Decode, Unwrap, Calibrate Process

This is a tutorial process document outlining and showing the process from decoding the images taken previously by CaptureCode.py to then decoded unwrapped and then calibrated. The process ends before reprojection and thus there is no information on the subject.

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

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

=2

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

If () AND () then reject/mask pixel

Unwrap

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

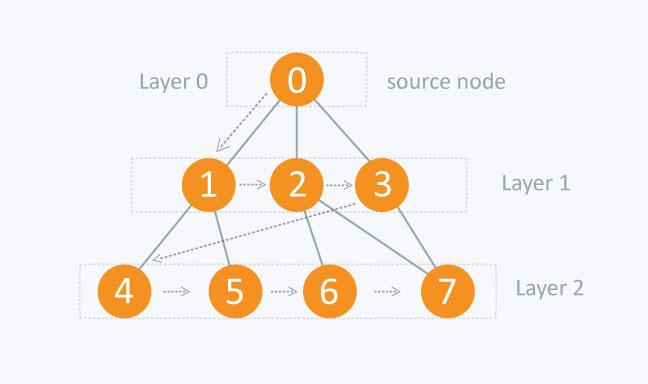
A graph of a function

AI-generated content may be incorrect.

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

: 1 Dimensional

Incorporating a queue driven Breadth First Search (BFS) algorithm to the 2D plane a new equation set acquired for the 2D image plane is



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)

A black and white checkered board

AI-generated content may be incorrect.

A close-up of a chess board

AI-generated content may be incorrect.

A close-up of a chess board

AI-generated content may be incorrect.

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

s=K

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

<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>