General
time sync
ROS
Calibration
Kinect
Hybrid Kinect
Mesa imager
Recording data for replay
Lighting & detection discussion

General

There are many sub-processes per running instance. Ctrl-C breaks the detection loop, but does not kill all subprocesses.

to do so, enter the following command - the system will behave cleanly / as if it's been rebooted.

pkill -f ros; pkill -f nodelet; pkill -f XnSensor

time sync

it's critical that all computers in the network are time-synchronised. Set up the a local machine (typically the master, though any should work) to broadcast it's time over LAN subnet, and have the other hosts *only* have the master on the NTP server list. Existing documentation online is somehow confusing and out of date, and the default files in ubuntu are very noisy / have a lot of stuff we don't need. Thus, the below:

server /etc/ntp.conf

or for a'standard' 192.168.100.X network:

server time1.ucla.edu iburst
server 127.127.1.0
fudge 127.127.1.0 stratum 10
… *(skip ahead in file until you see 'restrict' - make sure following is only uncommented)*
restrict 192.168.100.0 mask 255.255.255.0 nomodify notrap
…. *(skip ahead in file until you see 'restrict' - make sure following is only uncommented)*
broadcast 192.168.100.255

if you *just* want things to sync to master, and the master is otherwise disconnected from internet, get rid of the first external server.

note broadcast settings are different per lan… don't just copy the above IPs (or ntp server for that matter :)

client /etc/ntp.conf :

(note there is a lot of other stuff - get rid of it, on the client. some settings interfere w/ the ultimately very simple goal we have, of syncing the slave/client to a master on the lan)

```
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help
driftfile /var/lib/ntp/ntp.drift
# Enable this if you want statistics to be logged.
#statsdir /var/log/ntpstats/
statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# Specify one or more NTP servers.
server 192.168.100.101 iburst
disable auth
broadcastclient
```

after config is done, as ntp reconciliation is gradual, you'll want to make sure it's immediate (however likely not necessary, this is what the 'iburst' setting is above, accelerates convergence to 5-10 seconds):

```
sudo service ntp stop
sudo ntpd -gq
sudo service ntp start
```

and you can put in /etc/rc.local to start every time on system start (will add a few seconds to boot time)

```
( /etc/init.d/ntp stop
until ping -nq -c3 8.8.8.8; do
   echo "Waiting for network..."
done
ntpd -gq
/etc/init.d/ntp start )&
```

ROS

ROS is powerful; it's worthy mentioning some fancy tricks that are very useful for debugging.

fps of sensor	rostopic hz /KInect1/rgb/image_rect_color
json data	`roslaunch opt_utils udp_listener.launch`
more to come….	

Calibration

the instructions are great, but once can get lost. before starting, here's a high-level overview of the calibration process in a two camera system:

Remember to place Kinects in a configuration where the checkerboard can be seen by both of the cameras and follow this procedure:
1) show checkerboard to base camera
1.5) cover checkerboard with arm
2) align checkerboard to both cameras
2.5) remove arm
3) cover checkerboard with an arm
4) place the checkerboard on the ground so that (only) the base camera can see it
5) if stdout says everything is calibrated, save results
6) test multicamera tracking

repeat until there is very little to no jitter in a stationary person.

pretend the whole thing is a Tai-Chi exercise. Move slowly and with precision.

Covering the checkerboard is meant to reduce motion, so there is no frame-blur during calibration.

after that -

also adjust detection & tracking thresholds per environment & lighting condition - ie

open_ptrack/tracking/conf/tracker.yaml: min_confidence_initialization: 4.0
open_ptrack/tracking/conf/tracker.yaml: gate_distance_probability: 0.999
open_ptrack/detection/conf/haar_disp_ada_detector.yaml:

misc notes:

checkerboard must be shown to base camera before any other cameras
the order of camera calibrations, after base camera, does not matter / does not need to match camera_network.yaml
when saving ground plane / final step to calibration - the checkerboard can be visible to all cameras.

For improving calibration, here are some tips:

1) for every couple of cameras which are calibrated one with respect to the other, the last estimated transformation is kept, thus the last checkerboard they see is used, even if it is not the 'best' checkerboard they have seen (I am going to improve this in the next versions, but so far it works this way). Knowing this, you can move the checkerboard in order to place it in the best position where they both see it (best position is the position nearest to the cameras and with the minimum slant with respect to the cameras) and then

cover the checkerboard with an arm in order to avoid cameras to detect another checkerboard while you are moving to another couple of cameras and so on.
2) to avoid the problem of delays between images coming from different sensors, stop a bit in front of every couple of cameras you calibrate (before covering the checkerboard with the arm, etc...)
3) be sure that your checkerboard is rigid (the one I was using in the videos I sent was still far from being rigid), otherwise transformations between cameras are badly estimated.
4) be sure that your checkerboard has an even number of squares in one direction and an odd number of squares in the other direction. Otherwise, the checkerboard position can be flipped by the algorithm.

