USER MANUAL

Please see license information at the bottom

Authors: Cedric Cagniart (TUM),
Koen Buys (KU Leuven),
Caroline Pantofaru (Willow Garage)
Maarten De Smedt (KU Leuven)

Document conventions:

- These lines give explanation

$These lines are bash terminal commands

These lines are things needed to be changed or added inside a file

These lines are IMPORTANT

These lines indicates updates summer 2012

1. Setup

1.1 Repository checkout

- Checkout the human_tracking code from the git repos:
```
$git clone https://kforge.ros.org/sandbox/human_tracking
```
this creates a new folder in the directory you are currently in

- git branch to the rosified stack (keeps the latest changes)
```
$git checkout -b rosified origin/rosified
```
- make sure this git repo is in your ROS_PACKAGE_PATH, otherwise add it
```
$env | grep ROS
```

- checkout the human_detection code from the git repos:
```
$ git clone https://kforge.ros.org/sandbox/human_detection
```

- create the documentation for the libLabelSkel package
```
$ roscd rosdoc
$ ./rosdoc libLabelSkel -o <output_path>
```

UPDATE:
- The runtime code has moved to pcl/trunk/gpu/people
- The learning code is currently in http://kforge.ros.org/skeleton_tracking/skeleton_tracking

- The easiest way to do some of the following steps is to edit human_tracking/scripts/variables.sh and run the respective script for each of the following steps.
```
TTRAIN_ROOTDIR=$TRAINING_DIR
```
(see later on how this is set)

UPDATE:
- variables.sh is now in skeleton_tracking/code/tracking/scripts/

1.2 Hadoop Setup

These steps requires a Ubuntu Linux installation. Tested on Lucid LTS v 10.04.1, currently testing on Precise 12.04 LTS (UPDATES).

1.2.0 Install Hadoop (On the server machines, these steps are done on the KUL machines)

- install pre-requisites:

```
$ sudo apt-get install ssh rsync sun-java6-bin sun-java6-jdk sun-java6-jre
```

- Check if sshd is running (should be running):

```
$ ps -aux | grep sshd
```

- Get the hadoop software from debs (not recommended):

```
$ wget http://apache.belnet.be/hadoop/common/hadoop-1.0.2/hadoop_1.0.2-1_x86_64.deb
$ sudo dpkg -i hadoop_1.0.2-1_x86_64.deb
```

- Get hadoop from source:

```
$ wget http://apache.belnet.be/hadoop/common/hadoop-1.0.2/hadoop-1.0.2.tar.gz
$ tar xvfz hadoop-1.0.2.tar.gz
```

-----------------START IGNORE BLOCK----------------------
UPDATE:
- Currently testing with Hadoop 1.0.3:

```
$ wget http://apache.belnet.be/hadoop/common/hadoop-1.0.3/hadoop-1.0.3.tar.gz
$ tar xvfz hadoop-1.0.3.tar.gz
```

- Add the location of your recent hadoop download to your ~/.bashrc:

```
$ vim ~/.bashrc
export HADOOP_COMMON_HOME=/home/<your_username>/hadoop/hadoop-1.0.3
```

-----------------END IGNORE BLOCK----------------------

- Export the java home in conf/hadoop-env.sh:

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun-1.6.0.20
```

- On the KU Leuven Hadoop cluster this is:

```
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_04
```

- Test hadoop

```
$ cd $HADOOP_COMMON_HOME
$ bin/hadoop
```

Update for Ubuntu 12.04 install Java:
http://maketecheasier.com/install-java-runtime-in-ubuntu/2012/05/14

1.2.1 Find machines (Willow Garage Specific)

Reserve Hadoop machines here:
https://home.willowgarage.com/wgwiki/ClusterRack/ComputeCluster

1.2.2 Setup the cluster machines: (Willow Garage Specific)

- The data storage on the cluster machines is actually on drives lvdata1 and lvdata0. Cedric used lvdata0, we'll use lvdata1 for now.
- make mount directory for it

$sudo mkdir /mnt/lvdata1

- It's not mounted by default, so if the machines get rebooted, do the following to mount the data drives:
$sudo mount /dev/vgdata/lvdata1 /mnt/lvdata1

- If necessary (if the mount fails, it should autodetect the formatting), format by (create ext4 filesystem on lvdata1 HD):
$sudo mkfs.ext4 /dev/vgdata/lvdata1

- if expected to reboot machines often add an /etc/fstab entry
- create the two needed directories
$sudo mkdir /mnt/lvdata1/hadoopfs
$sudo mkdir /mnt/lvdata1/hadooptmp

- adjust the owner and the privileges
$sudo chown $USER /mnt/lvdata1/hadooptmp/
$sudo chown $USER /mnt/lvdata1/hadoopfs/
$sudo chgrp wgusers /mnt/lvdata1/hadooptmp/
$sudo chgrp wgusers /mnt/lvdata1/hadoopfs/

- check the settings:
$ ls -al /mnt/lvdata1

- should say something like:
drwxr-xr-x 2 kbuys wgusers 4.0K 2011-12-02 13:36 hadoopfs/
drwxr-xr-x 2 kbuys wgusers 4.0K 2011-12-02 13:36 hadooptmp/

1.2.3 Setup your own machine (the namenode): (WG Specific)

- Download hadoop-0.21.0 into ~/code/ (or any other NFS drive location) or copy it from /u/kbuys/code (this one has the WG specific configuration already in it, and makes it easier to adjust it correctly)
!!!REMEMBER to install hadoop-0.21.0 in your NFS dir so that all machines can accept it !!!
$wget http://apache.cs.utah.edu//hadoop/common/hadoop-0.21.0/hadoop-0.21.0.tar.gz
$tar xzf hadoop-0.21.0.tar.gz
$cd hadoop-0.21.0
$export HADOOP_COMMON_HOME=$PWD
$echo "export HADOOP_COMMON_HOME=$PWD" >> ~/.bashrc

UPDATE: from now on we try with Hadoop 1.0.3:

- Currently testing with Hadoop 1.0.3:
$ wget http://apache.belnet.be/hadoop/common/hadoop-1.0.3/hadoop-1.0.3.tar.gz

