# Guide for reproducing the results presented in the paper "An Indexing Scheme and Descriptor for 3D Object Retrieval Based on Local Shape Querying"

Bart Iver van Blokland

August 2020

## 1 Overview

Greetings!

The repository in front of you contains a (hopefully batteries included) package for the reproduction of all results presented in the paper. This guide is intended to be a series of step by step instructions. You should follow them in the order they are listed.

## 2 Background

Before we start, we should take a bird's eye view over all experiments in the paper's evaluation. It hopefully gives you an idea what to expect of the different experiments we've done. For any details you can of course refer to the paper.

#### Hamming Tree (Figure 7)

The hamming tree is a data structure for indexing binary images, which in the paper we're applying on the proposed QUICCI images. The idea is that you take a collection of binary images and produce a tree. You can subsequently use this tree to determine which images have the closest Hamming distance to a given image.

The evaluation in the paper only involves measuring the execution time of such queries.

## Clutterbox Experiment (Figures 8-11)

Most of the results in the paper are obtained from what's referred to as the 'clutterbox experiment'. In broad terms, it selects 10 3D objects (for instance a lamp, a plane, a sofa, ..) at random from a large dataset, and using one or more descriptor algorithms, computes some histograms.

During the experiment, a large number of search result pages of the kind you might find on Google is generated. When searching for something, you want what you look for to be on top (rank 0), but sometimes can find it further down the list. The histogram summarises these search results by counting in bin $n$ how often the desired search result was found a rank $n$ in the list of search results.

This procedure is repeated for a scene containing 1 object (of the 10 selected) only, a scene with 5, and one with all 10. []In turn, we performed this procedure on three descriptor algorithms; the Radial Intersection Count Image, the Spin Image, and the 3D Shape Context.

You will therefore see 1 histogram produced for every combination of object count and descriptor when running the experiment.

## Clutter Estimation (used for Figure 9)

In the paper, Figure 9 contains some heatmaps which require the degree of 'mess' (clutter) to be estimated in the vicinity of parts of an object. To visualise this, consider an empty room on which two identical cups have been placed at opposite ends. Now hide one of those two cups in between some boxes, a toothbrush, and some clothes. The degree of clutter surrounding this cup is said to be higher than that at the other one.

This degree of clutter is measured at each vertex of the 3D object. You will therefore see a long list of floating point values between 0 and 1, where 0 represents surroundings that are entirely clean.

## QUICCI distance function evaluation (Figures 12 and 13)

The paper proposes two ways of ranking QUICCI search result images for relevance to a given query. We put together an experiment to see which of the distance functions (along with Hamming distance) would make for the best one for retrieval purposes. We tested each function under two conditions; where the images being compared are very different, and where the images being compared are very similar.

The experiment is essentially producing many image pairs, computing distance function values for each pair, and creating a histogram counting how often each distance value occurred. For the cases where images are similar, we stick spheres on the object's surface to create pairs that are similar, but not entirely so. That's why for those you will see heatmaps being produced rather than simple line charts, as we thought this was the best way of visualising an array of histograms.

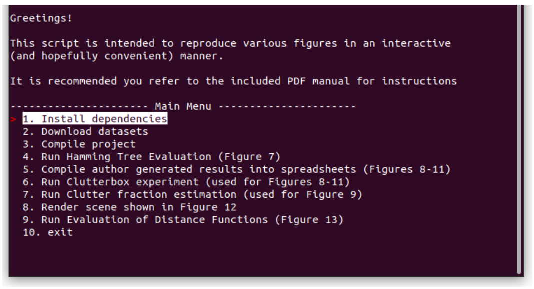
## 3 Preparation

### 3.1 Install Dependencies

Start the 'replicate.py' script found in the root of the repository:

```
python3 replicate.py
```

You will be greeted with the main menu:

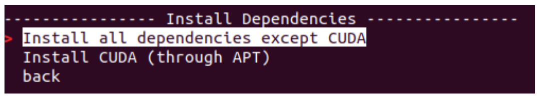


```
Greetings!

This script is intended to reproduce various figures in an interactive
(and hopefully convenient) manner.

It is recommended you refer to the included PDF manual for instructions

--------------------- Main Menu ---------------------
> 1. Install dependencies
  2. Download datasets
  3. Compile project
  4. Run Hamming Tree Evaluation (Figure 7)
  5. Compile author generated results into spreadsheets (Figures 8-11)
  6. Run Clutterbox experiment (used for Figures 8-11)
  7. Run Clutter fraction estimation (used for Figure 9)
  8. Render scene shown in Figure 12
  9. Run Evaluation of Distance Functions (Figure 13)
  10. exit
```

You can navigate these menus using the arrow keys, and press 'enter' to select.

The first step is to install all dependencies and download the auxiliary datasets (which were too large to host on github directly).

At this end, select the top entry called "Install dependencies":



```
--------------- Install Dependencies ---------------
> Install all dependencies except CUDA
  Install CUDA (through APT)
  back
```

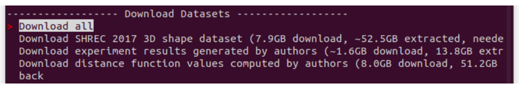
We have separated the dependencies into 'the CUDA SDK' and 'everything that is not the CUDA SDK', as when CUDA has been installed using the runfile method from NVidia's website, installing it through APT can render both installations unusable.

If you're installing the project on a fresh installation of Ubuntu, everything should work out of the box when you use the APT method.

Make sure all dependencies are met before proceeding.

## 3.2 Download Datasets

Return to the Main Manu, and select the option "2. Download datasets".



```
----------------- Download Datasets -----------------
> Download all
  Download SHREC 2017 3D shape dataset (7.9GB download, ~52.5GB extracted, neede
  Download experiment results generated by authors (~1.6GB download, 13.8GB extr
  Download distance function values computed by authors (8.0GB download, 51.2GB
  back
```

The SHREC 2017 is the dataset containing sample objects. Pretty much all experiments use this, so downloading this is a must.

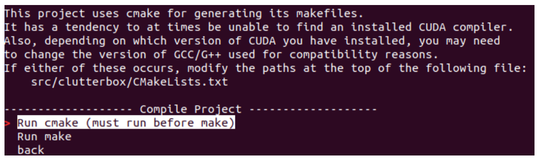
The other sets are used for various figures (the menu entries indicate which).

We recommend that if your disk has enough space available (you'll need ĩ35GB), just hit "download all", and fetch a cup of coffee.

## 3.3 Compile Project

The next step is to compile the C++/CUDA source code that comes with the repository. Select the "Compile project" entry in the Main Menu to do so, and run the 'cmake' and 'make' entries in order.

If you have installed CUDA through the runfile method, CMake may be unable to find the CUDA installation. The CMakeLists.txt file located at src/clutterbox/CMakeLists.txt in the repository contains some lines that manually specify the CUDA SDK location, which can be found near the top of the file. Uncommenting these should normally do the job.



```
This project uses cmake for generating its makefiles.
It has a tendency to at times be unable to find an installed CUDA compiler.
Also, depending on which version of CUDA you have installed, you may need
to change the version of GCC/G++ used for compatibility reasons.
If either of these occurs, modify the paths at the top of the following file:
    src/clutterbox/CMakeLists.txt

------------------- Compile Project -------------------
> Run cmake (must run before make)
  Run make
  back
```

## 4 Reproduction of Figures Shown in the Paper

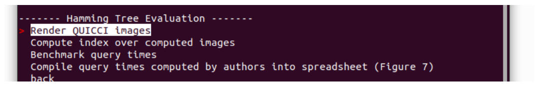
### 4.1 Figure 7: Hamming Tree

For testing the Hamming tree, we started by computing QUICCI images for each object in the SHREC 2017 dataset. This resulted in a dataset of 431.9GB (compressed). We subsequently computed an index over these images from 12500 / 51000 objects, resulting in a database size of 165.8GB (also compressed).

