

# MTF Mapper user documentation

Frans van den Bergh (fvdbergh#at#gmail#dot#com)

November 28, 2022

## Contents

<b>1 Overview</b>	<b>2</b>
<b>2 Getting started with the GUI</b>	<b>2</b>
<b>3 Modes</b>	<b>3</b>
3.1 Annotation mode (-a or --annotate flag) . . . . .	3
3.2 Profile mode (-p or --profile flag) . . . . .	3
3.3 MTF Surface mode (-s or --surface flag) . . . . .	4
<b>4 Autofocus fine-tuning</b>	<b>5</b>
4.1 Caveats and disclaimers . . . . .	6
<b>5 Other MTF mapper utilities</b>	<b>7</b>
5.1 Rectangle generator . . . . .	7
5.2 Test chart generator . . . . .	7
<b>6 Frequently Asked Questions</b>	<b>8</b>
<b>A Tips for printing the test charts</b>	<b>8</b>
<b>B Flat 45° test charts <i>vs.</i> perpendicular charts</b>	<b>9</b>
<b>C Internal workings</b>	<b>11</b>
<b>D Accuracy</b>	<b>12</b>
<b>E User manuals for individual applications</b>	<b>13</b>

# 1 Overview

The MTF mapper package offers a collection of tools to measure Modulation Transfer Function (MTF) values across edges in images. It does this by computing the edge spread function of a step edge in an image, using a method similar to the one described by Khom [1] (Section C provides a brief overview). MTF50 values correspond to perceived sharpness of edges; this means that the MTF mapper tools can be used to evaluate camera lens sharpness, as well as autofocus behaviour.

MTF mapper offers fully automated operation, producing the following outputs:

1. Annotated images, where the MTF50 value of an edge is printed on top of the edge itself;
2. Profile data sets, where the MTF50 values are represented in a one-dimensional projection of the image. This is the tool you want if you are interested in objectively adjusting your DSLR autofocus fine-tuning (see Section 4 for details of the method).
3. MTF surface images, where you can visualise the MTF50 values across the focal plane, to see the image centre MTF50 relative to edge MTF50, for example.

MTF mapper expects images containing dark rectangular objects on light backgrounds; the objects can be slightly out-of-square, e.g., trapezoids or parallelograms, provided the interior angles are at least reasonably close to  $90^\circ$ .

Special test charts are required for Profile mode. Section 3.2 describes them in more detail.

Special test charts are also required for MTF surface mode. Section 3.3 describes them in more detail.

The two main programs of the MTF mapper package are called `mtf_mapper` and `mtf_mapper_gui`. The GUI is just a front-end to the command line tool.

A full reference of command line options for the various tools are provided in Appendix E.

## 2 Getting started with the GUI

After launching the GUI, you will be faced with a screen that looks like the one portrayed in Figure 1.

The right panel (called “selection”) will contain a list of available views once some files have been processed. Selecting a view from this list will cause the relevant image to be displayed in the left panel. First off, though, you will have to choose some files to process.

This is accomplished by selecting File/Open from the menu. You can select multiple files to be processed; these files can be in a variety of image formats, including `.png`, `.jpg`, `.pgm`, or any raw format supported by `dcraw`, including `.nef`, `.cr2`, `.pef`, and `.arw`. Note that all previous outputs will be cleared from the “selection” panel when new files are selected. The implication of this is that all files must reside in the same directory.

Once the selection panel has been populated, you can click on any of the items to view the relevant data. Clicking on the image filename will display the original image in the output window; its children will be the outputs generated by the various modes. The checkboxes in the selection panel can be used to enable/disable the various output image types. This is

useful when using the keyboard to move up or down the list; disabled output types will be skipped.

You can drag the image in the output window to pan, or you may choose to use the scroll bars (if present). The +, -, and = keys may be used to zoom in, zoom out, or zoom to 100%, but you may have to click on the image first to set focus.

That concludes the very brief overview of the MTF mapper GUI. The rest of this document will describe what the various modes and output types can be used for.

## 3 Modes

The main tool in the package is called `mtf_mapper`; you could simply invoke it as

```
mtf_mapper <image> <working_dir> [-p|-r|-s|-a]
```

where `<image>` can be in a variety of image formats, including PNG, JPG, TIFF, PNM, and more. A working directory, and the desired combination of modes, must also be specified. After successful extraction of MTF50 values from the edges found in the image, the information is made available in several forms, called “modes”, detailed next.

### 3.1 Annotation mode (-a or --annotate flag)

No explanation required, really. Input images are searched for rectangular objects. Once found, the MTF50 value will be computed across each edge of all the rectangular objects. The resulting value is drawn on top of each edge, and the result saved as `annotated.png` by default.

### 3.2 Profile mode (-p or --profile flag)

Using a special test chart illustrated in Figure 2, the program will construct a profile such as the one shown in Figure 4, which was derived from the original image shown downsampled in Figure 2(b). The chart should be photographed at a 45° angle, preferably at the specified distance. Each chart is optimised for a specific viewing distance to counter the effects of perspective distortion. In practice, this means that the image of the chart will look like the one in Figure 2(b), i.e., the rows of blocks to not appear to taper, and the small blocks are all perfect little squares. Note how perspective distorts the shape of the foam board on which the chart has been mounted; the bottom edge of the foam board is clearly not parallel to the image edge, while the rows of square appear to be parallel to the image edge.

Thi

The large central block in the chart (Figure 2) is called the reference block, and the edge of this block closest to the centre of the chart (the bottom edge in Figure 2a) is called the *reference edge*. The MTF50 values computed across the edges of all the blocks in the image are projected onto the  $y$ -axis<sup>1</sup> of the image, thus forming a new set of data points of the form ( $y$ -value, MTF50 value). This process is illustrated in Figure 3. The idea is that about half of the chart will be in front of the plane of focus, and the other half behind. If the depth of

---

<sup>1</sup>MTF Mapper will auto-detect the orientation of the image. For ease of discussion, this is called the  $y$ -axis.

field is sufficiently shallow, then the closest and furthest of the small blocks will be noticeably blurry. By projecting the measured sharpness value (MTF50) of each block along horizontal lines (Figure 3), we obtain a roughly bell-shaped profile as shown in green on the left. The peak of this curve corresponds to the plane of focus, and blocks that are further or closer than this plane are out of focus to some degree. The blue line indicates the position of the *reference edge*.

A complete profile in its usual orientation is shown in Figure 4. The red dots represent individual MTF50 measurements, and the green curve is merely a smoothed representation of the same data.

Generally, *Profile* mode is only intended to be used to calibrate or evaluate the autofocus sensor of a DSLR; the details of this process are described in Section 4. The objective is to adjust the camera so that the blue line (reference edge, or focus position) lines up with the peak of the green curve / red point cloud. If the blue line is far from the peak, then you are experiencing either front- or back-focus. If your chart was positioned at a 45° angle so that the bottom edge was closer to you, then front focus would mean that the blue line appears to the left of the peak in the green curve. This depends on the orientation of the camera, though, so you may want to take a look at the annotated image (see Section 3.1) to orient yourself. The trick is to remember that the peak in the green curve (or red point cloud) corresponds to the *actual* plane of focus, whereas the blue line corresponds to where MTF Mapper *assumes* you have placed the autofocus sensor when you framed the shot.

### 3.3 MTF Surface mode (-s or --surface flag)

Using any test chart with a regular grid of rectangles (as illustrated in Figure 5), you can measure the acuity of your lens/camera system as it varies across the focal plane. Just shoot the chart at a reasonable distance, making sure that it covers the entire viewfinder. For typical lenses and unsharpened images, chart alignment is not critical.

Surface mode produces two images: one representing a 2D representation of MTF50 values across your image, and another showing the same data rendered as a 3D surface. Examples of both can be seen in Figures 6 and 7. These MTF maps show that this particular lens (a Nikon Nikkor 35 mm F1.8 prime lens, stopped down to F2.8) is fairly sharp, but appears to have some strange asymmetrical defect — not that you can tell by looking at photos taken with this lens, though.

## 4 Autofocus fine-tuning

The following steps can be used to calibrate the autofocus fine-tuning of a DSLR:

1. Figure 8 illustrates the basic set-up. The distance  $d$  is the “distance to chart”, and the angle  $\theta$  is the “angle with respect to the test chart”.
2. Print out the test chart at a large enough scale. Ideally, your test chart must be large enough so that you can use it at a distance of  $30\times$  the focal length of your lens. Appendix A offers some advice on printing your test charts.

Position your camera so that you see the chart from an angle of at least  $45^\circ$  — the idea is that you want some of the small blocks to be in front of the plane of focus, and some of the blocks behind the plane of focus; this is easier to achieve at angles of  $45^\circ$  and smaller. The reference edge (Section 3.2) should be exactly at the plane of focus, but since you are reading this, I take it you are still trying to adjust the autofocus fine-tuning to achieve this.
3. You must use a tripod, and it is recommended that you use a remote shutter release or a timed shutter release to minimise vibrations.
4. You should have enough light for an exposure value of 10–11 (for example, ISO 100 F/1.8 at 1/320 s), which translates into about 2500–5000 lux. This amount of light is required to achieve consistent performance from the AF sensor. I use indirect sunlight to reach these levels.
5. Set your camera to AF-S (single-servo AF). Select a single AF point — the centre AF sensor is recommended. This is critical, as any other AF mode / sensor selection will not produce the desired results. On Nikon bodies, I like to use the AF-ON mode so that the camera only focuses when I ask it to. For adjusting your AF fine tuning settings, you should use a single AF operation, i.e., press and hold the AF-ON button until focus lock is achieved, then release the button. Do not focus a second time.
6. Aim the AF sensor reticule so that it straddles the reference edge (see Figure 9). Take care that the autofocus sensor is sufficiently far away from other edges (e.g., the horizontal edges of the reference block in Figure 9, or any of the small blocks). Keep in mind that the actual sensing area of the autofocus sensor is typically larger than the reticule you see in the viewfinder, so leave some padding.
7. Manually set the focus of your lens to the near limit, or to infinity.
8. Initiate one AF operation.
9. Capture a shot of the test chart. Lower ISO values are better, since MTF measurements are sensitive to noise.
10. Feed the captured image through MTF mapper to produce a profile (such as illustrated in Figure 4. The vertical blue line denotes the position of the autofocus reference edge (at least, the one you should have been using to focus ...). The green curve (or red points) records the MTF50 values measured along the long axis of the image. Since the test chart was at a  $45^\circ$  angle with respect to the lens axis, the long axis of the image is a measure of the distance from the camera. MTF50 values measured at different  $x$ -values (in the Figure 4) thus indicate the sharpness, or degree of focus, at that specific

distance from the camera. The peak of the green curve represents the plane of focus — the objective is to line up the peak of the green curve with the blue vertical line.

