Inverse Simulation Tomography

Jannis.Maron@uni-siegen.de

October 7, 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.

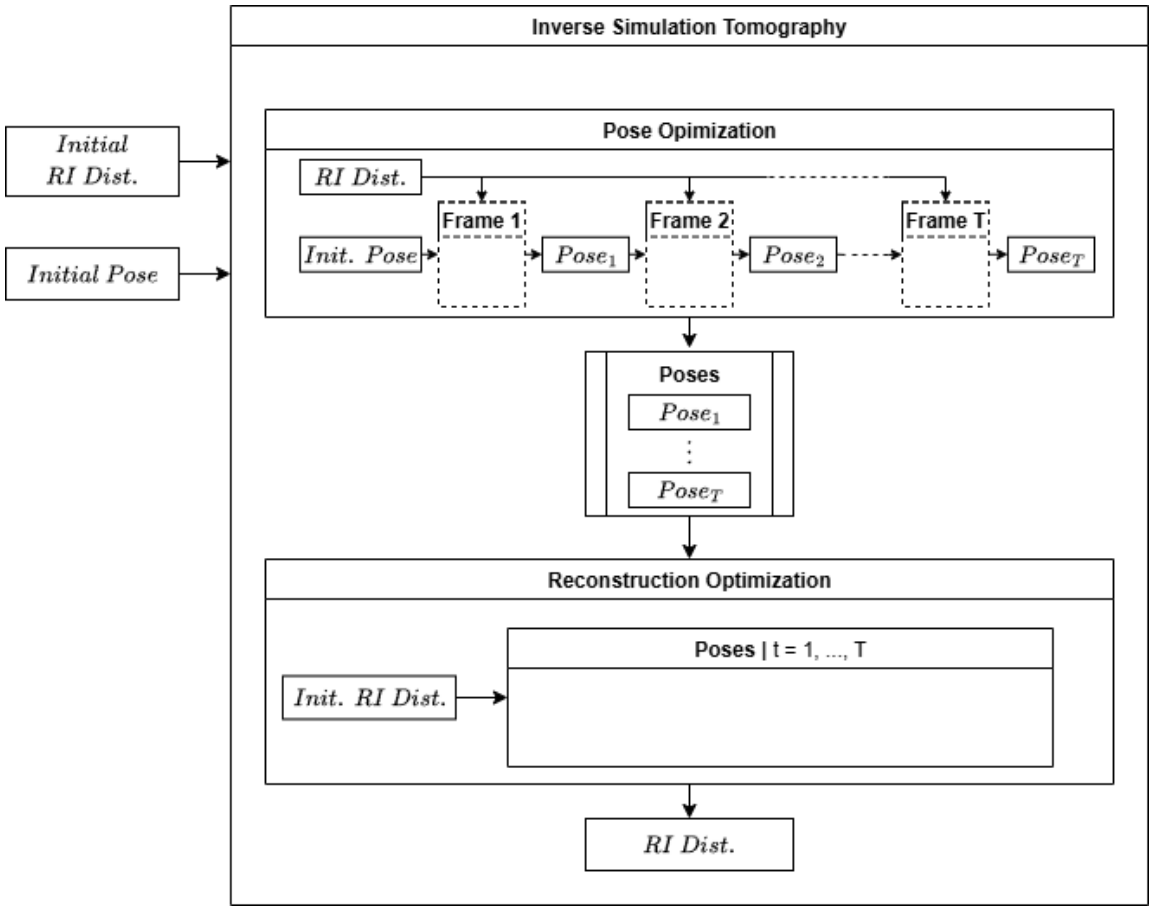

Figure 1: Top-level Overview

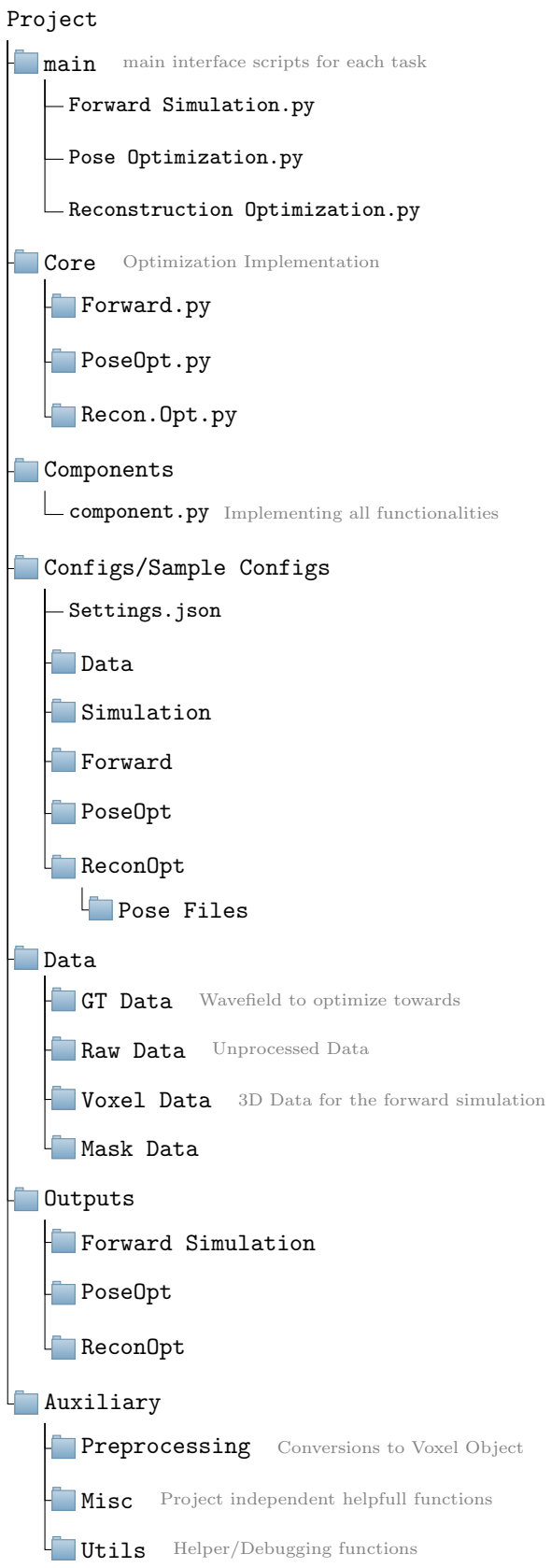

Figure 2: Project Hierarchy

Contents

1 Settings

Core to each task is a settings file. If you run any of the main scripts without specifying a settings file, you will be prompted to select one. Each settings file is a structured .json file and is required to run the code.
For readability and logical separation, we divide the settings file into multiple blocks: A block for common settings across all tasks and individual blocks for every task. Task-blocks, will be described in their corresponding sections [1].

Common Settings:

Example: *Settings.json* Common-Block.

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

}
```

- *config_dir*: Path w.r.t. to the executing code, leading to the Sub-Configs root directory.
- *device*: String indicating CPU or GPU computing.
  Options are: $\{null, "gpu", "cuda", "cpu"\}$. *null*, will prioritize GPU if available.
- *dtype*: Datatype precision.
  Options are: $\{float32, float64\}$
  Warning: Check Memory consumption.
- *phase_unwrap*: Flag swapping between wrapped and unwrapped phase visualization.
  Only affects visualizations.

[1] Except for reused sub-settings, which will have their own section

2 Sub-configs

