DNG noise model

Document version: 1.1
Document date: 18th November, 2014

1. Document scope

This document describes the process for generating the DNG noise model for a given camera model, using the `dng_noise_model.py` script (which builds on the ITS infrastructure). The output of this test is a C code snippet than can be cut-and-pasted into the camera HAL for a device; note that this is a per-model set of parameters, not a per-unit set.

2. Running the test

2.1. Physical setup

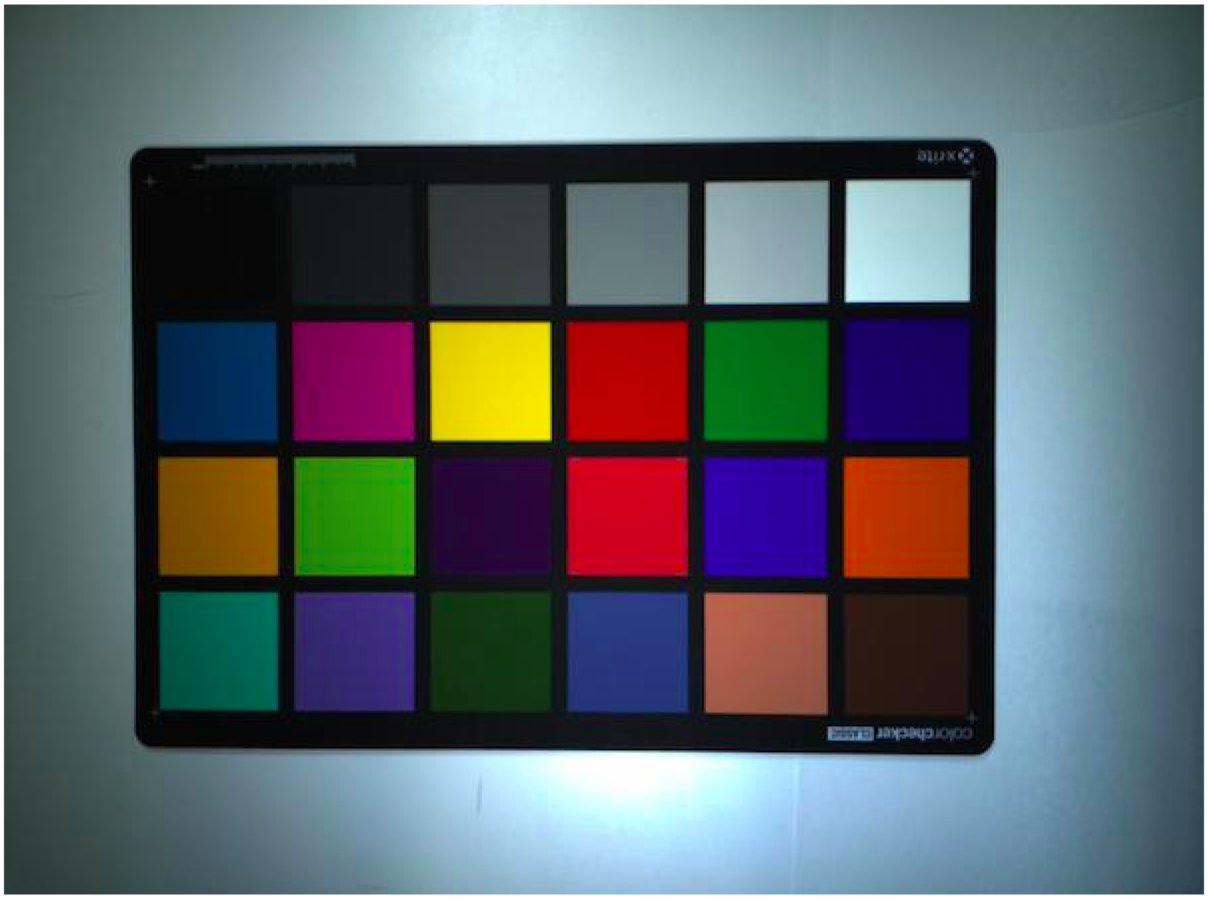
The target used for the script is a color checker chart, for example this one:
http://www.amazon.com/dp/B000JLO31C

The script uses a pretty simplistic way to automatically extract the color patches from the images, and there are a number of requirements for the physical scene setup to make this work reliably:

- Chart is vertical or horizontal w.r.t. camera, but not diagonal.
- Chart is (roughly) planar-parallel to the camera.
- Chart is centered in frame (roughly).
- Around/behind chart is white/grey background.
- The only black pixels in the image are from the chart.
- Chart is 100% visible and contained within image.
- No other objects are visible within the image.
- The chart doesn't cast a shadow (so it probably needs to be flat on a surface).

