FALCONS
supported by ASML

Falcons team update

Presentation Portugal Workshop 2015

Contents

- ASML’s mission for Robocup
- Software updates
- New vision system

FALCONS

supported by ASML

Mission

Vision for 2015-2016

- Inside ASML there is a growing awareness for the Falcons team, resulting in request to demonstrate the robots on job fairs and schools
- The Turtle 5k hardware is not used to its limit yet and winning multiple games should be possible with current hardware
- New hardware should be available before current hardware becomes the limit

This results in:

- Creation of a new demo team with dedicated demo software
- Optimization of current hardware and software, e.g ball handling and shooter/kicker
- Writing and execution of test plans
- Start development new hardware

Team constraints

- All work is done off-hours (evenings) with volunteers
- Team consists of approx. 34 volunteers
- Team looks big, but is “only” 5 FTEs

Upcoming activities

- Dutch open in March 2016
- About 9 demo events in H1 2016
- Mini games against TU/e and VDL robot sports
- WM in Leipzig

FALCONS

supported by ASML

Software updates

Philosophies to program by 1/2

- Share code with other teams
- KISS-principle
- Be future proof
  - ROS interface and types decoupling
- Behavior-driven development
  - Make use of continuous integration

Philosophies to program by 2/2

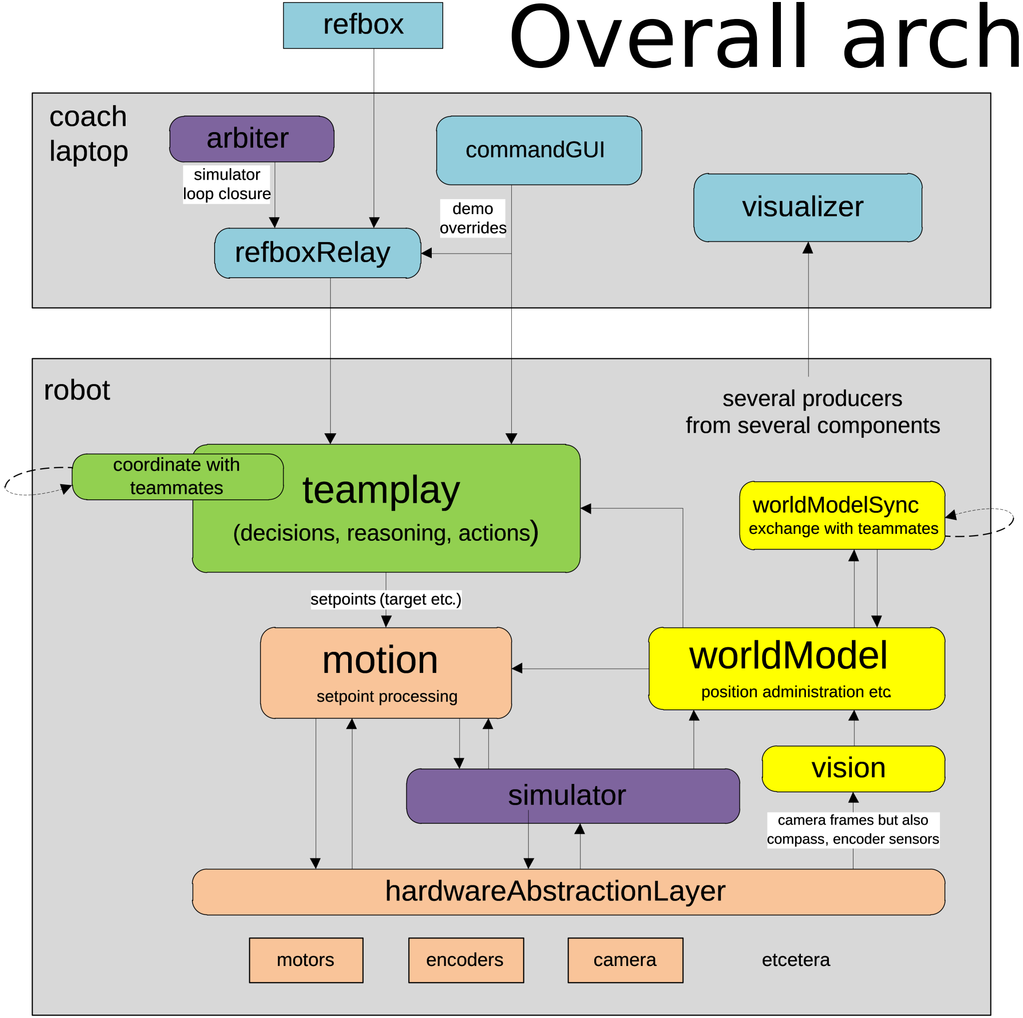
- Create software that is
  - Scalable
  - Maintainable
  - Testable
- Actively rework and refactor
- Document code with DoxyGen
- Maintain wiki pages

