Inverse Simulation Tomography

Jannis.Maron@uni-siegen.de

December 19, 2025

Foreword

This document should serve as a guide on how to run each task in the *Inverse Simulation Tomography* project.

Within this project, we implemented three tasks:

1. ***Forward Simulation***: Simulate multi-scattering wave propagation based on a given 3D Refractive Index (RI) distribution and a sequence of predefined poses.
2. ***Pose Optimization***: Find poses for our *Forward Simulation* that best match the phase and amplitude to a recorded video sequence.
3. ***Refractive Index Optimization***: Refine the estimated RI distribution by comparing the optimized poses to the ground truth video sequences.

A description of how to run each task and what outputs we expect can be found in the corresponding section.
Note that this guide does not aim to explain any technical details.

A high-level schematic overview for the combination of Pose and RI optimization can be found in Fig. 1.

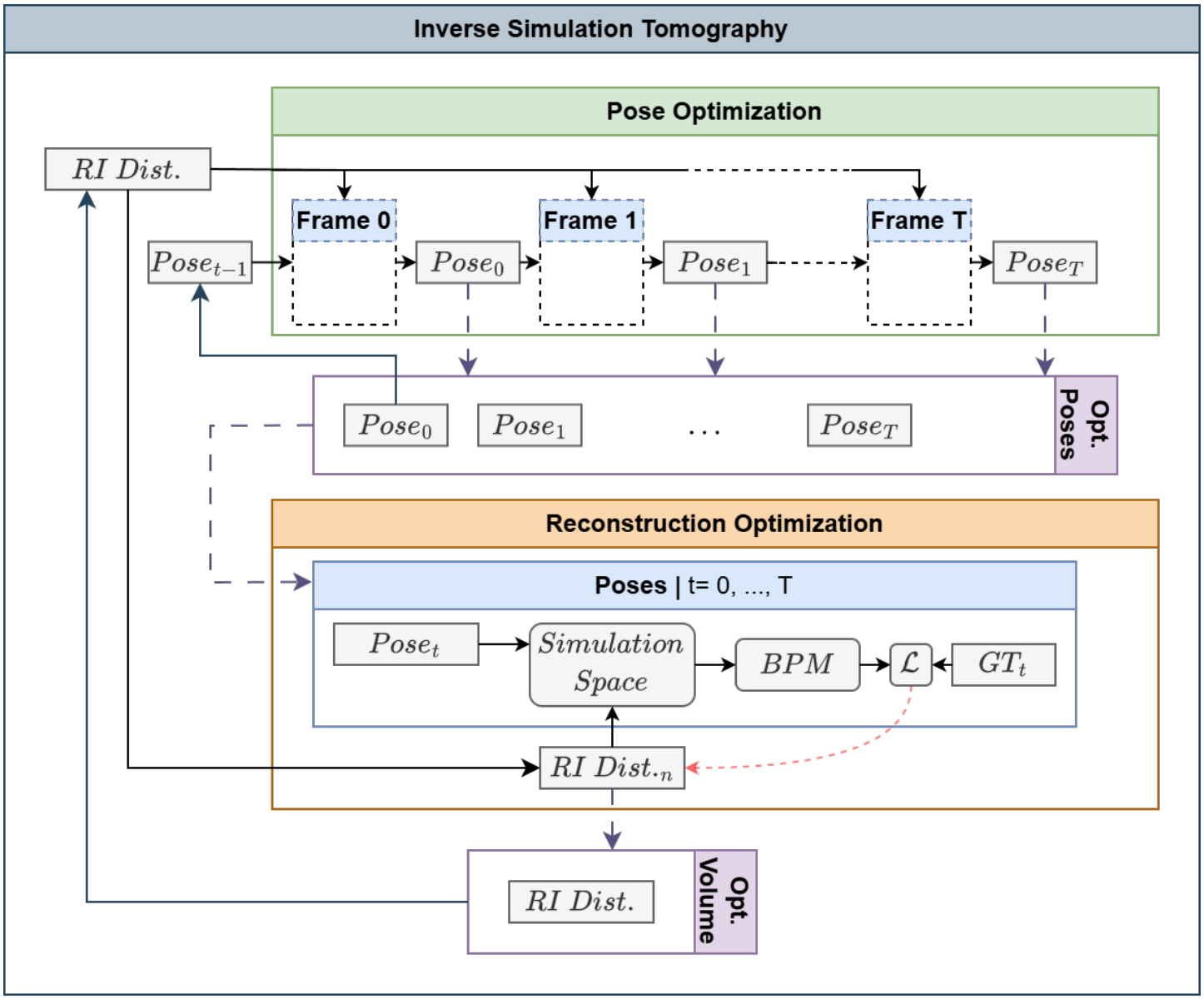

Figure 1: Top-level Overview

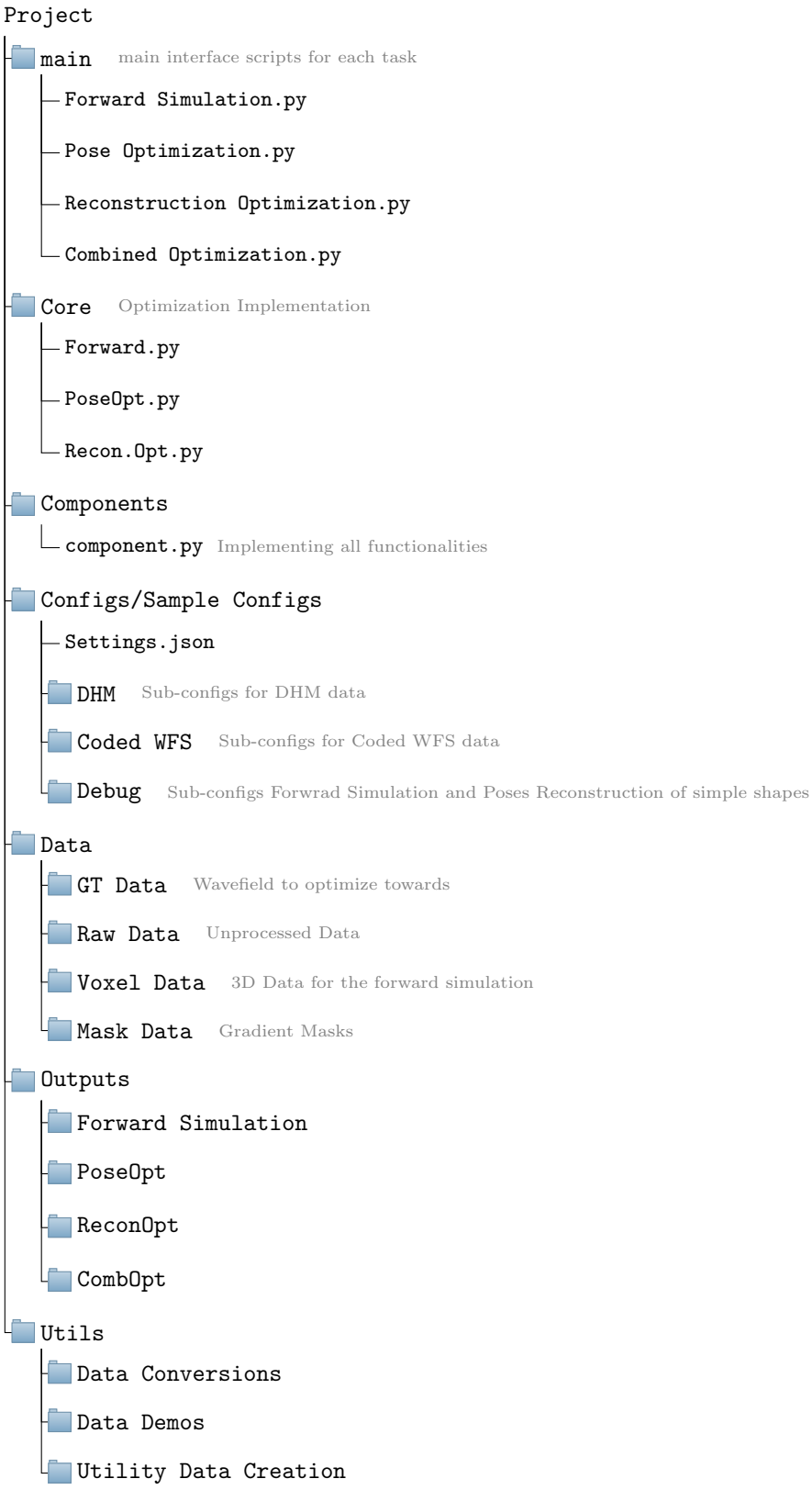

Figure 2: Assumed Project Hierarchy

Contents

1 Settings

Each task is configured through a dedicated settings file. If any main script is executed without specifying a settings file, the user will be prompted to select one interactively. The setting file is structured `.json` documents and consits of two main blocks:

- a **Common** block containing settings shared across all tasks,
- a **Task** block containing task-specific parameters.

Task blocks are described in their corresponding sections[1].

Common Settings

Example: Common Settings Block

```
{
  "Common": {
    "config_dir": "../Configs/Sample Configs",
    "device": null,
    "dtype": "float32",
    "phase_unwrap": false
  }
}
```

Parameter Description

`config_dir`
: Path relative to the executing script that points to the root directory containing all sub-configuration files.

`device`
: Compute device selection.

  - `null`: Automatically select GPU if available, otherwise CPU
  - `"gpu"`, `"cuda"`: Force GPU execution
  - `"cpu"`: Force CPU execution

`dtype`
: Floating-point precision used throughout the computation.

  - `float32`
  - `float64`

