The Hera framework
for
fault-tolerant sensor fusion on an Internet of Things network
with
application to inertial navigation and tracking

Sébastien Kalbusch
Vincent Verpoten

25/06/2021

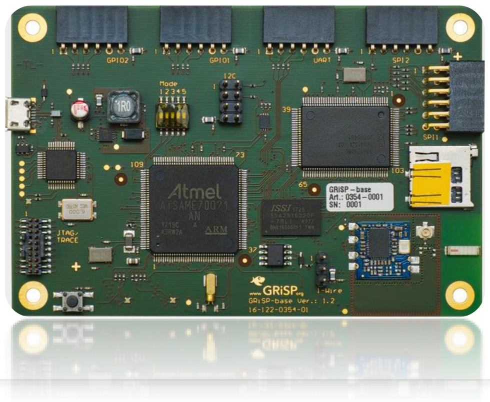
EPL21-241

Supervisor:
Peter Van Roy

Introduction

Context and motivation

Context



Purpose

Development of a sensor fusion platform at the edge

on GRiSP board

Concepts

- Sensor fusion
- Internet of Things, edge
- Fault tolerance
- Inertial navigation, tracking

Motivation

<u>Sensor fusion on low-cost platforms</u>

- Close to hardware, datasheet, soldering
- Low-level language (C)

Complex and error-prone

<u>Internet of Things infrastructure</u>

- Cloud computing
- Fog computing

Complex and expensive

<u>Erlang, GRiSP, Hera</u>

- Edge computing
- Low-cost, but high-level
- Focus on sensor fusion model

Easy to use, fault-tolerant

Outline

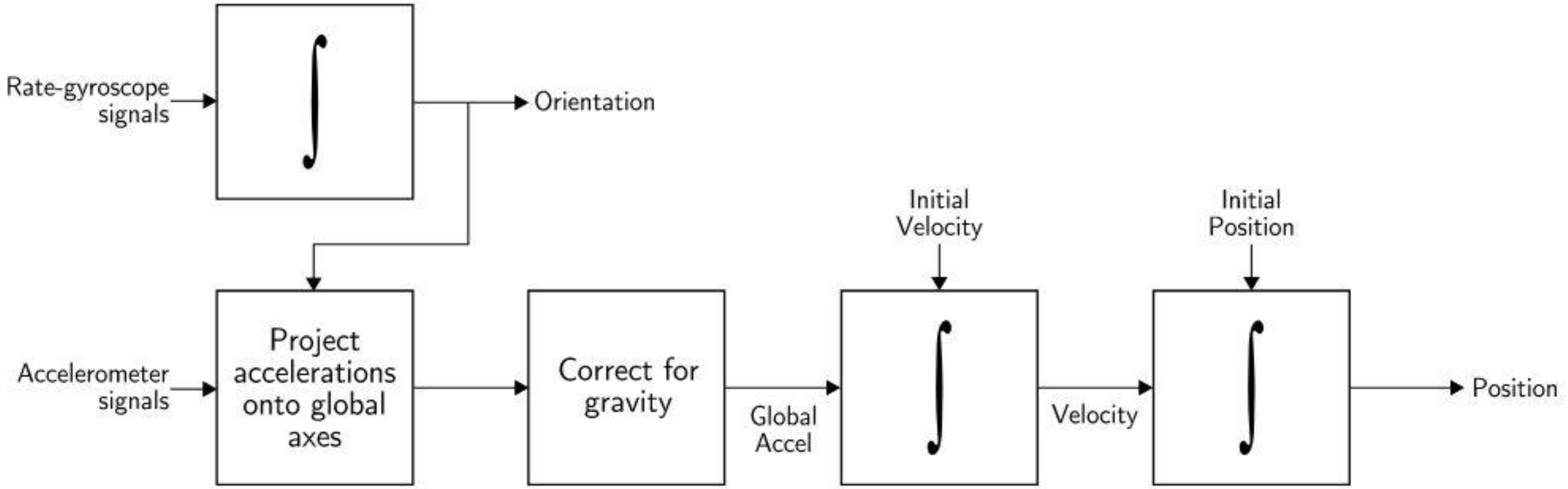
1. Inertial navigation and tracking
2. Software architecture
3. Fault tolerance
4. Experimental sensor fusion with Hera
5. Attitude and heading reference system
6. Verdict and future work

