# Introducing a Critics framework to mitigate foundation models' hallucinations in robotic application

Damien Alouges[1], Raphaël Lallement[1], Matteo Morelli[1], Sébastien Gérard[2]

***Abstract*— While Foundation Models (FM) hold potential for enhancing the autonomy and common sense of robotic AI systems, their tendency to hallucinate makes them unreliable in real-world applications. We seek to eliminate such hallucinations to enable controlled and trustworthy outputs for robotic agents. In the pursuit of this objective we initiated the definition and the implementation of a Critics framework inspired by a positional paper, the framework uses expert-rules-based critics to detect hallucinations and request a new generation with the hallucination pointed out to the foundation model. We propose a formalization of the framework, defining the concepts used for now and ideas for future upgrades. We did a first implementation on a basic scene-description use case to identify the main challenges of this approach. The results are promising as we observed a positive impact on the number of hallucinations the framework produced versus what the foundation model alone generated.**

## I. Introduction

Foundation Models have had a great impact on informatics in the past years. Robotics also went through a shift. Although those Artificial Intelligence (AI) allow to address complex problems and offer a 'common sense', their output are uncertain and hard to control. However, when robots are in interactions with humans we require guarantees on the possible outputs to ensure safety to the user and the robot. We will present our first formal definition and implementation on a basic scene-description use case of our framework inspired by the positional paper [1]. There are different approaches that aim to detect and mitigate the hallucination problem such as Retrieval Augmented Generation (RAG) [2] [3], uncertainty estimation [4] or the use of another FM or neural network [5] [6]. But those approaches are not entirely reliable and can be sources of more hallucinations depending on how they are used [7]. Our Contribution is to define and implement a framework to detect hallucinations and if possible correct them. The framework is not limited to the robotic domain but it is where we see its best potential, because it's a domain that need the insurance that all safety constraint are respected, which is where the use of foundation models can cause problems.

## II. Our approach

### A. Positional paper overview

The positional paper [1] introduces an approach to enhancing the reliability of FM outputs through the integration of an iterative validation loop involving critics/verifiers. In this framework, each generated output is evaluated by a set of critics before being finalized. If the generation fails to meet the validation criteria, the FM iteratively produces new candidates until all critics approve the proposed output. The authors distinguish between two categories of critics: hard critics, which assess the factual correctness and logical consistency of the generated content using symbolic methods, and soft critics, which evaluate more abstract qualities such as style, explainability and preferences conformance. In the paper the authors focus on planning use-cases, like in the strict application of the modulo framework in [8]. Our approach aspires to be more generalist, we aim to define a framework usable in every robotics use cases which involves a FM usage.

### B. Hallucinations definition

In the survey [7] hallucinations are defined as "generated content that appears nonsensical or unfaithful to the provided source content", in our approach we consider that any error generated by the FM is an hallucination, implying that a model output may contain multiple hallucinations. The authors identified multiple types of hallucinations, and named two major types : "factual hallucination" and "faithfulness hallucination" with multiple sub-types, which all have different possible causes and detection methods. At first glance a link could be done between the concept of hard/soft critics and factual/faithfulness hallucination, where hard critics are for factual hallucination and soft for faithfulness hallucination, but an expert verifier (hard critic) can also be used for detecting logical inconsistency, that is categorised as a faithfulness hallucination. So we define that hard critics would be critics that give a boolean review, either a negative review (in case of hallucination) or a validation review. And soft critics could give back a third type of review, the positive review, that is in case of model response that is valid but imperfect, with unwanted language or style.

### C. The framework description

The proposed Critics Framework (Figure 1) works as follows: first, the FM takes as entry a prompt, the user request, and other inputs that would be required by the use case it is applied on (eg. an image). The Model will generate its response called the proposed output. This output is fed in the critics block that will analyse it, the critics block also takes the prompt, the input and potential external resources (eg. an ontology or a documentation). All those data are used in the first part of the critics block : the critics,

*This work was funded by the European Union's Hori-zon Europe ResearchInnovation Program under Grant101070227 (CONVINCE).

