TECHNISCHE
UNIVERSITÄT
DRESDEN

Fakulty of Electrical and computer Engineering Institut for Automation Engineering

Matlab Simulation Framework

et al for Robotics

Dresden, July 27, 2017

Motivation

Simulator Framework for testing of guidance algorithms

Requirements:

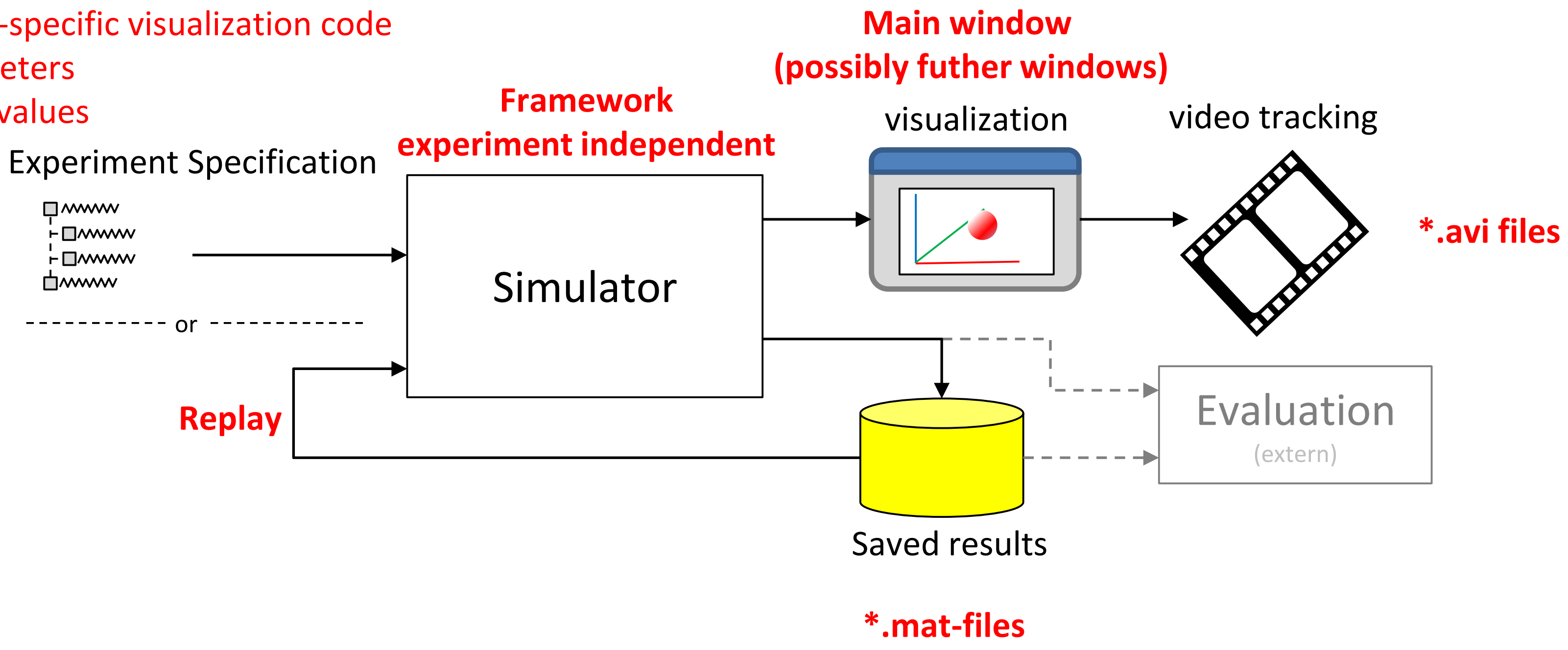
- Simple time discret simulation in *Matlab*
- *Stand-alone* (GUI) and *scripted operation* (non-GUI)
- Easily reconfigurable system structure
- Flexible visualization, easily adaptable
- Clear separation of experiment data and simulator core functionality
- Saves the timings of all simulation data
- Replay (only vilualization without recalculation)
- Video recording

➡ **Universal Version of 2D simulation framework by S. Horn**

Hauptkomponenten

Matlab structure

- system models (Blocks)
- model-specific visualization code
- parameters
- Initial values