Inertial navigation and tracking

Principles

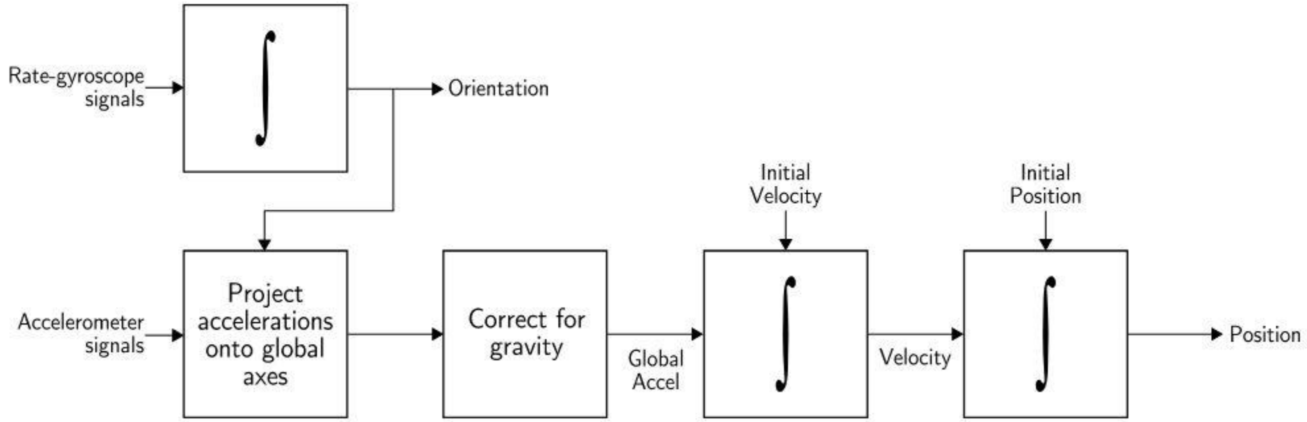
Strapdown inertial navigation

Pmod NAV



Oliver J. Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, 2007.