```
$ tar xvfz hadoop-1.0.3.tar.gz
```

- go to the HadoopTree

```
$roscd HadoopTree
```

- build HadoopTree (this should get the Apache Ant dependency)

```
$rosmake --rosdep-install
```

- Change lib dir in human_tracking/HadoopTree/build.xml to your hadoop-0.21.0 installation path
- build HadoopTree

```
$./build.sh
```

- This should have created the HadoopTree/lib/HadoopTree.jar file

You need to point the jarFile variable in human_tracking/scripts/variables.sh to this jar file.

- Create a hdfs-namenode folder on your local disk (not NFS!)

```
$mkdir /home/$USER/hdfs-namenode
```

- Create your local hadoopfs and hadooptmp folders (lvdata1 is to match location on cluster)

```
$sudo mkdir /mnt/lvdata1
$sudo mkdir /mnt/lvdata1/hadoopfs
$sudo mkdir /home/$USER/hadoop
$sudo mkdir /home/$USER/hadoop/hadoopTmp
```

- Create the symbolic link from mount to your local folder

```
$cd /mnt/lvdata1
$sudo ln -s /home/$USER/hadoop/hadoopTmp hadooptmp
```

- Create the logs folder

```
$sudo mkdir /home/$USER/hadoop/hadoopTmp/logs
```

- Change the ownership of the folders in mount

```
$cd /mnt/lvdata1
$sudo chown -R $USER *
$sudo chgrp -R wgusers *
$cd /home/$USER/hadoop
$sudo chown -R $USER *
$sudo chgrp -R wgusers *
```

- go to your hadoop-0.21.0 installation folder

```
$cd $HADOOP_COMMON_HOME
```

- edit conf/hdfs-site.xml, change the namenode dfs directory and data directory, replace kbuys by your username (`$echo $USER`)

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<!-- path on the local file systen where the datanode instance stores its stuff -->
  <property>
    <name>dfs.data.dir</name>
    <value>file:/mnt/lvdata1/hadoopfs</value>
  </property>
<!-- amount of replication in the system-->
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
<!-- block size -->
  <property>
    <name>dfs.blocksize</name>
    <value>67108864</value>
  </property>
  <property>
    <name>dfs.datanode.socket.write.timeout</name>
    <value>0</value>
  </property>
</configuration>
```

- edit conf/core-site.xml, change the hdfs default name, with hdfs://<your hostname>:9000 and hostname (`$echo $HOSTNAME`), change the hadoop.tmp.dir also

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<!-- the namenode guy-->
<property>
  <name>fs.default.name</name>
  <value>hdfs://bgq:9000</value>
</property>
<!--for some reason when i set this to a real dir the jobtracker crashes....-->
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/mnt/lvdata1/hadooptmp</value>
  </property>
</configuration>
```

- edit conf/mapred-site.xml, change the hostname in the mapred.job.tracker with your hostname

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<!-- the job tracker guy-->
<property>
  <name>mapred.job.tracker</name>
  <value>bgq:1234</value>
</property>
<property>
  <name>mapred.reduce.child.java.opts</name>
  <value>-Xmx2048m</value>
</property>
<property>
  <name>mapreduce.tasktracker.map.tasks.maximum</name>
  <value>4</value>
</property>
<property>
  <name>mapreduce.tasktracker.reduce.tasks.maximum</name>
  <value>6</value>
</property>
</configuration>
```

- Add the reserved cluster machines to conf/slaves
- Change conf/masters to your machine
- Make sure you have sun java6 and sun java6 jre installed on your machine

```
$sudo aptitude search sun-java
```

- These should mark i for installed, otherwise install them

```
$sudo aptitude install sun-java6-bin sun-java6-jdk sun-java6-jre
```

- Change hadoop-env.sh to:

```
export HADOOP_LOG_DIR=/mnt/lvdata1/hadooptmp/logs
export JAVA_HOME=/usr/lib/jvm/java-6-sun
export HADOOP_SLAVE_SLEEP=0.1
```