Experiment Specification

or

Replay

Framework
experiment independent

Simulator

Main window
(possibly futher windows)

visualization

video tracking

***.avi files**

Evaluation
(extern)

Saved results

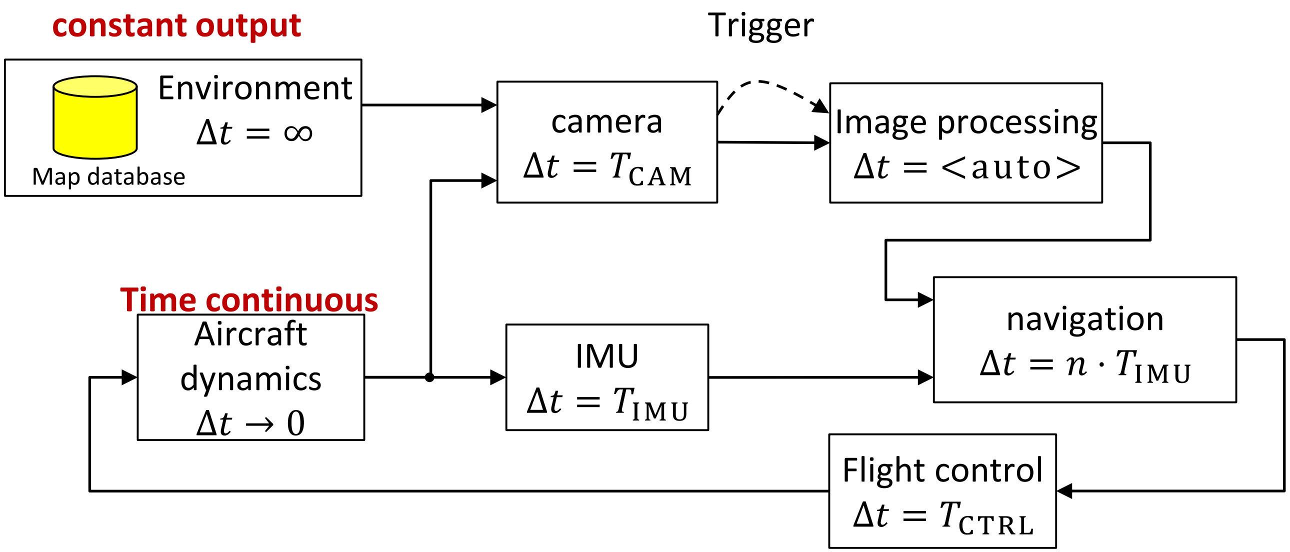
***.mat-files**

System Description

Model blocks

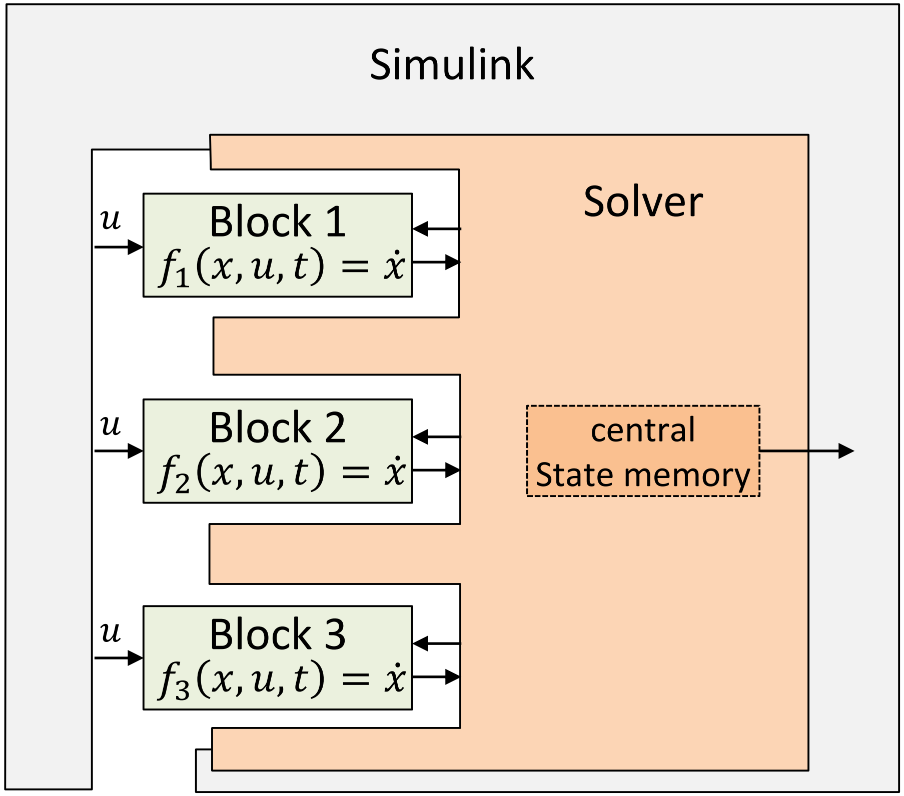
- 1 output, any number of input,
  any data format (Scalar, Matrices, Struct, …)
  ↳variable per calculation step (non-uniform output)
- Sampling time Δt (computing continuous blocks “if necessary“)