For the sake of readability and reusability, we create sub-configs, with settings relating to different sub-tasks as individual sub-settings JSON files. Each sub-config is then referenced in the task-block of our main settings file. Paths towards each subconfig have to be defined w.r.t to the *config_dir* in the common settings.
Both the *Simulation Config* and the *Data Config* are reused in all three task, and therefore described here to avoid repetition.

2.1 Simulation Config.json

The *Simulation Config* is defines the simulation space and the wavefield propagation using the BPM.

Example: '*Simulation Config.json*':

```
{
    "Base_Grid": {
        "unit": "um",
        "spatial_resolution": [0.1, 0.1, 0.1],
        "grid_shape": [500, 500, 500],
        "n_background": 1.334,
        "wavelength": 0.640,
        "backpropagation_distance": 26
    }
}
```

- *unit*: Unit used in the simulation space.
  Note: Every other unit will be converted to this unit
  Options: {"m","mm","um"}.
- *spatial_resolution*: Physical voxel size of the simulation space (in *unit*).
  Given by $[d_x, d_y, d_z]$
- *grid_shape*: Number of voxels in the simulation space.
  Given by $[n_x, n_y, n_z]$.
- *n_background*: Base background refractive index in the simulation space.
- *wavelength*: Wavelength of the propagating wavefield (in *unit*).
- *backpropagation_distance*: Distance (in *unit*) to propagate the wavefield backwards through simulation space after full BPM to the focus plane.

2.2 Data Config.json

The *Data Configs* stores the location of all external data. Paths are w.r.t. the executing code location.
Note that not all data paths are required for all tasks.

Example: '*Data Config.json*':