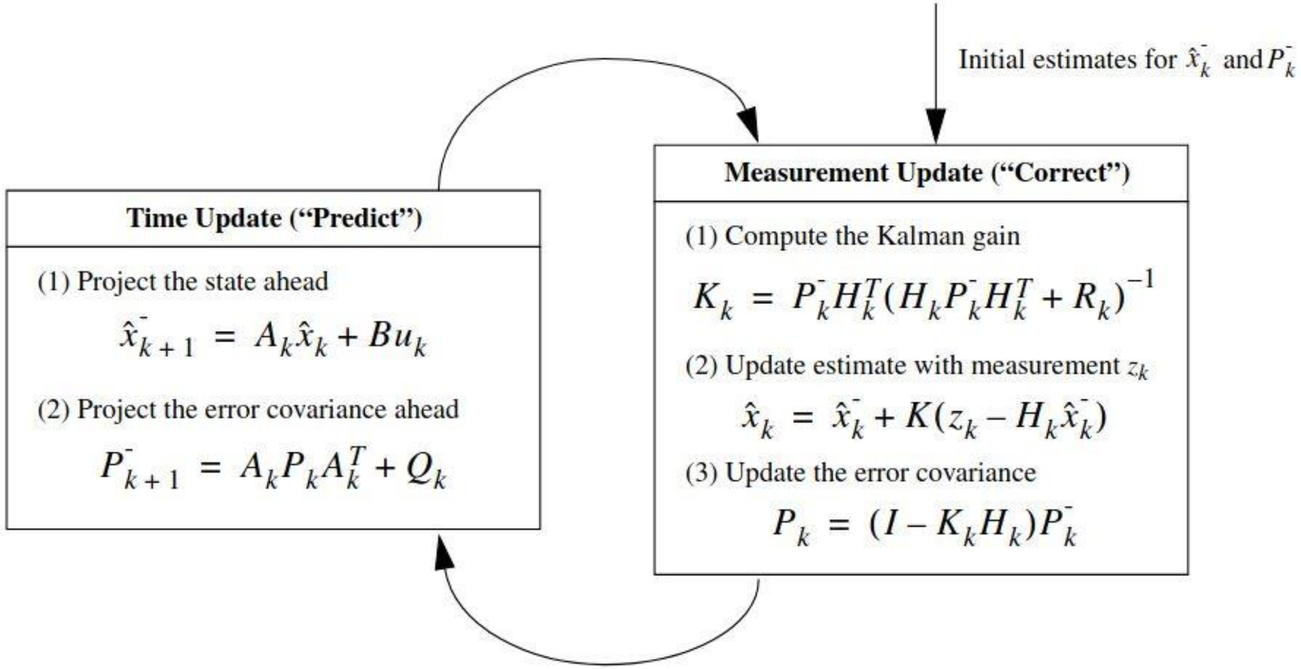
Inertial navigation is subject to drift



$$p_t = p_{t-1} + v_{t-1}\Delta t + \frac{1}{2}a_{t-1}\Delta t^2$$
$$v_t = v_{t-1} + a_{t-1}\Delta t^2$$
$$a_t = a_{t-1}$$

Drifts very quickly !

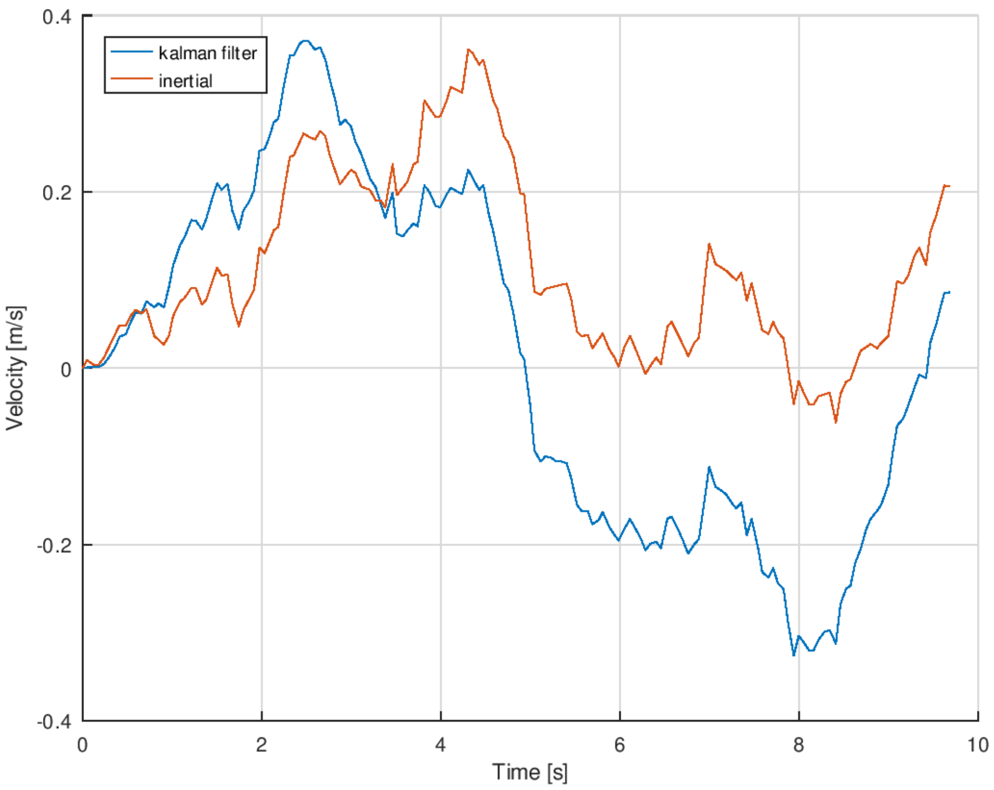
The Kalman filter: a solution to avoid drift

