QUADCOPTER Huginn

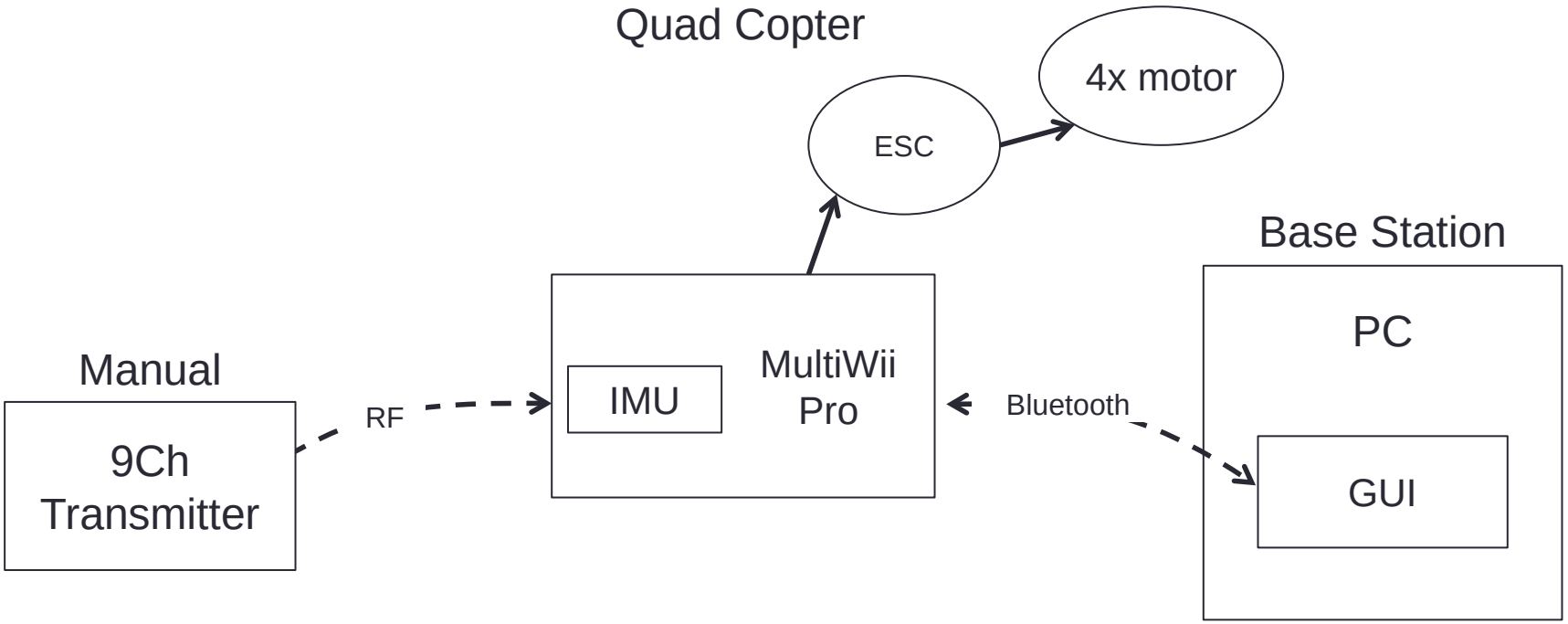
Group 1

- Floris Driessen
- Martin van Leussen
- Art Senders
- Tijl van Vliet
- Steven van der Vlugt

Presentation outlines

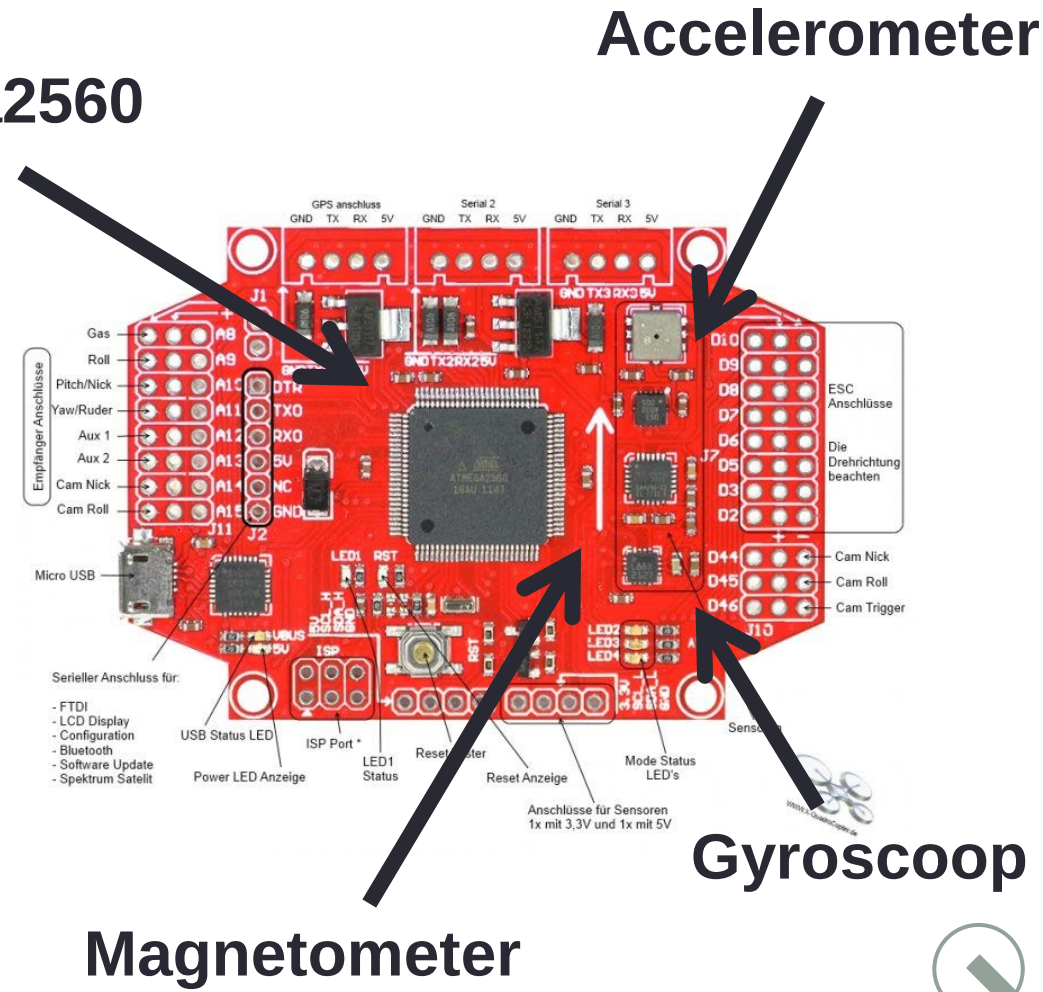
- Assignments and goals
  - A) Build the quadcopter and get it flying
  - B) PID tuning
  - C) Rotate 360 degrees around the z-axis
  - D) Marker detection
  - E) Environment map
- Results
- Recommendations

