Tornado/SLAMBench Installation Instructions

The instructions below regard installing Tornado and SLAMBench

There are three main repositories:

1. **tornado-maven**: has the tornado API, drivers, etc.
2. **slambench-java**: has the baseline of the Java SLAMBench code
3. **slambench-tornado**: That has the Tornado-enabled version of SLAMbench-java.

We need all three repositories in order to run SLAMBench on Tornado

Steps

1. Clone all repositories
   mkdir tornado
   cd tornado
   git clone https://kotselidis@bitbucket.org/clarksoj/tornado_maven.git
   git clone https://kotselidis@bitbucket.org/clarksoj/slambench-java.git
   git clone https://kotselidis@bitbucket.org/clarksoj/slambench-tornado.git

2. Build Tornado

   cd tornado_maven

   Create **tornado.env** file in **tornado_maven/etc/** directory (Appendix I has an example of a **tornado.env** file).

   We can see that we need three **env** variables.

   The first one (JAVA_HOME) has to point to a JVMCI enabled of HotSpot Graal (if you don't have one download it and install it, following the instructions of Graal).

   The second one (JVMCI_ROOT) has to point in the $JAVA_HOME directory.

   The third one (TORNADO_ROOT) has to point in your tornado root directory.

   source etc/tornado.env
   mvn -Pgraal-env clean install

   At that point everything should build successfully.
   There is a possibility that when you checkout a branch, the **pom.xml has hard-coded paths of the env variables described above. If that is the case, and the build fails, edit pom.xml (located in **tornado_maven** root directory) and add your correct paths.**

3. Build OpenCL drivers

   cd drivers/opencl/jni-bindings/
   autoreconf –f –i –s (**there might be a case where some tools like glibtoolize is missing, you have to install them)

   ./configure --prefix=${PWD} --with-jdk=${JAVA_HOME}
   make clean
   make
   make install

   <u>At that point the drivers should have been built successfully.</u>

4. Test if Tornado has been successfully installed

   Setup the Tornado environment. (Only if it has not been setup already).
   source etc/tornado.env

   Issue **tornado tornado.drivers.opencl.OpenCL**

   The command above should print the opencl compatible devices of your system, e.g.:
   [0]: platform: Apple
   [0:0] device: Intel(R) Core(TM) i7-4850HQ CPU @ 2.30GHz
   [0:1] device: Iris Pro
   [0:2] device: GeForce GT 750M

5. After having successfully installed and tested Tornado now we have to build both **slambench-java and slambench-tornado**.

6. Building slambench-java
   cd slambench-java
   export KFUSION_ROOT="${PWD}"
   export PATH="${PATH}:${KFUSION_ROOT}/bin"
   mvn clean install -DskipTests

7. Test if Slambench Java is working
   Add **slambench-java /bin** into your path
   Issued **kfusion kfusion.java.GUI** (This should display the KFusion GUI

window)

8. After having successfully installed slambench-java now we have to install slambench-tornado

   cd slambench-tornado
   export KFUSION_ROOT="${PWD}"
   export PATH="${PATH}:${KFUSION_ROOT}/bin"
   mvn clean install –DskipTests

9. Test if Slambench Tornado is working
   Add **Slambench-tornado/bin** into your **$PATH**

   kfusion kfusion.java.GUI (This should display the Kfusion GUI window)

10. Run slambench-tornado on ICL-NUMSet

    The first thing to do is to ensure that you have downloaded a copy of the ICL_NUMSET dataset.

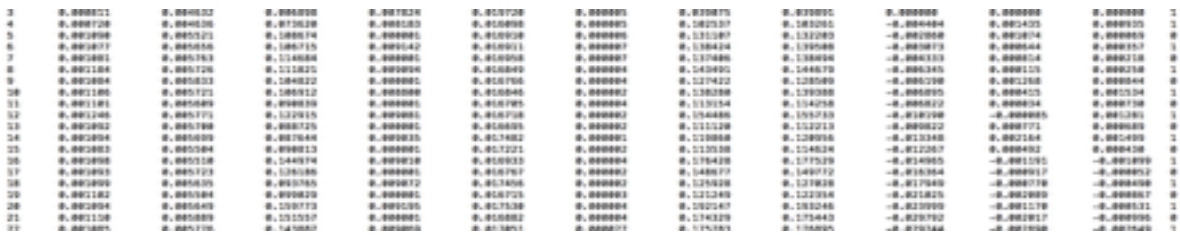
    Edit **conf/bm-traj<X>.settings** to update the path to correct raw image file. We normally run traj2 which is shorter.

    We do the same for **conf/kfusion.raw.file** in order to update the **kfusion.raw.file** variable

    After we updated the variables to point to the correct files we issue:

    kfusion kfusion.java.Benchmark conf/bm-traj2.settings

    You should get an output similar to the one below:



Every column corresponds to the metrics below:

frame	acquisition	preprocessing	tracking		integration	raycasting	rendering
	computation	total	X	Y	Z	tracked	integrated

The tracking column (second from the right) has to produce results after the 4th frame. If it does not (value set to 0) it means that the tracking algorithm does not work properly.