General technical notes

FALCONS supported by ASML

what	why	status
Migrated SVN to GIT	Development efficiency (branching)	**Done!**
Upgraded to Ubuntu 14 and ROS jade	Use latest and greatest packages	**Done!**
Isolate SW interfaces (ROS, UDP) from internal library	Future proof; improved code re-usability	**Mostly done!**
Reduce SW latency	Improved responsiveness	In progress
Replace python with C++	Improved performance / reduced CPU load	**Mostly done!**
Rewritten firmware from scratch	To solve China “drunken robots”; code maintainability; adding new features	Testing
Planning to use Gazebo	Replace custom with advanced out-of-the-box simulation; visualization capabilities	Not started

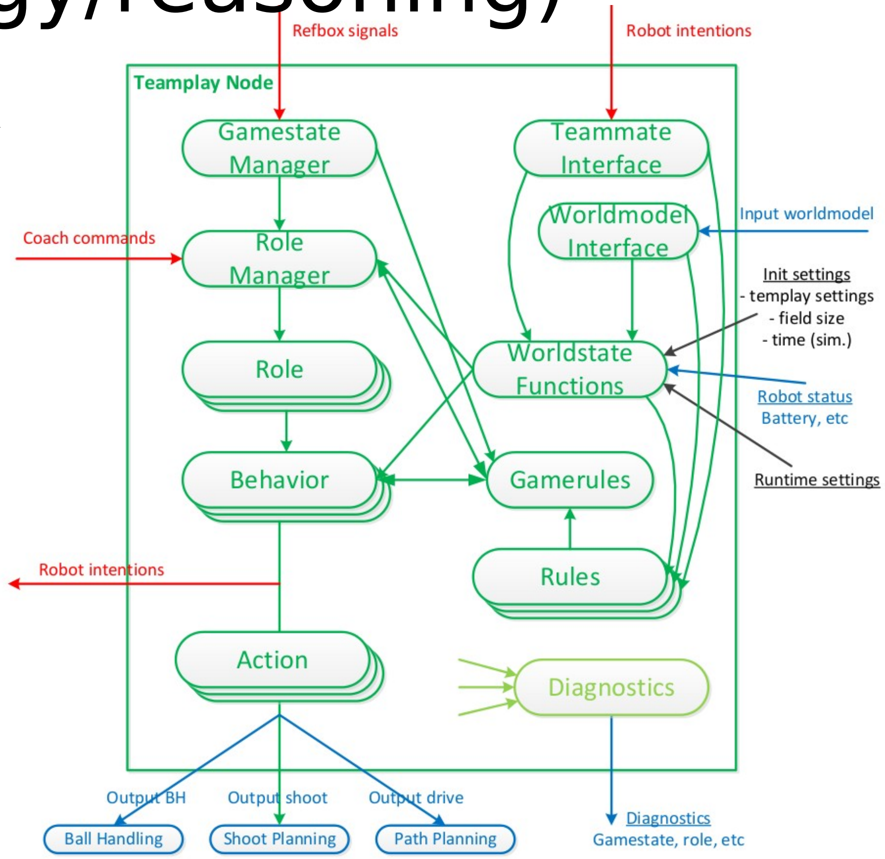
Overall architecture



FALCONS
supported by ASML

Teamplay (strategy/reasoning)

- Inspired by CAMBADA’s strategy software architecture
- Teamplay module (node) decoupled from ROS; therefore interchangeable in future
- Added layering to prevent code explosion
- Naming in line with other teams
- Convert Teamplay code from old (Hefei) to new (Leipzig)



Refbox signals

Robot intentions

Teamplay Node

Gamestate Manager

Teammate Interface

Worldmodel Interface

Input worldmodel

Coach commands

Role Manager

Init settings
- templay settings
- field size
- time (sim.)