11. This procedure (steps 7–10) should now be repeated at various autofocus fine-tuning settings on your camera. You should be able to see the peak of the green curve shift left or right as you adjust this value. I recommend capturing your images in batches, first stepping your autofocus fine-tuning through the range in large steps, running the images through MTF mapper, and then repeating this in the optimal range with smaller steps until you are satisfied that you have calibrated your autofocus fine-tuning to the desired level of accuracy.

## 4.1 Caveats and disclaimers

Please note that this procedure of calibrating autofocus fine-tuning on your DSLR is based on some of my own assumptions, which have not been tested rigorously before the release of this software. Here follows some background; you are most welcome to skip this section.

Phase-detection autofocus in DSLR cameras works by collecting light from opposite sides of the lens (the aperture, really), if the article on Wikipedia is accurate<sup>2</sup>. These two beams of light are steered to two independent linear sensors — I suspect that they are simply small strip-like CMOS sensors nowadays. Using cross-correlation, the AF module then measures the phase shift between the data collected from the two linear sensors; this phase shift will directly correspond to the degree of defocus. With this information, the AF module can then drive the AF motor by approximately the correct amount to eliminate the phase difference between the signal received by the two linear sensors, which should bring the object under the relevant AF sensor into focus.

So the real question is: what algorithms do the AF modules really use to measure the phase shift? Well, I currently do not know. If you design AF modules, please fill me in, and I can update my test charts to agree more closely with what the AF modules expect to see. Many AF test charts on are available on the Internet, however, most of them use thick line (bar-shaped) target to draw the AF sensor's attention. Someone on the Internet (now, there is a reference you can count on) pointed out that bar targets are a poor choice, because they may be too thin for the AF sensor to detect. I found this argument appealing, because the AF sensor must have limited resolution. There is an additional problem with a bar target: which edge of the bar target is the AF module going to focus on? And this process led me to the design of my own test charts. Rather than using a bar as an AF target, why not use a step edge? If the AF module really does use cross-correlation to measure the phase difference, then a step edge would produce the best possible results. There would also be not ambiguity as to where the sensor is focusing, since the step edge only has one feature to focus on.

Well, it seems to work. The results presented in Appendix B seem to support this. At least for my camera bodies and lenses.

I also found it annoying to have to use visual inspection to determine whether I have set the autofocus fine-tuning optimally. Visual inspection certainly is a quick way to evaluate the results in the field, but I want to see some objective data. I want to *know* that I have calibrated my lenses perfectly. Anyhow, if you have made it this far, you probably understand. MTF50 is certainly not the final word on image acuity (read the excellent Zeiss papers for a start), but it does provide a reasonable relative measure for the autofocus calibration problem.

---

<sup>2</sup><http://en.wikipedia.org/wiki/Autofocus>

The MTF50 estimates extracted by MTF Mapper are reasonably accurate (see Appendix D), at least as far as internal consistency goes with my own edge image generator. You should be able to obtain repeatable results within a 5% relative margin.

Testing against other implementations (Mitre SFR 1.4, sfrmat3) have shown very good agreement, with MTF Mapper delivering slightly more repeatable results than Mitre SFR (sfrmat4 not tested in bulk yet). The slanted edge plugin for ImageJ does not appear to be accurate.

Lastly, you should know one very important thing about autofocus systems: they are not perfect. The tolerances of the AF system (AF module, lens drive accuracy, etc.) are such that they strike a balance between speed and accuracy. In other words, the AF system only takes one stab at the problem: one phase difference measurement results in one focus motor action. The implication of this is that the AF system will only adjust focus so that it is “good enough”, that is, until the circle of confusion drops below the specified limit. For a Nikon D7000, the circle of confusion limit is 0.02 mm, which translates into an MTF50 value of around 36 lp/mm. The focus plane may therefore be set to any position within the range of distances that yield a sharpness of at least 36 lp/mm.

With MTF mapper, you can empirically observe this effect: manually set focus to infinity, switch to AF, capture image. Now select near focus (manually), switch to AF, capture image. Comparing the profile plots of produce by MTF mapper you should be able to see which image was captured from which direction, and the distance between the peaks in the profiles should roughly correspond to this “zone of acceptable focus” for that subject distance and magnification.

## 5 Other MTF mapper utilities

Two additional utilities are included in the MTF mapper package.

### 5.1 Rectangle generator

The `generate_rectangle` utility will create a synthetic image of a rectangle, sampled with a specified Gaussian point spread function. The rectangle images generated with this utility therefore have a known MTF50 value, which was calculated analytically.

In the current implementation, the Gaussian noise that is optionally added to the image is not taken into consideration in the analytical calculation of the expected MTF50 value. This noise certainly makes it more challenging for software tools to compute the MTF50 values from the generated images, but it should not bias the result in any way.

Either way, the `generate_rectangle` utility is useful during testing, and can be used to cross-calibrate the `mtf_mapper` utility with other packages available for computing MTF50 values. (If you have access to Imatest, please send me some results. Your feedback will help to improve MTF Mapper).

### 5.2 Test chart generator

The `generate_test_chart` utility can be used to generate SVG files containing various test charts. Currently, it can generate “perspective” charts that are suitable for DSLR autofocus calibration, as well as empirical measurements of the depth of field of your equipment. It

can also generate “grid” charts that contain squares arranged in a regular grid, which can be used to measure the flatness of field of your equipment.

Currently, the MTF mapper utilities do not separate MTF50 values into sagittal and meridional orientations, although such an option would be relatively straightforward to add, and is planned for a future version.

## 6 Frequently Asked Questions

A bit of a misnomer, since I have yet to receive actual questions :)

1. Your program gave me a value of  $x$  cycles per pixel. Is this any good? *Answer:* According to Norman Koren (<http://www.imatest.com/guides/modules/sfr>), a value of 0.33 cycles per pixel is pretty good *for unsharpened raw images* (emphasis mine). This is somewhat misleading, though, since expressing MTF50 as c/p is not independent of the sensor resolution, and Norman may have been referring to an 8 MP sensor. Take, for example, a sample of the Nikkor 35 mm f/1.8 prime lens on a D40 body. This combination achieves MTF50 values of around 0.28 c/p. The same lens on a D7000 body achieves around 0.22 c/p. If these values are expressed as line pairs per millimetre (lp/mm), we actually see 36 lp/mm on the D40, and 46 lp/mm on the D7000. In this case, it means that the lens is actually able to resolve more detail than what the D40 could capture. This also explains why I suddenly thought the lens looked softer on the D7000 — the per-pixel sharpness was definitely lower, even though the effective sharpness was higher.

So while c/p units are convenient because they do not require knowledge of the pixel (or sensor) size, they are not portable to other sensors for the very same reason. I prefer to use lp/mm when comparing lenses, but c/p are more natural for synthetic images. Keep in mind that a Bayer colour filter sensor camera will never be able to give you “perfect” MTF50 values of 0.5 c/p, and that anything above 0.25 c/p is actually pretty good (before sharpening), all things considered.

2. Cycles per pixel? I wanted lw/ph or lp/mm! *Answer:* Support has been added for lp/mm by specifying a pixel size with the `--pixelsize` option. You can also convert manually using the following relationship:

$$\text{MTF50}_{\text{lp/mm}} = \frac{\text{MTF50}_{\text{c/p}} \times h_{\text{pixels}}}{h_{\text{mm}}} \quad (1)$$

$$\text{MTF50}_{\text{lw/ph}} = 2 \times \text{MTF50}_{\text{c/p}} \times h_{\text{pixels}} \quad (2)$$

where  $h_{\text{mm}}$  is the image height in mm, and  $h_{\text{pixels}}$  is the image height in pixels.

## A Tips for printing the test charts

I have printed the MTF Mapper test charts using a variety of printers, and have the following advice to offer:

- If you are printing on paper, you should aim to use the thickest paper available. Something like 120 g/m<sup>2</sup> is probably the bare minimum. All my charts printed on standard 80 g/m<sup>2</sup> paper warped horrendously with changes in humidity.

- The test charts work best when they are quite flat. I have worked with an A0 print of the perspective chart that was simply taped to a plastered wall, and everything seemed to be fine. If, on the other hand, you are using the grid chart to look for MTF variations in the focal plane (i.e., Surface mode), then it is critical to keep the chart perfectly flat. I have found that foam board works very well, especially if you use a spray glue to fix the printed chart to the foam board.
- Even spray glue combined with a foam board is not good enough to prevent thin paper from warping due to changes in humidity. If you plan on using a chart more than once, you *must* use thick paper.
- I have yet to try this myself, but a local printer can print on a self-adhesive vinyl sheet. This should be immune to humidity, but obviously this will be much more expensive than printing on plain paper.
- Print quality is probably not critical when using the perspective charts — I have used badly printed (streaky, not quite solid) charts, and that seemed to be OK on an A0 scale.
- Print the largest chart you possibly can for a given focal length. In practice, I have found that the A0-size charts work very well for lenses shorter than 50 mm, since they allow me to keep the chart at a realistic distance. I have yet to perform extensive tests on A4 charts, but I suspect they are only safe to use with longer focal lengths (e.g., 200 mm or longer). Furthermore, at greater distances, you can get away with lower quality prints — 300 DPI prints are fine on an A0 chart, but my become a problem on A4 charts.
- One last tip: If you want the chart to be perfectly aligned at  $45^\circ$  degrees, you can use a small mirror mounted on a  $45^\circ$  block (like the one illustrated in Figure 10). Line up your tripod so that you see straight down the barrel of your lens in the mirror. This level of precision is probably not necessary, and simply measuring out a right-angled triangle to place your tripod is good enough for most users.