  Warning: Higher precision significantly increases memory usage.

`phase_unwrap`
: Boolean flag controlling wrapped versus unwrapped phase visualization. This setting affects visualization only and has no impact on computation.

[1] Reused sub-settings are documented in separate sections.

2 Sub-configs

For improved readability and reusability, configuration parameters that belong to distinct sub-tasks are stored in separate *sub-config* files. Each sub-config is referenced from the corresponding task block in the main settings file. All paths inside sub-config files are defined relative to the `config_dir` specified in the common settings.

The *Simulation Config* and *Data Config* are reused across all three tasks and are therefore documented here to avoid repetition.

2.1 Simulation Config.json

The *Simulation Config* defines the simulation space and the wavefield propagation using the Beam Propagation Method (BPM).

Example: Simulation Config

```
{
  "Base_Grid": {
    "unit": "um",
    "wavelength": 0.640,
    "spatial_resolution": [0.1, 0.1, 0.1],
    "grid_shape": [250, 250, 250],
    "n_background": 1.334,
    "sim_mask": {
        "type": "sphere",
        "size": 110
    },
    "backpropagation_distance": 12.5
  }
}
```

Parameter Description

`unit`
: Base unit used for Simulation Space and Propagation.

  - `"m"`, `"mm"`, `"um"`

  Note: All other unit will be converted to this unit.

`wavelength`
: Wavelength of the propagating wavefield, expressed in `unit`.

`spatial_resolution`
: Physical voxel size of the simulation grid, expressed in `unit`. Specified as $[d_x, d_y, d_z]$.

`grid_shape`
: Number of voxels along each spatial dimension of the simulation grid.
  Specified as $[n_x, n_y, n_z]$.

`n_background`
: Base background refractive index of the simulation space.

`sim_mask`
: Mask to limit the backwards mapping of the voxel object into the simulation space.

  - {}: No mask applied.
  - `type`:
    - `null`: No mask applied.
    - `sphere`: Spherical mask with radius `size`
    - `rect`: Rectangular mask wirth edges of length `size`

`backpropagation_distance`
: Distance (in `unit`) over which the wavefield is propagated backwards after the forward BPM step, in order to reach the focal plane.

