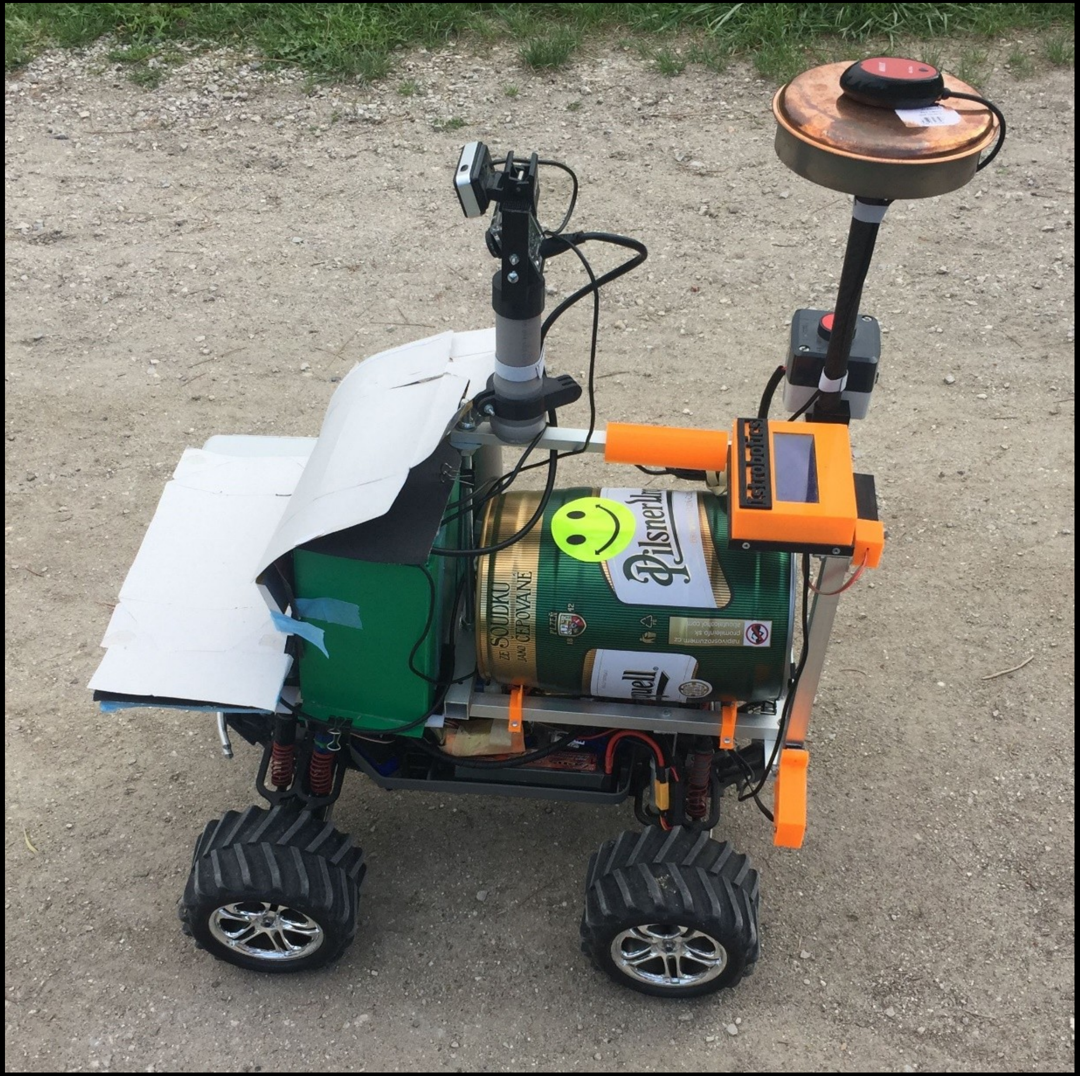
Istrobotics

Robotour 2018, 16.9.2018

Pavol Boško, Peter Boško, Radoslav Kováč, Tomáš Kováč, Ivana Kováčová

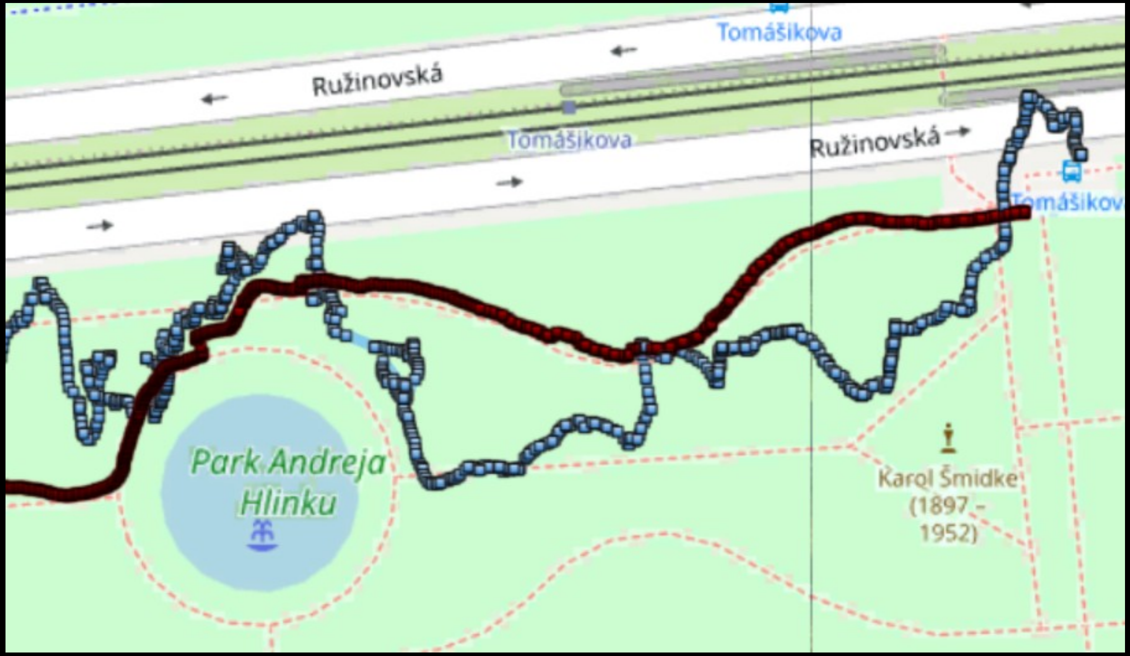
2016 2017 2018


Pilsner

NEW GPS TESTS

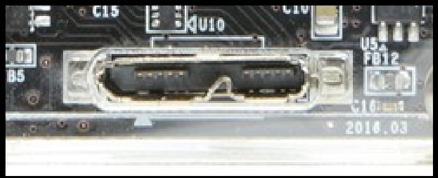
- Motivation: poor GPS performance last year
- Old GPS (blue): **MTK MT3333 GPS/GLONASS**
  - Holux M-215+ Dual GNSS: GPS/GLONASS
  - -165dBm, 66 searching, 22 tracking channels
- New GPS (green): **Ublox NEO-M8 + *Compass***
  - Concurrent reception of up to 3 GNSS (GPS, Galileo, GLONASS, BeiDou)
  - –167 dBm, 72 channels, 30 tracking
  - Position: every 200ms
- Issues: sometimes no course for more than 1s
- Results: ***the same position accuracy (10-20m)***