```
{
    "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"
    }
}
```

- *voxel_object*: Path towards the 3D RI distribution.
  Required for: {*Forward Simulation, Pose Optimization, Reconstruction Optimization*}
- *ground_truth*: Path towards ground truth csv file, containing ordered ground truth frame locations.
  Required for: {*Pose Optimization, Reconstruction Optimization*}
- *mask*: Path towards 3D Mask file, for masking out voxel object gradients
  Required for: {}, **Optional for:** {*Reconstruction Optimization*}

3 Forward Simulation

Usage

Given a set of key-frame poses, place the estimated RI Distribution (*voxel object*), with the interpolated pose, into the simulation space, and perform the BPM to obtain a scattered wavefield.
A schematic of that process can be found in Fig. 3

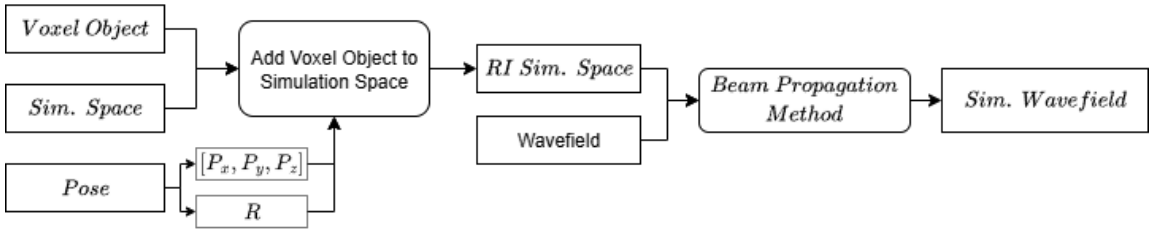

Figure 3: Forward Simulation schematic

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": "Sample Configs/Simulation/dhm_simulation.json",
        "data_config_file": "Sample Configs/Data/DHM_data.json",
        "forward_config_file": "Sample Configs/Forward/sample_movement.json",

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

```
        }
    },

}
```

- *simulation_config_file* : Path towards the config file defining the general simulation.
- *data_config_file*: Path towards the config file containing voxel object.
- *forward_config_file*: Path towards the config file defining movements over multiple keyframes, as well as post-processing operations.
- *output*: see 3.2 for more details.
  - *output_dir*: Path towards the desired outputs. (Created automatically)
  - *run_name*: Sub-folder for the specific run. (Created automatically)
  - *options*: Visualization options.
    Options: {"file", "phase", "amplitude", "slice", "sim_space"}

3.1.2 Forward Config.json

We describe the movement of the voxel object as poses at specific key frames. We then perform the simulation for every timestep by interpolating between key frames.
If only a single keyframe is provided, it is assumed to remain constant throughout the entire simulation. If the final keyframe occurs before the simulation's last timestep, the pose at that keyframe will persist until the end.
In addition to all this, the *Forward Config* also defines post-processing operations that are performed individually on the wavefield, amplitude image, and phase image. Details on the different post-processing in Sec, 6.1

Example: '*Movement Config.json*' for a full rotation around the x-axis, while the Voxel Object moves up and down along the y-axis. Rotation and placement are w.r.t. the object center. The simulation plays out over 37 timesteps total:

```
{
    "transforms":{
        "field": {
            "crop": {"x_min":125, "y_min":125, "x_max":-125, "y_max":-125},
            "upscale": {"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, 0.0, 0.0], "theta": 0, "time": 0 },
            { "axis": [1.0, 0.0, 0.0], "theta": 5, "time": 1 },
            { "axis": [1.0, 1.0, 0.0], "theta": 45, "time": 4 }
        ]
    }
}
```

Post-proceccing transforms

- *transforms* See 6.1 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*: The number of total timesteps.
- *unit*: Unit of position and offset.
  Options: {"m","mm","um"}.
- *Positions*: Physical positions of the center of the voxel object at specific keyframes.
  [{"pos" : $[p_x, p_y, p_z]$, "time" : t_i}, . . .]
- *Offset*: Translation vector shifting the object center.
  Position and rotation are w.r.t. the object center.
  [{"offset" : $[o_x, o_y, o_z]$, "time" : t_i}, . . .]
- *Rotation*: Rotation/Orientation of the voxel object w.r.t. its center
  [{"axis" : $[a_x, a_y, a_z]$, "theta" : θ_i, "time, . . .]" : t_i}

3.2 Outputs

Depending on the *options* set in the settings file, we create various outputs for every timestep. Outputs can be found in the specified *output_dir/run_name* directory, which will be created automatically if it does not already exist. Within this directory will be three subfolders: {*Data, Images, Videos*}

3.2.1 File

Should *options* include the *"file"* options we create an .pt output file for every timestep containing the following information:

- *wavefield*: Complex scattered wavefield after propagation. Includes only field post-processing.
- *amp*: Amplitude component of the scattered wavefield. Also includes amplitude post-processing in addition to field post-processing.
- *phase*: Phase component of the scattered wavefield. Also includes phase post-processing in addition to field post-processing.
- *pose_unit*: Unit of the position and offset (e.g. "um").
- *position*: Position of the voxel object in the simulation space at the current timestep.
- *offset*: Offset of the voxel object center from its geometric center at the current timestep.
- *axis*: Rotation axis of the voxel object at the current timestep.
- *angle*: Rotation angle of the voxel object in degrees at the current timestep.
- *transforms*: Description of the post-processing transforms.
- *sim_unit*: Unit of the simulation space (e.g. "um").
- *grid_shape*: Number of voxels in the simulation space.
- *spatial_resolution* Spatial resolution of the simulation space grid in sim_unit.
- *wavelength* Wavelength of the incident wave in sim_unit.

3.2.2 Images

If any of the *options* {"phase", "amplitude", "slice", "sim_space"} are set, an image for the corresponding option will be saved to the Images subfolder.

- slice: Slice of the xy-plane of the simulation space at a specific index.
- render 3D Rendering of the Simulation Space.
- phase Phase of the output wavefield: `img = unwrap_phase( np.angle( field ))`
- amplitude Amplitude of the output wavefield: `img = np.abs( field )`

3.2.3 Videos

If any of the *options* {"phase", "amplitude", "slice", "sim_space"} are set, an video for the corresponding option across all timesteps will be 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 video sequence of a recorded scattered wavefield, we attempt to reconstruct the pose of every frame in the sequence by optimizing the pose in our *Forwards Simulation* with our voxel object to best match the recorded scattered wavefield.
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 top-level schematic of the frame-by-frame approach can be found in Fig. 4, while the per-frame schematic is visualized in Fig. 5

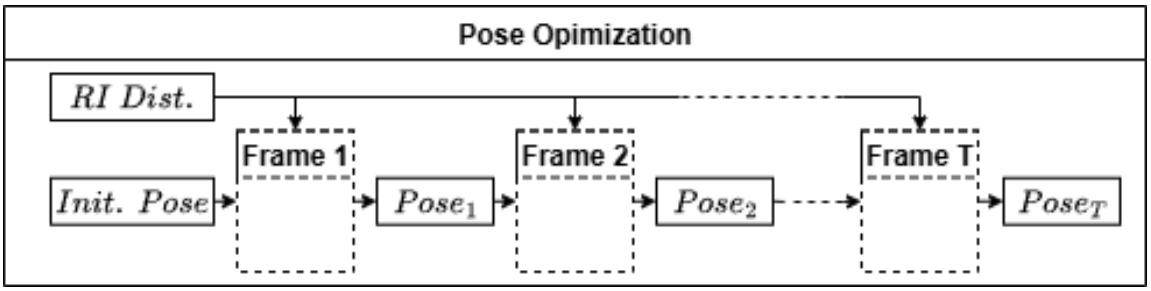

Figure 4: Pose Optimization details

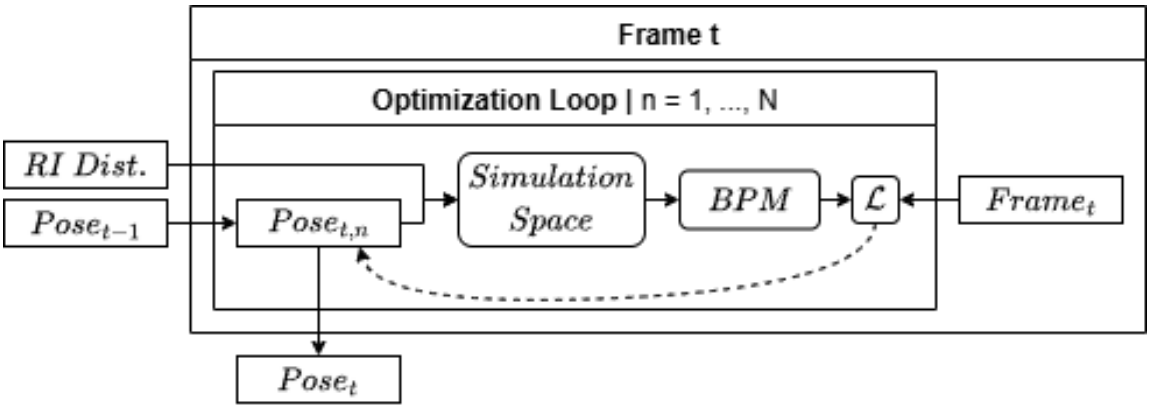

Figure 5: Pose Optimization details

Script

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

Required Data

- *Settings.json*
  - *Simulation Config.json*
  - *Data Config.json*
  - *PoseOpt Config.json*
- *Voxel Object.pt* (estimated RI distribution)
- Target data (recorded real data)

4.1 Configs

4.1.1 General Settings

Example: '*Setting.json*' for the Pose Optimization:

```
{
    "PoseOpt":{

        "simulation_config_file": "Sample Configs/Simulation/coded_wfs_simulation.json",
        "data_config_file": "Sample Configs/Data/Coded_WFS_data.json",
        "pose_opt_config_file": "Sample Configs/PoseOpt/coded_wfs_pose_opt.json",

