SIMULTANEOUS LOCALISATION AND MAPPING (SLAM) WITH ROLLO

Rabbia Asghar, BEng, Ernest Skrzypczyk, BSc

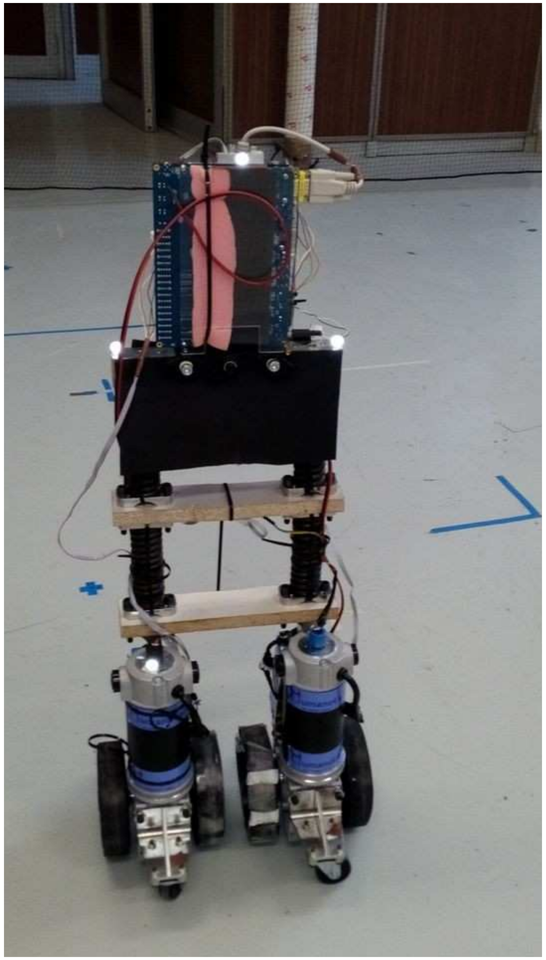
SLAM with Rollo

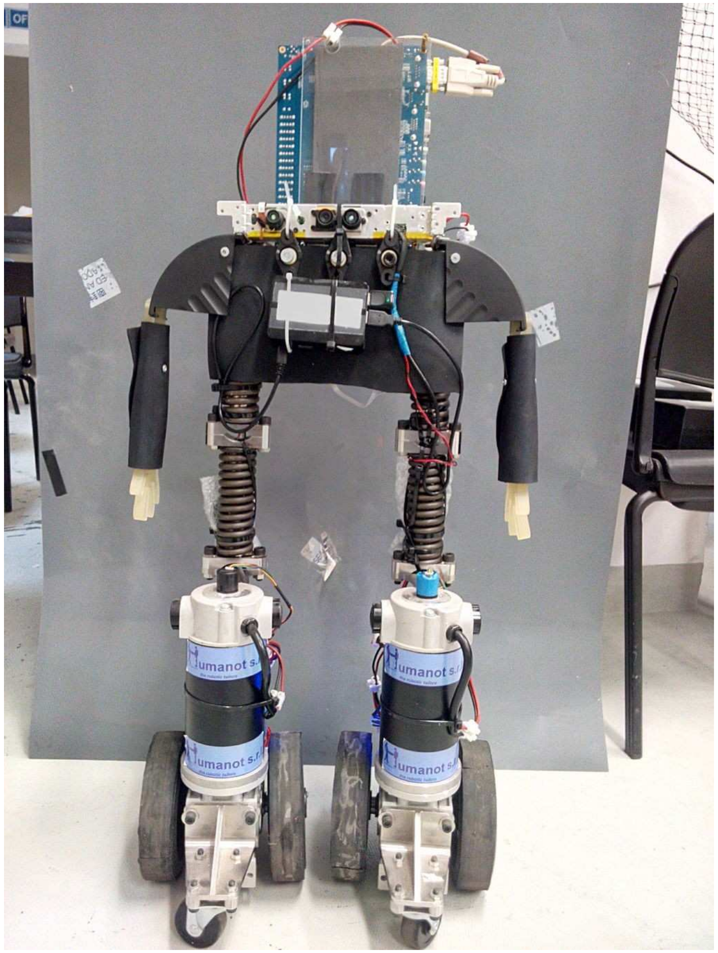
- Problem Statement
- Experimental Setup
- RGB-D SLAM algorithms
- Configuration of the whole system
- Evaluation of Algorithms

Problem Statement

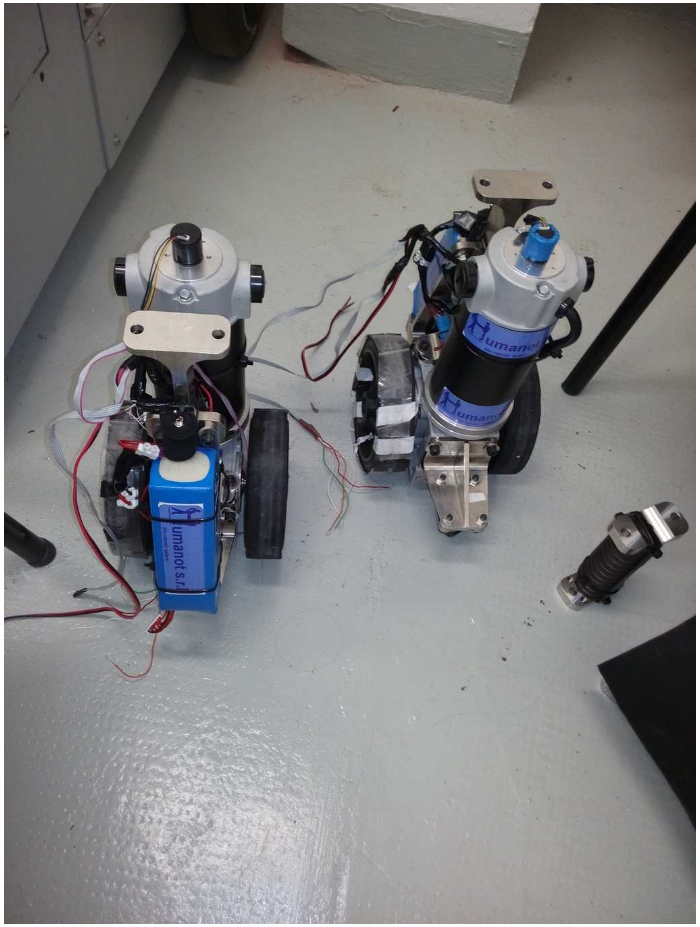
- What is SLAM?
  - Problem of building a map from surrounding environment while simultaneously performing localization
  - Often referred to as chicken and egg problem
- Where is it used?
  - Mapping an unknown environment
  - Localization, motion planning and navigation
- Why is it important?
  - Autonomous navigation of mobile robots
  - Complex dynamic environments