A) Hardware setup



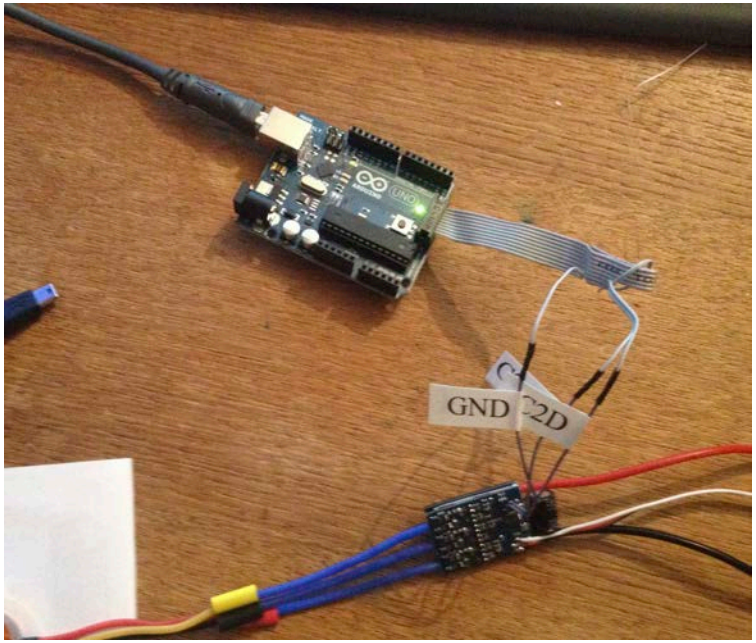
A) Multiwii

- Active community
- Open source
- Good quality
- Powerful controller

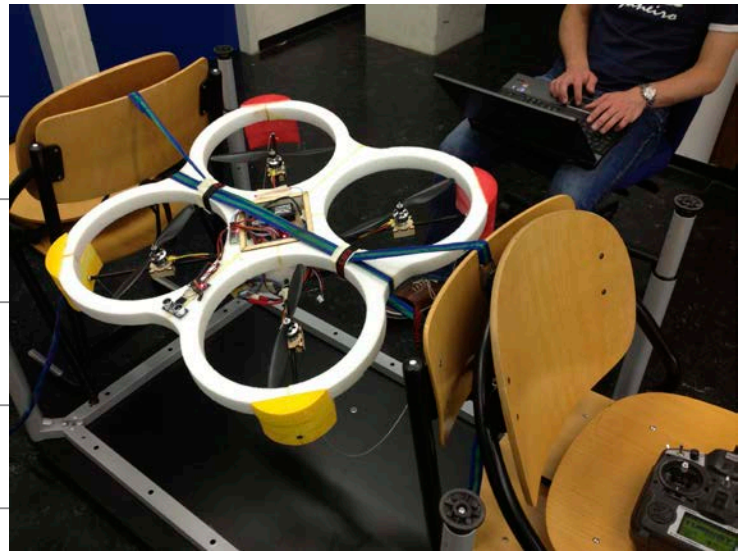


B) Stable Quadcopter

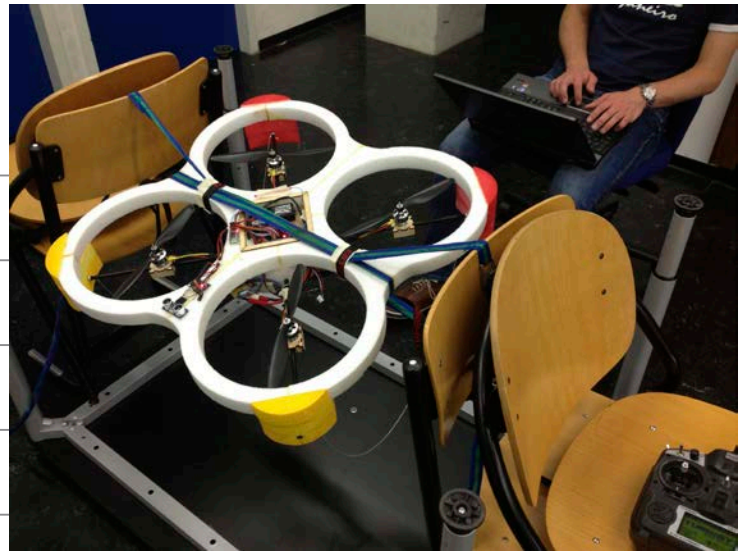
- Flash ESC with custom firmware
- PID tuning
  - Ziegler Nichols
  - Impulse response
  - Step response



B) Stable Quadcopter



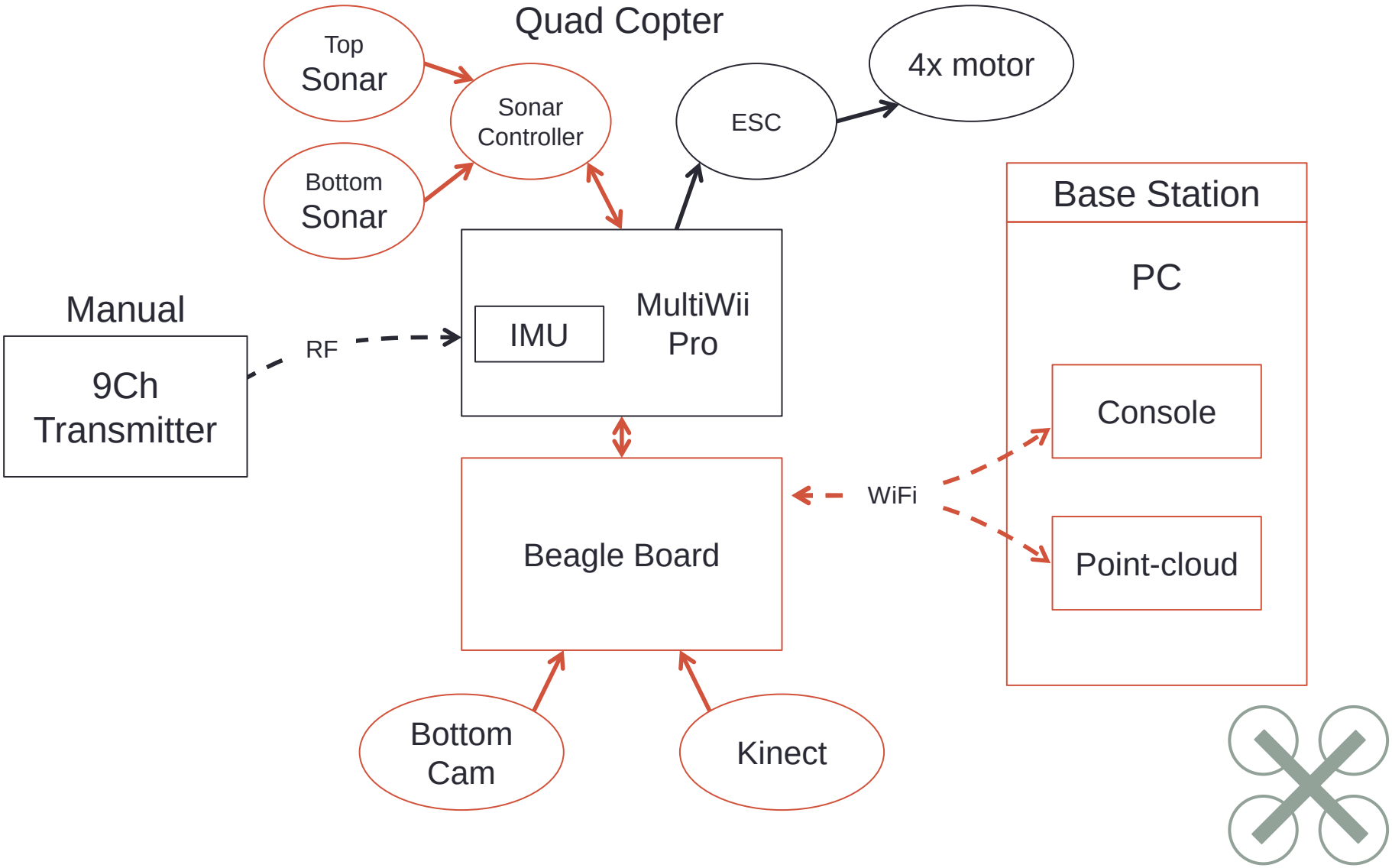
B) Stable Quadcopter



C) Rotate 360 degrees around the z-axis

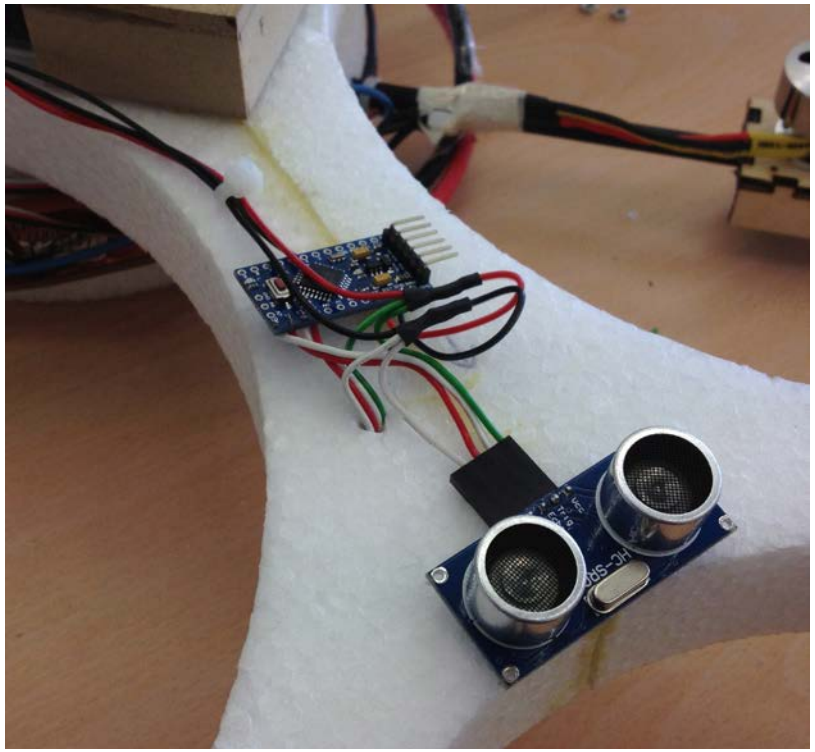
- Panorama
- Location hold
  - Fixed height
  - Fixed location

