3D Imaging Using Structured Light Summary

Dr. Gao

Ethan Wilson, Beau Blackburn, Alejandro Matlalcuatzi

This is a tutorial process document outlining and showing the process starting installing anaconda then setting the environment up, to the current end of progress near between calibration and reprojection.

Environment Set up

Download the following programs

<https://en.ids-imaging.com/download-peak.html>

<https://www.anaconda.com/docs/getting-started/anaconda/install>

<https://www.meshlab.net/>

<https://code.visualstudio.com/download>

https://www.mathworks.com/downloads/

Create an environment in the anaconda prompt. Make sure to replace <env-name> with

your preferred environment name.

conda create -n <env-name>

Then run the commands under neath inside the anaconda enviroment

conda install python=3.11

conda install numpy

pip install opencv-contrib-python==4.9.0.80

pip install rawpy

pip install open3d

pip install ids-peak ids-peak-ipl

pip install pillow

cd "C:\Program Files\MATLAB\R2023a\extern\engines\python"

python setup.py install

We need to install openni2 separately. Download this file and once downloaded you can

install primesense.

<https://structure.io/openni/>

pip install primesense

Running the Code, Gray Code

This section is about running the code, this will be done from inside the terminal of VScode.



Have a similar setup as this, with the camera and projector side by side aiming at the board placed against a wall preferably.

Open the IDS peak app to ensure the board is entirly in frame

Ensure the focal lens is adjusted so the codes on the board are in fine detail. Then for the brightness lens (one closer to the back). Turn it all the way to the right then back in 25% of the full range, assuming looking into the lens from the front. Loosen then tighten the screws to do this.

Next inside the terminal ensure the structured-light-master folder is open, then run the command

python .\orchestrator.py

Begin by taking Calibration sets for the camera and projector to then be calibration together, at least 5 must be taken

Angle the board at different angles from the wall and take a new set each time

such as slightly base offset, slightly turned, and more

Enter ‘1’ to take a set, then ‘y’ for calibration set

Press enter for the code to test if all corners are visable

If not, reangle to board and try again

Then a new popup window opens, full screen it and press enter for the process to being

Once the window closes, angle the board differently and run the code again for the next set

After gaining at least 5 sets, can be more, place the board near flush against the wall and then put the object to be imaged infront of the Charuco board and take an object set with running orchestrator.py ‘1’ then ‘n’ then enter the objects name to name the storage folder of the images captured.

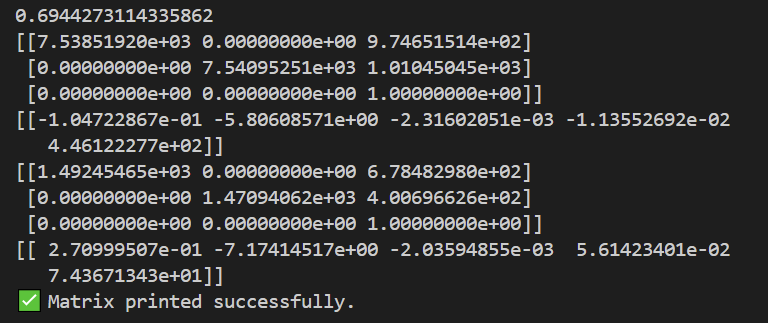
NOTE: after gaining the completion of a object set go inside the object folder, make a folder named “c\_0” and place all images inside from that set

Once the previous steps are complete the decode process can run with option 2 inside of the orchestrator file

This will take a large amount of time so be patient, errors can be shown in terminal, but the code is still running if no new line can be typed.

Then run the calibration code with option 3

Next confirm the data is good by running option 5, a matrix will be shown as similar to the one below

 the important parts are the first number printed and the first matrix

Ensure the first number is under 1, ideally as close to 0 as possible

then that M11 and M21 are the same

and M33 is 1

The next and final step is run the reprojection code with option 5

Go to structured-light-master > ReprojectImage > main.py, then on line 9 change the object name to be what you want to reproject

After running open the object folder and next to c\_0 folder will be 3 .ply files if run correctly. The one to view is the filtered one, open meshlabs then open that file to view results

Phase Shift Code and Math

Code

The automation inside the gray code using the orchestrator file does not exist in this file group near as much. The 5 files used, mainly that will be run, are