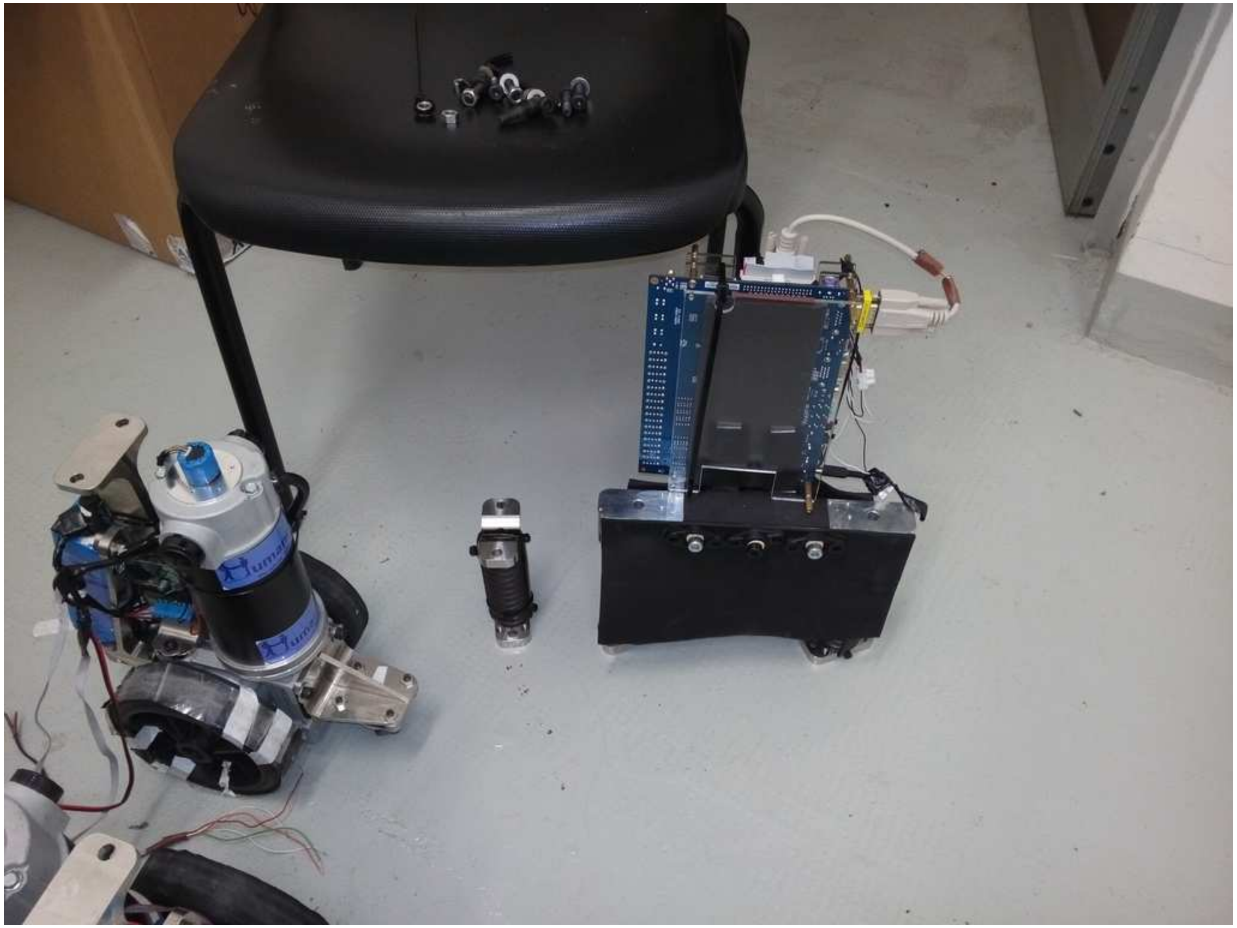
Experimental Setup



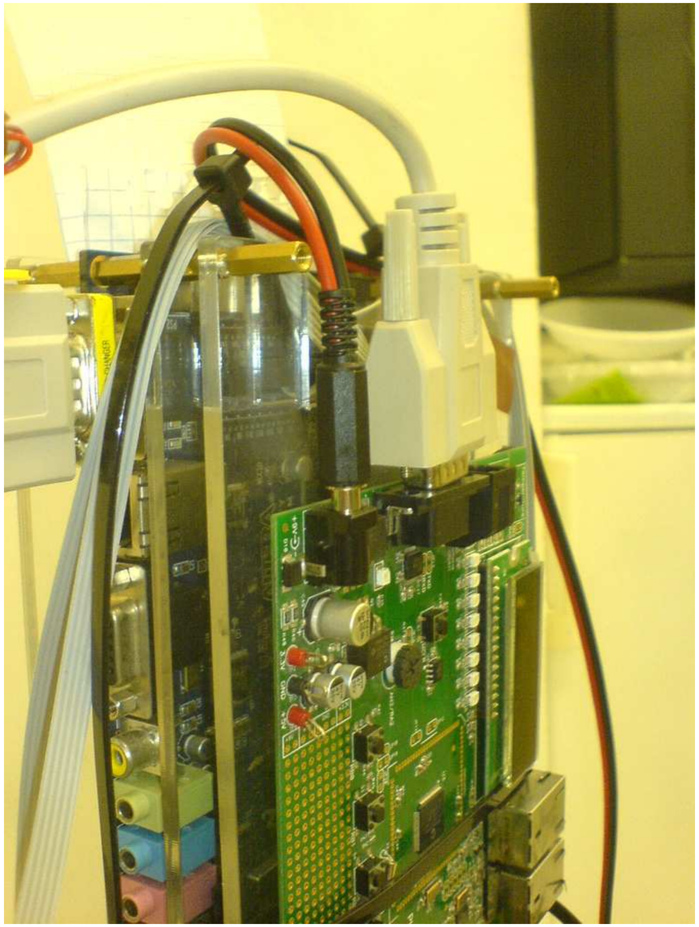


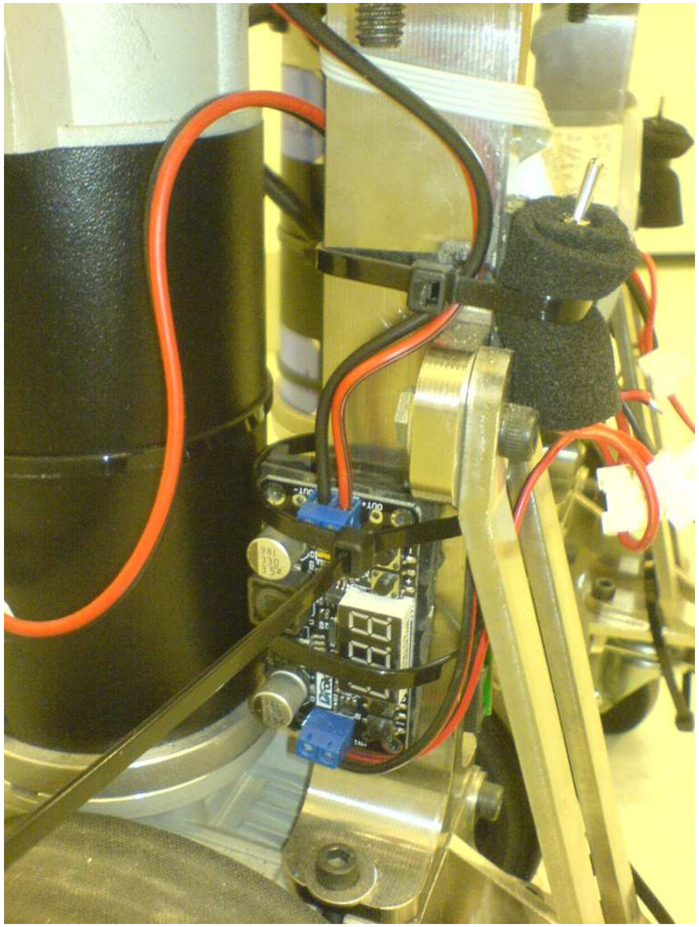
Mechanical Improvements



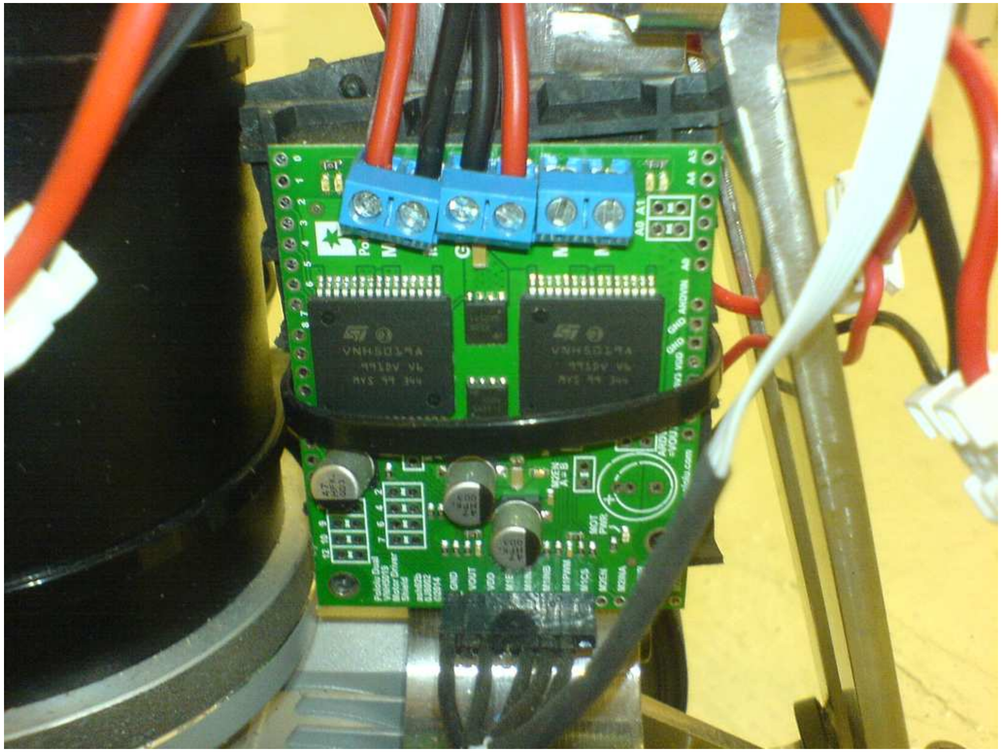


Electrical Improvements

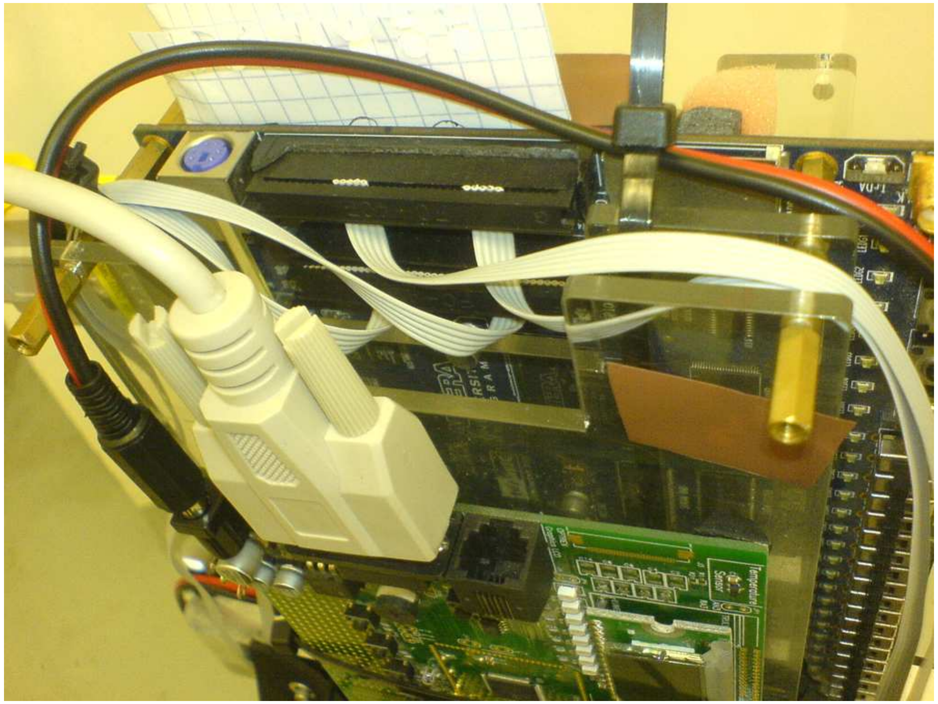




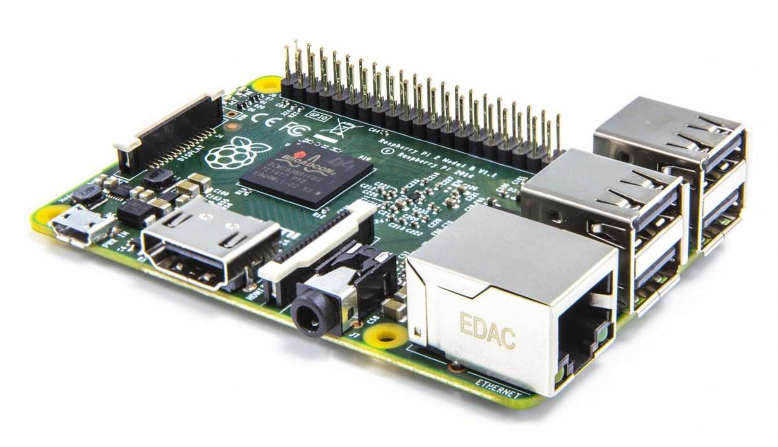
Electrical Improvements



Electrical Improvements

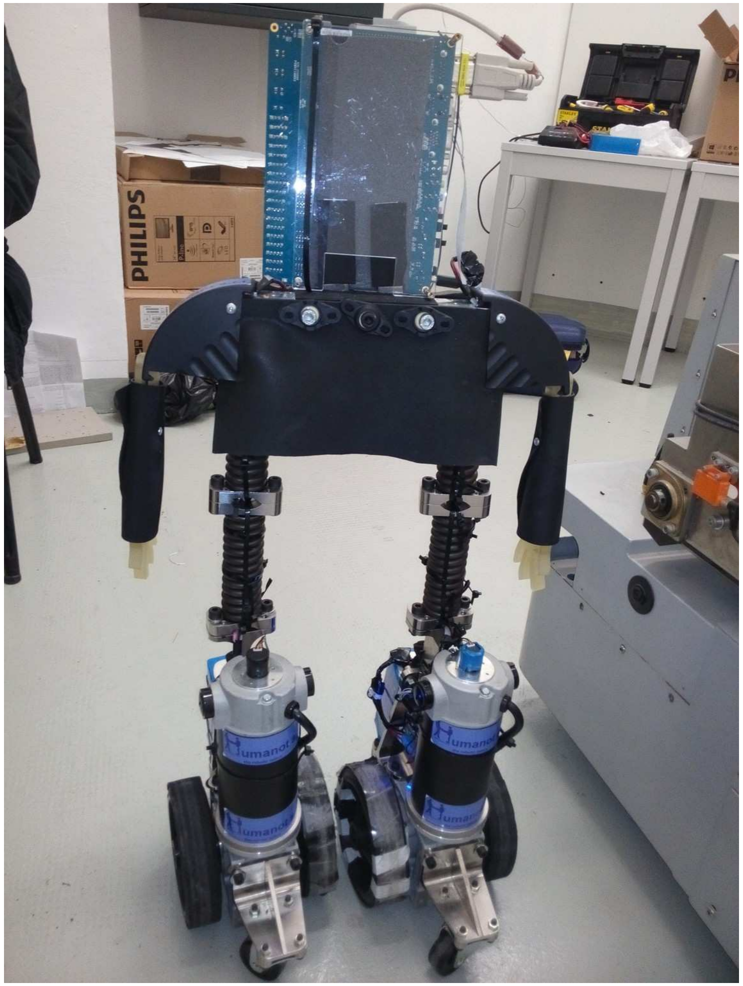


Main Components



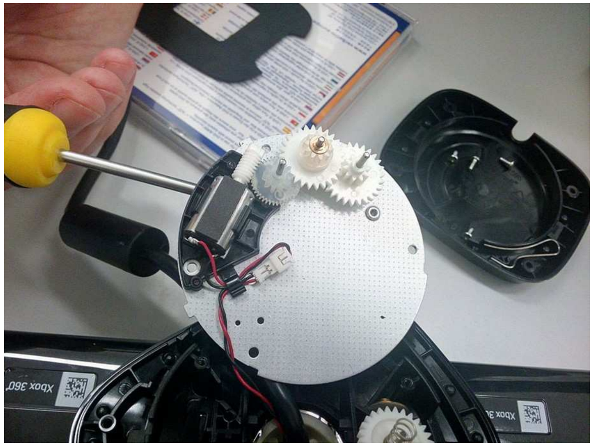
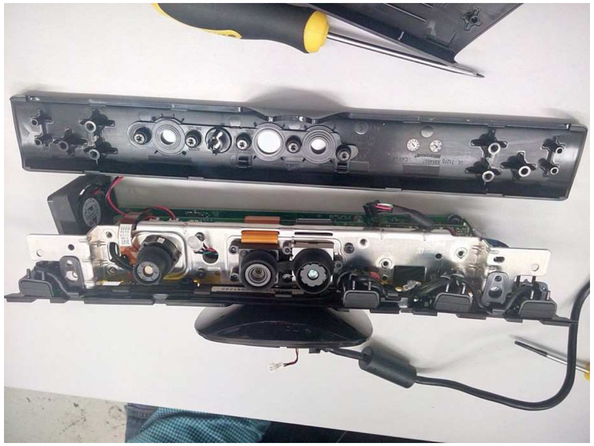
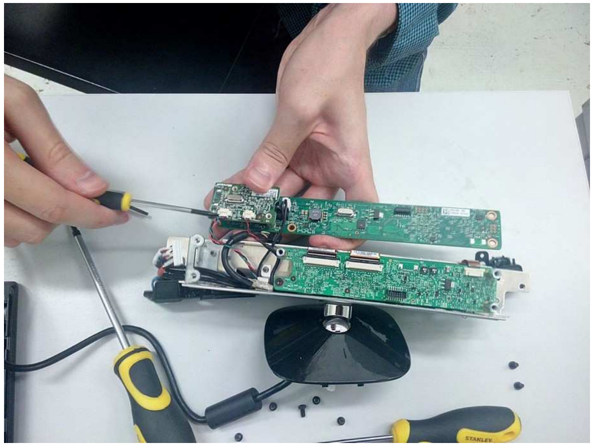
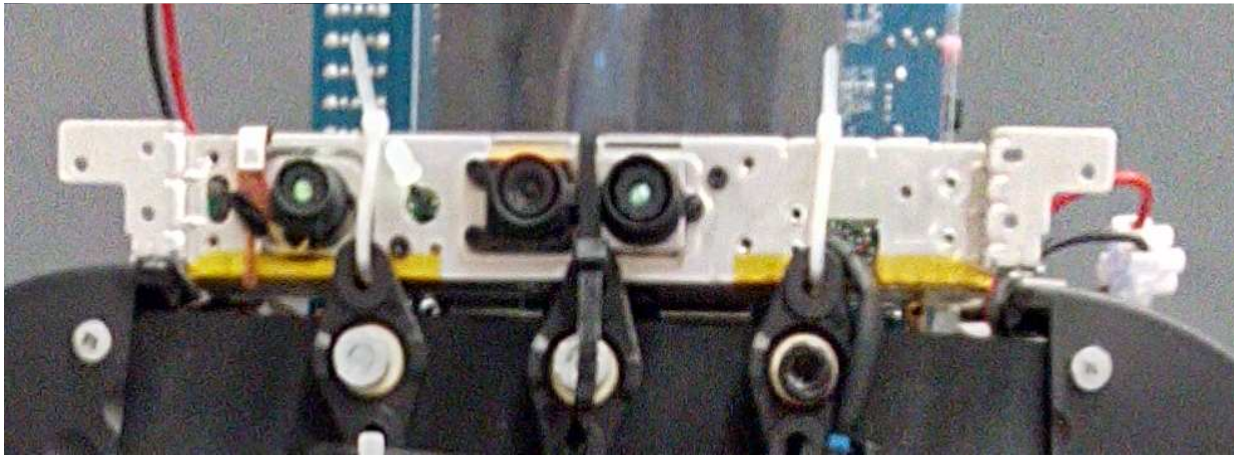
Raspberry Pi Model B v1.1

