# Create Poly Correction Tables

#### Research Work Notes

## Check better on single point convergence – 25/12/24

Add prints to special CSV file…

## Check C Tab Value – why a single point is not converging? – 24/12/24

The single point starts converging well – until it changes direction.

Should check more.

## New classes for geometric orientation and convergence – 23/12/24

C Tab Value - For a single point

C Tab Values – matrix of “C Tab Value”

First check convergence through a single “C Tab Value”

Later use array of values…

## Compute again IR and IIR for a small sub-region for image 163, tube 0 – 18/12/24

Define the relevant region by current IIR tables –

Rows 67 – 70 (4 rows)

Columns 1 – 170 🡪 170 – 345 (176 columns, detectors)

Run IR loop on 2 rows (67 and 70) and every 4 detectors starting from 170

#### RE-visit my IR loop

It is now working nicely on bands…

#### Problem due to strongly negative values

At some point – with a ring just around a black hole in the very center,

There are very strong negative values in the middle –

At r67 with d346 it is:

1. -8695 (min) in the middle
2. +6239 (max) at the corners of the small ring

Adding the “min” to all the volume creates artificially high values, where there is none.

This kills the “threshold” concept, and cause false radius identification:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| row | det | max | image | rad |
| 67 | 342 | 7658.18 | 159.967 | 1 |
| 67 | 346 | 14934.47 | 159.99 | 97.527 |

#### Problems in the very center – negative radius

The high values at the very center had radius “0” and hence were not counted…

As no high values were found – radius defaulted to “-1”

This special case is now corrected by adding offset of “1” to the radius value while identifying points above threshold – and subtracting this offset later from the average.

## New design for spaced-steps for setting multi-point derivations – 18/12/24

Every input point influences several output points –

The influence of every point is concentrated in some “neighborhood” around its top-influence

We can change a “grid” of input points so that the level of “crosstalk” would be limited.

When changing a grid – we can compute the gradient for the points that were changed…

Computation of the gradient would be by accumulating the influence on several output points in the relevant neighborhood.

#### Correcting the coordinates

For the last experiments I had: table [0,70,300] 🡨 🡪 Dev [163,30]

In the inverse table:

table [0,70,130] 🡨 🡪 Dev [163,30]

table [0,70,129] 🡨 🡪 Dev [163,31]

Apparently I have to add 170 to the detector index

## Try to flatten a whole image – 17/12/24

First must get the input-output geometric function for a whole image…

I used to work on table [0,70,300] 🡨 🡪 Dev [163,30]

#### Where is my data?

“D:\PolyCalib\ImpulseResponseTab/Tube0\_IR\_grid.csv”

“D:\PolyCalib\InverseIRTables”

Create “D:\PolyCalib\InverseIRTables\Image163\_sources.csv”

## Compute real derivation creates real convergence – 16/12/24

It seems to be working…

## Manual convergence on a single point – 16/12/24

Hold all the relevant information:

1. Table value
2. Deviation
3. Table value delta
4. Deviation delta
5. Real gradient is: deviation delta / table delta”
6. In general – higher table value should cause higher (signed) deviation value

Each time try to reduce a fraction (LR) of the variation – start with 0.1

To start:

1. Measure variation on a flat table
2. According to the direction (sign) of the deviation – try a small (constant) table step.
3. Loop training using the last two readings

## Back to single point – convergence is not consistent – 15/12/24

With LR at 0.1, sometimes it converges very well,

BUT:

1. Sometimes it reaches asymptotic convergence to high loss level
2. When gradient is negative, it still “correct” down and away from the target

#### Try another optimizer

I was using Adam.

Try single value input.

Try SGD – it does not seem to improve 😊

#### Try new approach – manual point-by-point convergence

I have all the information, at least for the “linear” case with a single volume.

Do manual computations per table point first.

It might be defined as home-made optimizer later.

## Single image train – 15/12/24

Use all radiuses for a single image.

What are their sources in the tables?

Start with a single table – than try 2 tables together.

## Single point train – continued – 15/12/24

1. Loss should return single value for single point
2. Gradient for a single point
3. Change only single point in table

Loss values are completely wrong. Check – through dump:

1. Tables
2. Images’ Volumes
3. Dev Maps
4. Loss

## Single point train by class CPolyDLTrainer0 – 10/12/24

1. First run on flat table –
   1. Save initial deviation map
   2. Identify target (for flatness)
2. **Second, try to change only one point**
3. Check difference
4. Identify hot point in dev diff
5. Save new dev map

Hottest point on diff is:

<data\_x> 30.106918 </data\_x>

<data\_y> 163.031235 </data\_y>

Fully consistent with computation

3) Define loss for a single point

## Make the training faster – 9/12/24

Do not need to repeat first and second recon each time…

May load ready target

#### Single point mode –

To make things simple – start with “impulse response” to some point…

Find the relevant point in the dev map

Give loss function only for this point – and optimize it!