## B Flat $45^\circ$ test charts *vs.* perpendicular charts

Many DIY test charts on the web are of the  $45^\circ$  degree type, i.e., you simply print out the chart with whatever printer you have, and you shoot the chart at a  $45^\circ$  angle. These charts have been criticized severely in the online fora, sometimes with reason. The main argument is that these charts often have a line (thick or thin) in the centre of the chart that is to be used as the target for the autofocus sensor. Thin lines can be too thin for the limited resolution of the AF sensor to detect reliably. Thick lines can (probably) be detected more reliably, but they actually present the AF sensor with two edges: a near and a far edge. Which edge did the AF sensor lock onto?

This brings us to the design of the test charts used in MTF mapper (see Figure 2). These charts have a solid block in the centre, with the centremost edge of this block acting as the AF target. The block is tall enough that the AF sensor sees an unbroken step edge along the height of the AF sensor (assuming, of course, that the chart is being photographed in a landscape orientation). The step edge resolves the potential ambiguity that may occur in other charts using thick lines as target. If the AF sensor is using a correlation algorithm to determine the phase shift (my best guess), then the step edge will provide an ideal target.

But does this work in real life? To answer this question, I built an additional prop out of polystyrene. The objective was to build an AF target that was perpendicular to the camera axis, similar to the chart that had been in use on LensTip (<http://www.lenstip.com>) before they switched to using one of the LensAlign tools. Figure 10 shows a crop of the resulting test chart shot.

Using this modified chart it becomes possible to test whether the plain MTF Mapper test chart at 45° produces different AF results than using a perpendicular AF target. Firstly, I should mention that the perpendicular AF target was significantly larger than the AF reticule in the viewfinder (even though this is unlikely to matter, since the entire perpendicular AF target is at the same distance). The following method was used to acquire the *perpendicular* data set:

1. Rotate the camera (mounted on a 3-way tripod head) so that the AF reticule is nicely centred on the checkerboard section of the perpendicular target.
2. Manually turn the lens to focus at infinity.
3. Press and hold the AF-ON button (i.e., engage AF) until focus lock is achieved. Release AF-ON button. Only one continuous AF operation is used.
4. Now rotate the camera so that the AF reticule is centered on the reference edge of the chart behind the perpendicular target. This is done to ensure that spatially variable lens properties (MTF, distortion) are comparable to the alternative set of images (described below).
5. Using mirror lock-up, capture a shot, and repeat from step 1.

The 45° degree (alternative) data set is collected as follows:

1. Point the central AF sensor at the reference edge of the chart.
2. Manually turn the lens to focus at infinity.
3. Perform one AF operation.
4. Capture the shot using mirror lock-up, and repeat from step 2.

The same camera settings were used for both data sets. This includes setting the camera to AF-S mode (Nikon jargon for single-servo AF) using only the centre AF sensor (definitely a cross-type sensor on the D7000). A critically important detail is that sufficient light was available throughout the experiment. I have yet to repeat the experiment to find a tight lower bound on the required amount of available light, but for this experiment I was able to use about 10–11 EV (ISO 100 F/1.8 at 1/320 s). I can also state that 7 EV (ISO 100 F/1.8 at 1/40 s) is definitely not enough light for the D7000 — the photos look fine, of course, but the AF performance was definitely unreliable (not repeatable). Note that this is a lot of light — definitely not representative of the typical shots that I take indoors with available light. The Nikon AF sensors are usually rated to work from  $\approx 1$  EV up to 19 EV, but I guess that does not mean the accuracy or repeatability is constant over that range.

Next, the images from the *perpendicular* data set were processed (automatically) to mask out the entire perpendicular target, effectively producing a set of images that look just like the images in the 45° degree data set. Now we have two similar data sets that can be processed

by MTF Mapper — the only systematic difference between the two data sets is the way in which the AF lock was achieved. The MTF profiles of 10 shots from each data set are shown in Figure 11.

Visually, it is clear that both AF targets produced similar results, even though quite a bit of variability in focus position is evident. The actual range of the variability in the focus plane distance was about 45 mm at a subject distance of 1.8 m. MTF Mapper computes a value known as the *peak shift*, which denotes the difference (in pixels) between the position of sharpest focus (the peak of the green curve on normal profile plots such as Figure 4) and the position of the reference edge (the assumed point of focus). The *peak shift* values were computed for both data sets, and compared using Wilcoxon’s non-parametric test to determine whether the means of the *peak shift* of the two AF methods are statistically equal. For this data set, the p-value of the Wilcoxon test (with N=10 for each AF method) was 0.393, meaning that we cannot reject the null-hypothesis that the means are equal. In other words, the two AF methods do not produce different results, *which means that the solid block reference edge AF target used in the MTF Mapper perspective test chart works perfectly well.*

## C Internal workings

Here is a brief outline of how MTF Mapper works:

1. The input image is read in, and converted to grayscale (using OpenCV’s default colour matrix). If the input image is an 8-bit image, the pixel intensities are scaled up to 16 bits; the default is to assume that 8-bit images are sRGB gamma corrected, so the values are linearised while being scaled to 16 bits. If the input image has a pixel bit depth of 16 bits, no gamma conversion or scaling is applied.
2. The image is thresholded using Bradley’s adaptive thresholding algorithm [2]. This is done to identify all the dark objects in the image.
3. The thresholded image is then scanned to extract all the connected components using the method of Chang *et al.*[3]. This step gathers the boundary lists of all the dark objects in the image.
4. The image gradient is computed on the original input image. This is combined with the object boundary lists to identify all roughly-rectangular objects (called blocks in the sequel).
5. For each edge of each detected block, the MTF50 value is computed as follows (roughly the method of Khom [1]):
  - (a) Define a rectangular buffer region that is aligned with and centered over each edge. The width of this buffer is 32 pixels.
  - (b) Compute a reasonable estimate of the orientation of the edge (using the image gradient information).
  - (c) Extract the intensity values of each pixel within the rectangular buffer, and project the coordinates of this pixel onto the direction normal to the edge.
  - (d) Systematically refine the edge normal to minimise the difference between sequential projected values; this effectively optimises the edge normal estimate.

- (e) For each pixel in the buffer, record a pair of values (**distance**, **intensity**), where **distance** denotes the length of the pixel coordinates projected onto the edge normal, and **intensity** represents the actual pixel intensity value. Note that the distance values are unevenly sampled. These values are a representation of the edge spread function (ESF).
  - (f) Binning to resample the (**distance**, **intensity**) values to a regular grid. The resampled points are generated at a spacing of  $1/8$  pixels, i.e., the profile is oversampled at a factor of 8.
  - (g) Apodization is performed by windowing the resampled LSF with a Hamming window.
  - (h) An FFT is computed on the resampled points, and the normalised FFT magnitude sequence is calculated.
  - (i) A systematic correction is applied to counter the distortion introduced by the finite-difference derivative operations used to compute the PSF.
  - (j) The frequency at which the FFT magnitude sequence reaches a value of 0.5 is computed using linear interpolation, yielding the MTF50 value.
6. The computed MTF50 values are then rendered in various ways, depending on which output options are selected.

## D Accuracy

The accuracy of the MTF50 estimates produced by MTF Mapper depends on the following factors:

1. Image resolution, and resulting edge length
2. Image noise levels
3. Image edge contrast (combined with noise levels, expressed as Signal-to-Noise Ratio, SNR)
4. Edge orientation relative to horizontal or vertical directions
5. Edge MTF50 value — the sharper the edge, the more critical the above factors become

Figures 12–13 present an indication of the error in MTF50 estimates. The following parameters applied:

1. Edge length was fixed at 75 pixels.
2. Two noise levels were investigated: edge contrast at 0.9 times dynamic range, with noise standard deviation at 0.01 times dynamic range, for an SNR of 90, and edge contrast at 0.67 with noise at 0.012, for an SNR of 57. The latter case approximates closely the real world conditions observed with a DSLR camera.
3. Edge MTF50 values over the range [0.06,0.7) were sampled.
4. Edge orientations of  $2^\circ$  to  $44^\circ$  were explored.

5. A total 200 data points were collected for each MTF50/angle combination.

Figures 12–13 clearly illustrate how the error increases with higher edge acuity, especially at slant angles that are further from ideal. The boxplots are interpreted as follows: the “whiskers” indicate the extreme values observed, although outliers may be excluded and shown separately as circles; the solid black line in the centre of each box is the median of the 200 data points, with the red box bounding the 50% middle-most values. If this data is expressed as a relative value, i.e., MTF50 error over MTF50 value, then the standard deviation of the relative error is roughly below 1%, with the extremes hovering around 5%, up to the Nyquist limit at 0.5 c/p, for the SNR=57 case. The 95% confidence interval width on the absolute errors are given in Figures 14 and 15.

Sharp lenses should produce values in the range of (0.2,0.35) cycles per pixel using unsharpened raw images. Overall, both absolute accuracy (bias) and uncertainty (variance) remain fairly well controlled up to the Nyquist frequency of 0.5 cycles per pixel, but some effort should be expended to keep the SNR high (i.e., limit image noise) if high accuracy is desired.

**Warning:** The slanted edge method does not work at all if the edge acuity is very high (MTF50 of 0.5 and higher) if your edges are perfectly aligned with the rows or columns of the image. That is why it is called the *slanted edge* method . . . . Anyway, MTF mapper does not currently protect you from this case, and will happily try to measure the MTF50 across any edge you throw at it. If the MTF50 value exceeds 1.0 c/p, the program will silently ignore that edge, but poorly oriented edges may produce very high MTF50 values that are still below 1.0, i.e., a sharp edge at an angle of  $0.5^\circ$  may very well produce an MTF50 value of 0.86.