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)**

NEW 170FOV CAMERA



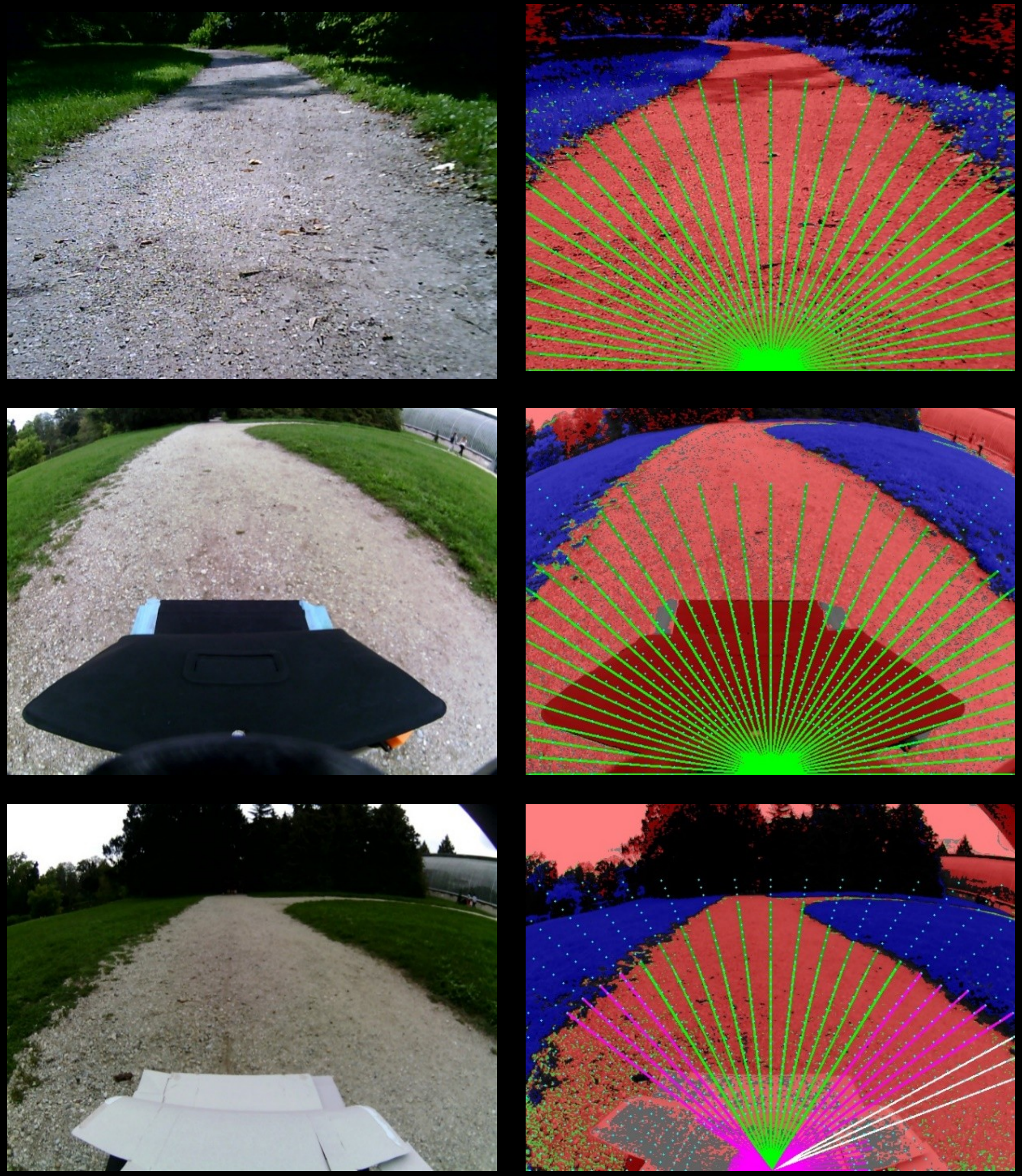
- **oCam : 5MP USB 3.0 Camera**
  - OmniVision OV5640 CMOS image sensor
  - Original lens Field Of View: 65 Degree
- Exchangeable Standard M12 Lens
  - Separate: 170 Degree Wide Angle
  - (standard accessory also for GoPro cameras)
- we 3d-printed an adjustable camera holder (fixed by screws)
- we broke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

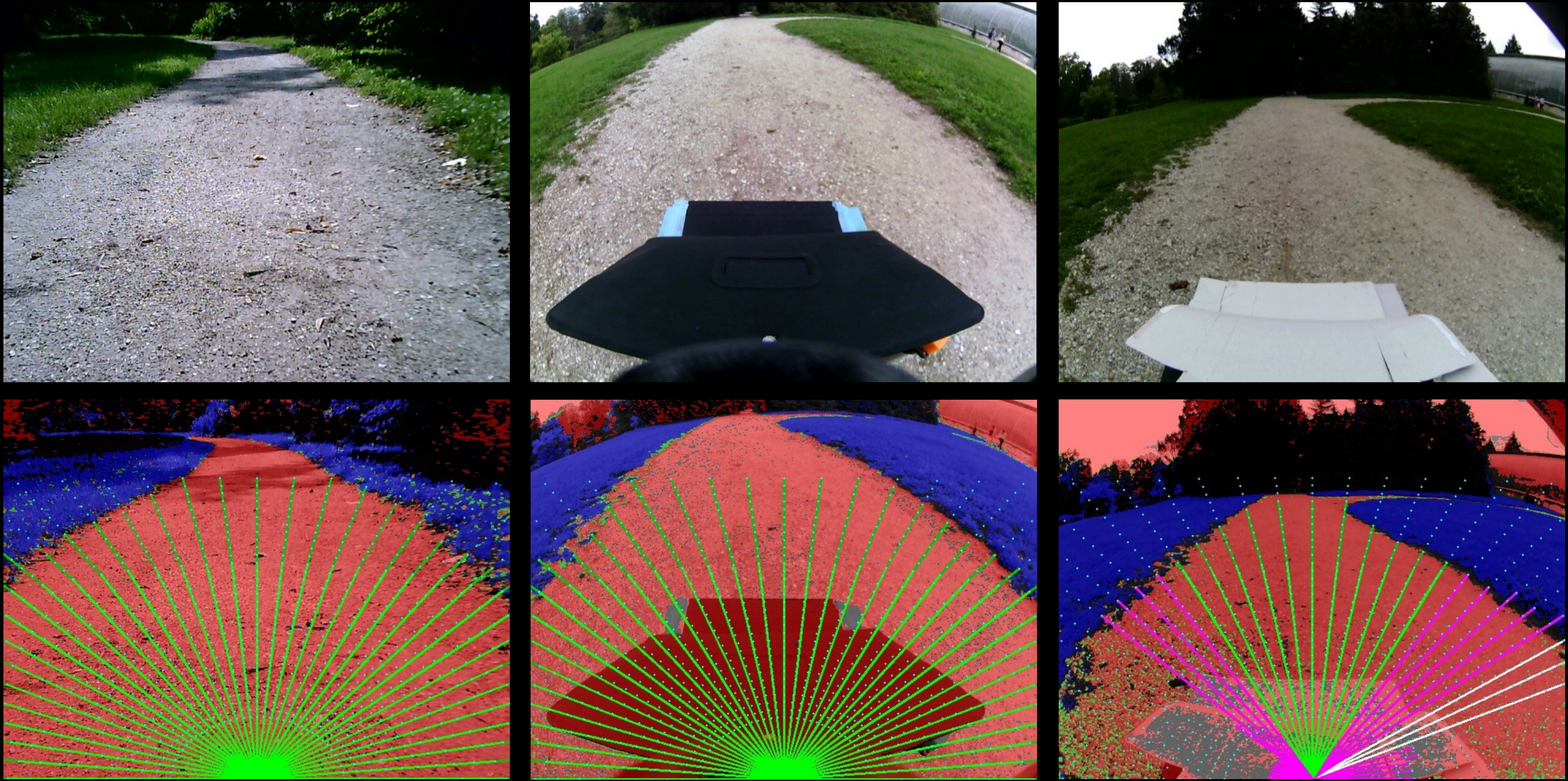
CAMERA CALIBRATION

- **Fisheye image warping correction**
  - OpenCV supports Camera Calibration
  - cv2.calibrateCamera()
  - cv2.undistort()
- Test in Python
  - execution times: 120-200 ms
- Not used because of the performance impact
- Better approach: correct only mapping between XY points and local map coordinates

