



Istrobotics

Robotour 2019, 15.9.2019

Pavol Boško, Peter Boško, Radoslav Kováč

2016



2017



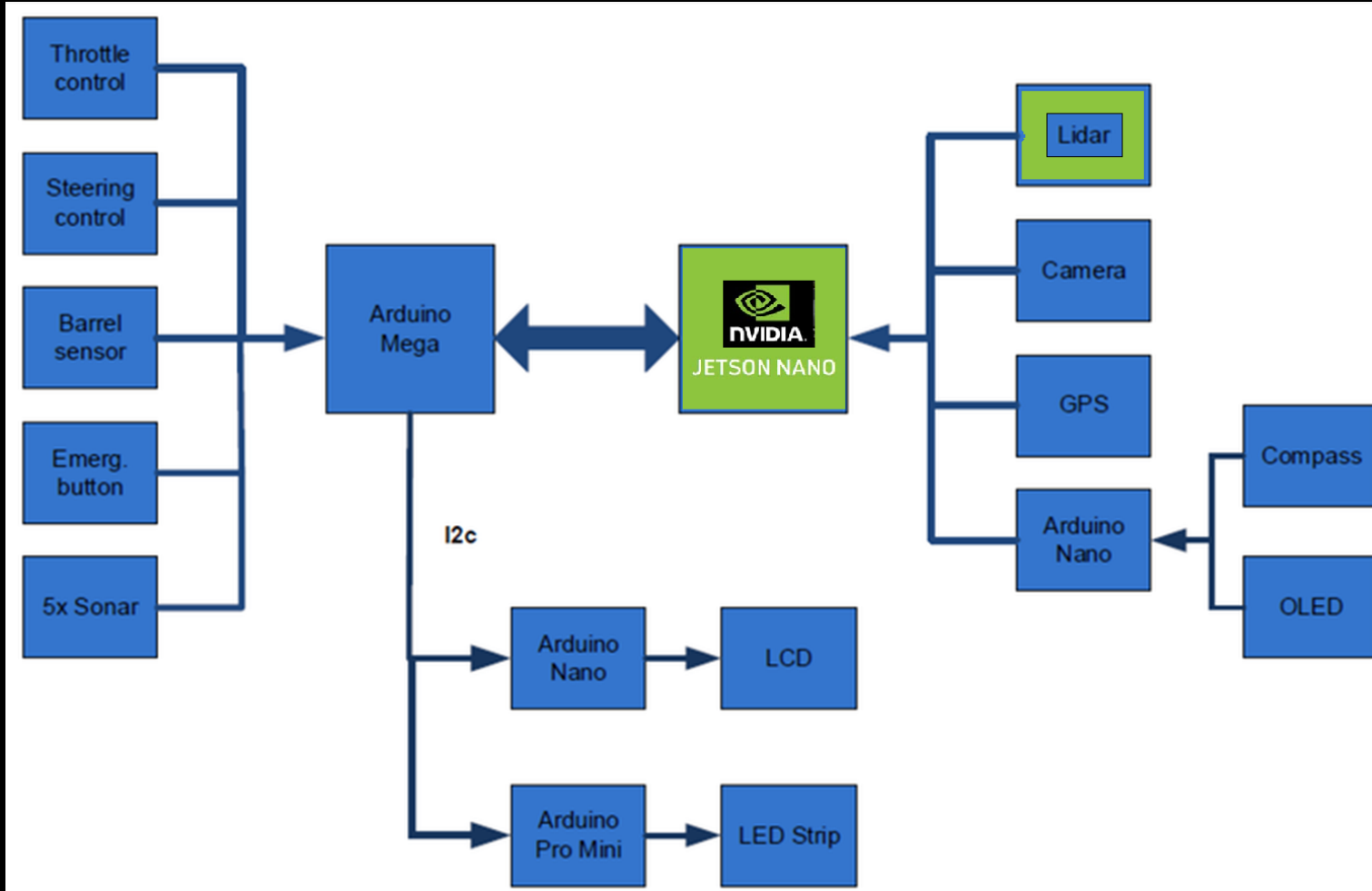
2018



2019



HARDWARE DESIGN



HARDWARE



- **Jetson Nano: 1.4GHz 4-core, 4GB RAM, 64GB SD**
 - Arduino Mega: 16MHz, 8KB RAM
 - 2x Arduino Nano: 16MHz, 2KB RAM
 - Arduino Pro Mini: 16MHz, 2KB RAM
- **2D Lidar: Sick TiM571, 15Hz, 0.3° res. (\$2400)**
- 2D Lidar: RoboPeak RPLIDAR 360, 7Hz, 1° res. (\$400)
- **Camera #1: Odroid oCam 5MP (640x480, 170 FOV)**
- Camera #2: Odroid USB Cam (640x480, 65 FOV)
- Mouse type GPS/Glonass: Holux M-215+
- Compass: Bosch BNO055
- 5x Sonar: HC - SR04
- LCD & OLED displays, 8x LED

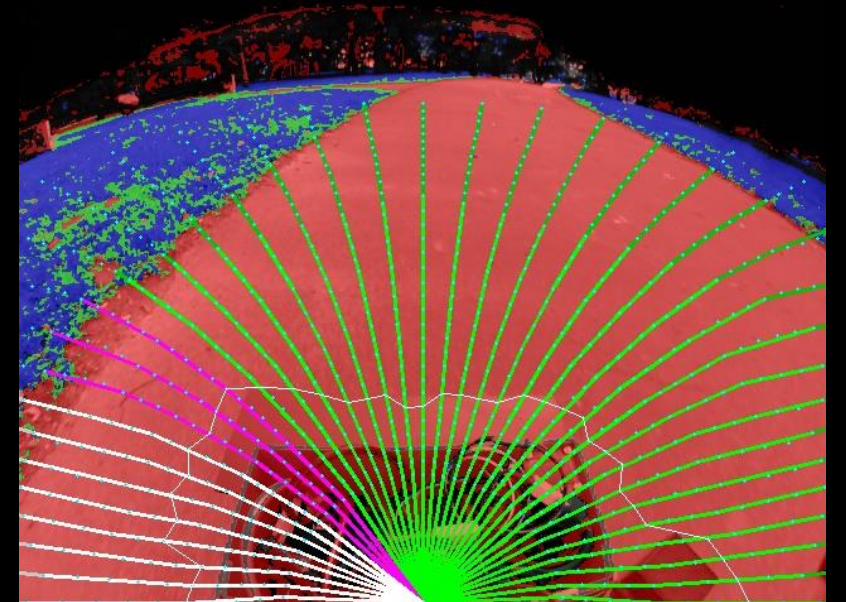
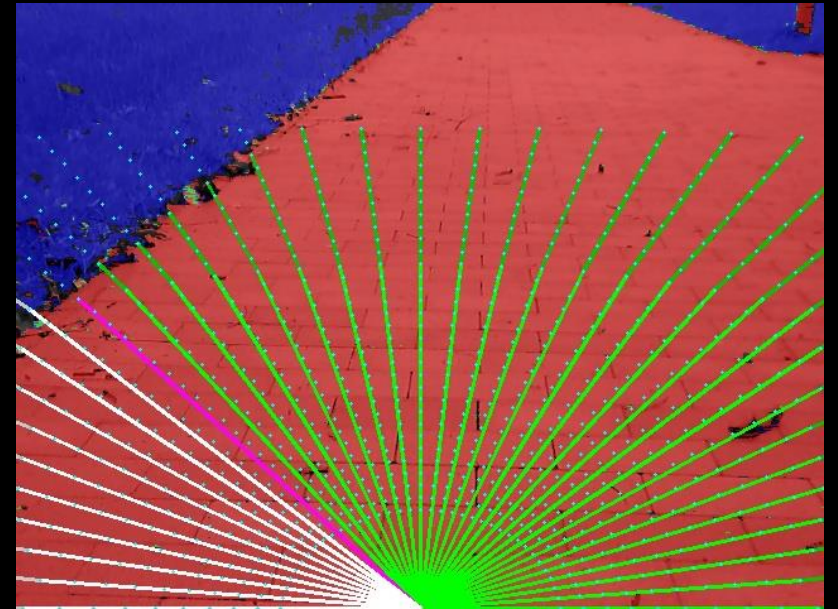
Jetson Nano vs Odroid-XU4

	Raspberry Pi3	Odroid-XU4	Jetson Nano
CPU	ARM Cortex-A53	Samsung Exynos5422 Cortex	ARM Cortex-A57
Clock	1.2 GHz	2 GHz	1,43 Ghz
Cores	4x	8x	4x
RAM	1GB LPDDR2	2GB LPDDR3	4GB LPDDR4
Flash	microSD	eMMC5.0 HS400	microSDXC Class 10 UHS-I
Ethernet	10/100 Mbit	1 Gigabit	1 Gigabit
USB	4x USB 2.0	2x USB 3.0 1x USB 2.0	4x USB 3.0