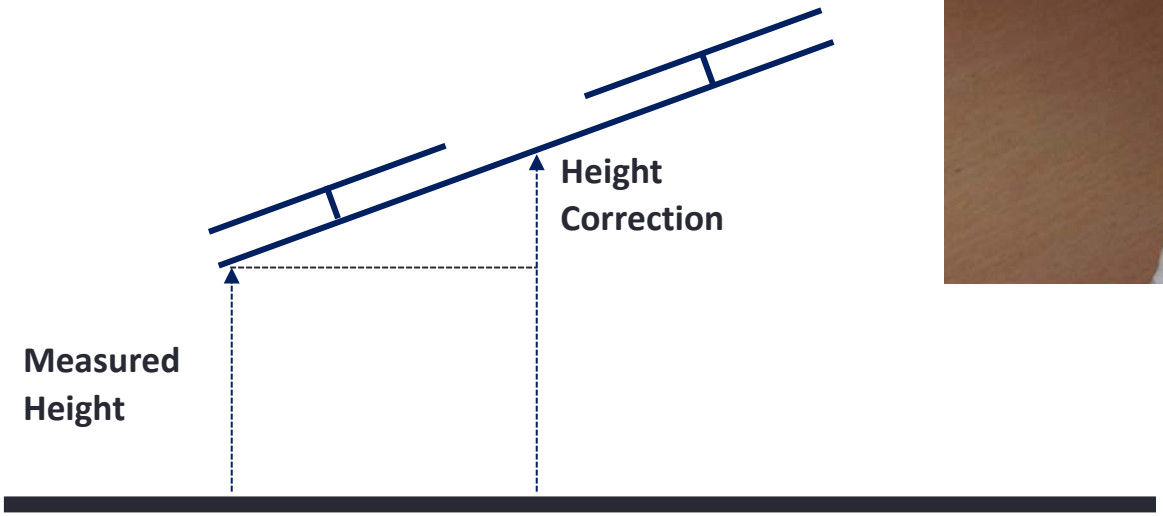
C) Hardware setup



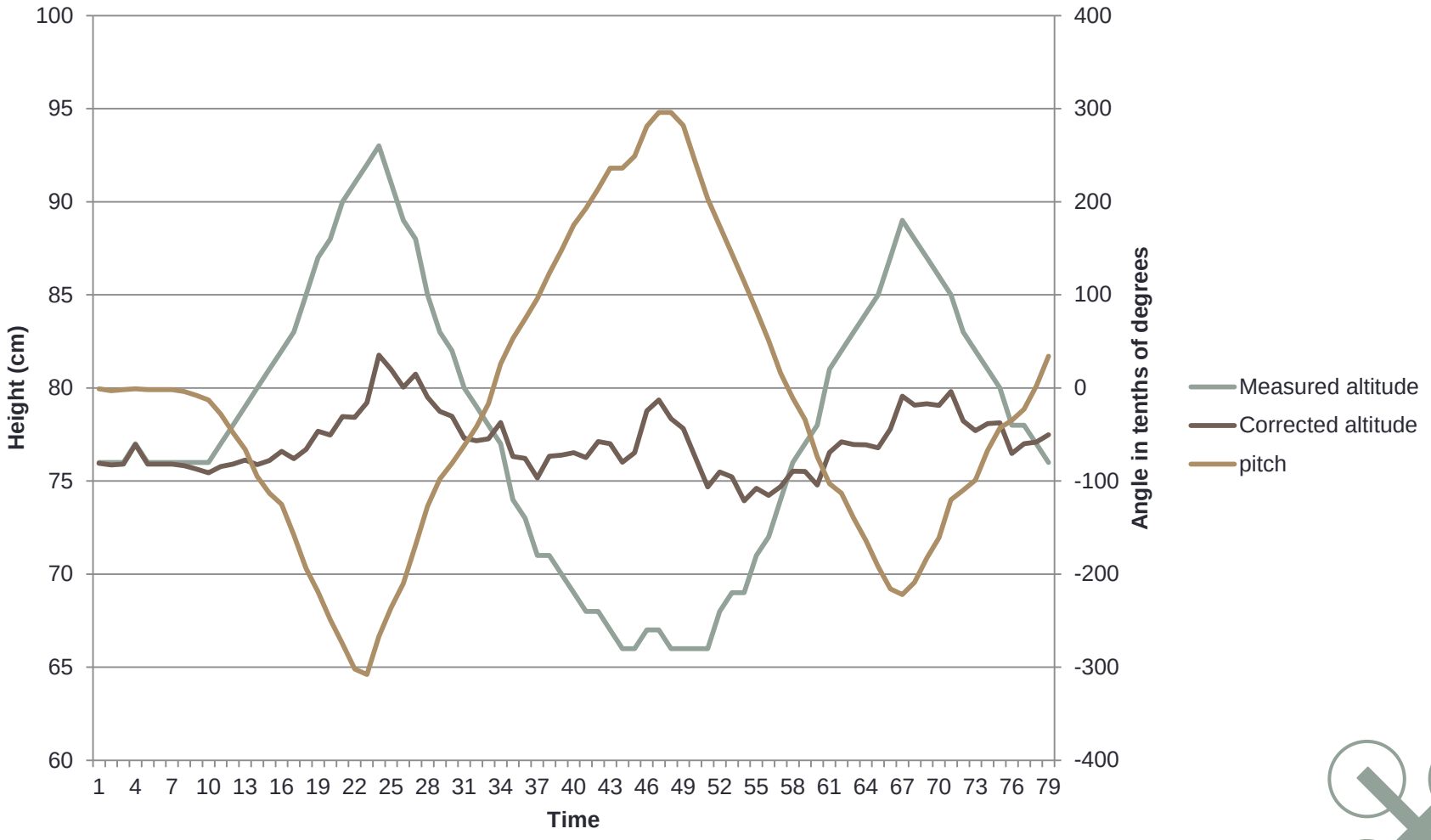
C) Height control

- Fix altitude
- Ultrasonic sensor
- Pitch correction



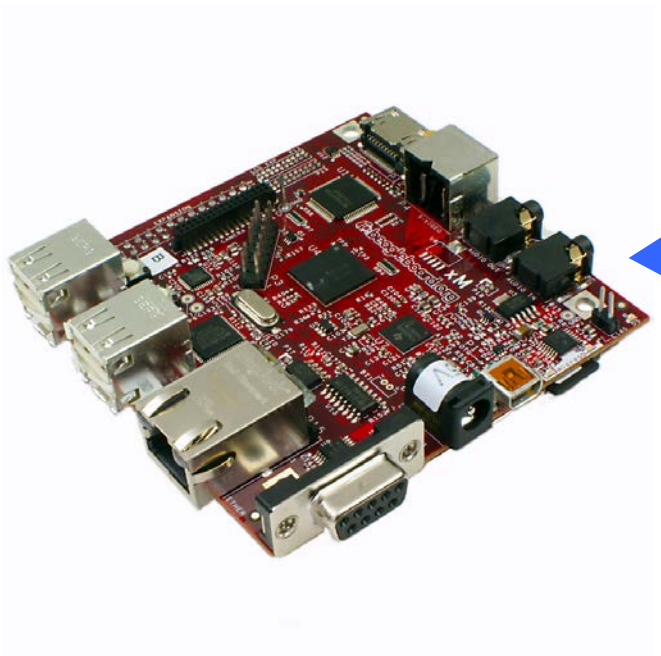


C) Height measurement

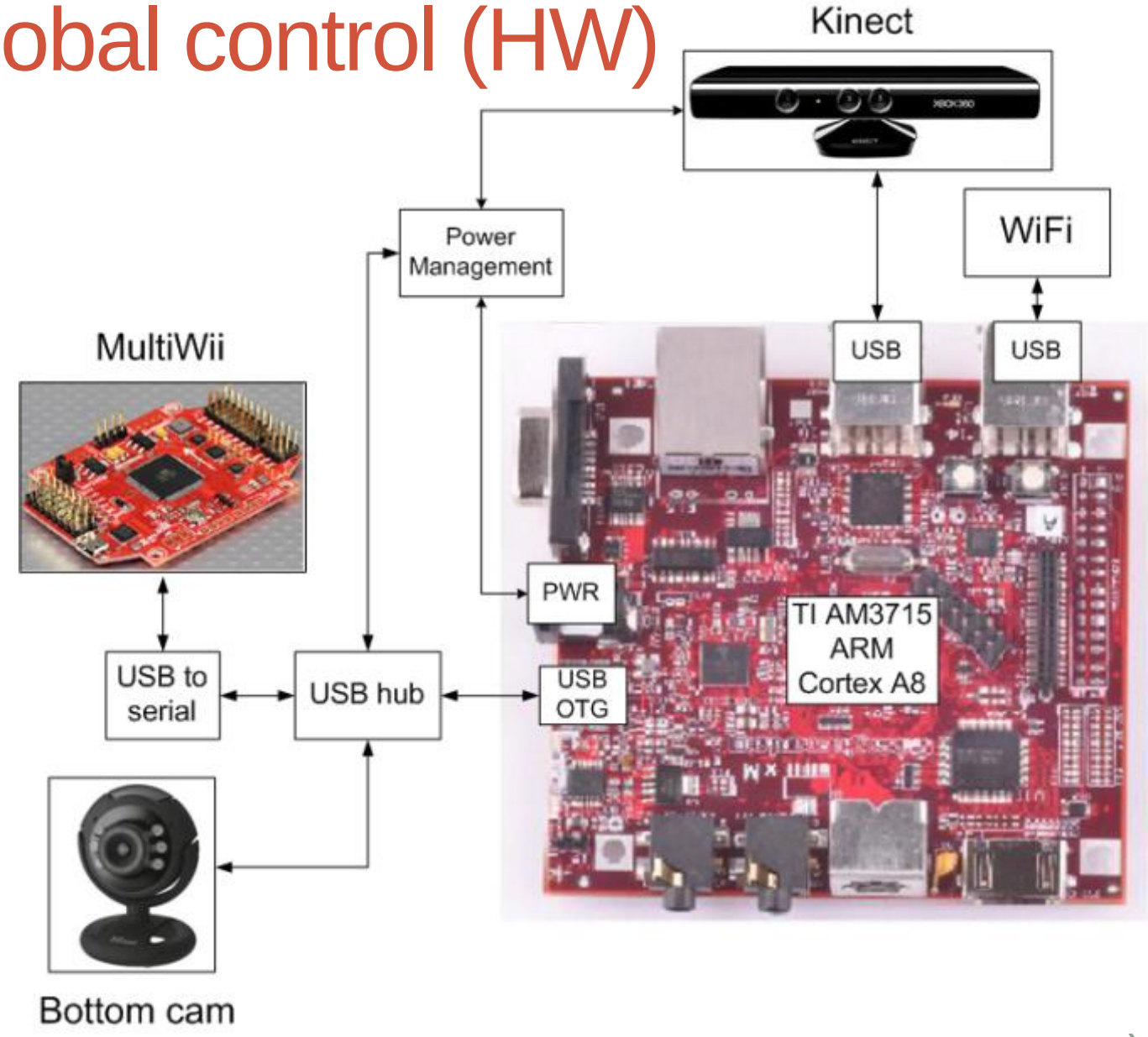


C) Control

- Start quad copter with RF transmitter
- Enable altitude hold
- Switch roll, pitch and yaw control to BeagleBoard

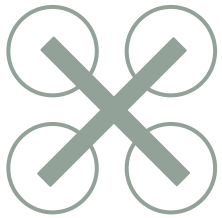


C) Global control (HW)



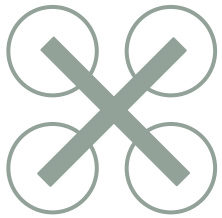