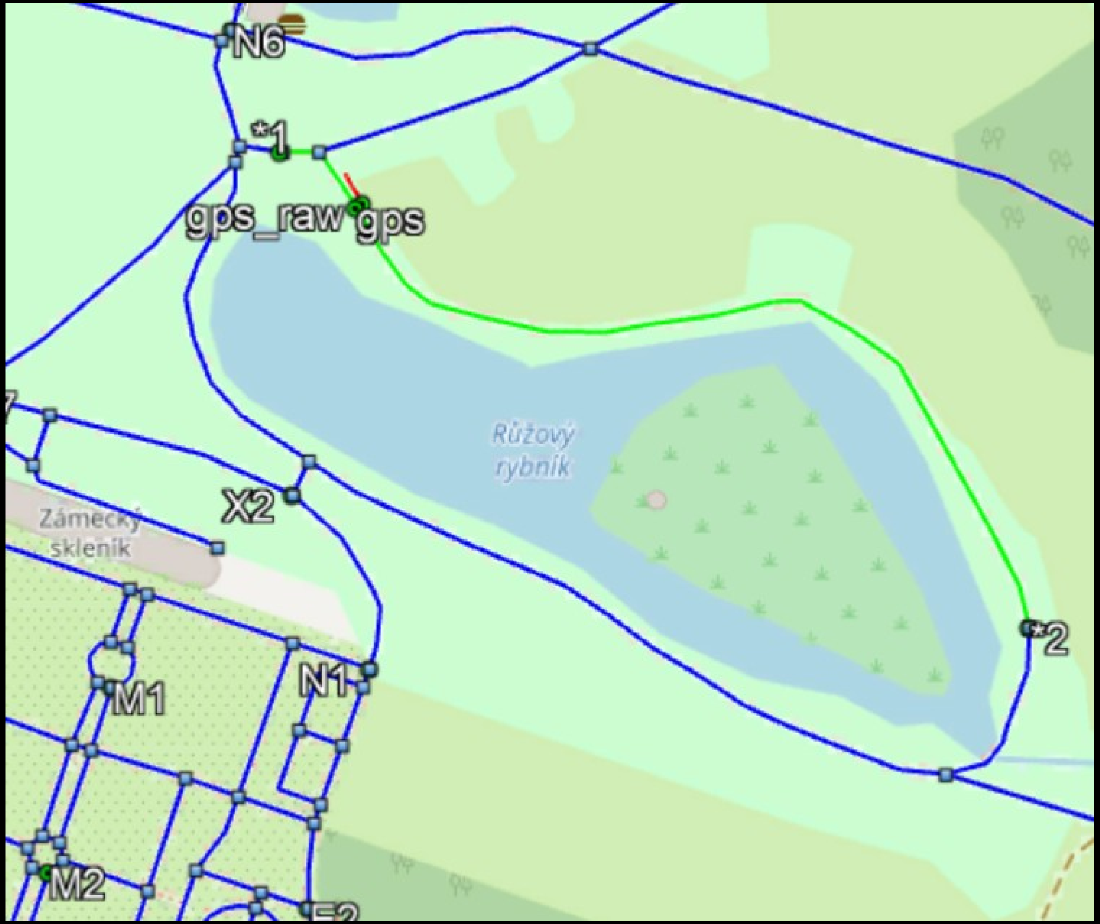
CAMERA ISSUES

- **Front side of the robot on every image**
  - problem to paint it with the “road” color
  - solution: to be fixed by SW masking
- **Image is too dark**
  - sun/sky causes high contrast images
  - our color maps did not work correctly
  - solution: use a shield to hide sky?
- **Roads appear too narrow**
  - algorithm was not able to use GPS navig. on roads
  - solution: local-map geometry to be corrected





NAVIGATION – ROUTE PLANNING



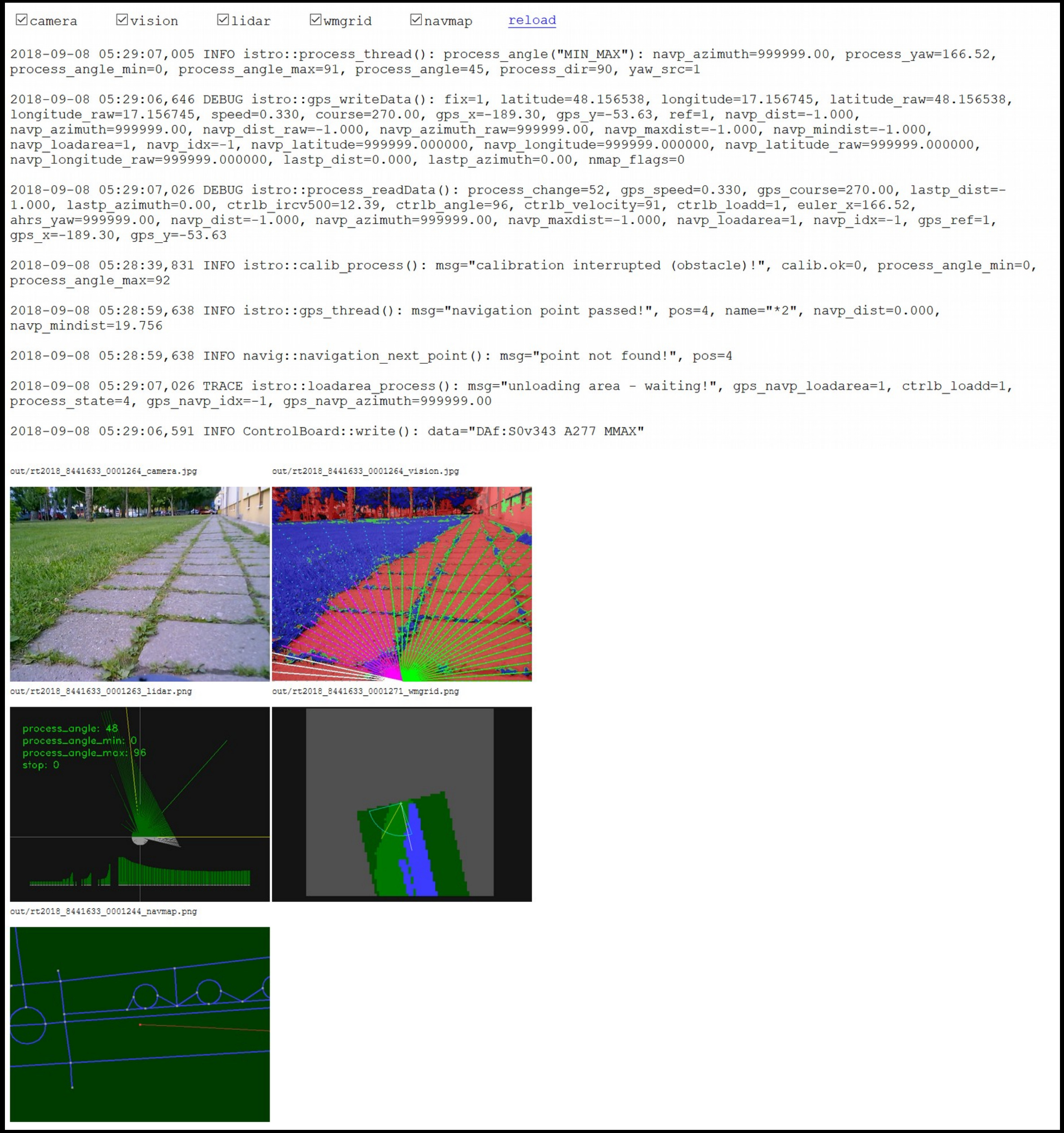
- OpenStreetMap data export
  - filter segments: footway, track with <grade3
- **Dijkstra's Shortest Path Algorithm**
  - 418 nodes, 504 segments
- Performance: < 2ms
- Visualisation: kml export, cpp export
- Navigation: keep azimuth towards a point that is 10m along planned route

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

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)