[1]Damien Alouges, Raphaël Lallement and Matteo Morelli are with Université Paris-Saclay, CEA-List, Palaiseau, France

[2]Sébastien Gérard with the IRT Jules Verne, Nantes, France

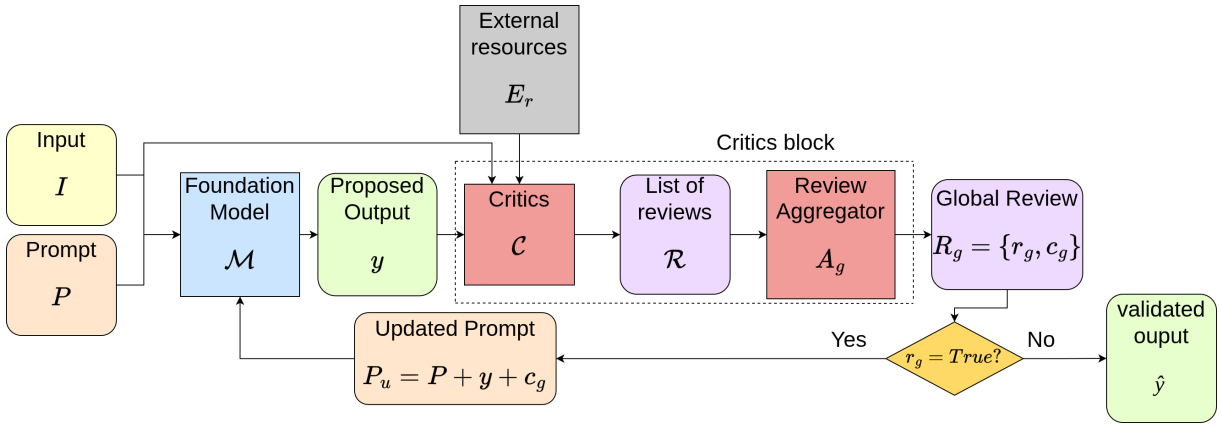


Fig. 1. Critics Framework

which are defined as tools to detect hallucinations. Each critic will return a review, containing two things : a rating corresponding to if the critic detected an hallucination or not (True or False) and the comment that is an explanation of the hallucination. Now we got list of reviews (one per critic) that we aggregate in a global review with the review aggregator. Based on the content of the global review we take a decision, if we do not have detected an hallucination, we consider the proposed out as valid and it goes out of the framework. If we have detected an hallucination, then we upgrade the prompt by adding to it the proposed output and the explanation of the detected hallucination and call once again the model with the upgraded prompt and the input, and we expect the model to correct the hallucination. The framework is supposed to loop until all hallucinations are corrected or if we reached the max number of loop defined by the user.

### D. Framework formalization

Our framework requires the following components : A foundation model $\mathcal{M}$, a prompt $P$, an input $I$. With those two inputs the model will do a prediction : the proposed output $y = \mathbb{P}_{\mathcal{M}}(P, x)$, composed of K tokens $y = \{T_0, ..., T_K\}$. It can be declined to multiple variants : complete $y_c = \{T_0, ..., T_K\}$, partial $y_p = \{T_i, ..., T_j\}$ with $0 <= i < j <= K$, last token $y_l = \{T_K\}$, and if the model is used in stream mode $y_c = \{T_0, ..., T_k\}$ with $0 < k < K$. This proposed output is sent to be analysed by the list of $l$ critics, $\mathcal{C} = \{C_0, C_1, ..., C_l\}$ that will return a list of reviews $\mathcal{R} = \{R_0, ...., R_l\}$. A Critic $C_x$ is a function that takes as input a Prompt $P$, the input $I$, the external resources $E_r$ and the proposed output $y$ and return a review $R$, resulting in : $C_x(P, I, E_r, y) = R_x$ with $R_x = \{r_x, c_x\}$. $r$ is the review rating, it can be a boolean $r = \{True, False\}$, with $r = True$ meaning the critic detected a targeted hallucination. $c$ is the comment of the review, it is a text that explain the hallucination and if possible how to correct it. The list of reviews $\mathcal{R}$ is given to the review aggregator $\mathcal{A}_g$ to aggregate it in a general review $R_g = \{r_g, c_g\}$. If the general rating $r_g$ is False, we turn the proposed output $y$ into the validated output $\hat{y}$. Otherwise the general review is then used, along with the proposed output, to update the prompt into the updated prompt $P_u = P + y + c_g$. This updated prompt is then used to query the model to produce a new output and the process is repeated as described in Algorithm 1. There is two conditions to end the loop of the framework, the first one is that no hallucinations are detected ($r_g$ = False). The second one is that the maximum number of iteration $n$ set by the user was attained.