C) Global control (SW)

- Ubuntu
  - Cross platform
  - Experience and know how
- OpenCV
  - Powerful open source computer vision library by Intel
  - General purpose vision functions
  - Functions to work with video streams
- Open Kinect
  - Light weight open source library

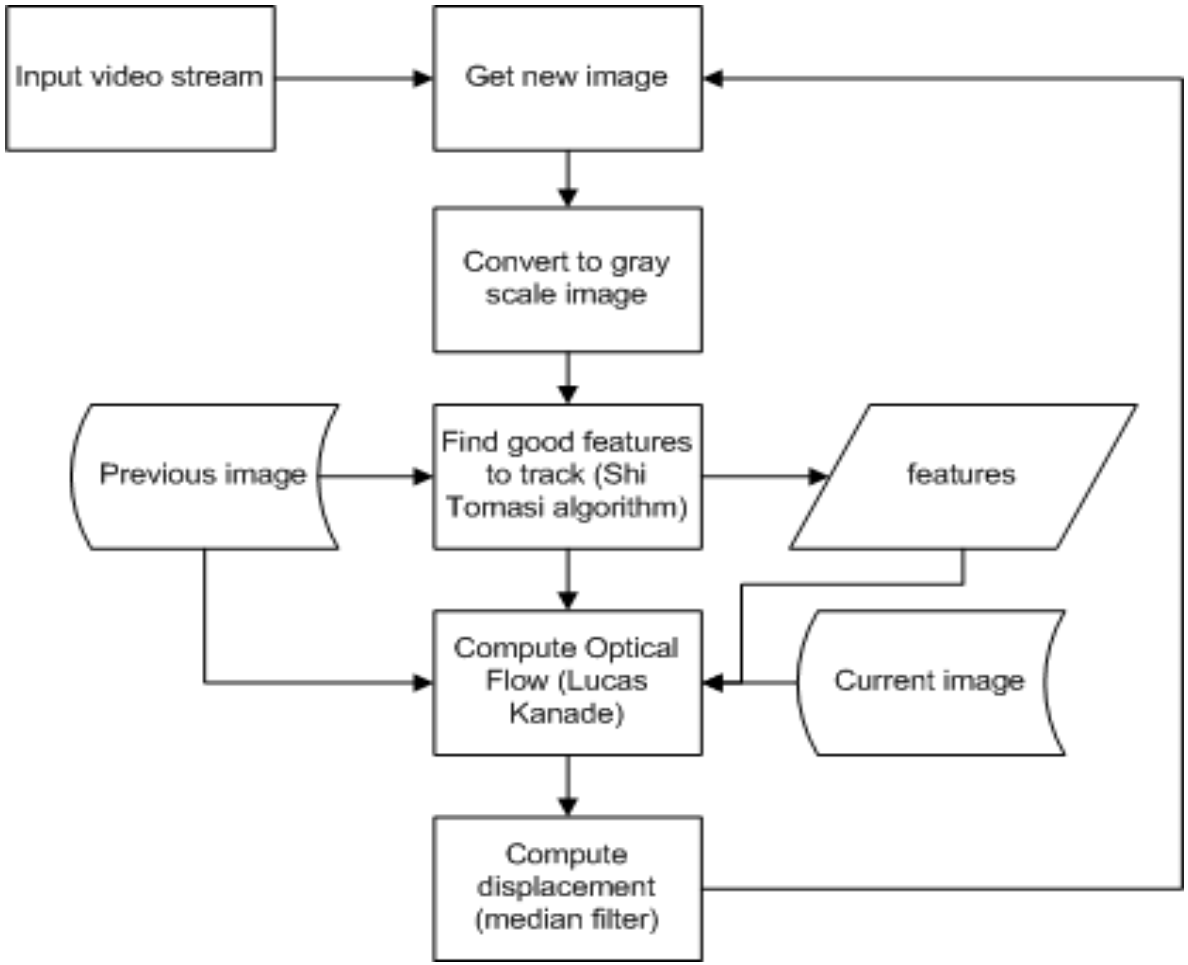


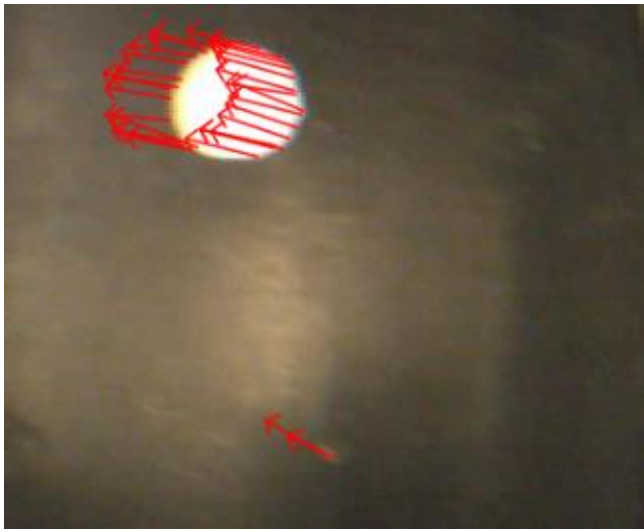
C) Fixed location

- Optical flow
  - Stabilization based on detected features
- Marker
  - Stabilization based on marker orientation

