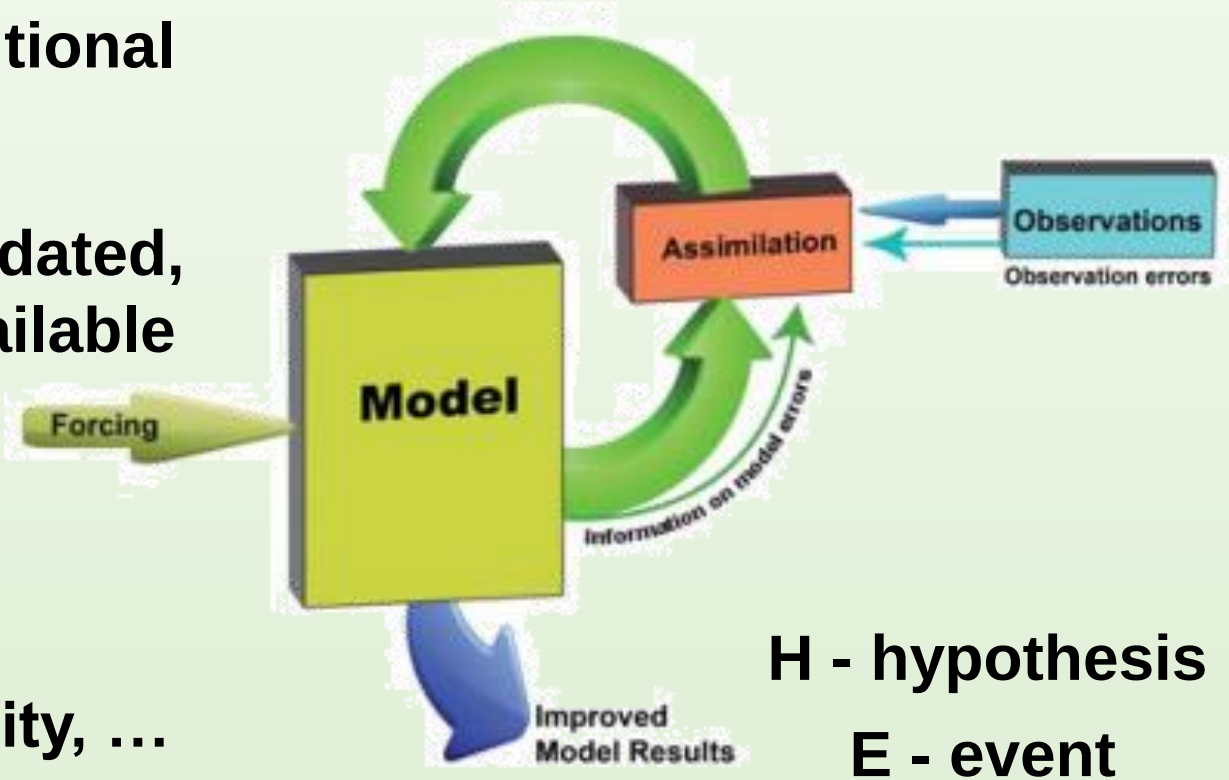
Plugin Optimus: Parameters estimation in biomechanical models

Parameters determination via Bayesian Inference

Bayesian inference:

- ► Statistical inference based on conditional probability
- ► The probability for hypothesis is updated, when more information becomes available



On-the-fly stochastic estimation of system state and parameters

- ► System state: model position, velocity, …
- ► Parameters: Young's modulus, Poisson ration, stiffness, …

H - hypothesis

E - event

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)}$$

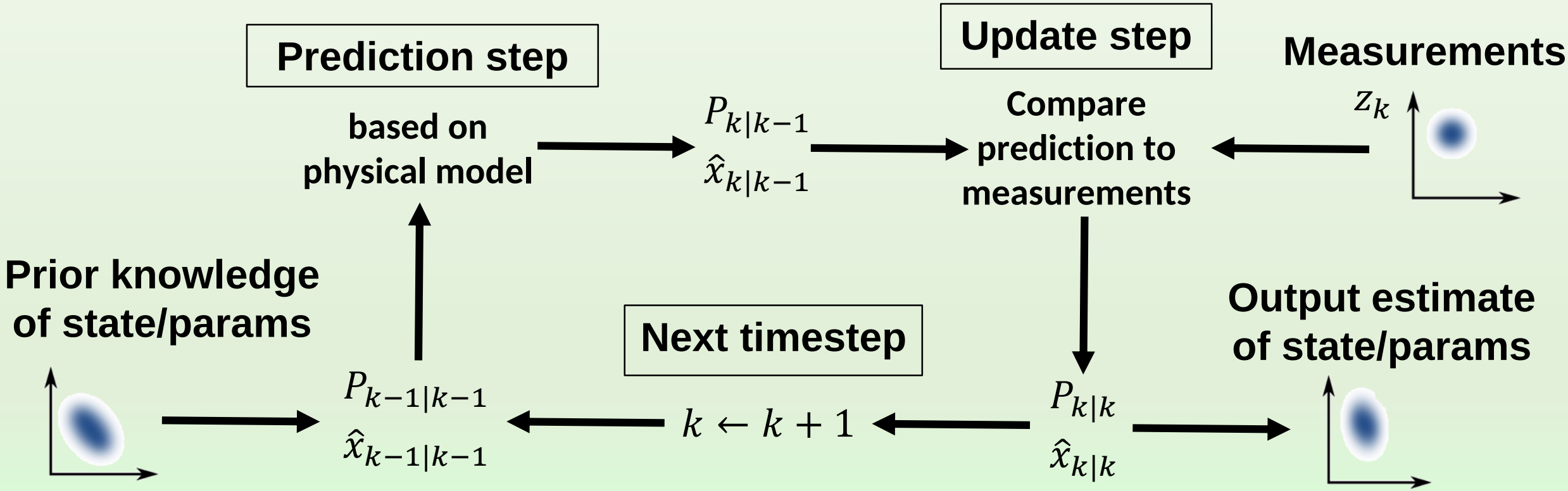
Kalman filtering process

linear-quadratic estimation (linear system, quadratic cost)