Kinect v1

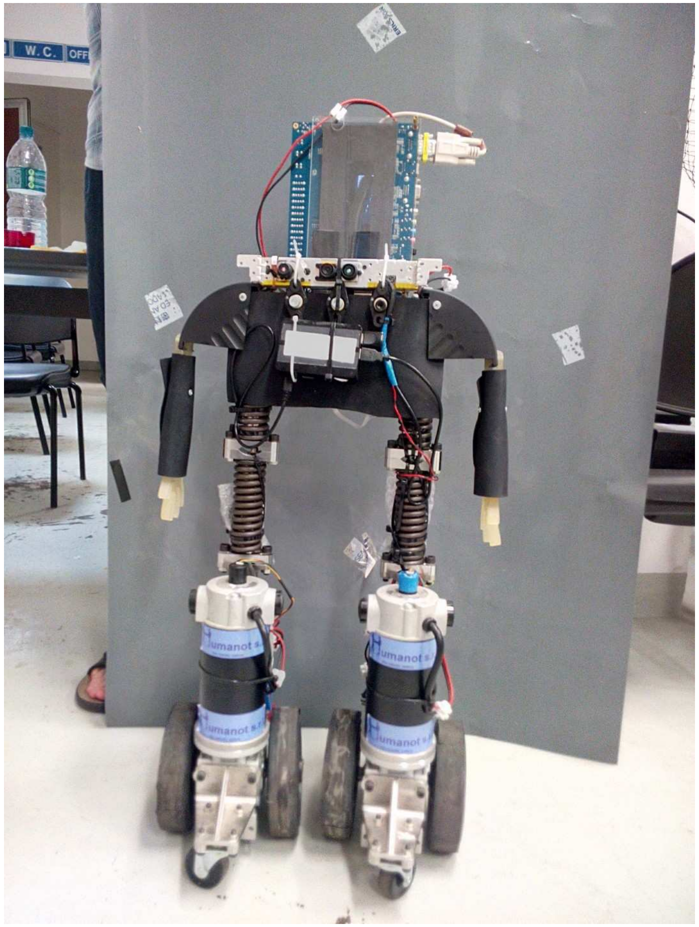


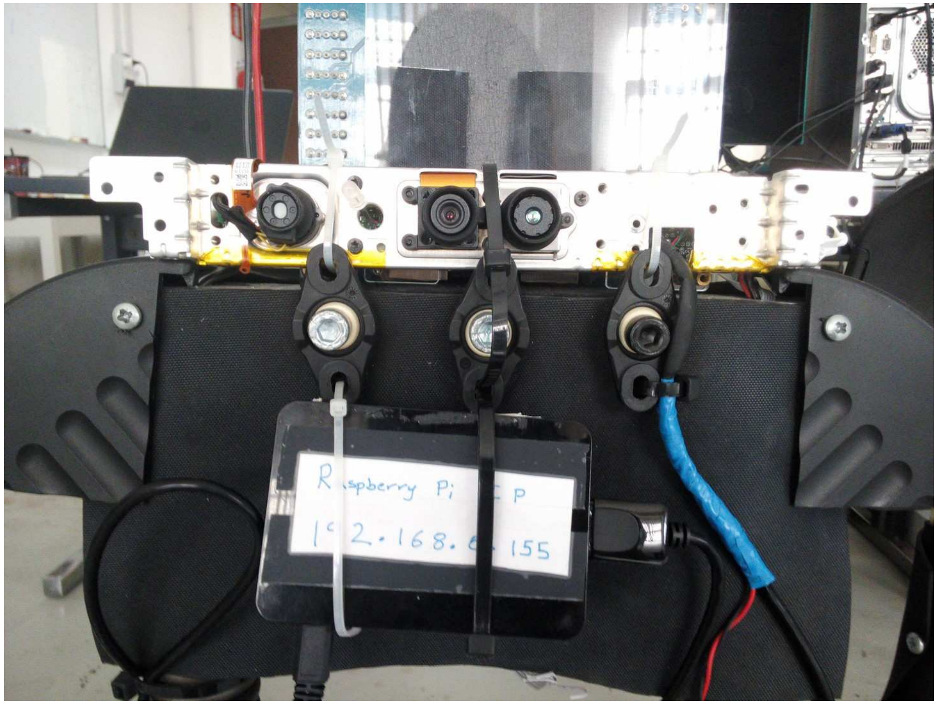
Rollo

Kinect Modification

Rollo – final state



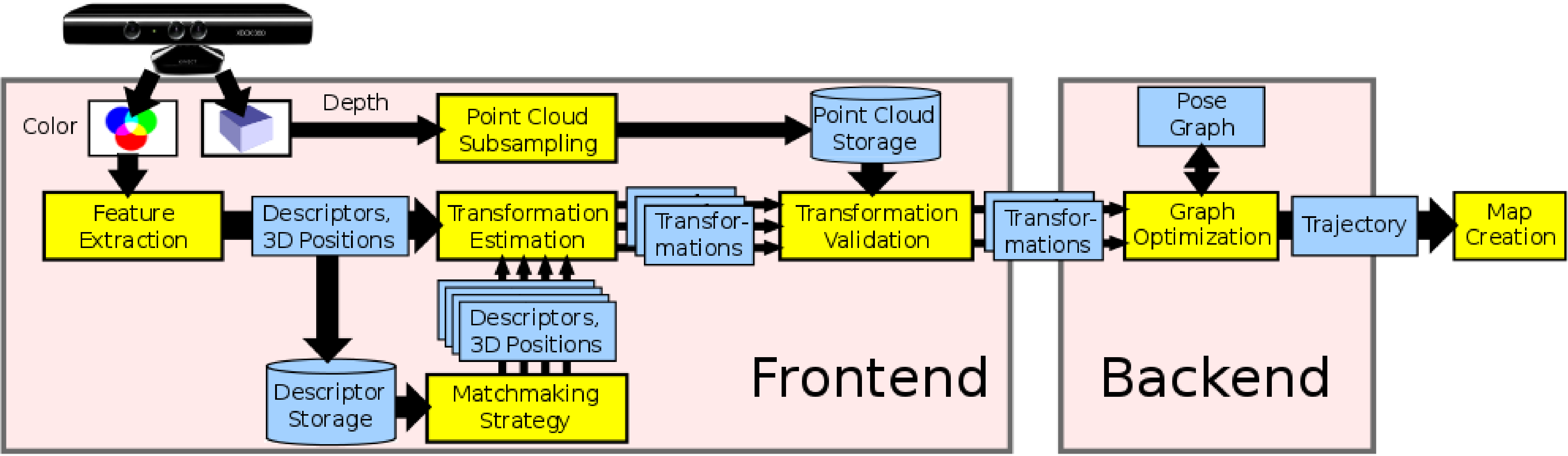


SLAM Algorithms

- RGBD SLAM v2
- RGBD Mapping and Relocalisation
- RTAB MAP (Real-Time Appearance-based Mapping)

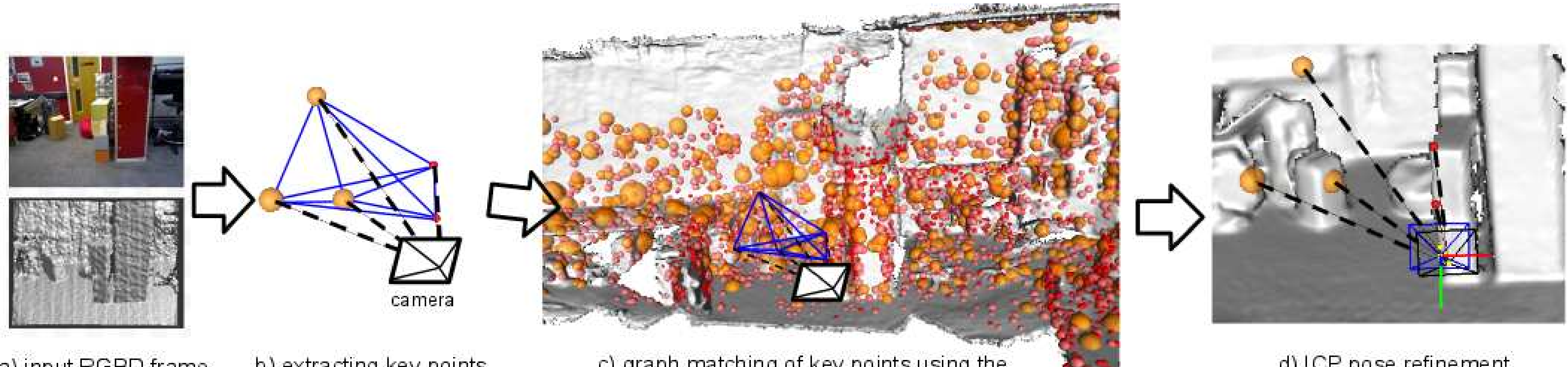
RGBD SLAM v2

- Graph-based SLAM system
- Three main modules
- Offers four extraction feature extractions techniques



RGBD Mapping and Relocalisation

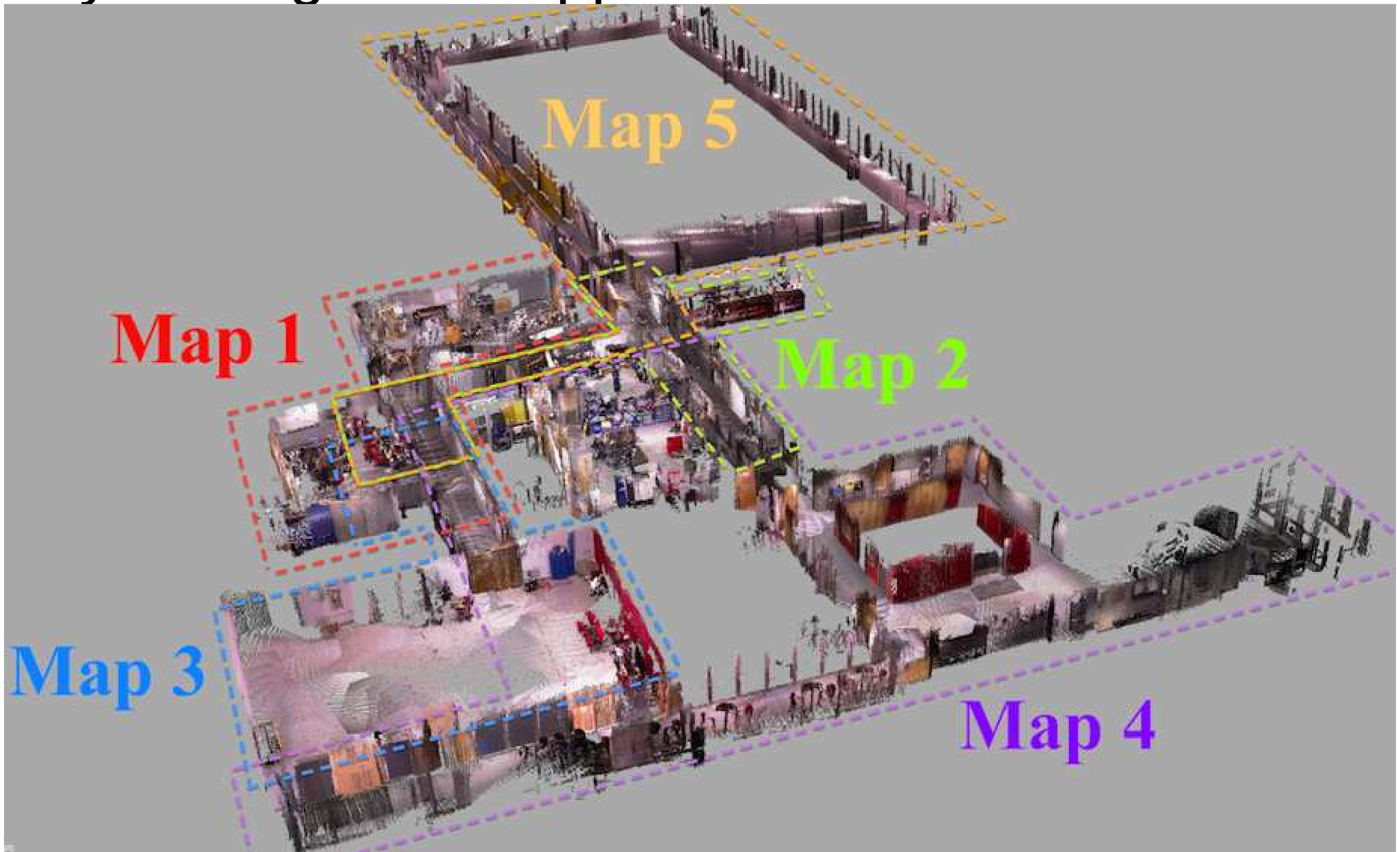
- Based on key point matching
- Two main components: a graph matching algorithm and points selection process
- Graph matching algorithm uses pairwise 3-D geometry amongst the key points for relocalisation
- points selection process (non-maximum suppression algorithm) is used to limit the number of keypoints



a) input RGBD frame b) extracting key points c) graph matching of key points using the pairwise geometry d) ICP pose refinement

RTAB MAP

- Graph-based SLAM
- Can deal with multi-session mapping problem
- Uses a bag-of-words approach
- Incorporates a memory management approach



Configuration and Troubleshooting Installation on GPU host

Host specifications:

- Ubuntu x86_64
- Intel Core i7 4790 with 8 cores @ 3,60 GHz
- NVidia GeFore GTX970