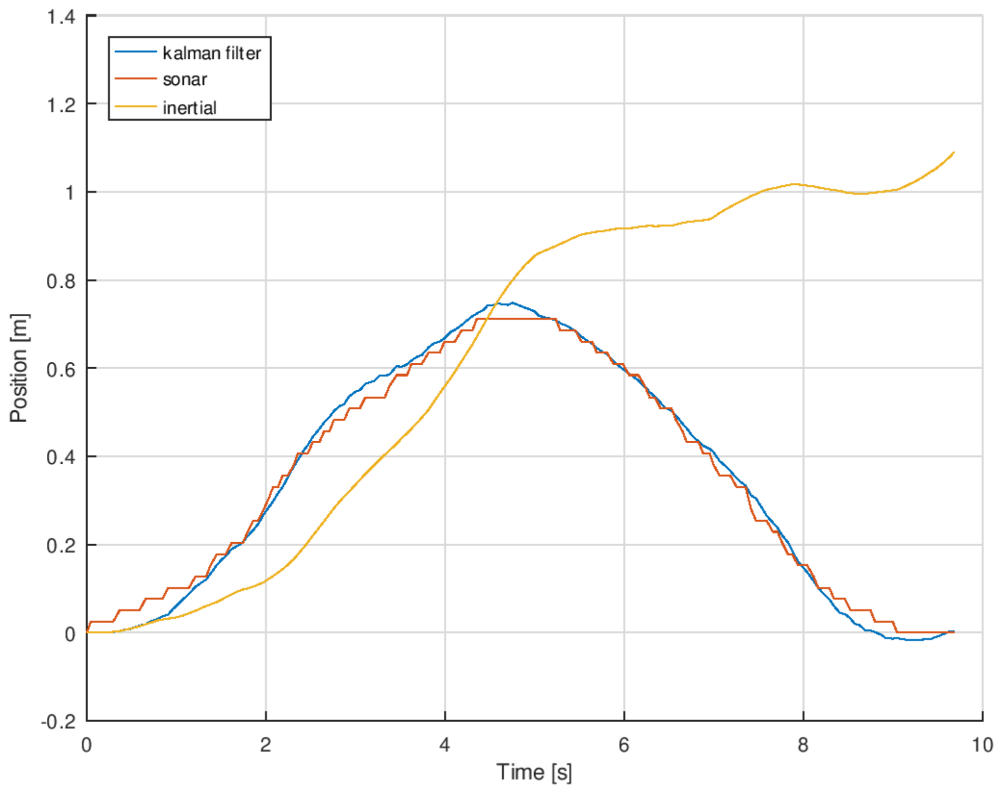


Greg Welch and Gary Bishop. An introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, 1995.

