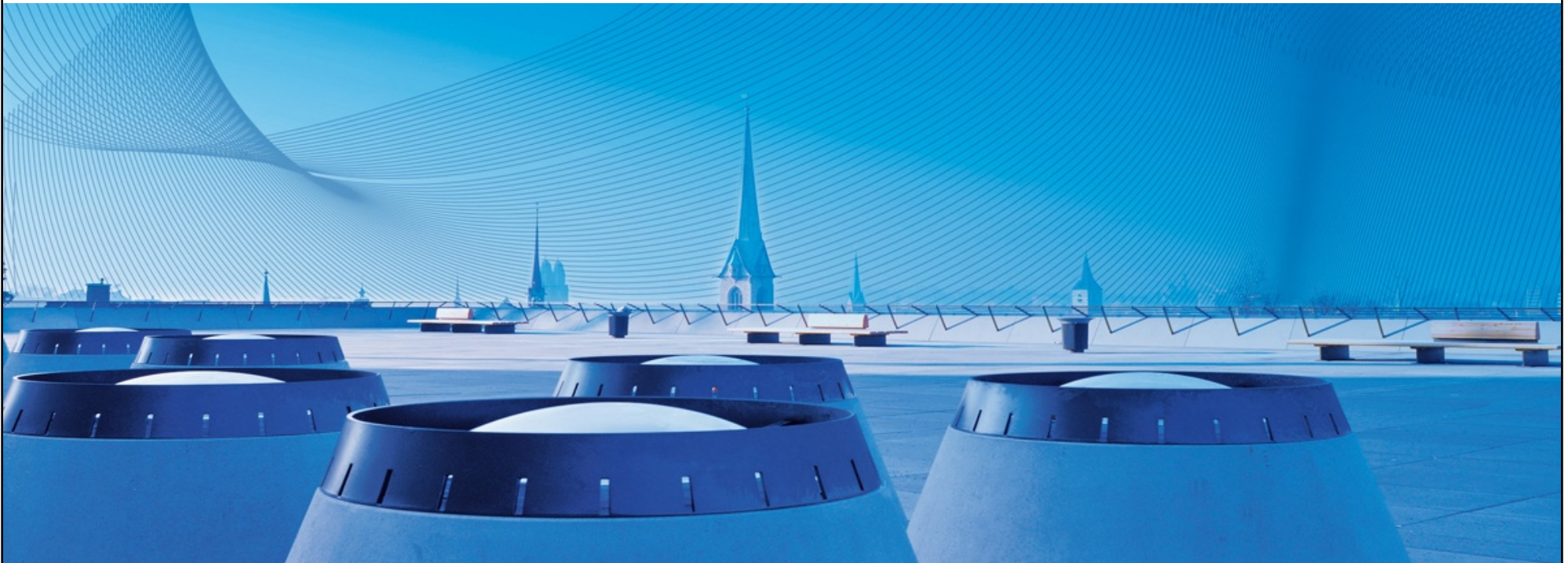
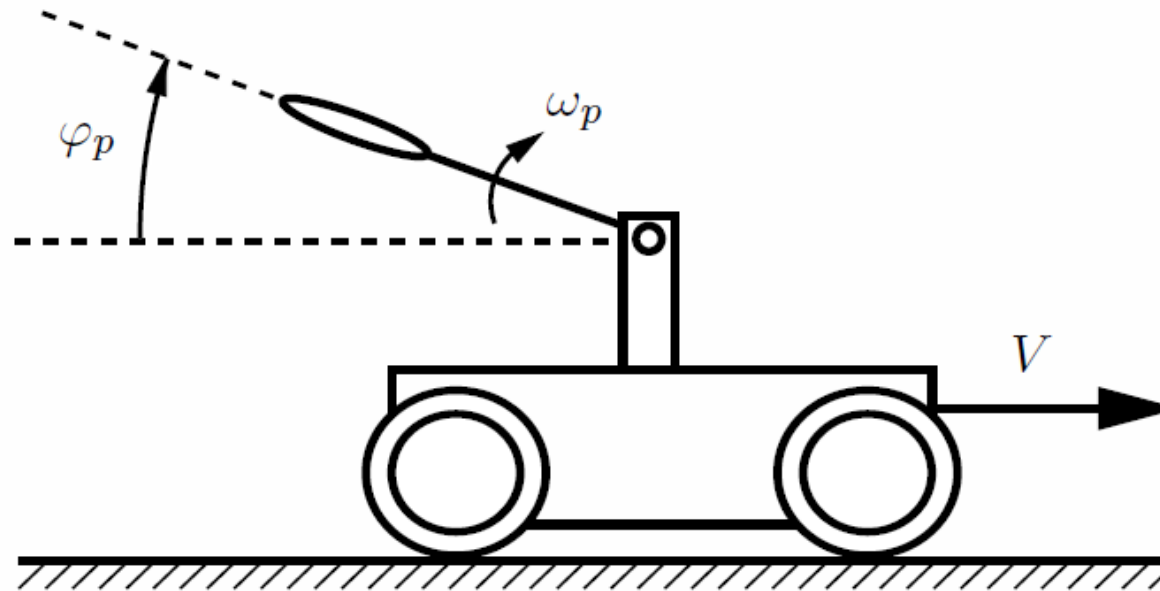


Paddle Trajectory Generation

Semester Project Fall 2010, Roman Bosshard
Supervision: Nico Hübel



The Ground Vehicle



- 4 mechatronic wheels
- Badminton racket
- 2 large batteries
- Up to one g and 5 m/s

The Ground Vehicle

- Long-term goal of the project
 - Play badminton match
 - .. with humans
 - .. with other robots (quadrotors)
 - .. with itself

- Requirements
 - Track balls using Vicon feedback
 - Predict ball trajectories
 - Accurately hit the ball to specified target
 - Trajectory planning

The Ground Vehicle

- Starting point in fall 2010
 - Comm. to motor controllers via CAN-Bus
 - Wireless comm. via USART unit
 - Manual control with joystick via MATLAB/Simulink program