	R-Pi3	O-XU4	J-Nano
Image processing	167,3 ms	39,4 ms	27,3 ms
JPG/PNG writing	55 ms	17,7 ms	8,1 ms
Processing lag	2 sec	100 ms	0 ms (ramdisk)

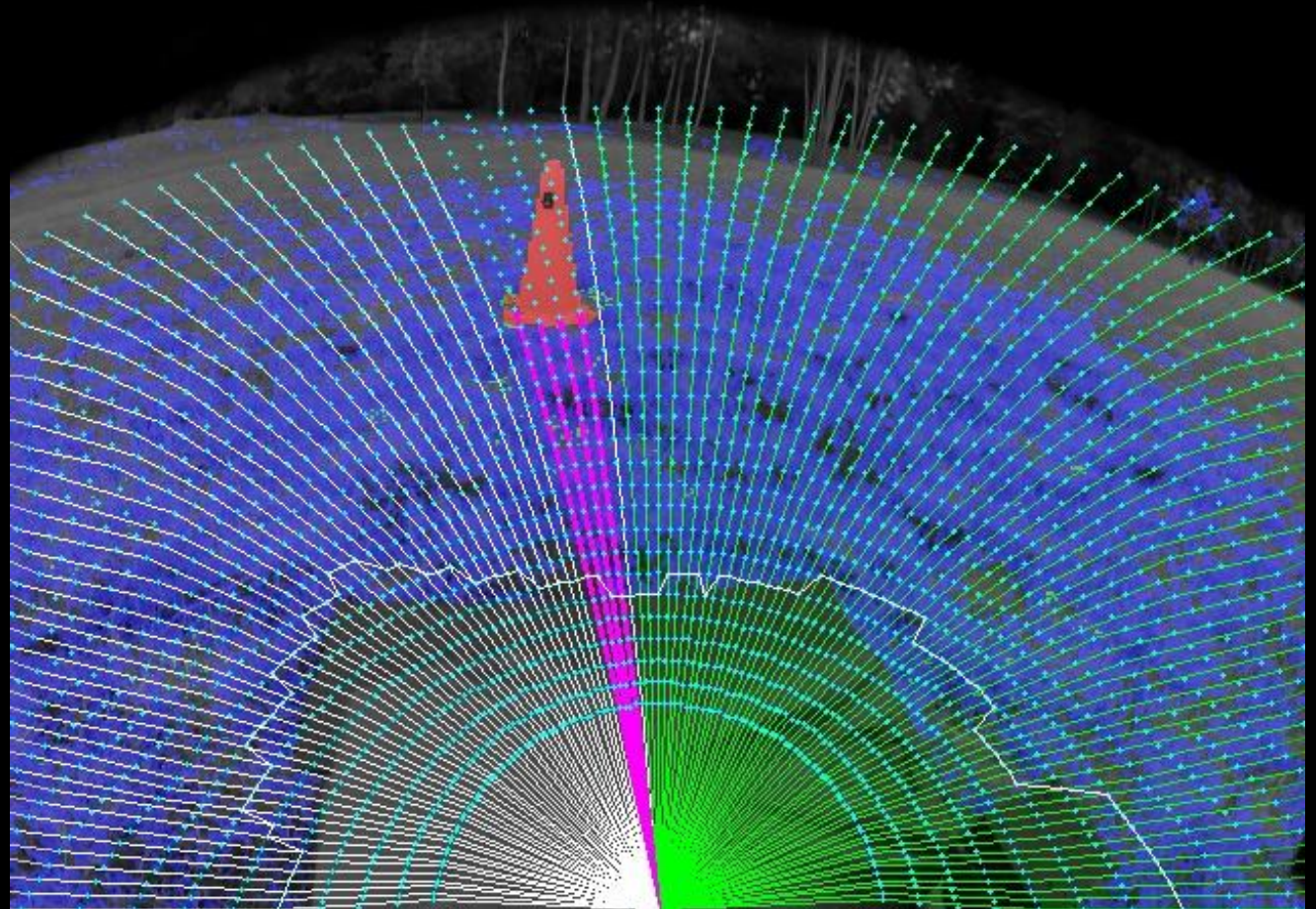
FISHEYE IMAGE PROCESSING

- **image transformation**
 - correction of mapping between XY points and local map coordinates – curved lines
 - OpenCV undistort() was not used (because of slow performance)
- **front side of the robot on every image**
 - was fixed by SW masking
 - presents an issue for training a neural network
- **sky was “masked” using a black tape**
 - for avoiding white ballancing camera issues

