Baboons on the Move

Final Oral Presentation

Debaditya Basu, Sananya Majumder, Zhi Wang, Baiqiang Zhao

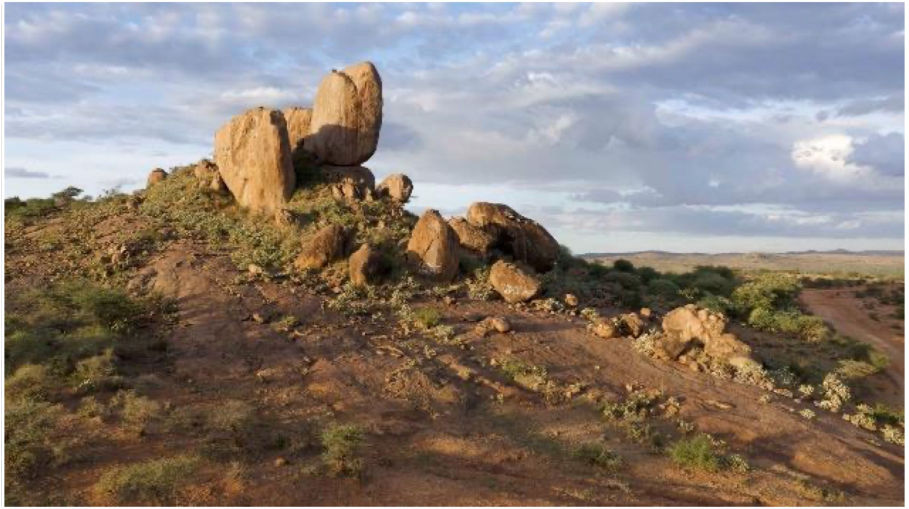
Project Overview

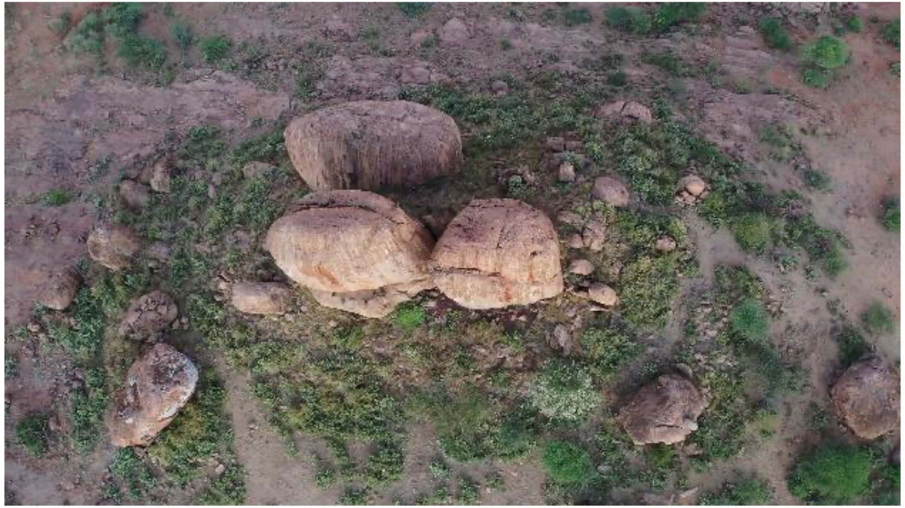
“The behavior of monkeys and apes has always held great fascination for [humans]”.

– Washburn and DeVore, 1961, "The Social Life of Baboons."

© Love Nature

Ground View vs Top View





Baboons on the Move: Use Computer Vision to Track Baboon Movement

Overview

- Key problem: The baboon-tracking algorithms takes a *huge* amount of time for post-processing in Python.

  Solution: Implement the algorithms in C++ and CUDA where we can take advantage of parallel execution.

Python

C++/CUDA

What did we do? - MVP

- Port the algorithms to C++
  - Divide the algorithms/stages among group members
  - Create a testing framework for C++ code

➔ % of closely-matching bounding boxes (CMBB) (C++ vs Python) : 77%

➔ CMBB IoU (C++ vs Python): 72%

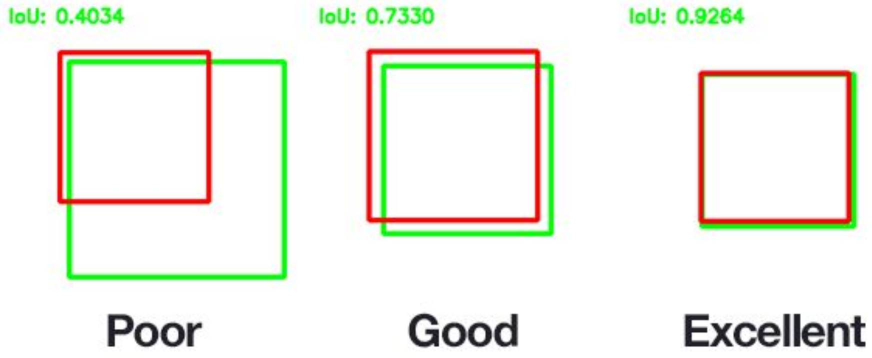


Figure from (Cowton et al, 2019)

What did we do? - MVP

- Performance benchmarking
  - Compare the runtime of each stage between C++ and Python code
  - Optimize the C++ code to ensure minimum performance improvement

➔ 2.5x runtime speed-up with C++ over Python

➔ 1.5x runtime speed-up of optimised stage (Compute Moving Foreground) in C++

What did we do? - Stretch Goals

- Refactoring existing implemented code
  - Perform feasibility study on parallelizable algorithms with CUDA

➔ Analyzed the stages which would use matrices for optimum parallelization

- CUDA implementation of the algorithms that can be ported
  - Divide the algorithms/stages among group members
  - Testing the CUDA implementation for performance benchmarking

➔ % of closely-matching bounding boxes (CMBB) (CUDA vs C++) : 92%

➔ CMBB IoU (CUDA vs C++): 80%

➔ 16x runtime speed-up with CUDA over C++

Performance Improvements

Stage	C++ Existing Implementation	C++ Refactored Implementation	Speed Up
Compute Moving Foreground	111ms	77ms	1.44 ✅

C++ vs CUDA:

Stages	C++	CUDA	Speed Up
Blur Gray	2834us	430us	6.6 ✅
Compute Transformation Matrices	2635ms	30ms	87.8 ✅
Erode Dilates	19ms	25ms	0.76 ❌
Total	3657ms	228ms	16.03 ✅

Problems faced and major challenges

- Execution of Python code only on Linux platform
- Understanding the Python code and the process to run it using CLI script
- Setting up docker container on Nautilus to implement the CUDA code

What did we not do?

- Implementing Kalman Filter for multi-baboon tracking and prediction
- Supporting execution on other platforms apart from Linux
- Improve the existing Python code and CLI script

Project Demonstration

Conclusion

- **Objective**: Improve the performance of baboon tracking algorithms by reimplementing them in C++ and CUDA

- **What we have done so far:**
    - Ensured functionality of the stages in C++ code
    - Benchmarked the performance in C++ code
    - Feasibility study of stages with CUDA
    - Ensured functionality of the stages in CUDA code
    - Benchmarked the performance in CUDA code

- **What's next?** Further optimize the runtime of certain stages and display output in CUDA