- No control of vehicle dynamics (wheels & paddle)
- No trajectory planning
- No communication protocol for FMA integration
- Timing functionality

Goals of this Semester Project

- Modeling
- Develop suitable paddle control strategy
- Implement controllers
- Test controllers in FMA
- Measurements
- Learning algorithm for accurate target hitting

Modeling - Trajectory

- Ball is very light and relatively large
- Drag influence high
- Non-linear diff. eq.

$$m_b \ddot{b} = F_g - \frac{1}{2} c_d \rho_{Air} A_b \|\dot{b}\| \dot{b}$$

b Position of the ball

m_b Mass of the ball

A_b Cross section of the ball

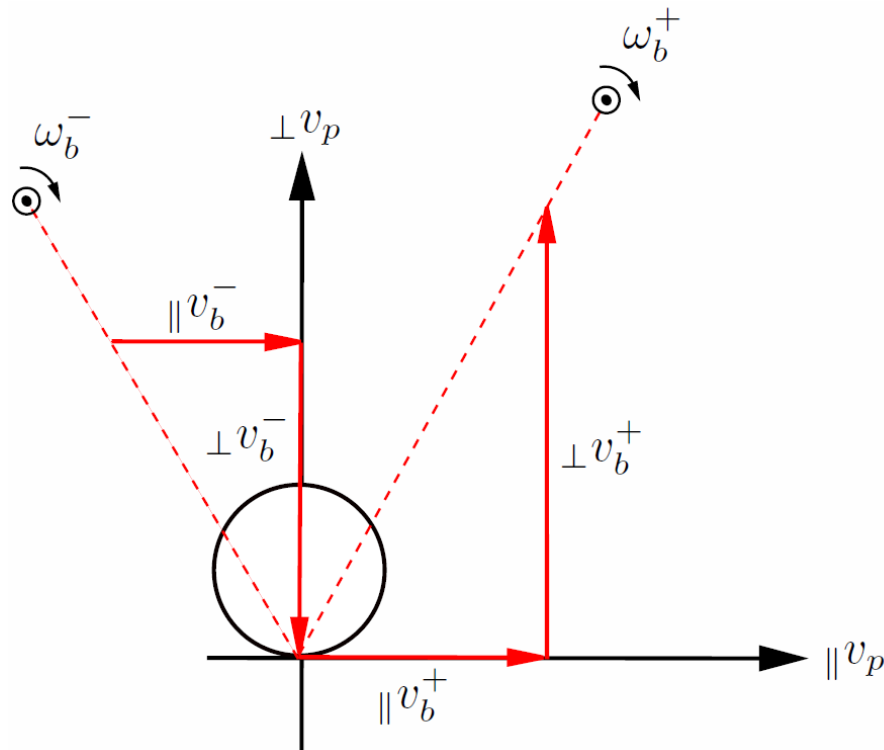
c_d Drag coefficient of a sphere

g Gravitational acceleration

ρ_{Air} Density of air

- Used Mark Mullers ball estimator / predictor

Modeling - Bounce



v_b Ball velocity

ω_b Ball spin

v_p Paddle velocity

e_z Vertical coefficient of restitution

-, + Value before/after the bounce

\perp, \parallel Perpendicular/Parallel component

- **Vertical:** Coefficient of restitution

$$\perp v_b^+ = -e_z \perp v_b^- + (1 + e_z) \perp v_p$$

Modeling - Bounce

- **Horizontal:** Two different bounce models analyzed

- Howard Brody's Model
 - Ball slides, then rolls on paddle
 - Impulse change due to sliding friction
 - Resulting velocity depends on normal force

- Richard Garwin's Model
 - Ball slides, then „grips“ the paddle
 - Conservation of angular momentum
 - Uses horizontal coeff. of restitution

Modeling - Bounce

- Richard Garwin's Model

$$e_x = - \frac{\|v_b^+ - r_b\omega_b^+ - \|v_p^+}{\|v_b^- - r_b\omega_b^- - \|v_p^-}$$

$$\|v_b^+ = \|v_b^- + \frac{J_b(1 + e_x)}{J_b + m_b r_b^2} \left(\|v_p - (\|v_b^- - r_b\omega_b^-) \right)$$

$$r_b\omega_b^+ = r_b\omega_b^- - \frac{m_b r_b^2(1 + e_x)}{J_b + m_b r_b^2} \left(\|v_p - (\|v_b^- - r_b\omega_b^-) \right)$$