Pmod MAXSONAR

The Kalman filter: a solution to avoid drift

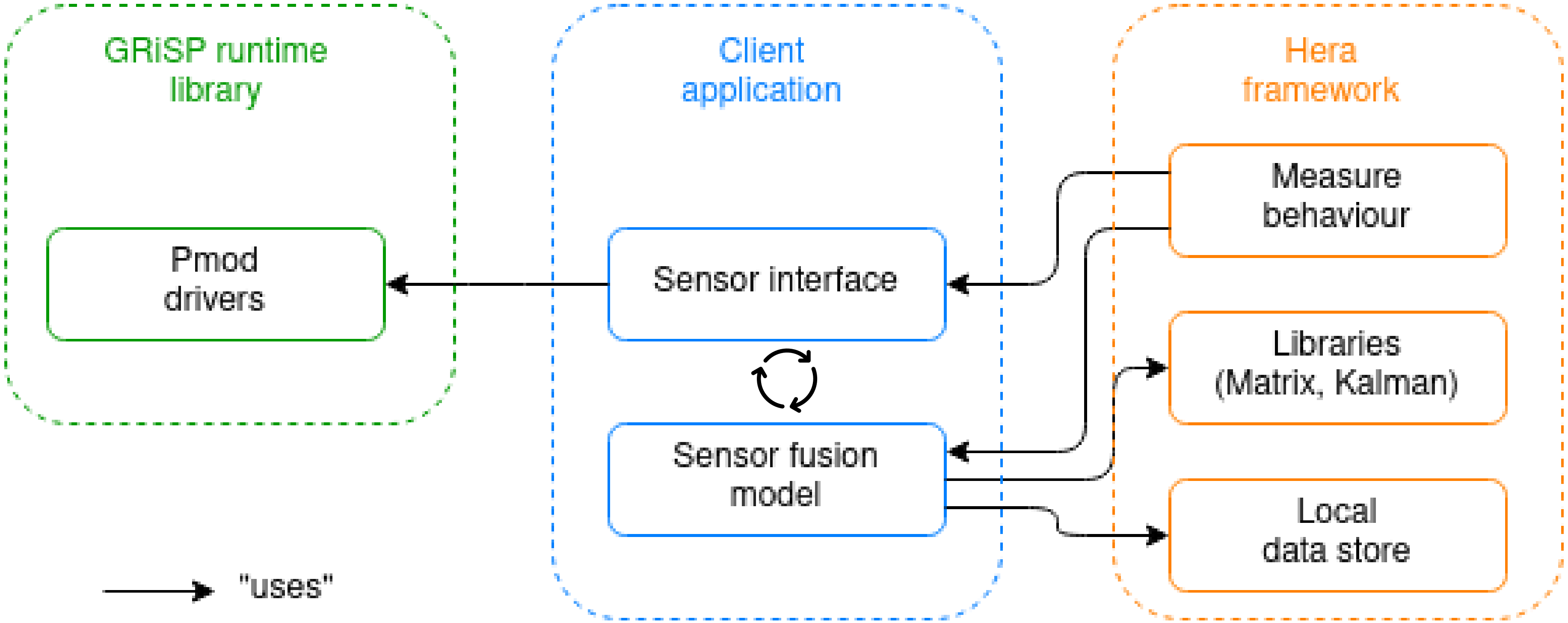




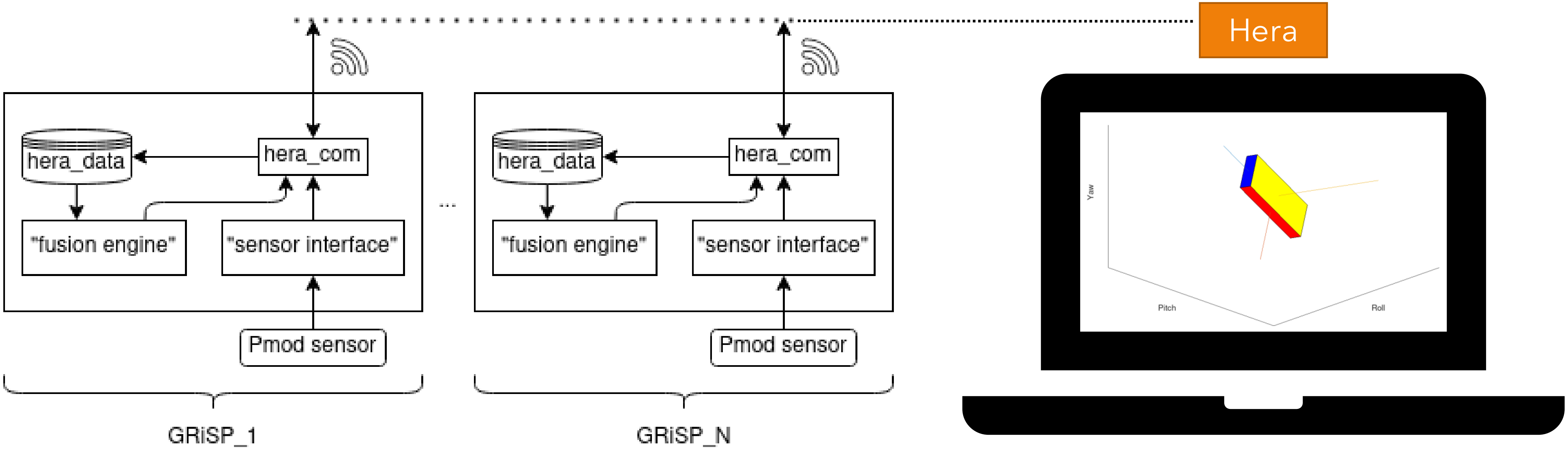
Software architecture

Overview and Properties

Software architecture



Software architecture



System properties

- Modular
- Soft real-time
- Asynchronous
- Dynamic
- Fault-tolerant

Fault tolerance

How and why

How does Hera achieve fault tolerance ?

Fault tolerance: system keeps running despite failures

- Asynchronous model
- Dynamic system ⋙ Supervision with restarting strategy
- Hardware redundancy

➜ Sensor fusion as long as one GRiSP board is alive

✓ **Proved by fault injection**

Why fault tolerance ?

- **Bug-free** software is a **myth** (edge cases)
- **Multiple points of failure**: hardware (sensor, board), network, software (driver, user code)
- Sensor fusion is **hard enough** – user should **not** do **defensive programming** nor **error handling**

