NADIA Software Project
Computational Workshop

T’Mir D. Julius     tdjulius@unimelb.edu.au
28/09/2012
NADIA Workshop Overview

- What is it for?
  Aims and Motivation for The NADIA Package

- What is it?
  Package description

- What does it do?
  Current progress and future plans

- What you can do, too.
  Examples and instructions for use
Today’s Aim

Simulate and reconstruct a Planar Coherent image, and use this code to reconstruct an image from Planar Coherent data.

Repeat this for a Partially Spatial Coherent image if you have time.
Today You Will

- Make a forward simulation and obtain the intensity of a provided complex transmission function.
- Reconstruct the transmission function of the simulation.
- Use this code to reconstruct actual data.
Extension

- Make a forward simulation and obtain the intensity of a provided complex transmission function for a Partial Spatial beam.

- Reconstruct the transmission function of the simulation.

- Using the Planar Reconstruction code and the Partial simulation code, write your own Partial Reconstruction code.
Motivation

- Currently everyone everywhere is creating and writing their own code for image reconstruction every time they begin a new project.

- NADIA aims to be a standard piece of software that will provide a base to allow new users to begin a project quickly and easily.

- NADIA can also work as a cross check tool to allow researchers to compare their results with the NADIA output.

- To efficiently provide this, NADIA must be:
  - User friendly - well documented, easy to install and use, multi-platform
  - Robust - gives the right results and is well tested
  - Flexible - open source and easily adaptable to the users needs
  - Reasonably fast
The NADIA Package

Usage
- Can be used as a set of IDL routines
- Command line tools
- A C++ library imported in to your C code

Abilities
- Planar Coherent Diffractive Imaging (PCDI)
- Fresnel Coherent Diffractive Imaging (FCDI)
- Partial Spatial Coherent Diffractive Imaging
- Partial Temporal Coherent Diffractive Imaging

Features
- Change between reconstruction methods (HIO, ER, etc.)
- Incorporate special methods such as charge flipping
- Update the support using the shrink-wrap algorithm
- Begin using a random initialisation
- Save and reload exit surface-wave estimates.

28/09/2012
#include <stdlib.h>
#include <cstdlib>
#include "io.h"
#include "Complex_2D.h"
#include "Double_2D.h"

int main(void){
    Complex_2D object_estimate(nx,ny);
    PartialCDI partial(object_estimate, beta, lcx, lcy, psize_x, psize_y, e_beam, z_sd, 4, 0);
    partial.initialise_estimate(7);
    write_cplx("PCDI_trans.cplx", object_estimate);
    return 0;
}
How it Works - Simulation

Read in Data
- Double_2D data;
  read_image("data_file_name.tif", data);

Initialise the CDI Object
- Complex_2D object_estimate(nx,ny);
  PlanarCDI planar(object_estimate);
  planar.set_support(support, false);
  planar.set_intensity(data);

Propagate to Detector
- planar.propagate_to_detector(data);

Output the Result
- Double_2D object_mag;
  object_estimate.get_2d(MAG, object_mag);
  write_image("mag.tif", object_mag);

Read the example data in from hdf, tiff, dbin or a ppm.

Create a Complex_2D object, and a CDI object.

Propagate the example data to the detector

Output the estimate
How it Works - Reconstruction

- **Read in Data**
  - Read the data in from hdf, tiff, dbin or a ppm.

- **Initialise the CDI Object**
  - Create a Complex_2d object, and a CDI object.

- **Iterate**
  - Iterate by propagating object_estimate to the detector plane, scaling the object estimate, and propagating the estimate back and applying the support.

- **Output the Result**
  - Output the revised object estimate.

- **Read in Data**
  - Double_2D data;
    read_image("data_file_name.tiff", data);

- **Initialise the CDI Object**
  - Complex_2D object_estimate (nx,ny);
    PlanarCDI planar (object_estimate);
    planar.set_support(support, false);
    planar.set_intensity(data);

- **Iterate**
  - for(int i=0; i<n_iterations; i++){
    planar.iterate();
  }

- **Output the Result**
  - write_cplx(temp_str.str(), object_estimate);
    Double_2D object_mag;
    object_estimate.get_2d(MAG, object_mag);
    write_image("mag.tiff", object_mag);
Installing the Software