e_x Horizontal coefficient of restitution

v_b Ball velocity

ω_b Ball spin

v_p Paddle velocity

r_b Radius of the ball

m_b Mass of the ball

J_b Moment of inertia of the ball, about center of mass

-, + Value before/after the bounce

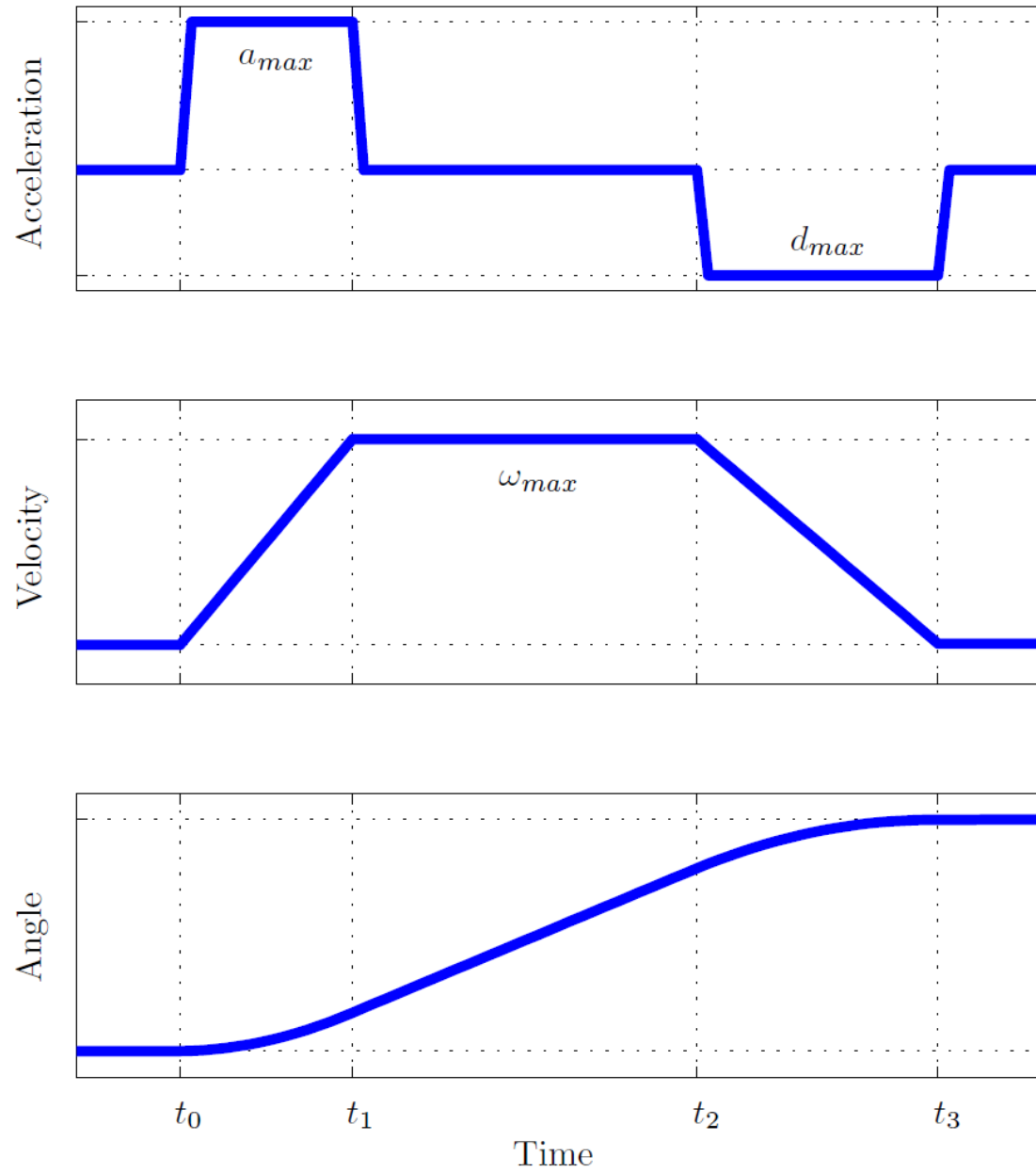
\perp, \parallel Perpendicular/Parallel component

Paddle Trajectories

- Timing is crucial
- All sensory equipment in FMA
- Data sent to GV via Aerocomm

- Simple parameterization
 - Paddle motor velocity
 - Start time (relative)