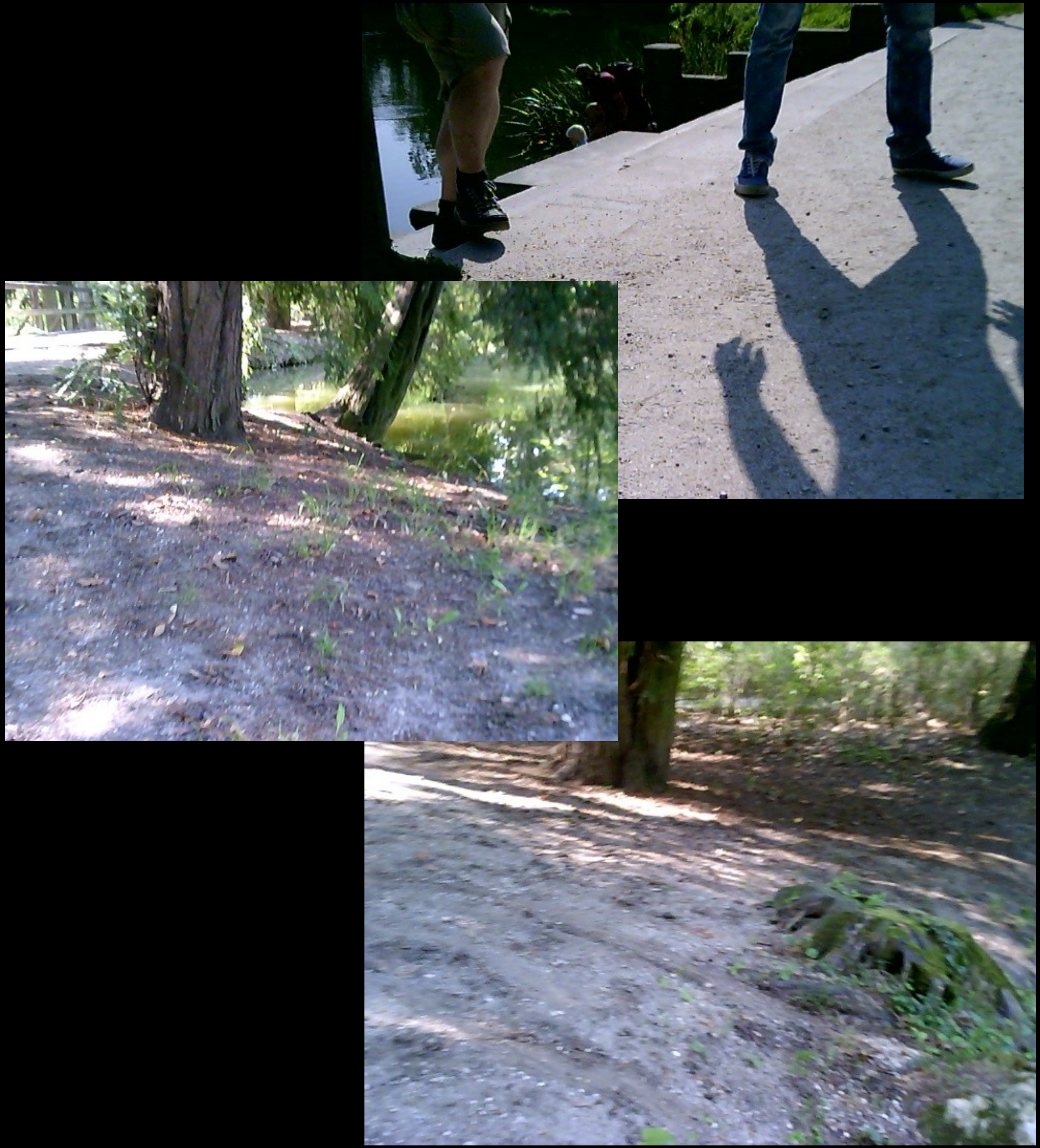
ROUND4 - DISK FULL

out_1034_zajko_kolizia.tar	242 660 kB
out_1046_round1_do_vody.tar	1 189 850 kB
out_1147_round2.tar	3 877 250 kB
out_1400_kalibracia_ok_nova_kamera.tar	223 170 kB
out_1402_kalibracia2_ok.tar	85 220 kB
out_1440_round3_nova_kamera.tar	3 518 490 kB
out_1453_round3_return.tar	982 100 kB

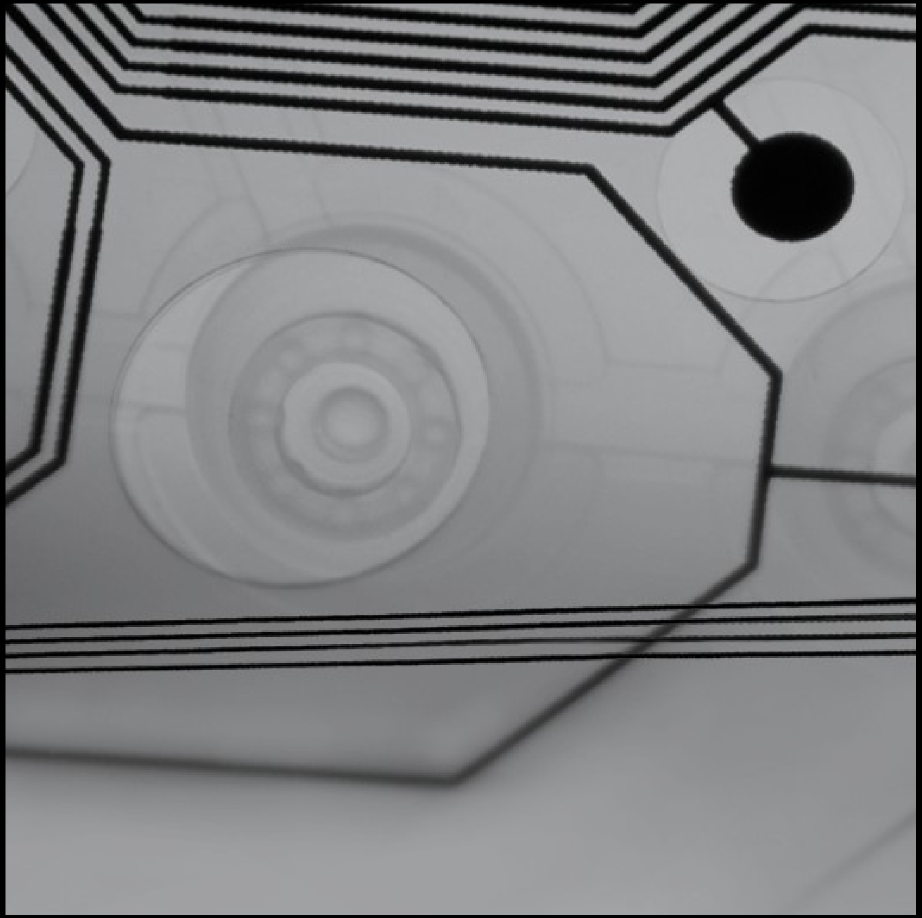
- Round4 failure: we run out disk space
  - Flash storage capacity: 64GB eMMC
- Storing to disk:
  - Log file
  - JPG, PNG, KML snapshots – 3-4 images/second
- **ROUND2 statistics**
  - 33 minutes, 28.000 files
  - Log file: 920MB (8.600.000)
  - Camera and vision JPGs: 6.500 each
  - Lidar and Grid PNGs: 6.000 each
  - Navigation KMLs: 1.500

PROBLEMS

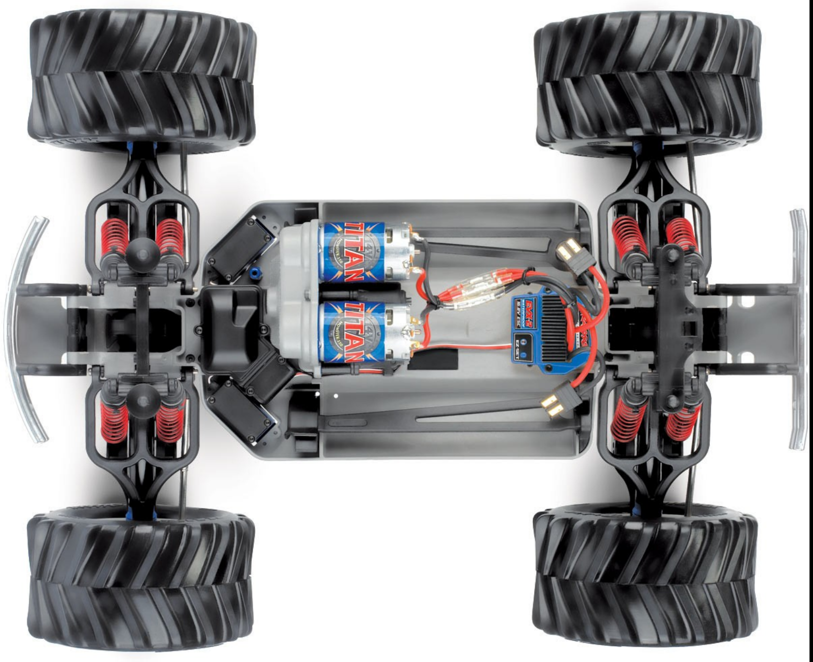
- ROUND1 (successful loading):
  - not able to make a turn-around near lake
- **ROUND2** (successful loading + unloading):
  - turning out of the road based on navig. azimuth (no grass)
- **ROUND3** (failed before loading), new camera:
  - camera was pointing too high
  - sometimes not recognising road at all
  - SW bug in “wrongway” algorithm
- **ROUND4** (successful loading + unloading):
  - issue with disk full terminated the application



