CSE 237D Project Specification

Baboons on the Move

Debaditya Basu, Sananya Majumder, Zhi Wang, Baiqiang Zhao

April 21, 2022

1 Project Charter

1.1 Project Overview

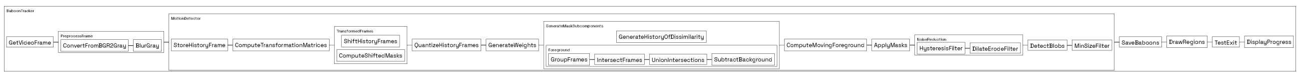
Baboons are social animals that often engage in group activities. Scientists have been interested in studying their social life and group decision-making [Washburn and DeVore, 1961, Stueckle and Zinner, 2008]. In order to help researchers track their movement and activities, the project Baboons on the Move introduces a system that tracks baboons in Kenya from drone footage [Crutchfield et al., 2020]—a drone with a camera stays in the air on top of a troop of baboons, stably and continuously takes video records of the baboons underneath, and then brings the record back for analysis. The project uses computer vision techniques to automate tracking and prediction of baboon movement in video footage, helping scientists better interpret how baboons collectively arrive at decisions.

Since the computer vision techniques involve multiple stages and take a huge amount of time for post-processing, run-time has been a bottleneck for the project. As the current codebase is implemented in Python, one way to speed up post-processing is by implementing the **core functionalities** in a machine level language like C++ and further implementing it in CUDA where we can take advantage of parallel execution. Our goal of the project for this quarter is to implement/evaluate the post-processing stages in C++/CUDA to improve on the performance while ensuring functionality is not compromised.

1.2 Project Approach

In this section, we provide details about our specific goals and our plan to accomplish them.

Our first goal is to successfully port the core functionalities in the current Python implementation to C++ (following a CLI chart) and ensure their correctness.



To accomplish this, we have created the following plan:

- The Baboons on the Move project has been going on for several years (see, e.g., [Crutchfield et al., 2020]) and many have contributed to the project with algorithms implemented for various functionalities, which involve preprocessing of videos, identifying individual baboons, and predicting their movement, among others. Our first step will involve understanding the existing Python-based implementation as well as the command line interface, etc.

- Since the current implementation involves computer vision-based techniques for identifying baboons and tracking and predicting their movement, as the second step, we will perform a literature review on the computer vision algorithms that we are not yet familiar with. This can help us form a better understanding of the existing implementation, which will (a) speed up our process of porting the Python-based implementation to C++, and (b) help us brainstorm potential algorithmic or implementation-wise techniques to optimize the performance of the computer vision algorithms.

- We will then compile a list of algorithms/stages that we will port to C++ and start to port the Python-based implementation to C++.

- Concurrently, we will also create a testing framework for the C++ code base. Once done, we will use the testing framework to ensure functional correctness. This may involve individually verifying the correctness of each ported C++ algorithm, either using automated testing code or human in-the-loop testing.

- Using the list of algorithms/stages our plan is to confirm that (a) all parts of the core algorithms/stages have been ported to C++; and (b) all the functionalities are working as intended.

Our second goal is to show that the runtime performance of the C++-based implementation is $\geq 10\%$ better than the existing Python-based implementation. More specifically, our plan is:

- We will select a few videos of variable lengths, and run both implementations with these videos as input. We will benchmark the runtime of the implementations for each video.

- Since our goal is to optimize the performance of the algorithms, we will also clock the runtime of each individual algorithm/stage using the C++ and Python code bases. This comparison will give us a better (quantitative) idea on which particular algorithm/stage can still be optimized in terms of runtime.

Early stretch goal. Our third goal is to perform a feasibility study beyond porting the existing implementation to C++ and potentially refactor our implementation for better optimization. We will:

- Examine which stages/algorithms can be parallelized. We will divide the algorithms/ stages among team members, and implement them in CUDA. After that, we will compare the CUDA implementation with the C++ implementation and the Python-based implementation for runtime performance.