Kinect

there is an issue with the openni driver - if dmesg reports all zeroes for kinect, then you have to follow ‘no serial’ calibration and are limited to one kinect per node.

however if dmesg returns a serial - great ! you can have N cameras per node (limited only by cpu and USB bus - one per 2.0 bus).

Ideally you want all serial #s, no error - yet sometimes you need to get fancy, see ‘hybrid kinect’.

note the you don’t have to do intrinsic just to get the serial #s - they’re visible in dmesg / syslog when you plug the cameras in.

Hybrid Kinect

For a ‘hybrid’ install - that is, a network with both model 1414 and 1473 (i.e., with kinects that give valid serial #s (visible in dmesg), and those that show 00000 (an error)), note:

1) ‘zeroed’ kinects cannot be used as base camera

2) there is some manual editing involved:

First, setup calibration normally. Set up camera_network.yaml to your network, ie -

```
num_cameras: 5
camera0_id: "A00365819196040A"        #normal serial
camera1_id: "B00363711241052B"        #normal serial
camera2_id: "Kinect2"                 #no serial
...
calibration_with_serials:  true
```

then run roslaunch opt_calibration calibration_initializer.launch

then before launching calibration, change one line in the file opt_calibration_master.launch:
(in ~/workspace/ros/catkin/src/open_ptrack/opt_calibration/launch)

<param name="calibration_with_serials" value="false" />

set to true, if it's not already.

also, before launching the sensor_KinectX.launch (or anything without serial) first remove line 9:

<arg name="device_id" value="$(arg camera_id)" />

then, launch, and calibrate per normal. (roslaunch opt_calibration opt_calibration_master.launch, sensors, etc)

after calibration is saved, inspect every detection_node launch file -

if it's a kinect with serial, make sure it is:

<include file="$(find detection)/launch/detector_serial.launch">

if it's a kinect without serial, make sure it is:

<include file="$(find detection)/launch/detector_with_name.launch">

then launch detection per normal.

Mesa imager

they often forget themselves and/or it's hard to find IP.

if you don't know the IP, but you do know the subnet, do

sudo nmap -v -sP 192.168.1.0/24

and you'll see something like

Nmap scan report for 192.168.1.40 [host down]
Nmap scan report for 192.168.1.41 [host down]
Nmap scan report for 192.168.1.42
Host is up (0.00028s latency).
MAC Address: 00:1C:8D:00:0B:53 (Mesa Imaging)

Note the mesa is far tricker to calibrate in a camera network than kinects, due to the low spatial resolution of it's 'RGB' image.

Recording data for replay

it is possible to easily record Kinect or SwissRanger or other data which are published as ROS messages.

rosbag is the ROS tool for doing this:
http://wiki.ros.org/rosbag
http://wiki.ros.org/rosbag/Commandline

rosbag record -b0 <topicName> -O <filename.bag>
records messages published to topic <topicName> in a file named <filename.bag>

The created .bag file can then be played with:
rosbag play <filename.bag>

You can also record and play multiple topics and they maintain their synchronization.

However, the problem of recording Kinect data is related to the big amount of data it publishes.
Recording Kinect point clouds requires 100MB/s of disk space.

What I usually do is to record rgb and depth images and then I re-create the point clouds afterwards from the recording.

Maybe this procedure is not very easy for everybody and still recording streams from many cameras is quite heavy, thus it would be better to just record tracking output and maybe the RGB stream of one camera to have a visual reference.

This could be done this way:
rosbag record -b0 /<cameraName>/rgb/image_color /tracker/tracks -O <filename.bag>

Lighting & detection discussion

in summary, both sensors - as well as algorithm - can indeed run in darkness.
the 2D image detection is done w/ IR / B&W, does not need RGB.

details:

The detection process uses the point cloud for segmentation and then humans are detected by means of classifiers on 2D images.
For Kinect, these 2D images can be RGB and/or disparity images.
For SwissRanger, these 2D images are infrared intensity images.

Thus, SwissRanger people detection will always work the same both in the dark and in the light.

For Kinect, there is a 'use_rgb' flag in 'detection/conf/ground_based_people_detector.yaml'. If that flag is set to false, people detection is done only in the disparity image, thus it will work also with no light.
If that flag is set true, RGB images are used to refine people detection if the mean luminance of the image is above the 'minimum_luminance' threshold, which is defined in the same configuration file. This parameter can take values between 0 and 255 and you can have the best results by setting this threshold according to your scene illumination.
Thus, if images are too dark, the depth-only approach is used and people detection continues to work.

About the tests from 'low' cameras, I have the feeling that the problem is related to the fact that you are using the SwissRanger and you have an object (the ground) very close to the sensor. In that case, the confidence of the points usually become very low, and since we are thresholding points based on confidence, you could end up with no valid points in the point clouds.
To see if this is the problem, you could visualize with RViz the '<swissranger_name>/confidence/image_raw' topic and check the grey values of most of the points.
In order for the system to work, most points should have confidence above the 'sr_conf_threshold' in the 'detection/conf/ground_based_people_detector_sr_multicamera.yaml' file.
Maybe you could visually compare the confidence image of the SwissRanger at a standard height with the confidence image you get when it is close to the ground.