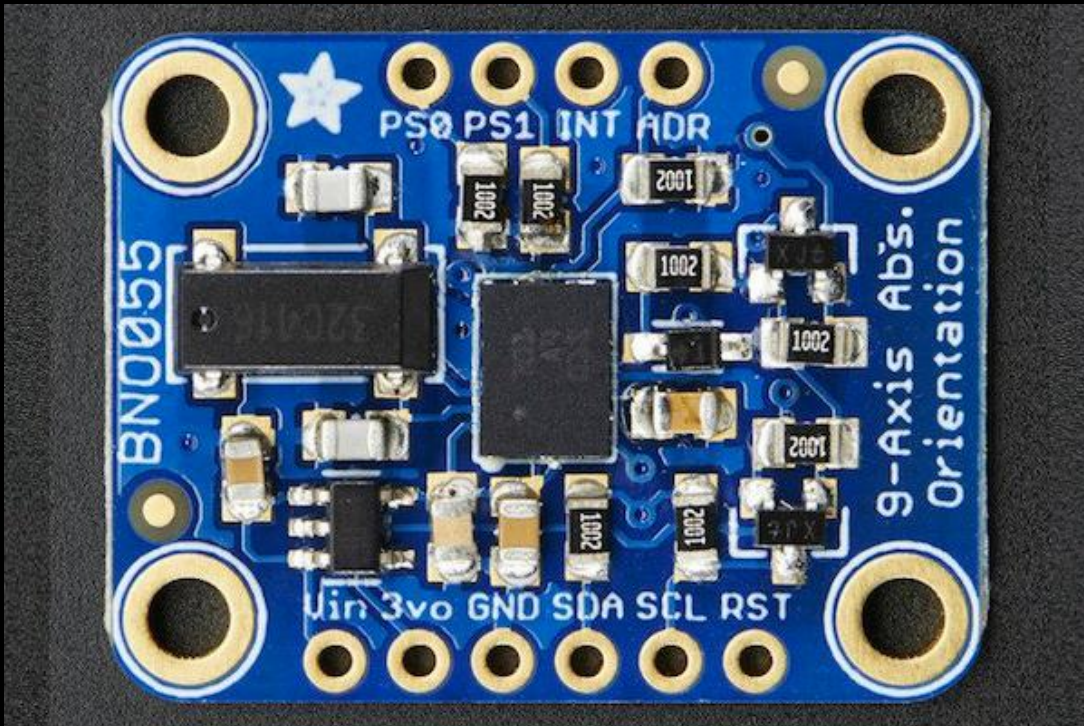


VISION - ROBOORIENTEERING

- Vision algorithm was modified to detect orange cones

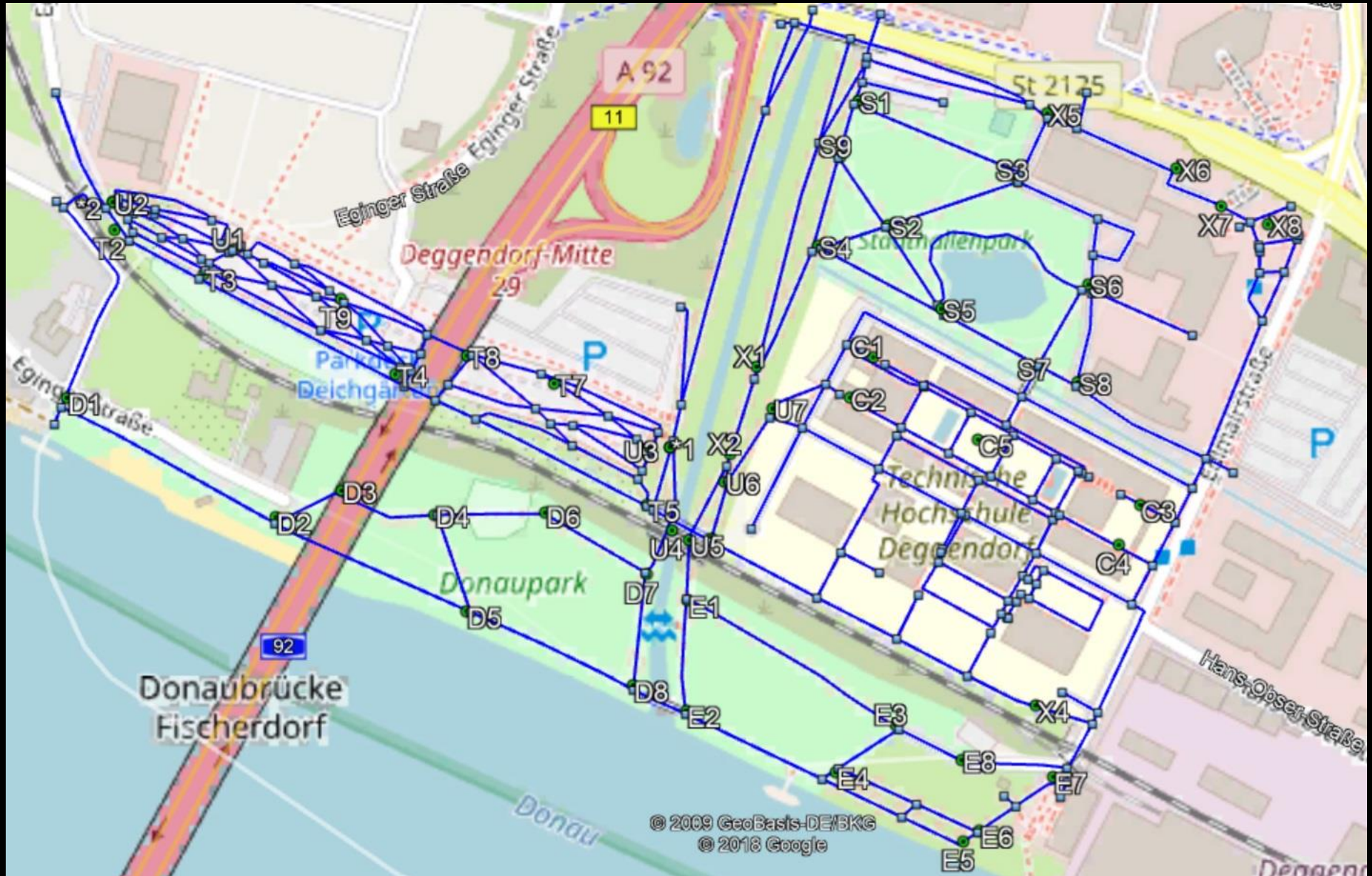


COMPASS ISSUES



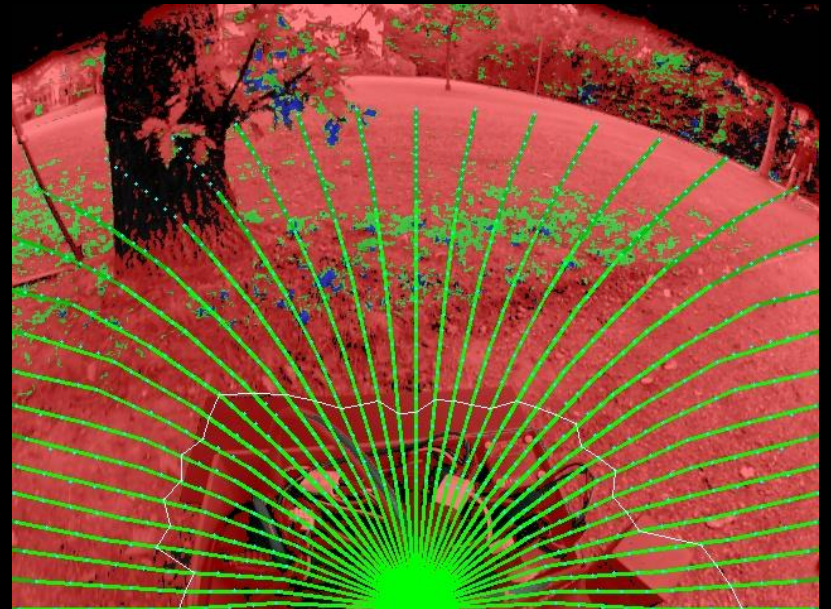
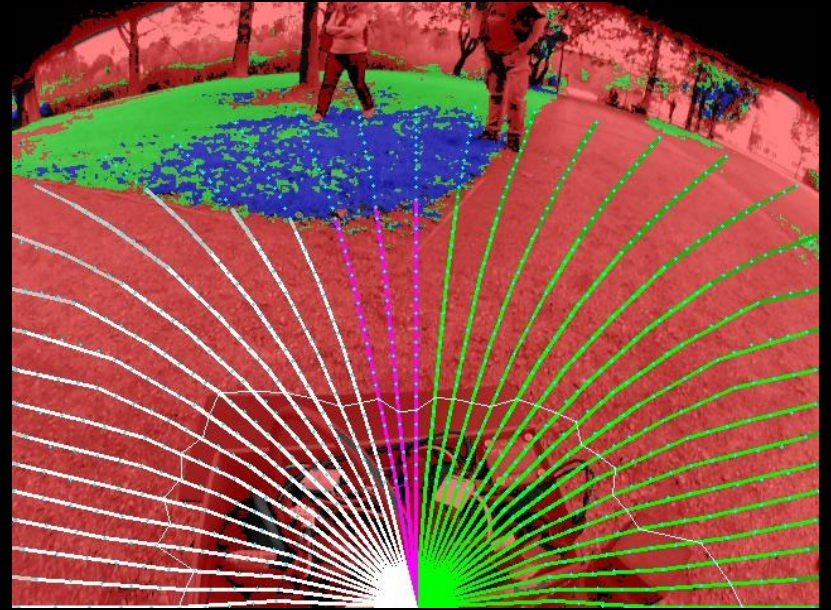
- Compass vs GPS calibration:
 - interval of 7 seconds - robot is moving straight
 - gps and compass azimuths to be fixed
 - could be performed every 30 seconds – 3 minutes
- Issues:
 - after Round #1 very inaccurate results – delta values between 45° and 90°
 - robot is sometimes slightly left turning
 - local changes during the day (bridge)
- for testing in Bratislava and also for competition we used fixed value 60°

NAVIGATION MAP



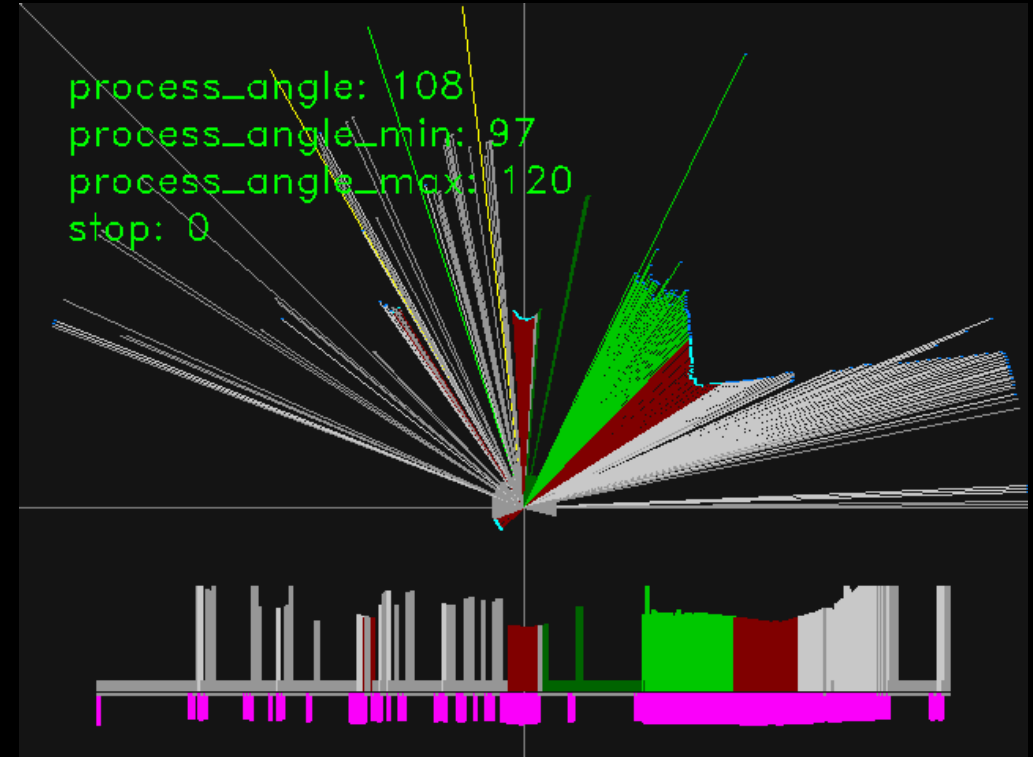
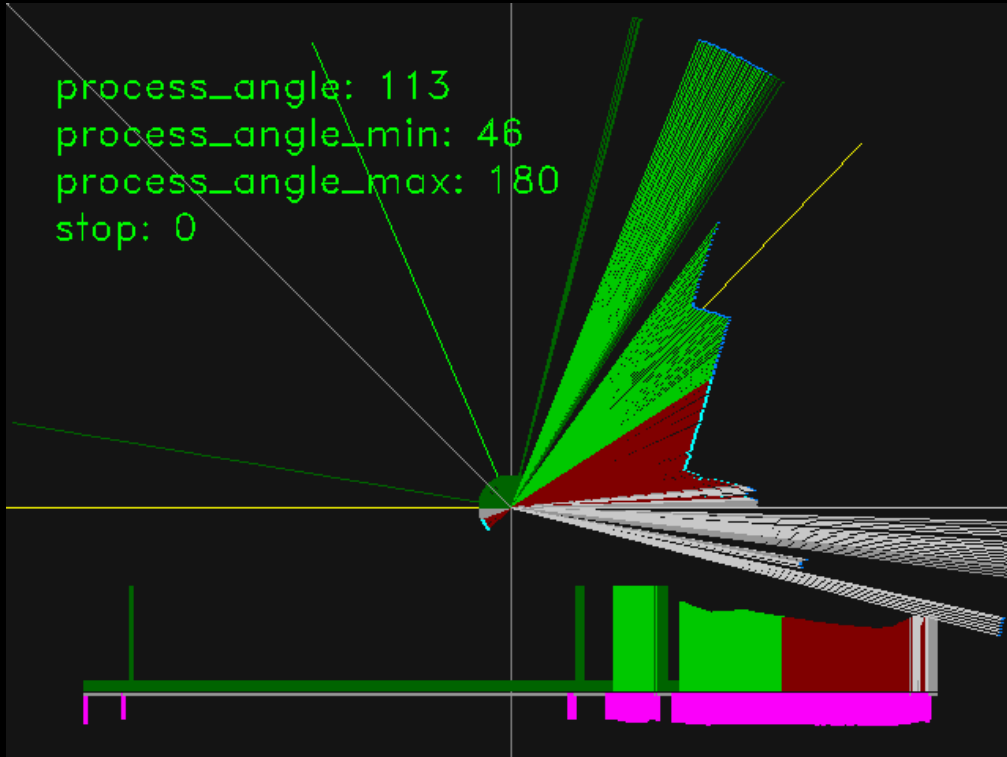
PROBLEMS - ROUND #1