Ultimate stretch goal. Our final goal is to examine the feasibility of Kalman filter [Welch et al., 1995] for (multi-target) tracking and prediction of the movement of a *group* of baboons. This is challenging because baboons often move behind rocks or in the trees, and we may also risk misidentifying them when they engage in social activities and interact among each other [Washburn and DeVore, 1961, Kang and Wolters, 2020]. Kalman filter has been considered in the Baboons on the Move project for tracking and predicting movement of baboons. If time permits, we will try re-implementing Kalman filter for tracking and identifying a large group of baboons. Since this part of the project is research-oriented and open-ended, it will likely require work beyond this quarter.

1.3 Minimum Viable Product

The current Python code to detect baboons is divided into multiple stages. Each stage has been implemented using different python modules.

The post-processing time for the python codebase is unrealistically long because of the following reasons:

- Each video has to be sufficiently long as it intends to capture all necessary information to understand the group-level decisions of baboon.
- Each video is of "4K" resolution.

Thus, for deployment, it is really important to reduce the runtime of post-processing video detection stage.

We aim to implement the **core functionalities** in C++ with full functional correctness, C++ being a machine language is significantly faster and more memory efficient than Python which is an interpreted language. Hence it is expected to give us some performance improvement over the current codebase. To achieve the Minimum Viable Product(MVP):

- Once we have completed the porting into C++ code, we need to create a testing framework for the same to ensure that the functionality remains the same.

- Then we will carry out a performance benchmarking comparison with the older implemented code to confirm that the new codebase is giving us potential speedup in post-processing.

Since the baseline codebase takes 1.5-2 hrs to post-process 1 min of captured video. If we are able to bring it down by even 10% we would have achieved our MVP. Considering this as the baseline, to improve over the MVP we plan to do the following:

- We can identify the stages which can be benefited from parallel execution and then port those modules in CUDA and run them on GPUs (Nautilus cluster).

The expectation from this step is to achieve a significant speedup over the MVP.

The long term goals for the project, which can require work beyond this quarter are:

- Achieving 10x performance improvement over the existing implementation and bringing down the runtime of post processing stage to less than 10 minutes
- Re-implementing Kalman Filter for multi-target identification, tracking and prediction
- Supporting execution on other platforms apart from Linux

1.4 Constraints, Risk, and Feasibility

Potential stumbling blocks:

- Understanding the existing Python codebase, since it has been developed over years by different developers
- Use of specific libraries in Python which may not exist in C++
- Execution of Python code only on Linux
- Use of docker containers to implement the C++/CUDA code
- Lack of testing platforms for the C++ code
- Re-implement the algorithms to parallelize them on CUDA

Realistically feasible:

- Get some of the algorithms working on C++ which may have similar libraries as Python
- Develop a testing and benchmarking platform for the code
- Feasibility study of some stages on CUDA

Minimize Risks:

- Have a docker container ready for C++ and implement one algorithm per member in C++
- Develop a testing and benchmarking platform for these collectively
- Ensure correct functionality and improved performance
- Move on to the other stages and use the same setup
- Check if any of the stages can be parallelized with CUDA

2 Group Management

2.1 What are the major roles in your group's management?

Development roles are as follows:

- Chris- Project mentor
- Zhi, Baiqiang - C++
- Sananya, Debaditya - CUDA

2.2 How will decisions be made? By leader, consensus?

Decisions will be made by consensus with consultation from the mentor.

2.3 How will you communicate? Facebook Messenger, email, physical meetings, slack, discord, . . . ?

Team communication will be done through Slack and we will be having weekly meetings on Zoom

2.4 How will you know when you're off schedule, and how will you deal with schedule slips?

Check for updates from each person about their milestone in the weekly meetings and ensure we are on schedule. If there are any slips, we will identify the crucial steps to get back on schedule and assign the work among team members who can dedicate extra time in the following weeks.