        "output": {
            "active": true,
            "output_dir": "../Outputs/PoseOpt/HEK Cells/Coded WFS/",
            "run_name": "Example",
            "options": ["losses", "amps", "phases", "slices", "renders"],
            "wandb": false
        }
    }
}
```

- *simulation_config_file* : Path towards the config file defining the general simulation.
- *data_config_file*: Path towards the config file containing voxel object and target data locations.
- *pose_opt_config_file*: Path towards the config file defining training parameters.
- *output*: see 4.2 for more details.
  - *active*: Turn logging on/off
  - *output_dir*: Path towards the desired outputs. (Created automatically)
  - *run_name*: Sub-folder for the specific run. (Created automatically)
  - *options*: Visualization options.
    Options: {"losses", "amps", "phases", "slices", "renders"}

4.1.2 PoseOpt.json

A config file that specifies the optimization progress of our simulation towards the sequential 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* frame section, and a *Sequential* frames section.
Example: '*PoseOpt.json*':

```
{
    "PoseOpt": {
        "loss_fn": "mse_ncc_loss",
        "weights": [20, 35, 0.5, 100],

        "gt_transforms": {
            "field": {},
            "amp": {
                "sqrt":{}
            },
            "phase": {
                "gaussian_blur": {"sigma": 10.5},
                "subtract_mean": {}
                }
        },

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

        "start_frame": 0,
        "end_frame": 5,
        "frame_steps": 1,

        "unit": "um",
        "Position": [25, 25, 15],
        "Axis": [1, 0, 0],
        "Angle": -15,


        "Initial":{
            "epochs": 201,

            "optimizer": {
                "params":{
                    "Position": {"active": true, "lr": 5e-1},
                    "Axis": {"active": true, "lr": 1e-0},
                    "Angle": {"active": true, "lr": 1e-0}
                },
                "scheduler": {
                    "MultiStepLR": {
                        "milestones": [150,175, 190],
```

```
                    "gamma": 0.5
                }
            }
        },

        "regularizers" : {
        }
    },

    "Sequential": {
        "epochs": 21,

        "optimizer": {
            "params":{
                "Position": {"active": true, "lr": 1e-1},
                "Axis": {"active": true, "lr": 1e-1},
                "Angle": {"active": true, "lr": 5e-1}
            },
            "scheduler": {
                "MultiStepLR": {
                    "milestones": [5, 10, 15],
                    "gamma": 0.5
                }
            }
        },