GENERAL SLIDES

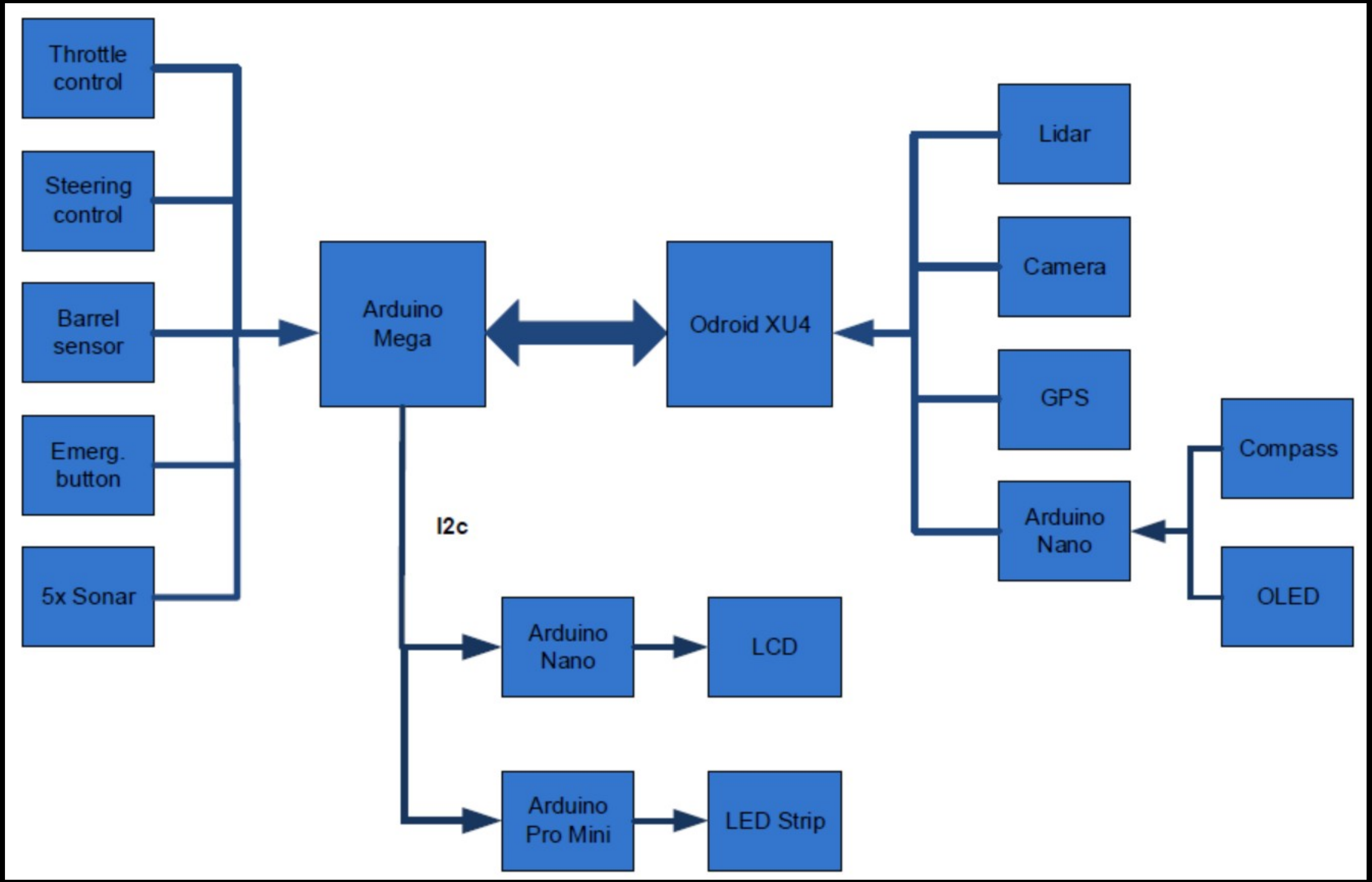


ROBOT CHASSIS

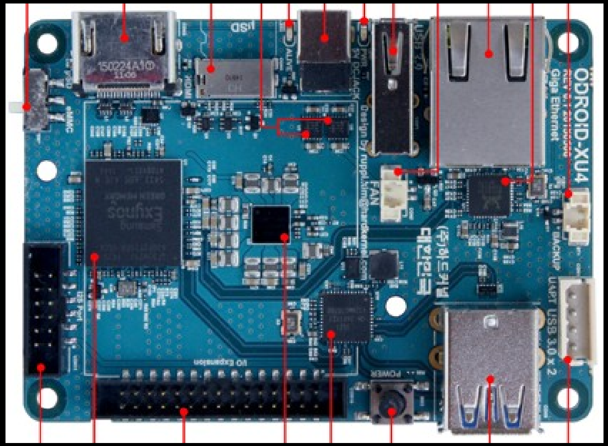


- RC model: Traxxas E-Maxx 4x4 monster truck
- Top Speed: 48 km/h
- Waterproof electronics, servos

HARDWARE DESIGN



HARDWARE

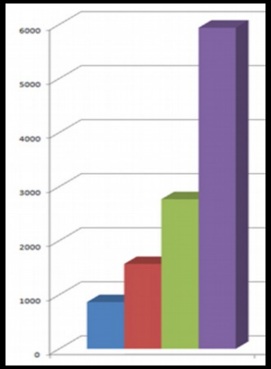


- **Odroid-XU4: 2GHz 8-core, 2GB RAM, 64GB Flash**
  - Arduino Mega: 16MHz, 8KB RAM
  - 2x Arduino Nano: 16MHz, 2KB RAM
  - Arduino Pro Mini: 16MHz, 2KB RAM
- 2D Lidar: RoboPeak RPLIDAR 360 ($400)
- Camera: Odroid USB Cam (640x480, 65 FOV)
- Mouse type GPS/Glonass: Holux M-215+
- Compass: Bosch BNO055
- **5x Sonar**: HC - SR04
- LCD & OLED displays, **8x LED**

Odroid-XU4 vs Raspberry Pi3

	Raspberry Pi3	Odroid-XU4
CPU	ARM Cortex-A53	Samsung Exynos5422 Cortex
Clock	1.2 GHz	2 GHz
Cores	4x	8x
RAM	1GB LPDDR2	2GB LPDDR3
Flash	microSD	eMMC5.0 HS400
Ethernet	10/100 Mbit	1 Gigabit
USB	4× USB 2.0	2x USB 3.0 1x USB 2.0