Role

Worldstate Functions

Robot status
Battery, etc

Behavior

Gamerules

Runtime settings

Robot intentions

Rules

Action

Diagnostics

Output BH

Output shoot

Output drive

Ball Handling

Shoot Planning

Path Planning

Diagnostics
Gamestate, role, etc

FALCONS

supported by ASML

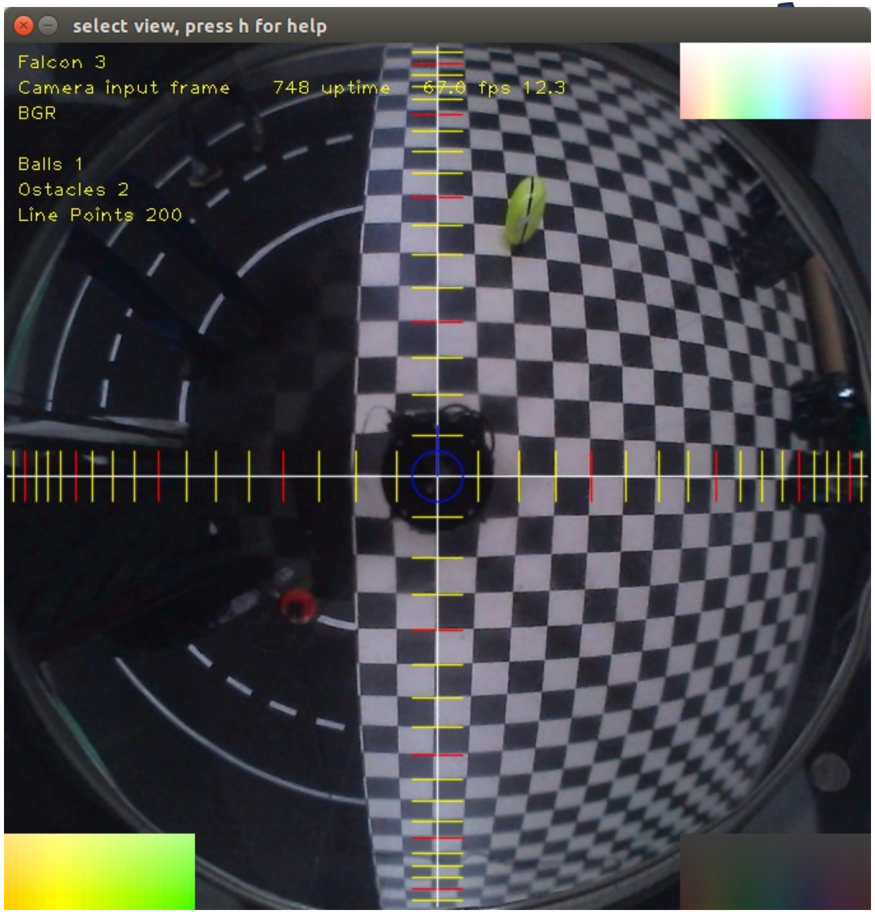
Vision system

Vision system - Contents

- Turtle5k vision
- Requirements
- Standard solution
- Do it yourself
- Proof of concept
- Conclusion
- The future