        "regularizers" : {
            "pos_reg": {"type": "L2", "var": "pos", "lambda": 0.5},
            "axis_reg": {"type": "L2", "var": "axis", "lambda": 2, "memory": 5,
                "weights":"linear"},
            "angle_reg": {"type": "L2", "var": "angle", "lambda": 0.025}
        }
    }
}
```

- **PoseOpt**
  - *loss_fn*: Name of pre-implemented loss function.
    Options: {"mse_ncc_loss"}
  - *weights*: Individual weights for each component in the loss function.
  - *gt_transforms*: Domain Adaptation transforms applied to the target data. See 6.1 for more details.
  - *sim_transforms*: Domain Adaptation transforms applied to the simulated wavefield. See 6.1 for more details.
  - *start_frame*: Index of the first frame in our optimization sequence.
  - *end_frame*: Index of the last frame in our optimization sequence. Allows dataset wrapping.
  - *frame_steps*: Index steps from frame to frame. Allows skipping frames.
  - *unit*: Unit for position.
  - *Position*: Initial Position for the first frame.

- *Axis*: Initial Axis for the first frame.
- *Angle*: Initial Angle for the first frame.

- **Initial**
  - *"epochs"*: Number of epochs for the first frame.
  - *"optimizer"*: Adam optimizer for the first frame. See 6.2 for more details.
  - *"regularizers"*: Regularization scheme for the first frame. See 6.3 for more details.
- **Sequential**
  - *"optimizer"*: Adam optimizer for the sequential frames. See 6.2 for more details.
  - *"regularizers"*: Regularization scheme for the sequential frames. See 6.3 for more details.

4.2 Outputs

Depending on the *options* set in the settings file, we create various outputs for every frame in our pose optimization as well as a summary of the whole sequence. Outputs can be found in the specified output dir/run name directory, which will be created automatically if it does not already exist. Within this directory will be three subfolders: Summary, Configs, Frames

4.2.1 Summary

We will save a summary of the whole sequence over all frames. If any of the *options* {*"amps", "phases", "slices", "renders"*} are set, we will create video sequences for the corresponding option of the best found poses from every frame.

- best_setting.json: Record of the best pose and the corresponding loss for every frame.
- best_epoch.png: Plot showing the best iteration index for each frame.
- best_positions.png: Plot showing the optimized *(x,y,z)* positions over all frames.
  Visualized as: $(x := red, y := green, z := blue)$
- best_axes.png: Plot showing the optimized *(x,y,z)* rotation axes over all frames.
  Visualized as: $(x := red, y := green, z := blue)$
- Best Amplitude.avi: Video comparing the amplitude of the simulated wavefield to the recorded ground truth counterpart.
  Includes domain adaptations on both simulated and recorded data.
- Best Phase.avi: Video comparing the phase of the simulated wavefield to the recorded ground truth counterpart.
  Includes domain adaptations on both simulated and recorded data.
- Best Slice.avi: Video of a center slice of the simulation space for each optimized pose.
- Best Render.avi: Video of a 3D render of the simulation space for each optimized pose.

4.2.2 Configs

Saves the execution setting configs for easy detail look-up.

- *Settings.json*
- *Simulation.json*
- *Data.json*
- *PoseOpt.json*

4.2.3 Frames

Saves optimization progress for every frame individually.
If any of the *options* {*"amps", "phases"*} are set, we will periodically create visualizations for the corresponding option. If any of the *option* {"amps", "phases", "slices", "renders"} are set, we will create a visualization of the corresponding option with the best pose found during training. Additionally, we create plots for the loss if the *"losses"* option is set.

- progress.json: Records the loss and pose for each training iteration.
- loss Plots for total loss, data loss, and regularization loss over all training iterations
- amp Sub-directory containing plots comparing the amplitude of the simulated to the recorded wavefield for every *n-th* iteration.
  Includes domain adaptations on both simulated and recorded data.
- phase Sub-directory containing plots comparing the phase of the simulated to the recorded wavefield for every *n-th* iteration.
  Includes domain adaptations on both simulated and recorded data.
- Best Amplitude.png: Comparison plot between the amplitude of the simulated wavefield with the best found pose and the recorded wavefield.
  Includes domain adaptations on both simulated and recorded data.
- Best Phase.png: Comparison plot between the phase of the simulated wavefield with the best found pose and the recorded wavefield.
  Includes domain adaptations on both simulated and recorded data.
- Best Slice.png: Image of a center slice of the simulation space for the best found pose.
- Best Render.png: Image of a 3D render of the simulation space for the best found pose.

5 Reconstruction Optimization

Usage

Given a video sequence of a recorded scattered wavefield and a list of optimized poses, we update the estimated RI distribution of our voxel object such that the produced wavefield of our Forward Simulation better matches that of the recorded frames.
Oftentimes, the estimated voxel object contains background noise, which we do not want to optimize. To focus only on the actual object, we allow a gradient mask that resets computed gradients before updating the object.
We only update our voxel object once we have seen all recorded frames in our dataset. A top-level schematic of the frame-by-frame approach can be found in Fig. 6

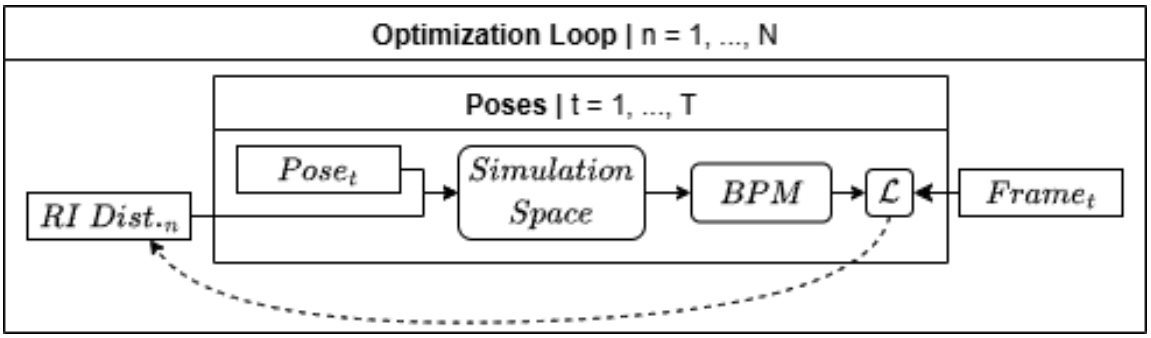

Figure 6: Reconstruction Details

Script

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

Required Data

- *Settings.json*
  - *Simulation Config.json*
  - *Data Config.json*
  - *ReconOpt Config.json*
- *Voxel Object.pt* (estimated RI distribution)
- *Optimized Poses.json* (i.e. from PoseOpt)
- Target data (recorded real data)

Optional

- *Gradient Mask.pt*

5.1 Configs

5.1.1 General Settings

Example: '*Setting.json*' for the Reconstruction Optimization:

```
{
    "ReconOpt":{
        "simulation_config_file": "Sample Configs/Simulation/coded_wfs_simulation.json",
        "data_config_file": "Sample Configs/Data/Coded_WFS_data.json",
        "recon_opt_config_file": "Sample Configs/ReconOpt/coded_wfs_pose_opt.json",
        "recon_pose_file": "Sample Configs/ReconOpt/coded_wfs_pose_opt.json",