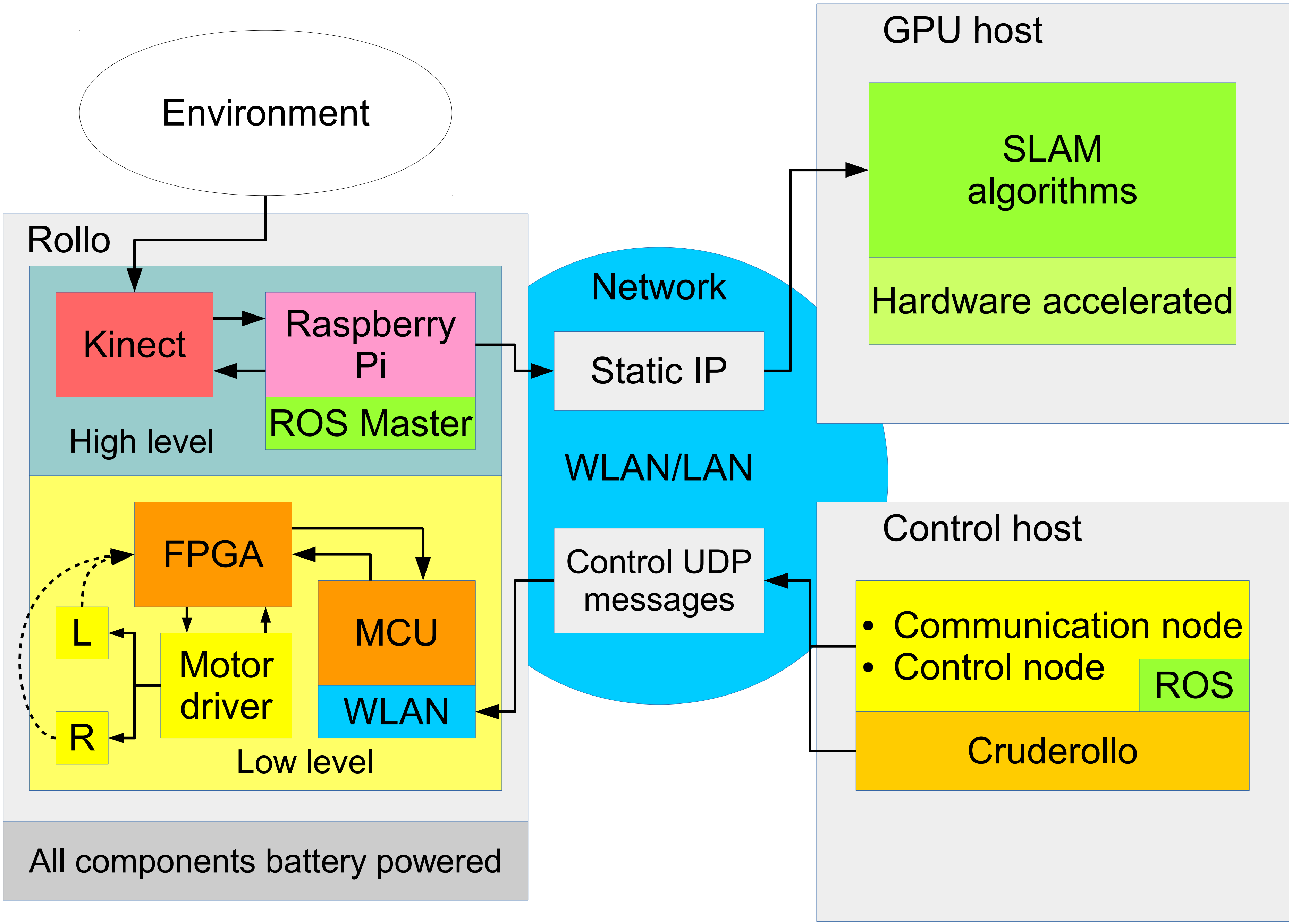
Installation of SLAM algorithms

- RGBD Mapping and Relocalisation manual
- RGBD SLAM v2
- RTabMap

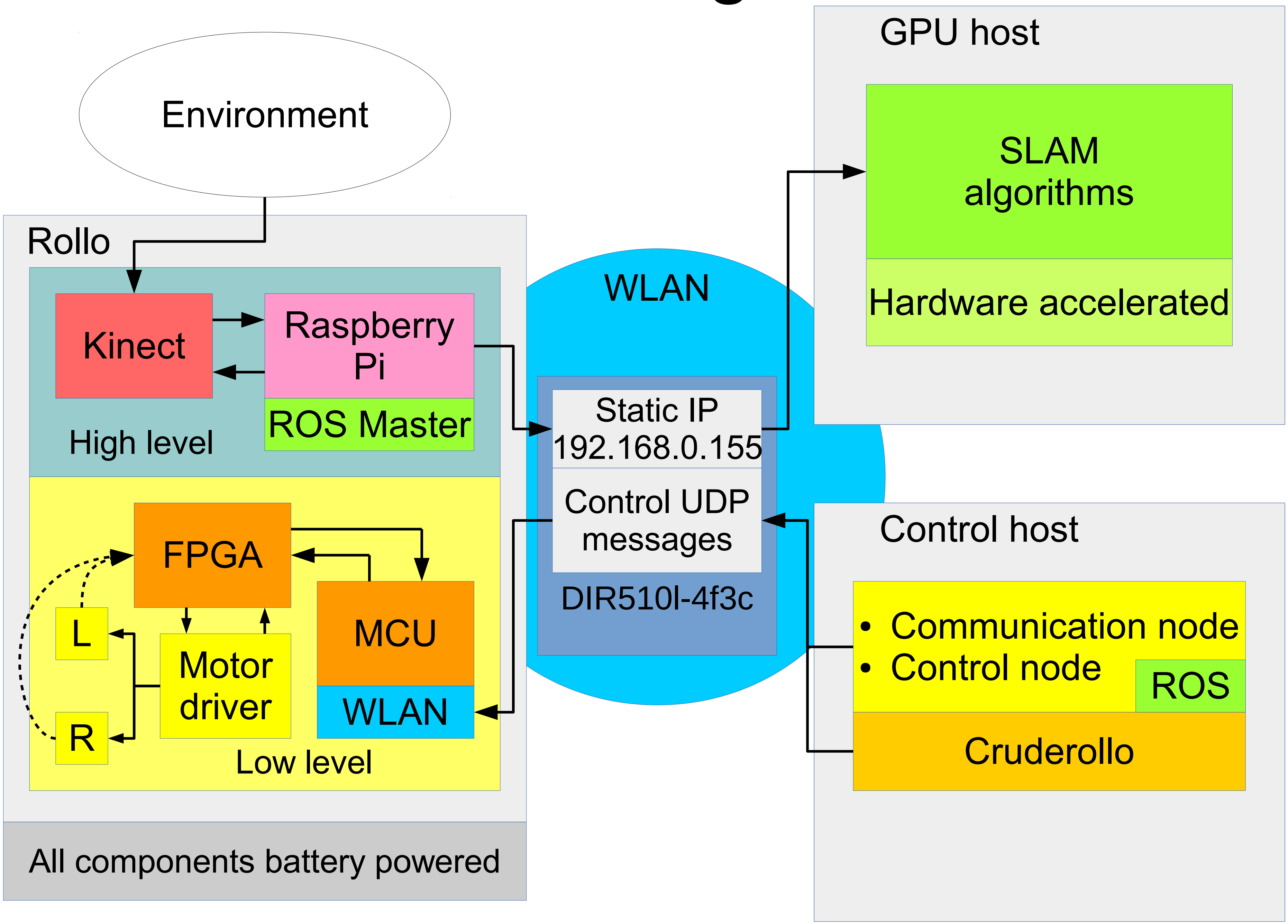
Network configurations

- Wireless network
  - Mobile router DIR510l-4f3c
  - Laboratory network EmaroLabUnify
- Wired network
  - Router functioning as a LAN bridge

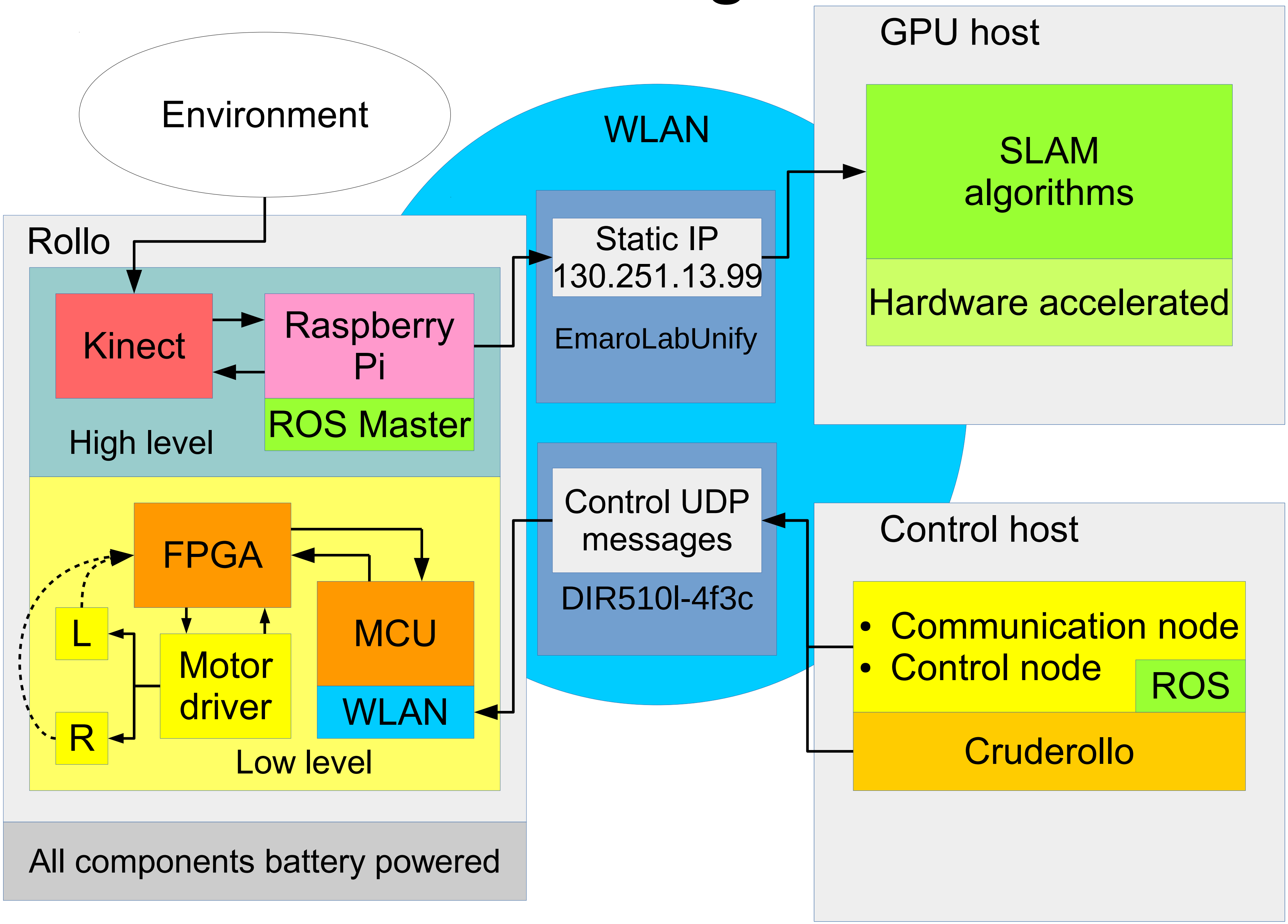
General scheme



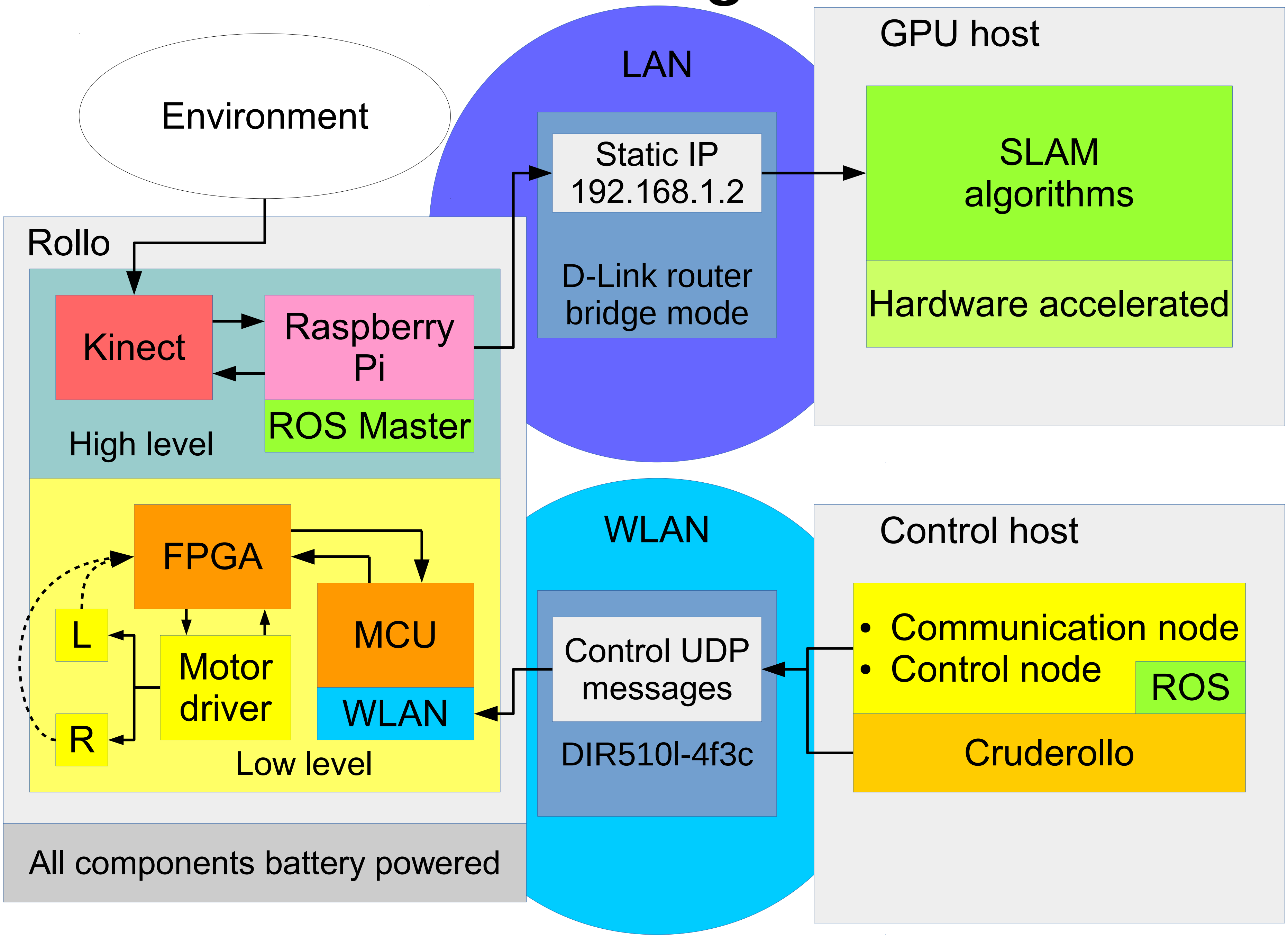
Network configuration A



Network configuration B



Network configuration C



Data transmission rates

Device	Data rates [Mbit/s]	Data rates [MB/s]
Raspberry Pi 2 Wifi Dongle	150	18.75
250 Mbps device	250	31.25
TP-link TL-WN823N	300	37.5
500 Mbps device	500	62.5
Kinect 640 x 480 @ 30 fps, 5 channels	368.64	43.95
Kinect 640 x 488 @ 30 fps, 5 channels	374.74	46.85
Kinect 1280 x 1024 @ 15 fps, 5 channels	786.43	93.75
Lan 1Gbit/s	1000	125