- generate ssh keys for auto login on all cluster machines

```
$roscd human_tracking
$scripts/setupSlaves.sh $HADOOP_COMMON_HOME/conf/slaves
```

Note: The setupSlaves.sh script contains the following commands:

```
#!/bin/bash
while read line; do
```

```
  echo $USER@$line
  ssh-copy-id $USER@$line
done < $1
```

- format the namenode:

```
$ $HADOOP_COMMON_HOME/bin/hadoop namenode -format
```

WARNING: only do this once as this will delete all data on the HDFS

- start the hadoop file system

```
$ $HADOOP_COMMON_HOME/bin/start-dfs.sh
```

- start the mapreduce

```
$ $HADOOP_COMMON_HOME/bin/start-mapred.sh
```

- The cluster is now functional

- You can now check with firefox if the cluster is working
    - NAMENODE:50070 is the web interface for HDFS
    - JOBTRACKER:50030 is the web interface for mapreduce

1.2.4 Setup your own machine: (KUL Specific)

At KUL the namenode runs on pma-robot-cutting and the second namenode runs on pma-robot-gosling. These are preconfigured for you. Pma-robot-cuda2 is dev. machine and is added as a datanode and tasktracker.
The Namenode and jobtracker can be found by entering this into your firefox:
pma-robot-cutting:50070 for the namenode
pma-robot-cutting:50030 for the tasktracker

- If this doesn't show any webpage you can start the machines by logging in into pma-robot-cutting as the hadoop user

```
$ ssh hadoop@pma-robot-cutting
```

- And run the dfs and mapred scripts

```
$ cd $HADOOP_COMMON_HOME/bin
$ ./start-dfs.sh
$ ./start-mapred.sh
$ exit
```

- Get Hadoop

```
$wget http://apache.cs.utah.edu//hadoop/common/hadoop-0.21.0/hadoop-0.21.0.tar.gz
$tar xzf hadoop-0.21.0.tar.gz
```

```
$cd hadoop-0.21.0
$export HADOOP_COMMON_HOME=$PWD
$echo “export HADOOP_COMMON_HOME=$PWD” >> ~/.bashrc
```

- add the path to your Java installation to your .bashrc

```
$ vim ~/.bashrc
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_04
```

- go to the HadoopTree

```
$roscd HadoopTree
```

- build HadoopTree (this should get the Apache Ant dependency)

```
$rosmake
```

this command created the build.sh file in the current directory

- Change lib dir in human_tracking/HadoopTree/build.xml to your hadoop-0.21.0 installation path

- build HadoopTree

```
$./build.sh
```

- This should have created the HadoopTree/lib/HadoopTree.jar file
You need to point the jarFile variable in human_tracking/scripts/variables.sh to this jar file.

UPDATE:
- variables.sh is now in skeleton_tracking/code/tracking/scripts/

1.3 Get the dataset

- Create a folder for the dataset on to your local disk (not NFS!)

```
$mkdir /home/$USER/data
$cd /home/$USER/data
$mkdir cmumocap
$cd cmumocap
$export CMUMOCAP_DIR=$PWD
$echo “export CMUMOCAP_DIR=$PWD” >> ~/.bashrc
```

- download the CMUMOCAP data in CMUMOCAP_DIR, these files come from
https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/cmu-bvh-conversion
The motionbuilder version 2 files of 2010 release

- Create a folder for the training set

```
$cd /home/$USER/data
$mkdir trainingsets
$cd trainingsets
$mkdir test_`date +%F`        (notice the space between ‘date’ and the + sign)
EdisonElectricLight
$cd test_`date +%F`
$export TRAINING_DIR=$PWD
$echo “export TRAINING_DIR=$PWD” >> ~/.bashrc
$mkdir mesh_files
$mkdir reffile
$mkdir lmaps; mkdir cmaps; mkdir dmaps; mkdir tree_files
```

1.4 Get Blender

- download blender from http://www.blender.org to a suitable location on your disk

```
$wget http://download.blender.org/release/Blender2.60/blender-2.60-linux-glibc27-x86_64.tar.bz2
$tar xjf blender-2.60-linux-glibc27-x86_64.tar.bz2
$cd blender-2.60-linux-glibc27-x86_64
$export BLENDER_DIR=$PWD
$echo “export BLENDER_DIR=$PWD” >> ~/.bashrc
```

1.5 Get MakeHuman

- download MakeHuman from http://www.makehuman.org/ to a suitable location on your disk:

```
$ wget http://makehuman.googlecode.com/files/makehuman1.0.0-alpha6-linux-amd64.tbz2
```

- Or get it through apt with the following location in your sources.list:

```
deb http://download.tuxfamily.org/makehuman/deb/amd64/ /
```

- MakeHuman will create a makehuman dir in your home folder to which all files are exported

1.6 Get PCL

- For this you will need to check out perception_pcl_unstable in your ROS_PACKAGE_PATH, overlayed over your current pcl installation (_DON’T_ build PCL just yet):

```
$svn co http://svn.pointclouds.org/ros/trunk/perception_pcl_electric_unstable
```