IDScapturecode, assuming new IDS camera being used

PhaseDecode.py

ObjectPhaseDecode.py

UnwrapPhase.py

PhaseCalibration.py

PhaseReproject.py

Runnign the IDS capture code will lead to structured-light-master>captures>c\_# folder creation with the images inside, depending on number of fringes (at least 3 makes 8+ images inside each calibration and object set)

Then once all sets are acquired run the PhaseDecode.py, pulls from master>captures folder automatically and leads to the creation of tiff files named

Out\_invalidH, out\_invalidV, wrapped\_phase\_x, wrapped\_phase\_y, x\_index, and y\_index

If there is an object set to decode run the ObjectPhaseDecode.py to gain those tiff files

MATH

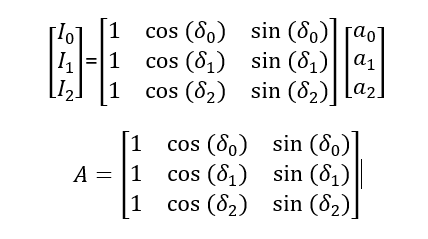
Decode

The images that are the input to this process (PhaseDecode.py) are b, w, h1-h2, and v1-v2 so there are eight images. Since there are 3 phase images, the images are a grey scale of 120\* -120\* and 0\*.

The light intensity from each image can be modeled in the formula in eq (1) below

(Eq1) 

Which then building off the three images from the three phases leads to the matrix and formula below



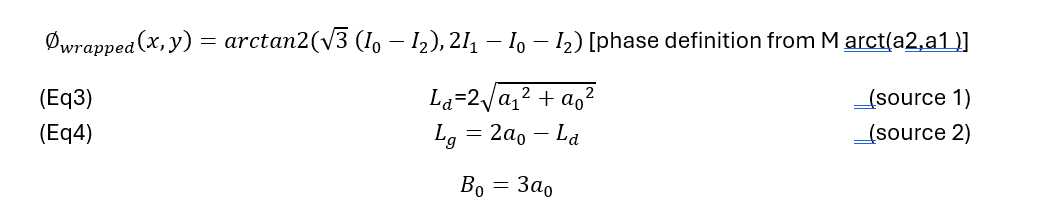
A is the design matrix of a sinusoid model

is the average intensity (DC term/offset)

proportional to the cosine of the underlying term

proportional to the sine of the underlying term

Once previously mentioned values are found, the wrapped (x, y) value can be calced by

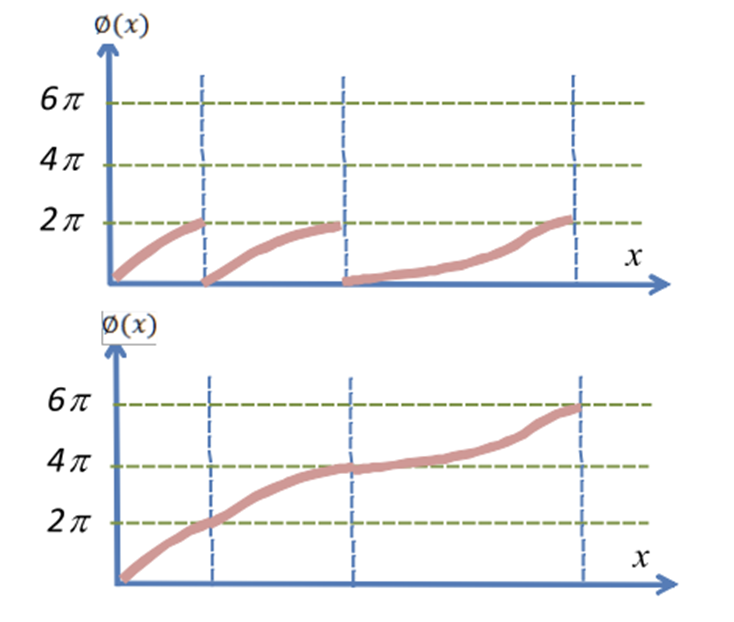


Those values then are able to lead to a filtered tiff file image of pixels by employing the formula