Comparison of data transmission rates for different devices

Connection schemes

- Kinect connected to Raspberry Pi
- Kinect directly connected to GPU host

Installation and Configuration Raspberry Pi

- Correct previous installation errors
- Setup for different network configurations
- Energy efficiency optimization

Energy Consumption Optimization

- Motivation
  - Longer operation time in the field
  - More maps and more comfortable operation
- Main configuration aspects
  - CPU, RAM, power profiles and governors
  - Specific mount parameters for block or other type mount devices
  - Module powersaving options
  - Medium and low level software and kernel optimization
  - System and ROS configuration optimized for given task performance

Energy Consumption Optimization

- Starting of rosmaster from ~/.bashrc was removed.
- The root password was set to rpi
- Several modules were disabled:
  - snd_bcm2835, snd_hda_intel, snd
- Additional entries for the temporary filesystem in file fstab
- ROS logging was disabled
  - Using environment variables and specific component configuration.
  - Set tmpfiles configuration for the log directory to periodically, here every 15 minutes, clean the directory.
- Swappiness in /proc/sys/vm/swappiness was already set to 1.
- Using the Raspberry Pi configuration tool raspi-config
  - Overclocking was disabled, ergo set to None.
- Enabled at multi-user.target level a systemd service and script for automatic start of the Freenect ROS launcher once the WLAN connection has been established.

Systemd service and script

- Problem
- Systemd solution
  - Service unit
  - Script
- Temporary files
  - Configuration
  - Timer
- ROS logging
  - Environment

Systemd service and script

```
[Unit]
Description=ROS
After=network.target

[Service]
Type=simple
Nice=-10
ExecStart=/etc/systemd/scripts/ros
ExecStop=/etc/systemd/scripts/roskill
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

```
#!/bin/bash
ROSPIDS=$(ps aux|
awk '/ros\/indigo/{print
$2}');
for ROSPID in
$ROSPIDS; do
    kill -9 $ROSPID;
done
```

Systemd service and script

```
#!/bin/bash
export ROS_IP="$(hostname -I | sed 's/\ /\n/g' | sed '/130.251/!d')";
export ROS_MASTER_URI="http://130.251.13.99:11311"

echo 0 > /sys/class/net/wlan0/device/driver/module/parameters/rtw_power_mgnt
echo 0 > /sys/class/net/wlan0/device/driver/module/parameters/rtw_lowrate_two_xmit
echo 0 > /sys/class/net/wlan0/device/driver/module/parameters/rtw_low_power
cat /sys/class/net/wlan0/device/driver/module/parameters/rtw_power_mgnt
/sys/class/net/wlan0/device/driver/module/parameters/rtw_lowrate_two_xmit
/sys/class/net/wlan0/device/driver/module/parameters/rtw_low_power

SLEEP_INIT=3;
SLEEP_PERIOD=3;
LOCK=/tmp/ros

sleep $SLEEP_INIT;
env > $LOCK

while :; do
if [[ $(ifconfig 2>&1 | grep 'wlan0' -A 4 | grep RUNNING) ]]; then
echo Wireless network $WLAN_NET found. Starting ROS Freenect launcher...;
```

Systemd service and script

```
if [[ $(ifconfig 2>&1 | grep 'wlan0' -A 4 | grep RUNNING) ]]; then
echo Wireless network $WLAN_NET found. Starting ROS Freenect launcher...;
source /opt/ros/indigo/setup.bash
source /home/pi/ros_catkin_ws/devel_isolated/setup.bash

/opt/ros/indigo/bin/roslaunch freenect_launch freenect.launch image_mode:=2 respawn:="true" &
sleep $SLEEP_PERIOD;
#Change image mode to high resolution 1:=1280x1024@15fps, 2:=640x480@30fps, 0:=small (320x240@30Hz)
#  /opt/ros/indigo/bin/rosrun dynamic_reconfigure dynparam set /camera/driver image_mode 1 &
echo ROSmaster PID $(ps aux | awk '/rosmaster/{print $2}') > "$LOCK"master;
echo Breaking
break;
fi
sleep $SLEEP_PERIOD;
done

