6	5	Using both gesture and speech	2
User 6	4	7	3	6	Using both gesture and speech	3

Table 3: UMUX-Lite Satisfaction Score

UMUX-Lite	**User 1**	**User 2**	**User 3**	**User 4**	**User 5**	**User 6**
Satisfaction Score	41	83.33	58.33	75	50	75

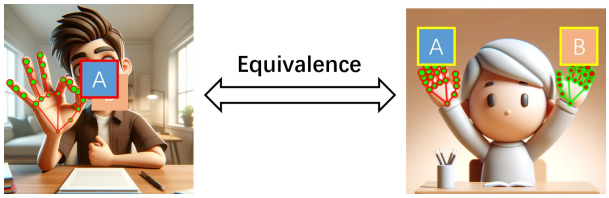

Figure 3: Equivalence of merge function

malize these interactions and conceptualize the potential relationships between input and output modalities.

For CASE, as shown in Figure 2, the system performs the functions in a **synergistic** way to enhance the overall system's functionality and performance. The combined effect of these components is greater than the sum of their individual effects.

For CARE, the systems perform most of the function in a **complementarity** way to enhance both the human's capabilities and the machine's capabilities. For example, when a user issues the "Create" command using a voice command, the system recognizes the speech accurately (voice recognition). Simultaneously, if the user also performs a specific hand gesture associated with the "Create" action, the system can combine these inputs to confirm the user's intention.

Special, for the merge function embodies the **equivalence** since we can do this not only by gesture plus voice but also by dragging and dropping by gesture only, as shown in the merge operation in Table 1 and Figure 3.

3 User Evaluation

For the user evaluation, we asked 6 different people to test our system The evaluation consists of two parts:

1. Task Completion Time (Quantitative method): Participants completed a set of tasks using our multimodal memo system. The time taken to complete each task was recorded.

2. Survey (Qualitative method): Participants filled out a survey after using the system. The survey assessed user experience, ease of use, perceived efficiency, and overall satisfaction.

The tasks that users have to do were:

- Create a new memo (**Create**)
- Merge the memos using multimodality (**Merge1**)
- Open the memo (**Open**)
- Add the sentence "Hello, how are you" on the opened memo (**Add**)
- Close the memo (**Close**)
- Merge the memos only using gestures (grab and drop) (**Merge2**)

The questions of the Survey were divided into 2 sections. The first section was the UMUX-Lite questionnaire and the second section was about the 2 modalities used in our project: speech and gesture

- Q1: This system's capabilities meet my requirements
- Q2: This system is easy to use.
- Q3: The system had some problems recognizing my voice.
- Q4: The system handles well the recognition of my hands
- Q5: Do you prefer the "merge" function by using both vocal commands (speech) and touching the 2 intended memos (gesture) or by "catching" the 2 memos and placing them together (only gesture)?
- Q6: I would use this Virtual Memo System instead of typing the letters on the keyboard by hand.

The formula for computing the standard UMUX LITE[8], where x1 and x2 are the ratings for Q1 and Q2 using a standard 7-point scale, is UMUXLITE $= (x1 + x2 - 2)(100/12)$. Also for the section about the multimodalities, it has been used a 7 point scale, where 1 represents a strong disagreement and 7 represents a strong agreement.

4 Results analysis

Among the 6 participants, 3 have a UMUX-Lite Satisfaction Score of 75 or above, 2 have a score between 60

Table 4: Task Completion Time (seconds)

	Create	Merge1	Open	Add	Close	Merge2
User 1	2.11	2.31	2.17	9.87	1.25	1.39
User 2	2.36	0.61	5.50	8.14	0.41	7.14
User 3	1.67	0.61	4.98	9.14	1.13	1.78
User 4	1.53	1.79	4.98	7.98	4.13	6.49
User 5	1.79	0.87	1.06	9.22	0.53	10.99
User 6	1.79	1.26	0.93	8.24	0.61	7.86

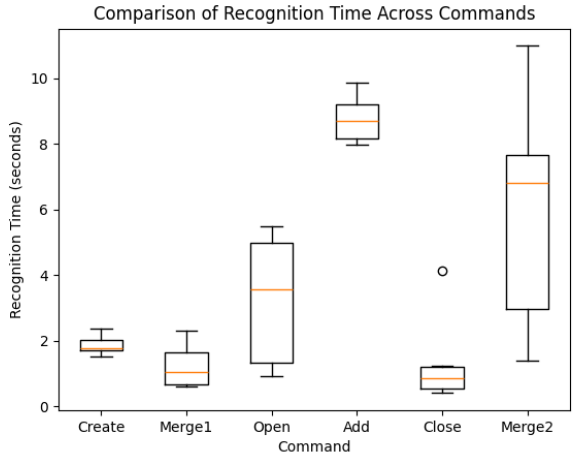

Figure 4: Box Plot of the Task Completion Time.