- **estimates unknown variables based on noisy observations which are sequentially acquired**

Cost function:

$$J_m = \tfrac{1}{2}(y_k - Hx_k)^T R^{-1}(y_k - Hx_k) + \tfrac{1}{2}(x_k - Fx_{k-1})^T P_{k|k-1}^{-1}(x_k - Fx_{k-1})$$



Optimus plugin: principal moments

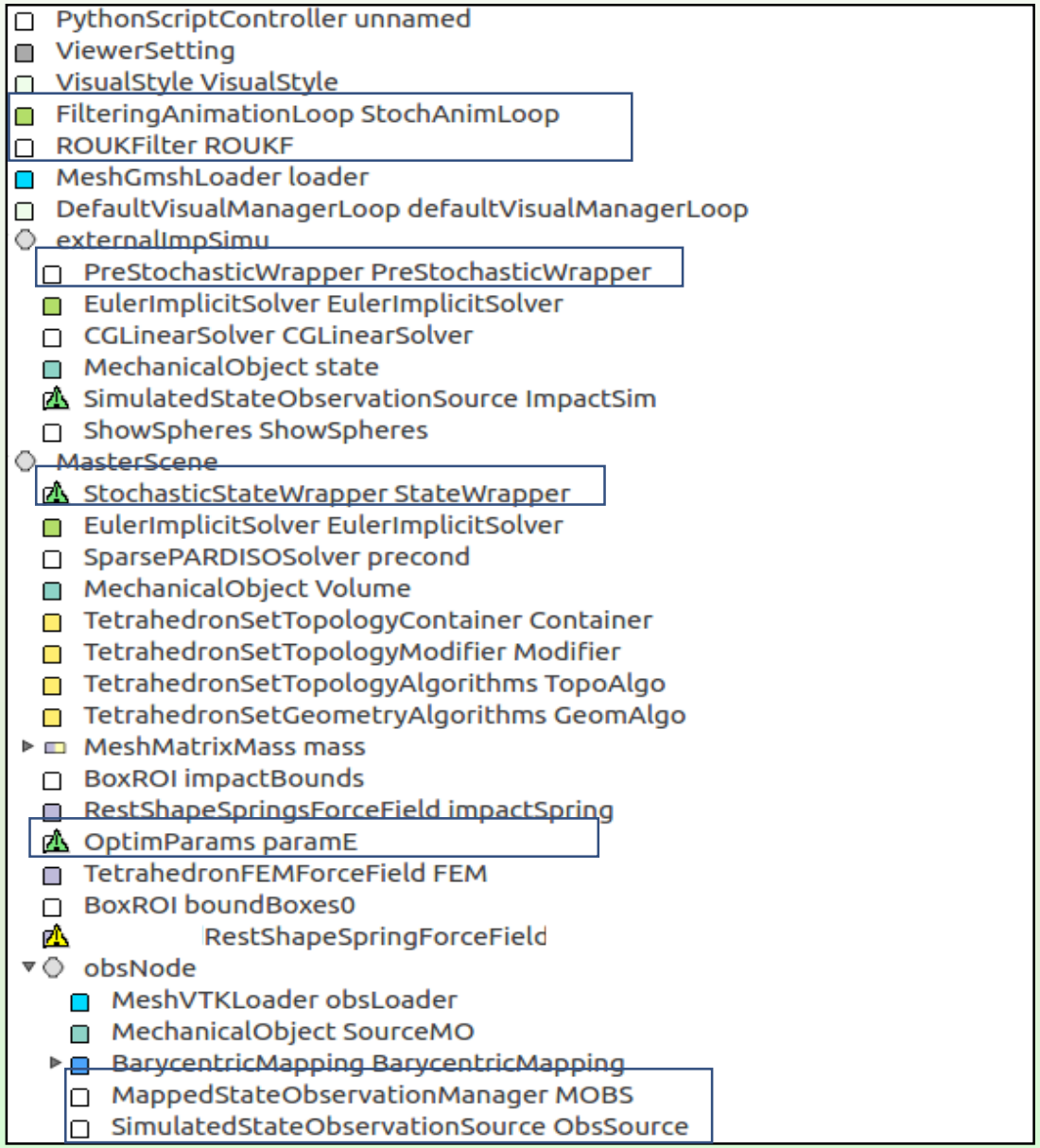
1. **Combine stochastic data assimilation steps with simulation steps based on correspondence**

2. **Create a wrapper around sofa components to avoid code duplications and recoding**

Documentation:

(ReadMe file and doc folder in code)