- **ROUND1** (failed before loading), 170FOV camera:
 - we realized that we forgot all our batteries in Bratislava
 - Pablo was soldering connectors for borrowed batteries – not enough time for testing
 - missed turn on the first crossing (still in semiautonomous area – we didn't know about it)
 - vision did not detect grass correctly during a U-turn



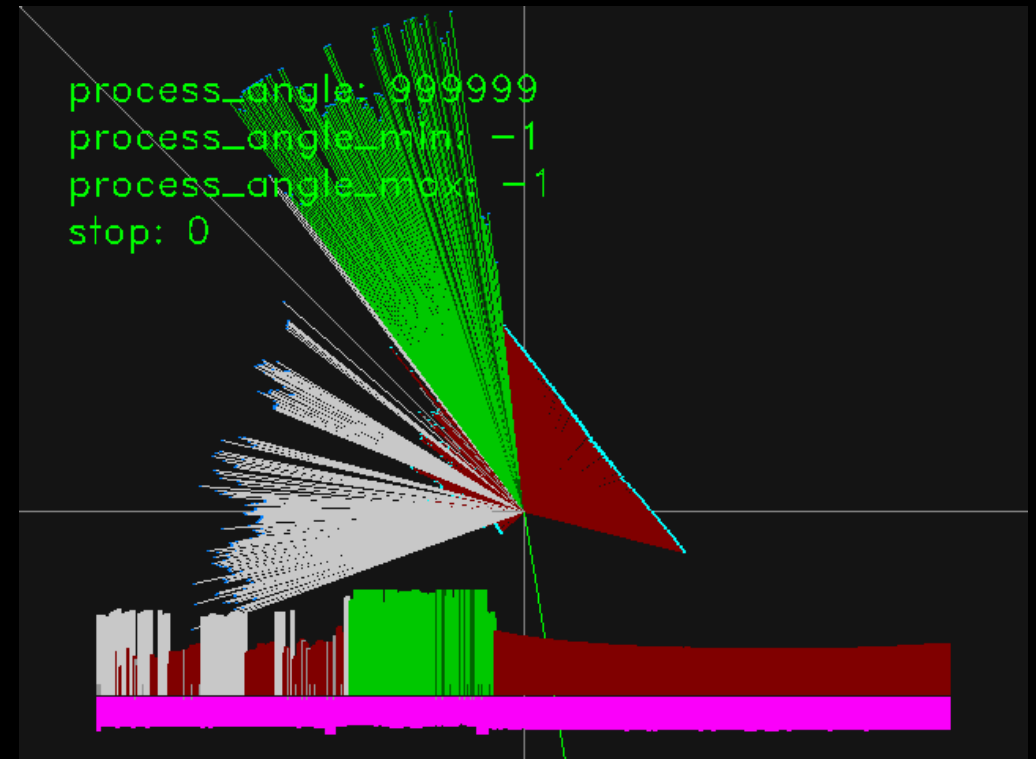
PROBLEMS - ROUND #2

- **ROUND2** (failed before loading) , 170FOV camera:
 - camera stopped working (USB3 issue)
 - robot started to turn too late on a T-crossing



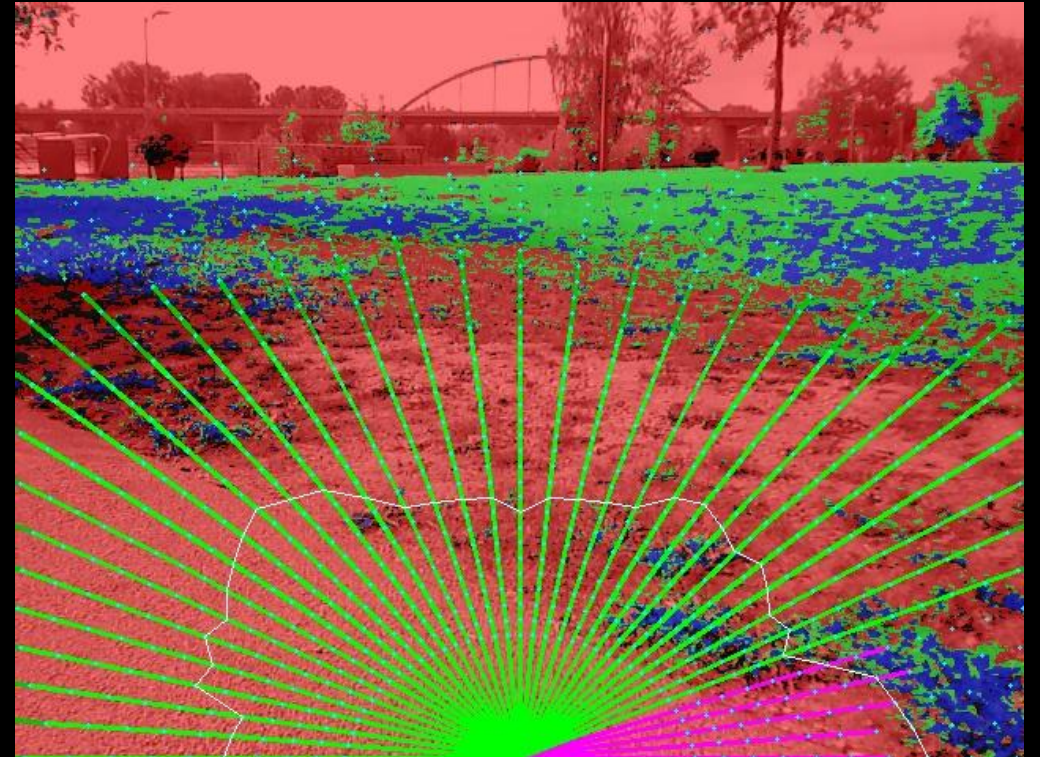
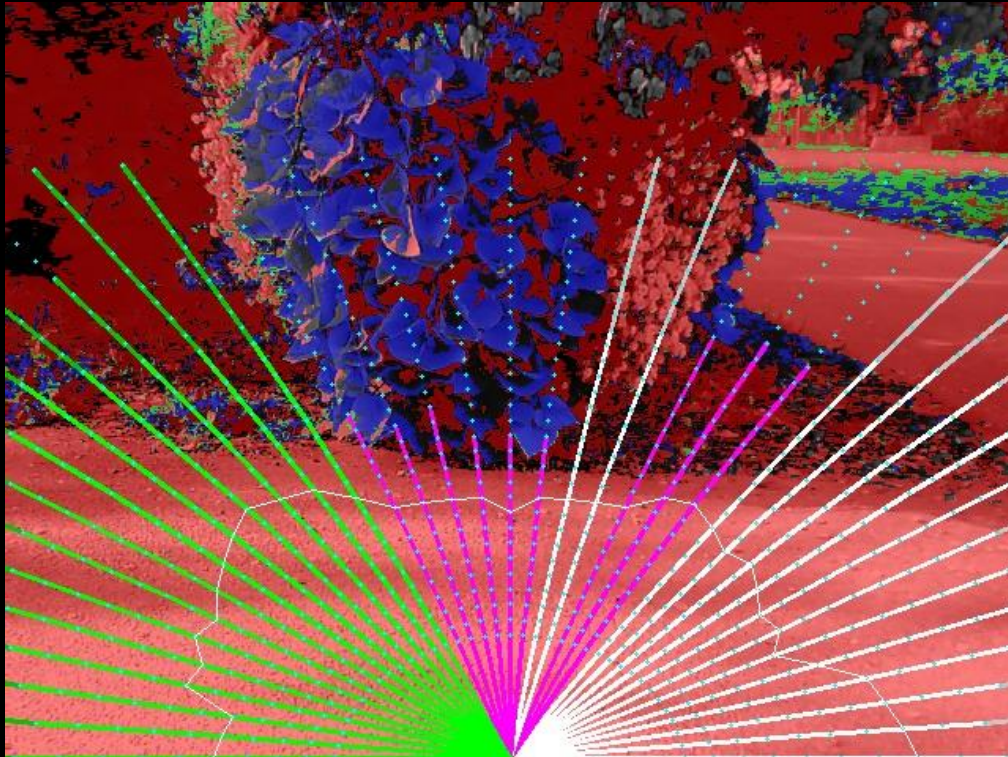
PROBLEMS - ROUND #3

