# Check_focus [0.1]

Check_focus is a suite of tools (currently just two) to ease the photographic workflow. It addresses basically one aspect, “*which images are in focus*”. This is not the trivial issue it might seem, as what means for an image to be in focus it not obvious.

*Figure 1: Gone to Seed*

Here is an Photograph I took in the garden (Figure 1 Gone to Seed) The image plays with depth of field (DoF); now I think it’s focused just right but in reality that’s just what I think. As an added complication the area I feel is in focus is red, which causes problems with some camera autofocus and also with the default settings of these tools

Our simplistic definition of focus Take  these two images

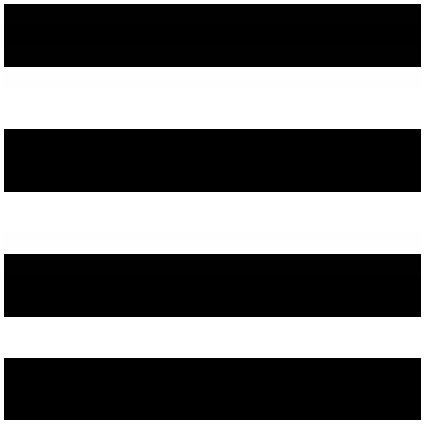


*Figure 2: Horizontal Lines*

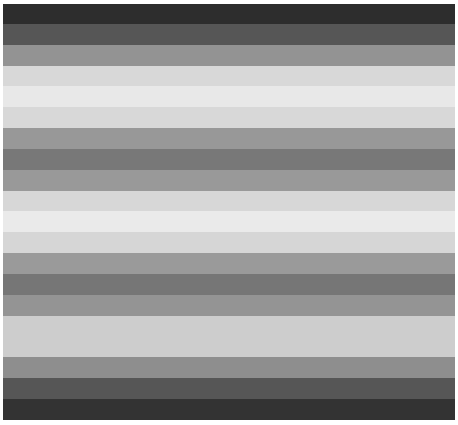


*Figure 3: Blurred Lines*

Generally Figure 2 Horizontal Lines would be seen as the more focused image. If we were to zoom in at the black/white edge, we would see:

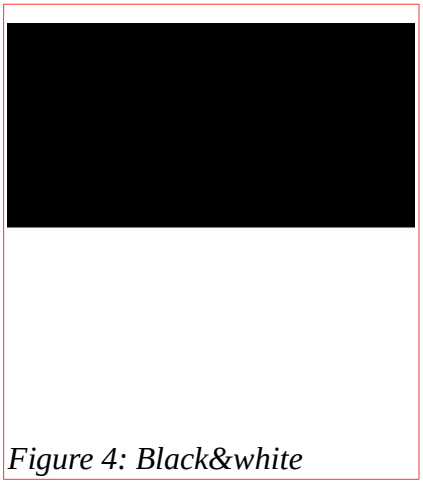


*Figure 4: Black&white*

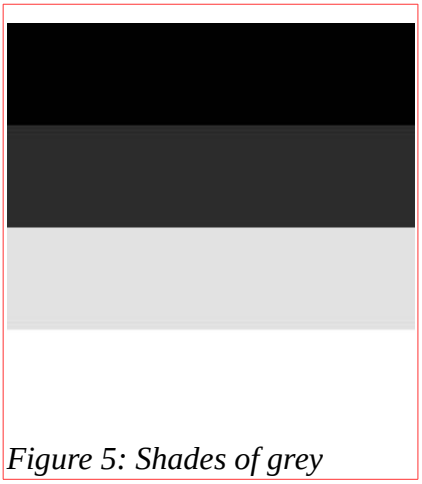


*Figure 5: Shades of grey*

If we assume a value of 0 (zero) is black and 255 is white, going from top to bottom with Figure 4 might be represented as [0,0,255,255] whereas Figure 5 might be [0,64,192,255] so how can we score these. We obviously want to choose that 0 to 255 boundary  Well if we score 0→0 as a score of 0 (zero) the 0→255 gets a score of 255 and  255→255 a score of 0. This looks good . If we apply the same rules to Figure 5 we get 64,128,63 . The 255 beats the 128 so the first image is “better” . This would be fine if we only had to consider one line on the image. However we need to somehow combine the scores of a larger area of the image in order to compare two images. We could simply add them; the 1st image gets a sum of 255 (obviously) the 2nd gets a score of … 255, that’s a

problem. There is also the issue that if we went black, white, black we might get 255 & -255 despite this being a sharp focus we would score 0 (zero)

