Guide for reproducing the results presented in the paper “Radial Intersection Count Image: a Clutter Resistant 3D Shape Descriptor”

Bart Iver van Blokland

July 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, and the data produced by them, in the paper’s evaluation. We’ll focus on the bits that are relevant to determine the validity of reproduced results. For details on the implemented algorithms themselves, you can refer to the paper.

Clutterbox Experiment

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

In the paper, Figure 12 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.

Projection Benchmark

The paper proposes a new method for projecting 3D points in a 2D coordinate space referred to as 'cylindrical coordinate space'. We created a benchmark which compares the execution time of a conventional implementation relative to ours.

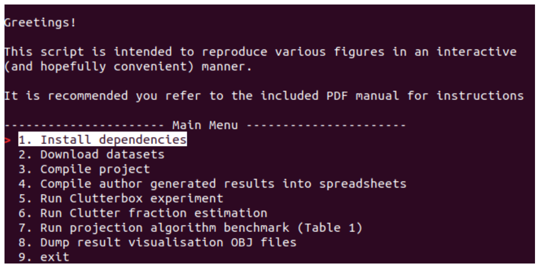
3 Preparation

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:

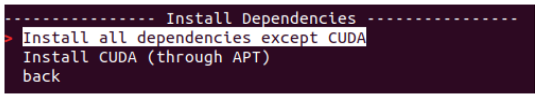


```
Greetings!

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. Compile author generated results into spreadsheets
  5. Run Clutterbox experiment
  6. Run Clutter fraction estimation
  7. Run projection algorithm benchmark (Table 1)
  8. Dump result visualisation OBJ files
  9. 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
```

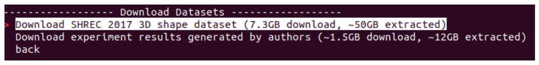
We have separated the dependencies into ‘the CUDA SDK’ and ‘everything that is not the CUDA SDK’, as when CUDA has been installed using the run-file method from NVidia’s website, installing it through APT can render both installations unusable.