Experimental sensor fusion with Hera

Phase 1

Angular velocity estimation

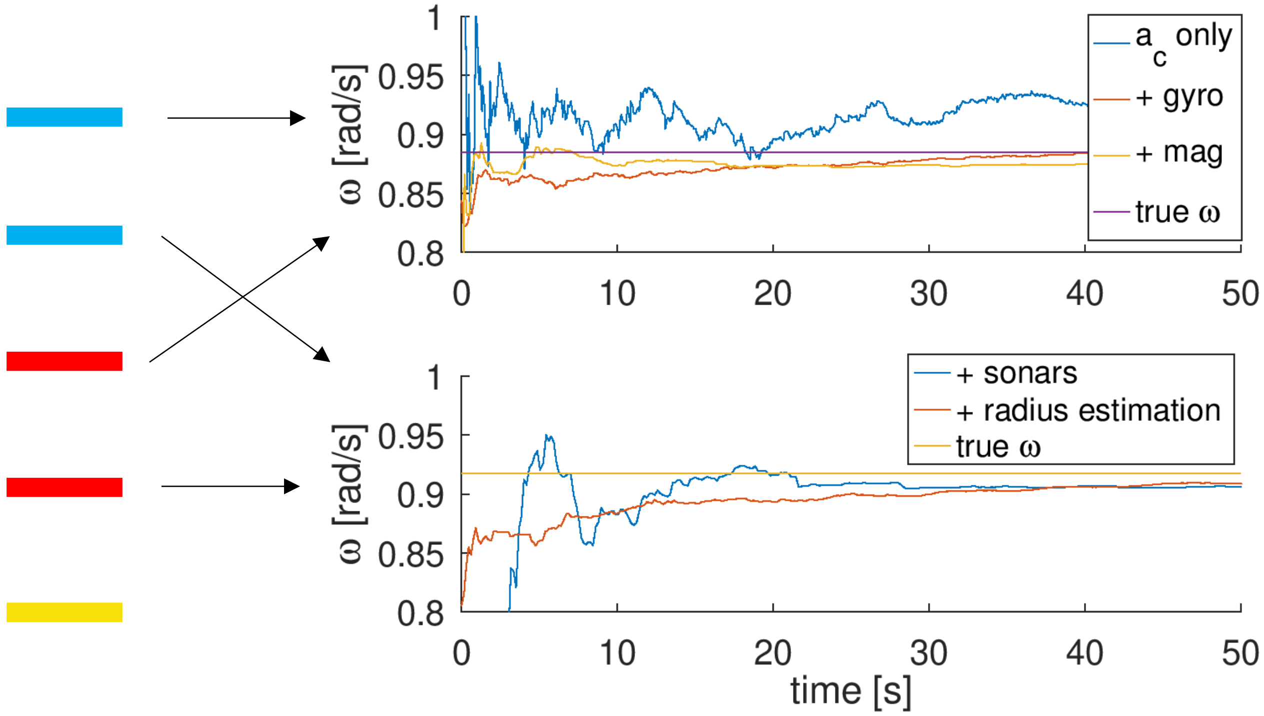
Accelerometer

+ Sonars

+ Gyroscope

+ Radius estimation

+ Magnetometer



Model description in Hera

- Final model encoded in Hera :
  - ✓ Accelerometer
  - ✓ Gyroscope
  - ✓ Magnetometer
  - ✓ Sonars x2
- Extended Kalman Filter

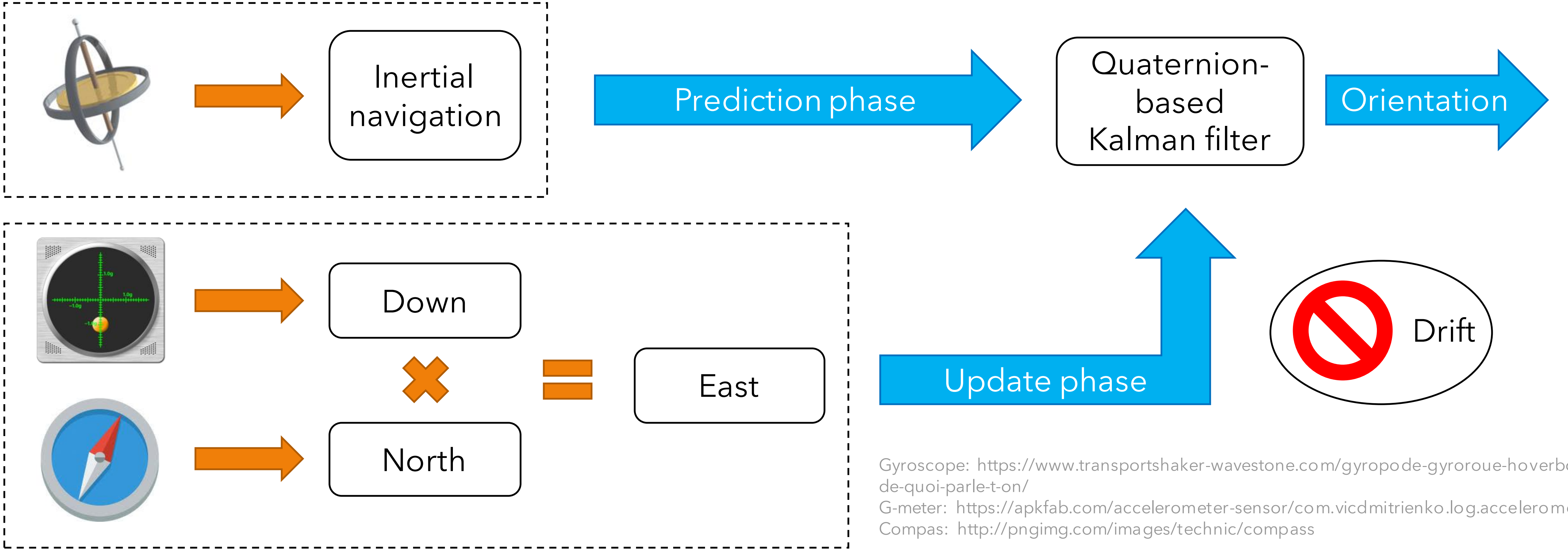
Easy-to-use !

```
measure({T0, X0, P0}) ->
N = [Data || {_,_,Ts,Data} <- hera_data:get(nav), T0 < Ts],
M = [Data || {_,_,Ts,Data} <- hera_data:get(mag), T0 < Ts],
S = [Data || {_,_,Ts,Data} <- hera_data:get(sonar), T0 < Ts],
T1 = hera:timestamp(),
if
length(N) + length(M) + length(S) == 0 ->                    1
  {undefined, {T0, X0, P0}};
true ->
  Dt = (T1 - T0)/1000,
  F = fun([_, _, [O], [W], [Radius]]) -> [
    [Radius*math:cos(O)],
    [Radius*math:sin(O)],
    [O+W*Dt],
    [W],
    [Radius]
  ] end,
  Jf = fun([_, _, [O], _, [Radius]]) -> [
    [0,0,-Radius*math:sin(O),0,math:cos(O)],