2.2 Data Config.json

The *Data Config* specifies the locations of all external data required by the different tasks. Paths are defined relative to the location of the executing script.

Not all data paths are required for every task.

Example: Data Config

```
{
  "Data": {
    "voxel_object": "../Data/Voxel Data/HEK Cells/DHM_Tomog.pt",
    "ground_truth": "../Data/GT Data/HEK Cells/DHM/DHM_Frames.csv",
    "mask": "../Data/Mask Data/HEK Cells/DHM_Mask.pt"
  }
}
```

Parameter Description

`voxel_object`
: Path to the 3D refractive-index distribution.
Required for: {*Forward Simulation, Pose Optimization, Reconstruction Optimization, Combined Optimization*}

`ground_truth`
: Path to a CSV file containing ordered ground-truth frame locations.
Required for: {*Pose Optimization, Reconstruction Optimization, Combined Optimization*}

`mask`
: Path towards 3D Mask file, for masking out voxel object gradients
Optional for: {*Reconstruction Optimization, Combined Optimization*}

3 Forward Simulation

Usage

Given a set of key-frame poses, the estimated refractive-index (RI) distribution (*voxel object*) is placed into the simulation space using interpolated poses. A Beam Propagation Method (BPM) simulation is then performed to obtain the scattered wavefield.
A schematic overview of this process is shown in Fig. 3.

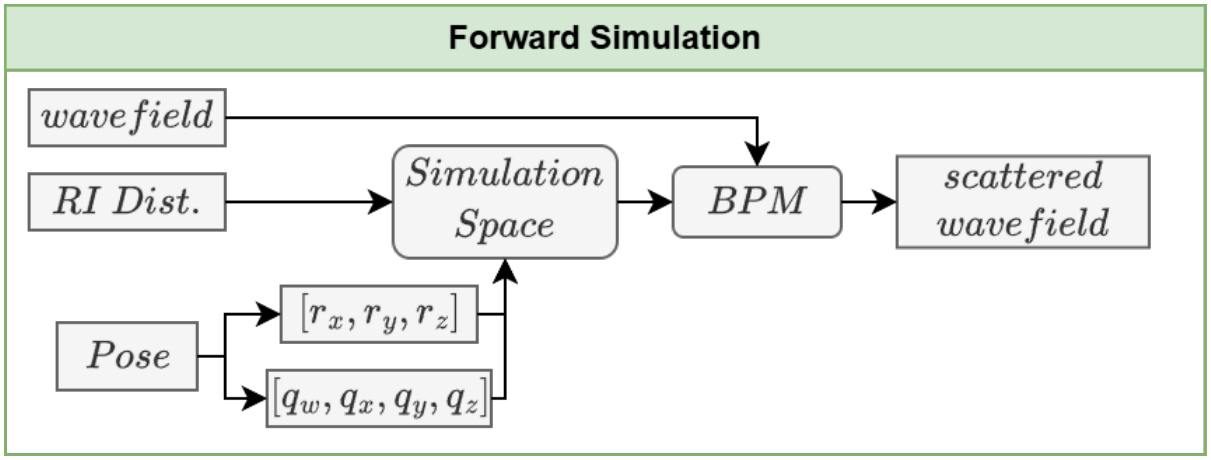

Figure 3: Schematic overview of the forward simulation process.

Script

- `Forward Simulation.py --settings "path/to/Settings.json"`

Required Data

- *Settings* (`.json`)
  - Simulation Config (`.json`)
  - Data Config (`.json`)
  - Forward Config (`.json`)
- *Voxel object* (`.pt`): Estimated RI distribution

3.1 Configs

3.1.1 General Settings

Example: Settings.json

```
{
  "Forward Simulation":{
    "simulation_config_file": "DHM/dhm_simulation.json",
    "data_config_file": "DHM/DHM_data.json",
    "forward_config_file": "Debug/Sim Movements/debug_move.json",

    "output": {
      "output_dir": "../Outputs/Forward Simulation/HEK Cells/DHM/CleanUp",
      "options":  ["phase", "amplitude", "slice", "sim_space", "file"]
    }
  }
}
```

Parameter Description

`simulation_config_file`
Path to the simulation configuration file defining the simulation space and wave propagation.

`data_config_file`
Path to the data configuration file containing the voxel object.

`forward_config_file`
Path to the configuration file defining object motion and post-processing operations.

`output`
see Sec. 3.2.

- `output_dir`: Output directory (created automatically)
- `options`: Output and visualization options
  Options: {`file`, `phase`, `amplitude`, `slice`, `sim_space`}

3.1.2 Forward Config.json

The *Forward Config* defines object motion via poses specified at discrete key frames. Intermediate poses are obtained by interpolation.
If only a single key frame is provided, the pose is assumed to remain constant throughout the entire simulation. If the final key frame occurs before the last simulation timestep, the pose at that key frame is held constant until the end.
In addition, the Forward Config specifies post-processing operations applied independently to the wavefield, amplitude, and phase components. Details on available transforms are provided in Sec. 7.

Example: Forward Config

```
{
  "transforms": {
    "field": {
      "crop": {"x_min":125, "y_min":125, "x_max":-125, "y_max":-125},
      "upscale_complex": {"target_shape":[500,500]}
    },
    "amp": {},
    "phase": {
      "subtract_mean": {}
    }
  },

  "Movement": {
    "time_steps": 5,
    "unit": "um",
    "Positions": [
      { "pos": [25, 25, 25], "time": 0 },
      { "pos": [25, 25, 25], "time": 1 },
      { "pos": [27, 23, 25], "time": 2 }
    ],
    "Offset": [
      { "offset": [0, 0, 0], "time": 0 }
    ],
    "Rotation": [
      { "axis": [1, 0, 0], "theta": 0, "time": 0 },
      { "axis": [1, 0, 0], "theta": 5, "time": 1 },
      { "axis": [1, 1, 0], "theta": 45, "time": 4 }
    ]
  }
}
```

Parameter Description

Post-processing Transforms

`transforms`
See 7 for more information

- `field`: post-processing applied to the scattered wavefield
- `amp`: post-processing applied to the amplitude component.
- `phase`: post-processing applied to the phase component.