- Illumination is reasonably uniform (though this doesn't need to be precise).

The following shot illustrates a well-framed chart for the purposes of this script. The lighting uniformity could have been a bit better, but it was good enough for the script to work.



Note that it's OK that the chart is upside down; it can be in any of the four main orientations and the script will work fine.

2.2. Invoking the script

Once the ITS environment is set up, the following command runs the script:

```
python tests/dng_noise_model/dng_noise_model.py
```

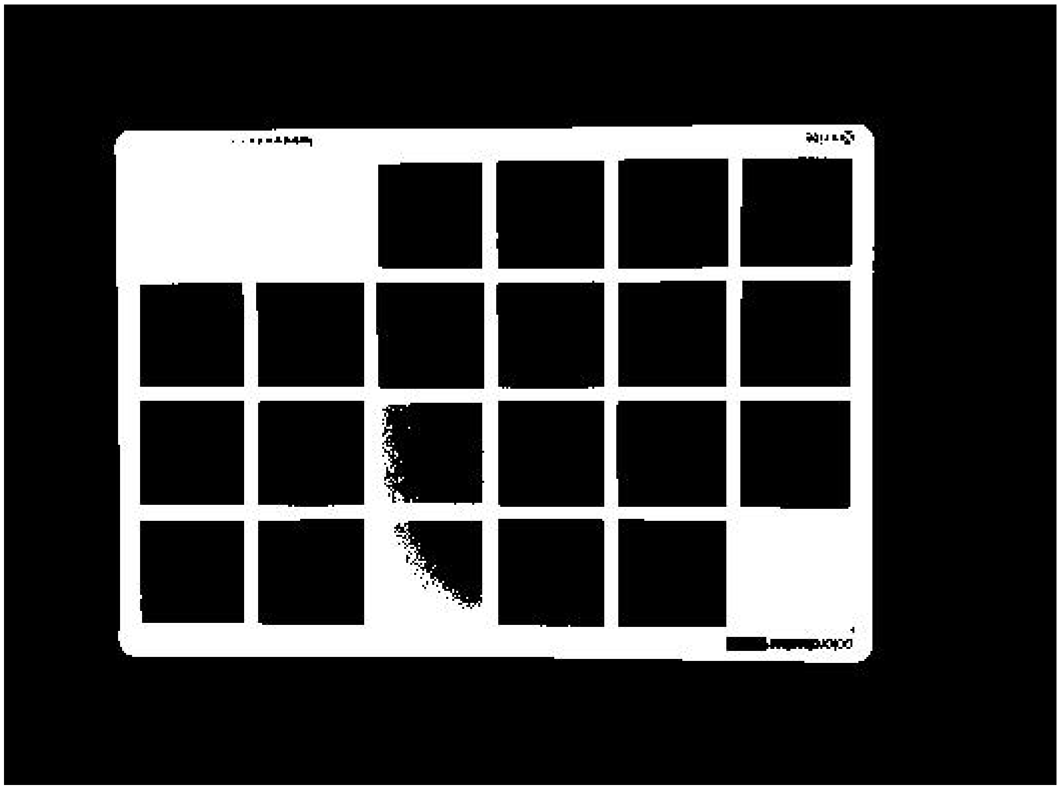
This captures several raw images from the device, copies them back to the host machine, and performs some analysis.

2.3. Script output

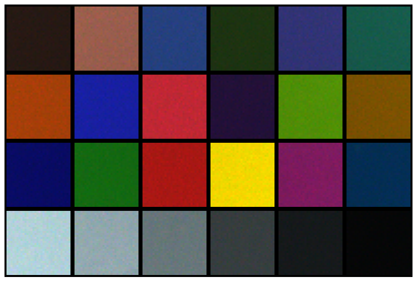
2.3.1. Debug images

The script outputs several debug images which should be inspected to ensure that it ran without problems. The first, `dng_noise_model_debug_0.jpg`, is just an image of the scene, such as the one shown above; the chart should be well-framed according to the requirements listed above.