echo Finished >> $LOCK
sleep $SLEEP_PERIOD;
echo Looping
while ps aux | grep -c freenect; do sleep $SLEEP_PERIOD; done &> /dev/null
```

Videos of RGBDSLAM v2

- Problem of receving a very limited number of frames in an inconsitent manner
- Mechanical challenges of Rollo

Evaluation of Algorithms

Benchmark	Device	Camera Trajectory	Scene Geometry	Global Coordinate system
Meister	Kinect v1	no	ground truth	no
Sturm	**Kinect v1**	**ground truth**	**no**	**no**
Zhou	Xtion Pro	computed	computed	no
Handa	synthetic	synthetic	synthetic	no
CoRBS	Kinect v2	ground truth	ground truth	yes

Comparison of state-of-the-art RGB-D benchmarks

Evaluation of Algorithms

- Offline evaluation
  - Public datasets
  - Datasets collected with Rollo
- Online evaluation

Offline evaluation with Public Dataset

- Sturm’s dataset offers 39 trajectories
- Already calibrated and time synchronised RGB-D data
- Run it with a specific SLAM algorithm and record estimated trajectory
- Evaluate with comparison with ground truth
  - Relative pose error (RPE)
  - Absolute trajectory error (ATE)

RPE and APE

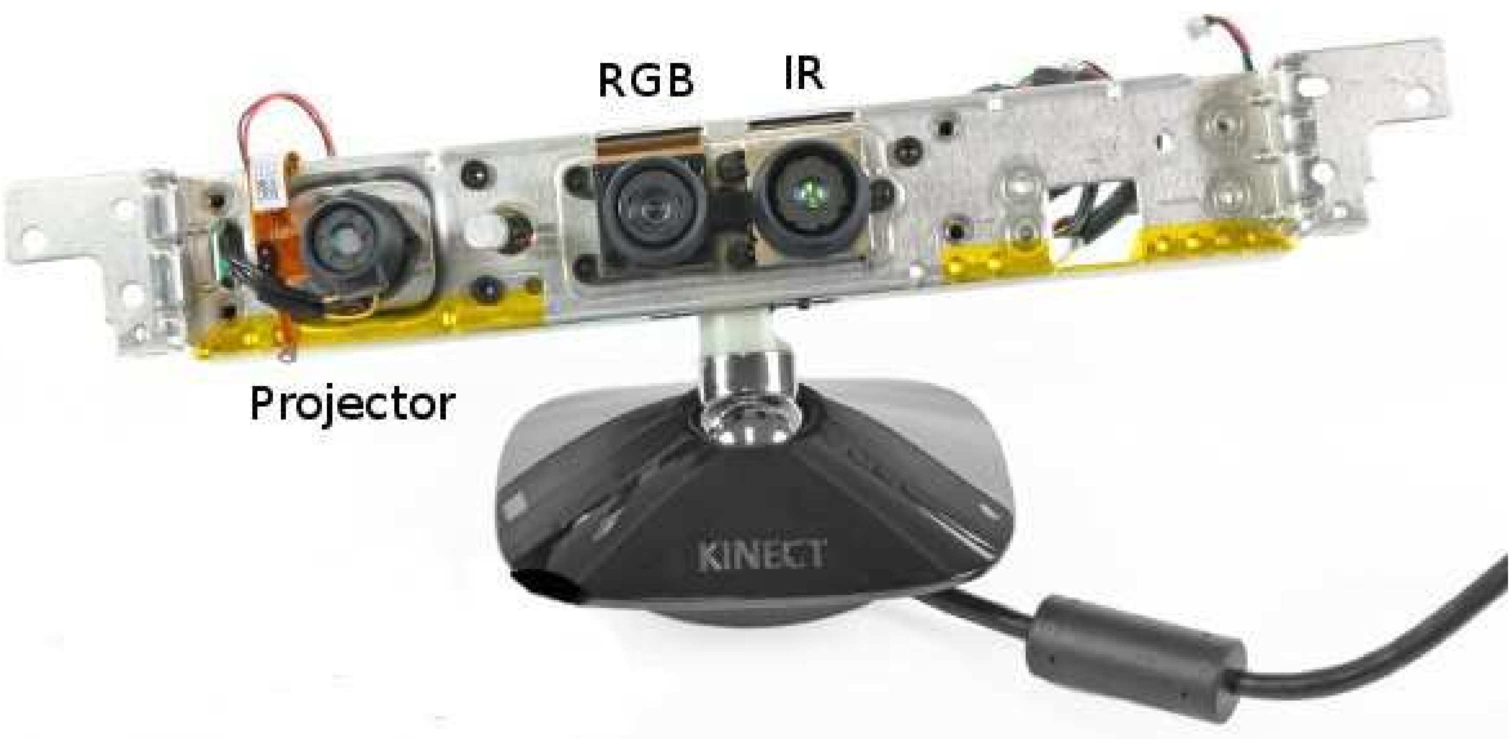
- Relative pose error (RPE): relative differences between the ground truth poses and the estimate for given time steps
- Corresponds to drift of trajectory
- Absolute trajectory error (ATE): difference in terms of the absolute offset between corresponding poses.
- ATE offers intuitive visualisation
- RPE and APE are highly correlated

Offline evaluation with dataset collected with Rollo

- Intrinsic camera calibration
- Extrinsic camera calibration with motion capture system
- Time synchronisation between RGB and depth image
- Time synchronisation between motion capture system and RGB camera