It is recommended that you ensure that your edges are between  $2^\circ$  and  $44^\circ$  degrees with respect to the horizontal or vertical directions. Some intermediate angles may produce very inaccurate results, e.g.,  $26.56^\circ$  and  $14^\circ$  angles are to be avoided at all costs (those two angles interfere with the binning algorithm, thus no oversampling is achieved). Future versions of MTF Mapper may include options to filter out these cases automatically, but currently you are responsible for keeping edge orientations reasonable.

## E User manuals for individual applications

(Rest of this page left blank intentionally)

## 1 mtf\_mapper

mtf\_mapper — compute MTF50 edge sharpness measure in images

### Synopsis

**mtf\_mapper** [*OPTIONS*] *INPUT\_IMAGE OUTPUT\_DIR*

### DESCRIPTION

**mtf\_mapper** computes the edge acuity (sharpness) of slanted edges in images. It automatically detects dark rectangular objects on light backgrounds, and computes the MTF50 values across each of the edges. Output takes several forms (see **-p**, **-s**, **-a** and **-q** output options). To test **mtf\_mapper**, images with rectangles containing known MTF50 values can be generated with **mtf\_generate\_rectangle**.

### OPTIONS

**-t threshold, --threshold threshold**

Specify the dark object threshold, with a default of 0.55. Lower values are required if your dark objects are light relative to the background, e.g., gray rectangles rather than black rectangles. You can try lower values, e.g., 0.3 or even 0.2 if MTF Mapper does not appear to detect any dark objects.

**--threshold-window fraction**

Specify the fraction of the smaller image dimension (height or width) that will be used as a window size during automatic thresholding to detect dark objects (see **-t**). The default of 0.3333 works well when the test chart has a white background. If your dark targets tend to have some other chart feature surrounding them (think QA-62 Scanner targets) then you can try to reduce this fraction; choosing the fraction to be roughly equal to the size of the dark target (as a fraction of image height) should work.

**-l, --linear**

Linear input mode; assumes that an 8-bit input image has a linear intensity scale. The default is to assume that 8-bit input images have an sRGB gamma intensity profile (approximately gamma 2.2).

**--pixelsize size**

Specify the sensor's pixel size (pitch) in microns. This option implicitly switches the MTF50 output units (for some output types) to line pairs per mm, or lp/mm (the default is cycles per pixel, or c/p).

**--bayer red|green|blue|none**

Process only the specified Bayer sites. This option can be used to bypass the effects of Bayer demosaicing interpolation when suitable raw images (e.g., dcrw -d output) are used. Specifying this option when a demosaiced image is provided will not produce the expected result, i.e., you must provide a raw image for this option to work correctly. Keep in mind that Bayer red and blue each cover only 25% of the sensor, so your edges will have to be 4 times longer to maintain the same signal-to-noise ratio. Minimum recommended edge lengths are thus 35 pixels for gray or interpolated images, 70 pixels for green Bayer sites, and 140 pixels for red and blue Bayer sites. Aim for edges of at least 200 pixels for best results on red and blue sites. See **--cfa-pattern** so specify the Bayer pattern of your raw image.

**--cfa-pattern rggblbggr|grbg|gbrg**

Select the Bayer pattern to use when the **--bayer** option has been specified. The default is rggbl, which appears to be the most popular choice amongst DSLRs.

**--esf-model kernel|loess**

Choose the algorithm that MTF Mapper uses to construct the Edge Spread Function (ESF) with. The 'loess' algorithm is recommended, unless you are trying to produce results that are compatible (ahem, 'roughly similar' is more like it) with older versions of MTF Mapper (pre- version 0.7.16).

### **--single-roi**

Treat the entire input image as the region of interest (ROI). This option is only intended for use with small cropped images containing only a single edge, typically if you cropped your ROI out of some larger image. Use this if you are performing your slanted-edge measurements with a backlit razor blade, or if you are working with an incompatible test chart (e.g., an older ISO 12233 chart). This option has largely superseded the **-b** option.

### **--zscale *scale-factor***

Adjust the minimum value of the z-axis scale of the 3D plots produced with the **--surface** output option. A value of 0 means the z-axis scale starts at zero, and 1.0 means the z-axis starts from minimum MTF50 measurement (thus emphasizing local differences).

### **--logfile *filename***

Logger output written to *filename* in stead of standard out.

### **--gnuplot-width *pixels***

Width of images rendered by gnuplot, typically affecting the output images of **--lensprofile**, **-s**, and **-p**.

### **-b, --border**

Add a white border of 100 pixels to the image. This option might be useful if your image contains only a single black target (e.g., rectangle) with a thin white border, or if your image sides clips some of your black test chart targets. Actually, this option is a kludge to fool MTF Mapper's automatic target detection, so you should not normally need this. Also see **--single-roi** for the correct way of dealing with single-edge images.

### **--snap-angle *angle***

Snap all edge angles to *angle*. Angles are snapped to the closest value modulo 90 degrees, i.e., specifying an angle of 4 degrees will force edge orientations to one of the following: 4, -4, 86, or 94 degrees. This option should be used with care, and is only appropriate if you are using synthetic images with a known edge orientation.

### **-g *angle*, --angle *angle***

Only report MTF50 values on edges with an orientation of *angle* degrees in raw output mode (**-r**)

### **--autocrop**

Automatically crop the input image to the chart area. The chart is assumed to be brighter than the background; the automatic cropping will try to remove the darker background. This option is mostly intended to speed up processing, and really should only be used if the background area is large in comparison to the test chart area.

### **--imatest-chart**

Automatically crop the input image so that the black bars at the top and bottom of Imatest-style charts (e.g., SFRplus) are suppressed, thus allowing full automatic processing of all of the square targets found in the chart. You may want to use a lower threshold value (**-t**) of 0.4 or even lower when using **--imatest-chart** to ensure that all the square targets are detected. This option should not be used with native MTF Mapper test charts.

### **--mtf *contrast***

By default MTF Mapper computes MTF50, i.e., the resolution (in lp/mm or cycles/pixel) at which the SFR curve first reaches a contrast of 50%. This option allows you to change the target contrast value to compute MTF20, for example. Valid *contrast* values are in the range [10, 90], and will be clamped to this range if necessary. This option affects all outputs, including the Annotated image, Profile, Grid and Focus position outputs; only the SFR outputs are unaffected.

### **--gnuplot-executable *filepath***

Specify the full path to the gnuplot executable. Defaults to `/usr/bin/gnuplot`, which is usually correct on most Linux distributions

### **-h**

Displays usage information

## OUTPUT TYPE RELATED OPTIONS

### **-a, --annotate**

Annotated output mode. If Annotate mode is enabled, **mtf\_mapper** produces an output file called *annotated.png* wherein each edge is annotated with its MTF50 value. Good quality edges are annotated in Cyan, with Yellow and Red annotation indicating progressively poorer edge quality (usually related to edge orientation and length).

## 1 mtf\_generate\_rectangle

mtf\_generate\_rectangle — generate images containing edges with known MTF50 values

### Synopsis

**mtf\_generate\_rectangle** [*OPTIONS*]

### DESCRIPTION

**mtf\_generate\_rectangle** renders an image (default `rect.png`) containing a rectangle rotated through the specified angle; other polygonal objects can also be rendered. The image is generated with a specified type of point spread function (PSF), which can either be a simple Gaussian, or a diffraction pattern with an optional optical low-pass filter. Various parameters can be adjusted to produce the desired size / shape of the PSF. Sensor noise is simulated using either a simple additive Gaussian model, or a more sophisticated sensor noise model. The program reports the analytical MTF50 value of the rendered edges. **mtf\_generate\_rectangle** can be used to evaluate the accuracy of the **mtf\_mapper** tool, since the accuracy of the images generated with **mtf\_generate\_rectangle** have been validated by third parties.

### OPTIONS

- b16**  
Render output image as a linear intensity 16-bit grayscale image (default is 8 bits gamma corrected)
- l**  
Render output image as a linear grayscale image (default is sRGB gamma)
- d pixels**  
Dimension of the long edge of the rectangle, in pixels. Please note that MTF Mapper may produce inaccurate results if this parameter is below 35 pixels.
- x pixels**  
Sub-pixel offset applied to rendered rectangle, x-axis direction.
- y pixels**  
Sub-pixel offset applied to rendered rectangle, y-axis direction.
- b std. dev**  
Specify the standard deviation of the Gaussian point spread function (PSF) with which the edges are sampled. This parameter effectively controls the resulting MTF50 value of the edges. Mutually exclusive with **-m**.
- m mtf50**  
Directly specify the desired MTF50 value. This indirectly controls the Gaussian PSF standard deviation. Mutually exclusive with **-b**.
- a angle**  
Specify the angle (in degrees) at which the rectangle is generated. Useful for characterising MTF measurement accuracy as a function of edge orientation
- r ratio**  
Specify the aspect ratio of the rectangle in the range [0,1]. Ratios above 1.0 are clipped. The aspect ratio is adjusted so that the rectangle is at least one pixel wide. Very thin rectangles (< 25 pixels wide) are not ideal for use with MTF Mapper.
- esf-only**  
Render only a cross-section across the edge of the rectangle at a 1/32th pixel spacing. This is mainly useful for performing synthetic image accuracy measurements, but can be used to generate Point Spread Functions (PSFs) for the *box* and *4dot-olpf* Airy pattern variants (these PSFs are hard to compute analytically). Default output filename is *profile.txt*, but can be specified with **-o**.

**-o filename**

Specify the name of the output file

**-h**

Displays usage information

## NOISE OPTIONS

**-n std. dev**

Specify the noise standard deviation [0-1] of the Gaussian noise that is added to the image. This invokes the simplified noise model (which is also the default), which produces noise that is independent of signal level. A more accurate sensor noise model can be obtained with the full sensor noise model (**--read-noise**, etc.); See examples below.

**--adc-depth bits**

Number of bits used in full noise model ADC. This parameter effectively scales photon shot noise. Recommended value is 12.