- Triggered via timed interrupt



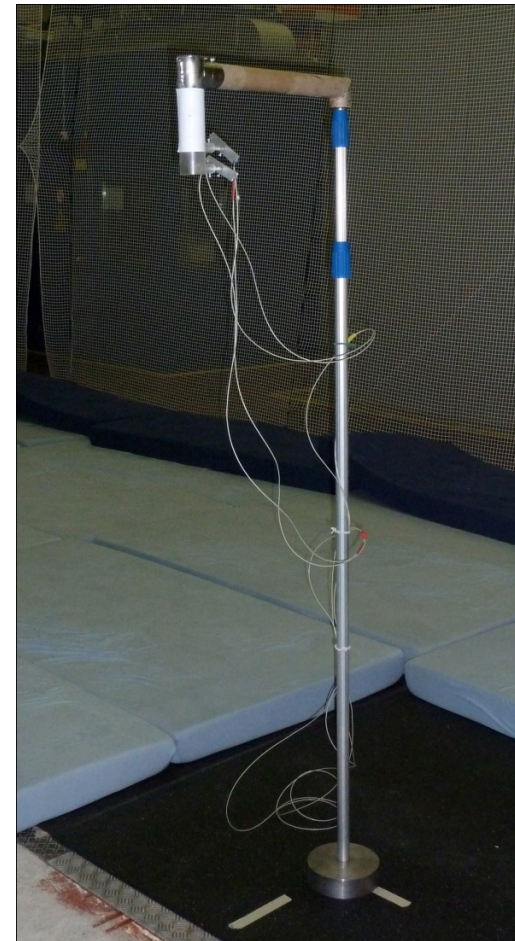
Off-Board Controller

Algorithm 3 Off-Board Controller main loop

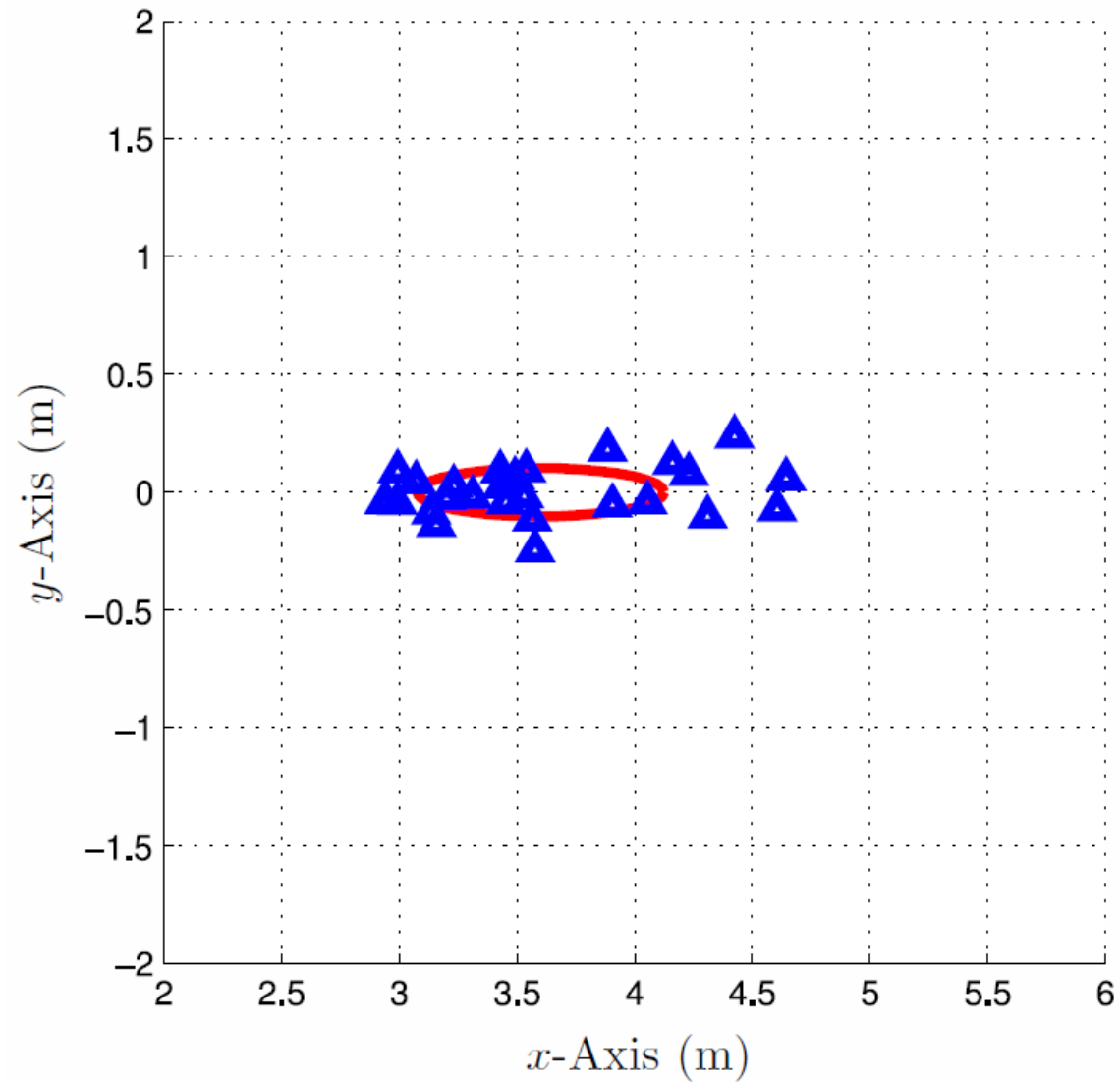
```
1: while experiment is running do  
2:   get time estimate  $\leftarrow t_0$   
  
3:   if first time ball is detected then  
4:     sendCommand( $n_p, t_0$ )  
5:   else  
6:     if  $|t_0 - t_{last}| > \Delta t_{max}$  then  
7:       sendCommand( $n_p, t_0$ )  
8:     end if  
9:   end if  
10: end while
```

Video

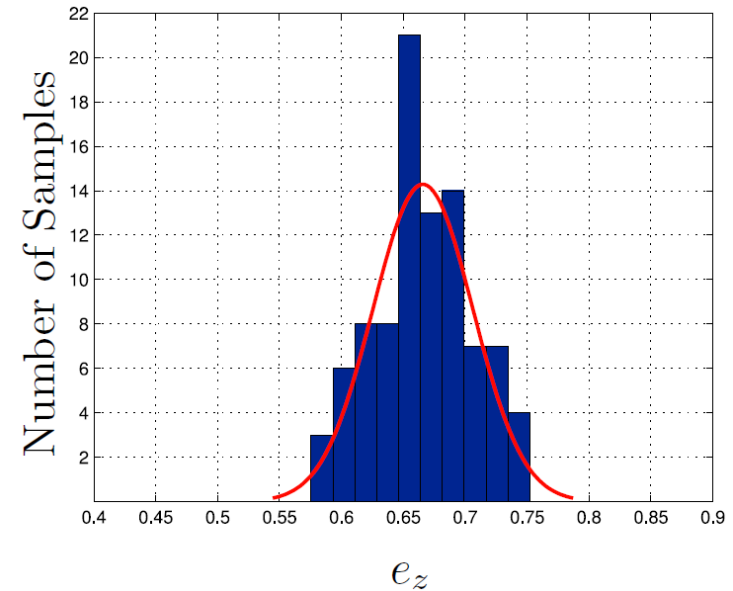
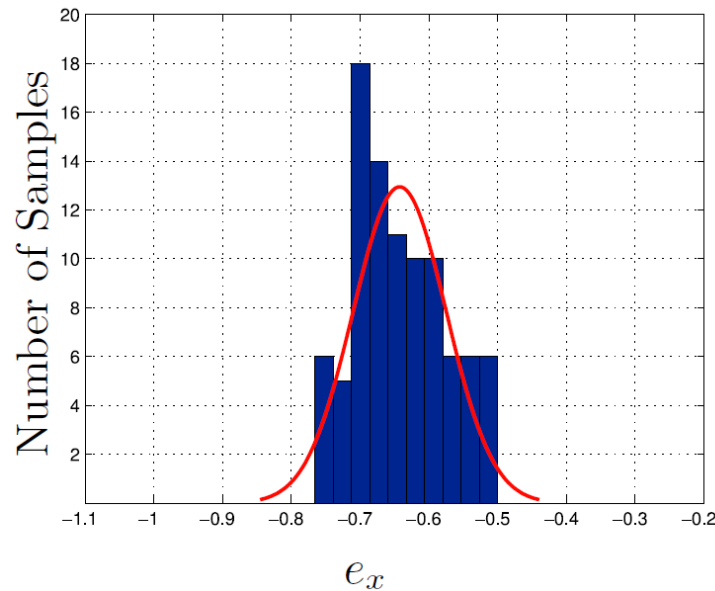
Measurement Results