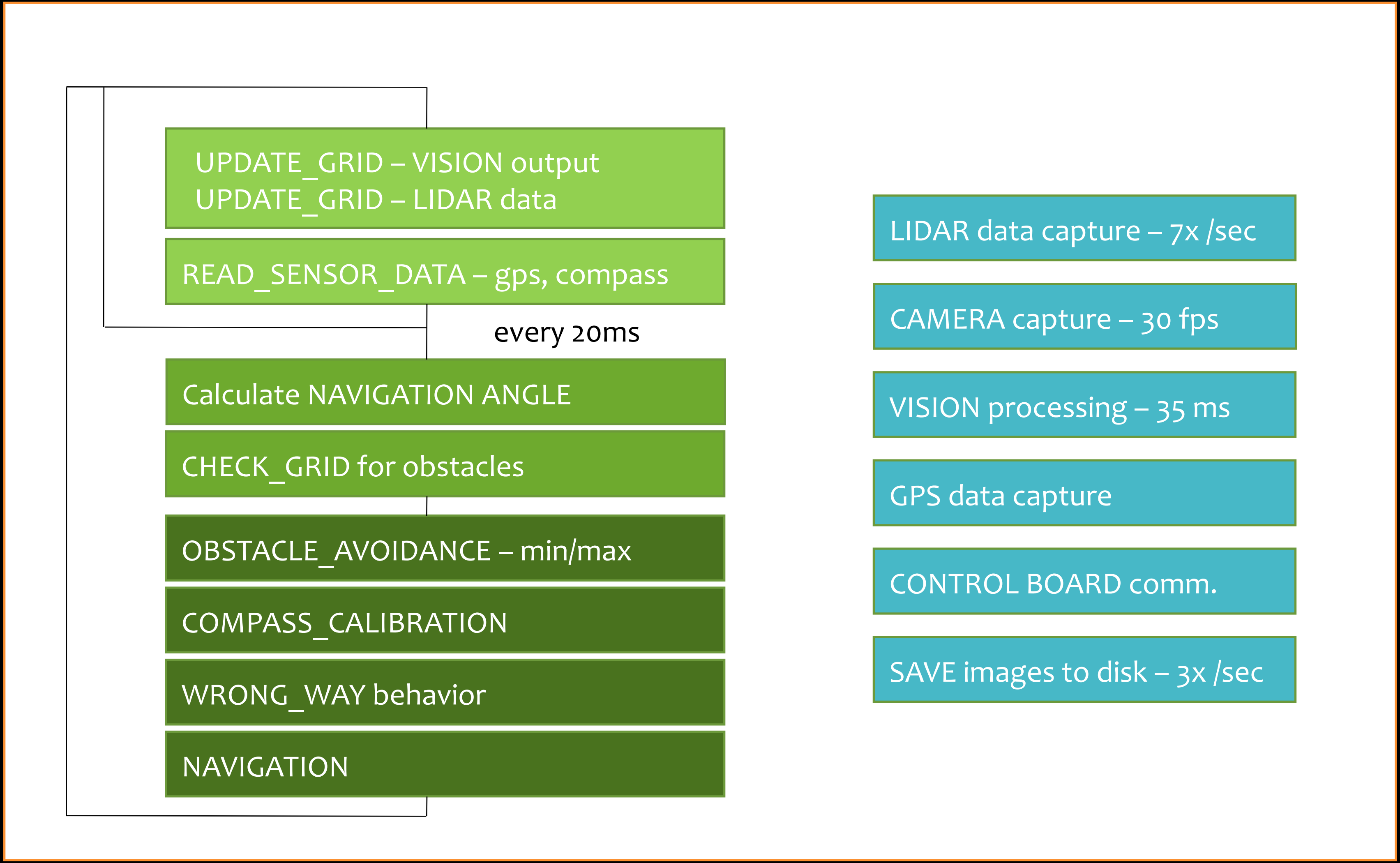
	R-Pi2	R-Pi3	O-XU4
Image processing	164,4 ms	167,3 ms	26,5 ms
JPG/PNG writing	-	-	3x faster
Processing lag	12 sec	2 sec	100 ms



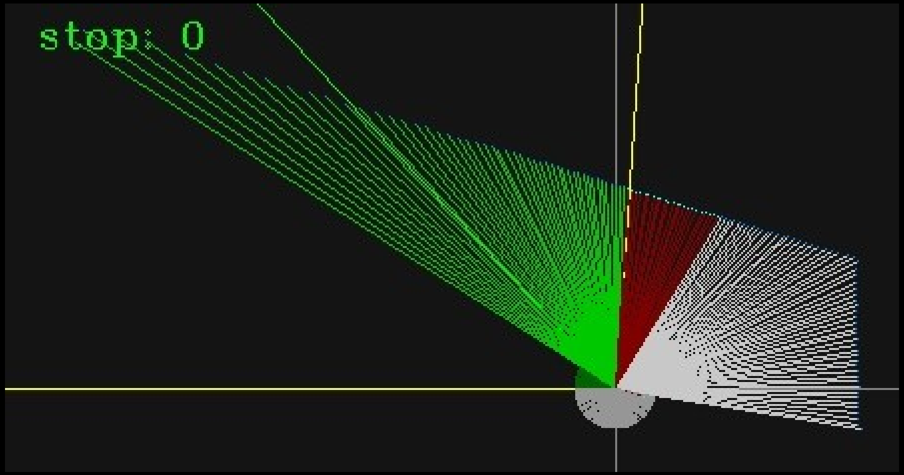
SOFTWARE

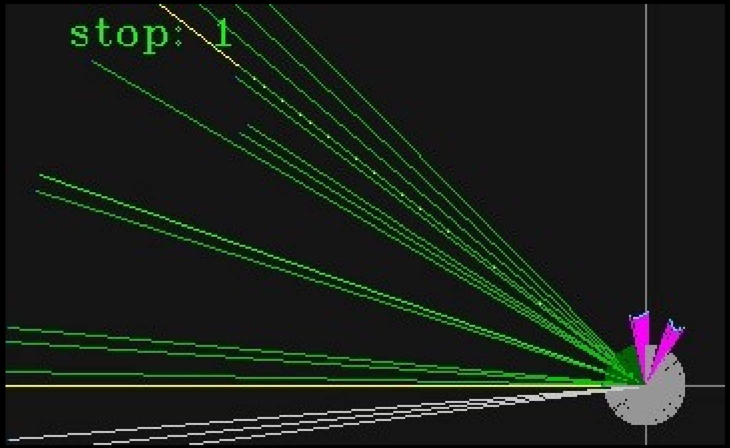
- Operating system: Ubuntu 16.04 Mate
- Source codes: C++, 430kB
  - 2017: 340kB, 2016: 180kB
- Libraries: OpenCV (vision), GeographicLib (Geo), Zbar (QR-Codes), Libxml2 (.osm), log4cxx (logging)
- Main application + 8x pthreads
  - 4x sensors (Camera, Lidar, GPS , Compass)
  - image capturing + vision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