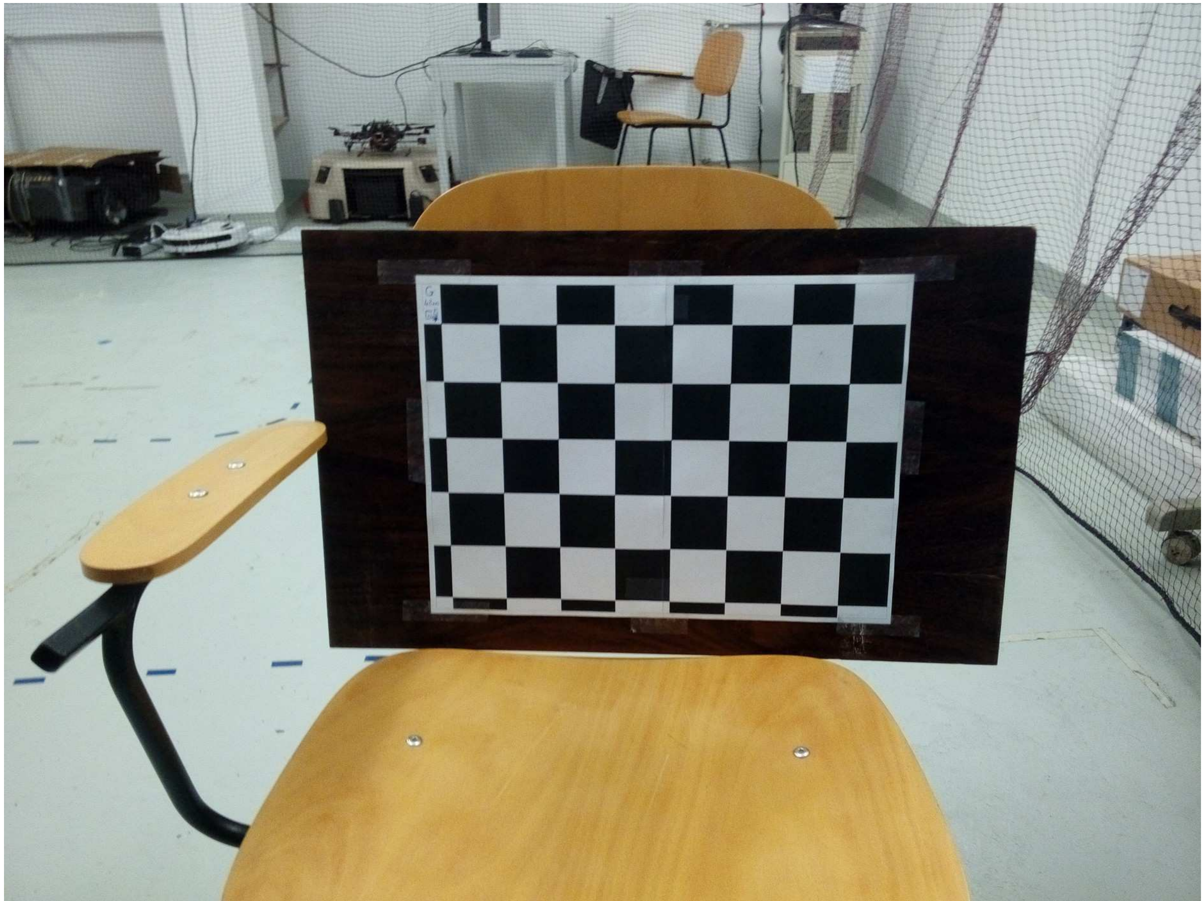
Camera calibration process

- RGB camera intrinsic calibration
- IR camera intrinsic calibration
- Depth image calibration



Calibration process setup

- Kinect connected to GPU host via USB 3.0 socket
- Rollo $\rightarrow$ Static
- Calibration pattern



Camera calibration

- Radial distortions correction

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

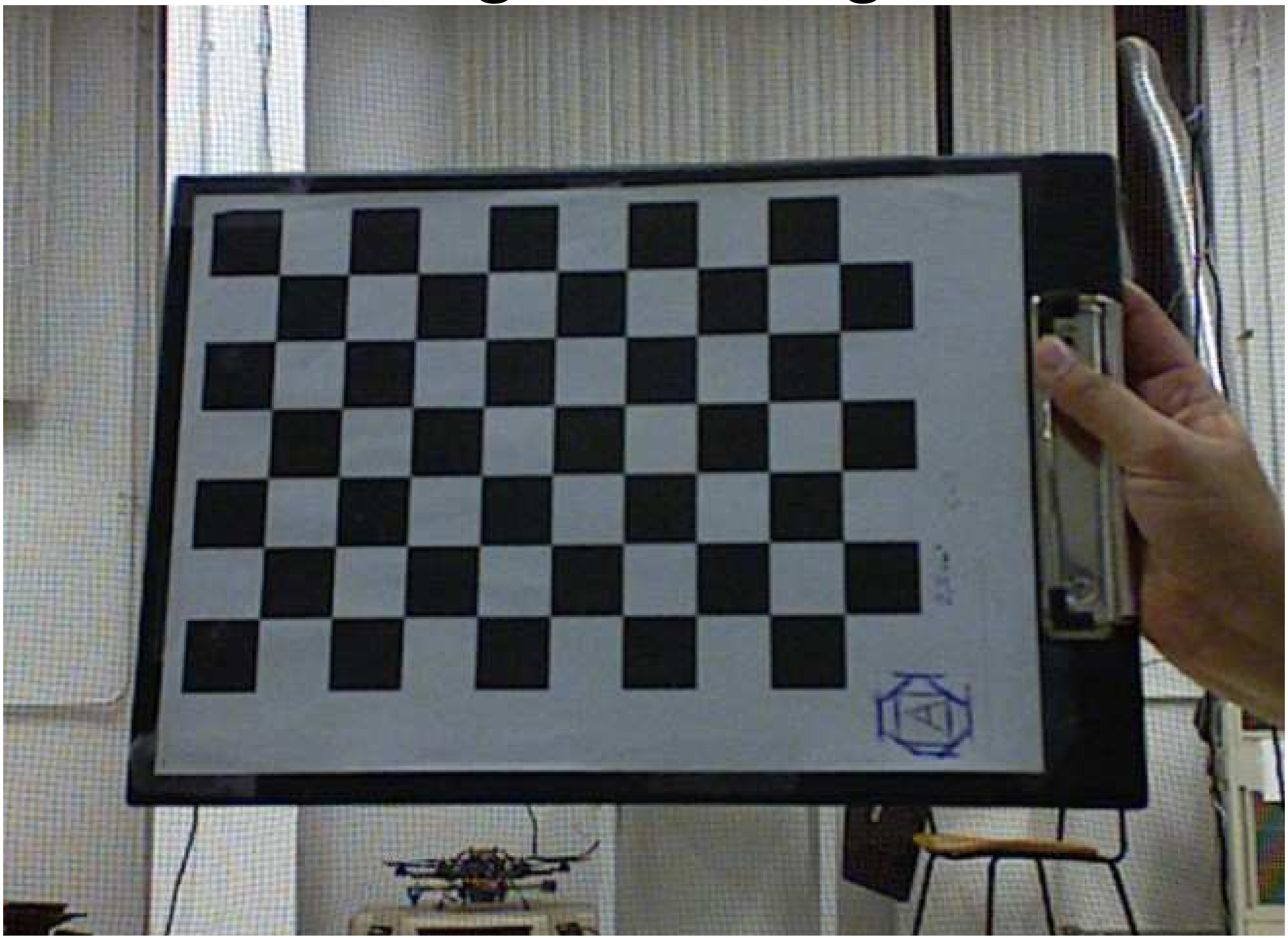
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

- Tangential distortions correction

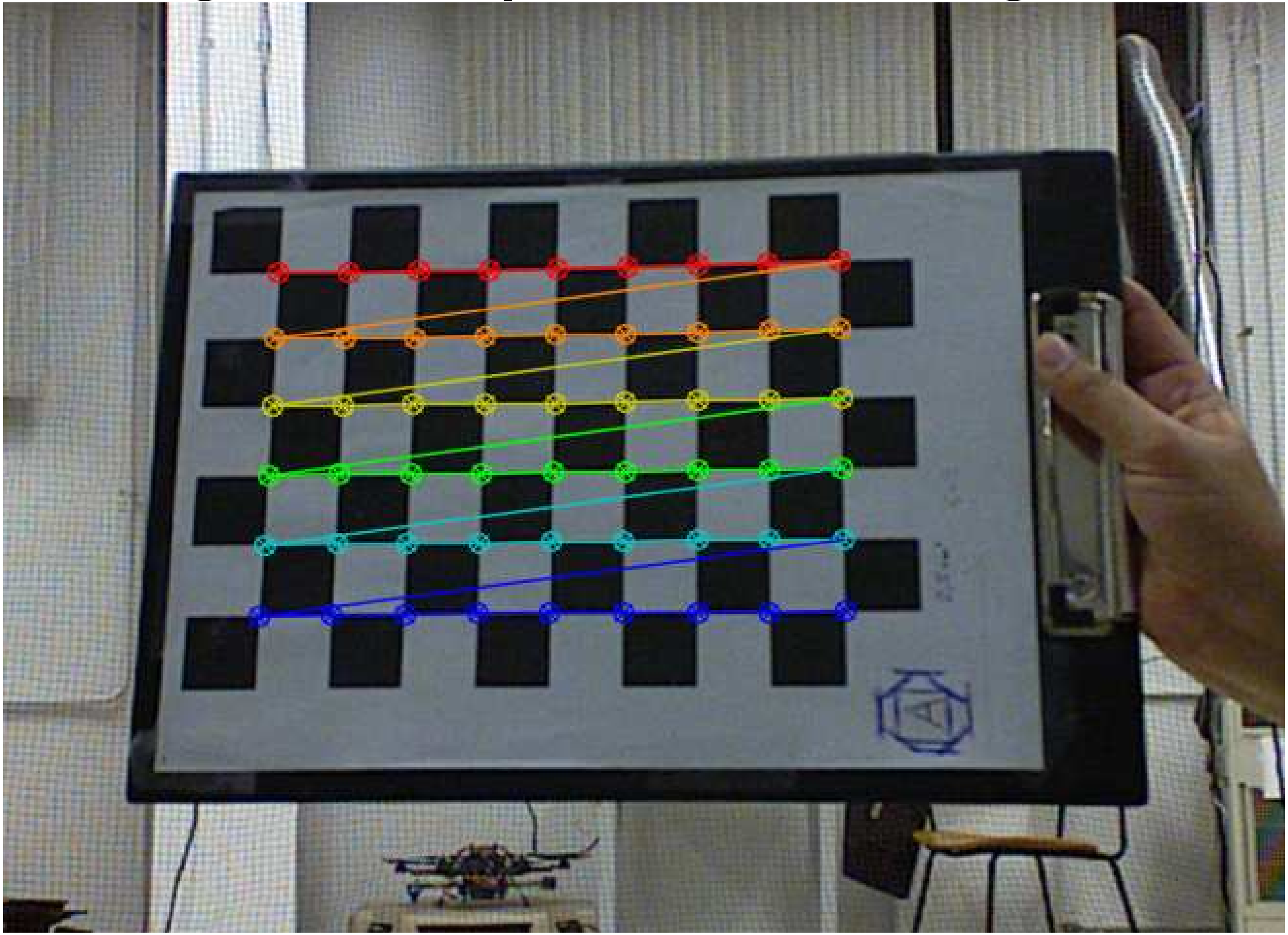
$$x_{corrected} = x[2p_1 xy + p_2(r^2 + 2x^2)]$$

$$y_{corrected} = y[2p_2 xy + p_1(r^2 + 2y^2)]$$

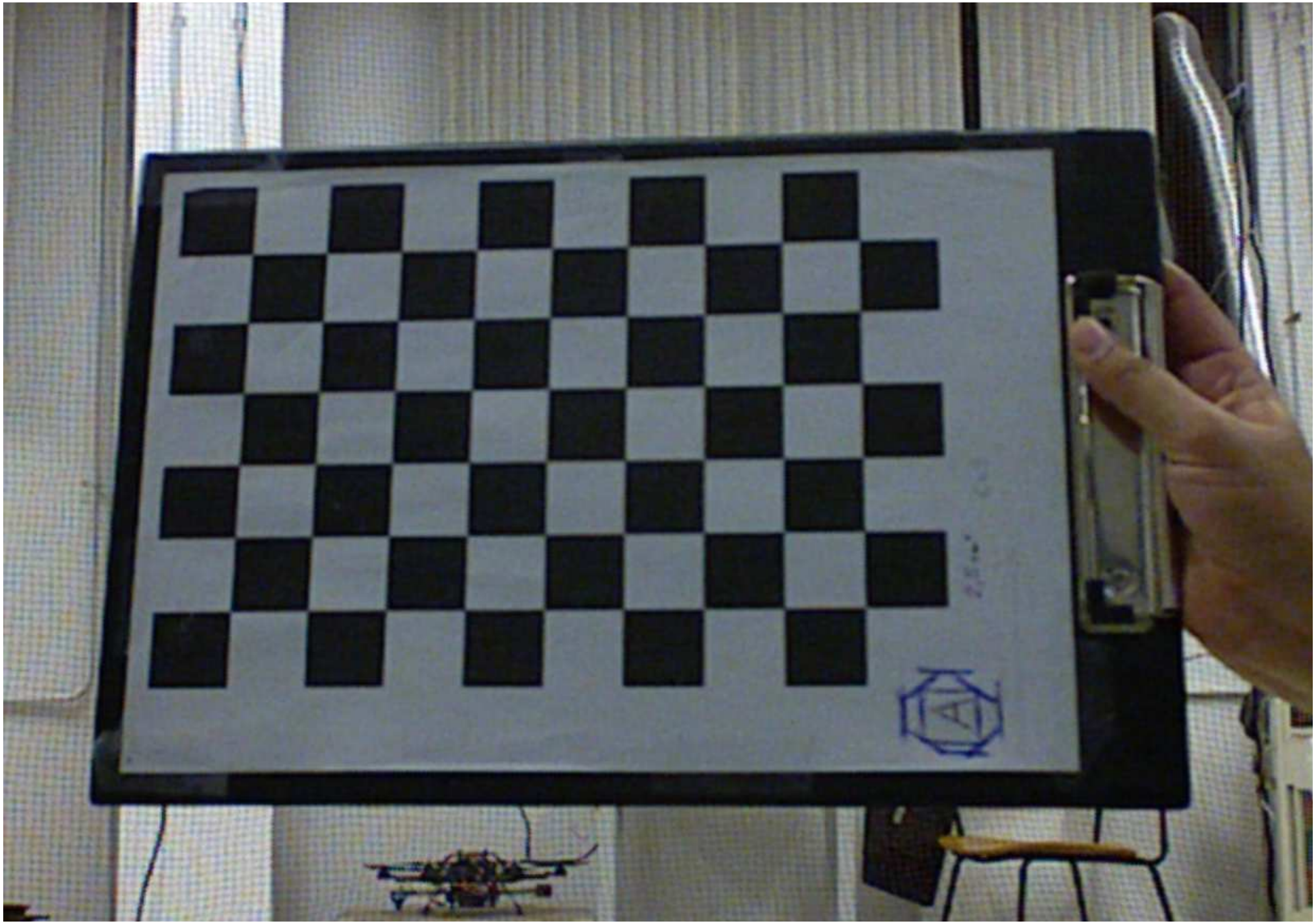
RGB camera calibration
Original image



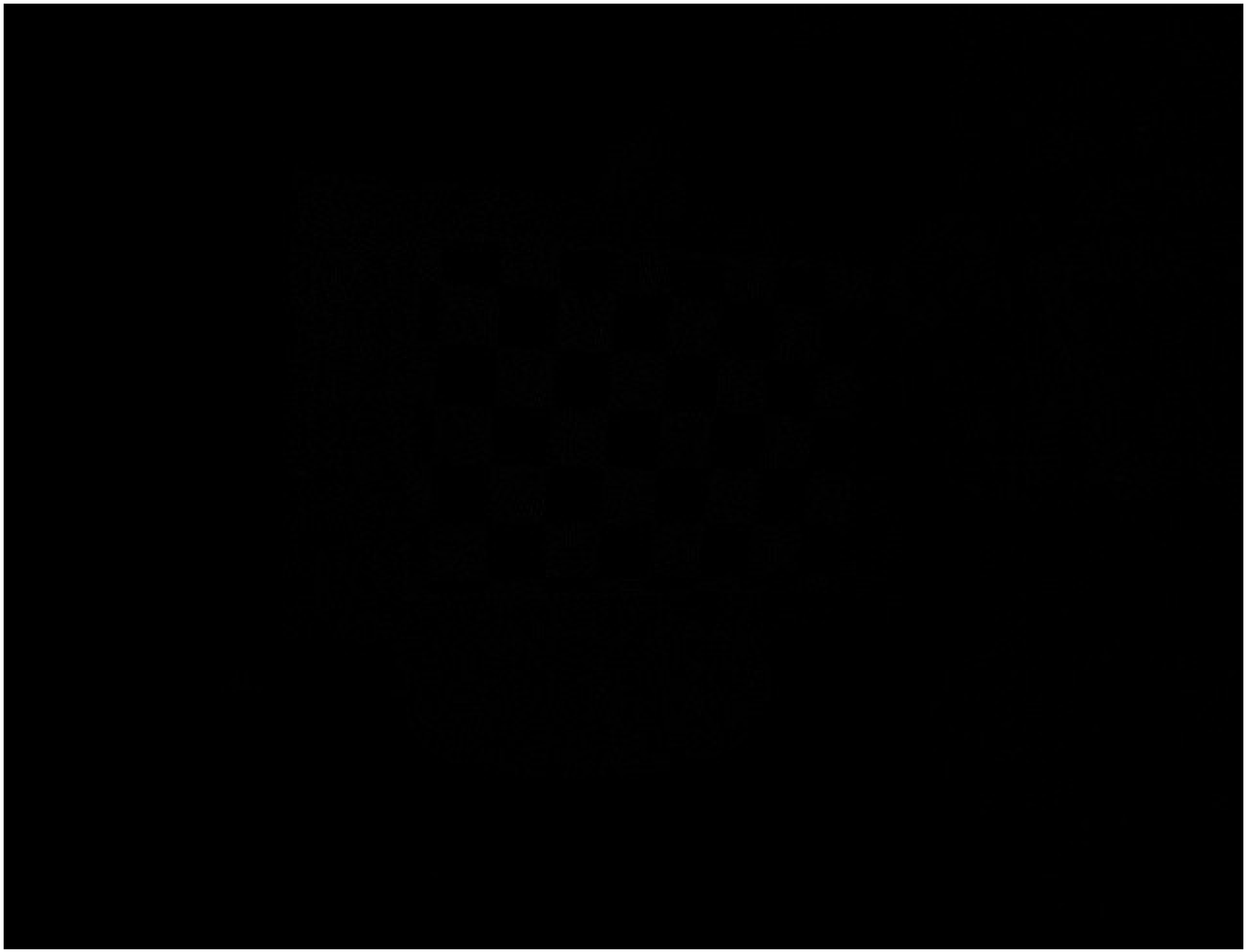
RGB camera calibration
Image with pattern recognized



RGB camera calibration
Undistorted image

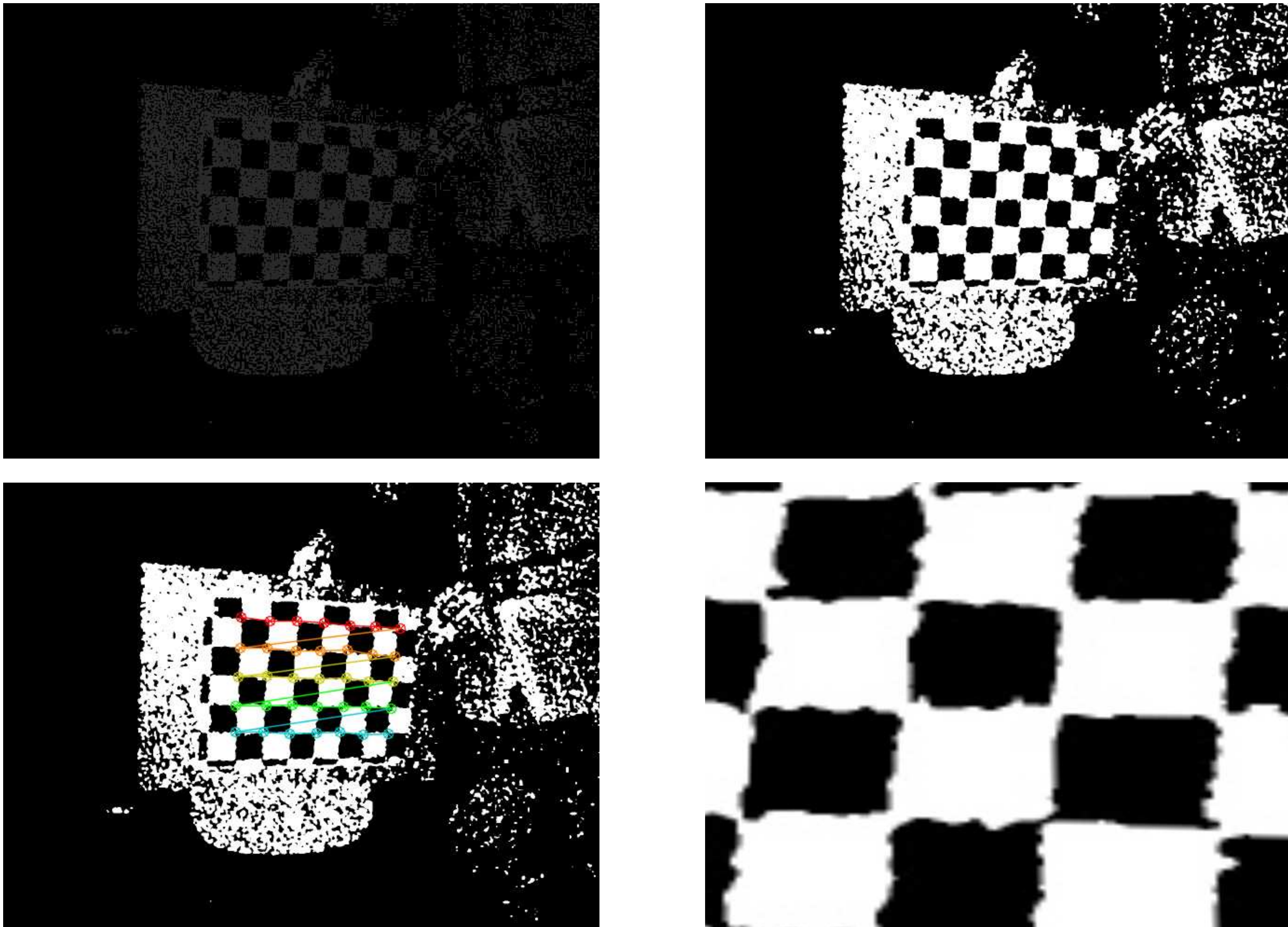


IR camera calibration

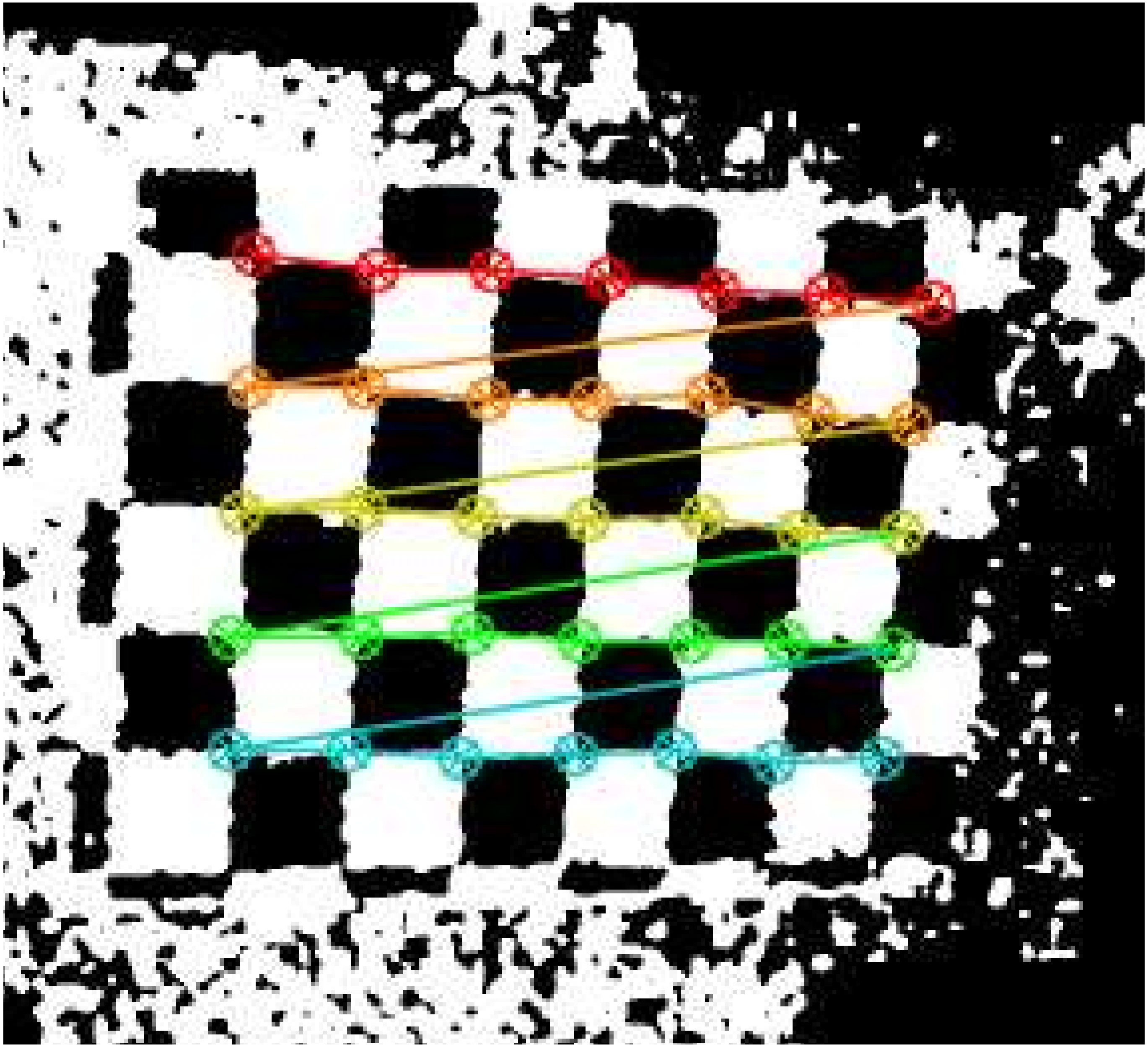


Raw IR image

IR camera calibration



IR camera calibration



Camera calibration script

- Algorithm and structure
- Summary results for RGB and IR calibration
- Commandline help and examples

Camera calibration script
Algorithm and structure

- Parse commandline arguments and set default values
- Load images
  - Image processing
    - Default filtering method
    - Binary filtering
    - Otsu filtering
  - Pattern recognition
  - Calibration parameters update
  - Save distorted images
- Save undistorted images
- Show summary
- Invoke plots generation

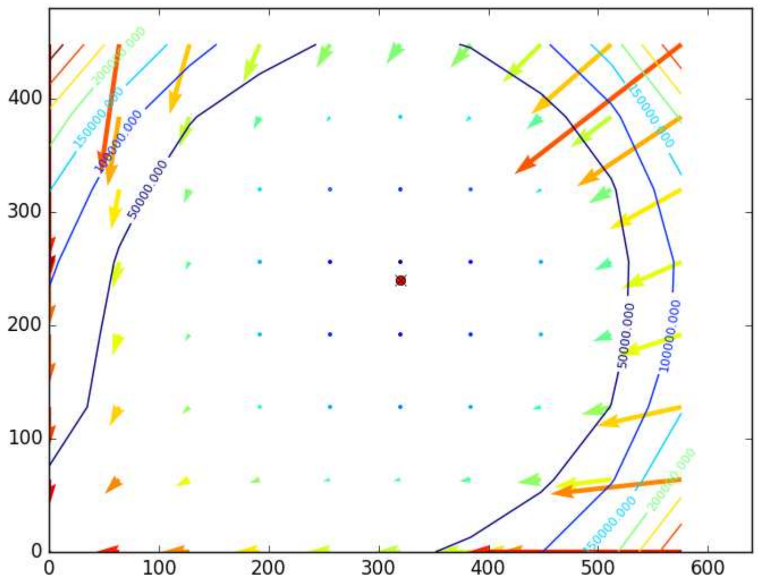
Camera calibration script
RGB calibration summary