We need a way to “reward” the great 255 over the OK 128. Turns out there’s a simple solution to both problems (negatives scores and small scores adding up) we simply square each score, so negative scores become positive and bigger scores get boosted. Using this technique our scores become:

[0,65025,0] = 65025 vs [4096,36864,3969] = 44929 , so the 1st image ( Figure 4) gets the higher score. If we were to imagine a very slow black to white transition, 0,2,4,6,...254,255

We would get lots of 2 scores which square to make 4, so if we had 128 such lines we would only end up with a total score of 512 (vs 65025) and a pretty clear winner. This technique is quite close to that used in contrast focusing in digital cameras (this was inspired by .https://www.cardinalpeak.com/blog/detecting-well-focused-images ).

There are some hidden traps:

- Imagine an image containing a chequerboard of single black and white pixels. This would get great scores but to a human viewer would appear as a grey blob/square
- This example assumes focus is based on horizontal lines, the same argument can be made for vertical lines
- Real images are frequently not greyscale but RGB. How should we consider each channel (just add them up?)
- As can be seen in Figure 1 often a “good photo” might only have a small area in focus but it’s the one part we care about. Indeed the photo is “made” by the large out of focus area.

Some of these issues are addressed by generic solutions in these tools. The first point however requires some human input but possibly is more of a theoretical case than a real world situation.

- Scores are calculated between nearest neighbour both vertically and horizontally, plus the two added together
- Scores for RGB images are normally calculated from the **green** channel, however in images such as Figure 1 you can switch to **blue** or **red** channels
    - On my Pentax K3, the camera autofocus has great difficulty focusing on red images (presumably in Live View only?)
- The tools use different techniques to locate/ score a good area of focusing

# The tools in action

```
$ ./checkfocus horitontal20x20.jpg blur_horizontal20x20.jpg
horitontal20x20.jpg      7374337 0 7374337 195075  0 195075
blur_horizontal20x20.jpg  738416 0  738416  15882  0  15882
```

This is a simple 20x20 image of black and white horizontal bars (blurred and unblurred). The numbers are in two groups of 3, the first group is the overall image, the 2nd group is a for a box in the middle that is half the width and height of the image the 3 numbers are:

1. The overall score
2. The Horizontal score (comparing each pixel to its horizontal neighbour)
3. The Vertical score (comparing each pixel to its vertical neighbour)

As can be seen the “sharp” version (without blur) gets a much higher score for both the whole image and the centre box.

The Horizontal score is always zero in this case because on each horizontal line there is no change, it is totally black or totally white (or the same grey) right across the line.

So in this case and for many simple real world cases, such as a landscape, the first number is what matters. Thus we could use it to order our photographs by “sharpness”[1]

```
$ date && checkfocus *.jpg | sort -nr -k2
Mon  9 Aug 19:03:42 BST 2021
date
IMGP5046.jpg 1765189378 1062116196 703073182 26055790 14910346 11145444
IMGP4893.jpg 1686902157 679345864 1007556293 12590278 6724574 5865704
IMGP4979.jpg 1591019506 756344096 834675410 14684180 4875487 9808693
IMG_20210725_131200.jpg 1474561661 573715965 900845696 4171019 702292 3468727
IMGP4978.jpg 1427908294 594412800 833495494 5280267 2360164 2920103
...120 rows elided ...
IMGP5028.jpg 61633429 29421546 32211883 680749 318284 362465
IMGP5030.jpg 59329254 30521142 28808112 392879 169584 223295
IMGP5016.jpg 56897740 27337819 29559921 1634891 479577 1155314

Mon  9 Aug 19:04:21 BST 2021


$ ls -1 *.jpg | wc
    128     128    1686

$ exiftool -ImageHeight  -ImageWidth  IMGP4926.jpg
Image Height                    : 4000
Image Width                     : 6016
```

So it took 39 seconds to process 128 images of 4000x6016 resolution.

So the best image is **IMGP5046.jpg**, which has a vastly better score than the worst **IMGP5016.jpg**.