        "output": {
            "active": true,
            "output_dir": "../Outputs/PoseOpt/HEK Cells/Coded WFS/",
            "run_name": "Example",
            "options": ["amps", "phases", "slices", "renders"],
            "wandb": false
        }
    }
}
```

- *simulation_config_file* : Path towards the config file defining the general simulation.
- *data_config_file*: Path towards the config file containing voxel object and target data locations.
- *recon_opt_config_file*: Path towards the config file defining training parameters.
- *recon_pose_file*: Path towards json file containing the optimized pose for each frame.
- *output*: see 5.2 for more details.
  - *active*: Turn logging on/off
  - *output_dir*: Path towards the desired outputs. (Created automatically)
  - *run_name*: Sub-folder for the specific run. (Created automatically)
  - *options*: Visualization options.
    Options: {"amps", "phases", "slices", "renders"}

5.1.2 ReconOpt.json

Example: '*ReconOpt.json*':

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

```
        },
        "sim_transforms": {
            "field": {
                "crop": {"x_min":125, "y_min":125, "x_max":-125, "y_max":-125}
            },
            "amp": {},
            "phase": {
                "subtract_mean": {}
            }
        },

        "epochs": 50,
        "optimizer": {
            "params":{
                "Voxel Object": {"active": true, "lr": 1e-3}
            },
            "scheduler": {
                "MultiStepLR": {
                    "milestones": [40,45],
                    "gamma": 0.5
                }
            }
        },

        "loss_fn": "mse_ncc_loss",
        "weights": [100, 50, 0.5, 150],

        "regularizers" : {
            "TV_reg": {"type": "TV", "lambda": 5e-5}
        }

    }
}
```

- *gt_transforms*: Domain Adaptation transforms applied to the target data. See 6.1 for more details.
- *sim_transforms*: Domain Adaptation transforms applied to the simulated wavefield. See 6.1 for more details.
- *"epochs"*: Number of training iterations. Each iteration is one full pass through the dataset
- *"optimizer"*: Adam optimizer to optimize the voxel object. See 6.2 for more details.
- *loss_fn*: Name of pre-implemented loss function.
  Options: {"mse_ncc_loss"}
- *weights*: Individual weights for each component in the loss function.
- *"regularizers"*: Regularization scheme for the first frame. See 6.3 for more details.

5.2 Outputs

Depending on the *options* set in the settings file, we create various outputs for every pass of the full dataset (i.e. whenever we update the voxel object). Outputs can be found in the specified output

dir/run name directory, which will be created automatically if it does not already exist. Within this directory will be three subfolders: Summary, Configs, Epochs

5.2.1 Summary

At the end of the optimization, we will visualize the progress and save the results in this subdirectory. If any of the *options* {*"amps", "phases"*} are set, we will run an additional time through the dataset and create for every pose comparison plots and videos for the corresponding option. If any of the *options* {*"slices", "renders"*} are set, we will visualize the best found reconstruction with the corresponding option as well as create a video across all training iterations.

- voxel_object.pt: Optimized Voxel object.
- losses: Plots of average total, data, and regularization loss through all training iterations. Normalized with respect to the number of frames/poses
- Amps: Comparison plots between the amplitude of the simulated wavefield, using the best reconstructed voxel object and the ground truth data, for every optimized pose.
  Does not include any domain adaptation.
- Phases: Comparison plots between the phase of the simulated wavefield, using the best reconstructed voxel object and the ground truth data, for every optimized pose.
  Does not include any domain adaptation.
- Amplitude.avi Video of Amps
- Phase.avi Video of Phases
- Slice_xxx.png Center Slice of the best reconstructed voxel object.
  xxx indicates the epoch at which the best object was found.
- Render_xxx.png 3D render of the best reconstructed voxel object.
  xxx indicates the epoch at which the best object was found.
- 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

Saves the execution setting configs for easy detail look-up.

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

5.2.3 Epochs

We visualize an update for every training iteration.
If any of the *options* {*"slices", "renders"*} are set, we will visualize the current voxel object, with the corresponding option. If any of the *options* {*"amps", "renders"*} are set, we will visualize the last pose of our dataset with the corresponding option.

6 Modular Components

As part of our optimization, we define several components, which can have a wide array of possible varying inputs. To keep the optimization sections clean, we will separate these components into their own sections.

6.1 Transforms

Transforms serve two main functions. The first is a post-processing of our simulation outputs. And the second is as domain adaptation during training, to increase the similarity between simulated and recorded data.
Conceptually, transforms are divided into three parts: {*field, amp, phase*}. Each of these defines a list of functions that will be applied to the related component. The order of transforms will be the same as they are ordered in each list. Because the phase and amplitude components are obtained from the complex wavefield after applying field transforms, any such transforms will have a direct effect on them. A schematic pipeline for applying domain adaptation can be found in Fig. 7.

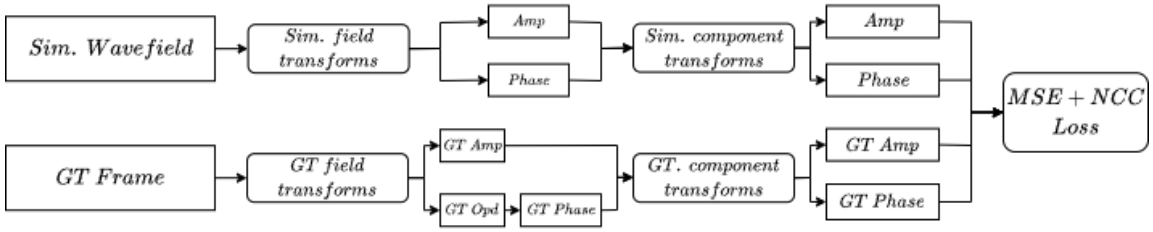

Figure 7: Domain Adaptation

Example: '*transforms*':

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

Options

Each transform function is defined by a *name* and a dict of variables.

Add Noise Adds noise based on the SNR

- *"noise"*:
  - *snr_db*: Decibel of noise amount

Cropping Crops (x_min,y_min)/(x_max,y_max) rectangle from the image

- *"crop"*:
  - *x_min*: pixel index
  - *y_min*: pixel index
  - *x_max*: pixel index
  - *y_max*: pixel index

Upscale Bilinear upscale image to target size

- *"upscale"*:
  - *target_shape*: tuple of new size

Gaussian Blur Add Gaussian blur with set sigma

- *gaussian_blur*
  - sigma: float

Zero Mean Subtracts mean from image

- *subtract_mean*
  - -

Shift to positive Subtracts the minimum value from the image

- *subtract_min*
  - -

Remove background Subtracts the mean value of a small patch of the corner near the origin

- *subtract_background*
  - -

Square root Takes the square root of the image values

- *sqrt*
  - -

6.2 Optimizer

Our modular optimizer consists of two components: i) the optimizer to update parameters and ii) a scheduler for controlling parameters over time.

Example: '*optimizer*':

