Benchmarking Gradient-Free Optimizers for 3D Performance Capture in the ∇ Nevergrad platform

Alexandros Doumanoglou
Nikolaos Zioulis*
aldoum@iti.gr
nzioulis@iti.gr

Vladimiros Sterzentsenko*
Antonis Karakottas*
vladster@iti.gr
ankarako@iti.gr

Dimitrios Zarpalas
Petros Daras
zarpalas@iti.gr
daras@iti.gr

Centre For Research and Technology
HELLAS
Thessaloniki, Greece

ABSTRACT

In this paper we document our work that extends the ∇ nevergrad platform by adding a real-world benchmark at the intersection of computer vision and graphics: 3D Performance Capture. Detailed documentation can be found at vcl3d.github.io/nevergrad/

CCS CONCEPTS

• **Theory of computation → Optimization with randomized search heuristics**.

KEYWORDS

Benchmarking, Open Source, Optimization, Performance Capture, Nevergrad

ACM Reference Format:
Alexandros Doumanoglou, Nikolaos Zioulis, Vladimiros Sterzentsenko, Antonis Karakottas, Dimitrios Zarpalas, and Petros Daras. 2021. Benchmarking Gradient-Free Optimizers for 3D Performance Capture in the ∇ Nevergrad platform. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3463281

1 INTRODUCTION

In this work we introduce a new benchmark to the ∇ nevergrad [4] platform for the real-world application of 3D Human Performance Capture. We consider the case of fitting an animatable, skinned, template 3D mesh to a post-processed 3D-reconstructed mesh created from sensed noisy volumetric data.

For the purposes of the benchmark we create and release[1] a volumetric dataset consisting of 11 human performances, recorded via a volumetric capture system [5]. For each performance, a skinned 3D template mesh of the performer has been automatically generated. Also, among all input volumetric frames, one individual frame is selected as a fitting target for the template mesh. The fitting of the animatable template mesh, is accomplished via solving for its animation (*i.e.* pose) parameters that better explain the target frame's captured performance data. The latter is a 3D reconstructed mesh [2] at each frame of the captured volumetric sequence. The quality of the fit is measured by a function which linearly combines multiple error terms (3D, projective 2D, and priors) into a single objective. This work aims to provide a fully automated benchmark to assess the performance of any ∇ nevergrad optimizer to the performance capture template fitting task. A detailed discussion on the fitting problem, the objective function, and evaluation results of the most popular ∇ nevergrad optimizers to this problem can be found in [1].

2 SOFTWARE

2.1 Benchmark Application Components

The objective function evaluation, used in this benchmark, requires loading and 3D processing of the recorded depthmaps, computing a 3D distance field in the GPU, for efficient run-time, as well as multiple GPU-accelerated rendering operations among others. Implementing the objective function evaluation pipeline in Python would be both impractical and costly in terms of development time and run-time performance, not to mention the increased code-base size which would be required to import in ∇ nevergrad . Thus, we chose to develop an external C++ application executable (called *Performance Capture Benchmark Server*), which is distributed in Windows binary form[2] which serves objective function evaluations. Its interface is decoupled via a RabbitMQ messaging broker [3]. On the ∇ nevergrad side, the implemented class realizing the objective function, talks to the benchmark server via RabbitMQ messages, whilst exposing a blocking call interface, hiding the behind-the-scenes asynchronous messaging. Thus, this benchmark comes with two major software components: the **C++ server** and the ∇ nevergrad **Python client**.

2.2 Installation, Requirements & Execution

Out-of-the-box execution of this benchmark, with minimal configuration, requires that the RabbitMQ server, the benchmark server and ∇ nevergrad , are all installed on the same machine[3]. The benchmark server requires an NVIDIA GPU supporting NVIDIA CUDA

*Authors with equal contribution
[1]github.com/VCL3D/PerformanceCapture/releases/tag/dataset_1.0

[2]github.com/VCL3D/PerformanceCapture/releases/tag/1.0
[3]Currently only Microsoft Windows 10 Operating System is supported

v9.2 and OpenGL v4.6. Regarding the ∇ nevergrad Python side, compatibility is ensured with `Python v3.8.2` with additional dependencies on `aio-pika v6.6.1`, `py7zr v0.11.3` and `multivolumefile v0.1.2` Python packages.

Before running the benchmark, users should edit the `uri` entry in `/nevergrad/functions/perfcap3d/configuration/rmqsettings.json` to match the RabbitMQ connection string of their RabbitMQ setup. The only thing to edit is the username and password, corresponding to the RabbitMQ server account.

Benchmark execution is accomplished via the standard ∇ nevergrad benchmark command: `python -m nevergrad.benchmark perfcap`. On the first run, the ∇ nevergrad Python client will automatically download the benchmark server executable and the dataset ($\geq 30GB$ free disk space required) and install the resources in `/nevergrad/functions/perfcap3d/resources` folder.