1 For our specific definition of “sharp” as discussed earlier.

Now these were images of my son's (faked , due to Covid) graduation, so nice bokeh with head in the centre, so maybe the centre is a better guide.

```
$ date && checkfocus *.jpg | sort -nr -k5
Mon  9 Aug 20:10:12 BST 2021
IMGP4934.jpg 666184535 420920329 245264206 74630781 38991860 35638921
IMGP4936.jpg 701627637 443238567 258389070 50278330 26101156 24177174
IMGP4935.jpg 877343294 556702164 320641130 46499568 24593945 21905623
IMGP5053.jpg 1027453157 541680512 485772645 46029330 22821180 23208150
```

*Drawing 1: Unexpected dark image*

Looking at just the centre, I see **IMGP4934.jpg** has risen up the ranks **74630781** vs **26055790** (for **IMGP5046.jpg**). An interesting image. It is indeed very sharp in the centre but I'd dismissed it in my initial sort as it was very dark [see later WRT version 0.3]. I do have the .DNG (Raw) files so it may be worth the effort reprocessing that negative to get a sharp + well exposed image.

```
$ bestfocus -V 12 -H 12 IMGP4934.jpg IMGP5046.jpg
IMGP4934.jpg 65537716 (2755:1831-3255:2163)
IMGP5046.jpg 44494377 (3006:3663-3506:3995)
```

Using **bestfocus(1)** with a grid of 12X12 frames (and boxes) says **IMGP4934.jpg** still has a better "score" and it's for a smaller box pretty close to the centre. The image **IMGP5046.jpg** has its best "box" (same size) near to bottom in the middle. BTW it's worth noting the K3 always produces images in landscape format; it simply adds a tag to say how it should be rotated, so these numbers are always for the landscape image.

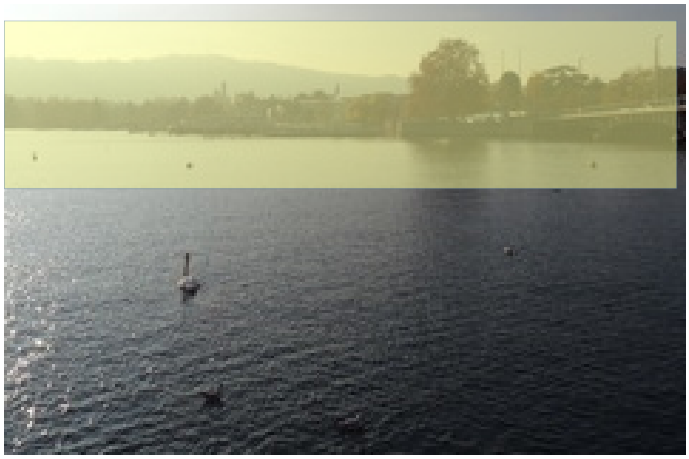
It's also worth noting the size of the numbers. A bigger box/image will produce a bigger number so only compare numbers for the same size box/image. Also the absolute size of these numbers. A perfect black white transition gets a score of 255x255= 65025 , in both horizontal & vertical so 130050 . Thus an image of 6000x4000 could produce a number like 3.12E12 something like 3,120,000,000,000) the maximum number the tool can process on my 64 bit Linux system is 18,446,744,073,709,551,615 , so room for bigger images to come. However check your other tools (like sort) can cope [see version 0.2].

## The tools in theory

Two tools one job, seems overkill. They do have different goals:

- **checkfocus(1)** is a much faster tool. It aims to give an overall “score” for the image plus another score for a smaller “box” by default the box is the “middle bit” but if you know you have a series of shot with say a deliberate out of focus top section you can change the size & shape of the box
- **bestfocus(1)** is a slower and more intense tool. This attempts to find the best “box” of focus on the image. The idea here is you may have something you what in sharp focus in front of a background, e.g. a rapidly moving car on a circuit, a close up of an insect on out of focus foliage, or just rapid candid shots where there was no time to compose.

*Drawing 2: default checkfocus*



*Drawing 3: Custom checkfocus*

So **checkfocus(1)** default will produce scores for the left hand image Drawing 2 (overall and yellow box) but you can use options to specify the box if a different size and shape.

You may however not know where the “good bit” is in an image, sure you can look, but this is meant to help our workflow by quickly sorting the images, The bigger gun is **bestfocus(1)** this works differently but follows the same rules for what constitutes good focus.

**Bestfocus(1)** first divides the image into X columns and Y rows (so X*Y frames) e.g. 10x10 = 100 boxes  Then a second set of alternates  offset midway, so each alternate frame overlaps 4 primary frames. Then in each frame it defines a box. It then scans the image looking for the box with the best focus.