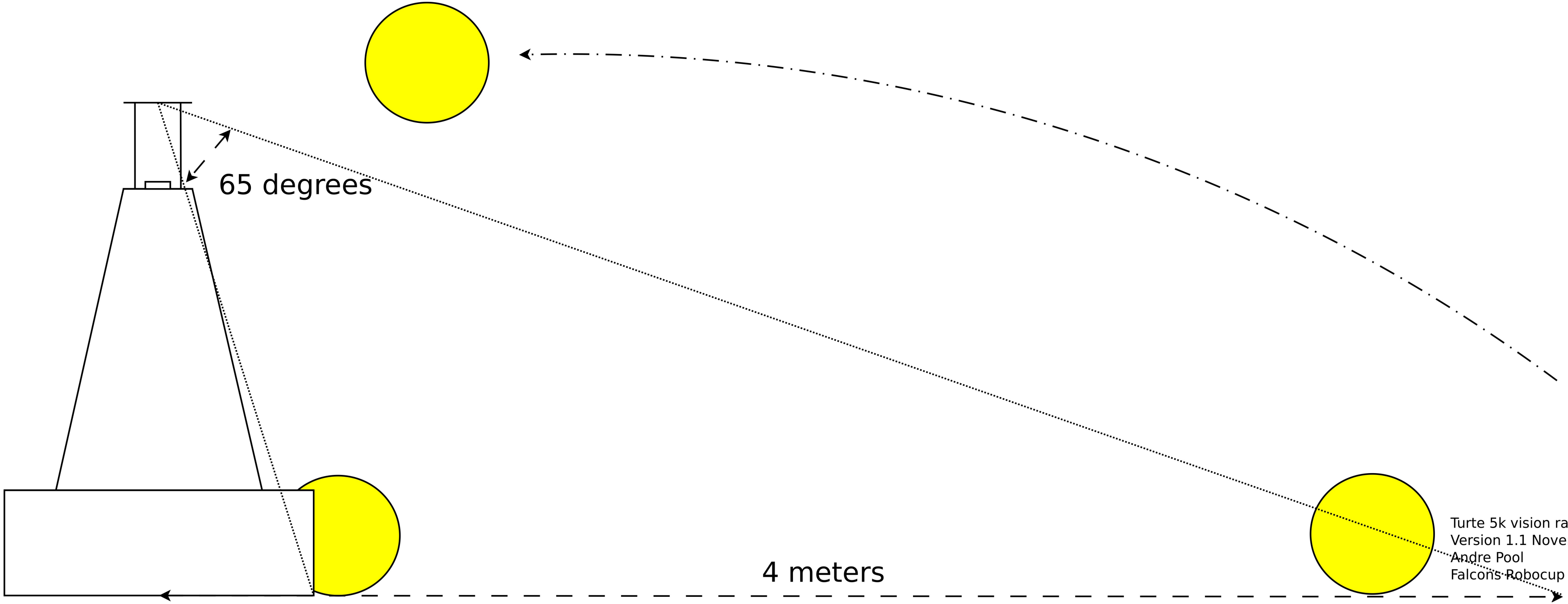
Turtle 5k vision

- Field 18x12 meters
  - Camera radius about 4.5 meters
  - Edge color distortion from tube
  - Edge color distortion from mirror
- Only down view
  - Not able to track lobbing balls
- Inefficient use of pixels
  - 16:9 to circle view
- Difficult to calibrate
  - No homogeneous tangent function
  - Optical/mechanical
- Difficult to determine distance