- Copy `~tjulius/NADIA_workshop` to your desktop
  
  ```
  cp ~tjulius/NADIA_workshop ~/Desktop
  ```

- Change directory to `~/Desktop/NADIA_workshop/`
  
  ```
  cd ~/Desktop/NADIA_workshop/
  ```

- This .pdf is in this directory

- Open `NADIAws.pdf` using `evince`
  
  `evince NADIAws.pdf`

- Change directory to `NADIA` directory
  
  ```
  cd NADIA
  ```

- Run `./configure`
Installing the Software

- make
- cd examples
- ./PlanarCDI_simulation_example.exe

The NADIA software relies on some non standard libraries. Specifically it requires the LAPACK, fftw, libtiff and hdf libraries. These are installed here, but you will need to install them at home (apt, yum, Synaptic Package Manager, etc).
**Included Examples Today**

- PlanarCDI_example.c - Reconstruct a Planar CDI image from data

- PlanarCDI_simulation_example.c - Create Planar CDI simulated data

- PartialCDI_simulation_example.c - Create Partial Spatial CDI simulated data

  - Based on:
    Diffractive imaging using partially coherent X rays
    Whitehead, L W, Physical review letters, 2009, v 103, 24, 243902
Included Examples in the full software

- Planar Coherent Diffractive Imaging
  - Simulation
  - Reconstruction from real data

- Fresnel Coherent Diffractive Imaging (FCDI)
  - Simulation
  - Reconstruction from simulated data

- Partial Spatial Coherent Diffractive Imaging
  - Simulation
  - Reconstruction from real data

- Partial Temporal Coherent Diffractive Imaging
  - Simulation
  - Reconstruction from real data (coming soon)
Your task

- Use the PlanarCDI_simulation_example.c to create some simulated data
- Use PlanarCDI_example.c to reconstruct. Save estimate and reload to continue
- Modify PlanarCDI_example.c to change algorithms
- Include the shrinkwrap algorithm
A few Linux commands

- `mv file1 file2` - move file1 to file2 (this can also be used for renaming files)
- `cp file1 file2` - copy file1 to file2
- `mkdir directory` - make a directory
- `ls` - list the files in the current directory
- `cd directory` - change into another directory
- `rm file` - remove file (be careful because it won’t make a copy in the trash folder)
- `emacs` or `gvim` - text editors

You can find out more information about a command by typing: `man command` or `command --help`.
Comment out `initialise_estimate`

```
Complex_2D esw(nx, ny);
read_cplx("esw.cplx", esw);

/*Do some stuff*/

write_cplx("esw.cplx", esw);
```
Changing Algorithms

```c
planar.set_algorithm( algorithm );
```

- ER - error reduction
- BIO - basic input-output
- BOO - basic output-output
- HIO - hybrid input-output
- DM - difference map
- SF - solvent-flipping,
- ASR - averaged successive reflections
- HPR - hybrid projection reflection
- RAAR - relaxed averaged alternating reflectors

Or `planar.set_custom_algorithm(0.5, 0, -1, 0, -1, 0, 0, 0, 0, 0);`

Apply Shrinkwrap

planar.apply_shrinkwrap(gauss_width, threshold);

 gauss_width in pixels, threshold is the percentage of the maximum pixel.
Reconstruct from Real Data

- string data_file_name = "image_files/planar_data.tif";

- image_files/planar_data.tif is planar data collected by Lachlan Whitehead.
Extension

- Use the PartialCDI_simulation_example.c to create some simulated data

- Copy PlanarCDI_example.c to PartialCDI_example.c, and edit it to include the PartialCDI object. Use this to reconstruct the simulated data.

- Modify Makefile to compile change PartialCDI_example.c
Partial Simulation

- Coherence lengths x and y
  - $l_{cx} = 13.3 \times 10^{-6}$ m, $l_{cy} = 40.0 \times 10^{-3}$ m

- Pixel size detector in m
  - Pixel size $x = Pixel$ size $y = 13.5 \times 10^{-6}$ m

- Energy of the beam
  - 1400.0 eV

- Distance between detector and sample
  - 1.4 m

- Precision of Legendre Polynomials
  - 32

- Number of Modes in one dimension
  - 7, therefore 49 modes total.

- The number of polynomials must be greater than the number of modes.
Defining a CDI object

- PlanarCDI
  planar(object_estimate, n_best_guesses);

- PartialCDI my_partial(object_estimate, beta, lcx, lcy, psize_x, psize_y, e_beam, z_sd, n_best_guesses, 0);
More Information