For an up to date version:
https://docs.google.com/
Doc?docid=0AWEDVAtaGbOtZGY2bjZwMmtfNTc5ZjViZ2p4Z3g&hl=en&authkey=CMbF_YkD

To Run Structured Light:
Two options: HDR or single capture

- run_dual (single capture) *or* run_bracketing (HDR)
- Modify root folder/output params/...
- run_bracketing only takes images
- run_dual takes images and creates depthmap+point cloud

To Run Stereo:
Use GigEViewer to grab images

- First rectify images
- stereo_main.m to run correlation algorithm -- adjust parameters!

Notes:

- Point cloud or "model" size: n-by-4 [x, y, z, grayscale color]
- Voxels are based on cells. Each cell contains all of the points that belong to that voxel.

Any questions email Colin (colincsl@gmail.com)

Stereo Functions
baseline.m: Calculates what the Stereo baseline should be for given parameters
depth_error.m: Calculates pixel error/quantization for a stereo setup

rectify_stereo_pair.m: [From calibration toolbox]
Stereo_main.m: Generate a point cloud from stereo images (rectify first!)
stereo_overnight: Used for batch rendering with Stereo_main.m
stereoCleanup.m: Interpolate depth map, clean, and generate point cloud

---

Structured Light Functions
run_bracketing: Main SL function (w/ HDR) Gathers images at several different exposures to generate HDR images. Does not create the final HDRs or generate a depthmap; use in the field.

run_dual: Scans one dataset and generates depthmap & point cloud
camera_setup: setups up camera. Sets parameters to 'manual' so the whitebalance/other features don't change over time.
cleanupModel: Uses standard deviation and gradient filters to reduce noise
display_patern: Only currently used to generate the black and white
getExposureBracket: returns images for each requested exposure time
getImage: return image from camera. Checks if the camera is running and if an image is ready
gray2dec: Converts graycode to a decimal value (from Siggraph Course, Doug Lanman)
hdrRun: Alternate way to generate HDR image from specific file locations.
interpImage: Interpolates a depthmap includes threshold input
outputModel: Output point cloud to X3D
outputModel_dir: Output point cloud to X3D (uses current directory)
processBracketing: Generate HDR from multi-exposure images. Creates depth map and point cloud

SLI_bin_decode_a: Compares each value in an image to check if there is a 'white' or 'black' stripe.
SLI_bin_decode_a_hdr: Slightly different implementation. Not currently used but useful if you don't have good ('white'-'black') difference image to reference.
SLI_bin_decode_dual_b: Takes decoded data, transforms, and generates a point cloud using stereo_triangulation (matlab calibration toolbox)
fullscreen.m: Sends a full screen image to the projector
closescreen.m: Closes the full screen mode

Tools
AlignData*:* Uses ICP to align datasets
boxAnalysis*:* Results of cardboard box fiducial analysis from the Cave 8/4/10
centerModel*:* center a point cloud about (0,0,0)
checkerSize*:* Used to analyze checkerboard model
cleanupVoxels*:* (Misnomer) Cleans a depth map using gradient filtering
cleanupVoxels2*:* Clean by filtering out voxels with low point densities.
depth2voxel*: generate voxels from a depthmap*
getVoxelValue*:* Used in model2voxel and depth2voxel. Not for analysis.
hdrBatch*:* batch generate HDRs from multiple exposures
model2voxel*:* generate voxels from a model. Dependent on bin size, *not* resolution.
outputModel*:Output point cloud to X3D*
outputVoxel*:*Output voxels to X3D
pca*:* Do principle component analysis on subset of voxels
processBracketing_SLpart2*:* Use SL HDR images and generate point cloud/depth map
processHDR*:* Process structured light HDR images
processHDR_SV*:*process stereo HDR images (currently does not work -- Use HDRshop instead??)
voxelPinhole*:* takes point cloud and outputs depthmap
X3D2model*:* takes X3D model and outputs point cloud
voxel_examples.m: Shows how the voxel functions can be used together.