- **ROUND3** (failed before loading), old camera:
 - wide road – robot came into U-shape corner (trap)

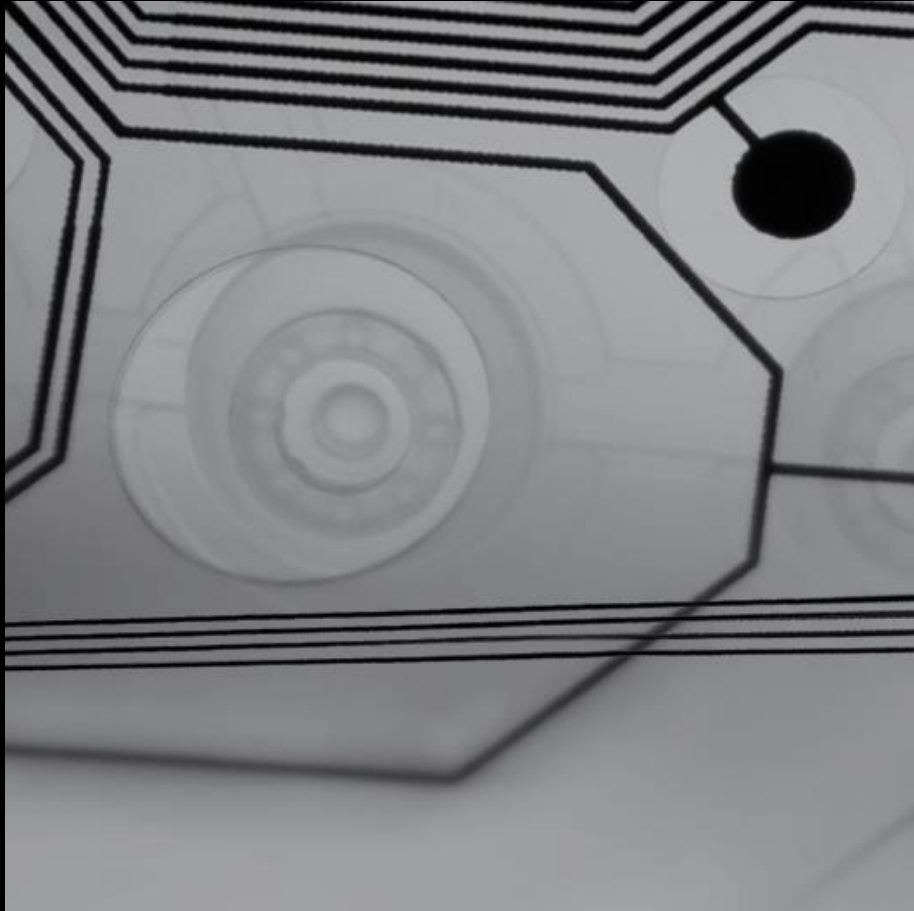


PROBLEMS - ROUND #4

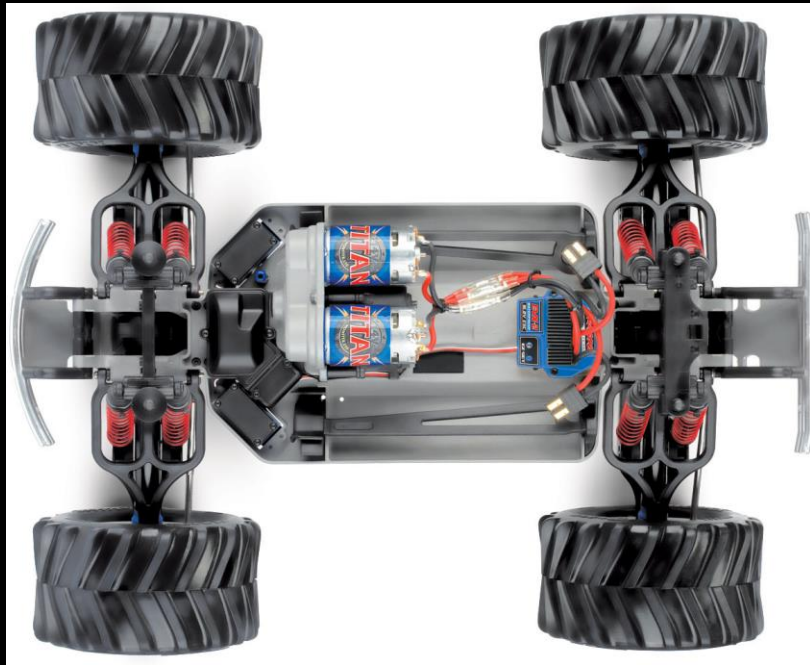
- **ROUND4** (failed before loading) , old camera:
 - missed turn on a 4-way crossing – deciding locally
 - vision did not detect grass correctly during a U-turn



GENERAL SLIDES



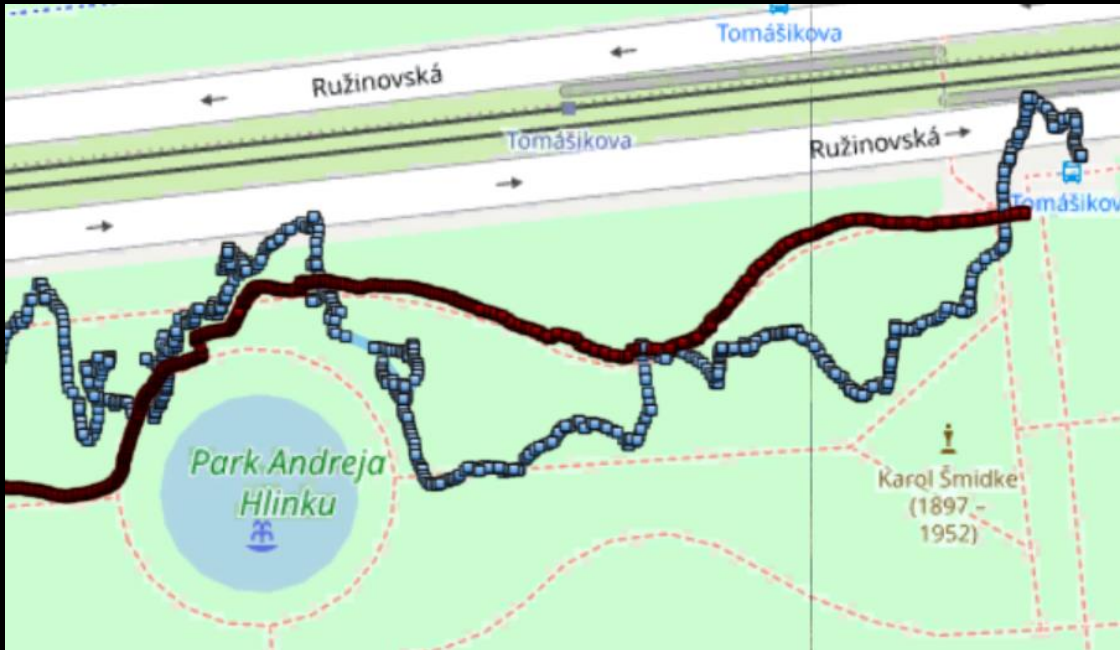
ROBOT CHASSIS



- RC model: Traxxas E-Maxx 4x4 monster truck
- Top Speed: 48 km/h
- Waterproof electronics, servos