- We are working with the trunk version (on your own risk, tested until r3427), so comment the line in your Makefile that claims a certain revision number, or go back to ours

(
This shouldn’t be necessary anymore with PCL 1.6:
- You will also need to install libflann, the easiest way is:

```
$sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl
```

```
$sudo apt-get update
$sudo apt-get install libflann-dev libflann1
)
```

- Now you should be able to build PCL

```
$roscd pcl
$rosmake
```

- Change your manifest to include the correct pcl-1.x version, you can check this with:

```
$roscd pcl
$ls include/
```

1.7 Eclipse

1.7.1 Install Eclipse

- Check wether your machine is 32 bit or 64 bit

```
$uname -m
```

⇒ i686 for 32 bit
⇒ x86_64 for 64 bit

- Download the correct version of eclipse from http://www.eclipse.org/downloads/
⇒ unpack (e.g. to your home folder) the file, Eclipse is then installed

1.7.2 Make alias for Eclipse

First make an alias to open eclipse with a simple command:
-locate Eclipse

```
$ locate *eclipse
```

if you installed Eclipse in your home folder, one of the lines should look like this:
/home/$USER/eclipse/eclipse
(with $USER your username, e.g. s0123456 → try $echo $USER)

- create an alias as a shortcut and run the .bashrc script to apply the changes

```
$ echo "alias eclipse=/home/$USER/eclipse/eclipse" >> .bashrc
$ source ~/.bashrc
```

- Eclipse should now open by the command

```
$ eclipse&
```

1.7.3 Create the *skeleton_tracking* project in Eclipse

- open Eclipse and choose the correct workspace folder

```
$ eclipse&
```

a prompt shows up to choose the workspace
browse to the folder that contains the /skeleton_tracking folder (that you checked out from the git repository, see 1.1 Repository checkout)
for example: if you checked out in your home folder, you should enter */home/s0123456/* as workspace
Note: it is important to choose the correct workspace to have your java code synchronized with the git repository.

- create a new Java Project: File->New->Java Project

- enter *skeleton_tracking* as Project name, click Next and you should see that the existing files and directories are included in the project

Some errors are detected by Eclipse, these are solved in the next section

1.7.4 Configure Build Path of ***skeleton_tracking***

The errors mention that some classes cannot be resolved to a type, this is because the hadoop libraries aren't attached yet.

- import the *hadoop-0.21.0.tar.gz* file:
in the package explorer: right click on the project *skeleton_tracking*, then click Import
in the General tab click *Archive File*, click next and browse for the *hadoop-0.21.0.tar.gz* file (that you downloaded in 1.2.3 Setup your own machine (the namenode): (WG Specific) or 1.2.4 Setup your own machine: (KUL Specific)) and click Finish

In the project there should now be a new folder *hadoop-0.21.0*. This contains several jar-files of which some will be needed in the *skeleton_tracking* project. The folder is in the directory that is synchronized with the git repository, but it doesn't need to be tracked by git.

-in the package explorer: right click on the project *skeleton_tracking* then click Build Path->Configure Build Path

-in the Libraries tab click Add JARs and select the following jar-files in the folder *hadoop-0.21.0*:

1. hadoop-common-0.21.0.jar
2. hadoop-mapred-0.21.0.jar
3. hadoop-hdfs-0.21.0.jar

and in the folder *hadoop-0.21.0/lib*:

4. commons.cli-1.2.jar

then click OK and another time OK.

The errors should now be resolved.

1.7.5 Import javadoc in Eclipse

-in the package explorer: right click on *hadoop-common-0.21.0.jar* (in *Referenced libraries*) then choose Properties
-in the tab Javadoc Location: click Java URL and browse for
$HADOOP_COMMON_HOME/hadoop-0.21.0/common/docs/api
-click then OK and the javadoc should now show up when moving the mouse over a class (e.g. LongWritable) or a method of a class in the hadoop common package.

Repeat the same for *hadoop-mapred-0.21.0.jar* and *hadoop-hdfs-0.21.0.jar*

2 DataSynth

2.1 Create mesh files

- Create a mesh using makehuman (www.makehuman.org, used version 1.0 Alpha 6 for this UM) export your mesh as a make human exchange file (.mhx) and as a Collada file (.dae)

```
$makehuman
```

- import the mesh in blender (This UM has been tested with Blender 2.57b and 2.59) using the import scripts provided by makehuman or by the Collada import from Blender

```
$cd $BLENDER_DIR
$./blender
```

- remove the clothes, armature and useless mesh pieces such as jaw or eyeballs
- import myreffile.bvh in Blender. This contains a skeleton and an animation. We don't care about the animation.
- in edit mode for the armature, roughly align the joints of the skeleton with the makehuman mesh.
- select mesh, select armature also, and shift-P or ctrl-P to make the armature the parent of the mesh. Select auto weights. This will fail because normals are important and fingers have crazy normals with respect to the forearm bone. If the mesh is simpler (rounded extremities, no holes in eyes or mouth), this usually works nicely.
.
- in edit mode for the mesh manually select each vertex group, and assign it to a vertex group that has the same name as the bone. Once an armature modifier is applied to the mesh, the rig will behave correctly. (you can check by moving the skeleton around in pose mode)
- The same idea allows to create vertex groups that are labelGroups. I usually do this on a duplicate of the original mesh so I don't have the vertex groups mixed.