**--adc-gain electrons**

Number of electrons per DN, i.e., the ADC gain of the full noise model. Recommended value is 2.64.

**--pattern-noise fraction**

Magnitude of Pixel Response Non-Uniformity (PRNU) component of full noise model. Recommended value is 0.0085.

**--read-noise std. dev**

Standard deviation of read noise component of full noise model, in electrons. Recommended value is 3.7.

**-c factor**

Reduce edge contrast by *factor*, i.e., if *factor* is 0.3, then the edge contrast is  $1 - 0.3 = 0.7$ .

**-s seed**

Specify the random seed for the RNG used to generate the additive Gaussian noise.

## PSF/RENDERING OPTIONS

**-p gaussian|gaussian-sampled|airy|airy-box|airy-4dot-olpf|wavefront|wavefront-box**

Specify the desired PSF type. PSF types *gaussian* and *gaussian-sampled* accept the **-m** and **-b** arguments to control PSF shape, and produce, unsurprisingly, Gaussian-shaped PSFs. Note that these PSFs are pure Gaussians, i.e., they have not been convolved with the photosite aperture, and are therefore not (directly) useful for sensor simulations.

The remaining PSFs, namely *airy*, *airy-box*, *airy-4dot-olpf*, *wavefront* and *wavefront-box* simulate an Airy pattern PSF, such as would be observed when diffraction through a circular aperture is considered. These accept the **--lambda**, **--pixel-pitch** and **--aperture** arguments to control the diffraction pattern. The *airy* PSF models only diffraction, and does not include convolution with a photosite aperture. In contrast, the *airy-box* PSF includes this final convolution step, and is thus a useful PSF for simulating lens + sensor combinations that do not include an Optical Low-Pass Filter (OLPF), also commonly called an Anti-Aliasing (AA) filter.

The *wavefront* PSF is similar to the *airy* PSF, and the *wavefront-box* PSF is similar to the *airy-box* PSF; the main difference being that the wavefront-PSFs can model the effects of defocus and spherical aberration in addition to aperture diffraction. The **--w020** option controls the amount of defocus, and the **--w040** option controls the amount of spherical aberration.

Lastly, the *airy-4dot-olpf* PSF simulates a lens + OLPF + sensor combination, assuming that the AA filter is a 4-way beam splitter (e.g., Lithium Niobate birefringent designs) with a displacement of 0.35 pixels, which appears to match current Nikon DSLR sensor designs.

**--airy-samples n**

Number of half-samples (n) per axis per pixel for Airy PSFs (actual number of samples is  $(2n+1)^2$ ).

**--olpf-offset pixels**

4-dot OLPF beam splitter offset, default value is 0.35.

### **--lambda** *micron*

Specify the wavelength of the light to simulate when rendering diffraction PSFs, in micron.

### **--pixel-pitch** *micron*

Specify the photosite pitch of the sensor. This is required to convert the wavelength parameter into a pixel-scale value.

### **--aperture** *f-number*

Specify the relative aperture (f-number) of the lens used to render a diffraction PSF.

### **--w020** *magnitude*

Specify the magnitude of defocus applied to the *wavefront* and *wavefront-box* PSF models. The *magnitude* is expressed as a multiple of the wavelength, meaning **--w020 1.0** results in a defocus wavefront distortion with a magnitude of one lambda (0.55 micron by default) at the edge of the simulated exit pupil. Note that changing the simulated lens aperture will change the size of the simulated exit pupil, and will therefore affect the amount of simulated defocus.

The recommended range for the defocus magnitude is [-20, 20]; values outside this range may lead to unacceptable truncation of the simulated PSF.

### **--w040** *magnitude*

Specify the magnitude of spherical aberration applied to the *wavefront* and *wavefront-box* PSF models. The *magnitude* is expressed as a multiple of the wavelength, meaning **--w040 1.0** results in a spherical aberration wavefront distortion with a magnitude of one lambda (0.55 micron by default) at the edge of the simulated exit pupil. Note that changing the simulated lens aperture will change the size of the simulated exit pupil, and will therefore affect the amount of simulated spherical aberration.

The recommended range for the spherical aberration magnitude is [-20, 20]; values outside this range may lead to unacceptable truncation of the simulated PSF.

### **--target-poly** *filename*

Target polygon data file. This option overrides the default target (a rectangle) with the geometry of the polygons specified in *filename*. The format of this file is just

```
_number_of_vertices_  
_x1_ _y1_  
_x2_ _y2_  
...  
...
```

which can be repeated for additional polygons. The polygons should be simple, but may be concave, and should have fewer than 400 vertices. The default ROI will range from (0,0) up to the extent of the vertices; see the **--roi-** options to override this behaviour.

### **--photosite-geom** *square|circle|rounded-square*

Photosite aperture geometry, default is *square*. The *circle* and *rounded-square* are 64-sided polygonal approximations. If you require a more general aperture geometry, use the **--photosite-poly** option.

### **--photosite-poly** *filename*

Photosite aperture polygon file name. The origin is (0,0), with a 100% fill-factor square pixel defined as the region (-0.5, -0.5) to (0.5, 0.5). The format of *filename* is the same as that of **--target-poly**.

### **--roi-height** *pixels*, **--roi-width** *pixels*, **--roi-col** *pixels*, **--roi-row** *pixels*

Specify the Region Of Interest (ROI) that should be rendered, the region (row, col) to (row + height, col + width). This is useful for generating a fixed-size output image, or just rendering a subset of a larger geometry (see **--target-poly**).

### **--fill-factor** *factor*

Fill-factor of photosite [0.01,1], default is 1.0.

## EXAMPLES AND NOTES

This utility supports both a simple Gaussian noise model, and a signal-dependent noise model that models sensor noise more accurately. If you just need *some* noise, then the simple model (specified with the **-n** parameter) will be sufficient. This method can also be used to generate noise-free images.

## 1 mtf\_generate\_test\_chart

mtf\_generate\_test\_chart — generate test charts for evaluating lens sharpness

### Synopsis

**mtf\_generate\_test\_chart** [*OPTIONS*]

### DESCRIPTION

**mtf\_generate\_test\_chart** generates scalable vector graphics (SVG) files containing test charts for evaluating digital camera lens performance (sharpness) as well as autofocus accuracy / fine-tuning. **mtf\_generate\_test\_chart** should be used to generate suitable test charts for use with the **mtf\_mapper** tool.

### OPTIONS

**-o filename**

Specify output file name (default is *chart.svg*).

**-s a4|A4|a3|A3|a2|A2|a1|A1|a0|A0**

Specify page size of the chart.

**-t perspective|grid|halfgrid|thirdgrid|lensgrid|mperspective|focus**

Specify chart type. The *perspective* chart is for evaluating autofocus behaviour, and is intended to be photographed at an angle, typically 45 degrees.

The *lensgrid* chart is intended to evaluate lens sharpness across the focal plane, and should be photographed perpendicular. This chart type is suitable for many **mtf\_mapper** output options, including **-a**, **-s**, **-q**, and **--lensprofile**. The circular fiducial markers also allow this chart type to be used for iteratively adjusting the chart orientation (to position it perpendicular to the camera's optical axis) using **mtf\_mapper**'s **--chart-orientation** output mode.

The *grid* chart is similar to the *lensgrid* chart, but it is an older design, and it is strongly recommended that you use the *lensgrid* chart instead. Similar grid-style charts, but with fewer target rectangles, can be generated using the *halfgrid* and *thirdgrid* types.

The *focus* chart is a single-purpose chart type designed to measure the position (distance) of peak focus. This chart must be photographed at an angle (45 degrees recommended), and requires manual focusing. The **--focus** output option of **mtf\_mapper** is specifically intended for this chart type.

The *mperspective* chart is another special chart type designed to visualize the intersection of the "*plane of best focus*" and the test chart, which should be tilted at around 45 degrees. The **--mfprofile** output option of **mtf\_mapper** is specifically intended for this chart type.

**-d distance**

Specify the desired viewing distance from the chart for perspective mode. Distance is in mm, measured from the reference edge on the chart, to the camera focal plane.

**-h**

Displays usage information.

## References

- [1] Kohm, K., Modulation transfer function measurement method and results for the Orbview-3 high resolution imaging satellite, Congress International Society for Photogrammetry and Remote Sensing, 20:12–23, 2004.
- [2] Bradley, D. and Roth, G., Adaptive thresholding using the integral image, *Journal of Graphics, GPU, & Game Tools*, **12(2)**:13–21, 2007.
- [3] Chang, F., Chen, C.J., Lu, C.J., A linear-time component-labeling algorithm using contour tracing technique, *Computer Vision and Image Understanding*, **93(2)**:206–220, 2004