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 SHREC 2017 3D shape dataset (7.3GB download, ~50GB extracted)
  Download experiment results generated by authors (~1.5GB download, ~12GB extracted)
  back
```

The SHREC 2017 is the dataset containing sample objects used as input for the Clutterbox experiment.

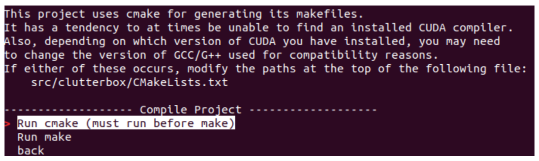
The experimental results consist of all output generated by us while we executed the Clutterbox experiment and clutter estimation.

Use the script to download each of these. It will automatically download the archives and extract them in the correct location.

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 Charts Shown in the Paper

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 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.

4.1 Computing the primary results spreadsheet

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 12 in the paper
2. A spreadsheet containing all necessary data to create Figures 10, 11, 13, and 14, 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

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.

4.2 Reproducing Figures 10, 11, 13, and 14

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.

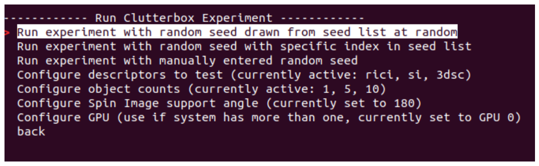
For comparison, the spreadsheets used by the authors to create the charts shown in the paper are supplied in the directory
'output/chart_spreadsheets_created_by_authors'. Note that for Figure 13 (generation rate), the order of the rows in the sheet may be different relative to the one created by us. However, ultimately the charts themselves should be exactly the same.

5 Reproduction of the data underlying the charts

The aforementioned charts require two sources of data; results from the Clutterbox experiment, and the heatmaps require an estimate of clutter in addition.

5.1 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)
  Configure object counts (currently active: 1, 5, 10)
  Configure Spin Image support angle (currently set to 180)
  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.

For the results shown in Figure 10, use the following settings:

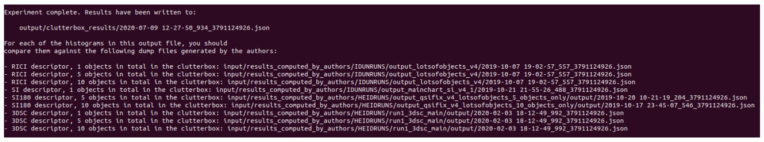
- Descriptors: rici, si, 3dsc
- Object counts: 1, 5, 10
- Spin Image support angle: 180

For the results shown in Figure 11, use the following settings:

- Descriptors: si
- Object counts: 1, 5, 10
- Spin Image support angle: 60 or 180 (depending on which curve you want to reproduce)

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/2020-07-09 12-27-50_934_3791124926.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/IDUNRUNS/output_lotsofobjects_v4/2019-10-07 19-02-57_557_3791124926.json
- RICI descriptor, 5 objects in total in the clutterbox: input/results_computed_by_authors/IDUNRUNS/output_lotsofobjects_v4/2019-10-07 19-02-57_557_3791124926.json
- RICI descriptor, 10 objects in total in the clutterbox: input/results_computed_by_authors/IDUNRUNS/output_lotsofobjects_v4/2019-10-07 19-02-57_557_3791124926.json
- SI descriptor, 1 objects in total in the clutterbox: input/results_computed_by_authors/IDUNRUNS/output_mainchart_si_v4_1/2019-10-21 21-55-26_488_3791124926.json
- SI180 descriptor, 5 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/output_qsifix_v4_lotsofobjects_5_objects_only/output/2019-10-20 10-21-19_204_3791124926.json
- SI180 descriptor, 10 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/output_qsifix_v4_lotsofobjects_10_objects_only/output/2019-10-17 23-45-07_546_3791124926.json
- 3DSC descriptor, 1 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/run1_3dsc_main/output/2020-02-03 18-12-49_992_3791124926.json
- 3DSC descriptor, 5 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/run1_3dsc_main/output/2020-02-03 18-12-49_992_3791124926.json
- 3DSC descriptor, 10 objects in total in the clutterbox: input/results_computed_by_authors/HEIDRUNS/run1_3dsc_main/output/2020-02-03 18-12-49_992_3791124926.json
```

When interpreting these files, scroll down past the experiment's metadata to the [RICI/SI/3DSC]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 3DSC 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 RICI 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 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.

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, and the spin image support angle should be set to 180.

5.2 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).

6 Reproduction of Projection Algorithm Benchmark

This benchmark is used to compute the results shown in Table 1 in the paper.

From the Main Menu, select 'Run projection algorithm benchmark', and let it run to completion. After testing each method it will print its execution time. Our method should perform better here than the one we tested against.

It also allocates around 30GB of RAM. It is therefore recommended you ensure the system has that much available before starting it.

7 Reproduction of Matching Performance Visualisations

Figure 15 and 16 were added to show matching performance of different methods under specific conditions that occurred. This is accomplished by dumping textured OBJ files during the execution of the Clutterbox experiment.

The final entry in the Main Menu, named 'Dump result visualisation OBJ files' allows each of these to reproduced.

Figure 15 shows all combinations of object counts and descriptors. The settings allow you to generate results for the desired combination of them. For the Spin Image, however, make sure the support angle is set to 180. The scene OBJs on the left hand side of each subfigure are generated automatically when the corresponding object count is selected.

The OBJ files are written to subfolders in 'output/highlightedobjects'. Opening them with any 3D object viewer should produce the same images as those shown in the paper (we used Meshlab for this purpose).

Finally, Figure 16 requires specific settings, which are automatically configured correctly. It produces OBJ files in a similar fashion to those for Figure 15, and writes them to the same directory.