-Now you can export the *.verts *.tris *.bones *.label files using the provided scripts. (alt-P while focusing on the text editor inside blender executes the script)

- Copy the mesh files to mesh_files in your training dir

```
$mv mesh.* $TRAINING_DIR/mesh_files/
```

2.2 Generate a BVH List File

- To generate this file you can build a findFiles.sh script or use the premade one in human_tracking

```
#!/bin/bash
find $1 | grep bvh > $2
```

- To use this script $1 is the location of the BVH files, $2 is the output file

```
$./findFiles $CMUMOCAP_DIR $TRAINING_DIR/BVHList.txt
```

2.3 Get the reference file and config files

- This file is the reference for all the other BVH files and is common for the CMUMOCAP dataset

```
$ cp <your git repo location>/human_tracking/scripts/reffile.bvh $TRAINING_DIR/reffile/
$ cp -R <your git repo location>/human_tracking/scripts/configFils $TRAINING_DIR
```

2.4 Sparsify BVH database

- We are only interested in using the skeleton from the ref bvh file.
- The bvhFilelist has on each line the name of a BVH file whose joint values we will read. All these BVH files must have same skel topology as the refFile.

```
$ roscd BVHDBSparsify/
$ rosmake       (this should have created a /bin folder)
$ roscd BVHDBSparsify/bin
$ ./BVHDBSparsify \
        -BVH_ref        $TRAINING_DIR/reffile/reffile.bvh \
         -BVH_list      $TRAINING_DIR/BVHList.txt \
        -DB             $TRAINING_DIR/DB.txt
```

- the DB.txt will contain the value vectors copied from the bvhFiles from the bvhFileList, but with the duplicate value vectors removed.
- for this you can also use the runBVHDBSparsify.sh script in scripts after correctly setting variables.sh
- after running this you must set the last frame in variables.sh to match the number of found vectors

NOTE: *this step takes hours to run*

Update:
There is now an openmp version available for the BVHDBSparsify step: BVHDBSparsify_omp and runBVHDBSparsify_omp.sh in scripts

Tim Beyl:

On Amd Phenom II X6 1090T (4GB of RAM) top indicated a runtime of approximately 70 hours when using the OMP version. However it took in between 12-14hours to run

2.5 Convert BVH files to RGBD files

If not generated in Section 2.1: Ask Koen for this 4 files: *mesh.bones, mesh.verts, mesh.labels, mesh.tris*
(these can maybe be added to the git repository??)

Note: This cannot run over ssh.

```
$ roscd BVHtoRGBD
$ rosmake       (this should have created a /bin folder)
$ BVHtoRGBD \
        -MMat         MMat.txt \
        -BVH_ref      myreffile.bvh \
        -MotionFile   DB.txt \
        -verts        mesh.verts \
        -tris         mesh.tris \
        -bones        mesh.bones\
        -labels       mesh.labels \
        -dout         d_%05d.png \
        -lout         l_%05d.png \
        -bgdepth_mm   someIntValue \
        -F            0 \
        -L            65000 \
        -angleRange   someFloatValue in radians
```

- MMat is the opengl modelview matrix (it has to be stored in its transposed (openGL) form, that means that the last column is [0 0 0 1] and that the last line contains the translation parameters).
- BVH_ref, MotionFile : the ref file and the output from BVHDBSparsify
- mesh.verts, mesh.tris, mesh.bones, mesh.labels : output of the blender scripts
- For this you need to look at the ./runBVHtoRGBD script in scripts

2.6 Convert to color images

- run the ./runColor.sh script with as argument the number of parallel processes your system allows:

```
$./runColor.sh 8
```

Work with Mihai to combine to execute our training data generation in Gazebo

- Import CMU BVH into blender 2.62.
- Import mesh from the MakeHuman collada file into blender 2.62.
- Align the bvh skeleton with the MakeHuman mesh. First align the origin (center of hips) and work out from there.
- A lot of hand fitting to get the skeleton and the mesh to align. This doesn't seem reasonable, it's too much guessing. Koen: We should consider mapping bones instead of hand aligning process.
- Export joint locations as well next to depth and label image!

3 Trees

3.1 Create config files (tree.txt?)

```
$ roscd CreateConfigFiles
$ rosmake
$ CreateConfigFiles \
        -numThreshs 50 \
        -maxThreshAbs 100 \
        -maxRadius 100 \
        -oAttribLoc attribLoc.txt \
        -oThreshs threshs.txt
```

- numThreshs : the number of random thresholds to generate
- maxThreshAbs : the range of threshold (remember that everything is in mm so that 100 = 10cm)
- maxRadius: the maximum distance from the classified pixel at which pixel queries are made. This is given for a distance = 1m of the camera. Meaning that if the value is 100 and the person is 2m away from the camera, the query will be at most 50pix away.