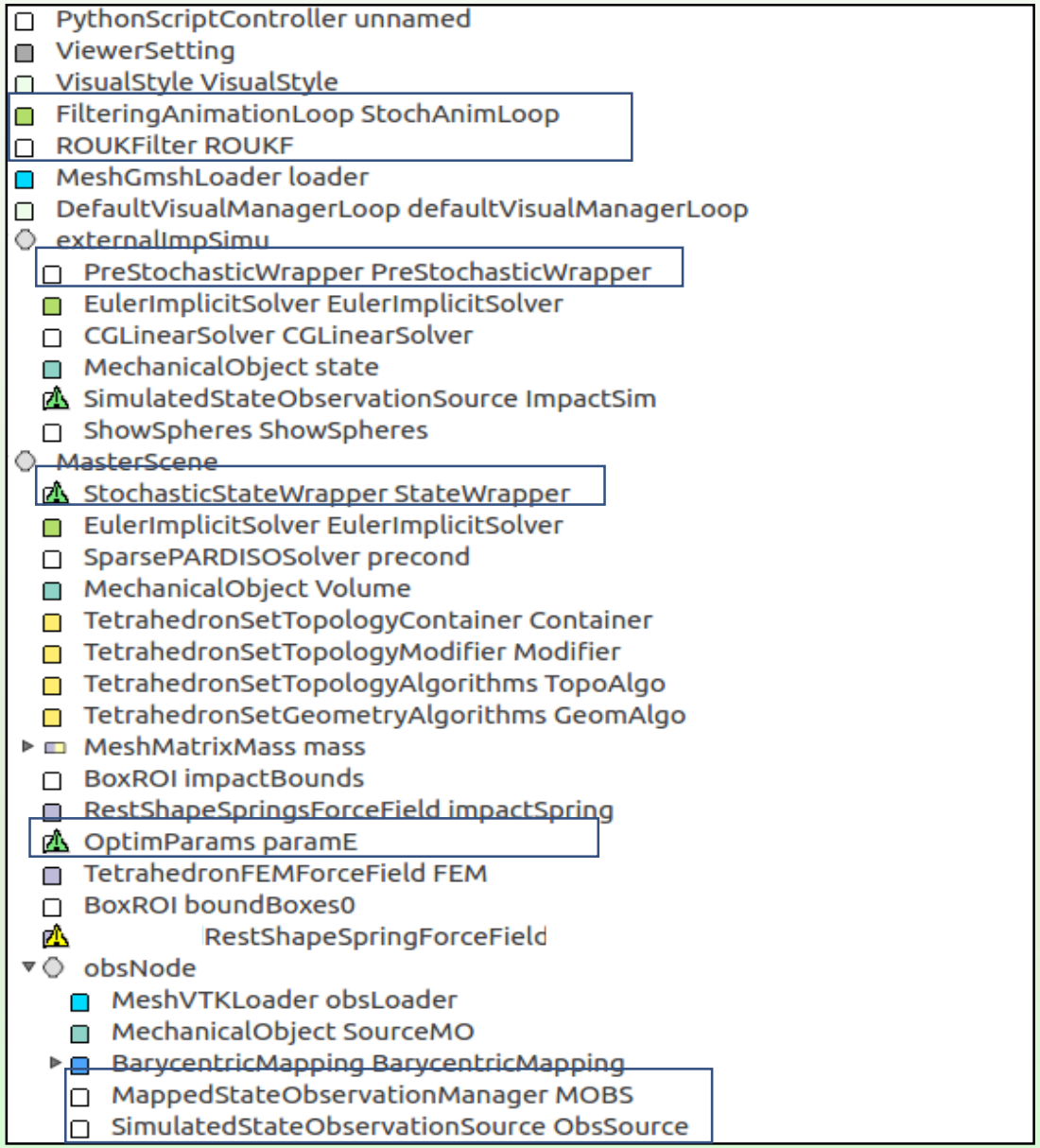
Proposed architecture



High level components:

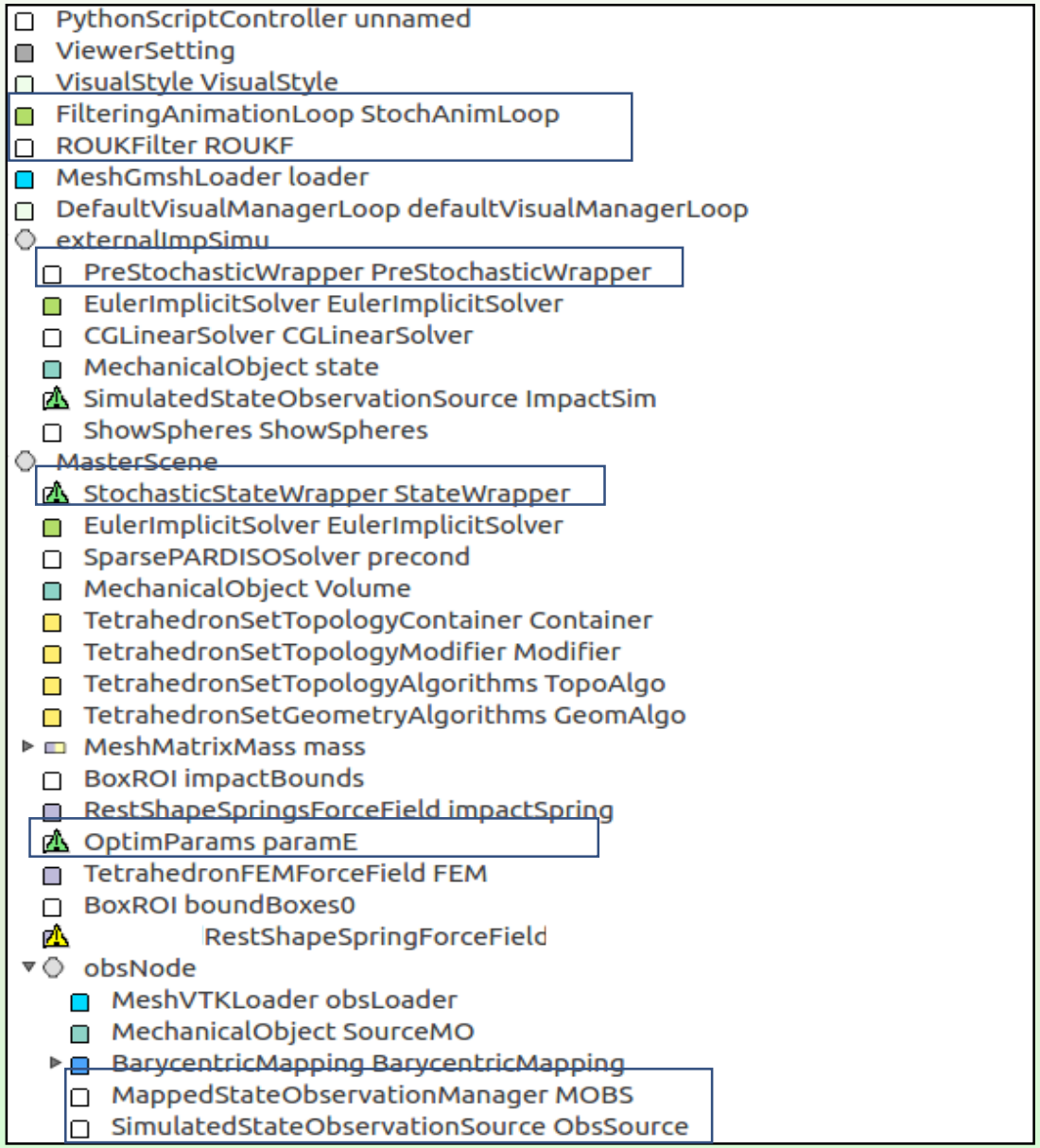
- Filtering animation loop
- Filtering approach
- State wrappers
  - StochasticStateWrapper
  - PreStochasticWrapper
  - OptimParams
- Observation handlers
  - ObservationManager
  - SimulationStateObservationSource

