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.

**Top‑Level Layout**

CaptureImages/

Image capture utilities along with ability to capture images for Gray Code or Phase Shift Structured Light patterns

DecodeGrayImages/

Files needed to decode Gray or Phase projected pattern captured images

CameraCalibration/

Files needed to calibrate the Gray or Phase generated .tiff files from decoding step

Camera\_calibration\_out/

Where detected corner images are stored from calibration step

ReprojectImage/

Final step file which generates the point cloud .ply files to be rendered in *application*

.vscode/

.json files used for running code inside Visual Studio Code sestup

OpenNI2/

Visual Studio project managment folder

Build/

Files to make C files for c++ code, such an executable

Docs/pngs/

ReadME document images

RunOctomap/

Used for Octomap creation

**Usage Overview**

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

* CaptureImages includes scripts for generating patterns, projecting said images, then capturing projected images using connected camera. It uses CaptureCode.py to generate, project, and then capture images of created scenes.
* DecodeGrayImages converts captured pattern images to binary/gray code maps; it includes a Makefile for compiling a C++ program.
* CameraCalibration contains code and patterns for projector-camera calibration.
* ReprojectImage reprojects the decoded correspondences using calibration results, producing point clouds. main.py and a variant with reduced distortion perform the reprojection and downsampling.

**Key Components**

*Calibration*

CameraCalibration/main.py loads captured images, detects ChArUco markers, and performs stereo calibration between projector and camera.

*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 a camera, and optionally interacts with a Kinect device. It also invokes the compiled decoder for Gray codes when required.

*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.

*Reprojection and Point Cloud Export*

ReprojectImage/main.py loads calibration parameters and decoded correspondence data, computes 3D points using OpenCV and Open3D, and writes both raw and filtered point clouds to disk.

**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.