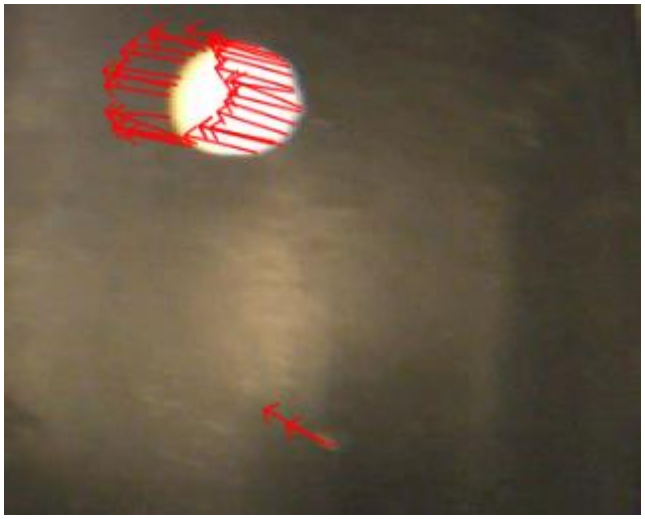


C) Optical Flow

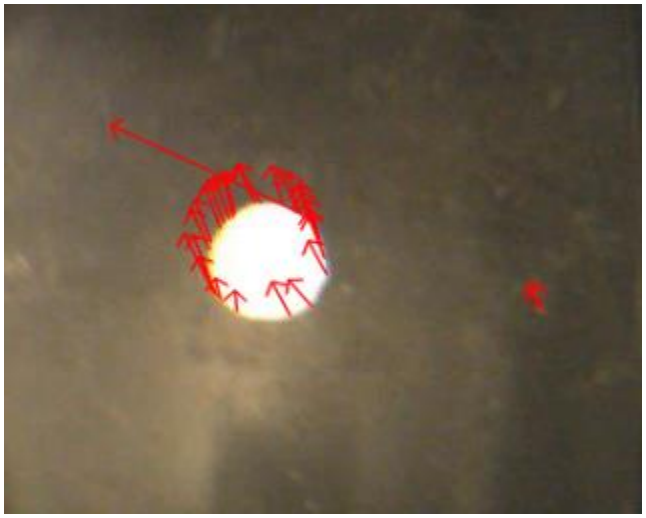




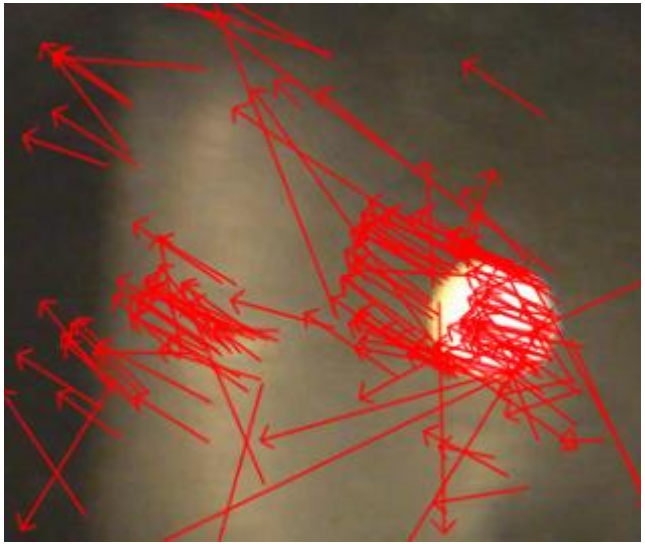
C) Optical Flow



50 strongest features



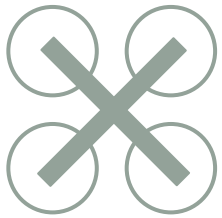
outlier



350 strongest features

C) Fixed location

- Optical flow
  - Requires trajectory planning to navigate the quadcopter
- Marker
  - Marker can be followed throughout a building

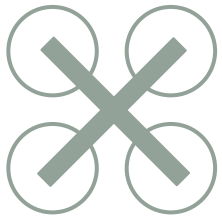


D) Marker detection

- Follow line instead need for trajectory planning
- Take a corner in a line

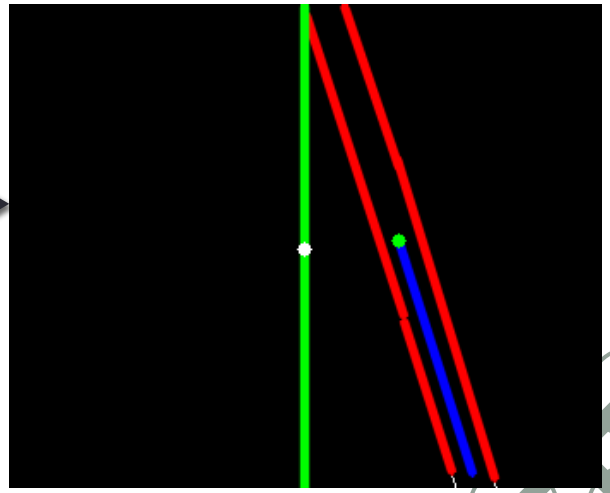
- Line detection
  - Houghline: vector extraction from edges
  - Classification: combine vectors to lines

- Marker identification



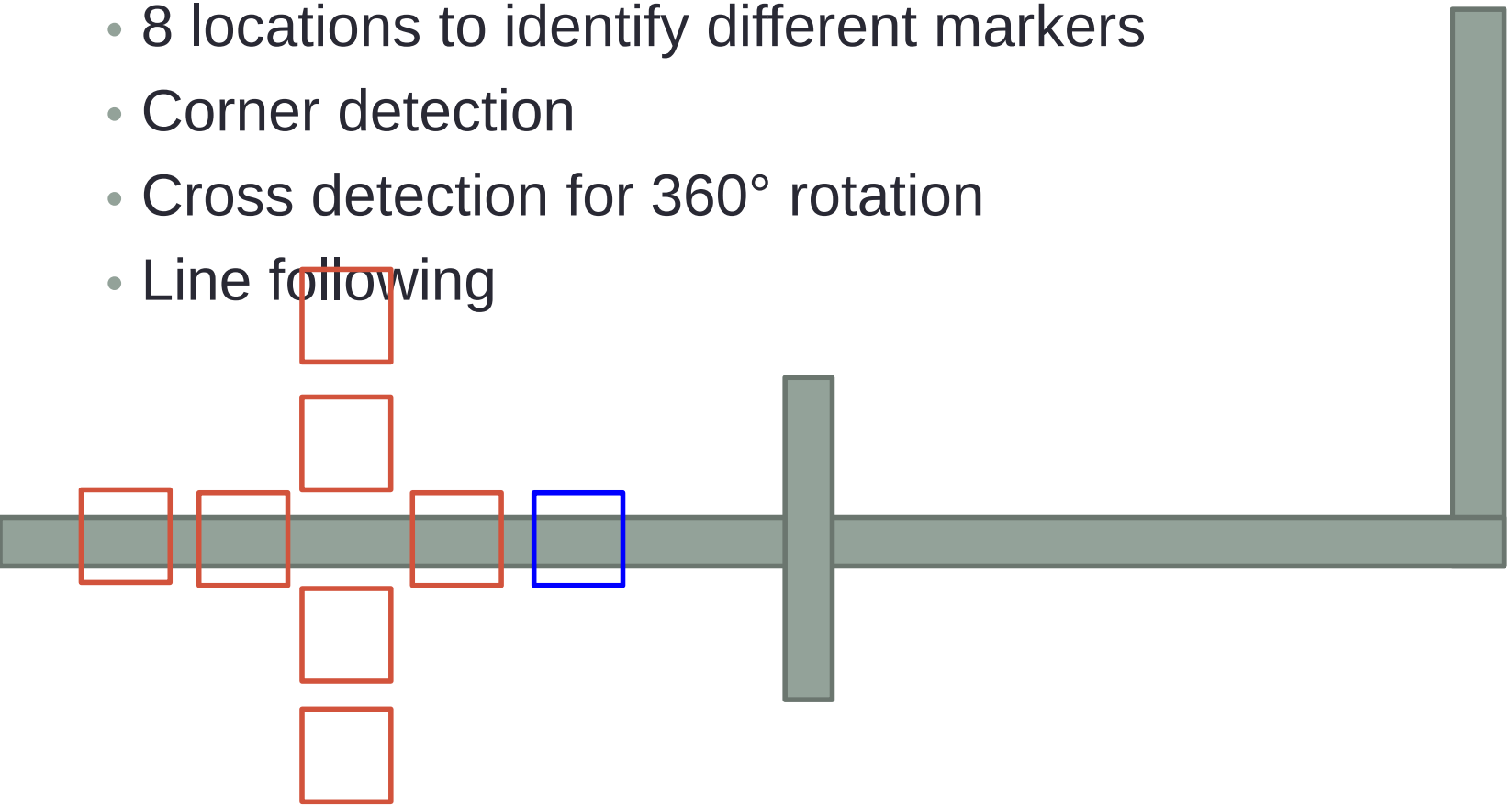
D) Image processing

1. Color image
2. Blur (to remove noise from processing)
3. Black and white image (not conventional)
4. Canny edge detection
5. Houghlines transform (vector output)
6. Vectors to lines (classification)
7. Detect crossing + type



D) Marker classification algorithm

- Map 8 fields next to the crossing of the lines
- 8 locations to identify different markers
- Corner detection
- Cross detection for 360° rotation
- Line following