Components description (1)



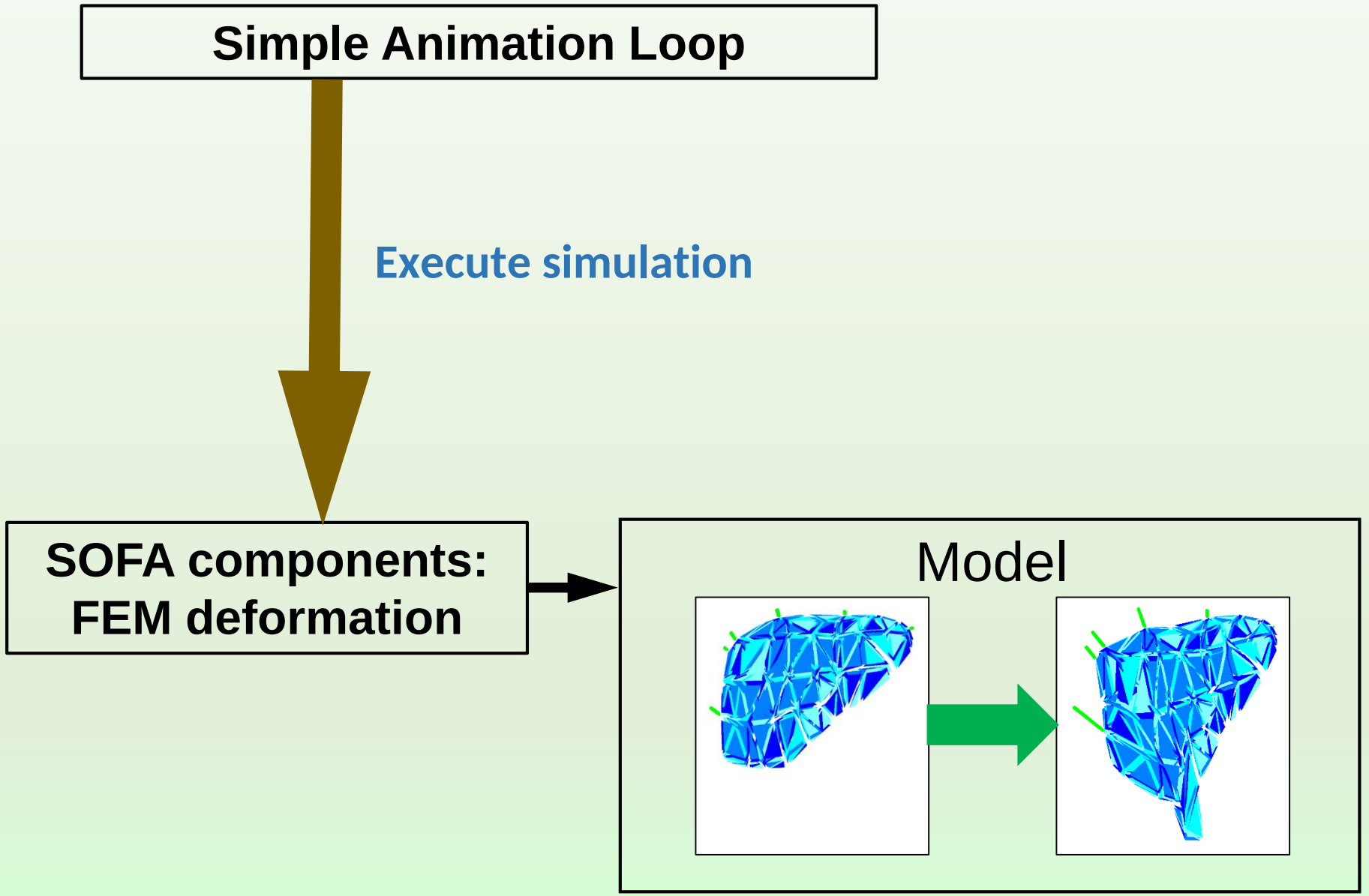
- **Filtering animation loop + filter**
  - **Responsible for execution at each time step**
  - **Implementation of the filtering algorithm step**
- **StochasticStateWrapper**
  - **Interface between filter and SOFA**
  - **Implements *transformState*, which require several model execution in each time step**

Components description (2)