    [0,0,Radius*math:cos(O),0,math:sin(O)],
    [0,0,1,Dt,0],
    [0,0,0,1,0],
    [0,0,0,0,1]
  ] end,
  Q = mat:zeros(5,5),
  H = fun([[X], [Y], [O], [W], [Radius]]) ->
    [[Radius*W*W] || _ <- N] ++
    [[W] || _ <- N] ++
    [[shortest_path(-OZ, O)] || [OZ] <- M] ++
    [[dist({X,Y},{Px,Py})] || [_,Px,Py] <- S]
  end,
  Jh = fun([[X], [Y], _, [W], [Radius]]) ->
    [[0,0,0,2*Radius*W,W*W] || _ <- N] ++
    [[0,0,0,1,0] || _ <- N] ++
    [[0,0,1,0,0] || _ <- M] ++
    [[dhdx({X,Y},{Px,Py}),dhdx({Y,X},{Py,Px}),0,0,0]
      || [_,Px,Py] <- S]
  end,
  Z = [[-Ay] || [Ay,_] <- N] ++
    [[-Gz] || [_,Gz] <- N] ++
    [[-O] || [O] <- M] ++
    [[Range] || [Range,_,_] <- S],
  R = mat:diag(
    [?VAR_A || _ <- N] ++
    [?VAR_G || _ <- N] ++
    [?VAR_M || _ <- M] ++
    [?VAR_S || _ <- S]
  ),
  {X1,P1}=kalman:ekf({X0,P0},{F,Jf},{H,Jh},Q,R,Z),
  {ok, lists:append(X1), {T1, X1, P1}}
end.                                                          2
```