There are two files. one contains the threshold values (there are 50 evaluated thresholds) and the other contains the Attribute Locations.. which are (du1, dv1, du2,dv2) tuples indicating where to query in the image relative to the location of the classified pixels

3.2 Create samples, (feature descriptors?)

```
$ roscd CreateSamples
$ rosmake
$ createSamples \
        -attribLoc attribLoc.txt \
        -threshFile threshs.txt \
        -lmapBasename d_%05d.png \
        -dmapBasename l_%05d.png \
        -F 0 \
        -L 65000 \
        -o splitBuffs.dat
```

-> they are files (that i call splitBuffs i think) containing split points for each dimension of the feature descriptors (the index of the first threshold whose value is > the attribute value for this dimension of the feature vector)
- For this there is a runCreateSamples.sh in scripts (TODO: search for errors)

3.3 Train the tree

3.3.1 Commands before the summer of 2012

NOTE: *the following commands are outdated for the changes made in the code during the summer of 2012, so check which version of the Java code you are using.*

- a combination of the following steps can be found in the kbuys_run.sh script in HADOOP_COMMON_HOME/myapps

```
$./kbuys_run.sh 1
```

Where $1 is a suffix to identify the database or a identifier name

-upload the attribThreshFile (from the $TTRAIN_ROOTDIR/configFiles/ folder)

```
$bin/hadoop fs -put attribThresh.txt attribThresh.txt
```

- upload the big file containing the splitBuffs to the DFS.. this will convert the binary file to a hadoop .seq file which accounts for record (feature vecs) boundaries when splitting the file over the network.

```
$bin/hadoop jar HadoopTree.jar \      ?? $bin/hadoop jar HadoopTree.jar HadoopTree.apps.Driver \
        upload_sp \
        <localpath>/splitBuffs.dat \
        <dfspath>/splitBuffs.seq       ?? the java code asks for splitbuffs.seq   (with small b i.o. B)
```

- launch the training

```
$bin/hadoop jar HadoopTree.jar \
        trainTree \
        <dfspath> \
        <maxDepth> \
        <localoutpath>
```

- the result is a <localoutpath>/tree.txt file containg the (du1, dv1, du2,dv2, thresh) tuples for inside nodes, and the (labels) for all the leaves of the tree.

3.3.2 Commands after the summer of 2012

These are the commands that need to be used when working with the version of the Java code after the summer of 2012. Some changes were made to remove hard coded paths and some minor bugs were removed. And the name of the mentioned node is from the cluster at KUL.

-make a local directory on the name node (pma-robot-cutting)

```
$ssh hadoop@pma-robot-cutting
$mkdir localfiles
$exit
```

-build the Java code

```
$roscd HadoopTree
$./build.sh
```

this created the file *HadoopTree.jar* in the *lib* directory and this file needs to be copied to the name node's folder *localfiles*

```
$scp lib/HadoopTree.jar hadoop@pma-robot-cutting:localfiles
```

-then copy the *attribLoc.txt* and the *attribThresh.txt* files to the *localfiles* folder

```
$scp $TTRAIN_ROOTDIR/configfiles/attribThresh.txt \
                                        $TTRAIN_ROOTDIR/configfiles/attribLoc.txt \
                                        hadoop@pma-robot-cutting:localfiles
```

-then copy the *splitBuffs.dat* file to the *localfiles* folder on the namenode

```
$scp <path>/splitBuffs.dat hadoop@pma-robot-cutting:/data2/<create your directrory>
```

make sure you copy that (large) file on the local hard drive of the name node

-log in into the namenode and these two file on the hadoop file system

```
$ssh hadoop@pma-robot-cutting
$cd localfiles
$bin/hadoop fs -put attribThresh.txt attribThresh.txt
```

This last command puts the that file on the Hadoop file system because the jobs running on the cluster will make use of that file.

-make sure you are still working in the *localfiles* directory and upload the next file

```
$time bin/hadoop jar HadoopTree.jar HadoopTree.apps.Driver upload_sp /data2/<your
directory>/splitbuffs.dat  <your dircetory2>/splitbuffs.seq
```

<your dircetory2> is a directory on the hadoop file system that you call whatever you want, but make sure *splitbuffs.seq* is correctly spelled since the Java code depends on this name. This may take hours since this file is probably very large.

-then start the actual training (still from the *localfiles* directory)

```
$ time bin/hadoop jar HadoopTree.jar HadoopTree.apps.Driver trainTree   <your dircetory2>   20
<your result directory>
```

3.4 Willow garage backups

- Because backups take in a large amount of diskspace, and you don't want to keep them on your own disk, we've been backing up data here:

```
$cd /wg/wgss0_shelf2/people_tree_training/
```

- You can find the old back up still alive here:

```
$cd /wg/wgss0_shelf1/cedric_treetraining/
```

4 Running

This can now be run as an independent program or as a ROS node
UPDATE: latest version is now beeing developped in PCL under gpu/people/tools