The second debug image, `dng_noise_model_debug_1.jpg`, shows the result of the script's attempt to isolate the chart within the image. If it runs successfully, this image should look something like the following, where the chart's outline is visible and there is only black all around it:



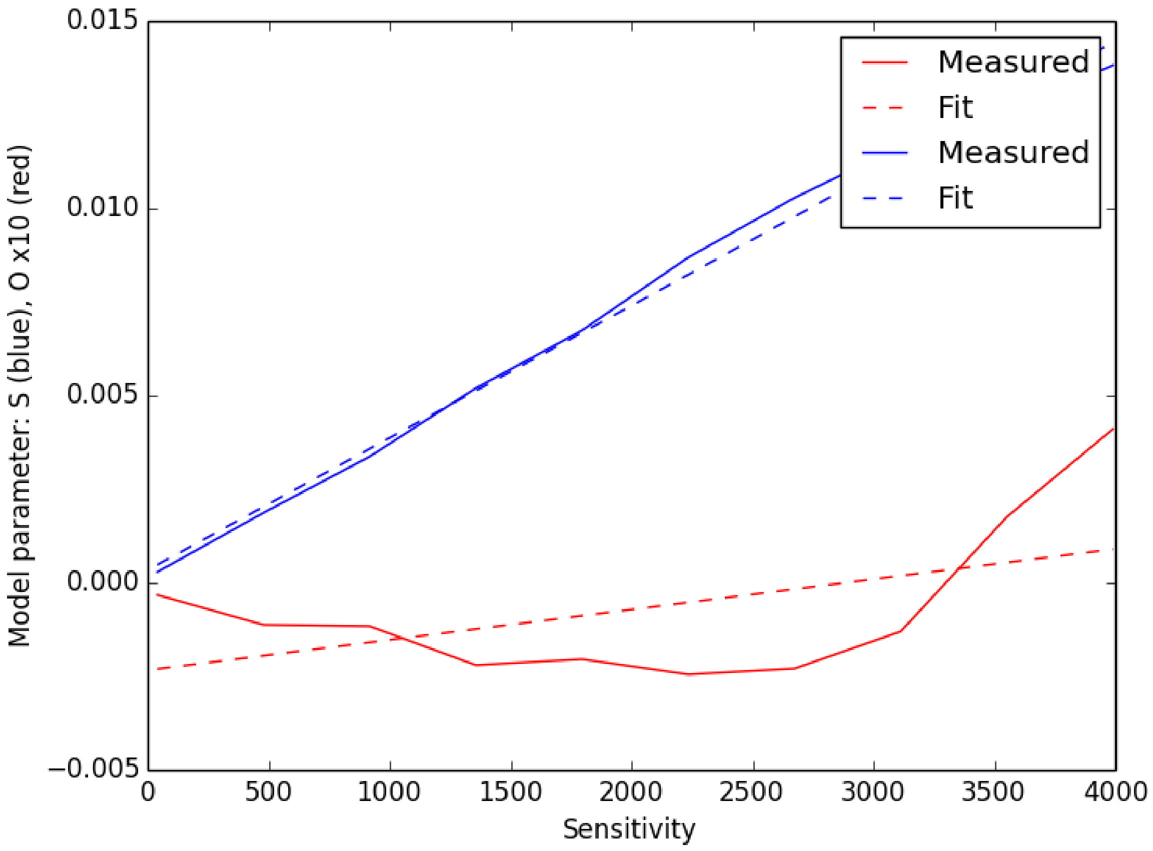
The third debug image, `dng_noise_model_debug_2.png`, contains the array of 6x4 patches that were automatically extracted from the chart image, and if all goes well it should look like the following image:



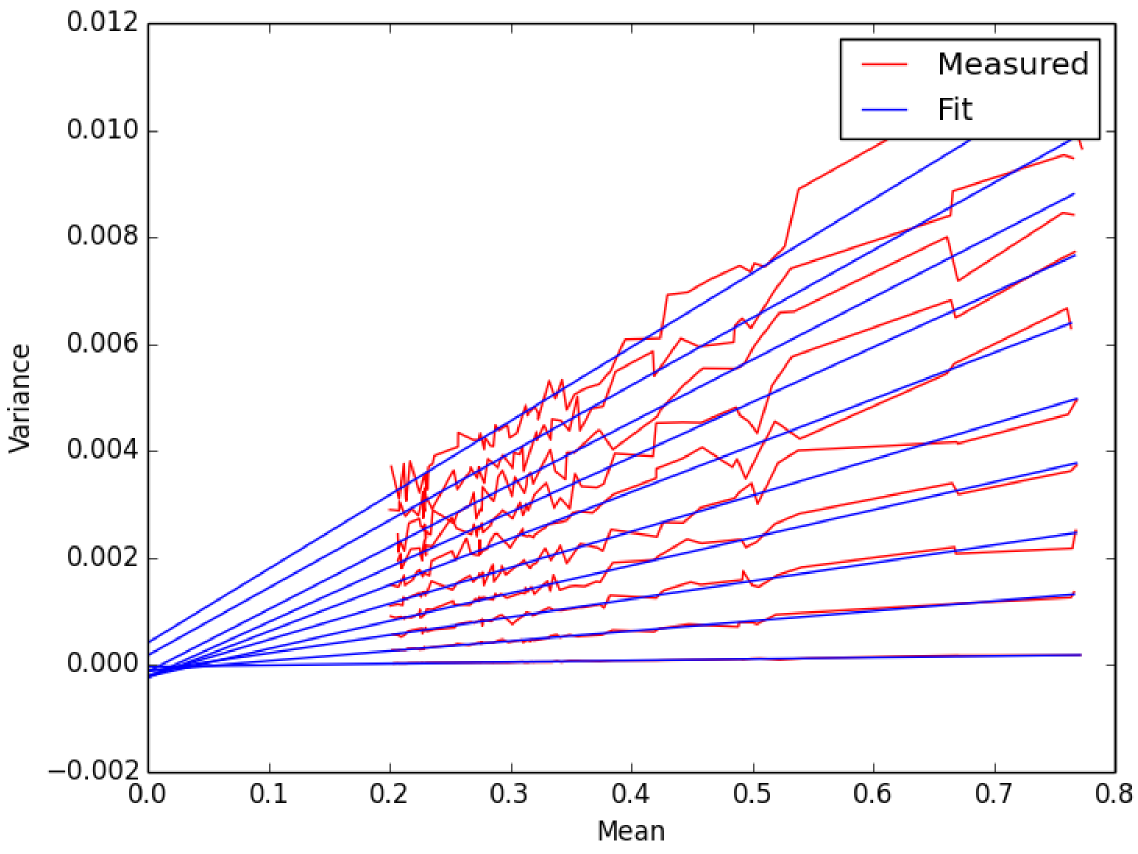
If the debug files generated by the script run don't look like those above, then the script should be considered as having failed, and if any noise model parameters were generated they should be considered garbage.

2.3.2. Plots

The script also outputs two plots, to help visualize the model that was generated. The first, `dng_noise_model_plot_S_O.png`, shows the actual (O,S) tuple that is generated in the model for each sensitivity, in the form of a pair of lines (with both the measured data and the fit lines shown in the plot).



The second plot, `dng_noise_model_plot_mean_vs_variance.png`, shows the measured (mean, variance) pairs for each patch, for each sensitivity; each blue line depicts the relationship between mean and variance for a single sensitivity level, with the data for the plot obtained by sampling each patch from the raw image captured at that sensitivity.



It's not important to understand the meaning of these plots, however the plots generated by a run of the script should resemble the two above, else the script should be considered to have not run successfully.

2.3.3. Code snippet

The test also prints some code to the console, which will look like the following:

```
/* Generated test code to dump a table of data for external
 * validation of the noise model parameters.
 */
#include <stdio.h>
#include <assert.h>
double compute_noise_model_entry_S(int sens);
double compute_noise_model_entry_O(int sens);
```

```
int main(void) {
    int sens;
    for (sens = 40; sens <= 4000; sens += 100) {
        double o = compute_noise_model_entry_O(sens);
        double s = compute_noise_model_entry_S(sens);
        printf("%d,%lf,%lf\n", sens, o, s);
    }
    return 0;
}

/* Generated functions to map a given sensitivity to the O and
 * S noise model parameters in the DNG noise model.
 */
double compute_noise_model_entry_S(int sens) {
    double s = 3.530980e-06 * sens + 3.357493e-04;
    return s < 0.0 ? 0.0 : s;
}
double compute_noise_model_entry_O(int sens) {
    double o = 8.098436e-08 * sens + -2.336529e-04;
    return o < 0.0 ? 0.0 : o;
}
```

The entire code snippet can be pasted into a C file, compiled, and executed, and it will dump data that can be used to visualize the generated model by plotting it (for example using a Google Docs spreadsheet).

