A. Artifact Appendix

A.1 Abstract

The artifacts consist of the ILLIXR system, test applications, and results scripts.

A.2 Artifact check-list (meta-information)

- **Program:** ILLIXR, Monado, test applications, ILLIXR results scripts.
- **Compilation:** Make 4.2, clang 10.0, CUDA 11.1, Python 3.8 (included in install scripts)
- **Data set:** See §IIIC (included in artifact) and §IIID (downloaded by program).
- **Run-time environment:** Ubuntu 18.04 + install scripts.
- **Hardware:** Any x86-64 system with CUDA can run, but one needs the hardware in §IIIA for exact repeatability.
- **Output:** `results/output/*` and `results/Graphs/*`.
- **Experiments:** For each hardware platform, for each app, run ILLIXR.
- **How much disk space required:** 5Gb, including downloaded datasets
- **How much time is needed to prepare workflow:** 1 hour
- **How much time is needed to complete experiments:** 20 minutes per hardware platform
- **Publicly available:** Yes, see archive URL.
- **Code licenses:** The system licensed under NCSA. Each component is licensed under one of: ElasticFusion License, NCSA, MIT, Simplified BSD, LGPL v3.0, LGPL v2.1, Boost Software License v1.0, GPL v3.0. See `https://github.com/ILLIXR/ILLIXR/#licensing-structure`
- **Data licenses:** Each dataset is licensed under one of: proprietary, ElasticFusion License, Creative Commons 0.
- **Archived (provide DOI):** `https://doi.org/10.5281/zenodo.5520211`

A.3 Description

A.3.1 How to access

One can find the version we used for this paper here (`https://doi.org/10.5281/zenodo.5520211`), and a rolling release here (`https://github.com/ILLIXR/`). We suggest using the rolling release, unless exact repeatability is desired.

A.3.2 Hardware dependencies

One can find the hardware we used for this paper in §IIIA. ILLIXR will still work on a generic x86-64 Ubuntu system, but the results may not be exactly repeatable.

A.3.3 Software dependencies

```
./ILLIXR/install\_deps.sh
python3 -m pip install poetry
sudo apt-get install build-essential scons pkg-config \
  libx11-dev libxcursor-dev libxinerama-dev libgl1-mesa-dev \
  libglu-dev libasound2-dev libpulse-dev libudev-dev \
  libxi-dev libxrandr-dev yasm
scons -C godot -j$(nproc) platform=x11 target=release_debug
cd results/analysis; poetry install; cd ../..
sudo sh -c 'echo 0 >/proc/sys/kernel/perf_event_paranoid'
```

See our online documentation for more details:
`https://illixr.github.io/ILLIXR/getting_started`

A.3.4 Data sets

The software pulls required datasets automatically at runtime.

A.4 Installation

Not applicable.

A.5 Experiment workflow

First, we have to compile each application:

```
for app_path in OpenXR-Apps/*; do
  ./godot/bin/godot.x11.opt.tools.64
  # Import project (project.godot) (fig 1)
  # Export project (fig 2)
  # Select "Linux (Runnable)" (fig 3)
  # Select Custom Template="./godot/bin/godot.x11.opt.tools.64"
  # Select Export Path="./OpenXR-Apps/$app/bin"
  # Where app is replaced by $app shorname (e.g. "sponza")
done
```

Then, we run each application:

```
hardware=""
# manually set to one of "jetsonlp", "jetsonhp", "desktop"
for app_path in OpenXR-Apps/*; do
  app=$(basename ${app_path})
  cmd="./ILLIXR/runner.sh ILLIXR/configs/${app}.yaml"
  ${cmd}
  nvidia-smi -q --display=UTILIZATION,POWER,TEMPERATURE \
    --loop-ms=200
  perf stat -e power/energy-cores/,power/energy-pkg/,power/energy-ram/ \
    -- ${cmd}
  mv ILLIXR/metrics results/metrics/metrics-${hardware}-${app}
done
```

To switch between high- and low-power mode on Jetson

```
sudo jetson_clocks --restore ${lp_or_hp}_mode.txt
```

A.6 Evaluation and expected results

Make sure to synchronize the `results/metrics/` directory from all hardware platforms onto the desktop before continuing. On the desktop, run

```
cd results/analysis
poetry run python3 main.py
```

The output from our run is available in `metrics-snapshot` and our graphs are available in `Graphs-snapshot`.

- Fig 3 is `results/Graphs/fps-jlp/jhp/desktop.pdf`
- Fig 4 is `results/Graphs/timeseries-platformer-desktop-1.pdf` and `results/Graphs/timeseries-platformer-desktop-2.pdf`
- Fig 5 is `results/Graphs/cpu-breakdown.pdf`
- Fig 6 is `results/Graphs/power-total.pdf` and `results/Graphs/power-breakdown.pdf`
- Fig 7 is `results/Graphs/mtp-platformer.pdf`
- Fig 8 is `results/Graphs/microarchitecture.pdf`

To replicate Table IV and V, see `ILLIXR-Evaluation/README.md`. The output from our run of this script is available at `ILLIXR_SSIM_FLIP`.

A.7 Experiment customization

Here are cutomizations we anticipate:

- Modify or add your own app to `ILLIXR/app/*`.
- Add your own analysis pass to `results/analysis/*.py`
- Add your own plugin to the ILLIXR system in `ILLIXR/*`. See our online documentation for details: `https://illixr.github.io/ILLIXR/writing_your_plugin/`

A.8 Figures

Figure 1. Import a project in Godot

Figure 2. Export project in Godot

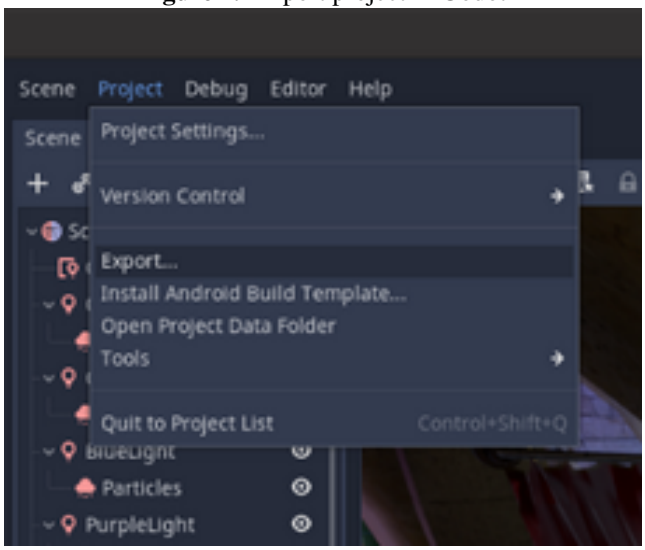

Figure 3. Select export options Godot