Both of these sets are extremely large, and we did not find a permanent location to host them, so instead we'll make them available on request. Since the Figure presents execution times, exact reproduction is probably not feasible anyway.

The reproduction of this Figure therefore consists of two parts; a way to create and benchmark a Hamming Tree from raw 3D object (OBJ) files, and the script that converts the benchmark result files into the chart shown in the paper.

At this end, select "Run Hamming Tree Evaluation" from the main menu, and run "Render QUICCI images". For reasonable execution times, we recommend rendering images for something in the order of 25 objects.



```
------- Hamming Tree Evaluation -------
> Render QUICCI images
  Compute index over computed images
  Benchmark query times
  Compile query times computed by authors into spreadsheet (Figure 7)
  back
```

Next, run the "Compute index over computed images" and "Benchmark query times" options in that order. You can run the latter any number of times to try different query images. The program will first do a query using the constructed index, and subsequently another using a sequential search. You may also notice it shows a rough visualisation of the query image.

Upon completion of each of these benchmarks, a JSON file is written with timings. We ran 1082 of those on our index, and the final option in the menu titled "Compile query times computed by authors into spreadsheet" compiles these into the chart shown in Figure 7. After executing this, open the file output/figure_7_query_times.csv, and create an XY scatterplot using the file's contents. The sequential search times are not here; due to their long execution times those were based on the average of 6 iterations (their run to run variance was not very high).

## 4.2 Figures 8-11: Computing the primary results spreadsheet

Rather than starting at the beginning of the pipeline, we will first use the results we computed as part of our evaluation to recreate the matching performance charts shown in the paper. The primary reason for this is that a part of this step also computes an auxiliary file we need for later.

The first step is to compile the results for the Clutterbox experiment and clutter estimation, obtained as part of our evaluation, into a spreadsheet from which the charts can be put together.

At this end, run the "Compile author generated results into spreadsheet" entry from the Main Menu.

The script produces three things:

1. The exact heatmaps shown in Figure 9 in the paper
2. A spreadsheet containing all necessary data to create Figures 8, 9, 10, and 11, which is written to 'output/charts_spreadsheet.xls' in the repository
3. A mapping file which allows the script to point out which files should be compared when running the clutterbox experiment (the aforementioned auxiliary file).

Some other things of note regarding this script:

- The script needs to perform some splitting and merging of different datasets. It will therefore load all obtained results into memory, which means it consumes a relatively large amount of it.
- You may notice the descriptor name 'QSI' being used in places. It was used as a working title for the RICI descriptor presented in the paper. We did not feel it was right to backpatch this name into results that had already been obtained.
- After computing and showing the heatmaps, the script will wait until you press enter, to allow you to see/save the heatmap images.
- While the paper only presents 1500 results, a set of 1600+ seeds were used to generate them. When for a given seed a single descriptor+object

count combination is missing (most commonly due to the GPU running out of VRAM), all other results for that seed are discarded. This allowed for some margin of error. The final result set has approximately 1560 valid results, and is cut down to 1500. Worth noting is that this uses the file system ordering (not intentionally so), so we have included an extra filtering step that ensures the right subset of seeds is used for generating the charts.

- The FPFH results took an extremely long amount of time to compute, and we had to cut it short at 500/1500 experiments. Unfortunately, 28 of these seeds (5.6%) were part of the set of 1600, but not of the 1500 used for the other descriptors. This is the reason you'll see some extra rows for FPFH in the different spreadsheets.

#### 4.2.1 Reproducing the figures themselves

The spreadsheet 'output/charts_spreadsheet.xls' created in the previous step contains one sheet per chart. To recreate the charts, select all columns in the respective sheet and create an XY scatter plot, ensuring that the x-axis is always set to the leftmost column.