Unwrap

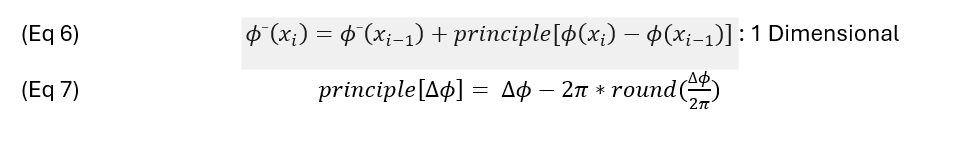
The unwrapping then must take place, imaged below is a visual of the general idea of what must occur



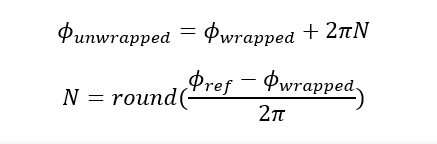
Picture from: Structured-light 3D surface imaging: a tutorial Jason Geng

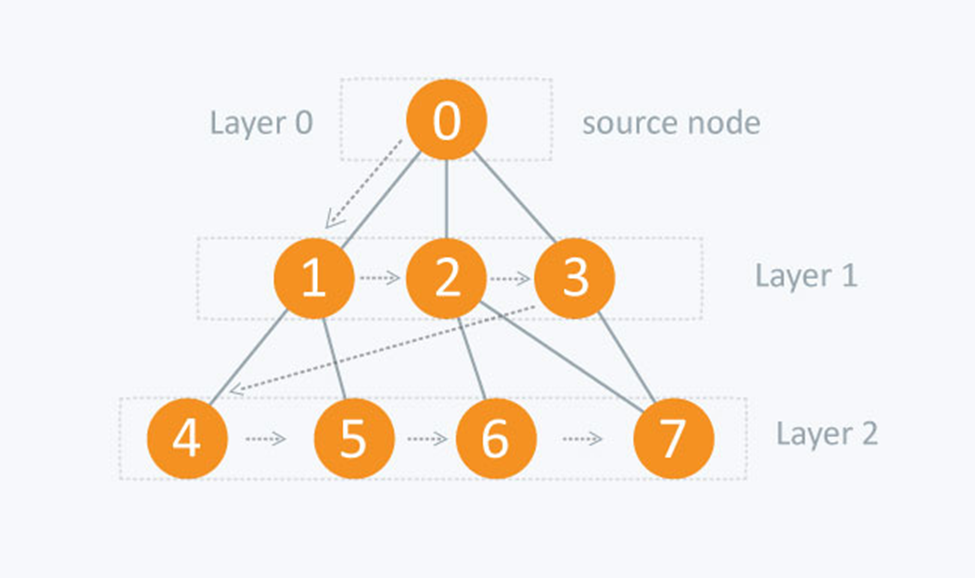
Taking Itohs method, which is below, the formula can then be extended to a 2D plane

(Eq 6 and 7) from (source 3)



Incorporating a queue driven Breadth First Search (BFS) algorithm to the 2D plane a new equation set acquired for the 2D image plane is [from source 4 and adapted to our setup]