#### Tube 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| row | det | max | im | rad |
| 70 | 300 | 2159.782 | 163 | 30 |

#### Tube 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| row | det | max | im | rad |
| 70 | 300 | 3805.391 | 67 | 30.218 |

## Working with Recon as external Function and Back Propagation – 9/12/24

C Ex Recon is working within training –

#### How to create meaningful gradients?

Start with a single point in the tables – and the corresponding single point in the Dev Map.

## Problems in integrating Recon as external function – 2/12/24

Work with C Ex Recon

It includes “forward” and “backward”

BUT – the backward format is not accepted!

## Try small input – 1/12/24

Try to use a small vector of ones as input!

## Start training on small matrices – 27/11/24

Questions:

1. How to select a small region where the dev map and poly table are both relevant?
2. What is the preferred artificial derivation for recon?

Thinking:

Start with central area:

1. Limited number of detectors
2. Limited number of rows
3. Limited number of images
4. Only small radiuses

In the central area I can easily use “Linear” with full connectivity…

Target is flat deviation map – flat value is taken from empty map

#### Data dimensions

Dev Map is 280 images \* 128 radiuses 🡺

## Now I know how to integrate external function in loss function – 27/11/24

Example is in “D:\SW\TrainDL - ExFunc example”

## Ideas for using DL to create poly tables – 26/11/24

Input can be one or several Deviation Maps.

The connections can be spatially oriented – for each location use only relevant input space.

#### Gradual approach

1. Do some training for fully connected
2. Use central area of a few images for small input and output matrices
3. Use BP results for loss function
4. …

## Create real DL prototype – 25/11/24

New Spyder project at “D:\SW\PolyDL”

INPUT for DL:

The Dev Map – 280\*260

NN:

All 2 All, few layers

Output – the fraction part of the 1st coefficient – valued between -0.01 and +0.01

Output size is 192 \* 668

Try training initially for flat table!

## After discussion – 22/11/24

1. Where are the edges of the water phantom container?
   1. They are well seen in FOV 450
2. Galit suggest to separate data by FOV – set new sets of data
   1. New data from Galit is in “h:\Poly Calibration by AI\From Galit 2”
3. I promise to try to create a prototype with real DL where recon is used only as LOSS

## Prepared presentation for Physics team – 21/11/24

Presentation is under GIT in:

D:\SW\PyIP\Docs

Result Driven Preparation of Poly Correction Tables - Initial Research Report.pptx

## Use IIR to correct poly tables – 16/11/24

Optimizing to target is working – but poorly!

1. The target itself is usually not improving - First check why?
2. Make more bold steps

#### Check how the target is missed…

The correction is not at the exact position to improve the target!

First make the errors apparent – check their direction!

Find location and direction of most change in the new [image, radius] raster

#### New flow: Several steps per target

First select a target to correct

Then take several steps until it is significantly improved (at least 40%)

For each target there are 4 spatial options: Each of the 2 tubes, Left & Right

One of the tubes may not be effective.

## Use IIR to correct poly tables – 2/11/24

Some ideas:

1. Separate the “abs diff from target” score from all other scores
   1. Try accepting steps only by “average abs diff”
   2. Make sure that the max point that was targeted really improved
2. Maybe try correcting both tubes – and see which one is better?
3. Select patches of different radius for deviations of different size
   1. Prepare several initial patches
4. Avoid traps of repeatedly trying to correct failed corrections

## Use IIR to correct poly tables – 2/11/24

Add global log to help understand how steps are selected and what is the result

## Use IIR to correct poly tables – 30/10/24

Set a single target-level and strive to bring all rings to this level.

Find point with biggest deviation from target – and select relevant correction.

Direction of correction (up or down) is evident from the selected deviation.

After correction & recon, consider both local and global deviation score.

Later this method will also help to decide width and amplitude of correction.

Avoid marginal spaces – at least initially.

#### Create new “score” – distance from flat target at ring

Dump Deviation [Image, Radius] as displayable matrix

Apparently, there are very strong deviations on the margins –

Try to cut out the margins! Peel function seems to work OK.

#### First verify that problems are correctly identified

Even after peel – result seems to be wrong!

self.avgDev=-1.0128281116485596, iImage=92, iRad=0, maxDev=tensor(-458.4808)

Second run even worse:

self.avgDev=-0.9682614803314209, iImage=92, iRad=0, maxDev=tensor(-999.0014)

The inner rings should be deterministic – as there are only 4 pixels!

Something is basically wrong with the average per radius computations!

## Check IR and IIR in a single automatic loop – 29/10/24

In Identify Ring Source.py

First activate new recon option:

Set config file “d:\Config\Poly\Impulse.txt”: Tube, row, detector

Set BP dump name at: “d:\Config\Poly\BPDumpFileName.txt”

To something like: “d:/PolyCalib/Impulse\Poli\_AI\_t1\_r70\_d300\_width256\_height256\_zoom2.float.rvol”

## Correcting table by Reverse-IR function – 23-27/10/24

Work with new “impulse response values” to check full loop of IIR

D:\PolyCalib\Impulse

Poli\_AI\_t1\_r70\_d300\_width256\_height256\_zoom2.float.rvol

Load it and analyze it – to find source of ring!