SOFTWARE DESIGN – PROCESSING



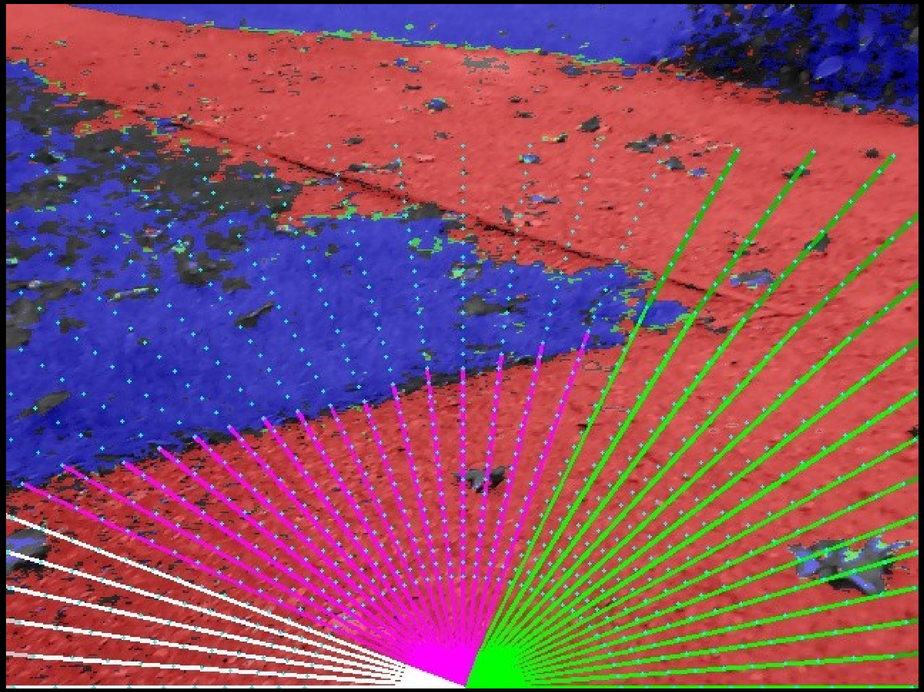
LIDAR – obstacle detection





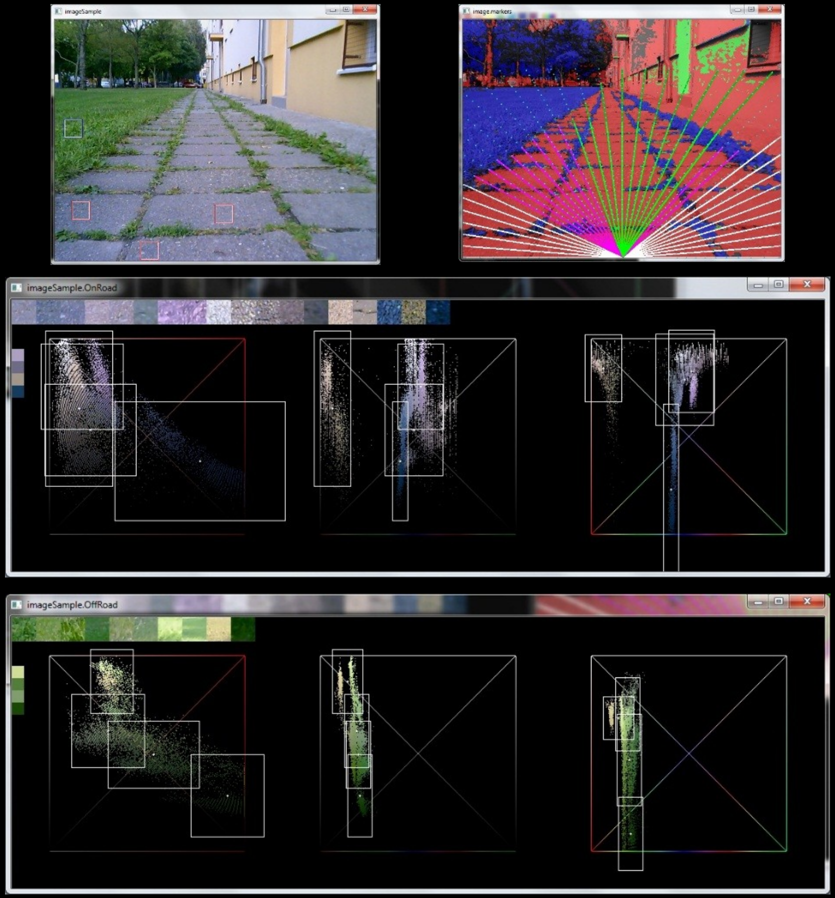
- Obstacle detection condition (red):
  - If distance is < 100 cm
  - Filtering: distance < 1cm (grey)
- Stop condition (pink):
  - Check angle: -45 to +45 degrees
  - If distance is < 50 cm at 3 diff. degs
  - Sonars were also used (rain issue)
- Obstacle avoidance (green/white)
  - Find OK intervals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

VISION – approach



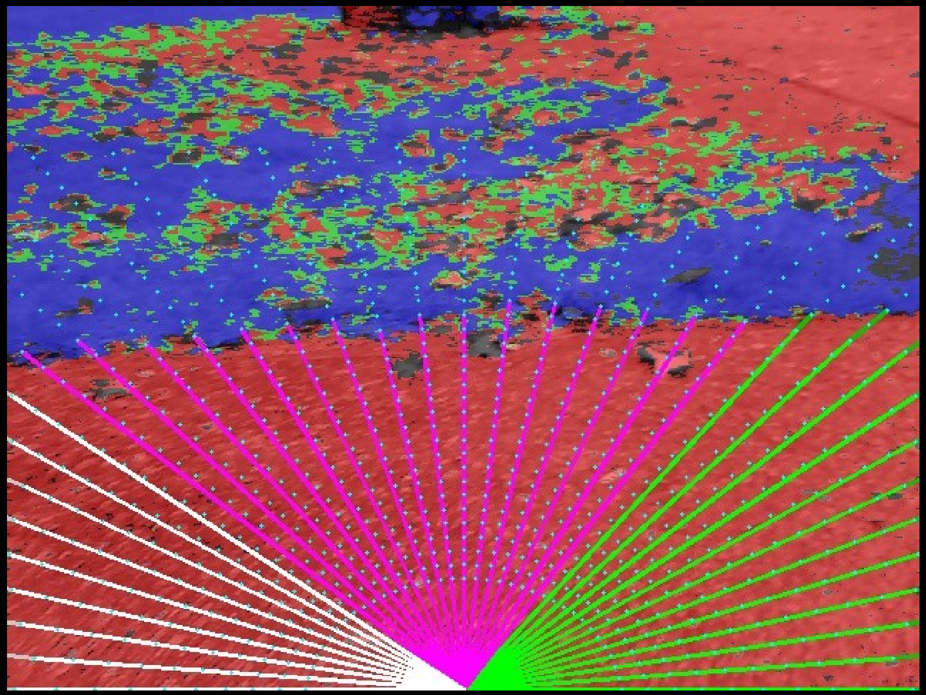
- Our approach: **lidar-like local map**
  - For any seen angle is obstacle closer than 1 meter?
  - 1 meter or to the image border
- Algorithm:
  - Pixel color classification
  - Evaluate grid points
  - Calculate distance to obstacle
  - Find OK intervals – same like LIDAR

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

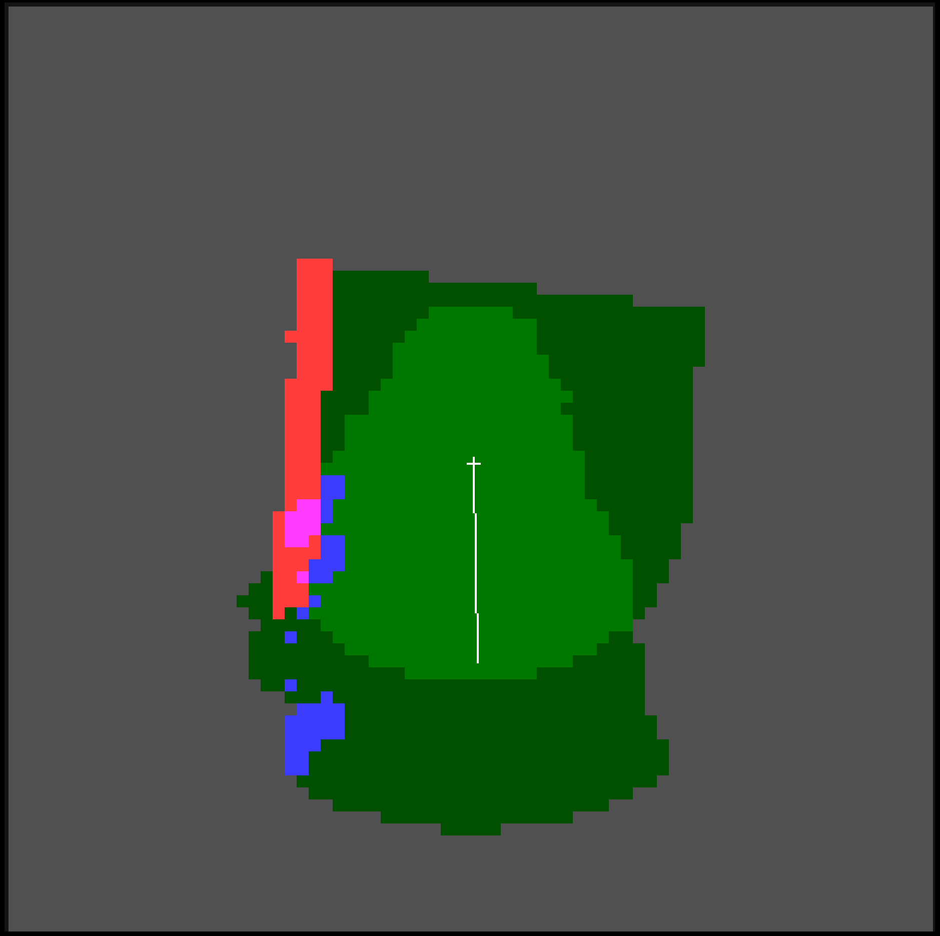


- Pixel color classification - 4 results:
  - Road (red)
  - Off-road (blue)
  - Both (green)
  - None (grey)
- Evaluate grid points
  - Cca 1000 points in 37 lines (5 deg)
  - Evaluating nearby pixels (80x80)
  - Majority of “Road” pixels is checked
- Calculate distance to obstacle
- Find OK intervals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