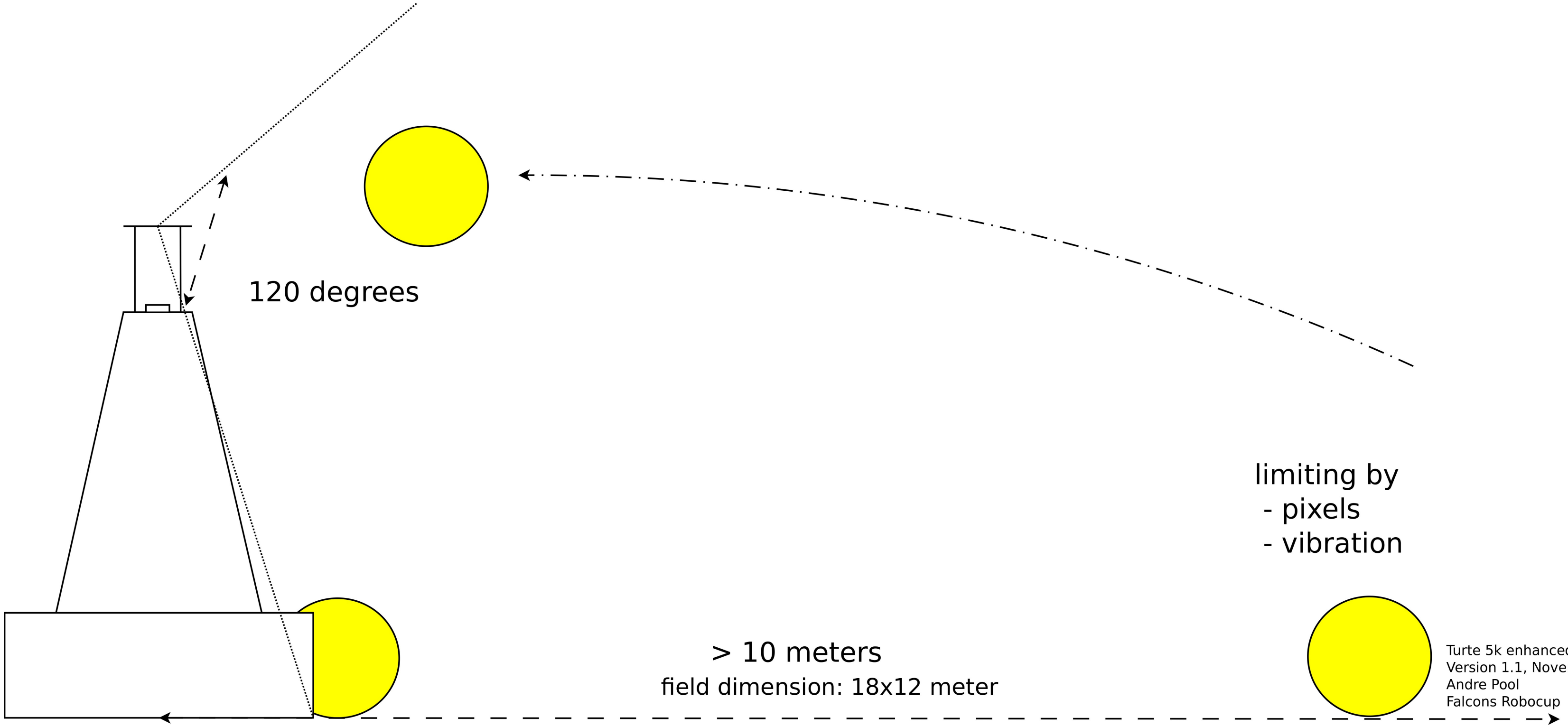
```
Falcon 3
Camera input frame    748 uptime  67.0 fps 12.3
BGR

Balls 1
Ostacles 2
Line Points 200
```

Down view



Turte 5k vision range
Version 1.1 November 2015
Andre Pool
Falcons Robocup

Increased view



Turte 5k enhanced vision range
Version 1.1, November 2015
Andre Pool
Falcons Robocup

Requirements

- 360x120 degrees
- 60 FPS
- Resolution 800x600 per camera
- Low latency less then 5ms
- Synchronized (global) shutter less then 1ms
- Low optical distortion (< 80 degrees per camera)
- Total size assembly (h<15cm, d<20cm)
- Communication Interface
- Linux / OpenCV compatible
- Availability
- Affordable (less then €1000 per robot)

Standard solution

Most of them are

- Large
- Expensive
- No streaming (SD card)
- Limited vertical view
- Closed design