```
Summary
Processed images: 189
Pattern recognition failure/Read files error: 2
Total image number: 191
Recognition successrate: 99.0%

Default method
Root mean square: 0.3161884375136998
Camera matrix:
[[  1.22043450e+03   0.00000000e+00   6.53170990e+02]
 [  0.00000000e+00   1.22557748e+03   4.81508974e+02]
 [  0.00000000e+00   0.00000000e+00   1.00000000e+00]]
Distortion coefficients: [  5.14525924e-01  -4.48093497e+00  -1.37059575e-02  -1.25218825e-03
   1.88941359e+01]
Radial distortion coefficients: [  0.51452592  -4.48093497  18.89413591]
Tangential distortion coefficients: [-0.01370596 -0.00125219]
Rotation vector: [array([[-0.23421558],
       [-0.09779963],
       [-0.04997646]])]
Translation vector: [array([[-111.29530548],
       [ -36.91200595],
       [ 438.99391301]])]
Focal length: f_x = 1220.43449812, f_y = 1225.57748393
Principal point: P_x = 640.0, P_y = 512.0
Optical center point: O_x = 653.170990322, O_y = 481.508973956
```

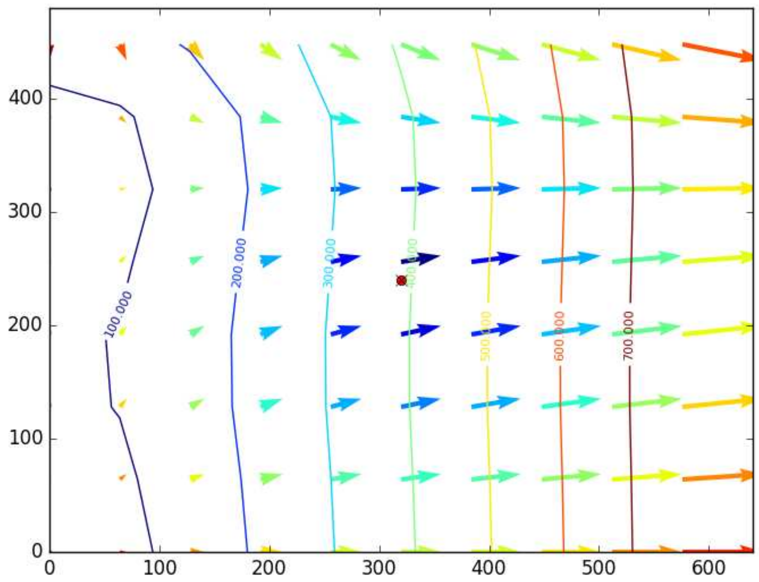
Camera calibration script
RGB calibration summary

```
Skew: S = [0.0]

Binary filtration
Root mean square:0.31714954236374265
Camera matrix:
[[  1.22409371e+03   0.00000000e+00   6.53180975e+02]
 [  0.00000000e+00   1.22926745e+03   4.80356095e+02]
 [  0.00000000e+00   0.00000000e+00   1.00000000e+00]]
Distortion coefficients: [  5.15577363e-01  -4.50560343e+00  -1.40264954e-02  -1.31189239e-03
   1.92077473e+01]

Otsu filtration
Root mean square: 1.5426388499070194
Camera matrix:
[[  1.10564550e+03   0.00000000e+00   6.52674126e+02]
 [  0.00000000e+00   1.11958032e+03   5.10104134e+02]
 [  0.00000000e+00   0.00000000e+00   1.00000000e+00]]
Distortion coefficients: [  4.83544777e+00  -2.74730680e+02   2.25659032e-02   1.86982849e-02
   4.67382928e+03]


Plots:

Generating plots
Exit
```

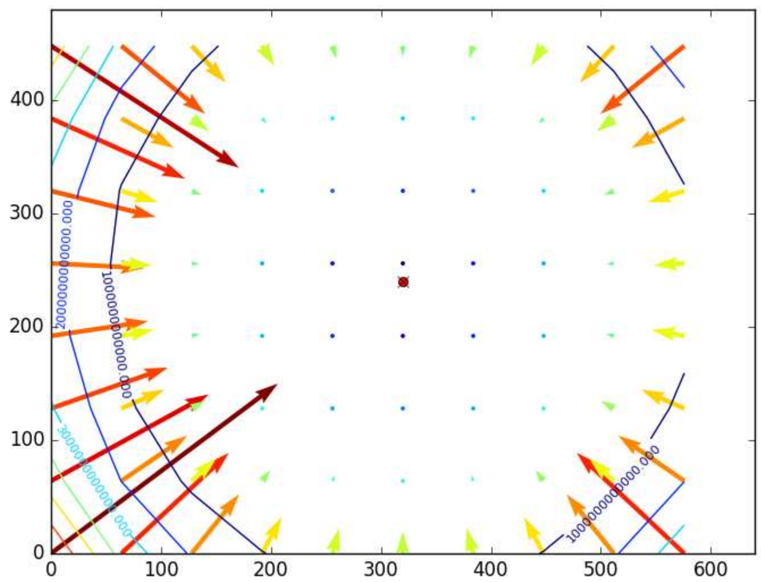
RGB camera distortions



Radial



Tangential



Complete

Camera calibration script
IR calibration summary

Summary
Processed images: 42
Pattern recognition failure/Read files error: 53
Total image number: 95
Recognition successrate: 44.0%

Default method
Root mean square: 1.2269150908833022
Camera matrix:

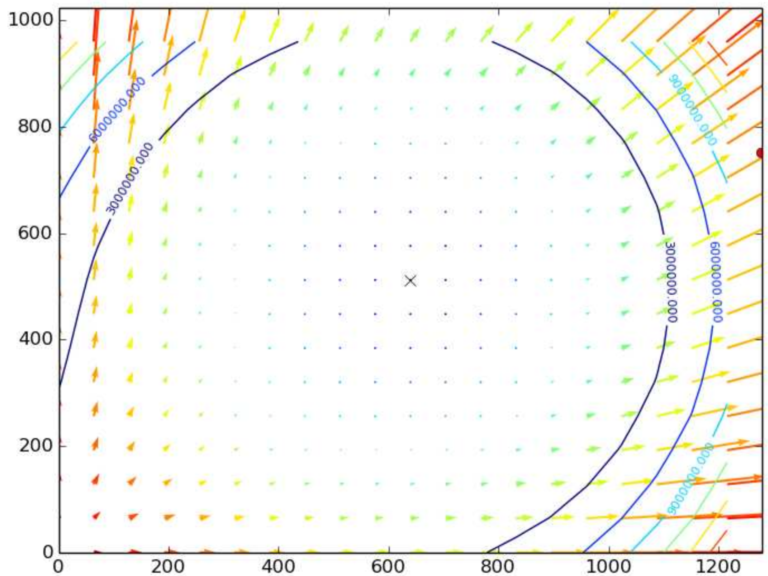
```
[[ 783.28651435    0.          348.25093001]
 [   0.          733.4864873   208.18289971]
 [   0.            0.            1.        ]]
```

Distortion coefficients: [-5.39955121e+00   1.94384459e+02   8.08030446e-02 -2.16378818e-02
  -2.46675157e+03]
Radial distortion coefficients: [-5.39955121   194.38445877 -2466.75157186]
Tangential distortion coefficients: [0.08080304 -0.02163788]
Rotation vector: [array([[-0.33153994],
       [-0.42348496],
       [-0.05135867]])]
Translation vector: [array([[-163.25342614],
       [-12.43951787],
       [1110.61126797]])]
Focal length: f_x = 783.286514346, f_y = 733.4864873
Principal point: P_x = 640.0, P_y = 512.0
Optical center point: O_x = 348.250930007, O_y = 208.182899706

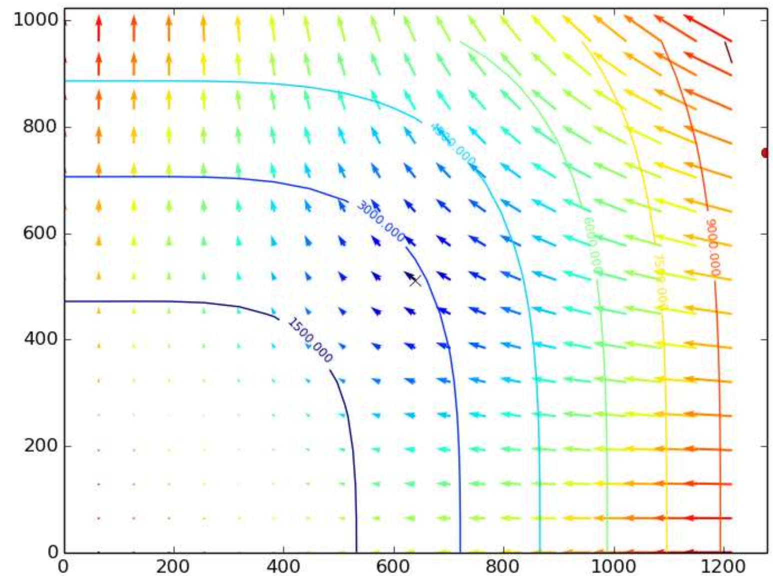
Camera calibration script
IR calibration summary

```
Skew: S = [0.0]

Binary filtration
Root mean square:1.2415525723371863
Camera matrix:
[[ 801.99989476    0.          348.98903294]
 [   0.          760.71626928  203.8613809 ]
 [   0.            0.            1.        ]]
Distortion coefficients: [ -5.56742321e+00   2.02474869e+02   8.85581401e-02  -3.25646526e-02
  -2.64242303e+03]

Otsu filtration
Root mean square: 1.5025754901934407
Camera matrix:
[[ 476.43284444    0.          339.40023596]
 [   0.          494.72448832  290.47636817]
 [   0.            0.            1.        ]]
Distortion coefficients: [  3.80005325e+00  -9.96798146e+01  -3.34833961e-02   8.63111064e-03
   7.32924984e+02]


Plots:

Generating plots
Exit
```

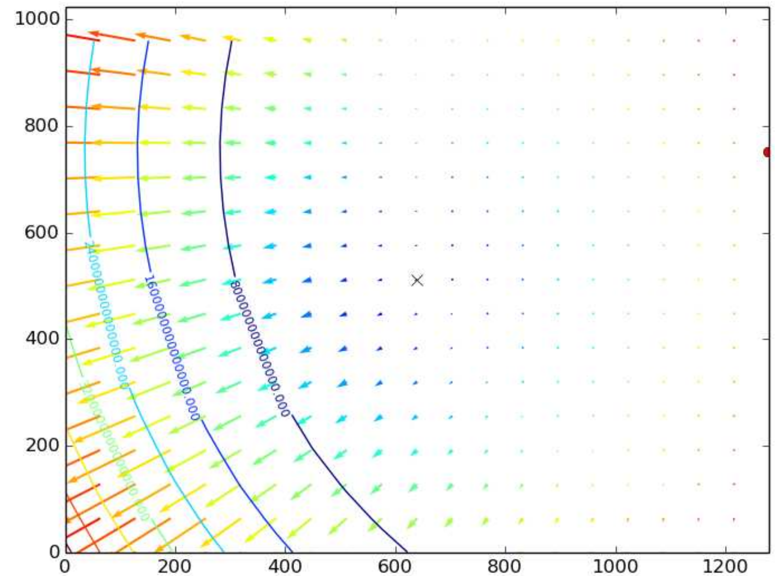
IR Camera Distortions



Radial



Tangential



Complete

Camera calibration script Commandline help

```
usage: camera-calibration.py [-h] [-1] [-bt BINARYTHRESHOLD <0-255><!127>]
                             [-br BLURRANGE <in pixel><!5>]
                             [-cr CAMERARESOLUTION <X resolution, Y resolution><!1280, 1024> CAMERARESOLUTION <X resolution, Y resolution><!1280, 1024>]
                             [-l LOGFILENAME]
                             [-ps PATTERNSIZE <x, y><!9, 6> PATTERNSIZE <x, y><!9, 6>]
                             [-si] [-s] [-ss SQUARESIZE <in mm><!25.0>]
                             [-gp {0,1,2}] [-ia INVERTAXES]
                             [-sp <SAVEPATH><!/tmp>] [-sr3]
                             [IMAGEFILES, FILEMASK) [(IMAGEFILES, FILEMASK ...]]

Calculate camera calibration matrix and additional parameters based on a
provided sequence of images with a chessboard calibration pattern.

positional arguments:
  (IMAGEFILES, FILEMASK)
                        Images globbing mask

optional arguments:
  -h, --help            show this help message and exit
  -1, -p, --pause       Pause between processed images
  -bt BINARYTHRESHOLD <0-255><!127>, --binary-threshold BINARYTHRESHOLD <0-255><!127>
                        Threshold for binary filtering
```

Camera calibration script Commandline help

```
  -br BLURRANGE <in pixel><!5>, --blur-range BLURRANGE <in pixel><!5>
                        Range for blur filter during Otsu filtering
  -cr CAMERARESOLUTION <X resolution, Y resolution><!1280, 1024> CAMERARESOLUTION <X resolution, Y resolution><!1280, 1024>,
--camera-resolution CAMERARESOLUTION <X resolution, Y resolution><!1280, 1024> CAMERARESOLUTION <X resolution, Y
resolution><!1280, 1024>
                        Pixel resolution of the camera used
  -l LOGFILENAME, --log LOGFILENAME
                        Write a log file with results
  -ps PATTERNSIZE <x, y><!9, 6> PATTERNSIZE <x, y><!9, 6>, --pattern-size PATTERNSIZE <x, y><!9, 6> PATTERNSIZE <x, y><!9, 6>
                        Number of corners in the outter pattern block
  -si, --show-images    Show images
  -s, --save-images     Save images
  -ss SQUARESIZE <in mm><!25.0>, --square-size SQUARESIZE <in mm><!25.0>
                        Size of squares in millimeters
  -gp {0,1,2}, --generate-plots {0,1,2}
                        Generate and/or colorize distortion plots: 0 -
                        disable, 1 - color, 2 - grayscale
  -ia INVERTAXES, --invert-axes INVERTAXES
                        Invert axes on distortion plots
  -sp <SAVEPATH><!/tmp>, --save-path <SAVEPATH><!/tmp>
                        Definitions path for generated images, implies save
                        images option
  -sr3, --skip-rc3      Ignore the third radial distortion parameter during
                        calculations
```

Camera calibration script Usage example

```
#!/bin/bash

python camera-calibration.py -sr3 -sp "latest-rgb/cal" -l results.log -ss 25 -ps 9 6 -cr 1280 1024 "latest-rgb/raw/"*

echo Done
```

Future work and Conclusion

- Implementation of developed calibration software into the ROS framework
- Offline and online processing
- Pseudo realtime feedback
- Streamlining calibration process

Thank you for your attention.