4.1 Independent run

```
$roscd TLive
$rosmake
$ bin/TLive -numTrees 3 \
        -tree0 ~/data/results/forest1/tree_20.txt \
        -tree1 ~/data/results/forest2/tree_20.txt \
        -tree2 ~/data/results/forest3/tree_20.txt \
        -tree3 ~/data/results/forest3/tree_20.txt \
        -mask 0 -FG 0
```

4.2 ROS node implementation

(
Shouldn't need to do this anymore:
For electric, overlay openni_ros (the released version has a bug)
)

```
$ roscore
$ roscd TLive_pcl
$roslaunch launch/run_local_rviz.launch
```

```
$roscd TLive_pcl
$rosmake
$./run.sh
```

Visualization messages

in images one is the labeled image after blobs, one is the seed image points
one is the flower (after seeded hue seg)
and the last one is the labeled2 image, labels in second iteration
you also have a markerarray with the kinematic chain
for each image you also have the corresponding pointcloud

4.3 Running under PCL trunk

4.3.1 Installation for run-time use only

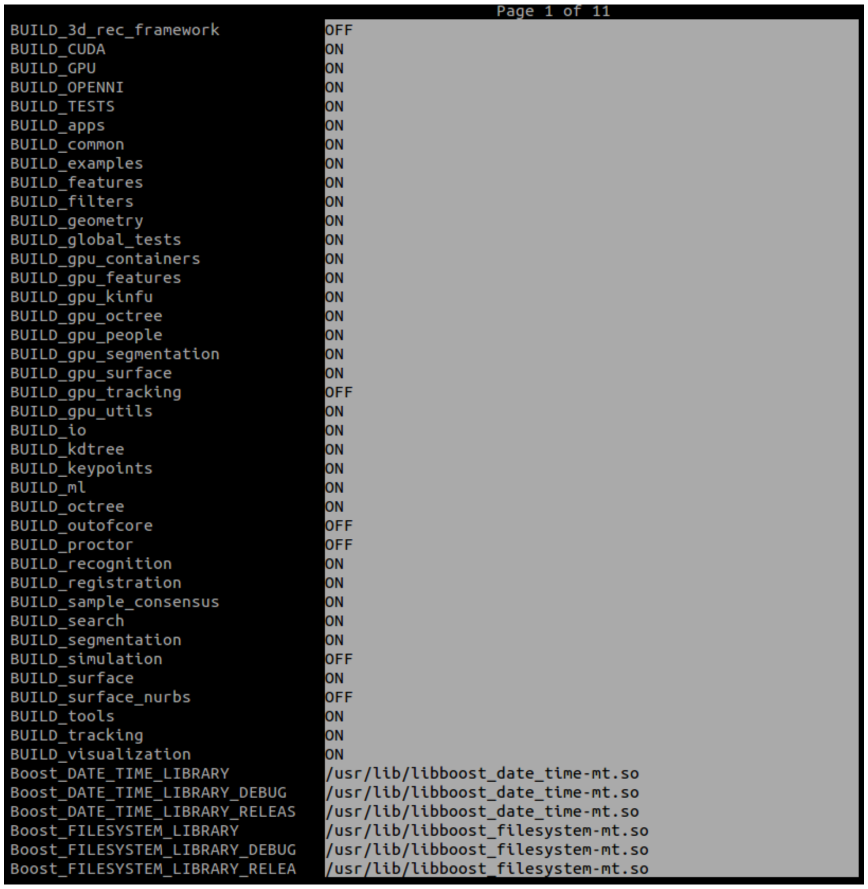
- My current system configuration (for reference):
  - Ubuntu 11.10 Oneiric - x86_64
  - Quadro FX580 for display
  - Tesla C2050 for GPGPU
  - PCL trunk r5705
  - Intel Xeon E5630 CPU
  - Cuda toolkit 4.017
  - lib cuda version 280.13

- Install latest nvidia device drivers for your GPU from here:
http://developer.nvidia.com/cuda-downloads

- Currently this is version 295.41
- Download and install the CUDA Toolkit from the same website, version for Ubuntu 11.04 installs fine on Ubuntu 11.10. Latest version is 4.2

- Install PCL from trunk, up till now tested fully functional up to revision r5705. I will update this revision information in the future