Panone 360x360

- 36 camera's (45 degrees each)
- €1499, streaming expected in 2018

Sphericam 360x360

- 6 camera's (90 degrees each)
- 60 FPS
- Pre-order $1999, expected January 2016

Do it yourself

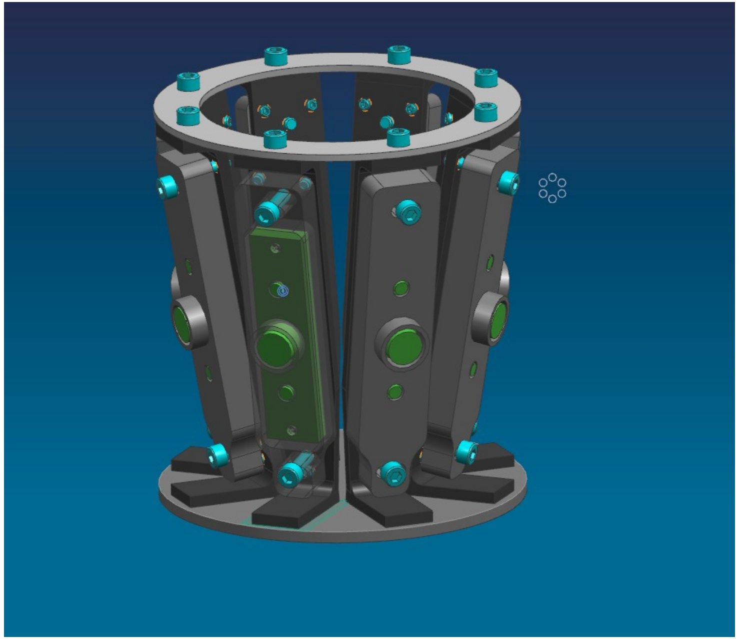
Standard x86-64 interface

Genius Widecam F100

- 1080p @ 30 FPS
- 120 x 65 degrees
- 8 x 45 = 360 degrees (vertical)
- 8 x €40

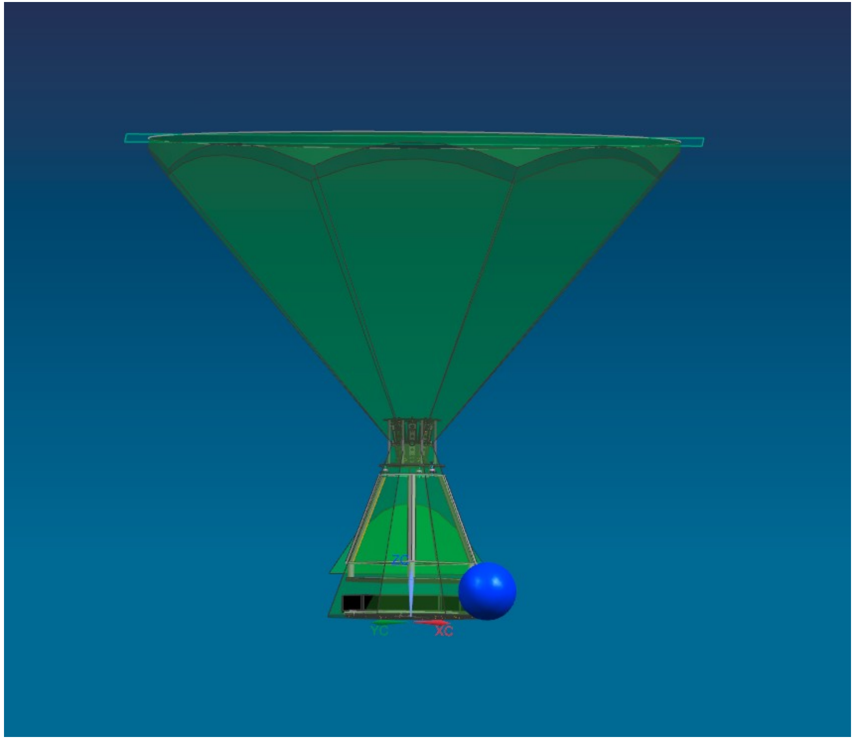
Unknowns

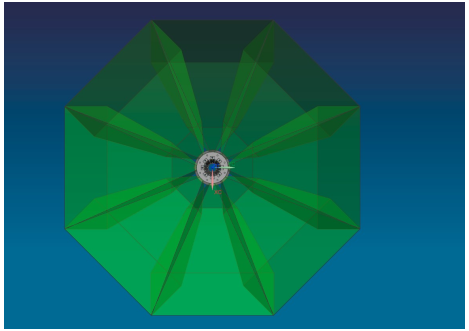
- Bandwidth (USB 2.0)
- Latency
- Shutter control
- 30 FPS
- Camera control from Linux