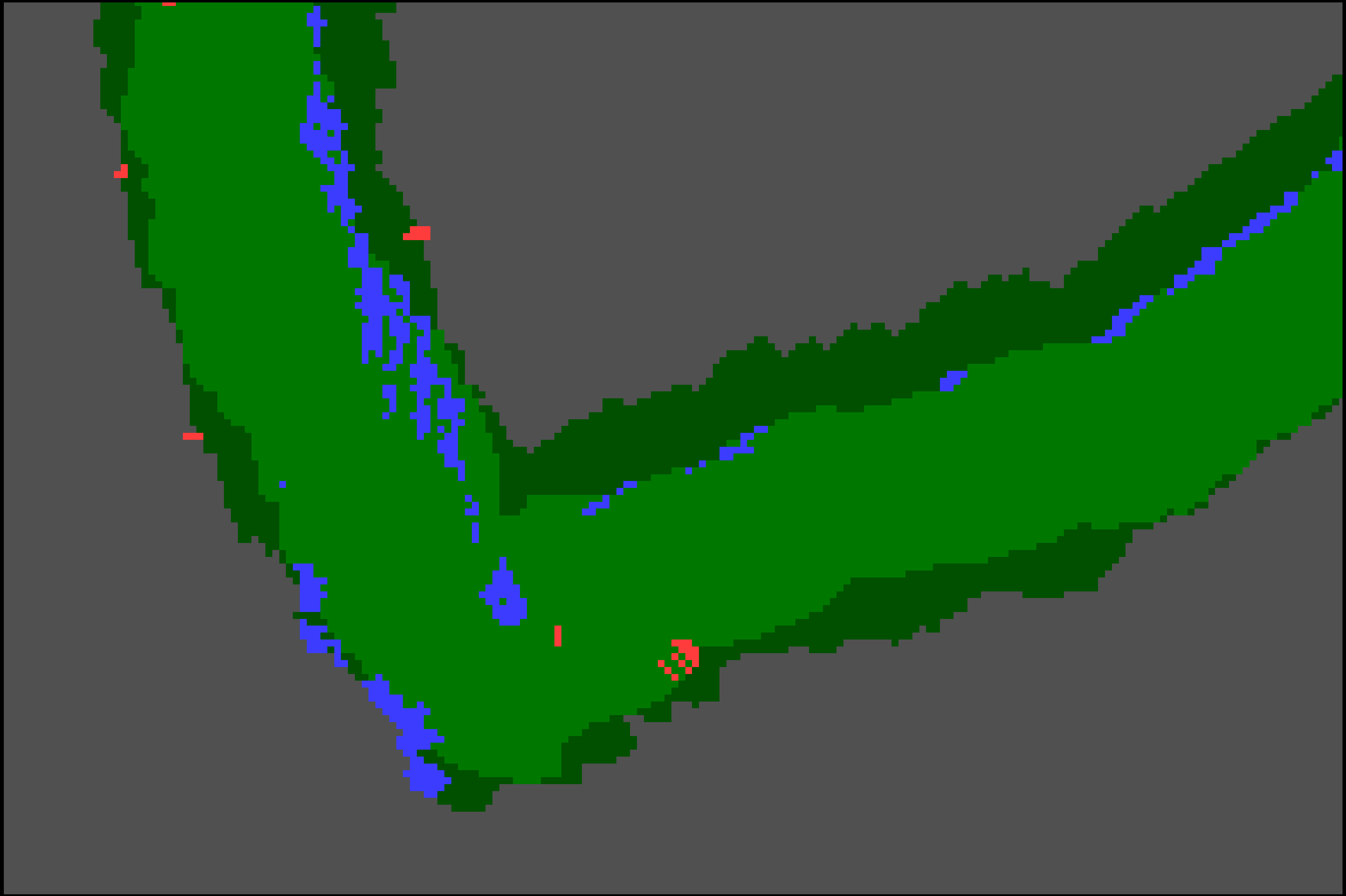
WAYPOINT NAVIGATION

- Navigation points: 26 manually defined points
- GPS positions taken from OpenStreetMap
- Pickup and dropoff points in arrays (navig.cpp)
- Navigation path – defined by a string
  “N2N1 *1 M4M5N7M3M8 *2 M7M2M3N7M5N3”
- Compass calibration:
  - interval of 7 seconds - robot is moving straight
  - gps and compass azimuths to be fixed
  - performed every 30 seconds – 3 minutes

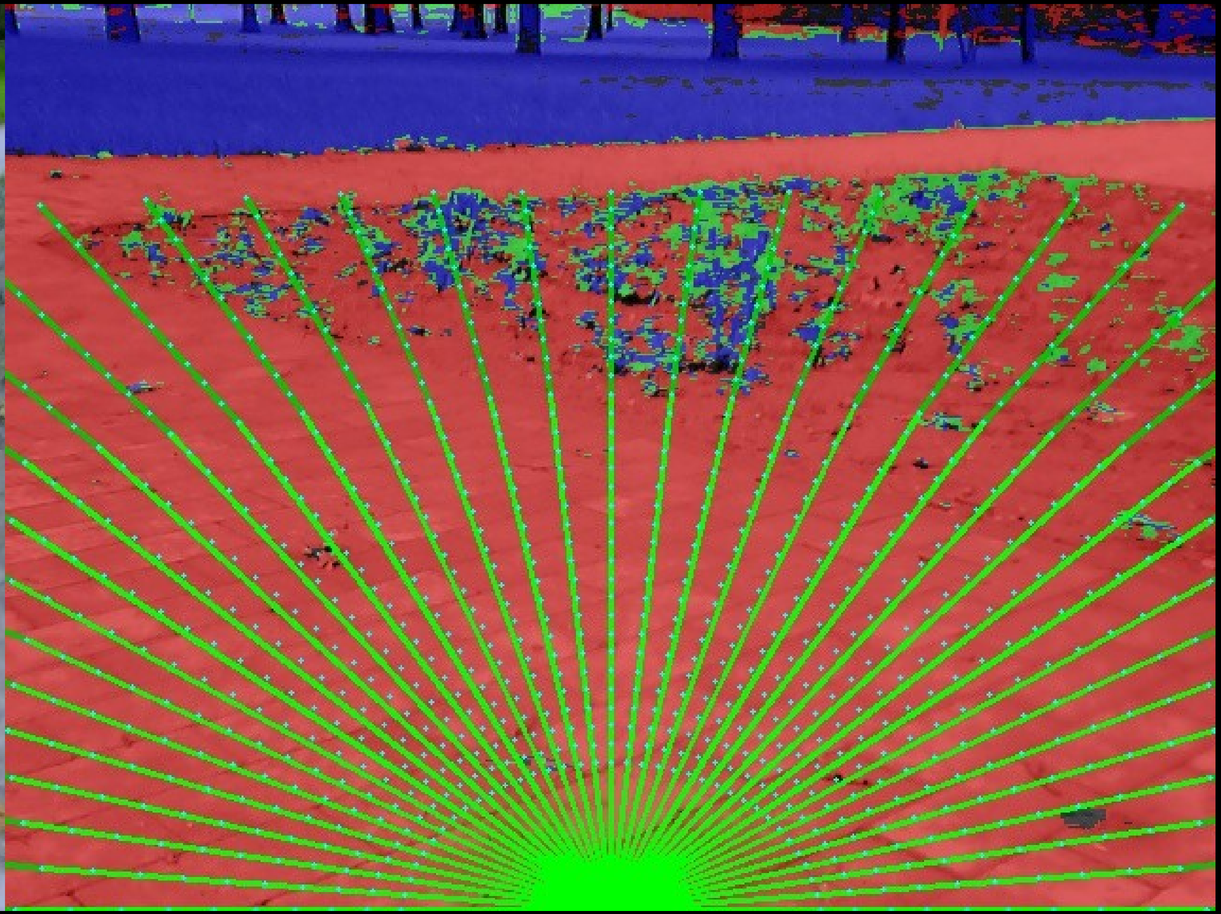
WORLD MAP – BUILDING LOCAL GRID



- Local grid map
  - to store polar information from lidar and vision
  - 1 cell: 10 x 10 cm, 1 byte pre 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)



PROBLEMS – VISION FAILS DETECT ROAD



FREE SOURCE CODES

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

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

THANK YOU