Attitude and heading reference system

Phase 2

Orientation estimation

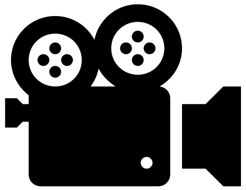


Gyroscope: https://www.transportshaker-wavestone.com/gyropode-gyroroue-hoverboard-de-quoi-parle-t-on/
G-meter: https://apkfab.com/accelerometer-sensor/com.vicdmitrienko.log.accelerometer
Compas: http://pngimg.com/images/technic/compass

Orientation estimation

State of the art
real-time MEMS AHRS
$\simeq$ 100 Hz

Hera = 3.75 Hz



Video of AHRS

Conclusion

Verdict
and
Future work

Verdict

✚

- **Satisfactory results** for simple sensor fusion
- **No Cloud**: simpler, cheaper, more reliable
- **High-level** with focus on sensor fusion model: no soldering, no datasheet, no C
- **Fault-tolerant**, **Easy-to-use**, **Low-cost**

➜ Ideal for education and prototyping

—

- **Performance** limited by GRiSP-base board and by Erlang numerical computation
- Designed for a **small cluster**
- **Security**

Future work

Performance improvements

- **GRiSP 2: 10x to 20x** faster
- **New matrix library: 10x to 100x** faster
  Tanguy Losseau. 2021. Concurrent Matrix and Vector Functions for Erlang. Master's thesis. UCLouvain.
- **Improved driver** for Pmod NAV

✓ **Summer 2021**

Possible experimentations

- Machine learning with Hera (e.g. motion recognition)
- Controlling physical devices
- Targeting rugged terrains

Defence

Questions and answers