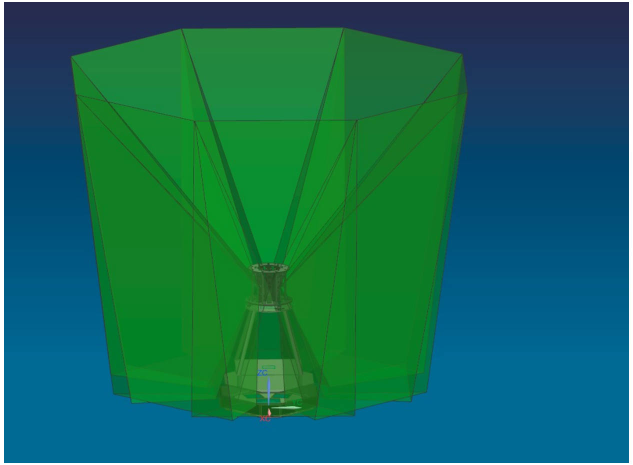


Visible area

FALCONS
supported by ASML







Proof of concept

Hardware

- 2x Genius Widecam F100
- 3x Logitech C525
- 3x Streams
- 4 x i7-3540m @ 3GHz

Synchronization

- Each camera/stream in thread
- Use create – join to synchronize

Results

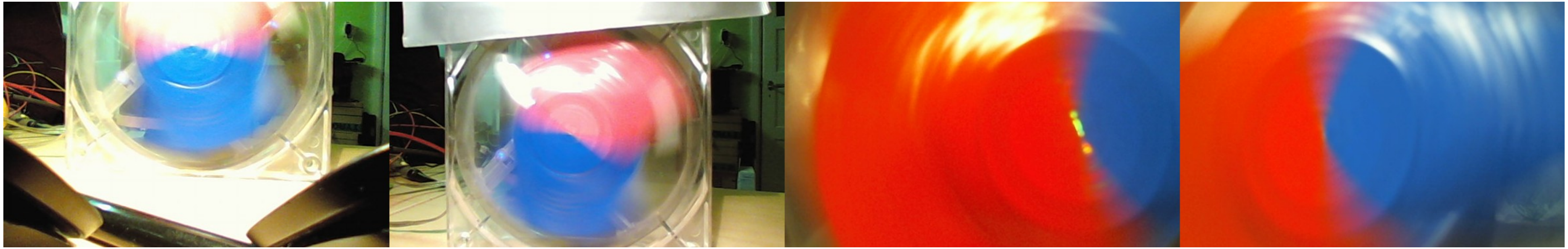
Success

- 5 camera's 800x600
- 30 FPS
- Camera control

Fail

- Maximal 1 camera per USB hub
- Synchronization error > 5ms

Synchronization fan test



Shutter synchronization

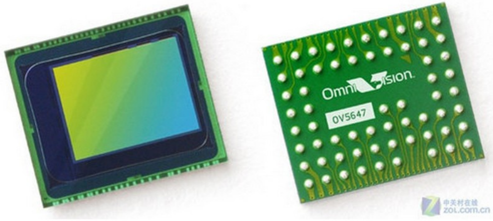
- 250ms per turn
- 250*20/360 = 14ms

Conclusion

- Closed design
- Dead end

The future

Smart-phone camera solutions



OmniVision

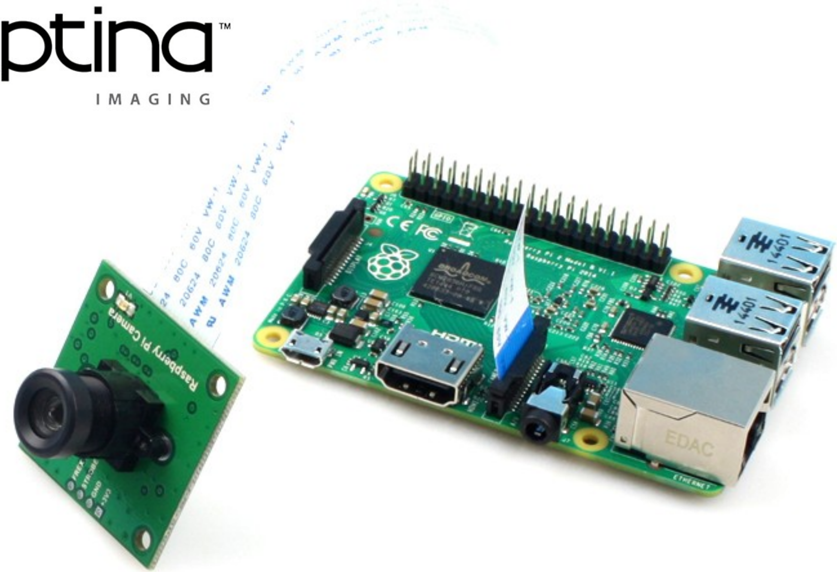
Aptina IMAGING

Raspberry Pi camera

Ardocam added shutter control

Multiple camera's to FPGA

Stereoscopy



Questions