Example system:

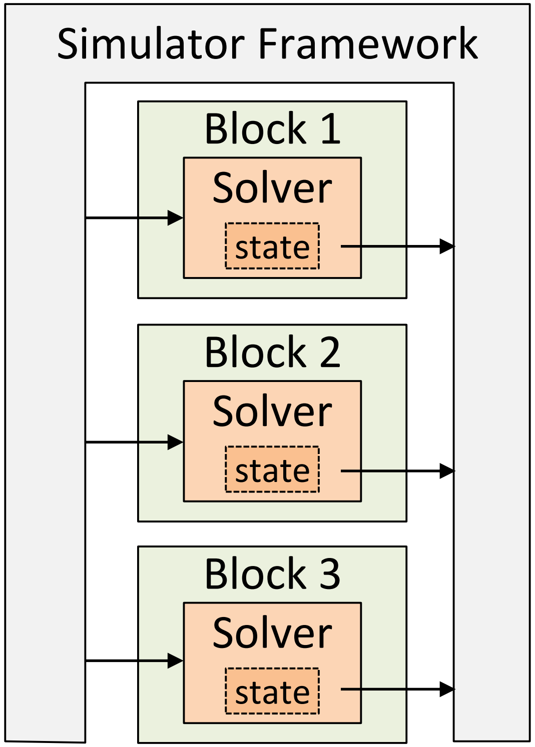


Abgrenzung zu Simulink

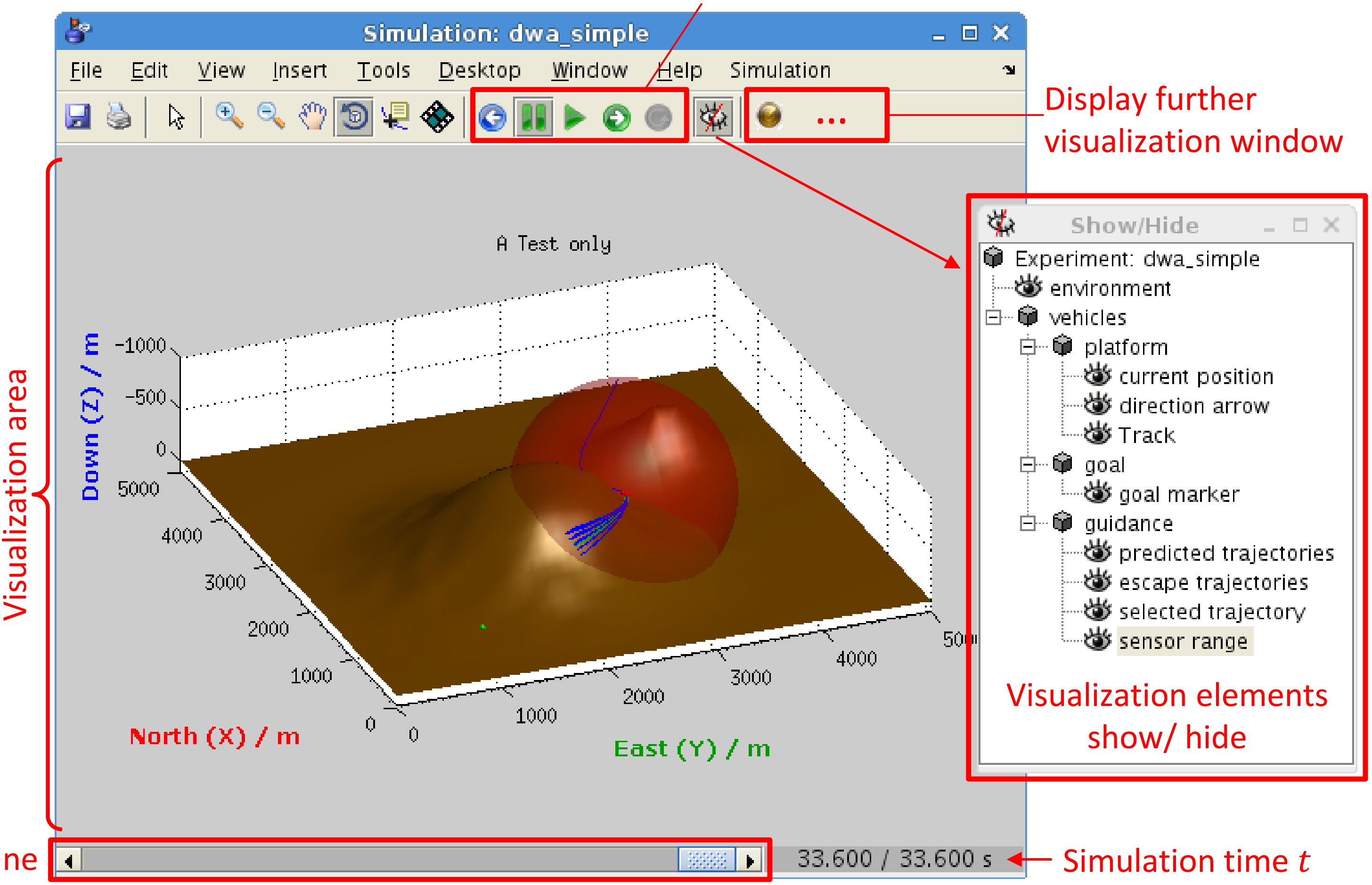
Architecture ofSimulink



Architektur of
Matlab Simulation Framework



Stand-Alone GUI



Simulation control

Display further visualization window

Visualization area

Visualization elements show/ hide

Timeline

Simulation time t

Experiment Description

Matlab strukture → hierarchic, object-oriented approach

```
function exp = experiment()
  exp = experiment_base('dwa_simple'); % skeleton struct for experiment

  exp.vehicle = struct( ... % blocks in nested structure
      'platform', model_masspoint([4800, 4000, -150, -150 * pi / 180, 0]), ...
      'goal', const_goal3d([500 1000 -250]), ...
      'guidance', guidance_dwa());
  exp.environment = env_heightmap('DEMs/two-hills_50x50.png');

  exp.display.title = 'A Test only'; % configure visualization area
  exp.display.settings = {'XLim', [0 5000], 'YLim', [0 5000], 'ZLim', [-1000 100]};
  exp.display.view = [-112, 26];
  exp.display.axis = 'equal';

  exp.stop = stop_distance(500); % simulation finished?
  exp.depends = {'*guidance'};   % tweak dependency analysis
end
```

blocks

Special fields

Aufruf des Simulators:

`simulate(experiment());` With GUI

- oder -

`simulate_unattended(experiment());` Without GUI

Implementierung eines Blocks

```
function b = test_block()
    b = block_base(1/10, {'input1'}, @process);
```

Block framework

Sampling time

```
    % Algorithm parameters
    b.parameter = 1000;
    b.parameter2 = 'abc';

    function [state, out, debug] = process(block, t, state, in)
        % Compute block state & outputs for time t
        ...
    end
```

Model propagation

```
    % visualization
    b.graphicElements(1).draw = @visualize;
    b.graphicElements(1).name = 'Graphic Object';
    function handles = visualize(block, ax, handles, out, debug, state, in)
        if isempty(handles)
            % create visualization objects
            handles = ...
        end
        % update visualization
        set(handles, 'Property', value, ...
    end
end
```

A visualization element

Thanks for your attention!