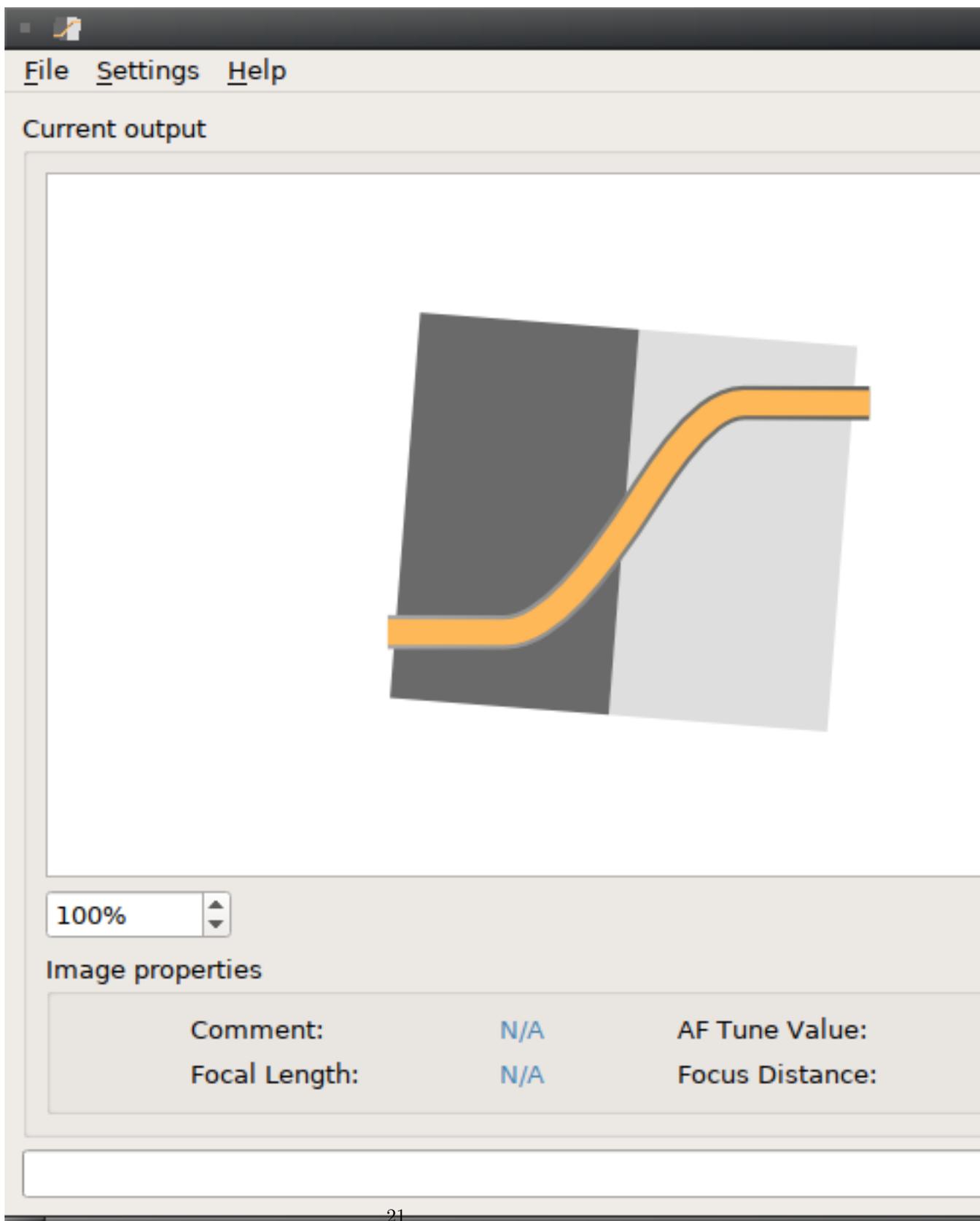
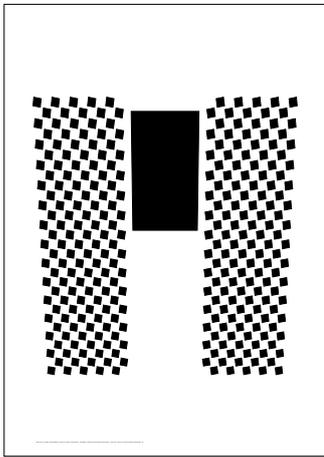
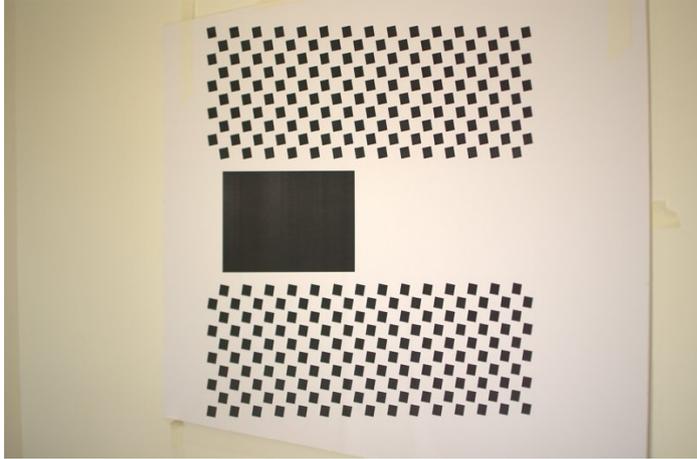


Figure 1: Screen shot of the MTF mapper gui



(a) Perspective chart



(b) A photo of this type of chart at 45°

Figure 2: An illustration of the type of test chart used in *profile* mode.

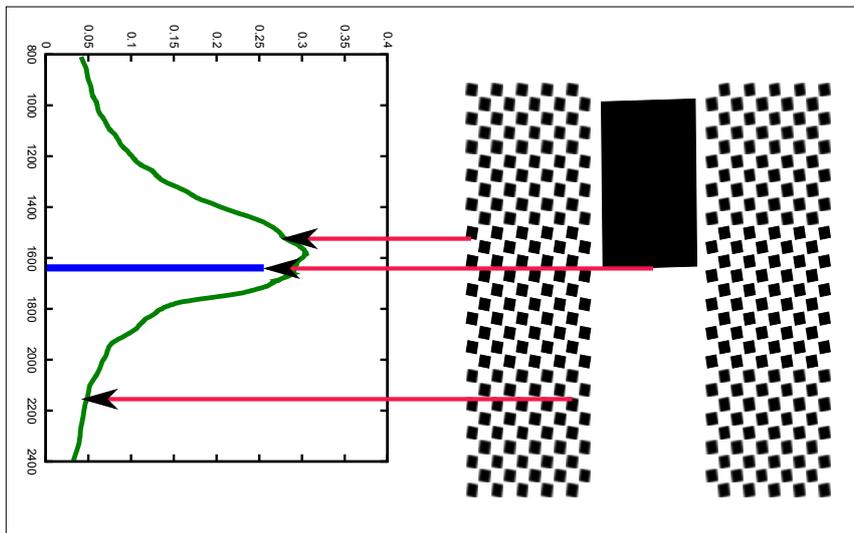


Figure 3: How the profile is constructed: MTF50 values are collapsed horizontally onto the *y*-axis to form the profile.

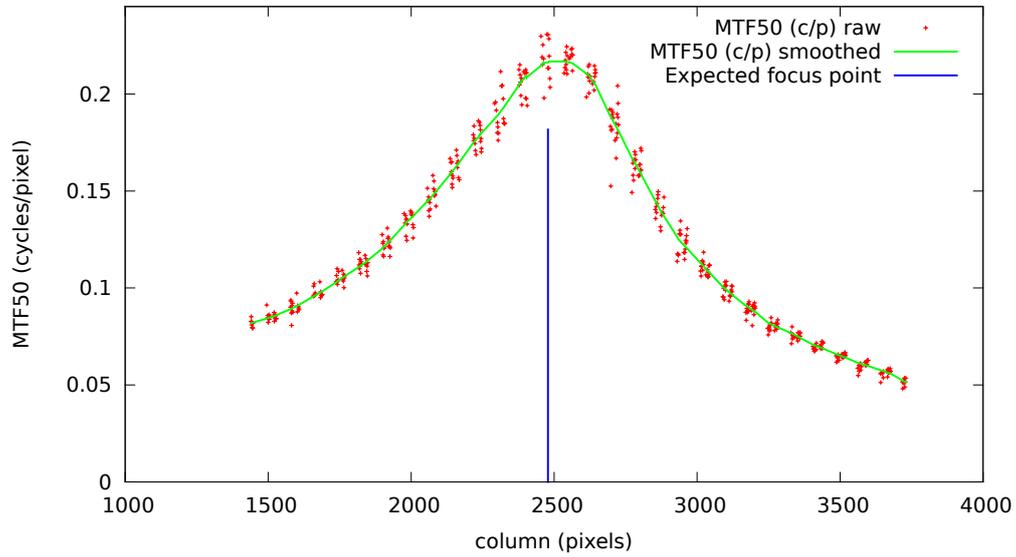


Figure 4: Example of profile generated by MTF mapper. Derived from the image shown in Figure 2(b).