Movement

`time_steps`
Total number of simulation timesteps.

`unit`
Unit for position, offset
Options: {`"m"`, `"mm"`, `"um"`}

`Positions`
Object center positions at specified key frames.
$[\{\texttt{"pos"} : [p_x, p_y, p_z],\ \texttt{"time"} : t_i\},\ \ldots]$

`Offset`
Translation vector shifting the object center. Position and rotation are w.r.t. the object center.
$[\{\texttt{"offset"} : [o_x, o_y, o_z],\ \texttt{"time"} : t_i\},\ \ldots]$

`Rotation`
Object rotation specified by axis/angle (deg) representation at key frames.
$[\{\texttt{"axis"} : [a_x, a_y, a_z],\ \texttt{"theta"} : \theta_i,\ \texttt{"time"} : t_i\},\ \ldots]$

3.2 Outputs

Depending on the selected output `options`, various outputs are generated for each timestep. All outputs are stored in `output_dir/run_name`, which is created automatically if it does not exist.
The outputfolder contains three subfolders: {*Data*, *Images*, *Videos*}.

3.2.1 File Output

If `"file"` is in `options`, a `.pt` file is saved for each timestep containing:

`wavefield`
Complex scattered wavefield (only field post-processing)

`amp` Amplitude image (includes amplitude post-processing)

`phase`
Phase image (includes phase post-processing)

`pose_unit`
Unit of pose parameters

`position`
Object position

`offset`
Object offset

`axis`
Rotation axis

`angle`
Rotation angle (degrees)

`transforms`
Applied post-processing transforms

`sim_unit`
: Simulation-space unit

`grid_shape`
: Simulation grid size

`spatial_resolution`
: Simulation grid resolution

`wavelength`
: Incident wavelength

3.2.2 Images

If any of {`"phase"`, `"amplitude"`, `"slice"`, `"sim_space"`} are set in `options`, corresponding images are saved to the *Images* subfolder.

- Slice of the xy-plane of the simulation space at a specific index.
- 3D rendering of the simulation space
- Phase image: `unwrap_phase(np.angle(field))`
- Amplitude image: `np.abs(field)`

3.2.3 Videos

If any of {`"phase"`, `"amplitude"`, `"slice"`, `"sim_space"`} are set in `options`, corresponding Videos over all timesteps are saved to the *Videos* subfolder.

- Sequence of Slice images over all timesteps.
- Sequence of 3D renderings of the Simulation Space.
- Sequence of Phase of output wavefields over all timesteps.
- Sequence of Amplitude of output wavefields over all timesteps.

4 Pose Optimization

Usage

Given a recorded video sequence of scattered wavefields, the goal of pose optimization is to reconstruct the pose of the voxel object for every frame in the sequence.
This is achieved by optimizing the pose parameters of the *Forward Simulation* such that the simulated scattered wavefield best matches the recorded data.
We utilize a frame-by-frame approach, where we optimize each frame individually. Since video sequences are usually temporally smooth, we utilize the best pose of the previous frame as initialization for the next frame, to decrease the number of iterations until convergence.

A high-level overview of the frame-by-frame strategy is shown in Fig. 4, while the per-frame optimization workflow is illustrated in Fig. 5.

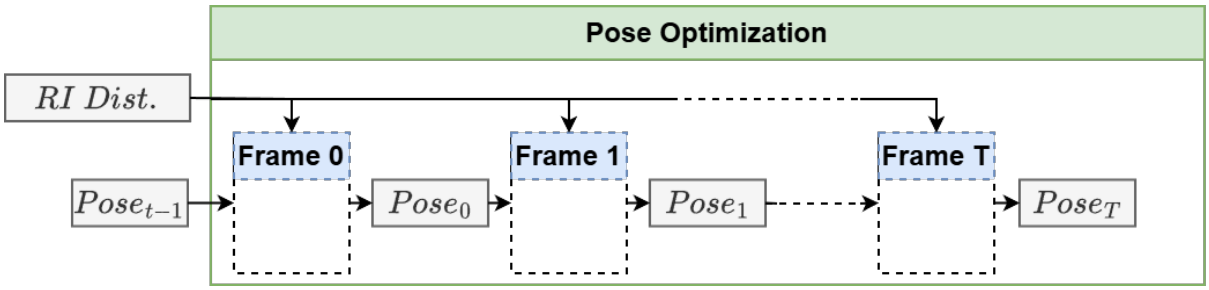

Figure 4: Frame-by-frame pose optimization overview.

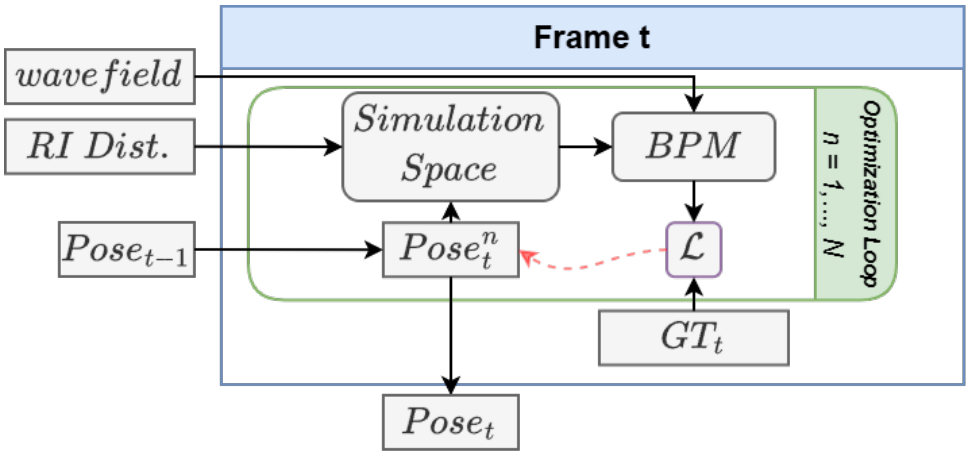

Figure 5: Per-frame pose optimization workflow.