Measurement Results



Parameter Estimates



Parameter	Mean	Variance	Standard Deviation	#Samples
e_x	-0.6425	0.0046	0.0677	92
e_z	0.6658	0.0016	0.0404	91

Iterative Parameter Learning

- Hit the ball to reach specified height and distance
- Iterative parameter learning strategy

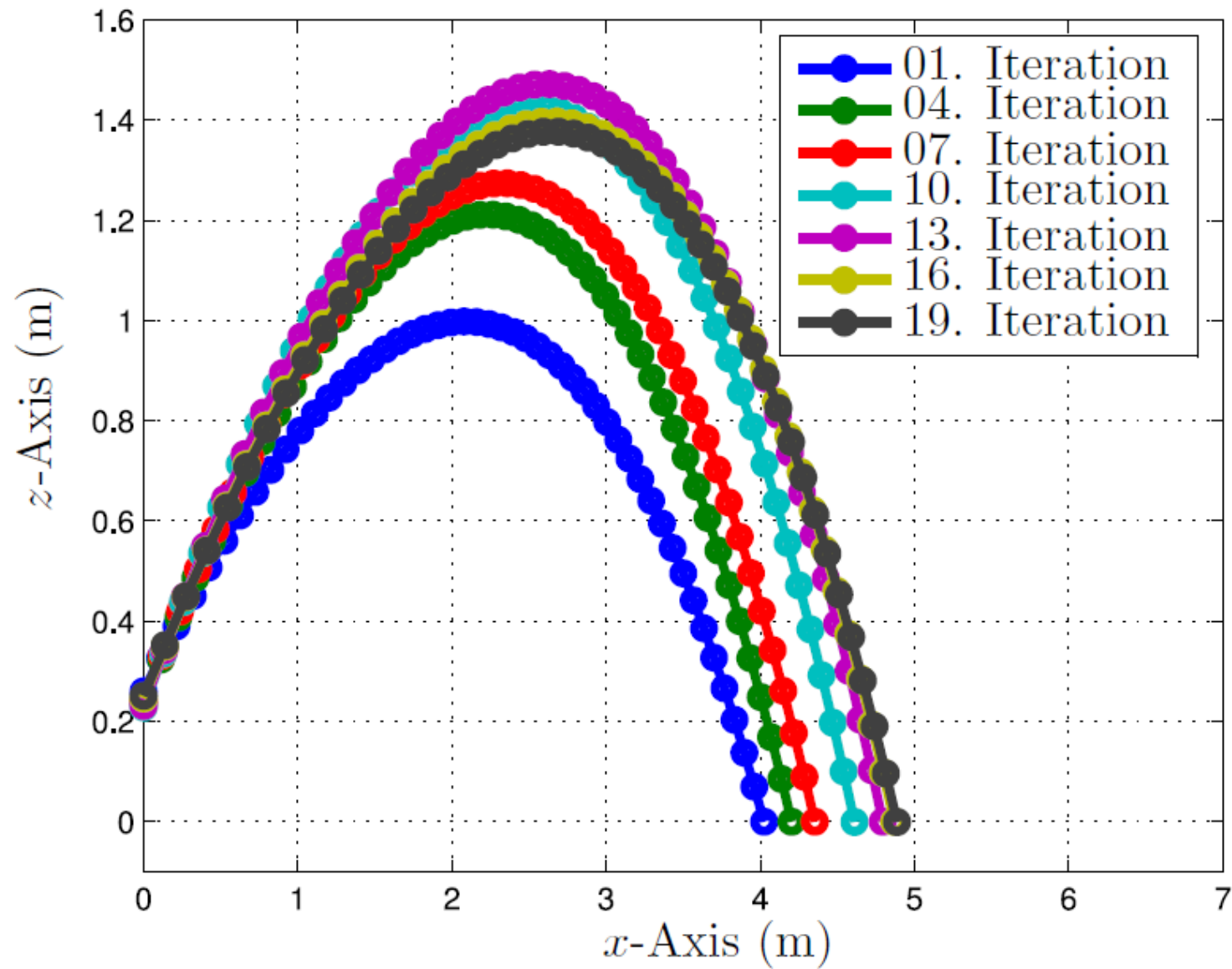
- Measure error

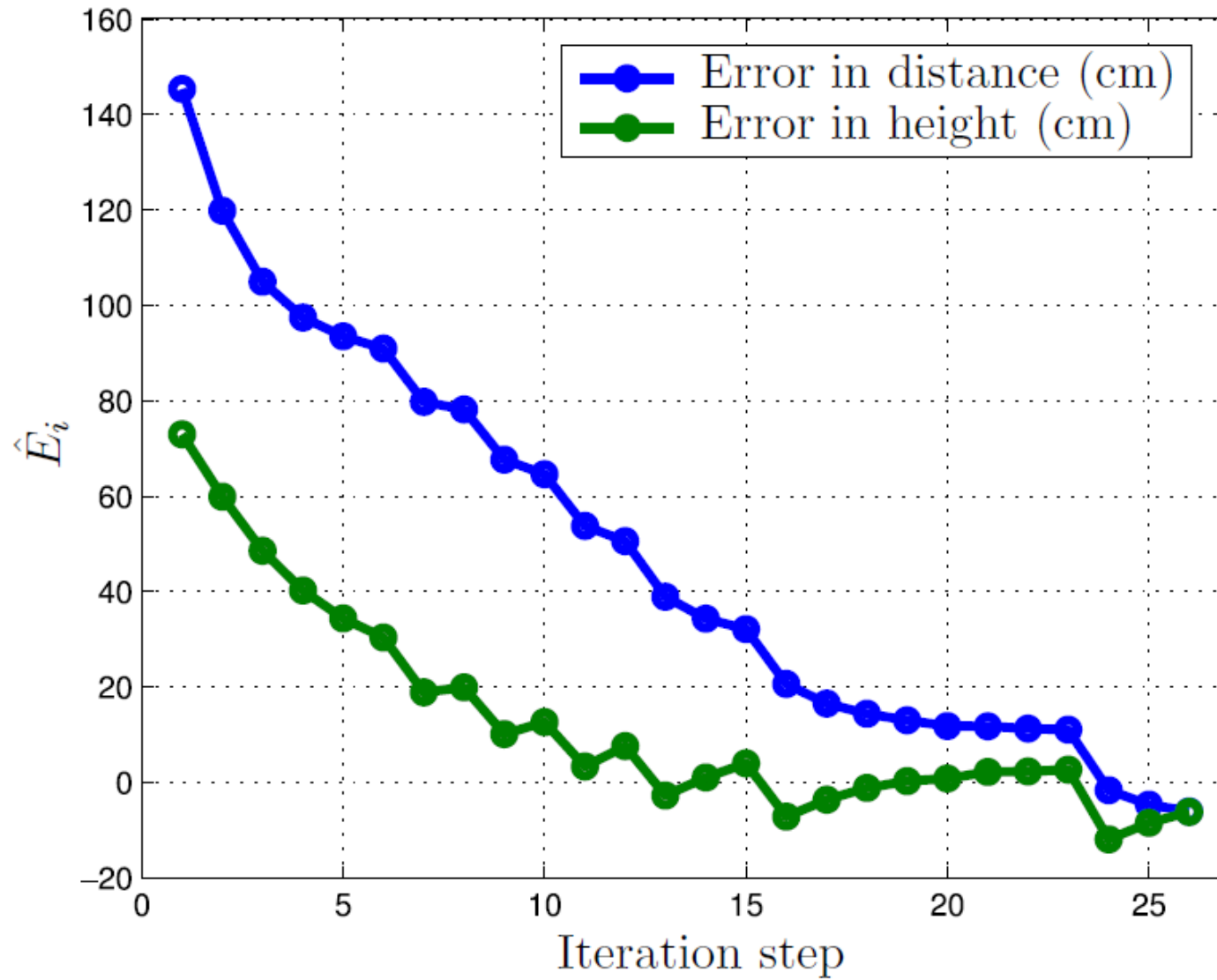
$$\hat{E}_i = \begin{bmatrix} x_{Target} - b_x[N] \\ h_{Target} - \max_k(b_z[k]) \end{bmatrix}$$

- Learn from past error(s)

$$\hat{U}_{i+1} = \hat{U}_i - \gamma J^{-1} \hat{E}_i$$

- Determine optimal paddle velocity & angle
 - For set of (dist./height) combinations
 - Store to database for later use
- Extension for model parameter estimation possible





Conclusion

- Development of paddle control strategy
- Integration into FMA with a basic controller
- First measurements show high target accuracy
- Development of parameter learning strategy
- Simulation to show feasibility

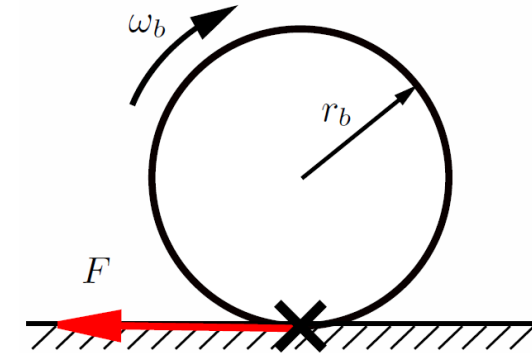
- Suggestions for future work on the GV
 - Structure firmware (framework for ctrl. dev., timed interrupts, interrupt based comm., ...)