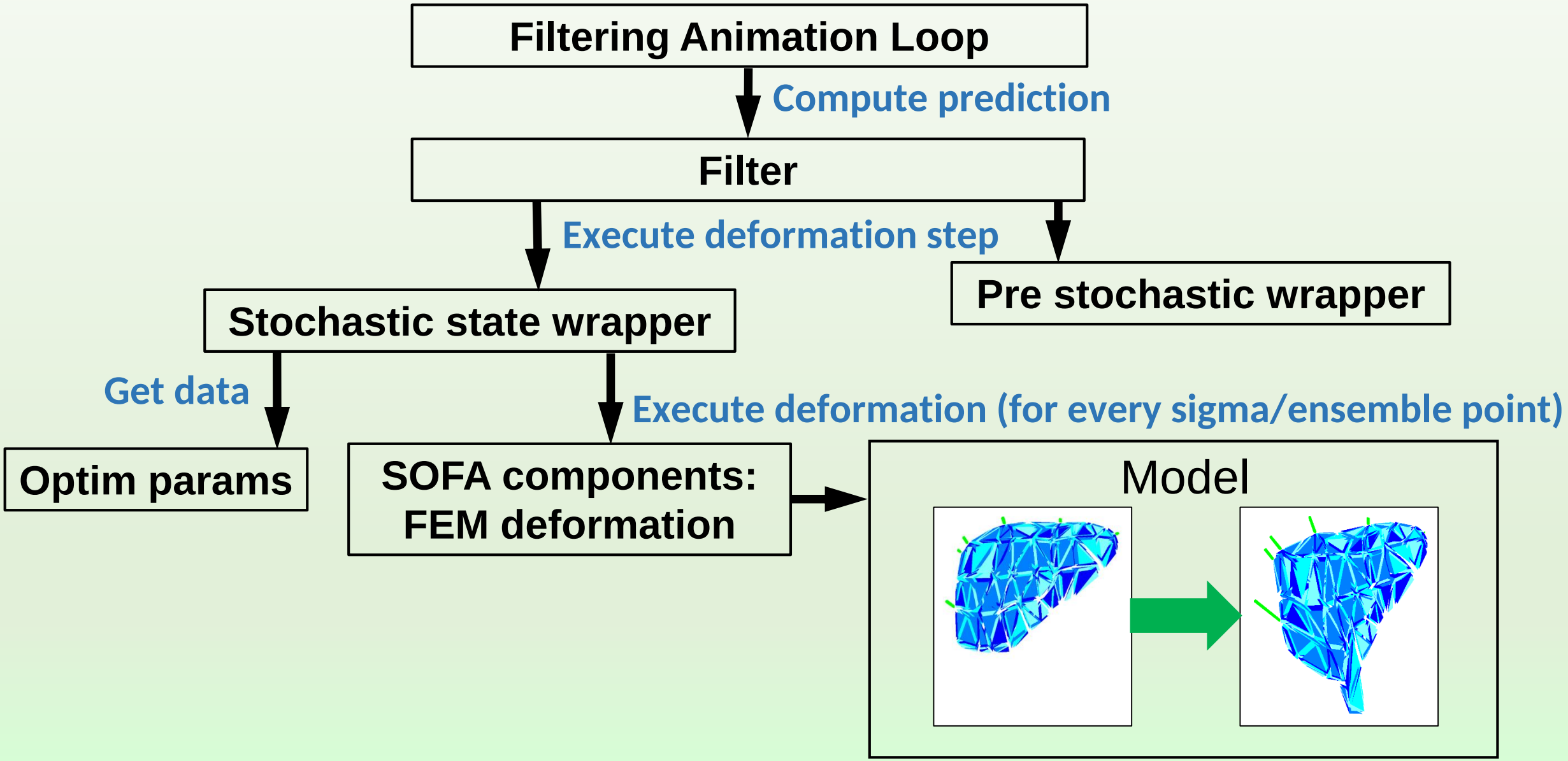


- **OptimParams**
  - **Container of estimated parameters**
  - **Transfers the parameters to physics (typically to forceField)**
  - **The force must be updated properly (no hidden precomputations)**
- **ObservationManager**
  - **Provides metric between reality and model**
  - **Requires observation source (feature extractor, data from sensor, file)**

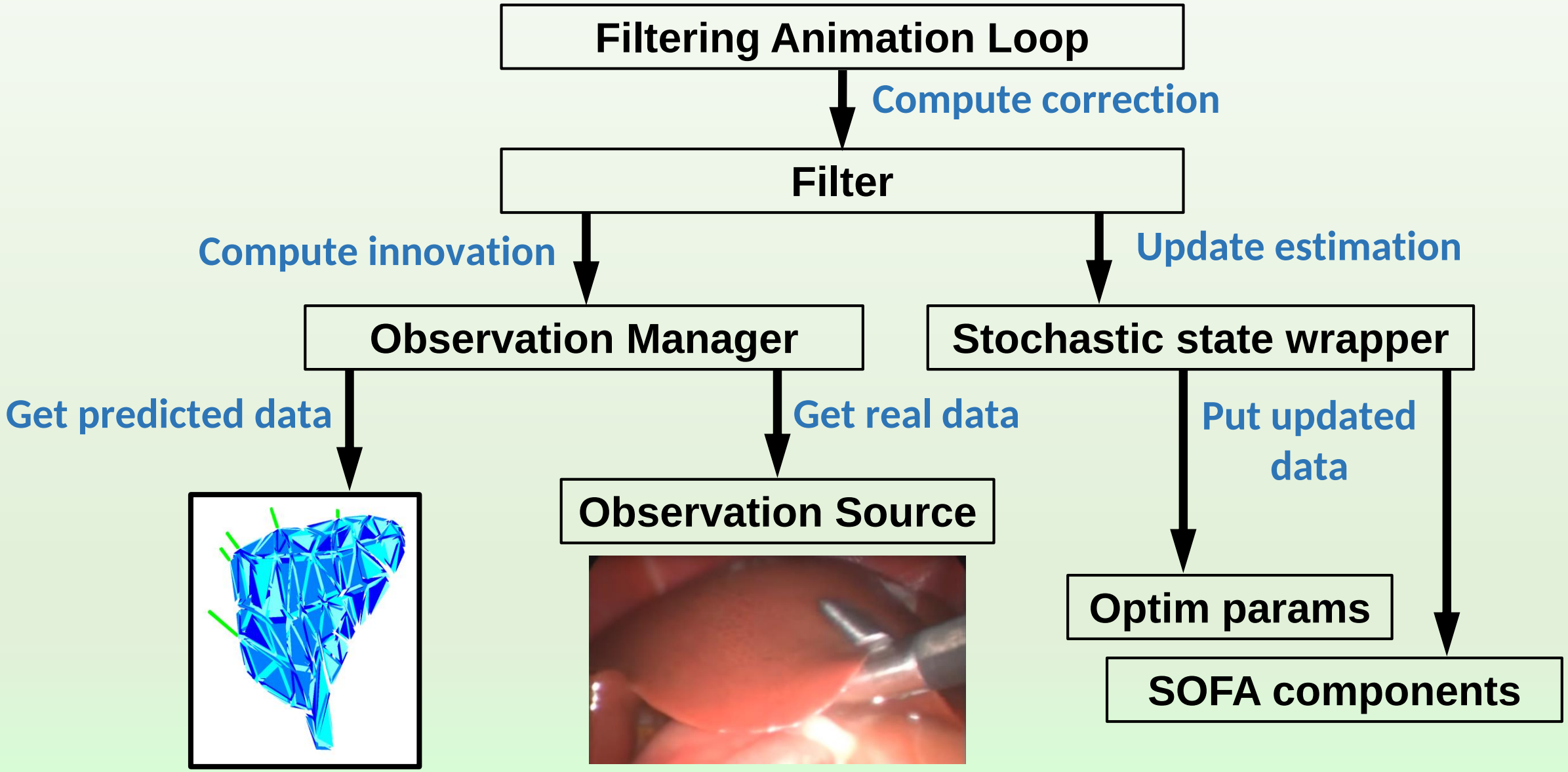
Workflow overview: direct simulation



Workflow overview: prediction



Workflow overview: correction (analysis)