For comparison, the spreadsheets used by the authors to create the charts shown in the paper are supplied in the directory
'output/chart_spreadsheets_created_by_authors'. Note that for Figure 11 (generation rate), the order of the rows in the sheet may be different relative to the one created by us. However, ultimately the charts themselves should be exactly the same.

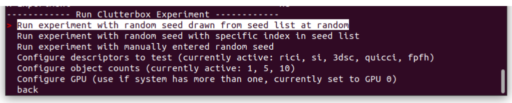
#### 4.2.2 Reproduction of data underlying the charts

We're now taking going to focus on reproducing the data underlying the charts from the previous steps. This data has two sources; results from the Clutterbox experiment, and the heatmaps require an estimate of clutter in addition.

Computing the entire result set is not exactly feasible within a reasonable amount of time. We'll therefore focus on reproducing the results of individual experiments.

**Reproducing Clutterbox results**

From the Main Menu, select the "Run Clutterbox experiment" option, which will bring you to the following menu:

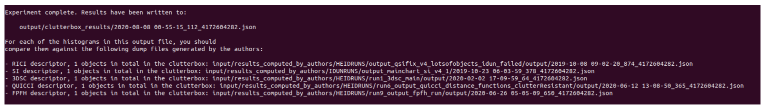


```
------------ Run Clutterbox Experiment ------------
> Run experiment with random seed drawn from seed list at random
  Run experiment with random seed with specific index in seed list
  Run experiment with manually entered random seed
  Configure descriptors to test (currently active: rici, si, 3dsc, quicci, fpfh)
  Configure object counts (currently active: 1, 5, 10)
  Configure GPU (use if system has more than one, currently set to GPU 0)
  back
```

If your system has more than one GPU, make sure you use the second to last option to configure which one to execute kernels on first. The other options configure parameters of the experiment.

The random seed used for this experiment determines all randomly chosen parameter of the experiment (such as the set of objects selected). The top option in the menu allows you to run a random seed selected from the pool of 1600+ used to generate our results.

When the experiment completes, it writes an output file in JSON format. Using an auxiliary file computed in the previous step, it will also be able to point you at the specific files from the dataset computed by us:



```
Experiment complete. Results have been written to:

    output/clutterbox_results/2020-08-08 00-55-15_112_4172604282.json

For each of the histograms in this output file, you should
compare them against the following dump files generated by the authors:

- RICI descriptor, 1 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/output_qsifix_v4_lotsofobjects_idun_failed/output/2019-10-08 09-02-20_874_4172604282.json
- SI descriptor, 1 objects in total in the clutterbox: input/results_computed_by_authors/IDUNRUNS/output_mainchart_si_v4_1/2019-10-23 06-03-59_378_4172604282.json
- 3DSC descriptor, 1 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/run1_3dsc_main/output/2020-02-02 17-09-59_64_4172604282.json
- QUICCI descriptor, 1 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/run6_output_quicci_distance_functions_clutterResistant/output/2020-06-12 13-08-50_365_4172604282.json
- FPFH descriptor, 1 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/run9_output_fpfh_run/output/2020-06-26 05-05-09_650_4172604282.json
```

When interpreting these files, scroll down past the experiment's metadata to the [QUICCI/RICI/SI/3DSC/FPFH]histograms key(s). For the RICI and SI results generated by us, the entry will contain histograms labelled '0', '1', or '2'. These refer to the index of the list of object counts for that execution of the experiments. You can find that list in the metadata near the top of the file under the 'sampleObjectCounts' key. The QUICCI, 3DSC, FPFH results, and the implementation which comes with this repository labels these in a more readable way; '1 objects', '5 objects', and '10 objects'.

You should compare the histograms you find in the output file produced by the experiment with those in our results.