D) Marker classification algorithm

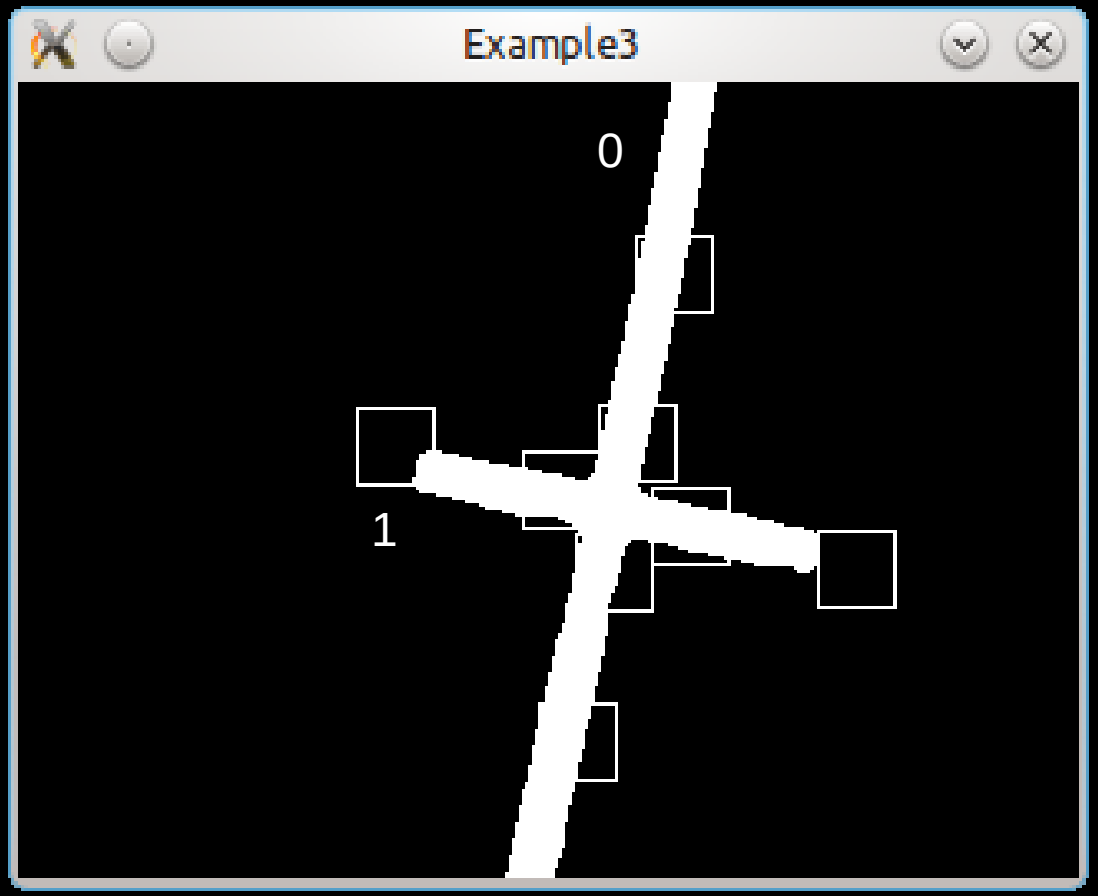
```
line_element = le_cross
value line0detected mean = f
value line1detected mean = 3
line0
angle=0.255714
n=6
line1
angle=-1.356325
n=2
fps : 1.04

line_element = le_cross
value line0detected mean = f
value line1detected mean = 3
line0
angle=0.257526
n=5
line1
angle=-1.369479
n=2
fps : 0.30
```

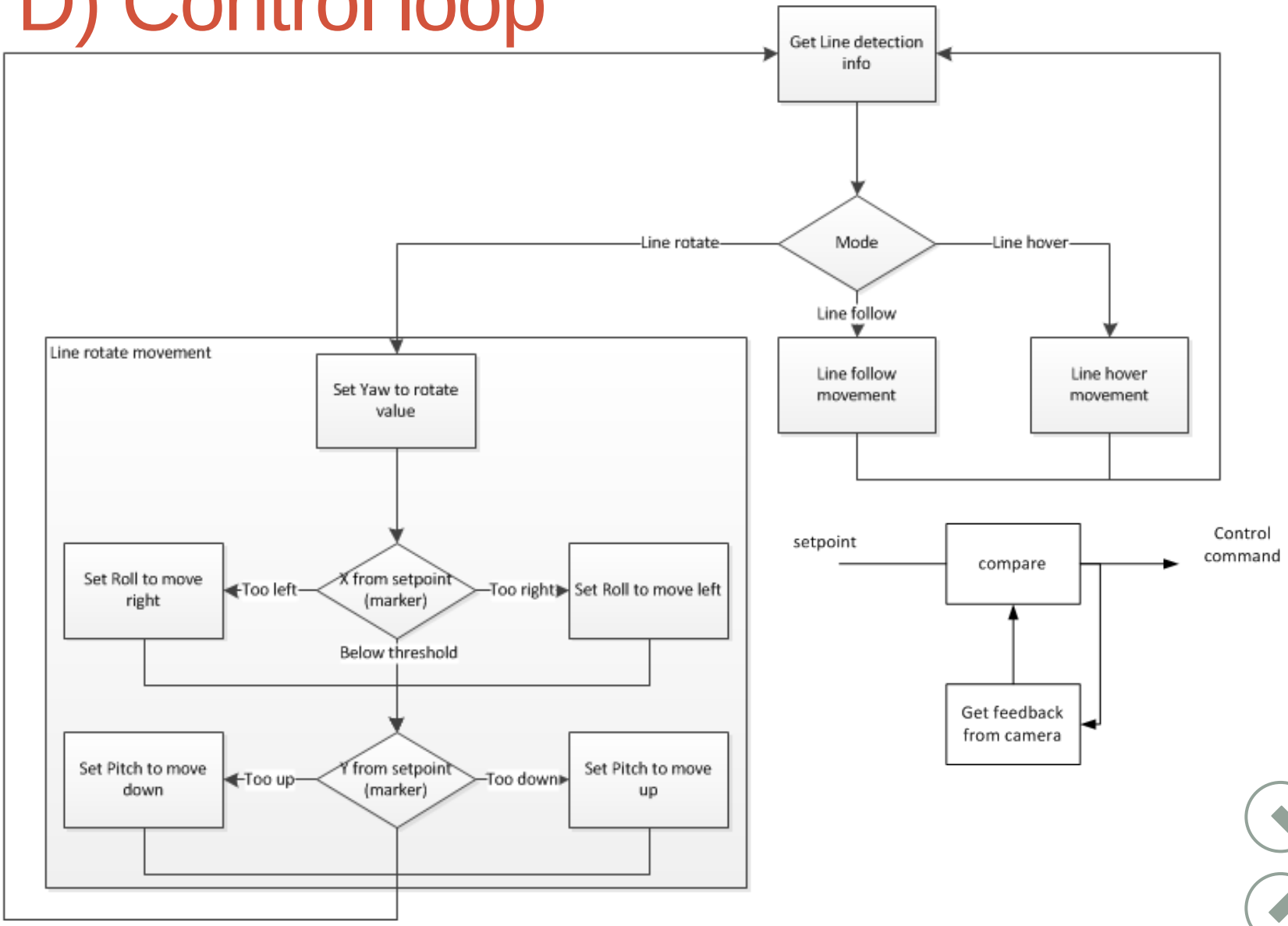
Bitfields of squared where a line is detected

Angles of lines detected

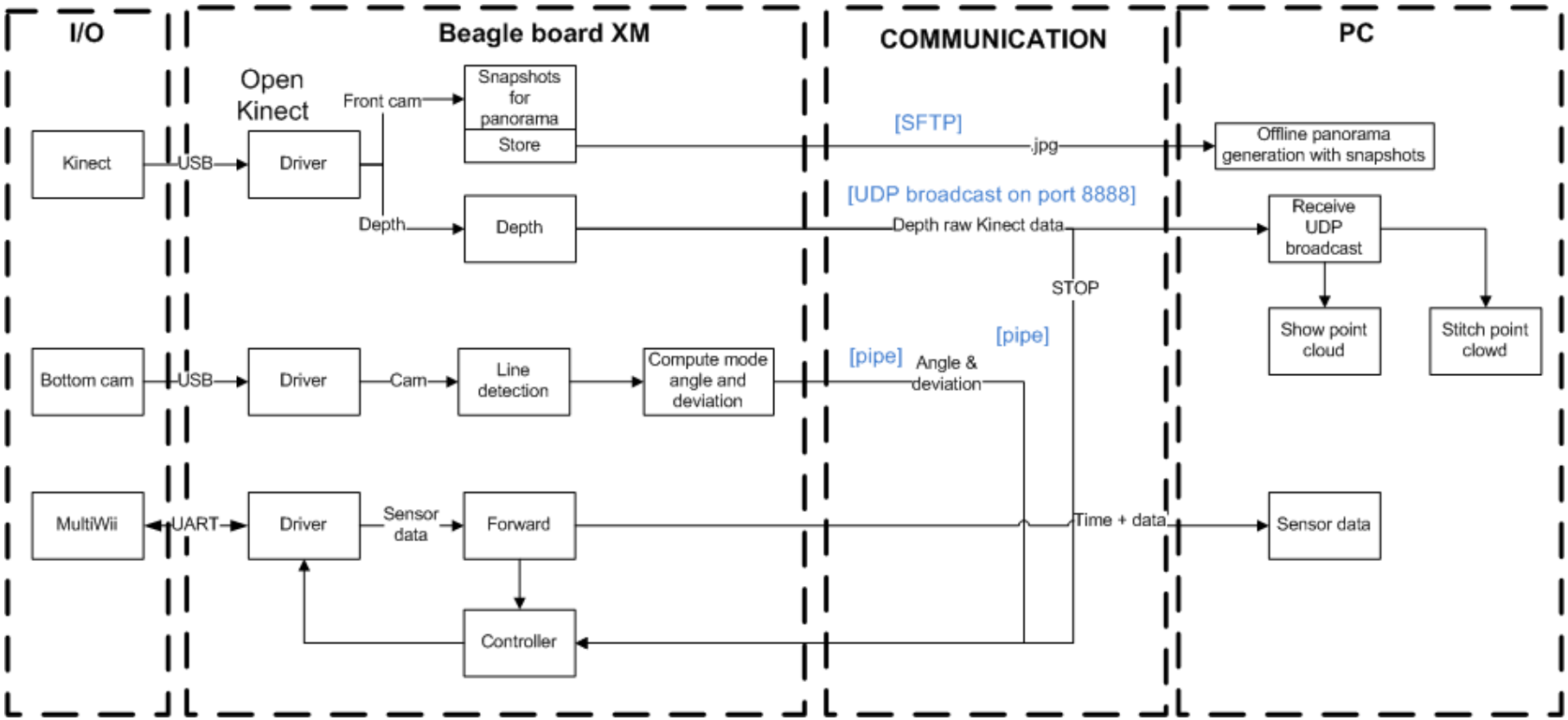
Low fps due to manual debugging



D) Control loop

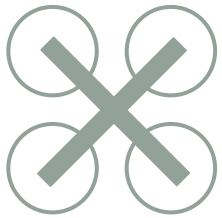


D) Global control (SW)