- In ccmake toggle the advanced mode on and build with the following settings:



```
                                                Page 1 of 11
BUILD_3d_rec_framework           OFF
BUILD_CUDA                       ON
BUILD_GPU                        ON
BUILD_OPENNI                     ON
BUILD_TESTS                      ON
BUILD_apps                       ON
BUILD_common                     ON
BUILD_examples                   ON
BUILD_features                   ON
BUILD_filters                    ON
BUILD_geometry                   ON
BUILD_global_tests               ON
BUILD_gpu_containers             ON
BUILD_gpu_features               ON
BUILD_gpu_kinfu                  ON
BUILD_gpu_octree                 ON
BUILD_gpu_people                 ON
BUILD_gpu_segmentation           ON
BUILD_gpu_surface                ON
BUILD_gpu_tracking               OFF
BUILD_gpu_utils                  ON
BUILD_io                         ON
BUILD_kdtree                     ON
BUILD_keypoints                  ON
BUILD_ml                         ON
BUILD_octree                     ON
BUILD_outofcore                  OFF
BUILD_proctor                    OFF
BUILD_recognition                ON
BUILD_registration               ON
BUILD_sample_consensus           ON
BUILD_search                     ON
BUILD_segmentation               ON
BUILD_simulation                 OFF
BUILD_surface                    ON
BUILD_surface_nurbs              OFF
BUILD_tools                      ON
BUILD_tracking                   ON
BUILD_visualization              ON
Boost_DATE_TIME_LIBRARY          /usr/lib/libboost_date_time-mt.so
Boost_DATE_TIME_LIBRARY_DEBUG    /usr/lib/libboost_date_time-mt.so
Boost_DATE_TIME_LIBRARY_RELEAS   /usr/lib/libboost_date_time-mt.so
Boost_FILESYSTEM_LIBRARY         /usr/lib/libboost_filesystem-mt.so
Boost_FILESYSTEM_LIBRARY_DEBUG   /usr/lib/libboost_filesystem-mt.so
Boost_FILESYSTEM_LIBRARY_RELEA   /usr/lib/libboost_filesystem-mt.so
```

- Install the tree files into ~/Data/results/ (update they are now in the data folder in people)
- Bash shell scripts are provided in pcl/trunk/gpu/people/tools, change them if you put the RDF tree files in a different location

5 File Formats

5.1 labelNames

5.2 fileformats

6 Links

6.1 Dependencies

This software depends on:
- Hadoop
- PCL (the perception_pcl_unstable adjusted to follow trunk)
- Apache Ant
- OpenCV
- Boost
- Cuda Toolkit 4 and SM2.1 compatible Nvidia GPU
- Sun Java and JRE

6.2 Databases

- https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture

BVH conversions of the 2500-motion Carnegie-Mellon motion capture dataset:

- http://kitchen.cs.cmu.edu/index.php

The CMU Multi-Modal Activity Database (CMU-MMAC) database contains multimodal measures of the human activity of subjects performing the tasks involved in cooking and food preparation.
The CMU-MMAC database was collected in Carnegie Mellon's Motion Capture Lab.
A kitchen was built and to date twenty-five subjects have been recorded cooking five different recipes: brownies, pizza, sandwich, salad, and scrambled eggs.

- http://mocap.cs.cmu.edu/
  the Carnegie Mellon University Motion Capture Database
- http://accad.osu.edu/research/mocap/mocap_data.htm
- http://www.mocapdata.com/ (commercial)
- http://www.motekentertainment.com/index.php (commercial)
- http://www.mocapclub.com/Pages/Library.htm
- http://www.centralsource.com/blender/bvh/files.htm
- http://freemotionfiles.blogspot.com/

6.3 Useful software

- BVHPlay
  https://sites.google.com/a/cgspeed.com/cgspeed/bvhplay
  a free, lightweight, Python-based, multi-OS, open-source playback utility for BVH animation files
- http://brekel.com/
  Brings kinect data into motionbuilder
- http://www.vipbase.net/amc2bvh/

Amc2Bvh provides batch conversion from ASF/AMC to BVH
- http://davedub.co.uk/bvhacker/
  The free bvh file editing tool

7 Pictorial structures

7.1 Links

http://www.ics.uci.edu/~dramanan/

7.2 Process

Downloaded http://phoenix.ics.uci.edu/software/pose/ release 1.2
Extracted
cd pose-basic
matlab
compile
demo
WORKS! > and now in C code

8 License

9 TODO

Change libBVH to Eigen3
http://eigen.tuxfamily.org/dox/Eigen2ToEigen3.html

a Hadoop

jarfile=/xx/xxxx/xxx/HadoopTree.jar
splitBuffFile=/home/cedric/data/db_$suffix/splitbuffs.dat

# this uploads the huge data file to hdfs while making sure that the split boundaries do not split individual records.
bin/hadoop jar $jarfile HadoopTree.apps.Driver \
upload_sp splitbuffs.dat $rootDir/splitbuffs.seq

# this first uploads the config file with the threshold values
bin/hadoop fs -put /xxx/xxx/attribThresh.txt attribThresh.txt
# this launches the tree building.. outDir is a local path on the client where the result will be written
bin/hadoop jar $jarfile HadoopTree.apps.Driver \
trainTree $rootDir $depth $outDir

b Steps from Cedric's usermanual

- generate the image data

```
$bin/BVHKinect -MMat MMat.txt \
         -BVH_ref 63_01.bvh \
         -MotionFile framedb.txt.txt \
         -verts KINECT/data/mesh.verts \
         -tris KINECT/data/mesh.tris \
         -bones  KINECT/data/mesh.bones \
         -labels KINECT/data/mesh.labels \
         -dout dmap/d_%06d.png \
         -lout lmap/l_%06d.png
```

- generate the feature data

```
$bin/TTrain_createSamples -attribLoc $attribLocFile \
                  -lmapBasename $lmapBasename \
                  -dmapBasename $dmapBasename \
                  -F $firstFrame -L $lastFrame \
                  -o $featureFile \
```

- train