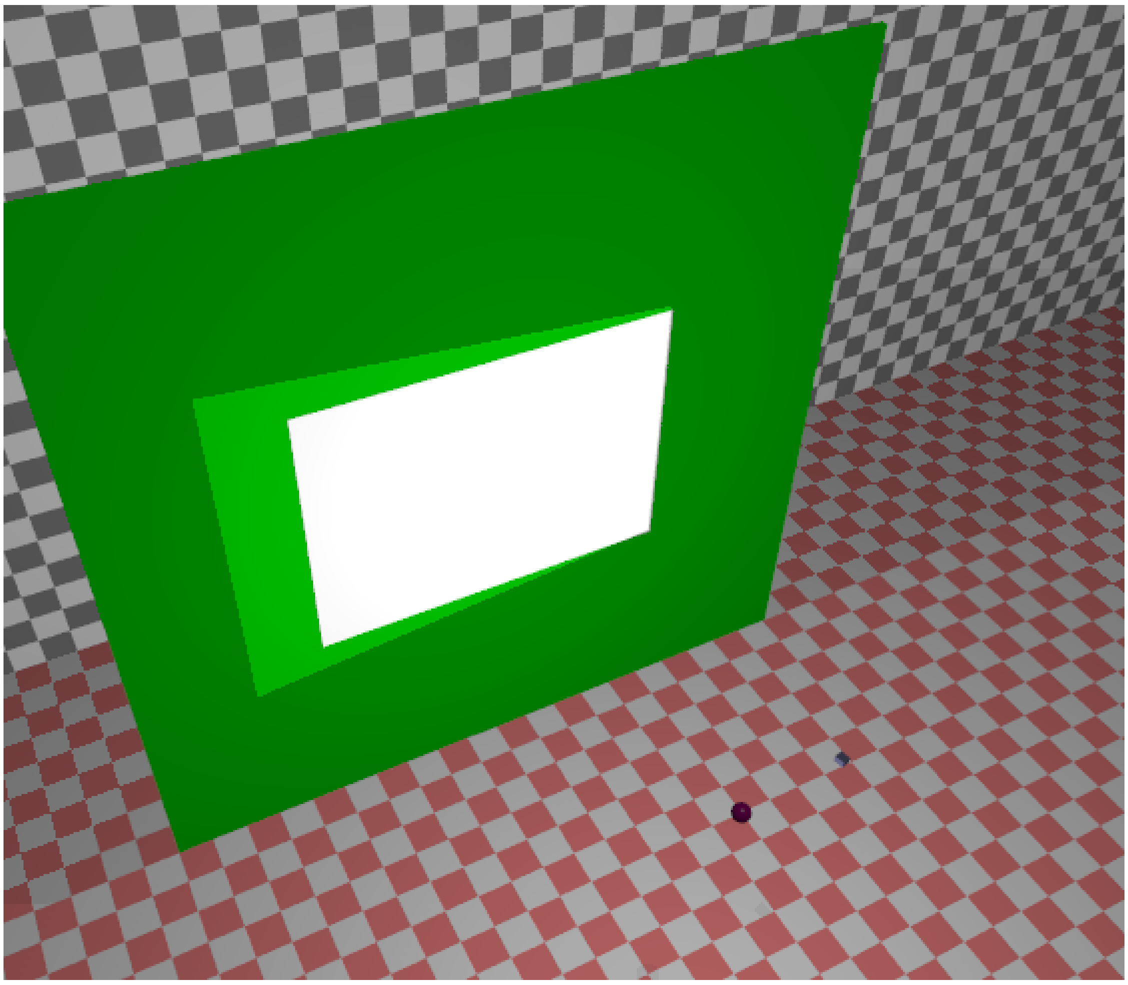
Structured Light Survey

christoph(a)structuredLightSurvey.de

Version 1.1 – March 2009

Here are some details about how the survey should work.

I have prepared a virtual structured light system with the open source raytracer povray. It consists of one camera and one projector. Adding a second camera would not be a problem. You can see the geometry in the following image. The sphere is the camera, the box is the projector, the green is the screen.



Step 1 – Calibration

The calibration data is listed below, in arbitrary povray units. We can assume 1 au = 1000 mm. Note that povray uses a left-handed coordinate system, so the signs on y might have to be changed.

Camera
- Location <0,0,-1>
- Rotation <unity> (facing origin)
- Focal Length 0.012

Resolution 800 x 600
Principal point <400,300> pixels
Pixel Size 1.25e-5 square
no distortion, ie kappa = 0

Projector
Location <0.25,0.01,-1>
Rotation around y by -atan(0.25) (also facing origin)
Focal Length 0.02
Resolution 1024 x 768
Principal point <512,384> pixels
note that the projector is a bit unusual in that it
does not use off-axis projection
Pixel Size 1.953125e-5 square
no distortion, ie kappa = 0

You can enter the data given above into your structured light software. Of course the raytracer may also be used to generate calibration images that can then be fed into your calibration algorithm to test that.

Step 2 – Illuminating the scenes

Various object are placed in the scene and illuminated with the respective structured light pattern. This should be provided as a bitmap which will work like a transparency, ie white will yield white, red will yield red. The standard resolution is 1024x768. Other sizes with the same aspect ratio of 4:3 should also work, but you will have to adjust the pixel size of the projector.

The scene is then raytraced in a resolution of 800x600. The images will be too perfect, so they will be degraded with Gaussian noise to make for a more interesting challenge. This can be achieved easily with Matlab's imnoise function. Variances of 10^{-4}, 10^{-3} and 10^{-2} will be used.

Ground truth images for each scene are generated by the raytracer as well.

Right now there are eight scenes, some based on 3d mesh data from other people, some wholly synthetic.

dragon
an example of a volume scatterer
http://graphics.stanford.edu/data/3Dscanrep/

blade
a typical technical part. rendered once with reflective silver material and once with almost black and rough material.
http://www.cc.gatech.edu/projects/large_models/

cells
a field of small cubes with different z
lots of small shadows break up the pattern

keyboard
a computer keyboard - example of a discontinuous surface

aleuts
a relatively smooth surface, but with high contrast texture
generated from an aerial image of the aleutian islands

sun
another heightmap generated from an image of the sun's surface
large distortions of the pattern, also some texture

grid
an object with a lot of holes

ring
a scene that violates the ordering constraint

The surface properties of the objects can be manipulated within limits, i.e. there is color, specular reflection, diffuse reflection and volume scattering. The scenes contain a selection of these. There is currently no color crosstalk in the virtual camera, but it could be simulated by post-processing the image. Defocus blur is also possible, but not used at the moment.

I am open for suggestions regarding additional scenes.

Step 3 – Putting it together

The raytraced images are used as input for the respective structured light decoding software. The results can then be compared using the Matlab script also provided on the website. It expects input in simple ascii x,y,z format. The ground truth data must also be converted from depthmap format to true 3d x,y,z.

The evaluation criteria are the following. Please turn off all interpolation and smoothing to ensure a fair comparison. Only one set of parameters should be used for all images.

- number of depth values recovered
- difference to ground truth (mean and standard deviation)
- runtime (only order-of-magnitude because of hardware differences)