One important note here: while the QUICCI, RICI, and FPFH histograms should be completely the same, there will be slight variations in the histograms of SI and 3DSC. While we have controlled all variables for the inputs of these methods, due to these descriptors accumulating floating point numbers, there will inherently be variations in the final sum due to the GPU accumulating numbers in parallel in different orders. These differences should nonetheless be small.

We have however seen some slight differences between the produced QUICCI and RICI histograms, which best we can tell is caused by a change in the CUDA compiler causing slightly different floating point instruction ordering in the GPU kernels.

We also have some recommendations:

- Keep a terminal window open in the root of the repository, and copy the paths printed out by the script directly into a parameter of 'cat' or 'vim'. Finding these specific files in the file manager takes a while.
- Particularly on GPU's with less VRAM, the experiment has a tendency to run out of VRAM. You might want to consider only testing one descriptor at a time, since the results output file is only written if the experiment executes in full.
- For the same reason, we recommend testing the 10 object results separately from those with 1 and 5 objects.
- After running the spreadsheet generation script from the previous step, you can open the file 'output/master_spreadsheet.xls', select the sheet called 'Total Image Count', and use this to determine which random seed will generate many images (and therefore will take long to finish) for a given object count you're about to test. If you are very limited in the amount of available VRAM, this can also be used to find random seeds that are likely to fit.
- FPFH is SLOW. Which is also the reason the paper only has 500 results for it, rather than the 1500 used for the other tested methods.

The timing results can be difficult to reproduce if you do not have the same GPU model that we used in our experiments (the NVidia Tesla V100 SXM3), but if you do, the results are all based on scenes with 5 objects.

**Reproducing clutter estimates**

Reproducing the clutter estimate files functions much in the same way as the Clutterbox ones, and can be found under 'Run Clutter fraction estimation' in the Main Menu. We recommend you use the option to run a random file index; most of these do not take long to execute on most GPU's. The script will write an output file, and point you to the one generated by us. We have not controlled the random seed used to generate clutter files (not on purpose), but the run to run variations seem to be within an error of 0.01 (where clutter fractions range between 0 and 1).

## 4.3 Figures 12 and 13: Distance Function Evaluation

The final experiment of the paper is the one comparing the distance values produced by three different distance functions (which quantify similarity of image pairs).

Figure 12 is quite easy to reproduce; just run the entry from the main menu, and it produces an OBJ file containing the sofa with spheres on it shown in the paper.

Figure 13, as with the Clutterbox experiment before, consists of a step which generates JSON dump files, and one which converts those dump files computed by us into the charts and heatmaps shown in the paper. We provided the latter because the former takes a significant amount of computation time.

### 4.3.1 Generating distance function response dump files

The top 4 entries in the Figure 13 submenu are dedicated to computing the responses for each of the three distance functions tested. The “nominal” results are used in Figure 13a (the one on top), while the “similar” results are those used to generate the heatmaps in Figure 13b.

As with the CLutterbox experiment, a single random seed determines which object(s) are selected from the dataset. You can for both pick one such seed that was used in the evaluation at random, or select a specific one from the list by its index.

After running the respective evaluation, two file paths will be printed to the terminal; one that was just produced, and one that was generated by us. The lists of numbers you find in these JSON files should be identical.

### 4.3.2 Computing the heatmaps and charts shown in Figure 13

The fifth option in the menu (“Compile results computed by authors ..”) reads all dump files computed by us, and produces an XLS spreadsheet with the contents of Figure 13a, and the heatmaps from Figure 13b.

The histograms of Figure 13a can be found in the spreadsheet named ‘nominal_distance_function_result_histograms.xls‘ located in the ‘output/distance_function_evaluation/‘ directory.

The values in this spreadsheet should be identical to the one created by us called ‘baseline_distance_distributions.ods‘ in the directory ‘output/chart_spreadsheets_by_authors‘.

The heatmaps produced by the script should be identical to those shown in Figure 13b.