**Reviewer 1 Comments**

R1.1: "I would encourage you to elaborate more on the most important concepts in the introduction - specifically, object detection vs instance segmentation, transfer learning vs foundation models, and vision language models. To that end it would make a lot of sense to move and dissolve the first part of the methods section into the introduction (which is currently very short), and then start the methods directly with a description of SegmentR."

Author response: Thank you for the suggestion. I have moved the description of object detection and segmentation into the introduction. I have also added some additional details about foundation models and vision language models in the introduction and discussion.

R1.2: "I would not just mention citizen science projects but any sorts of biological settings - e.g., museomics or laboratory experiments. I think you want your audience to be as broad as possible"

Author response: I have changed the writing to make it clear that this can be used for all sorts of biological images.

R1.3: "probably mention 'foundation models' or 'foundational AI', and highlight that these models don't require context-specific training or fine tuning"

Author response: It is now clear that these models do not need additional training or tuning. However, I did not discuss foundation models in the Introduction because I could not find a natural way to incorporate it with the writing. Instead, I have briefly touched on foundation models in the discussion, particularly in the context of base models for fine-tuning.

R1.4: "you should mention here that all segmentation your package does through grounded SAM is based on text input by users"

Author response: This is now clear.

R1.5: "I think you should move this part to the introduction - it does not really make sense to start the methods with a treatise on computer vision"

Author response: Done.

R1.6: "since users need to use conda, this statement is not really helpful, as python will be part of the environment that the user has to create. This section is pretty confusing - probably just state that users need to install miniconda or anaconda, and then continue in R"

Author response: Good point, I now just state that SegmentR is python based and that users should install conda.

R1.7: "I am not really sure what to make of this part. I do think it is a good idea to let folks in a bit on how the package works, but there is so much jargon throughout this paragraph that is tangential to the actual use that it might confuse users more than it helps them. How about a little schematic figure that ties all of this together?"

Author response: Good point, I agree that a flowchart type diagram would be better suited for describing the workflow and that has now been added.

R1.8: "this table should give an overview of the most important functions rather than describing all arguments in detail. functions can (and probably will) change as you maintain the package, and users will be able to refer to the R help files, which is the best place for this information. I would either condense this down to a much shorter version, or integrate with a schematic figure I proposed above"

Author response: In line with this comment and R2.3, I have simplified the table greatly.

R1.9: "looking at this batch example I wondered whether it is possible to run the segmentation and keep the results in the R-environment to pass it on directly to packages like patternize and recolorize? or is it necessary to save and reload the results? this creates a lot of extra IO operations that will slow down batch processing of large datasets"

Author response: This is a good thought and my original intent with the package was to do something along these lines. However, I could not find a good way to do this in practice. Keeping everything in the R environment by using something like reticulate slowed down image processing massively (reticulate was unable to distribute tasks to multiple cores). So I chose to use command line inference where the built-in functionality of the python libraries could be better utilized. However, this means that the python functions and the R functions don’t share data unless it is saved to the HDD. I considered making the python files temporary, loaded into R automatically for a user to do with as they wish. But I was concerned that if I had too many segmentations loaded into R, the memory demands would be too great and ultimately it would be better just to keep everything saved on the HDD to be loaded as necessary.

R1.10: "this hints at some of the points the other reviewer and myself have made: you present only very few examples, and it would be appropriate and informative to apply the pipeline to a few more contexts (i.e., standardized and natural settings) and organisms (i.e., poorly vs. well represented in training datasets). this would allow you to build a very interesting discussion, rather than speculating about cases you did not actually explore"

Author response: Good point. I have expanded the use cases to include some other types of examples: cell assays, standardized museum specimens, etc. These are now included in the supplemental as examples.

**Reviewer 2 Comments**

R2.1: "I suggest instead adding a short section to the package description on writing effective prompts/labels with basic recommendations, e.g. that 'a bee.' is a significantly different prompt from 'bee'."

Author response: Excellent point. This is also in line with comment R1.4 which wanted me to more explicitly highlight the text-based input.

R2.2: "This is where I'd add a short section on labeling/prompting. [Lines 107-126]"

Author response: Added.

R2.3: "I don't think there's much point to having a table of package functions that goes through every argument for every function, since that's redundant with the actual package documentation (which is likely to be updated anyways). Realistically, people who use the package will access that documentation in the manual/help pages in R, not from the manuscript, and people reading the manuscript will skim this. Instead I suggest organizing the table with four attributes: function name, input(s), output(s), and perhaps any comments specific to that function."

Author response: Agreed, Reviewer 1 made a similar point (R1.8). I have followed your suggestion. This with the addition of the flow chart should help users better understand how the package functions.

R2.4: "Suggest including the example code as actual text in the manuscript rather than as a panel of the figure. That way people can copy-paste it more easily and you won't encounter the inevitable weird resizing issues when the MS gets typeset."

Author response: I have done this for formatting reasons and would like to keep it this way. However, I have now noted that the exact example text can be found in the XX…

R2.5: "The export\_transparent\_png() function isn't exported from the package namespace, so I had to use the triple-colon notation (SegmentR:::export\_transparent\_png) to use the function. There's also at least one imager function in that export\_transparent\_png that isn't accessed using the double-colon notation (as.pixset on line 139 of export.R)."

Author response: Thank you for pointing this out. It is now corrected.

R2.6: "Even if the author doesn't plan to submit the package to CRAN, I really suggest working through some of the earlier local package/build checks (e.g.<https://r-pkgs.org/workflow101.html#sec-workflow101-r-cmd-check>). It catches a lot of these things automatically, which is frustrating at first, but saves you from a lot of confused emails from users later."

Author response: Thank you for the suggestion. I have gone through the CRAN checks.