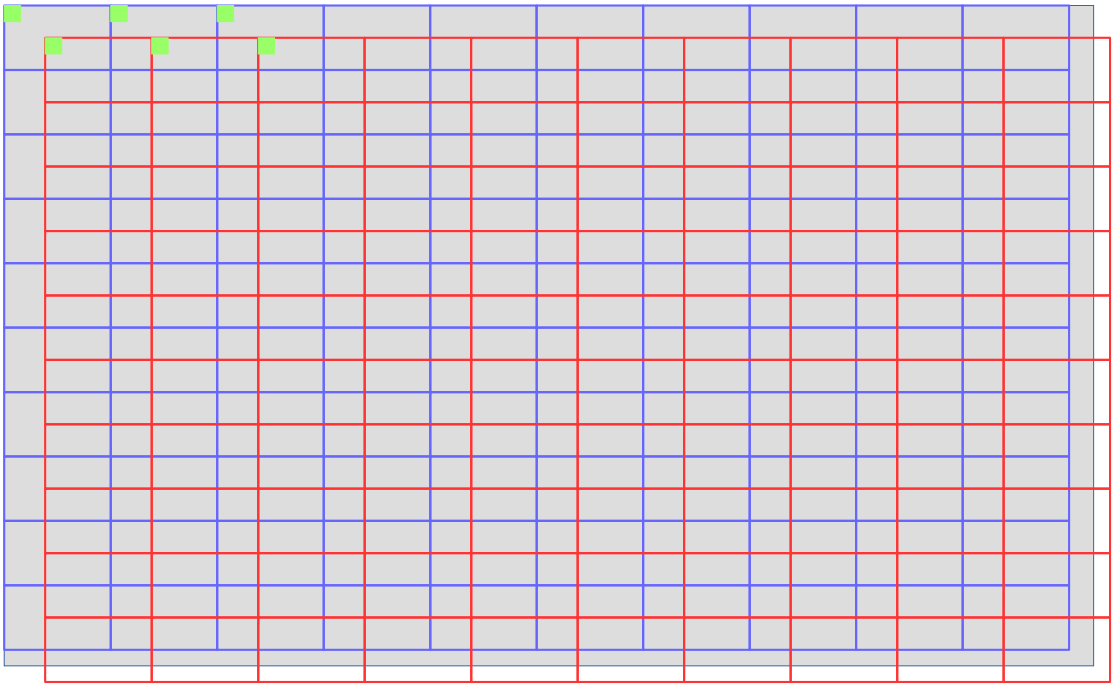


*Figure 6: Bestfocus grid*

In Figure 6 the image is the large grey rectangle. The primary frames are 10x10 blue, the secondary 10x10 frames are red. The small green boxes represent the focus boxes. Only a sample of boxes are displayed. This example would have 200 focus boxes.

A simple use of this tool would default to 10x10 frames, with each box the same size as the frame. There would then be 200 possible boxes where it could find good focus. I've run the tool with 100x100 frames (so 20,000 frames in total). Why do this? Because the particular definition we have of good focus means lots of colour transitions equates to good focus, so for example some small text in a image might cause lots of of black/white transitions in a small space. And a really big red

triangle might be bypassed . If we reduced the size of the box to the size a letter they compete on a more equal footing For speed we may not need e.g a 12x12 box every 12 pixels we might be happy with sampling spaced out by a bigger frame. So a 12x12 box in a 120x120 frame might usually find the “object” we’re expecting to be in focus.

## The tools under test

I tested this on a tabletop. I produced some rather amusing pointers as real physical paper pointers. The images had an arrow and the word focus, however to avoid “capturing” the focus the arrow an text are printed out of focus.*So we end up with the word focus printed out of focus.*


*Figure 7: Test overview*

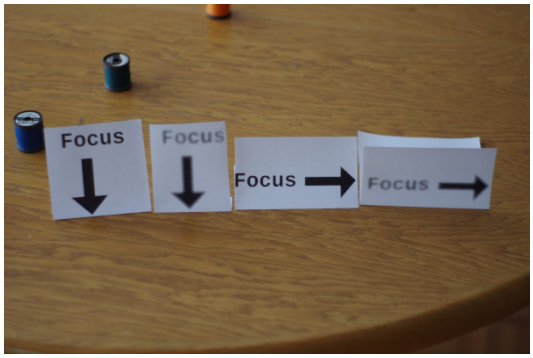



*Figure 8: Out of focus arrows*

An example image would be

Figure 9 here the sharp focus arrow might distract the focus, so only the defocussed version is used in tests.