 - FMA integration (framework for ctrl. dev., comm. protocol, ...)
 - Learning: test strategy, vehicle vel. > 0 , extension to 3D

Questions?



Modeling

- Take $J_b \dot{\omega}_b = r_b F$
 $J_b \omega_b^+ = J_b \omega_b^- + r_b m_b (\|v_b^+ - \|v_b^-)$



- Use $e_x = -\frac{\|v_b^+ - r_b \omega_b^+ - \|v_p^+}{\|v_b^- - r_b \omega_b^- - \|v_p^-}$

- Leads to $\|v_b^+ = \|v_b^- + \frac{J_b(1 + e_x)}{J_b + m_b r_b^2} \left(\|v_p - (\|v_b^- - r_b \omega_b^-) \right)$

Algorithm 1 Firmware main loop

```
1: while 1 do
2:   if PaddleTimerFlag set then
3:     reset PaddleTimerFlag
4:     block paddle timer restart
5:     start paddle motion
6:     unblock paddle timer
7:   end if

8:   if new USART command available then
9:     receive data ← RxBuffer
10:    setPaddleCommands(RxBuffer)
11:   end if
12: end while
```

Algorithm 2 Function `setPaddleCommands(RxBuffer)`

- 1: load first byte of `RxBuffer` $\leftarrow n_p$
 - 2: **if** $n_p \neq$ last received velocity command **then**
 - 3: set paddle motor velocity to $100n_p$
 - 4: **end if**