## III. Experiment

### A. Context

To experiment the framework we applied it on different uses case like scene description, planning and situation explanation. Despite good results on low numbers of examples in planning and situation explanation, we decided to focus on the scene description use case because it was easier use case to create a large database to test the framework. This experiment test the Critics Framework implementation within a specific use case involving a Vision-Language Model (VLM) tasked with doing scene description. The VLM is asked to describe a scene composed of five cubes randomly positioned on a table. The generated description is expected to be in JSON format and should first enumerate each cube and then detail the spatial relationships between the cubes and the table. Three types of relations are used : ”on”, indicating a cube placed atop another; ”clear”, describing a cube with no objects on it; and ”ontable”, identifying a cube directly resting on the table.

For this experiment, we used InternVL2-llama3-73B-AWQ model, configured with a temperature of 0 and a top-p value of 1. These parameters ensure deterministic outputs, minimizing randomness in the generated descriptions. The VLM is treated as a black box, receiving a textual prompt and one or two images of the scene as input.

### B. Database

For the database, we use 360 simulated scenes of cubes that are are stacked in different manners on a table pictured in Figure 2. The scenes are randomly created in a simulator and we have 3 different point-of-views, 120 scenes are

**Algorithm 1** Critics framework pseudo-code

**Require:** Prompt $P$, input $I$, Model $\mathcal{M}$, Critics $\mathcal{C} = \{C_0, C_1, ..., C_k\}$, Review aggregator $\mathcal{A}_g$, max_iteration $n$
**Return:** validated output $\hat{y}$

```
i = 0, r_g = True, P_u = P                                   ▷ Initialization
while i < n or r_g ≠ False do
    i = i + 1                                                ▷ Counting number of iteration
    y_i = P_M(P_u, x)                                        ▷ Generate proposed output y_i
    for j in [0, k] do                                       ▷ Send proposed output y_i in critics to generate the list of reviews R
        r, c = C_j(P_u, x, E_r, y_i)
        R = R + (r, c)
    end for
    r_g, c_g = A_g(R)                                        ▷ Review aggregator generate the global review R_g = {r_g, c_g} from the list of reviews
    P_u = P + y_i + c_g                                      ▷ Upgrade the initial prompt with the global review comment and the proposed output
end while
return ŷ = y_i
```

viewed with a 'basic' robot view (monocular, from a robot representative height), 120 with a top view and a 'basic' robot view and the 120 last ones are viewed through a binocular point of view.

### C. Experiment process

To evaluate the effectiveness of our framework we first tested the model alone to assess the presence and frequency of hallucinations in the generated scene descriptions. Hallucinations are identified by comparing the model's output against the ground truth of the scene. Secondly, the Critics Framework is integrated into the pipeline. The framework's impact on the model's performance is then measured by re-evaluating the outputs for hallucinations on the same input images. This comparative analysis aims to quantify the reduction or transformation of hallucinatory content, providing insights into the critic framework's effectiveness in improving output reliability.

### D. The implemented critics

We developed 2 types of critics, the *realistic critics* and the *oracle critics*. The *realistic critics* are algorithms that detect most of the time the hallucinations it targets, but not constantly due to the difficulty to detect certain types of hallucinations. List of *realistic critics* implemented:

- wrong_color critic: triggered if color name is detected and will search in the pixel of the image if a block of this color is indeed in the image.
- ontable_on critic: verify if a block is described as on the table with the "on" relation instead of "ontable"
- number_cubes critic: verify that the description contain 5 blocks, because the environment was defined with 5 blocks present.
- missing_ontable critic: verify if all cubes that are not described as on another cube is described as on the table
- false_on critic: for all "on" relations will check if the cubes described as on one another are in contact in the image

The *oracle critics* are allowed to cheat and look in the ground truth to detect hallucinations. That way the hallucinations are always detected by the *oracle critics* and we can evaluate the effectiveness of the framework in the theoretical best case scenario of perfect hallucination detection.