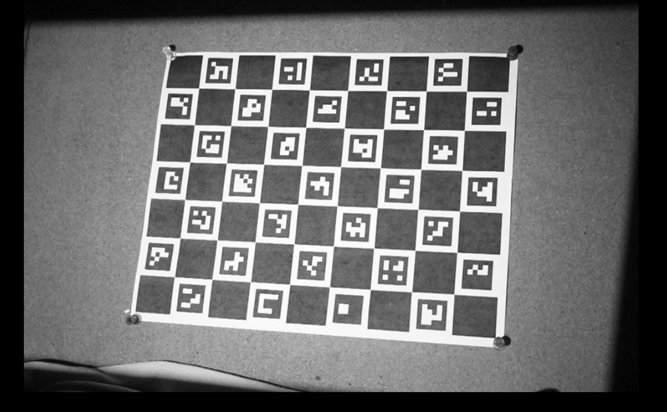


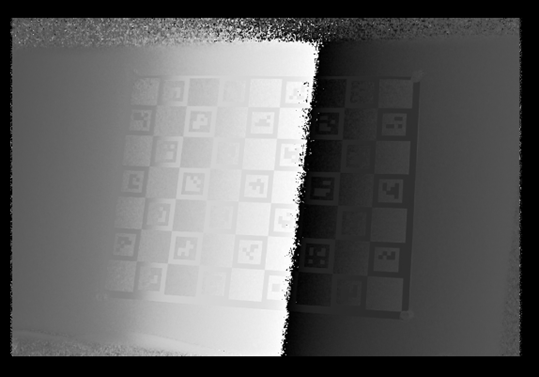


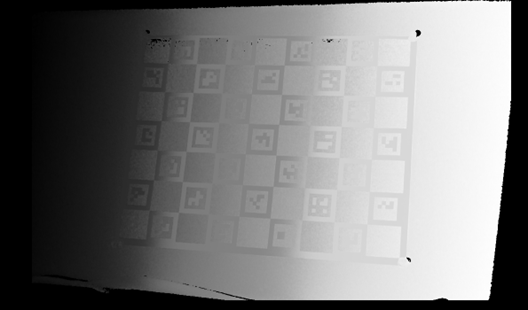
In the context of the 2D plane image the BFS works as the following

Each pixel is a node, which has 4 edges named up, down, left, right

The BFS then flood fills the image with unwrapped phase values vising the neighbors of the unwrapped nodes, then continuing the process, (similar to the 2D forest fire algorithm)







Calibrate

The next step is to calibrate the current setup, this means finding intrinsic parameters and the extrinsic relationship between the camera and projector, relative positioning.

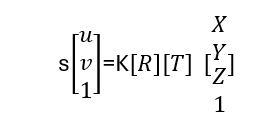
Frist, detect the corners in the white image taken from the set aruco.detectMarkers() which will get the 2D coordiantes of the camera under and the marker IDs

Second is to find the projector coordinates for each corner, using the unwrapped tiff files and the invalid mask images the script calls getCameraCoordinates() to then compute the value of the 2D being floating projector coordinates.

Third, is to get world coordinates from calibration board. The Charuco board provides known coordinates in 3D locations being (X, Y, 0)

Call BoardInfo.charucoBoard.getChessBoardCorners()=

Fourth is the intrinsic calibration, running twice and separately for the camera and projector using cv2.aruco.calibrateCameraCharuco() solving for the eqution below



Where K is the intrinsic matrix 3x3 that has focal lengths and point

R,T are the pose of the board in camera/projector frame

S is the scale factor

Finally, the stereo calibration cv2.stereoCalibrate(…) solve for R, T, E, and F ensuring consistent geometry between both devices for 3D triangulation. Storing all data inside the npz file that will be used in reprojection and can be used by the reader to display said information

References

1. <https://www.sciencedirect.com/science/article/pii/S0143816609000529>
2. <https://dl.acm.org/doi/pdf/10.1145/1179352.1141977>
3. <https://opg.optica.org/ao/fulltext.cfm?uri=ao-21-14-2470&id=25943>
4. Ghiglia and Pritt, *Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software*, 1998.

<https://www.researchgate.net/publication/334660630_High-speed_three-dimensional_shape_measurement_based_on_shifting_Gray-code_light>

<https://www.sciencedirect.com/science/article/abs/pii/S0143816618302550>

<https://www.sciencedirect.com/science/article/abs/pii/S0263224122005437>

<https://www.sciencedirect.com/science/article/abs/pii/S0143816616300653>

<https://engineering.purdue.edu/ZhangLab/publications/papers/2006-oe-calib.pdf>

<https://iopscience.iop.org/article/10.1086/500723/fulltext/>

<http://eia.udg.edu/~qsalvi/papers/2004-PR.pdf>

<https://scispace.com/pdf/temporal-phase-unwrapping-algorithms-for-fringe-projection-1sjzvmx12j.pdf>

<https://dl.acm.org/doi/pdf/10.1145/1141911.1141977>

<https://www.sciencedirect.com/science/article/pii/S0143816609000529>

<https://link.springer.com/article/10.1007/BF01230201>

<https://www.spiedigitallibrary.org/journals/optical-engineering/volume-45/issue-8/083601/Novel-method-for-structured-light-system-calibration/10.1117/1.2336196.full>

<https://opg.optica.org/ao/fulltext.cfm?uri=ao-21-14-2470&id=25943>