Image processing on esp32-cam vs python

While working on the project, one key consideration is whether to perform image processing directly on the ESP32-CAM board or remotely using Python on a more capable computer. These are the conclusions we came to while investigating the topic.

1. Processing Power:
   The ESP32-CAM, while powerful, has limitations in processing power due to its resource constraints. Utilizing a dedicated computer running Python provides access to high-performance processors and GPUs, enabling faster and more complex image processing tasks. With Python's extensive libraries and tools, computationally intensive algorithms and advanced techniques become feasible, leading to more accurate and sophisticated results.
2. Handling Larger Images:
   With remote Python processing, you can overcome the memory limitations of the ESP32-CAM and process larger images. This is particularly advantageous in applications that require high-resolution image analysis or deal with images captured by high-resolution cameras. The larger memory capacity of the computer allows for storing and manipulating these images without compromising performance or accuracy.
3. Memory-Intensive Operations:
   Certain image processing algorithms, such as deep learning models or complex feature extraction techniques, may require significant memory resources. Remote Python processing provides the advantage of access to high-performance processors and abundant memory, enabling the execution of memory-intensive operations efficiently. This allows for the implementation of advanced algorithms that require extensive memory for parameter storage or intermediate computations.
4. Freedom in Algorithm Selection:
   Python offers a wide variety of libraries and tools for image processing, providing freedom in selecting the best algorithms for particular jobs. We are given access to a wide range of functions, filters, and machine learning capabilities thanks to libraries like OpenCV, PIL, and scikit-image. The ESP32-CAM's restricted image processing libraries, on the other hand, might limit the alternatives available and make it difficult to create customized algorithms.
5. Real-Time Decision-Making:
   The on-board image processing on the ESP32-CAM clearly offers a benefit for time-sensitive applications, that must react instantly to its surroundings. It can provide an improved real-time analysis and decision-making, which enables prompt identification of objects. This capacity to make decisions right away cuts down on potential delays from sending images and waiting for remote processing.

6. Reduced Network Bandwidth:
   The quantity of data that must be transmitted over the network is reduced by processing images on the ESP32-CAM. This is especially helpful in situations where network is slow. With limited resources, communication can be carried out more effectively by sending only the outcomes of processing or usable data rather than raw image data.

Conclusion:

While it might help with internet issues that we might stumble upon, we would rather stick to the python image processing for the time being and focus on developing it, while keeping the ESP32-CAM image processing option at the side.
As we found out that there are advantages and disadvantages for each one, we feel more comfortable with the python image processing, as we studied it and know about it much more, and in general there are much more libraries and tools for that. We did read about some libraries for the ESP32-CAM using Arduino like Arduino Image Processing Library (ArduImg) and such, and we will keep a close eye for that option, while focusing on python.