Experimental setup

original values

specific cases

save

Observed positions

use as observations

save

Estimated values

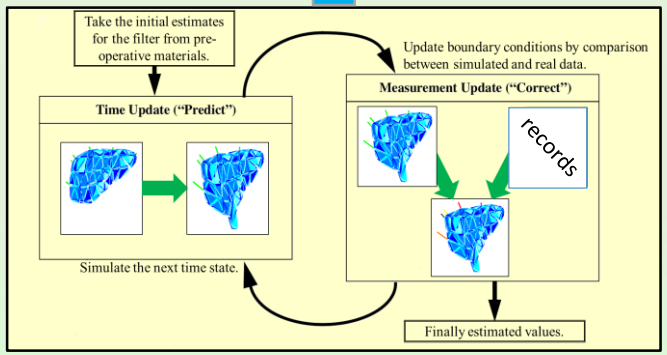
use for simulation

1. Direct simulation to generate liver behavior data

2. Stochastic simulation to estimate BCs

3. Direct simulation to estimate the difference

Periodic forces

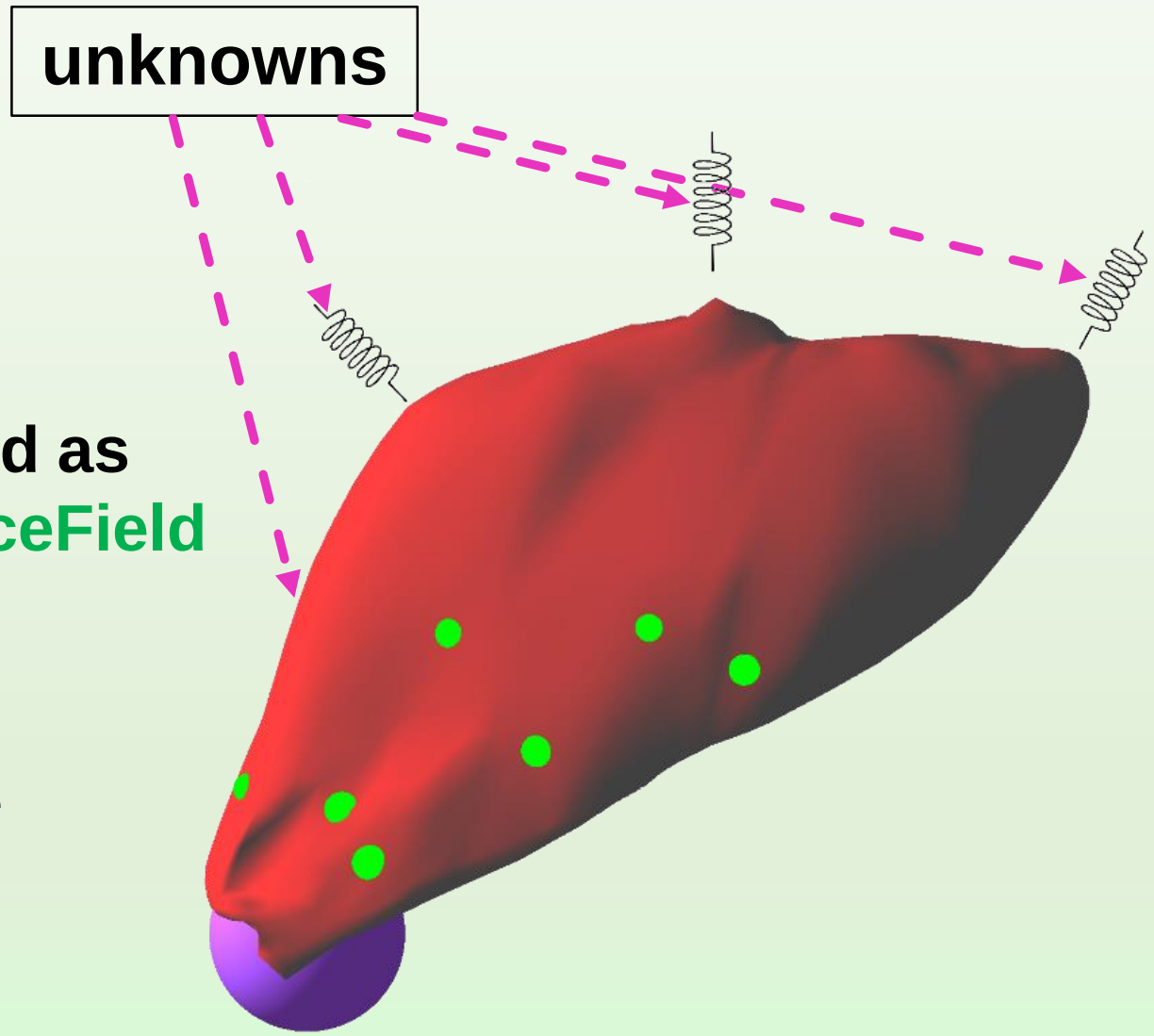


Two cases:

- Estimated BCs
- Fixed BCs

Example: liver with boundary conditions (1)

- The hyperelastic FEM with StVK material is used to simulate liver behavior
- Boundary conditions are presented as (Polynomial)RestShapeSpringForceField with unknown stiffness values
- OptimParams is used to store the spring unknowns