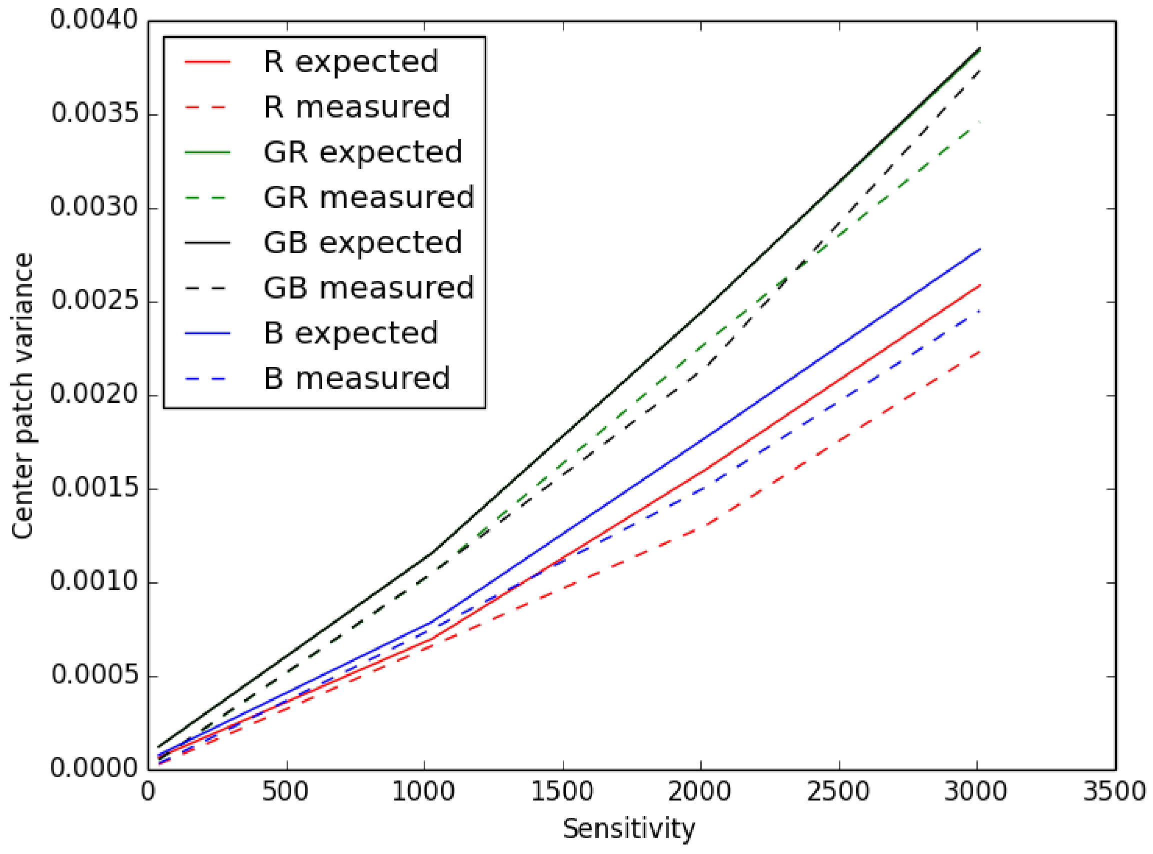
The bolded portion of the C code output can be used in the camera HAL to generate the O,S model parameters for any given sensitivity. The generated code can of course be modified as appropriate to fit within the camera HAL's code style and other requirements.

3. External validation of the DNG noise model

Once a device is up and running with the camera HAL producing the O,S model parameters with each raw shot, one of the automated ITS tests can be used to validate that it's working as expected. This is a "scene 1" ITS test, meaning that it assumes a simple grey card inside a light box as the test scene. The test can be run manually as follows:

```
python tests/scene1/test_dng_noise_model.py
```

The output of this test (aside from an automated pass/fail check) is a plot, `test_dng_noise_model_plot.png`, which depicts the measured variance of a center patch of the grey card in raw shots captured over a range of sensitivities, and compares these values with the variance that is expected at each sensitivity by the DNG noise model in the camera HAL (based on the O,S parameters returned in the capture result objects). A successful run will look as follows:


R expected
R measured
GR expected
GR measured
GB expected
GB measured
B expected
B measured
Center patch variance
Sensitivity