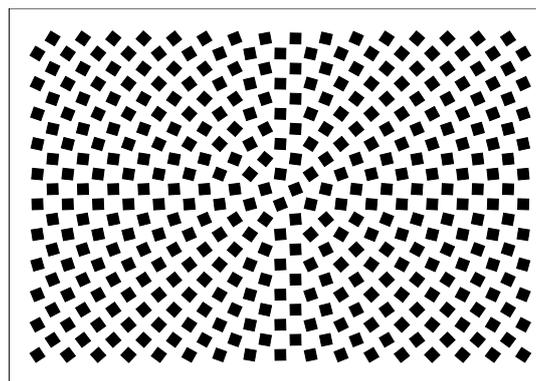


Figure 5: Example of an MTF surface mode test chart

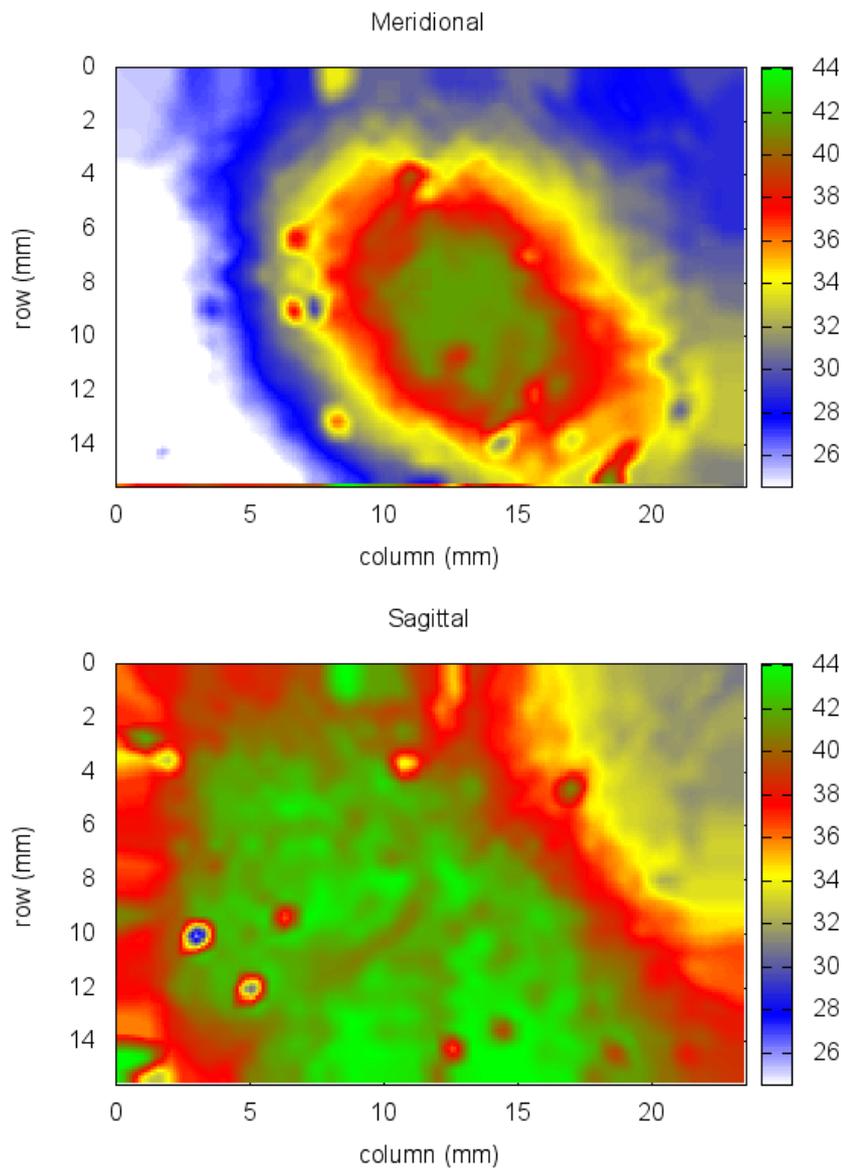


Figure 6: Example of MTF50 image generated by MTF mapper

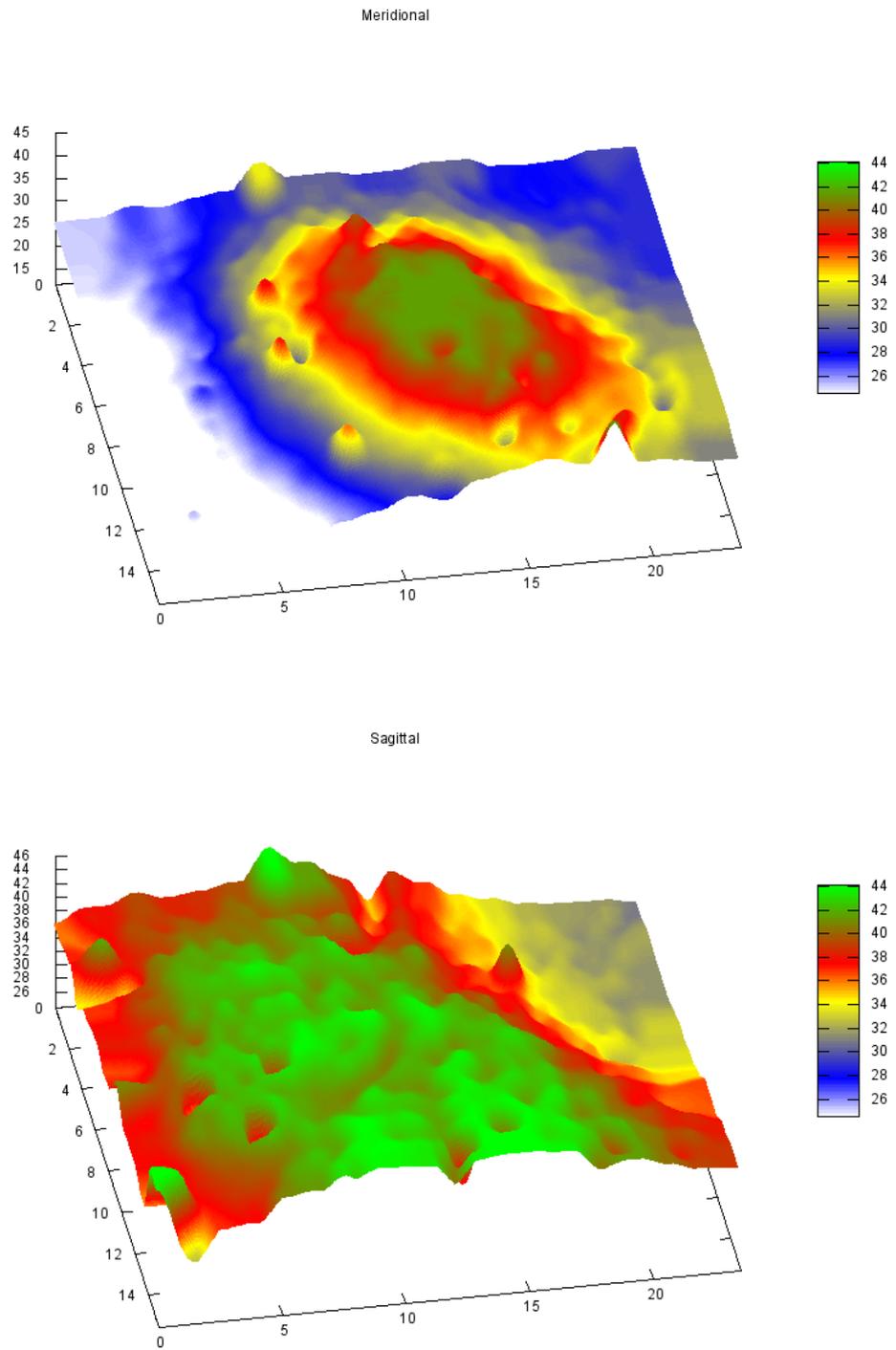


Figure 7: Example of MTF50 surface generated by MTF mapper

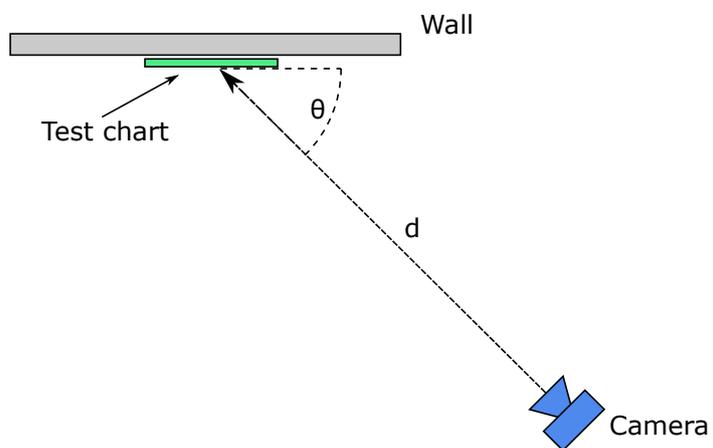


Figure 8: Illustration showing the top view of the autofocus calibration set-up

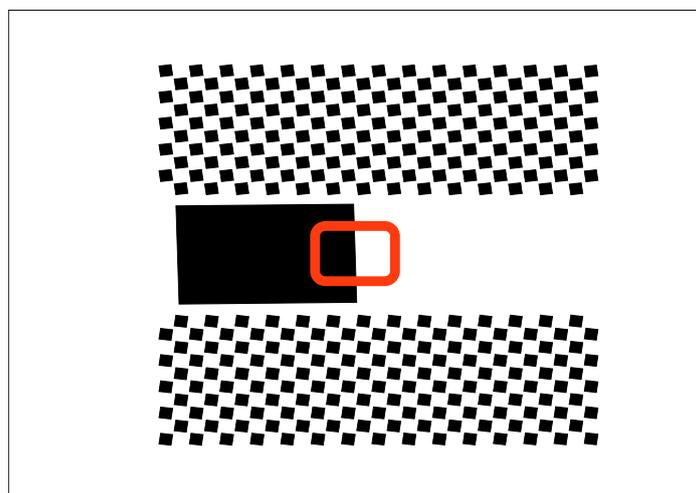


Figure 9: Where to place your autofocus sensor

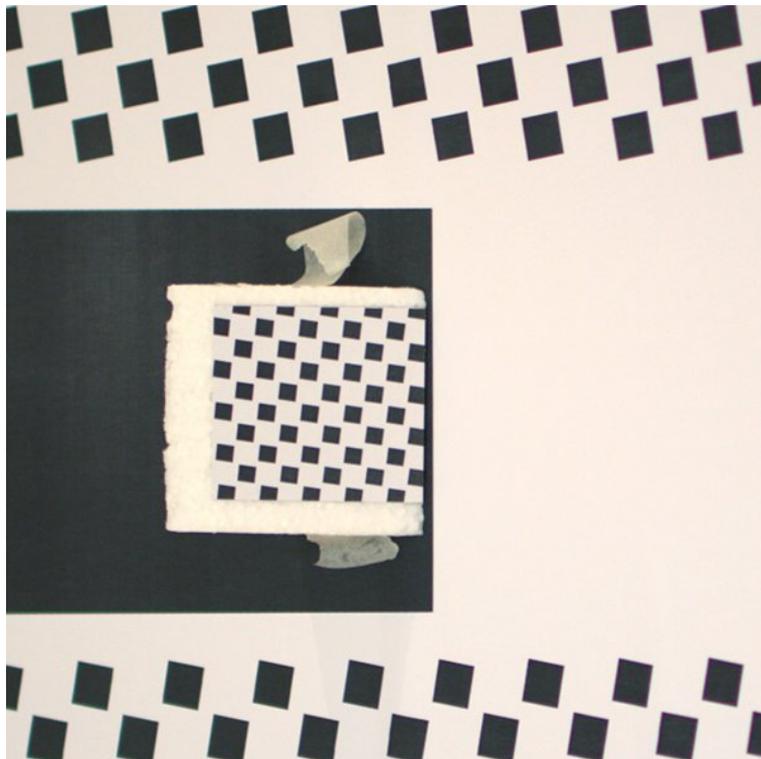


Figure 10: Crop of test chart, showing the perpendicular AF target attached to the flat chart. The checkerboard pattern fixed to the polystyrene block is at a  $45^\circ$  angle with respect to the chart behind it.