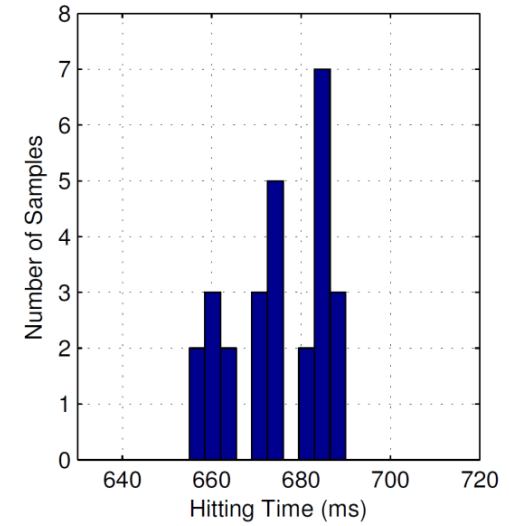
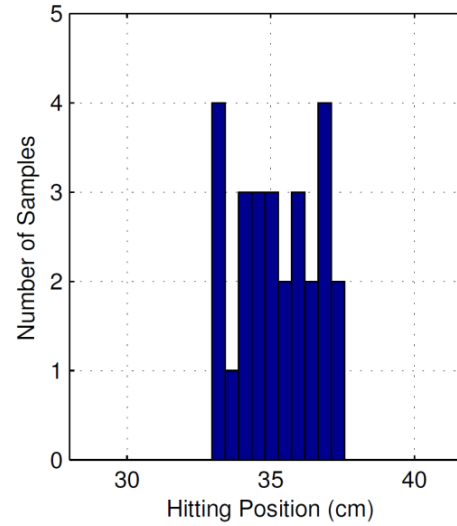
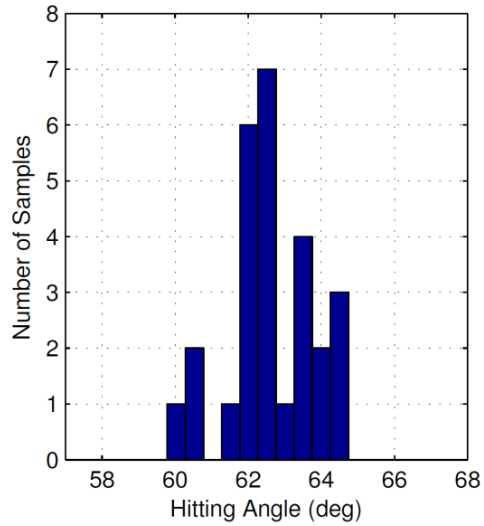
 - 5: load second and third byte of `RxBuffer` $\leftarrow a, b$
 - 6: form 2-byte variable $(a \ll 8) \mid b \leftarrow t_0$

 - 7: **if** $t_0 \neq$ last received time command **then**
 - 8: load fourth and fifth byte of `RxBuffer` $\leftarrow c, d$
 - 9: form 2-byte variable $(c \ll 8) \mid d \leftarrow t_s$
 - 10: set paddle timer to time $t_0 - t_s$
 - 11: **end if**
-

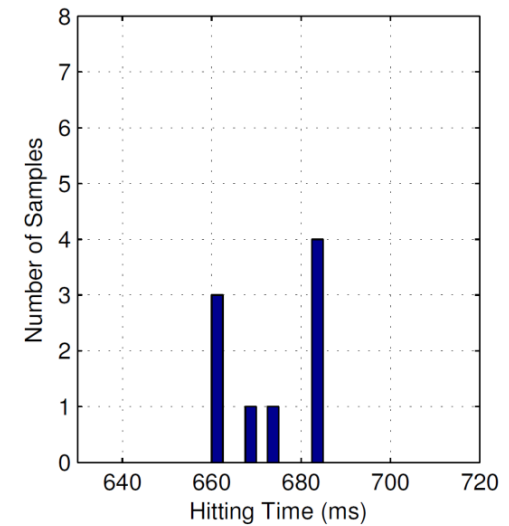
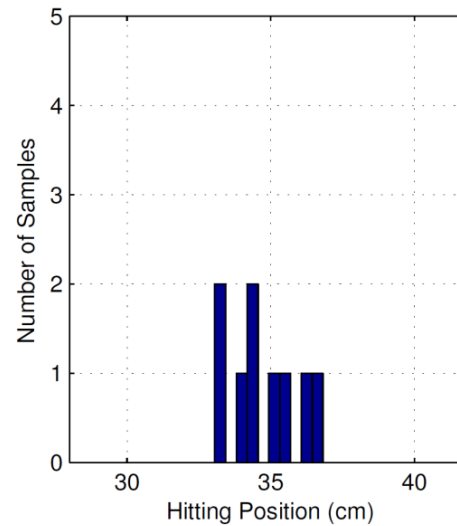
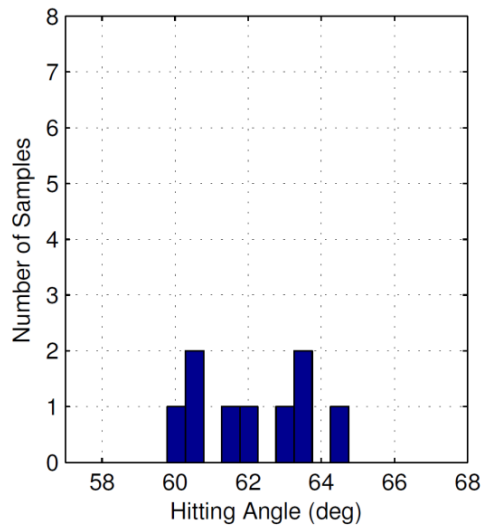
Algorithm 4 Function `sendCommand(n_p, t_0)`

- 1: get time stamp $t_s = \text{current system time} - \text{timeOffset} \leftarrow t_s$
 - 2: send n_p, t_0 and t_s to Ground Vehicle
 - 3: save time $t_{last} = t_0 \leftarrow t_{last}$
 - 4: set `timeOffset` to current system time
-

