Structured Light Summary

**Repository Purpose**

The repository implements a pipeline for 3D reconstruction using structured‑light techniques, specifically *Gray Code* OR *Phase Shifting*. The README introduces the project’s goal: using a projector and camera to acquire a scene and reconstruct it in 3D, referencing related research papers.

**Usage Overview**

Lists each major folder and its role in the 3D imaging using Structured Light process:

* CaptureImages includes the scripts needed to generate patterns for Gray of Phase methods, which are then projected then captured using a camera. The main code is named CaptureCode.py to generate the images inside of a folder named by the user under the captures folder which is inside the CaptureImages folder
* DecodeGrayImages contains script that converts the images captured from CaptureImages into codemaps which are .tiff files (can be viewed in image rendering app)
* CameraCalibration contains code and patterns for projector-camera calibration, leading to the .npz file, can be read with npz-reader, which will display the return value and instrinsic matrix to be viewed
* ReprojectImage reprojects the decoded correspondences using calibration results, producing point clouds. main.py and a variant with reduced distortion perform the reprojection and downsampling. .ply files are created, filtered is the final important file, filter values can be changed inside the code

**Key Components**

*Structured Light Pattern Generation*

CaptureImages/structuredlight holds a modular framework for pattern generation and decoding. The base class defines helpers for splitting, merging, and thresholding pattern images. Implementations like phaseshifting.py and gray.py provide specific algorithms for phase-shifting and Gray codes.

*Image Capture*

CaptureImages/CaptureCode.py drives the projection of patterns, captures images from the camera. Can be Gray or Phase patterns that are projected and captured, decided by a boolean value in the script.

*Pattern Decoding*

The DecodeGrayImages directory contains C++ sources such as DecodeGrayImages.cpp, which read captured frames, separate direct/indirect light, decode Gray code patterns, and output correspondence maps.

*Calibration*

CameraCalibration/main.py loads captured images, detects ChArUco markers, and performs stereo calibration between projector and camera, creating a intrinsic matrix and return value, shown in the .npz file if read by the npz-reader

*Reprojection and Point Cloud Export*

ReprojectImage/main.py loads calibration parameters and decoded correspondence data from object captured images (not included in the calibration sets), computes 3D points using OpenCV and Open3D, and writes both raw and filtered point clouds to disk which can be opened and viewed

*Orchestrator.py*

This is a script that runs through this entire process, removing a large amount of the human factor from the tutorial document if followed. Follows a IO script and displays current step, able to start at different areas with different parameters as well.

fil

**Supporting Assets**

* OpenNI2 provides the OpenNI2 SDK, used in Kinect-related scripts.
* docs/pngs holds many figures referenced in the README.
* CameraCalibration/BoardInfo.py and related files define board layouts for calibration.

Overall, the repository is organized around distinct stages of structured‑light 3D reconstruction: calibration, image acquisition, decoding, reprojection, and optional map generation. The Python scripts drive most stages, while C++ utilities handle performance‑critical decoding.