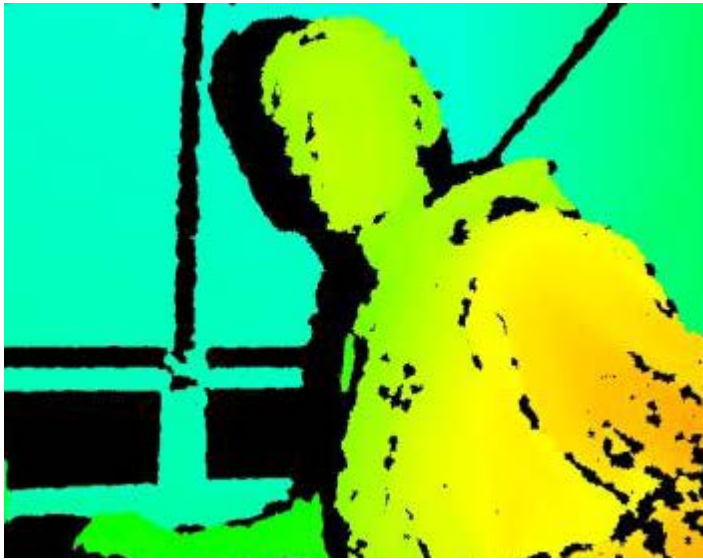
E) Environment mapping

- Virtual 3d map of the environment
- Navigation purposes (Autonomous navigation)
- Search and rescue
- Digitalization of buildings

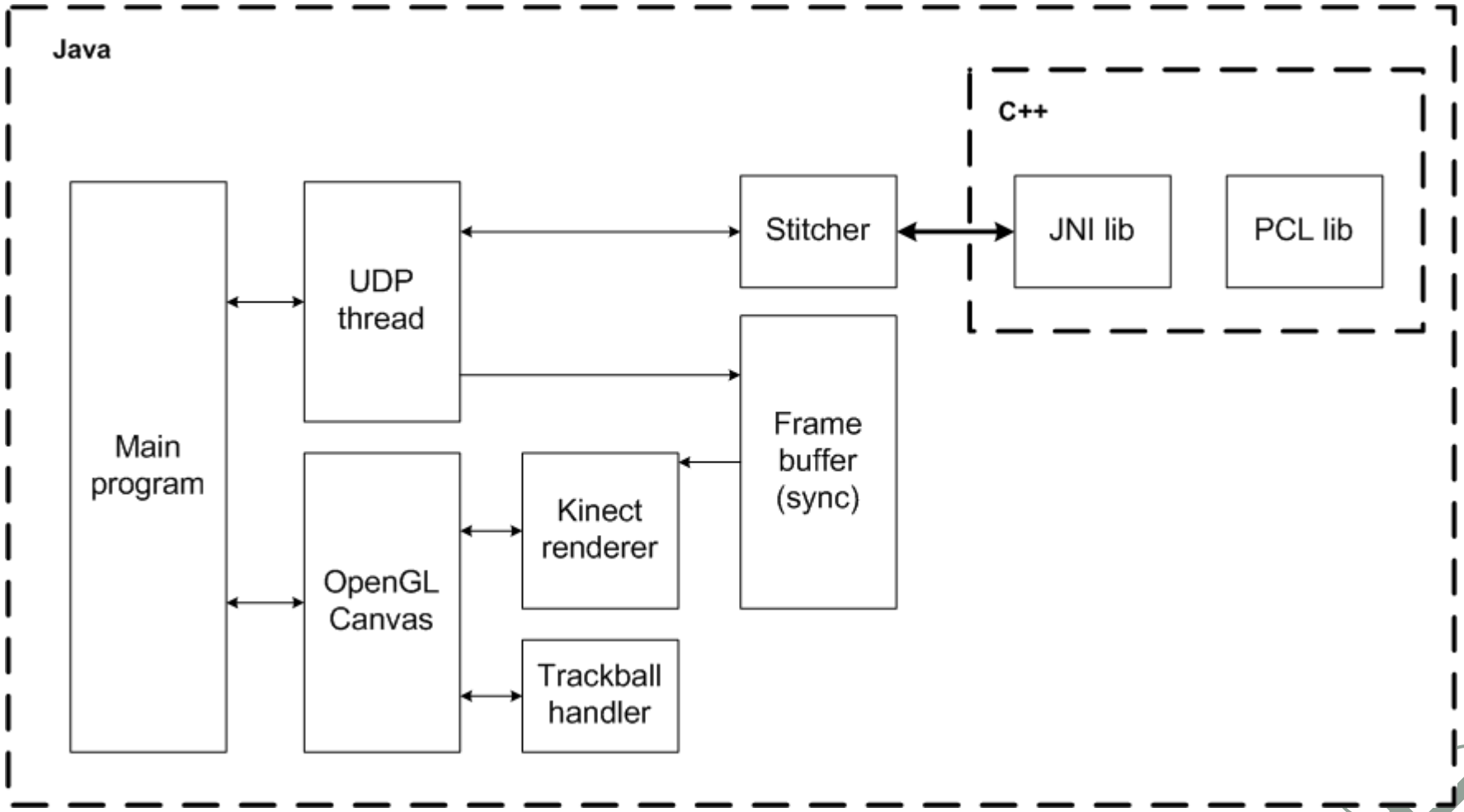


E) Point cloud

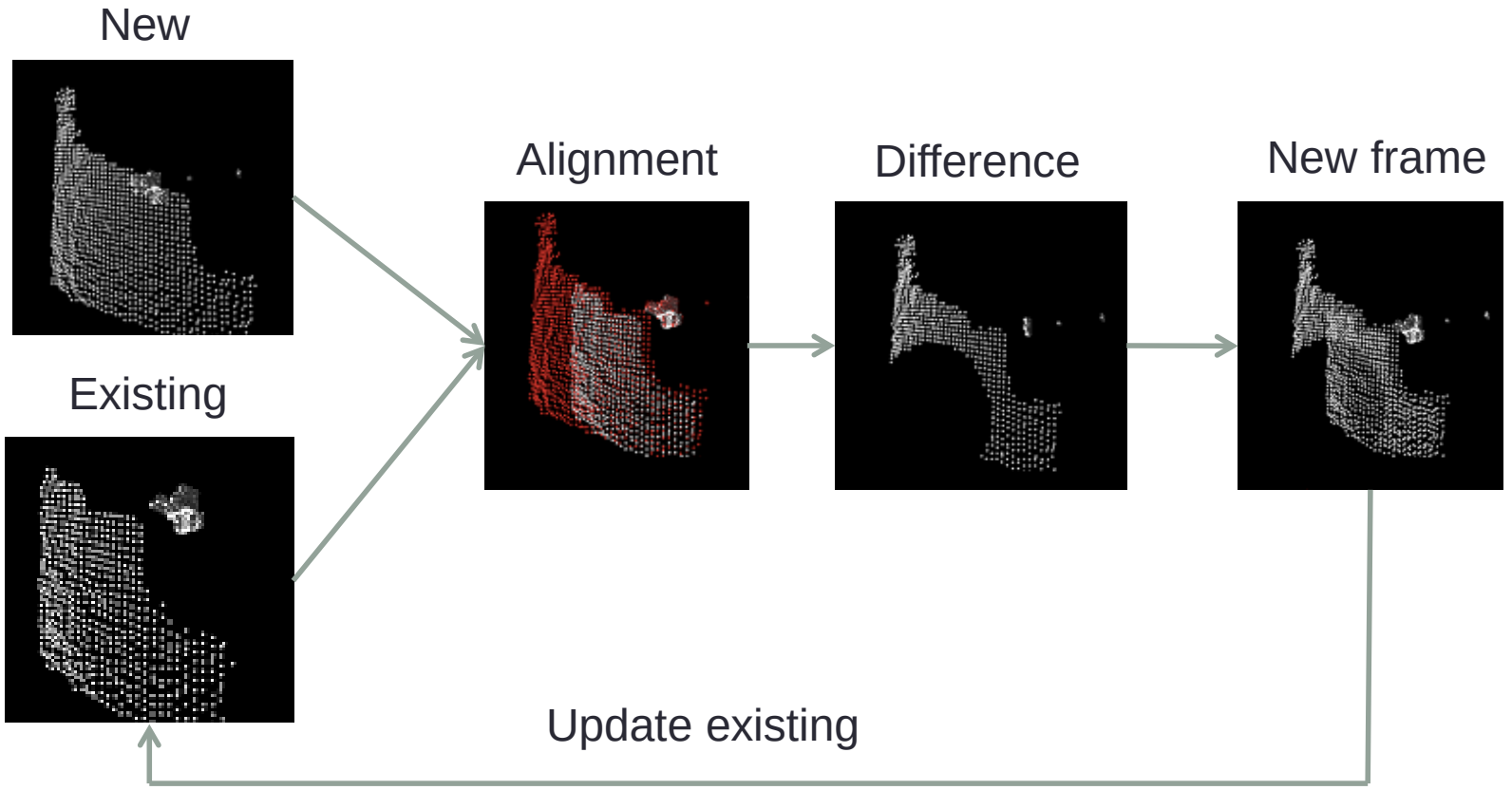
- Kinect Depth image
- UDP Stream with depth image from Beagle Board
- Use PCL (Piont Cloud Library) for stitching
- Mixture Java and C++
- JNI for java to C++ communication