and 50 and 1 participant has a score below 50. Variation in Satisfaction: There is a significant variation in satisfaction scores among users, with User 2 being the most satisfied (83.33) and User 1 being the least satisfied (41). Most users have a score above 50, indicating a generally positive experience, with the exception of User 1 who is less satisfied. And about the multimodalities, it is possible to retrieve the following information:

Q3 (Voice recognition problems): Ratings here are generally mid-range (3 to 6), suggesting that some users experienced issues with voice recognition, while others did not.

Q4 (Hand recognition): Most users rated this highly (5 to 7), indicating that hand recognition is generally well-handled by the system.

Q5: Using only gestures: Only User 1 prefers this method. Using both gesture and speech: The majority of users (Users 2, 3, 4, 5, 6) prefer this combined method, indicating a preference for multimodal interaction.

Q6: The responses range from 2 to 6, showing varying degrees of willingness to adopt the system over traditional typing. Most users rated this around the mid-point, indicating some hesitation or conditions under which they would prefer the system.

For the Task Completion Time, it has been considered the starting point of measurement when the user begins to say the command.

Create: The times are fairly consistent, with User 2 taking the longest (2.36 seconds) and User 4 being the fastest (1.53 seconds).

Merge 1: User 1 took the longest time (2.31 seconds) while Users 2 and 3 were significantly faster (0.61 seconds).

Open: User 2 took the longest (5.50 seconds) and User 6 was the fastest (0.93 seconds).

Add: This command generally took the longest for most users, with times ranging from 7.98 to 9.87 seconds.

Close: User 4 took the longest time (4.13 seconds) while User 2 was the fastest (0.41 seconds).

Merge 2: There is significant variability, with User 5 taking the longest (10.99 seconds) and User 2 the shortest (0.61 seconds). This is because the starting points of the memos were always different.

There is some indication of a moderate positive relationship between task completion times and the perception that the system's capabilities meet user requirements (Q1). However, correlations with other survey questions are weak, suggesting that task completion times are not strongly related to most user perceptions captured in the survey. This analysis highlights that while task performance metrics are important, user satisfaction and perceptions involve additional factors that may not directly correlate with completion times.

5 Conclusion

In this work, we propose MI-Memo, a new type of memo that realizes multimodal fusion and interaction based entirely on gestures and speech. We adopt a parallel architecture to handle multiple functions and use multiple modalities to control specific operational content, while fully considering the processing time problem and using different precision for different audio processing situations. The user evaluation demonstrates that a multimodal way of interacting with memos is faster and more interesting than a function (in our project the command Merge) with just one modality.

Limitation: Our work also suffers from the problem of long program execution time, which makes it difficult to achieve timely feedback. It is also a pity that the words cannot be printed out sequentially when the content of the model is input by voice.

Acknowledgements

The success of this project would not have been possible without the help of the professor and two teaching assistants. We would also like to thank our friends who were willing to patiently walk us through the test, including but not limited to Francesco's friends and family, Chenrui's sister, and Qinglin's friends. Thank you all.

References

[1] Jane Mayer. The memo. *The New Yorker*, 27:41, 2006.

[2] Kapil Chandra, Frank Plaschke, and Ishaan Seth. Memo to the cfo: Get in front of digital finance—or get left back. *McKinsey & Company*, 2018.

[3] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[4] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[5] Vibha Tiwari. Mfcc and its applications in speaker recognition. *International journal on emerging technologies*, 1(1):19–22, 2010.

[6] Faisal Rahutomo, Teruaki Kitasuka, Masayoshi Aritsugi, et al. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1. University of Seoul South Korea, 2012.

[7] Zina A Aziz, Diler Naseradeen Abdulqader, Amira Bibo Sallow, and Herman Khalid Omer. Python parallel processing and multiprocessing: A rivew. *Academic Journal of Nawroz University*, 10(3):345–354, 2021.

[8] James R. Lewis. Measuring perceived usability: The csuq, sus, and umux. *International Journal of Human-Computer Interaction*, 2018.