Script

- `Pose Optimization.py --settings "path/to/Settings.json"`

Required Data

- **Settings file** (`Settings.json`)
  - Simulation Config
  - Data Config
  - PoseOpt Config
- **Voxel object** (`.pt`): Estimated RI distribution
- **Target data**: Recorded scattered wavefield sequence

4.1 Configs

4.1.1 General Settings

Example: Settings.json

```
{
  "PoseOpt":{
    "simulation_config_file": "DHM/dhm_simulation.json",
    "data_config_file": "DHM/DHM_data.json",
    "pose_opt_config_file": "DHM/DHM_pose_opt.json",

    "output": {
      "active": true,
      "output_dir": "../Outputs/PoseOpt/HEK Cells/DHM/Sample",
      "options": ["losses", "amps", "phases", "slices", "renders"]
    }
  }
}
```

Parameter Description

`simulation_config_file`
: Path towards the config file defining the general simulation.

`data_config_file`
: Path to the data configuration file containing voxel object and target data locations.

`pose_opt_config_file`
: Path to the pose-optimization configuration file.

`output`
: Output configuration (see Sec. 4.2).

- `active`: Enable or disable logging
- `output_dir`: Output directory (created automatically)
- `options`: Visualization and logging options
  Options: {`amps`, `phases`, `slices`, `renders`}

4.1.2 PoseOpt Config.json

The *PoseOpt Config* specifies the optimization procedure used to estimate object poses over a sequence of frames. We handle the first frame within a sequence separately from all the other sequential frames. Therefore, we divide our PoseOpt Config into a general *Training*, a *Initial* farme, and *Sequential* frame section.

Example: PoseOpt Config

```
{
  "PoseOpt": {
    "gt_transforms": {
      "field": {},
      "amp": {
        "sqrt":{}
      },
      "phase": {
        "gaussian_blur": {"sigma": 7.5},
        "subtract_mean": {}
      }
    },

    "sim_transforms": {
      "field": {
        "upscale_complex": {"target_shape":[500,500]}
      },
      "amp": {},
```

```
      "phase": {
        "subtract_mean": {}
      }
    },

    "weights": [20, 75, 0.3, 300],

    "start_frame": 0,
    "end_frame": null,
    "frame_steps": 1,

    "unit": "um",
    "Position": [12.5, 12.5, 10.5],
    "Axis": [1, 0, 0],
    "Angle": -15,

    "Initial":{
      "epochs": 200,

      "optimizer":{
        "Position": {"lr": 0.01},
        "Quaternion": {"lr": 0.005}
      },

      "scheduler":{
        "milestones": [175, 190],
        "gamma": 0.1
      },

      "regularizers" : {}
    },

    "Sequential": {
      "epochs": 20,

      "optimizer":{
        "Position": {"lr": 0.001},
        "Quaternion": {"lr": 0.0005}
      },

      "scheduler":{
        "milestones": [15,18],
        "gamma": 0.1
      },

      "regularizers" : {
        "Position": {"var": "Position", "lambda": 0.05},
        "Quaternion": {"var": "Quaternion", "lambda": 5, "w": [1e-5,1e-5,1e-3,1e-3], "v
            ": [1e-5,1e-5,1e-3,1e-3]}
      }
    }
}
```

Parameter Description

Training

`gt_transforms`
: Domain-adaptation transforms applied to the recorded data (see Sec. 7).

`sim_transforms`
: Domain-adaptation transforms applied to the simulated wavefield (see Sec. 7).

`weights`
: Component-wise weights for the loss function.

`start_frame`, `end_frame`, `frame_steps`
: Frame indices defining the optimization range and sampling.

`unit`
: Unit used for positional parameters.

`Position, Axis, Angle`
Initial pose parameters for the first frame.

Initial Frame

`epochs`
Number of optimization epochs for the first frame.

`optimizer`
Adam Optimizer for the first frame

- `Position`, `Quaternion`
  - `lr`: Learning rate for the respective parameter

`scheduler`
MultiStepLR scheduler for the first frame

- `milestones`: Epochs at which to decay the learning rate
- `gamma`: Decay factor

`regularizers`
By default, no regularization is applied for the initial frame.

Sequential Frames

`epochs`
Number of optimization epochs per subsequent frame.

`optimizer`
Adam Optimizer configuration for sequential frames.

- `Position`, `Quaternion`
  - `lr`: Learning rate for the respective parameter

`scheduler`
MultiStepLR scheduler for sequential frames.

- `milestones`: Epochs at which to decay the learning rate
- `gamma`: Decay factor

`regularizers`
L2 Position and KalmanFilter rotation regularization for sequential frames

- `Position`
  - `var`: Variable to regularize (`Position`)
  - `lambda`: Regularization strength
- `Quaternion`
  - `var`: Variable to regularize (`Quaternion`)
  - `lambda`: Regularization strength
  - `w`: Process noise covariance (float or 4-element list)
  - `v`: Measurement noise covariance (float or 4-element list)

4.2 Outputs

Depending on the selected output options, pose optimization produces per-frame outputs as well as a summary of the whole sequence. All outputs are written to `output_dir/run_name`, which is created automatically if it does not exist.
The directory contains three subfolders: {*Summary*, *Configs*, *Frames*}.

4.2.1 Summary

Final summary results over all frames. Visualizations are dependent on the selected `options` in the settings file.

`best_setting.json`
: Best pose and corresponding loss for each frame.

`Losses.png`
: Plot of final loss values for each frame.

`best_epochs.png`
: Plot of best iteration index for each frame.

`best_positions.png`
: Plot of optimized *(x,y,z)* positions over all frames.
Visualized as: $(x := red, y := green, z := blue)$

`best_quaternions.png`
: Plot of optimized *(w,x,y,z)* quaternions over all frames.
Visualized as individual component plots.