E) Point cloud program



E) Stitching phases



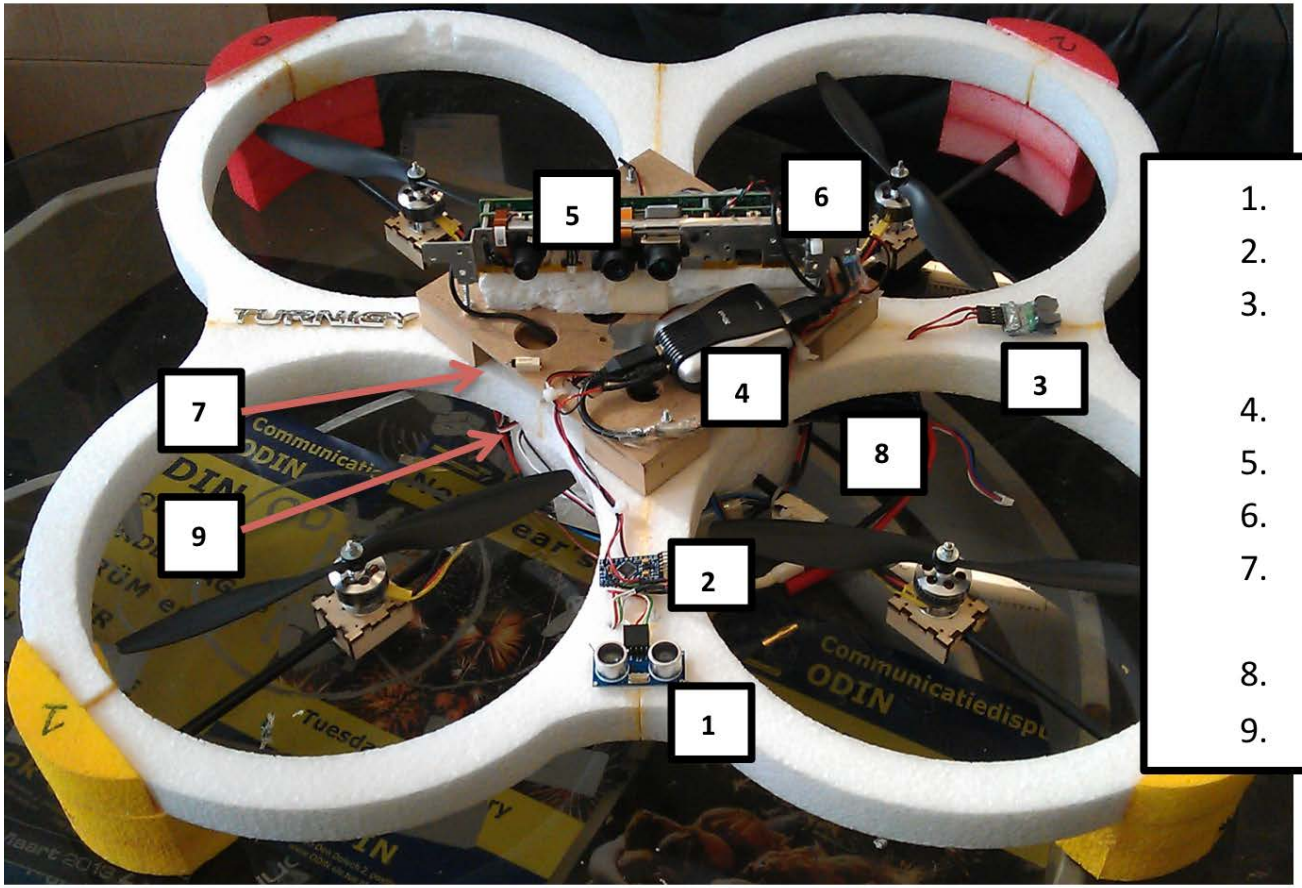
E) Iterative Closest Point steps

1. Find for each point the closest point in the second cloud.
    1. Linear search
    2. K-nearest neighbors search
2. Find the transformations per previews found points.
    1. Mean square cost function
3. Transform the points.
4. Iterate this process.

Results

- A) Quadcopter is built
- B) Is flying stable
- C+D) Algorithm are working correctly
  - But cannot track the line over time
    - Too small webcam angle
    - Too slow processing
- E) Showing mapping
  - Mapping does not grow
    - ICP not working correctly

Quadcopter result



1. Top and bottom sonar
2. Sonar processing
3. Battery low voltage alarm
4. USB hub
5. Kinect
6. Power regulator
7. Beagle board, Multiwii and RF receiver
8. LiPo battery pack
9. One of the 4 ESCs

Recommendations

- Wide angle lens
- Acceleration of algorithms
  - DSP
  - Resolution down scaling
- Mapping
  - Do not use JNI within mapping program
  - Build program complete in PCL

Demo

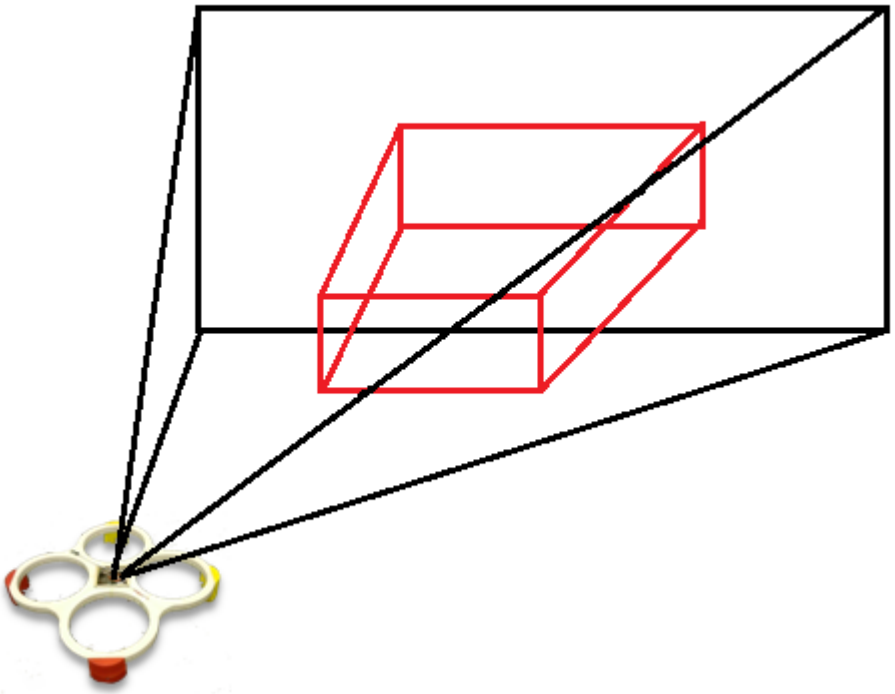
Questions?

?

?

?

Obstacle Detection



Global control (HW)

	Raspberry PI, Model B	BeagleBoard-xM	BeagleBoard	BeagleBone	BeagleBone Black	PandaBoard	PandaBoard/ES
Price [$]	35	**149**	149	89	45	174	182
SOC Type	Broadcom BCM2835	**TI AM3715**	TI OMAP3530	TI AM3359	TI AM3359	TI OMAP4430	TI OMAP4460
Core	ARM1176JZF-S	**Cortex-A8**	Cortex-A8	Cortex-A8	Cortex-A8	Cortex-A9	Cortex-A9
no. of Cores	1	**1**	1	1	1	2	2
CPU Clock [GHz]	0,7	**1**	0,6	0,72	1	1	1,2
GPU	VideoCore IV	**PowerVR**	PowerVR	SGX530	SGX530	no	no
DSP	yes	**C64x**	C64x	no	no	C64x lite	C64x lite
Open GPU/DSP	no	**yes**	yes	yes	yes	yes	yes
RAM [MB]	512	**512**	128	256	512	1024	1024
USB ports	2 on Host	**4 on Host, 1 on OTG**	1 on Host	1 on Host, 1 on OTG	1 on Host, 1 on OTG	2 on Host, 1 on OTG	2 on Host, 1 on OTG

Became available after start of the project

E) Iterative Closest Point Phases