2.5 Who is responsible for which deliverables and milestones?

- Milestone 1: Port the algorithms to C++ based on CLI chart – All team members
- Milestone 2: Performance benchmarking and optimization – Zhi and Baiqiang
- Milestone 3: Evaluating the usage of CUDA to parallelize the stages – Sananya, Debaditya and Zhi
- Milestone 4: CUDA Implementation of the algorithms that can be ported – Sananya and Debaditya

3 Project Development

3.1 What are the development roles and who will handle them?

- Port algorithms to C++: Divide the algorithms and each team member is responsible for development of their algorithms.
- Performance benchmarking and optimization: Zhi and Baiqiang are responsible for benchmarking and optimization of the algorithms.
- Evaluating which algorithms are parallelizable – Sananya, Debaditya and Zhi will evaluate different algorithms which can give a better performance if parallelized in CUDA.
- Implementation of the algorithms that can be parallelized- Divide the algorithms between Sananya and Debaditya, and each team member will be responsible for the development and testing of their algorithms. The benchmarking and optimization of their algorithms will be done together.

3.2 What hardware/software will you use? What do you have available? What do you need?

No additional hardware would be needed.

3.3 If there is software/hardware that is needed, provide a justification for its cost. Where will you order it? When will it arrive?

- Software- Docker, Kubernetes, VS Code
- Hardware- GPU (accessed through Nautilus Cluster)

3.4 How will you do testing?

Create a framework which matches the detected regions of baboons from Python and C++/CUDA codebase within a margin of error for the given input video .

3.5 How will you do documentation?

All team members will work on documentation together on working meetings through Google docs/slides and Overleaf.

4 Project Milestones and Schedule

Milestone 1 (MVP): Port the algorithms to C++ following the CLI Chart

- Compile a list of stages. Divide the algorithms/stages among group members.
- Create a testing framework for C++ codebase to ensure functional correctness.

Milestone 2 (MVP): Performance benchmarking and optimization

- Compare the runtime of each stages between C++ and Python code bases.
- Optimize the C++ codebase to ensure minimum performance improvement

Milestone 3 (Early Stretch Goal): Refactoring Existing Implemented Code

- Perform feasibility study on parallelizable algorithms with CUDA

Milestone 4 (Long Stretch Goal): CUDA Implementation of the algorithms that can have parallel execution

- Divide the algorithms/stages among group members.
- Testing the CUDA implementation for performance benchmarking.
- Feasibility study of Kalman filter for multi-baboon motion tracking and prediction.

Schedule.

- Week 3: Understanding existing Python based implementation
- Week 4: Literature review on computer vision algorithms for motion tracking/prediction
- Weeks 5 and 6: Porting the implementation to C++ and creating a testing framework to ensure functionality correctness of C++ based code.
- Weeks 7: Benchmarking and optimising code to achieve better performance
- Week 8: Feasibility study of parallelizable algorithms
- Week 9: CUDA implementation and Benchmarking for runtime comparison
- Week 10: Report and Final Video Presentation

References

[Crutchfield et al., 2020] Crutchfield, C. L., Sutton, J., Ngo, A., Zadorian, E., Hourany, G., Nelson, D., Wang, A., McHenry-Crutchfield, F., Forster, D., Strum, S. C., Kastner, R., and Schurgers, C. (2020). Baboons on the move: Enhancing understanding of collective decision making through automated motion detection from aerial drone footage.

[Kang and Wolters, 2020] Kang, J. and Wolters, C. (2020). Baboons on the move. *UCSD CSE 237D Final Report*.

[Stueckle and Zinner, 2008] Stueckle, S. and Zinner, D. (2008). To follow or not to follow: decision making and leadership during the morning departure in chacma baboons. *Animal Behaviour*, 75(6):1995–2004.

[Washburn and DeVore, 1961] Washburn, S. L. and DeVore, I. (1961). The social life of baboons. *Scientific American*, 204(6):62–71.

[Welch et al., 1995] Welch, G., Bishop, G., et al. (1995). An introduction to the kalman filter.