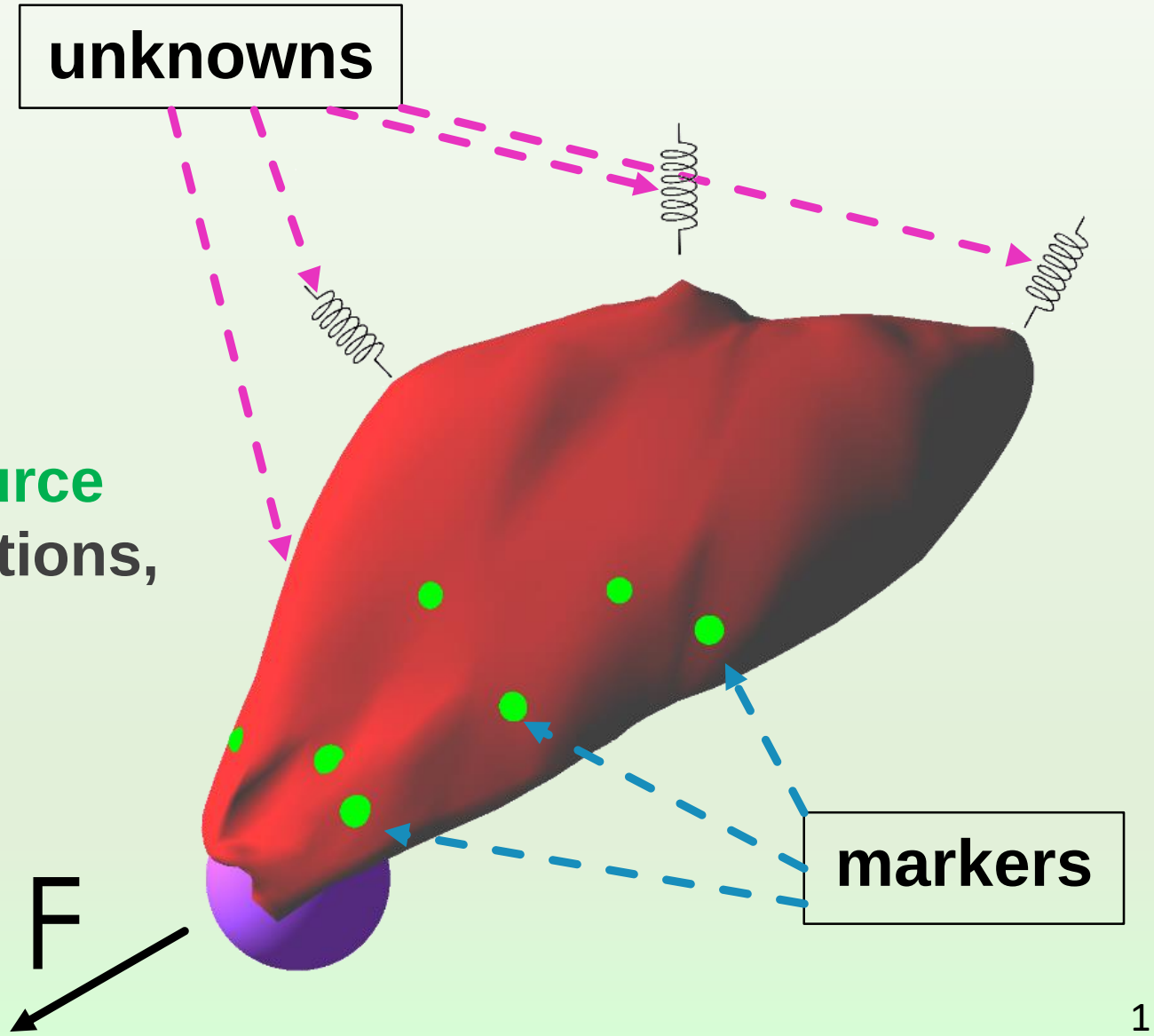


Example: liver with boundary conditions (2)

- Points that present the known features are attached as markers to the liver model using BarycentricMapping

- SimulationStateObservationSource is used to load the markers positions, generated on a separate scene

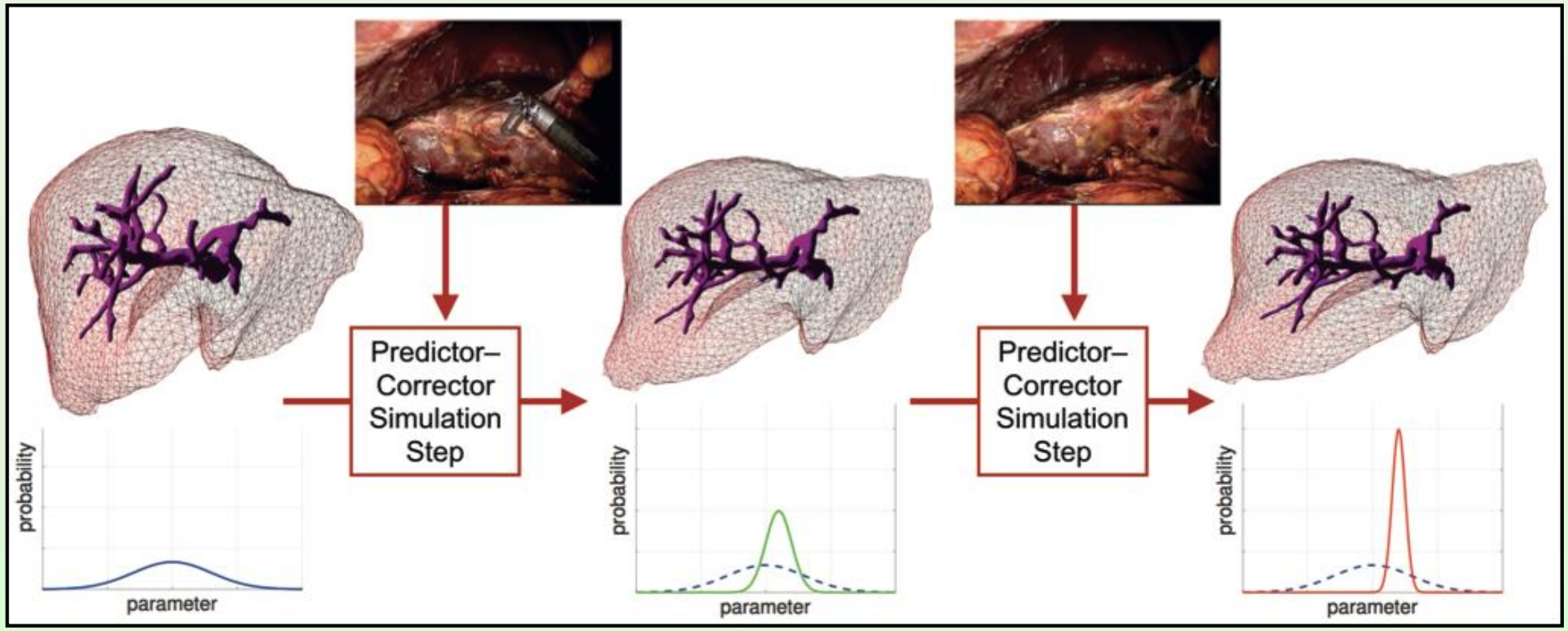
- The external impact is added to scene using PreStochasticWrapper