`best_axes.png`
: Plot of optimized *(x,y,z)* rotation axes over all frames (converted from quaternions).
Visualized as: $(x := red, y := green, z := blue)$

`best_angles.png`
: Plot of optimized rotation angles over all frames (converted from quaternions).

`Best Amplitude.avi`
: Simulated vs. recorded amplitude comparison video.

`Best Phase.avi`
: Simulated vs. recorded phase comparison video.

`Best Slice.avi`
: Center slice of the simulation space for each optimized pose.

`Best Render.avi`
: 3D render of the simulation space for each optimized pose.

4.2.2 Configs

Copies of all configuration files used during execution.

- `Settings.json`
- `Simulation.json`
- `Data.json`
- `PoseOpt.json`

4.2.3 Frames

Per-frame optimization logs and visualizations.

`Losses.csv`
: Loss values for all iterations of current frame.

`Positions.csv`
: Position values for all iterations of current frame.

`Rotations.csv`
: Quaternion values for all iterations of current frame.

`Frame Loss.png`
: Loss plot (total, data, and regularization) for current frame.

`amp/`, `phase/`
: Periodic comparisons between simulated and recorded data.

`Best Amplitude.png`
: Best-pose amplitude comparison.

`Best Phase.png`
: Best-pose phase comparison.

`Best Slice.png`
: Center slice of the simulation space.

`Best Render.png`
: 3D render of the simulation space.

5 Reconstruction Optimization

Usage

Given a recorded video sequence of scattered wavefields and a list of poses, reconstruction optimization updates the estimated refractive-index (RI) distribution of the voxel object such that the simulated wavefields generated by the *Forward Simulation* better match the recorded data.
The voxel object is updated only after processing the full dataset; that is, one optimization step corresponds to a complete pass through all recorded frames.

A schematic overview of the reconstruction process is shown in Fig. 6.

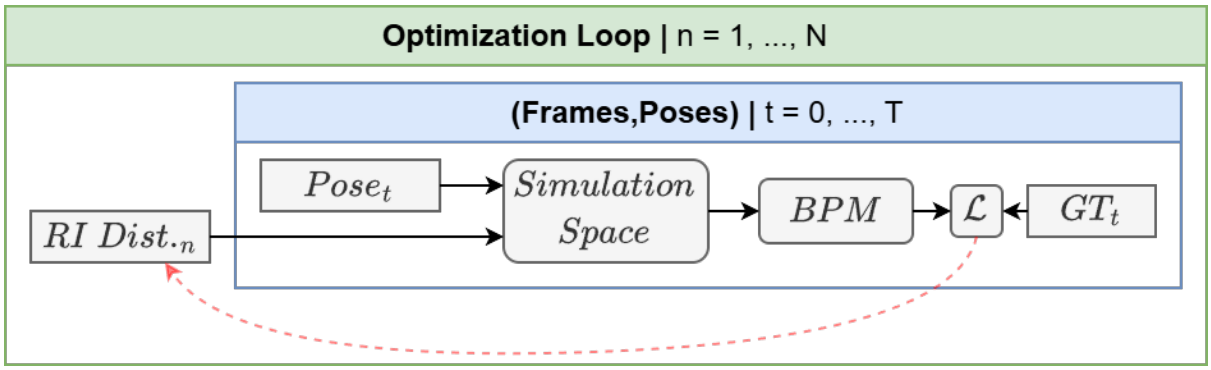

Figure 6: Reconstruction optimization workflow.

Script

- `Reconstruction Optimization.py --settings "path/to/Settings.json"`

Required Data

- **Settings file** (`Settings.json`)
  - Simulation Config
  - Data Config
  - ReconOpt Config
- **Voxel object** (`.pt`): Initial RI distribution
- **Poses** (`.json`): i.e. output of Pose Optimization
- **Target data**: Recorded scattered wavefield sequence

Optional

- **Gradient mask** (`.pt`): Mask to restrict voxel object updates

5.1 Configs

5.1.1 General Settings

Example: Settings.json

```
{
  "ReconOpt":{
    "simulation_config_file": "DHM/dhm_simulation.json",
    "data_config_file": "DHM/DHM_data.json",
    "recon_opt_config_file": "DHM/dhm_recon_opt.json",
    "recon_poses_file": "Debug/Optimized Poses/dhm_poses_sample.json",

    "output": {
      "active": true,
```

```
      "output_dir": "../Outputs/ReconOpt/HEK Cells/DHM/Sample",
      "options": ["slices", "renders", "amps", "phases"]
    }
  }
}
```

Parameter Description

`simulation_config_file`
: Path to the simulation configuration file.

`data_config_file`
: Path to the data configuration file containing voxel object and target data locations.

`recon_opt_config_file`
: Path to the reconstruction optimization configuration file.

`recon_pose_file`
: Path to a JSON file containing optimized poses for all frames.

`output`
: Output configuration (see Sec. 5.2).

- `active`: Enable or disable logging
- `output_dir`: Output directory (created automatically)
- `options`: Visualization options
  Options: {`amps`, `phases`, `slices`, `renders`}

5.1.2 ReconOpt Config.json

Example: ReconOpt Config

```
{
  "ReconOpt":{
    "gt_transforms": {
      "field": {},
      "amp": {
        "sqrt":{}
      },
      "phase": {
        "gaussian_blur": {"sigma": 7.5},
        "subtract_mean": {}
      }
    },

    "sim_transforms": {
      "field": {
        "upscale_complex": {"target_shape":[500,500]}
      },
      "amp": {},
      "phase": {
        "subtract_mean": {}
      }
    },

    "epochs": 20,
    "weights": [20, 75, 0.3, 300],

    "optimizer": {
      "Voxel Object": {"lr": 1e-4}
    },

    "scheduler": {
      "milestones": [15,18],
      "gamma": 0.1
    },
```

```
    "regularizers" : {
      "TV Reg": {"lambda": 500}
    }

  }
}
```

Parameter Description

`gt_transforms`
Domain-adaptation transforms applied to recorded data (see Sec. 7).