GPS GROUND PLANE



- U-Blox GPS Antenna documentation
 - Patch antennas - flat surface is ideal
 - can show very high gain, if mounted on large ground plane (70x70mm)
- USB 3.0 impact on GPS
 - Intel paper: USB 3.0* Radio Frequency Interference Impact on 2.4 GHz Wireless Devices
- We used simple shield for GPS
- Results: **great improvement (3m accuracy)**

170FOV CAMERA



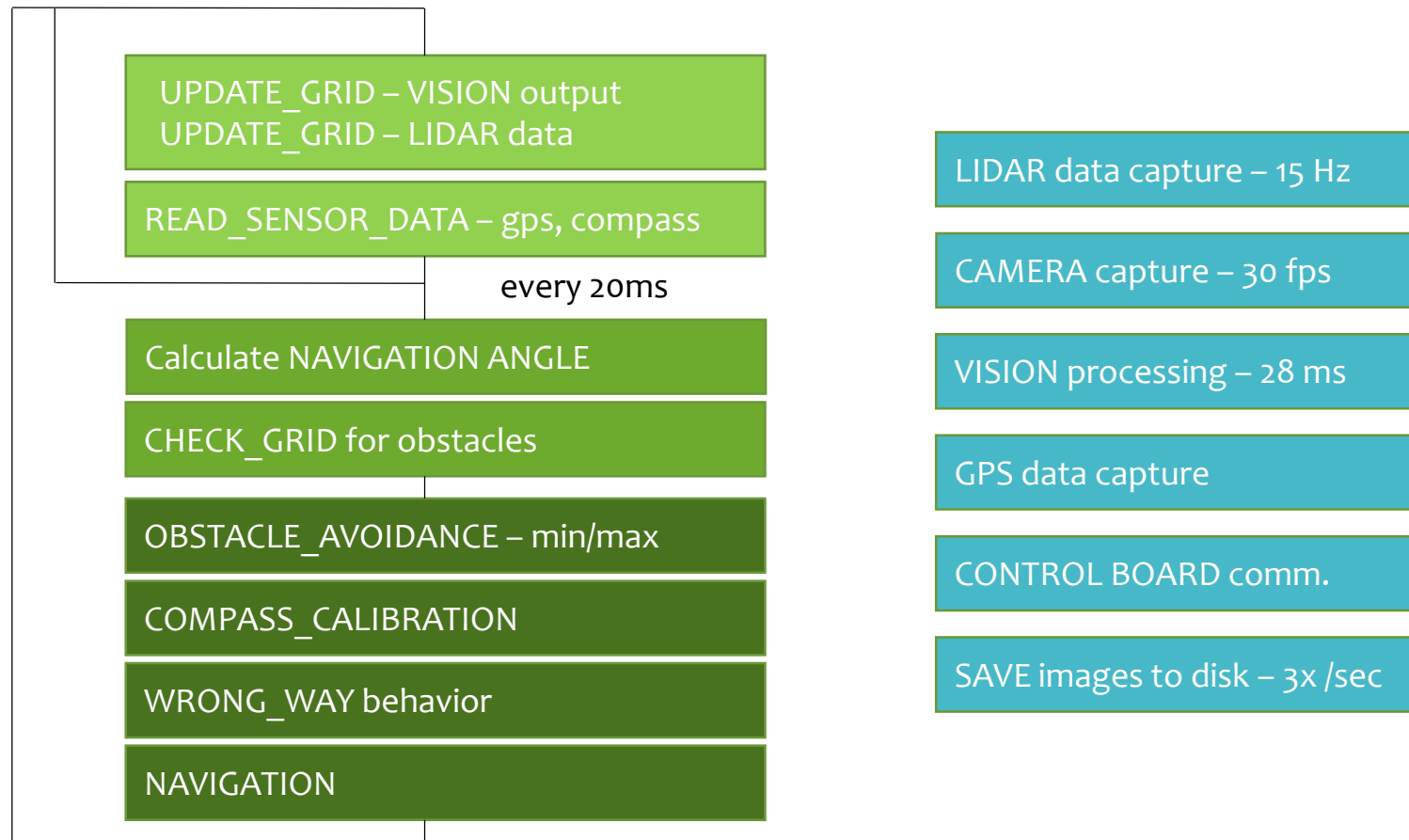
- **oCam : 5MP USB 3.0 Camera**
 - OmniVision OV5640 CMOS image sensor
 - Original lens Field Of View: 65 Degree
- **Exchangeable Standard M12 Lens**
 - Separate: 170 Degree Wide Angle
 - (standard accessory also for GoPro cameras)
- we 3d-printed an adjustable camera holder (fixed by screws)
- we broke the USB connector during the first test drive

SOFTWARE

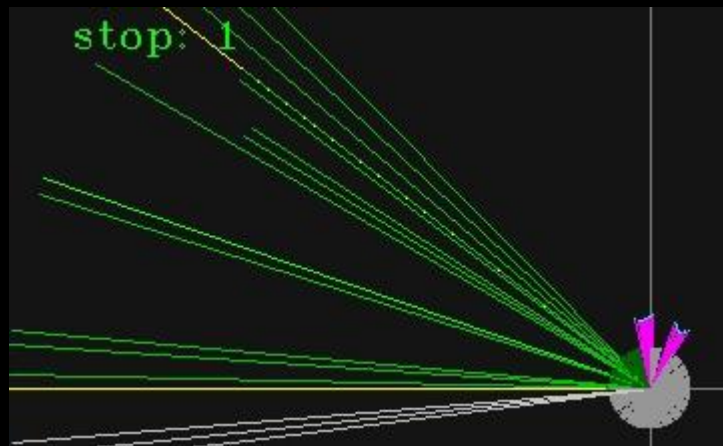
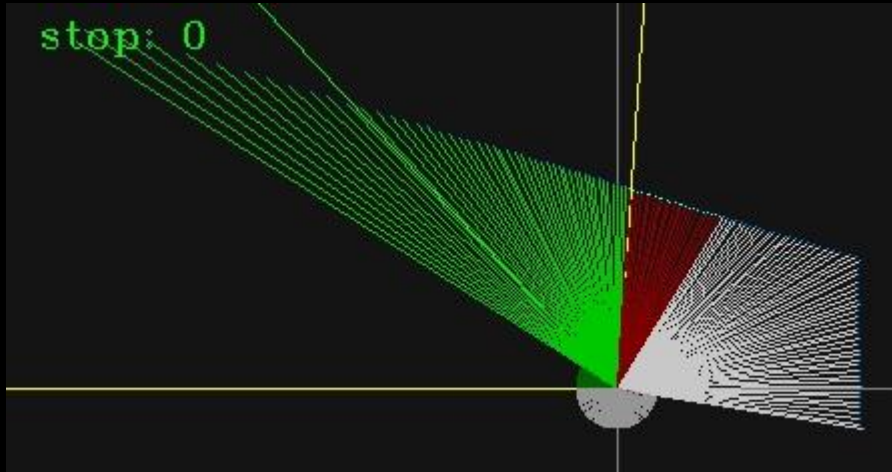


- Operating system: Ubuntu 16.04 Mate
- Source codes: C++, 430kB
 - 2017: 340kB, 2016: 180kB
- Libraries: OpenCV (vision), GeographicLib (Geo), Zbar (QR-Codes), Libxml2 (.osm), log4cxx (logging)
- Main application + 8x pthreads
 - 4x sensors (Camera, Lidar, GPS , Compass)
 - image capturing + vision processing
 - output: image saving (1GB of data/ round)
 - control board (Compass)