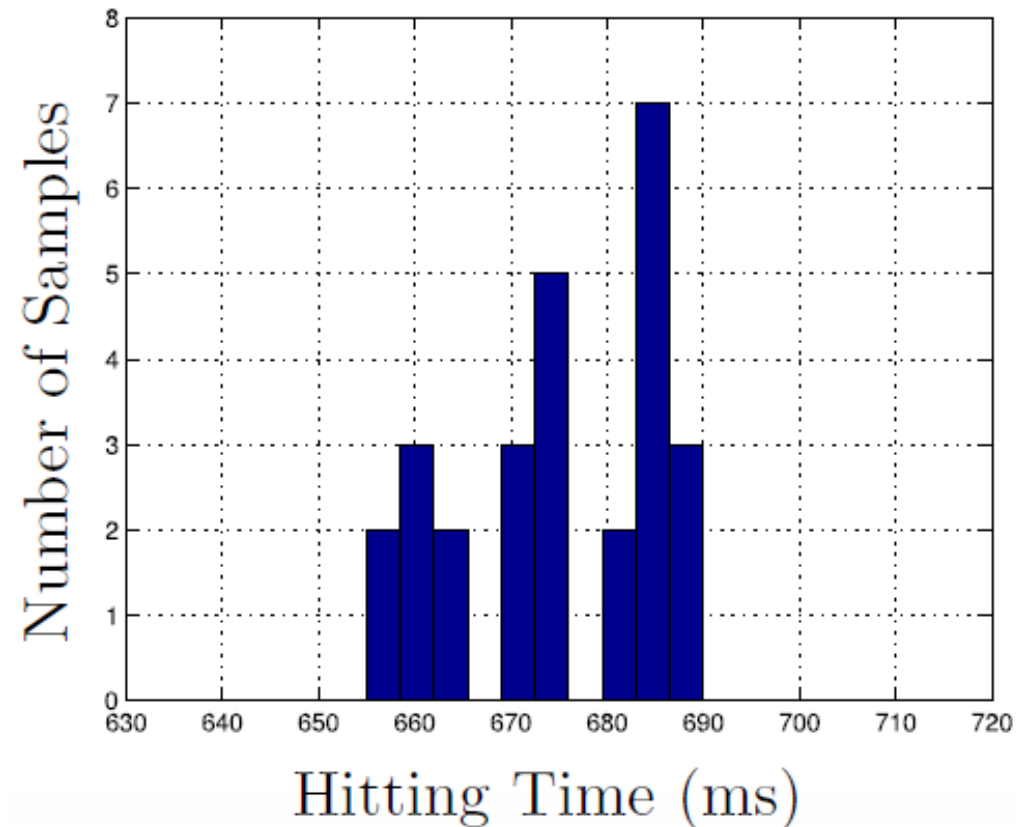
Samples of both clusters



Only the 9 samples of right cluster

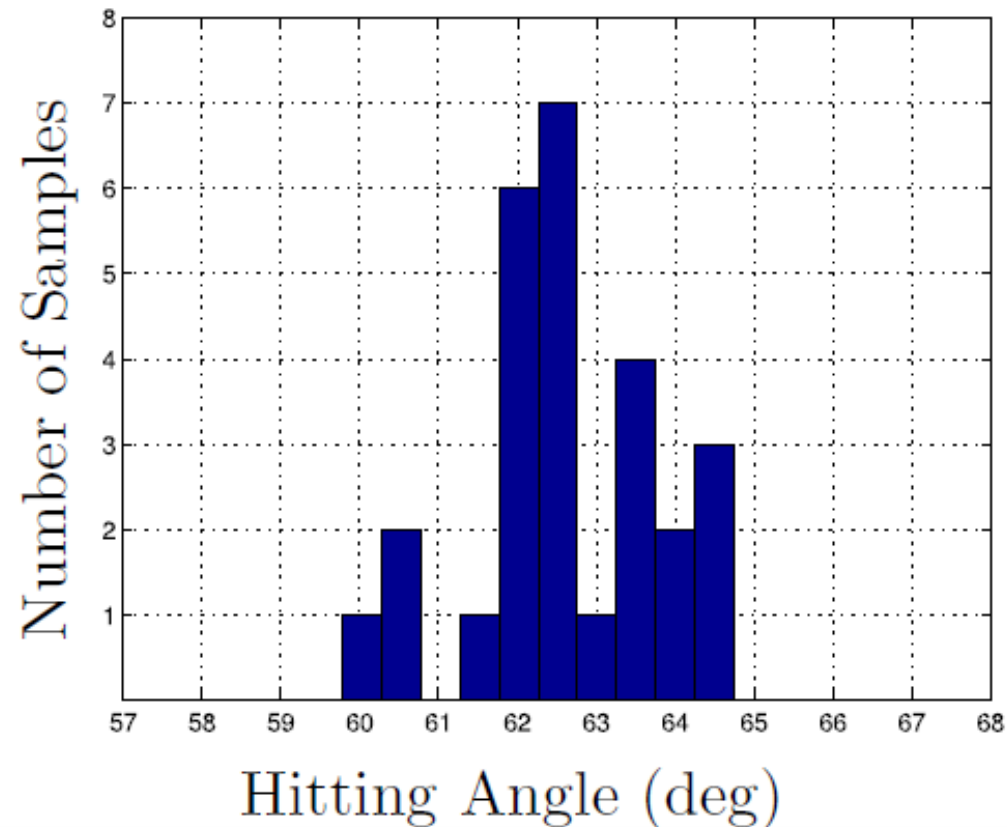


Measurement Results



- Standard deviation in timing: 11 ms

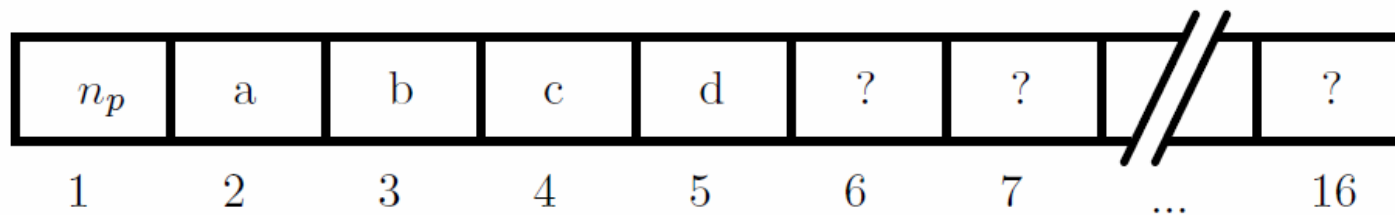
Measurement Results



- Standard deviation in angle: 1.2°

Off-Board Controller

- Detect ball in FMA
- Get time estimate for ball to reach GV
- Send data to GV:



- Motor velocity (n_p)
- Release time for paddle (a | b)
- Timestamp (c | d)