`sim_transforms`
Domain-adaptation transforms applied to simulated wavefields (see Sec. 7).

`epochs`
Number of training iterations. Each iteration corresponds to one full pass through the dataset.

`weights`
Component-wise weights for the loss function.

`optimizer`
Adam Optimizer configuration used to update the voxel object

- `Voxel Object`
  - `lr`: Learning rate for the respective parameter

`scheduler`
MultiStepLR scheduler configuration.

- `milestones`: Epochs at which to decay the learning rate
- `gamma`: Decay factor

`regularizers`
Total Variation regularization for the voxel object

- `TV Reg`:
  - `lambda`: Regularization strength

5.2 Outputs

Depending on the selected output options, reconstruction optimization produces outputs after each full dataset pass. All outputs are written to `output_dir/run_name`, which is created automatically if it does not exist.
The directory contains three subfolders: {*Summary*, *Configs*, *Epochs*}.

5.2.1 Summary

Final reconstruction results and optimization progress summaries.

`voxel_object.pt`
Optimized voxel object.

`Losses.png`
Average total, data, and regularization loss over all training iterations, normalized by the number of frames.

`Amps/`, `Phases/`
Comparison plots between simulated and recorded data for all poses (no domain adaptation applied).

`Amplitude.avi`, `Phase.avi`
Video of amplitude and phase comparisons over all frames.

`Slice_xxx.png`, `Render_xxx.png`
: Best reconstructed voxel object visualizations, where `xxx` denotes the epoch of best performance.

`Voxel Object Slice.avi`
: Video of the optimization progress of the center slice of the voxel object over all train iterations.

`Voxel Object Render.avi`
: Video of the optimization progress of the 3D render of the voxel object over all train iterations.

5.2.2 Configs

Copies of all configuration files used during execution.

- `Settings.json`
- `Simulation.json`
- `Data.json`
- `ReconOpt.json`
- `ReconPoses.json`

5.2.3 Epochs

Per-epoch visualizations and diagnostics.

- Current voxel object visualization (Slice + render)
- Amplitude and phase comparisons for the last pose in the dataset

6 Combined Optimization

Usage

Combined optimization jointly optimizes both the object poses and the refractive-index (RI) distribution of the voxel object given a recorded video sequence of scattered wavefields.
This is achieved by alternating between pose optimization (see Sec. 4) and reconstruction optimization (see Sec. 5) for a set number of cycles.
A high-level overview of the combined optimization procedure is shown in Fig. 7.

At the momement the number of combined optimization cycles is fixed in the code itself (pose optimization followed by reconstruction optimization). Furthermore, Combined Optimization expects a *gaussian_blur* domain adaptation transform for the phase component in the *gt_transforms* of both the PoseOpt and ReconOpt configs. The sigma value of the blur is used to progressively decrease the amount of domain adaptation over the combined optimization cycles.