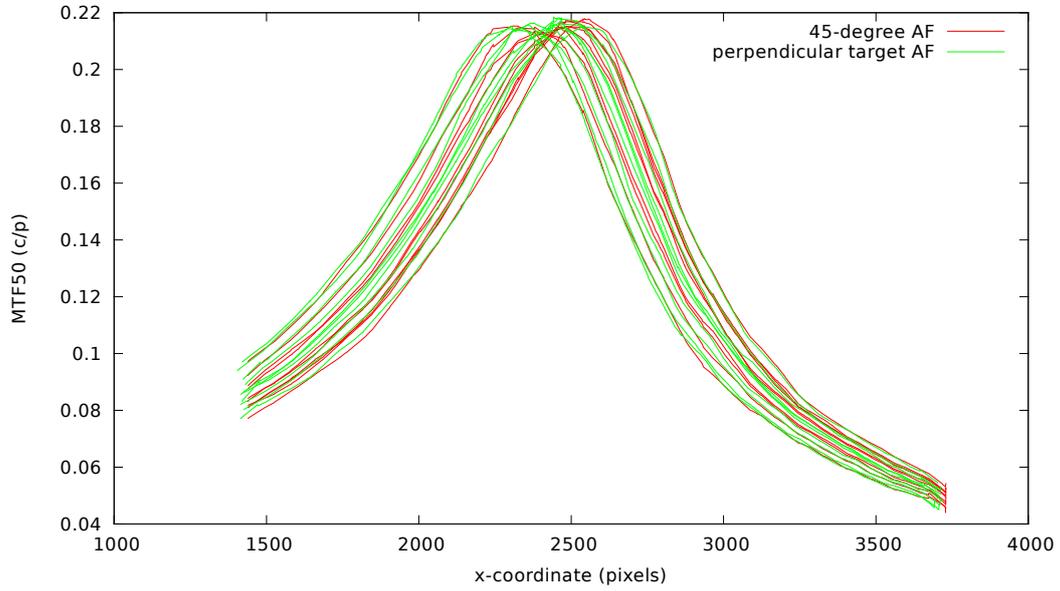


Figure 11: MTF50 profiles of 10 shots using 45° AF on reference edge (red), and 10 shots using the perpendicular AF target of Figure 10 (green).

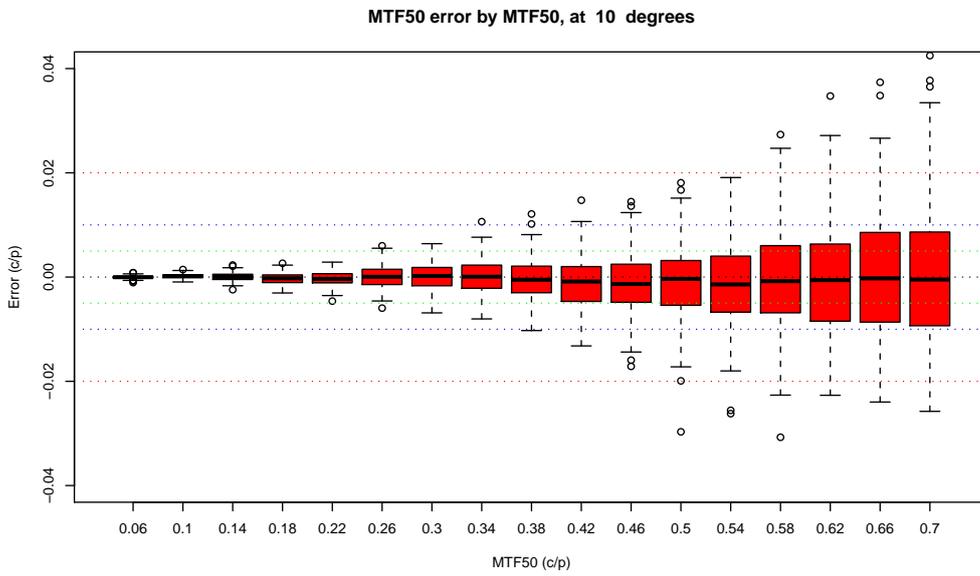


Figure 12: Error in MTF50 estimate for 10° edges, SNR=90

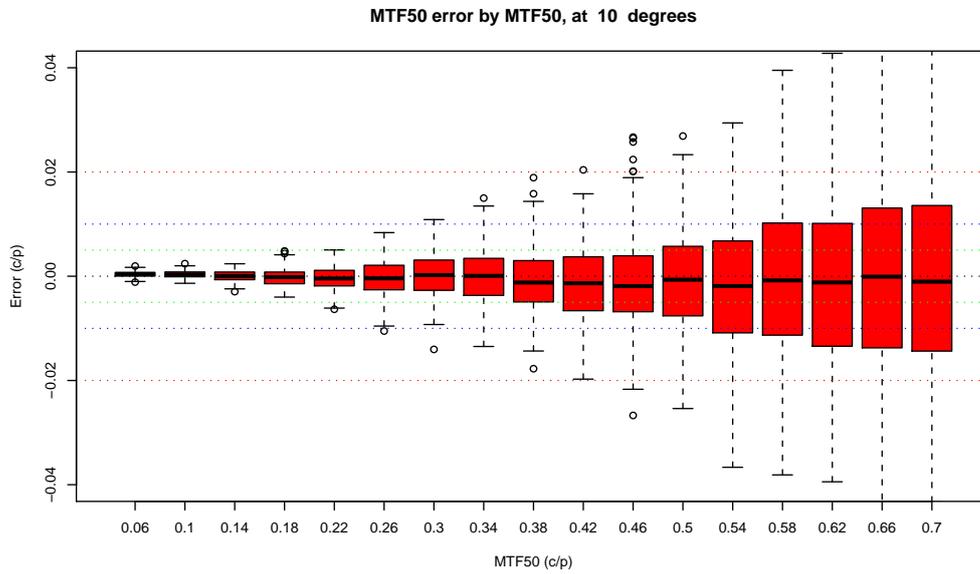


Figure 13: Error in MTF50 estimate for  $10^\circ$  edges, SNR=57

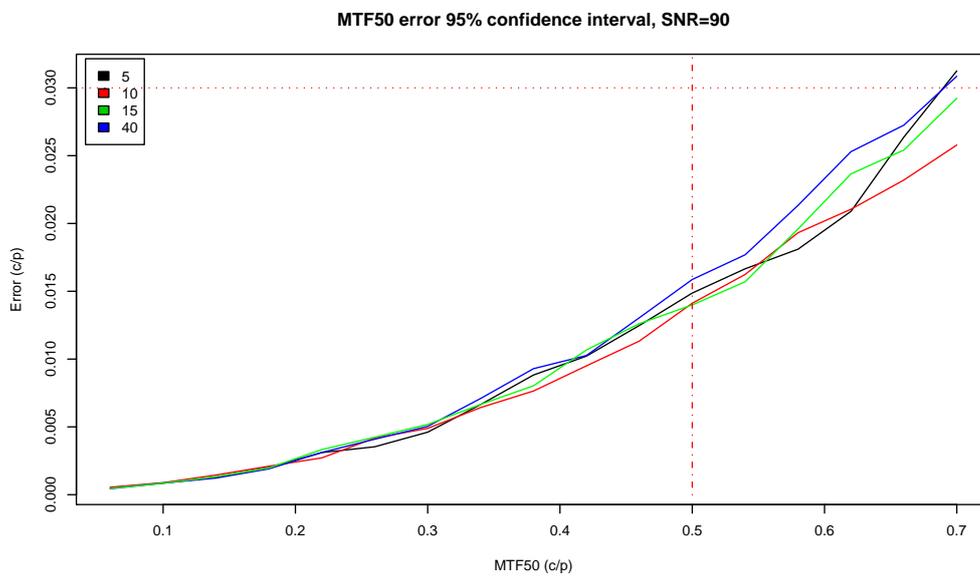


Figure 14: 95% confidence interval for MTF50 estimate at SNR=90

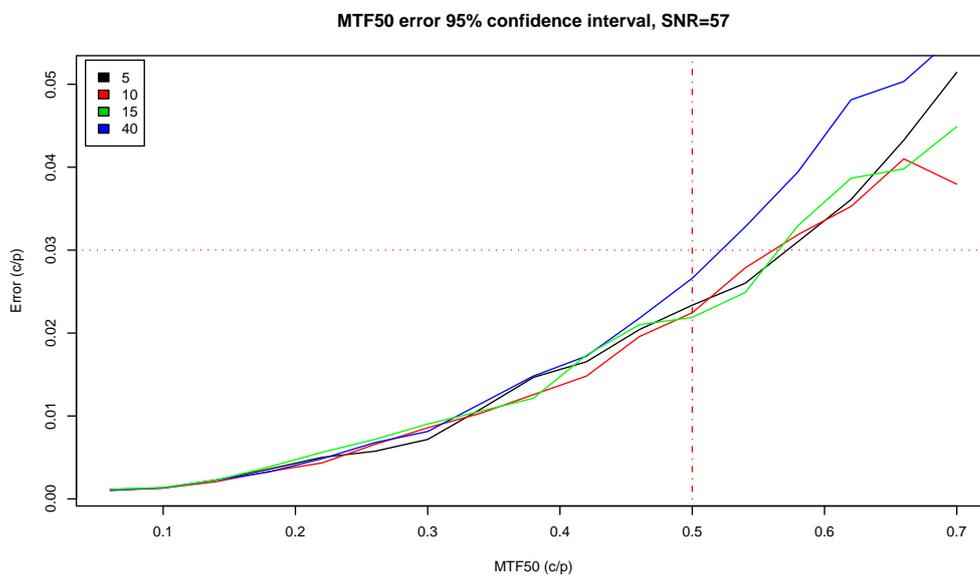


Figure 15: 95% confidence interval for MTF50 estimate at SNR=57