```
{
    "optimizer": {
        "params":{
            "Position": {"active": true, "lr": 5e-1},
            "Axis": {"active": true, "lr": 1e-0},
            "Angle": {"active": true, "lr": 1e-0}
        },
        "scheduler": {
            "MultiStepLR": {
                "milestones": [150,175, 190],
                "gamma": 0.5
            }
        }
    },
}
```

Parameters

Each parameter we will optimize consists of three components:

- "name": A name that links the string in the settings to the parameter (see below)
- *active*: toggle learnable parameter on/off
- *lr*: learning rate for that specific parameters

A combination of names the the corresponding parameters are:

- "Position": Position
- "Axis": Rotation axis
- "Angle": Rotation Angle
- "Voxel Object": 3D RI distribution

Scheduler

Currently, we only support a multi-step learning rate scheduler.

- *MultiStepLR*
  - *milestones*: iterations at which we decrease the learning rate
  - *gamma*: Factor by how much we decrease the learning rate.

6.3 Regularizer

The modular construction of our regularizer allows us to define a combination of regularization schemes for different variables completely independently, without modifying the code.
Each regularizer has a name (for display only), some mandatory variables, and possible optional variables. We currently differentiate between the PoseOpt and ReconOpt regularizers.

6.3.1 PoseOpt Regularizer

Example: '*poseopt regularizer*':

```
{
    "regularizers" : {
        "pos_reg": {"type": "L2", "var": "pos", "lambda": 0.5},
        "axis_reg": {"type": "L2", "var": "axis", "lambda": 2, "memory": 5, "weights":"linear
            "},
        "angle_reg": {"type": "L2", "var": "angle", "lambda": 0.0075}
    }

}
```

- *type*: String defining what kind of regularization method will be used.
  Current options are:
  - "L2": L2 loss between input an all targets
- *var*: String linking the parameter to be regularized.
  Current options are:
  - "pos": Position
  - "axis": Axis
  - "angle": Angle
- *lambda*: Weighting of this regularizer

Optional parameters

- *fixed*: Flag for static (unchanging) targets or dynamically updating targets.
  Default: *false* (dynamic updating)
- *target*: Fixed Target to regularize towards. Only works if *fixed=true*.
  Default: null
- *memory*: Number of targets we can regularize towards.
  Default: 1
- *weights*: Importance scaling for individual targets.
  Default: "equal"
  Options are:
  - "equal": Each target has equal importance
  - "linear": Linear decreasing importance, the longer ago the target was acquired.
  - "exp": Exponential decreasing importance, the longer ago the target was acquired.

6.3.2 ReconOpt Regularizer

Currently, the only relevant parameter in reconstruction optimization is the voxel object.
Example: '*poseopt regularizer*':

```
{
    "regularizers" : {
            "TV_reg": {"type": "TV", "lambda": 5e-5}
        }
}
```

- *type*: String defining what kind of regularization method will be used.
  Current options are:
  - "TV": Total variation loss
- *lambda*: Weighting of this regularizer