SOFTWARE DESIGN – PROCESSING

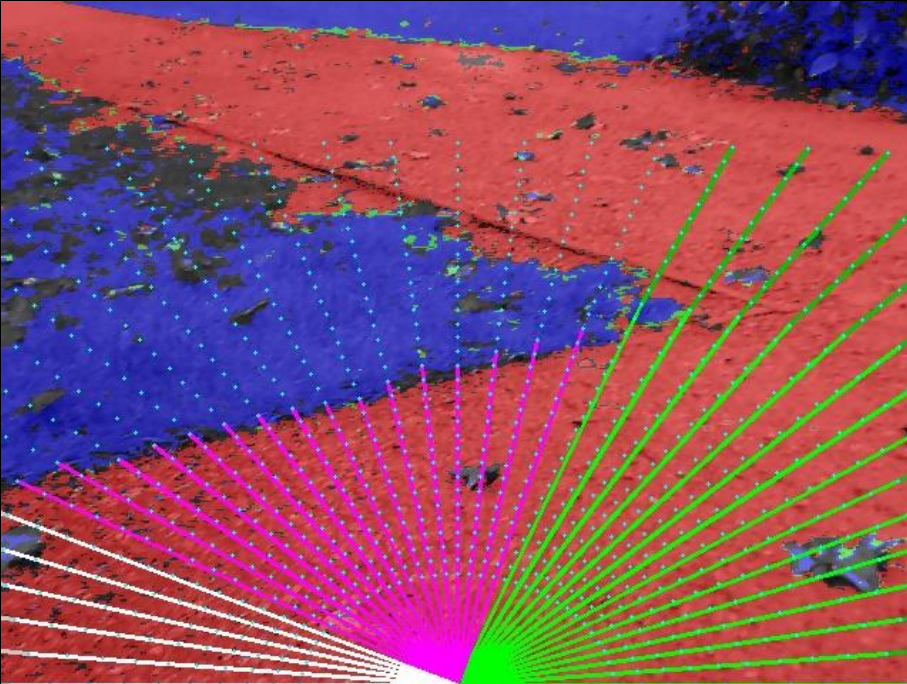


LIDAR – obstacle detection



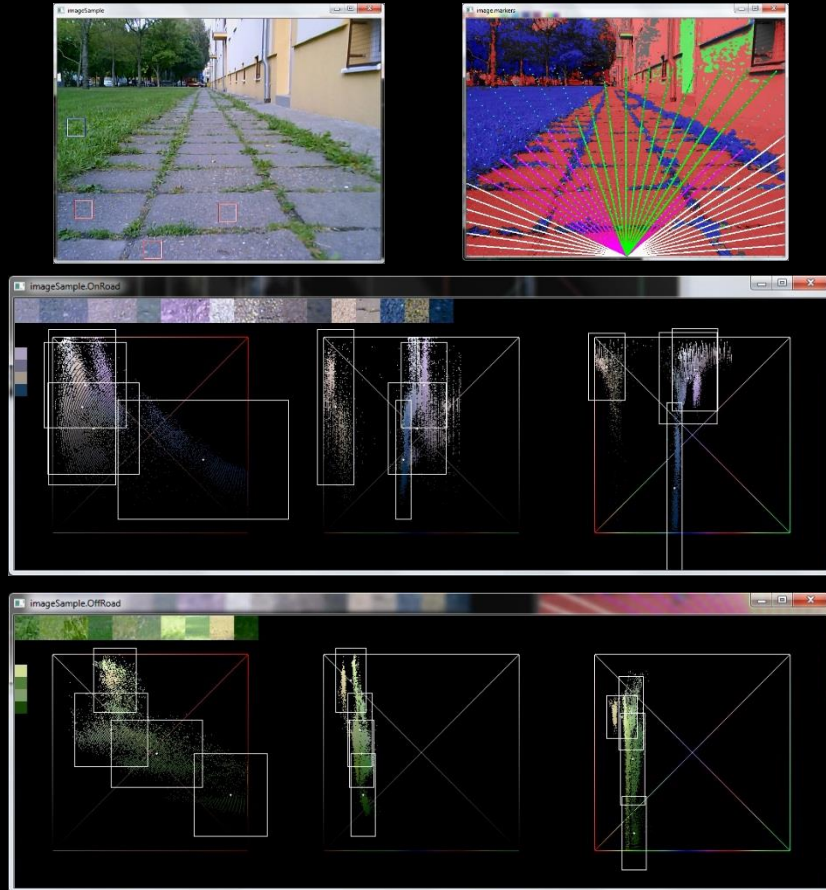
- Obstacle detection condition (red):
 - If distance is < 100 cm
 - Filtering: distance < 1 cm (grey)
- Stop condition (pink):
 - Check angle: -45 to $+45$ degrees
 - If distance is < 50 cm at 3 diff. degs
 - Sonars were also used (rain issue)
- Obstacle avoidance (green/white)
 - Find OK intervals of > 20 degrees
 - Choose the closest to going straight

VISION – approach



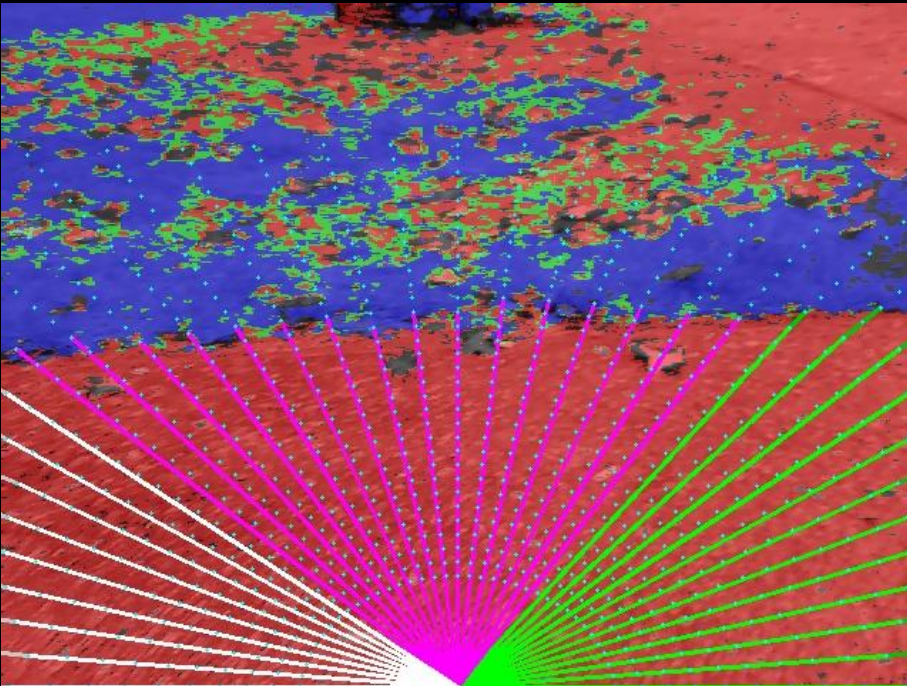
- Our approach: **lidar-like local map**
 - For any seen angle is obstacle closer than 1 meter?
 - 1 meter or to the image border
- Algorithm:
 - Pixel color classification
 - Evaluate grid points
 - Calculate distance to obstacle
 - Find OK intervals – same like LIDAR

VISION – Pixel color classification



- Approach:
 - Choose sample pixel blocks (32x32) from training images
 - Calculate 4 clusters centers in color space (OpenCV kmeans)
 - Calculate cluster radius (histogram based)
 - Repeat for 2 classifiers : road and off-road (grass)
- HSV color space + Euclidian distance
- Tool was developed - to define pixel blocks and evaluate images

VISION – Algorithm



- Pixel color classification - 4 results:
 - Road (red)
 - Off-road (blue)
 - Both (green)
 - None (grey)
- Evaluate grid points
 - Cca 1000 points in 37 lines (5 deg)
 - Evaluating nearby pixels (80x80)
 - Majority of “Road” pixels is checked
- Calculate distance to obstacle
- Find OK intervals + merge with LIDAR

NAVIGATION – ROUTE PLANNING



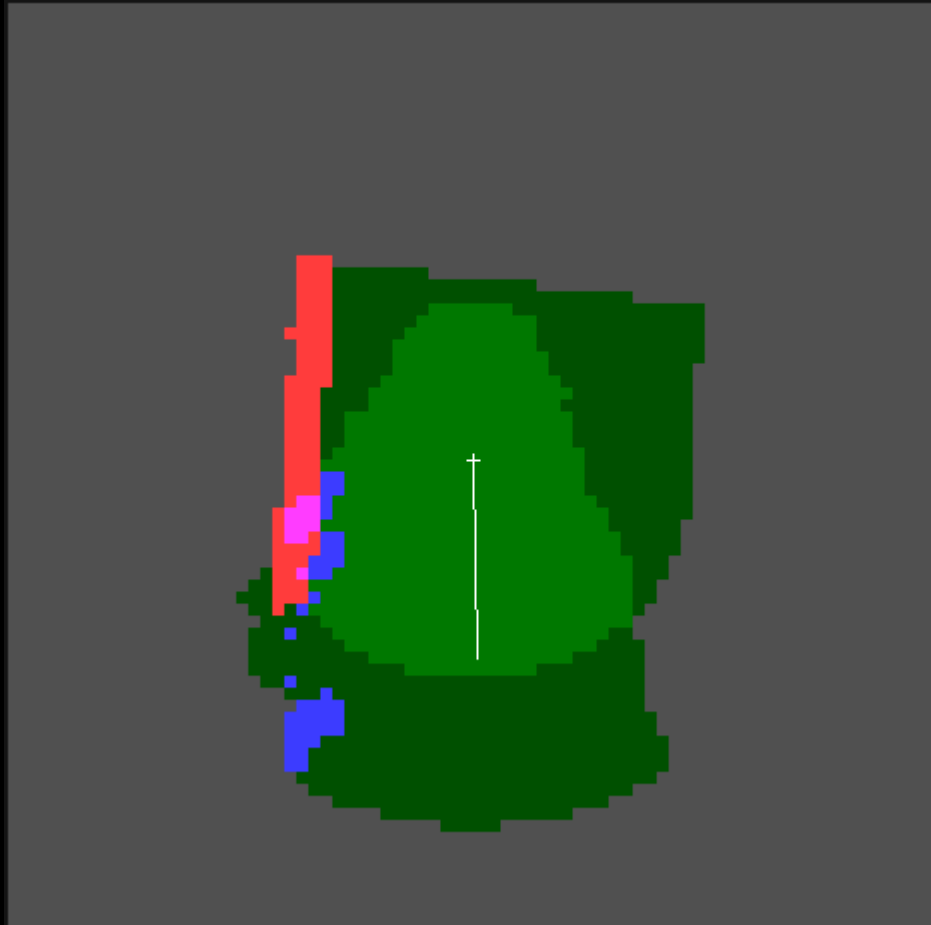
- OpenStreetMap data export
 - filter segments: footway, track with `<grade3`
- **Dijkstra's Shortest Path Algorithm**
 - 418 nodes, 504 segments
- Performance: `< 2ms`
- Visualisation: kml export, cpp export
- Navigation: keep azimuth towards a point that is 10m along planned route