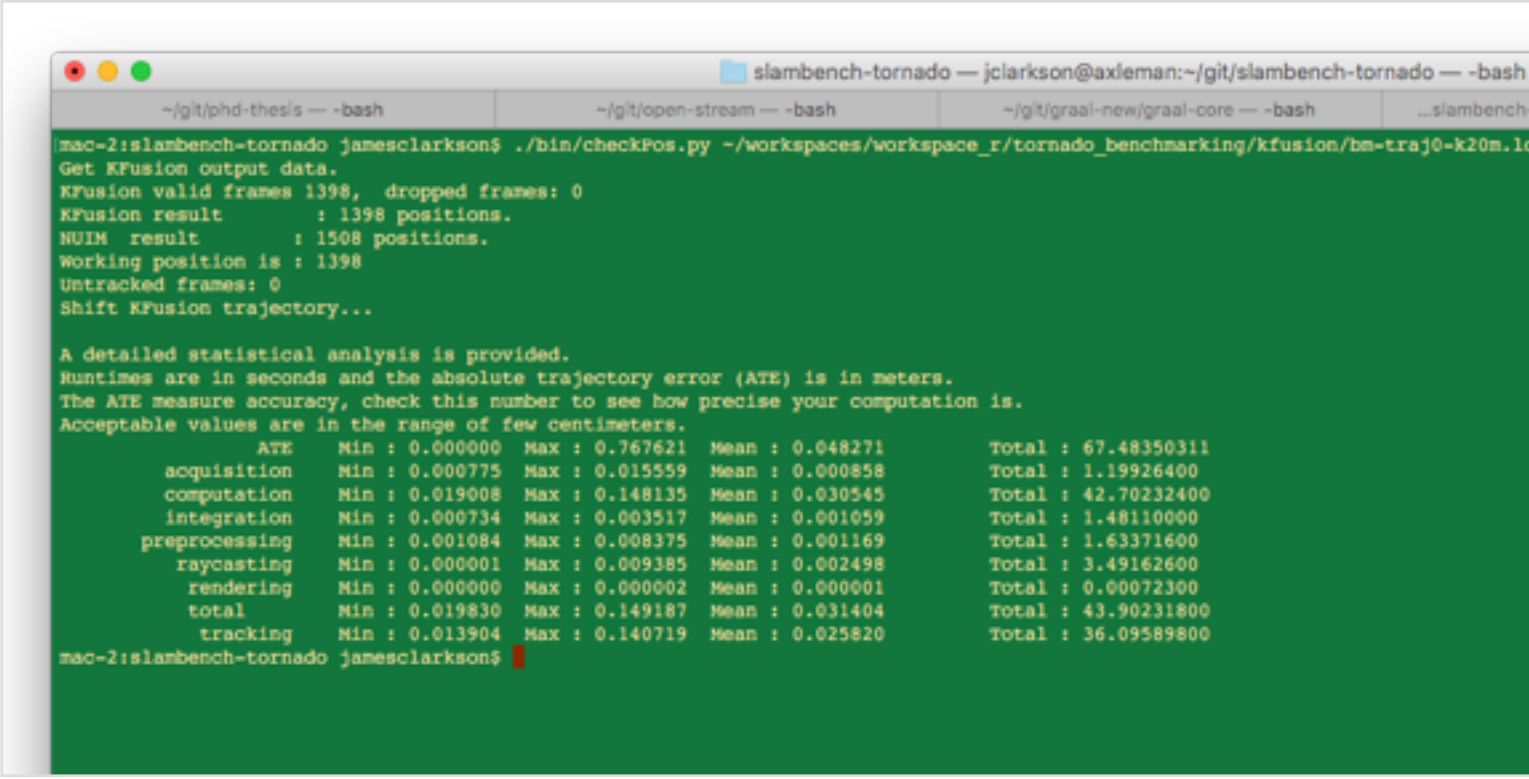
The values of: acquisition, preprocessing, tracking, integration, raycasting, and rendering are in seconds and show the time spent in each of these stages per frame.

The values of computation and total show the time spent in computation and total process of each frame.

Finally, X, Y, Z show the current camera pose.

To calculate the ATE (Absolute Trajectory Error) use the checkPos.py script.

kfusion kfusion.java.Benchmark conf/bm-traj2.settings | tee output.log
checkPos.py output.log livingRoom2.gt.freiburg



Example of tornado.env

```
#!/bin/bash



# need to be set to tornado's jvmci enabled OpenJDK build
export JAVA_HOME="/Users/kotselidis/Desktop/projects/jdk1.8.0_73-graal/"
export JVMCI_ROOT=$JAVA_HOME

if [ -z "${JAVA_HOME}" ]; then
```

```
echo "tornado: JAVA_HOME needs to be set to an JVMCI enabled OpenJDK build"
fi

export TORNADO_ROOT="/Users/kotselidis/Desktop/projects/tornado/
tornado_maven"
if [ -z "${TORNADO_ROOT}" ]; then
echo "tornado: TORNADO_ROOT needs to be set"
fi

if [ ! -z "${PATH}" ]; then
	export PATH="${PATH}:${TORNADO_ROOT}/bin"
else
	export PATH="${TORNADO_ROOT}/bin"
fi
```