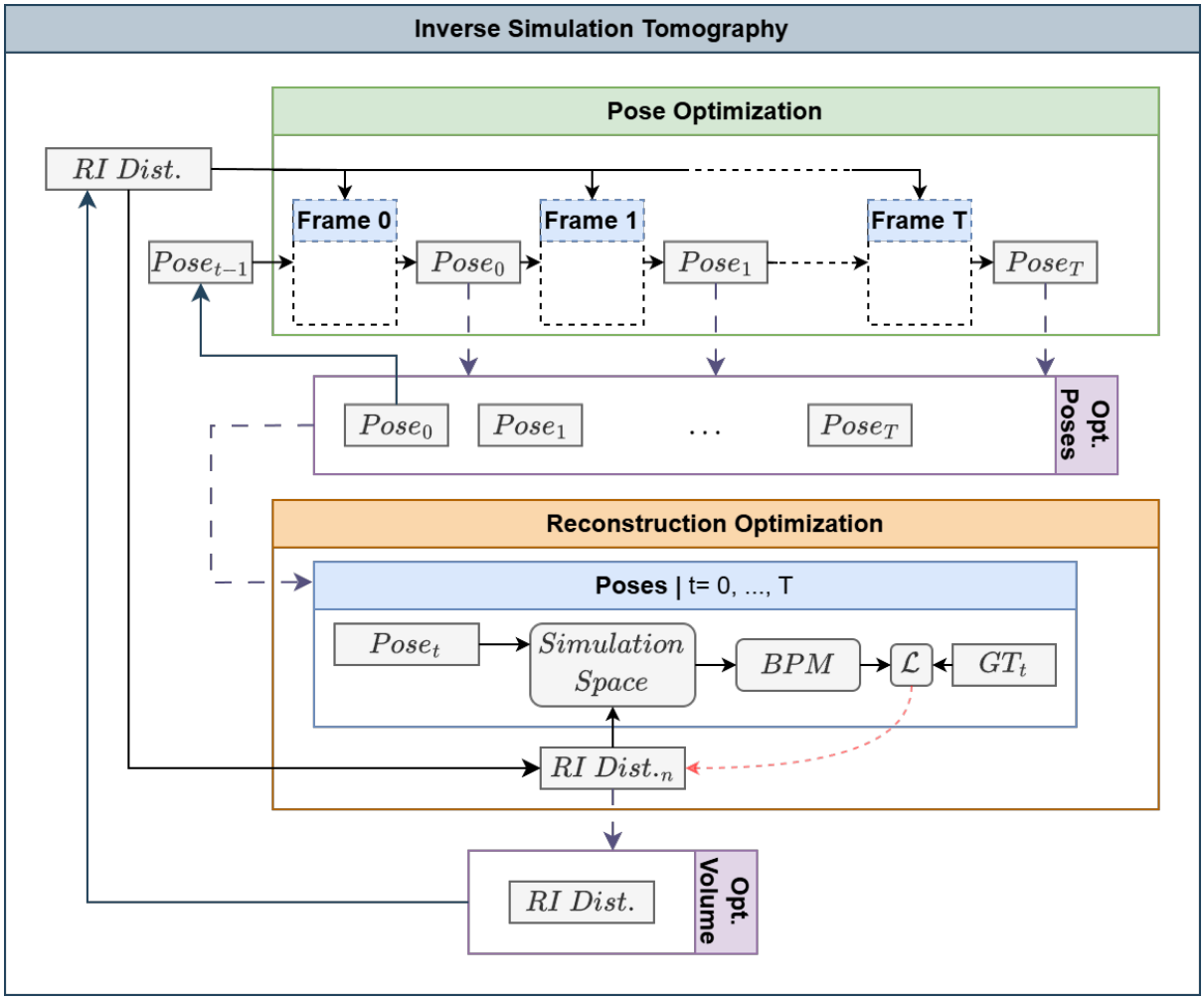

Figure 7: Combined optimization workflow.

Script

- `Combined Optimization.py --settings "path/to/Settings.json"`

Required Data

- **Settings file** (`Settings.json`)
  - Simulation Config

  - Data Config
  - PoseOpt Config
  - ReconOpt Config

- **Voxel object** (`.pt`): Initial RI distribution

Optional

- **Gradient mask** (`.pt`): Mask to restrict voxel object updates

6.1 Configs

6.1.1 General Settings

Example: Settings.json

```
{
  "CombOpt": {
    "simulation_config_file": "DHM/dhm_simulation.json",
    "data_config_file": "DHM/DHM_data.json",
    "pose_opt_config_file": "DHM/dhm_pose_opt.json",
    "recon_opt_config_file": "DHM/dhm_recon_opt.json"

    "output": {
      "output_dir": "../Outputs/CombOpt/HEK Cells/DHM/12_12_25 Long + Always Update",
      "options": ["slices", "renders", "amps", "phases"]
    }
  }
}
```

Parameter Description

`simulation_config_file`
: Path to the simulation configuration file.

`data_config_file`
: Path to the data configuration file containing voxel object and target data locations.

`pose_opt_config_file`
: Path to the pose-optimization configuration file.

`recon_opt_config_file`
: Path to the reconstruction-optimization configuration file.

`output`
: Output configuration (see Sec. 6.2).

  - `output_dir`: Output directory (created automatically)
  - `options`: Visualization options
    Options: {`amps`, `phases`, `slices`, `renders`}

6.2 Outputs

For every combined optimization cycle, outputs from both the PoseOpt and ReconOpt procedures are saved in separate subfolders within `output_dir/run_name`.
The directory contains a subfolder for each cycle named: `iter xxx`, where `xxx` denotes the cycle index (starting from 0).
Within each cycle subfolder, outputs from both optimization procedures are stored in their respective subfolders.

7 Domain Adaptation / Post-Processing Transforms

Transforms serve two main purposes:

1. Post-processing simulation outputs.
2. Domain adaptation during training, increasing similarity between simulated and recorded data.

Conceptually, transforms are divided into three components: `field`, `amp`, and `phase`. Each component defines an ordered list of functions applied sequentially. Since the amplitude and phase are derived from the complex wavefield after applying `field` transforms, any such transforms will directly affect them.

A schematic pipeline for domain adaptation is shown in Fig. 8.

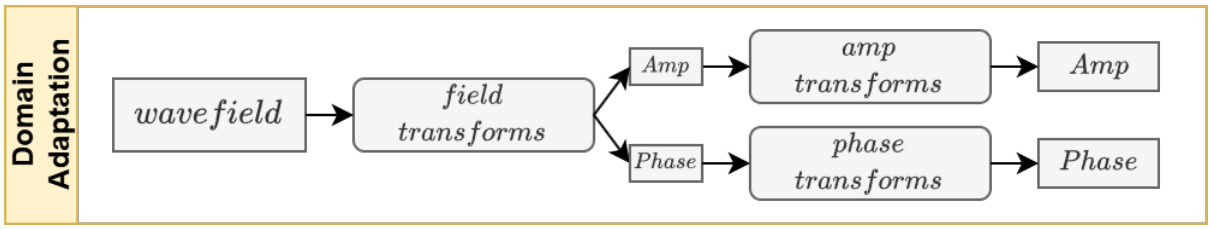

Figure 8: Domain adaptation pipeline showing transform application.

Example

```
{
    "transforms": {
        "field": {
            "crop": {"x_min":125, "y_min":125, "x_max":-125, "y_max":-125},
            "upscale_complex": {"target_shape":[500,500]}
        },
        "amp": {
            "sqrt": {}
        },
        "phase": {
            "subtract_mean": {}
        }
    }
}
```

Available Transform Functions

Each transform is defined by a `name` and a dictionary of parameters.

`crop`
: Crops a rectangle defined by pixel coordinates.

  `x_min`, `y_min` Starting pixel index.

  `x_max`, `y_max` Ending pixel index.

`sqrt`
: Computes the square root of the image values.

  - No additional parameters.

`upscale_complex`
: Bilinear upscale complex valued 2D tensor to a target size.

  `target_shape` Tuple specifying the new image size.

`subtract_mean`
: Subtracts the mean of the image.

  - No additional parameters.

`subtract_min`
: Shifts all pixel values to be positive by subtracting the minimum.

  - No additional parameters.

`subtract_background`
: Subtracts the mean of a small corner patch near the origin.

  - No additional parameters.

`gaussian_blur`
: Applies Gaussian blur.

  `sigma` Standard deviation for Gaussian kernel.

`gaussian_blur_scheduled`
: Applies Gaussian blur, decreasing over a number of steps.

  `max_sigma` Starting sigma value.

  `min_sigma` Ending sigma value.

  `num_steps` Number of steps to decrease sigma over.

`snr_noise`
: Adds noise based on the SNR.

  `snr_db` Noise level in decibels.

`noise`
: Adds Gaussian noise.

  `sigma` Standard deviation of the noise.