QR CODES

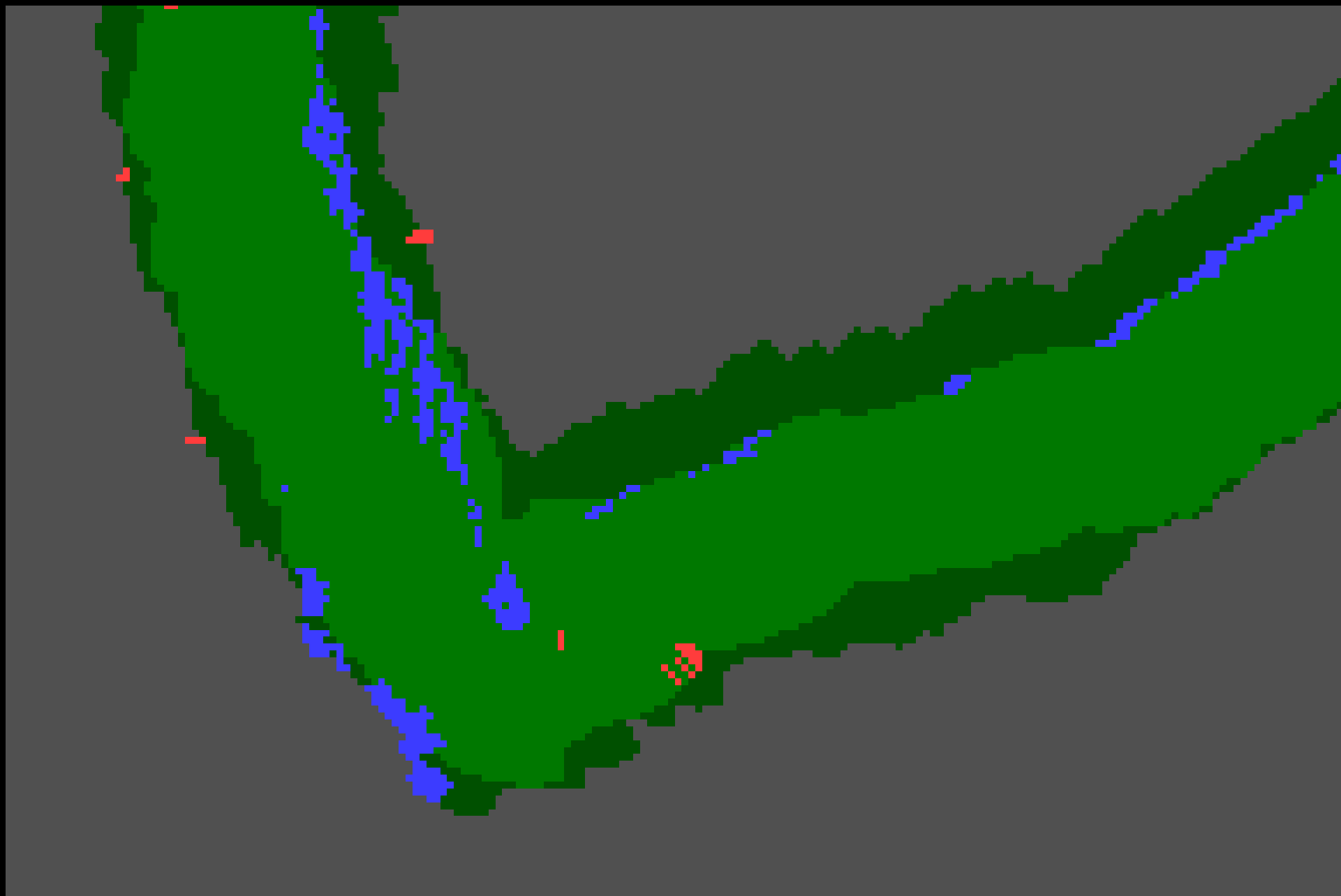


- **ZBar bar code reader**
 - open source software suite
 - supports: EAN-13/UPC-A, UPC-E, EAN-8, Code 128, Code 39, Interleaved 2 of 5 and QR Code
- Performance impact:
 - one execution: 50-200ms
 - our target 30-50ms per frame
- Solution: images are scanned for QR codes only when waiting for navigation coordinates

WORLD MAP – BUILDING LOCAL GRID



- Local grid map
 - to store polar information from lidar and vision
 - 1 cell: 10 x 10 cm, 1 byte per cell, array[2000][2000]
 - always overwriting with new data - no heatmap
- Local position taken from GPS + odometry
 - wheel encoders provide speed information
- Colors used for visualization
 - Blue – grass (not-a-road) detected
 - Red – lidar obstacle
 - Green - no obstacle (light green = both sensors)



VISUALISATION

- Visualisation in web browser
 - works on notebook/tablet/phone
- Messages from log files
 - last set of lines matching selected substrings
 - Info: robot display, gps, processing, calibration
- Log parser is exporting interesting data do .json file
- Web page performs Ajax JSON requests every 1 sec
 - Images are only downloaded on demand (checkbox)

☒camera ☒vision ☒lidar ☒wmgrid ☒navmap [reload](#)

```
2018-09-08 05:29:07,005 INFO istro::process_thread(): process_angle("MIN_MAX"): navp_azimuth=999999.00, process_yaw=166.52, process_angle_min=0, process_angle_max=91, process_angle=45, process_dir=90, yaw_src=1

2018-09-08 05:29:06,646 DEBUG istro::gps_writeData(): fix=1, latitude=48.156538, longitude=17.156745, latitude_raw=48.156538, longitude_raw=17.156745, speed=0.330, course=270.00, gps_x=-189.30, gps_y=-53.63, ref=1, navp_dist=-1.000, navp_azimuth=999999.00, navp_dist_raw=-1.000, navp_azimuth_raw=999999.00, navp_maxdist=-1.000, navp_mindist=-1.000, navp_loadarea=1, navp_idx=-1, navp_latitude=999999.000000, navp_longitude=999999.000000, navp_latitude_raw=999999.000000, navp_longitude_raw=999999.000000, lastp_dist=0.000, lastp_azimuth=0.00, nmap_flags=0

2018-09-08 05:29:07,026 DEBUG istro::process_readData(): process_change=52, gps_speed=0.330, gps_course=270.00, lastp_dist=-1.000, lastp_azimuth=0.00, ctrlb_ircv500=12.39, ctrlb_angle=96, ctrlb_velocity=91, ctrlb_loadd=1, euler_x=166.52, ahrs_yaw=999999.00, navp_dist=-1.000, navp_azimuth=999999.00, navp_maxdist=-1.000, navp_loadarea=1, navp_idx=-1, gps_ref=1, gps_x=-189.30, gps_y=-53.63

2018-09-08 05:28:39,831 INFO istro::calib_process(): msg="calibration interrupted (obstacle)!", calib.ok=0, process_angle_min=0, process_angle_max=92

2018-09-08 05:28:59,638 INFO istro::gps_thread(): msg="navigation point passed!", pos=4, name="*2", navp_dist=0.000, navp_mindist=19.756

2018-09-08 05:28:59,638 INFO navig::navigation_next_point(): msg="point not found!", pos=4

2018-09-08 05:29:07,026 TRACE istro::loadarea_process(): msg="unloading area - waiting!", gps_navp_loadarea=1, ctrlb_loadd=1, process_state=4, gps_navp_idx=-1, gps_navp_azimuth=999999.00

2018-09-08 05:29:06,591 INFO ControlBoard::write(): data="DAf:S0v343 A277 MMAX"
```

out/zt2018_8441633_0001264_camera.jpg

out/zt2018_8441633_0001264_vision.jpg

out/zt2018_8441633_0001263_lidar.png

out/zt2018_8441633_0001271_wmgrid.png

process_angle: 48
process_angle_min: 0
process_angle_max: 96
stop: 0

out/zt2018_8441633_0001244_navmap.png

FREE SOURCE CODES



- Sources codes are available at GitHub as public project **Istro RT**:

<https://github.com/lnx-git/istro-rt>

THANK YOU