Real-time estimation

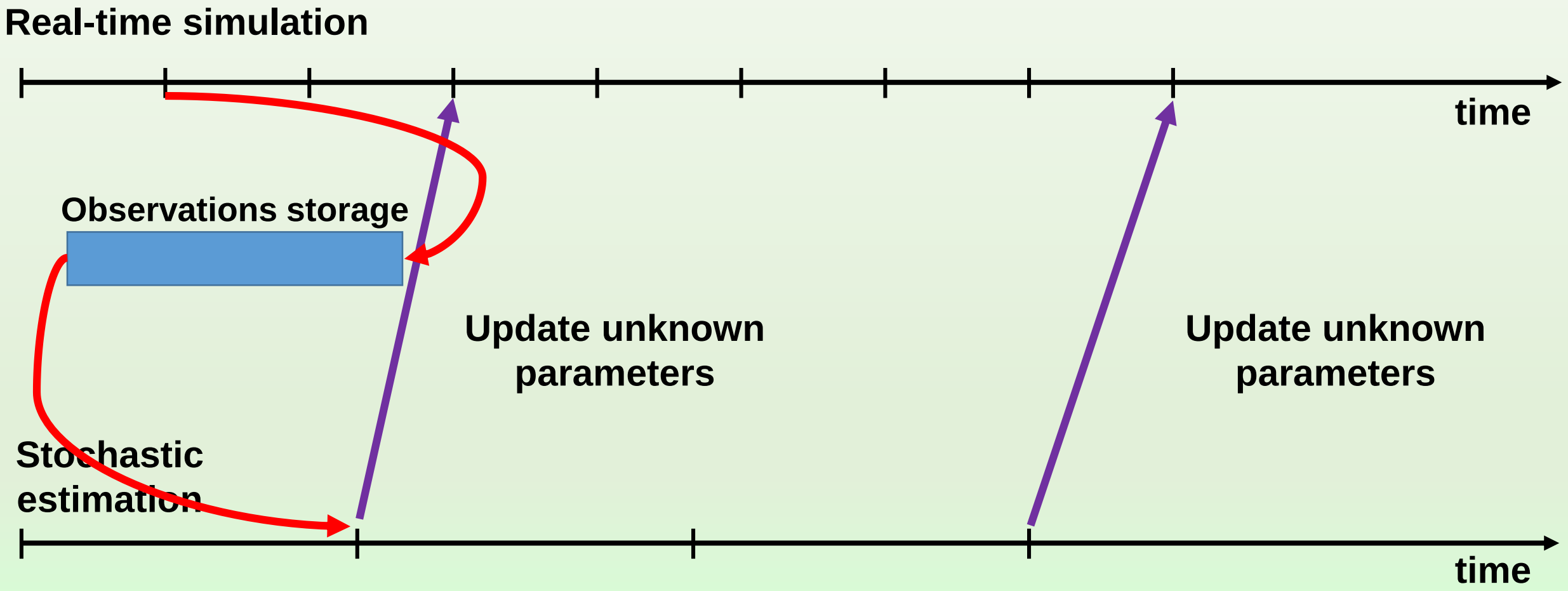
Prediction-correction (analysis) iteration instead of normal simulation step

- **Prediction: perform a simulation transformation given actual state/parameters**
- **Correction: compare the prediction with reality via optimization and update state/parameters**



Delayed estimation

Update estimated parameters after prediction-correction (analysis) iteration



Available data assimilation approaches

4 filters are available:

- **UKFClassicOrig – classical unscented Kalman filter + modifications made by Raffaella**
- **UKFSimmCorr – classical unscented Kalman filter to estimated only unchangeable parameters**
- **ROUKF – reduce order unscented Kalman filter**
- **ETKF – ensemble transform Kalman filter**

Plugin dependences

Required dependences:

- SOFA framework
- SOFA Python/Python3 plugin – for python 2/python 3 compatible scenes
- SOFA pardiso solver – to solve linear system with pardiso (optional)
- Eigen – for covariance/correlation matrix processing
- BLAS – highly efficient matrix operation processing
- pthread – threads for a sigma points /ensemble members computations

Tested setups

- **System solver: dynamic (Euler Implicit) and static (Newton-Raphson)**
- **Identification of homogeneous Young's modulus (Corotational, hyperelastic StVK), stiffness of elastic/hyperelastic springs (including boundary conditions), and contact plane positions/orientations**
- **Estimation of state (velocities, forces) with and without contacts including those modeled as constraints**
- **Pre-conditioner acceleration (multiple executions of model per time step)**
- **Python scenes (including Python3) + YAML configurations and results MatPlotLib plotting**

Conclusion

- Easy to add new data assimilation approaches
- Can be combined with other SOFA plugins
- Possible to use other SOFA scenes without modifications
- Configuration file provides a way to set various setups and compare approaches