2.3 Configuration Files

For each one of the 11 human performances included in the released dataset, we created one respective experiment configuration file, named `experimentX.json`, `X` $\in \{1, 2, ..., 11\}$, and located in `/nevergrad/functions/perfcap3d/configuration`. One benchmark experiment configuration file contains several parameters related to that experiment. Running the `perfcap` benchmark, consists of executing all individual experiments one by one for 10 optimizers and 3 different budgets, which, nonetheless, is configurable in the source-code. The included experiment configuration files contain the parameter values that we used for the *local* experiments described in [1]. However, new experiments can be added at will, by adding their configuration file to the aforementioned folder. Each experiment configuration file contains a reference to a `.perfproj` project file containing metadata about the multi-view recorded performance and the skinned 3D template mesh. Additionally, a reference to the target frame index of the recorded performance is provided and individual weights for the linear combination of objective function error terms. Finally, these files also include values defining the initial pose of the 3D template, and extensive configuration, on the overall degrees of freedom (*i.e.* variables) of each joint, the variable bounds, the variable mutation variances, and boolean flags indicating whether mutation variances are mutable by themselves. When the `relative_search_space` is set to `true`, all pose variables are considered relative to the initial pose, while in all cases, the pose values for the degrees of freedom excluded by each joint are fixed to the respective values from the initial pose. Essentially, the experiment configuration files cover the parametrization of the objective function in the number of variables, variable bounds, error terms and mutation variances.

2.4 Data Viewing, Error Term Visualizations and Debugging

When the benchmark server executable (`performance_capture.exe`) is run without the `--benchmark_server` flag, and no other command line arguments, it can be used to load a `.perfproj` file in order to inspect recorded data and see some visualizations of selected individual error terms. Apart from integrating a typical player functionality to allow 3D viewing of the performance capture data, the application can be used to view and inspect the created 3D template at its initial pose, as well as visualize in 2D and 3D the values of the individual objective function error terms. Through manual manipulation and editing of the template's pose parameters, the users are able to trigger objective function evaluations and visualizations of the selected error terms. This functionality aims to serve as a means for researchers and practitioners to familiarize with the problem and debug potential issues when developing new or evaluating existing optimization algorithms.

3 EXPERIMENT RESULTS & LOGGING

Each individual instance of an experiment run, receives an `experiment_tag_id`, uniquely identifying the optimizer, budget and repetition number, under which the experiment was run. The `experiment_id_tag` is also appended in the ∇ nevergrad logs and is an identifier which can help in associating ∇ nevergrad logs with benchmark server logs.

For each finished instance of an experiment identified by `experiment_tag_id`, the benchmark server writes 3 kinds of logs (all located in `/nevergrad/functions/perfcap3d/resources/benchmark_server`):

- `/logs/{experiment_tag_id}.json`: for each iteration of the optimization, this file contains the objective function query point (animation pose), the individual error term values, and the aggregated (weighed) error value, of the objective function at this query point.
- `/live_meshes/experiment_{experiment_id}.ply`: the 3D mesh that corresponds to the target frame of the performance that the animated template should fit. The `experiment_id` values can be found in `experiment_{X}.json` configuration files.
- `/animated_meshes/{experiment_tag_id}.ply`: these files contain the animated template 3D mesh at the optimizer's recommendation pose, after the end of the optimization.

4 FUTURE EXTENSIONS

Running the benchmark on other operating systems, requires a distributed setup, which however, is not currently supported in an automated way and can be considered for future extension. Another potential future extension regards treating the problem as a multi-objective optimization problem by splitting the objective function to its individual error terms. This is straightforward to accomplish and does not require any modification to the performance capture benchmark server executable.

5 ACKNOWLEDGEMENT

This work has been supported by the EU funded H2020 Program: INFINITY, Grant Agreement 883293.

REFERENCES

[1] A. Doumanoglou, P. Drakoulis, K. Chrstaki, N. Zioulis, V. Sterzentsenko, A. Karakottas, D. Zarpalas, and P. Daras. 2021. Zeroth-Order Optimizer Benchmarking for 3D Performance Capture: A real-world use case analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference 2021 (GECCO '21)*.
[2] Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 3 (2013), 1–13.
[3] RabbitMQ 2021. - Messaging that just works. https://www.rabbitmq.com/.
[4] J. Rapin and O. Teytaud. 2018. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad.
[5] Vladimiros Sterzentsenko, Antonis Karakottas, Alexandros Papachristou, Nikolaos Zioulis, Alexandros Doumanoglou, Dimitrios Zarpalas, and Petros Daras. 2018. A low-cost, flexible and portable volumetric capturing system. In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 200–207.