## IV. Results

The results of our tests are displayed in table I, which shows the occurrences of every hallucination type and the difference in percent of the VLM acting alone and with critics. Without any addition, the VLM produced only approximately **47% of responses that do not contain hallucinations**.

With the **realistic critics** we have obtained **53% of non-hallucinatory responses**, an increase of 7% more than the VLM alone. We can see that for the targeted hallucinations, the critics helped reducing the hallucinations. Some hallucinations are even down to 0 occurrences, (false_cube and ontable_on). But the inversed_on has increased. In our opinion it is due to the fact that it was not criticized and so while attempting to correct another hallucination the VLM end up generating this one.

With **oracle critics** that target all hallucinations, we have obtained **71%** of final generations **without hallucinations**. The 29% remaining ones were detected as hallucinatory but the model did not achieve to correct it. It is a phenomenon we called recidiv behaviour, it is when a critic detects a hallucination and points it out to the VLM, but the model still generates the same hallucination again and again.

We suspect that the use case involving image processing is harder than just text type use case, so we expect better results on use cases like planning.

In an attempt to mitigate the recidiv behaviour we tried to implement the framework with incremental temperature. When a recidiv is detected, the temperature is increased by 0.1. We thought that by doing that the model will have enough flexibility to listen to the critics and still enough rigidity to generate responses in the expected format. We tried only on one third of the database (single_image_view) to see if it was a relevant approach and if it deserves to dig deeper. The increment to temperature as we implemented did not show better results, the recidiv behavior, while weakened, was still happening and most of the time, the model created new hallucinations while correcting others, making the loop infinite if not ended by reaching the max_iteration number.

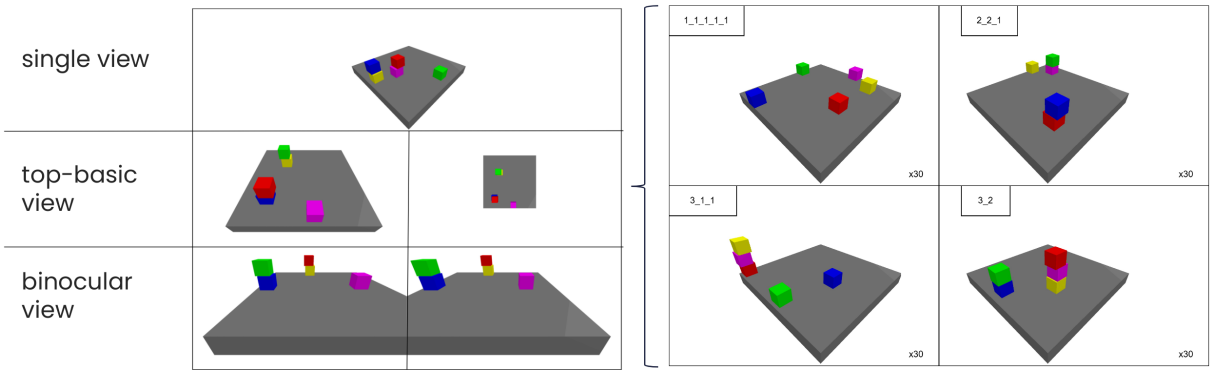


Fig. 2. Samples from our generated dataset, (left) demonstrates the three views we consider, (right) shows some random cube placement in the four defined configurations

| | Without critics | Realistic critics | | Oracle critics | |
|---|---|---|---|---|---|
| hallucination type | occurences | occurences | difference | occurences | difference |
| missing_ontable | 225 | 204 | -9% | 145 | -35% |
| false_on | 186 | 153 | -18% | 133 | -28% |
| missing_clear | 168 | 152 | -9% | 89 | -47% |
| false_clear | 61 | 55 | -10% | 3 | -95% |
| inversed_on | 52 | 75 | +25% | 12 | -77% |
| false_ontable | 45 | 38 | -15% | 1 | -98% |
| missing_on | 27 | 10 | -63% | 0 | -100% |
| missing_cube | 21 | 3 | -86% | 0 | -100% |
| false_cube | 19 | 0 | -100% | 0 | -100% |
| ontable_on | 9 | 0 | -100% | 2 | -78% |
| wrong_color | 7 | 2 | -71% | 0 | -100% |

TABLE I

COUNT OF THE HALLUCINATIONS BY TYPE OF THE VLM TESTED ALONE, WITH THE REAL CRITICS AND WITH THE ORACLE CRITICS ON 360 SCENES. AND ALSO THE PERCENTAGES OF DIFFERENCE BETWEEN VLM WITH AND WITHOUT CRITICS, IF THE PERCENTAGE IS NEGATIVE IT MEANS A DECREASE OF HALLUCINATIONS, IF POSITIVE IT MEANS THERE IS AN AUGMENTATION.

## V. CONCLUSION

In this work we proposed a formal definition of the Critics Framework, and an application on a scene description use case. With our first implementation of the Critics Framework we observed a decrease of hallucinations in the VLM's responses. We identified some key subjects to explore to have better result with the framework.

In the future we seek firstly to a deeper exploration of reviews' definition and how to make it more impactful to reduce the appearance of the recidiv behaviour, doing so would greatly upgrade the framework efficiency. Secondly, exploring new type of critics, like critics that would evaluate the uncertainty of the model. Finally, upgrading the formal definition of the framework introduced here, exploring the definition and integration in the framework of different types of critics and reviews. We want to test adding an information to reviews and critics, the Review precision $R_p$ that would indicate the capacity of a critic to explain an hallucination it detected, and then the Critic confidence $c_c = r + R_p$ that would take in account the review rating $r$ and the review precision $R_p$ to mimic the concept of confidence level of reviewer in scientific paper publication. It would help hierarchies the critics reviews from most important to less valuable.

## APPENDIX

| Notation | Description |
|---|---|
| $P$ | Prompt |
| $P_u$ | Updated prompt |
| $I$ | Input |
| $E_r$ | External resources |
| $y$ | Proposed output |
| $\hat{y}$ | Validated output |
| $\mathcal{M}$ | Foundation model |
| $\mathcal{C}$ | List of critics |
| $C$ | Critic |
| $\mathcal{R}$ | List of reviews |
| $R$ | Review |
| $r$ | Review rating |
| $c$ | Review comment |
| $\mathcal{A}_g$ | Review aggregator |
| $R_g$ | Global review |
| $r_g$ | Global review rating |
| $c_g$ | Global review comment |

TABLE II

SUMMARY OF NOTATIONS

## ACKNOWLEDGMENT

This work was funded by the European Union's Horizon Europe Research Innovation Program under Grant 101070227 (CONVINCE).

## References


[1] S. Kambhampati et al. "LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks," pre-published.

[2] O. Ram et al., *In-context retrieval-augmented language models*, 2023. arXiv: 2302.00083 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2302.00083

[3] S. Barnett, S. Kurniawan, S. Thudumu, Z. Brannelly, and M. Abdelrazek, *Seven failure points when engineering a retrieval augmented generation system*, 2024. arXiv: 2401.05856 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2401.05856

[4] N. Varshney, W. Yao, H. Zhang, J. Chen, and D. Yu, *A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation*, 2023. arXiv: 2307.03987 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2307.03987

[5] M. Gao, J. Ruan, R. Sun, X. Yin, S. Yang, and X. Wan, *Human-like summarization evaluation with chatgpt*, 2023. arXiv: 2304.02554 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2304.02554

[6] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 1906–1919. DOI: 10.18653/v1/2020.acl-main.173 [Online]. Available: https://aclanthology.org/2020.acl-main.173/

[7] L. Huang et al., "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *ACM Trans. Inf. Syst.*, vol. 43, no. 2, Jan. 2025, ISSN: 1046-8188. DOI: 10.1145/3703155 [Online]. Available: https://doi.org/10.1145/3703155

[8] A. Gundawar, M. Verma, L. Guan, K. Valmeekam, S. Bhambri, and S. Kambhampati, "Robust planning with llm-modulo framework: Case study in travel planning," *arXiv preprint arXiv:2405.20625*, 2024.