*Figure 9: Reel with arrow*

*Figure 10: Sweep1*

*Figure 11: Sweep2*


*Figure 12: Sweep3*

*Figure 13: sweep4*


*Figure 14: Sweep5*

*Figure 15: Sweep6*


*Figure 16: sweep7*

Note these are 25% crops. The original images are jpegs produced on a Pentax K3 with a fixed 50mm lens (Manual focus from a K1000). This is series of images where focus sweeps across the cotton reels (left to right) while the camera is fixed on a tripod. The exif info for sweep1 is:

```
$ exiftool SW1-IMGP5111.jpg
File Name                       : SW1-IMGP5111.jpg
File Size                       : 9.6 MB
File Type                       : JPEG
Make                            : RICOH IMAGING COMPANY, LTD.
Camera Model Name               : PENTAX K-3
Orientation                     : Horizontal (normal)
X Resolution                    : 300
Y Resolution                    : 300
Software                        : PENTAX K-3 Ver. 1.21
Artist                          : GRAEME VETTERLEIN
Copyright                       : CREATIVE COMMONS
Exposure Time                   : 1/125
F Number                        : 2.0
Exposure Program                : Aperture-priority AE
Sensitivity Type                : Standard Output Sensitivity
Flash                           : Off, Did not fire
Focal Length                    : 50.0 mm
ISO                             : 200
Data Scaling                    : 8192
Lens Type                       : A Series Lens
Camera Temperature              : 29 C
Focus Mode                      : Manual
Noise Reduction                 : Off
Shake Reduction                 : On (AA simulation off)
Shutter Count                   : 14910
White Balance Auto Adjustment   : Off
Contrast Highlight/Shadow Adj   : Off
Fine Sharpness                  : Off; Normal
Image Size                      : 6016x4000
```

Running bestfocus(1) on these images yields:

```
$ ../../bestfocus SW?-IMGP*.jpg
SW1-IMGP5111.jpg 4576818 (1502:1800-2102:2199)
SW2-IMGP5110.jpg 5428227 (2103:1800-2703:2199)
SW3-IMGP5109.jpg 5320010 (2704:1800-3304:2199)
SW4-IMGP5108.jpg 7948667 (2704:1400-3304:1799)
SW5-IMGP5107.jpg 7626049 (3305:1800-3905:2199)
SW6-IMGP5106.jpg 1490741 (3005:2000-3605:2399)
SW7-IMGP5105.jpg 8020503 (4207:2000-4807:2399)
```

Not seen here, for brevity, but the “frame size” is 601 wide by 400 high. The focus box entirely fills the frame, so the default 200 frames.

The BESTBOX found in each image can been seen in this modified images:

*Figure 17: sweep1-box10*

*Figure 18: sweep2-box10*

*Figure 19: sweep3-box10*

*Figure 20: Sweep4-box10*

*Figure 21: Sweep5-box10*

*Figure 22: Sweep6-Box10*


*Figure 23: Sweep7-Box10*

For this image, it seems to me the 10x10 frames is good enough. The “best” image is “Sweep7” and the worst is “Sweep6” . Looking at them this appears right. I didn’t intend to do a bad job of focusing Sweep6 (by hand) so it’s a genuine error.

But to investigate the options of bestfocus(1):

Consider this image:


*Figure 24: centre focus*

```
$ ../../checkfocus centre-IMGP5095.jpg
centre-IMGP5095.jpg 102335574 49388352 52947222 4228336 2270247 1958089 0 2

$ ../../checkfocus ../sweeping/sweep7.jpg
../sweeping/sweep7.jpg 11983989 5502251 6481738 136271 90988 45283 0 1

$ ../../bestfocus -v centre-IMGP5095.jpg
Package check_focus[0.3] ../../bestfocus - find best area of focus
Processing file: centre-IMGP5095.jpg
Image will be divided into 10 horizontal frames by 10 vertical frames (plus as many again alternate frames)
horizontal frames are 601 pixels wide by 400 high
Each frame contains a single focus box of 601:400 high
Colour Space     : JCS_RGB
Output Width     : 6016
Output Height    : 4000
Output Colour Components: 3 Number of components in colour Space)
Output Components       : 3 (actual number per Pixel [1 iff indexed])
Vertical Frames  : 10
Horizontal Frames: 10
Frame Height     : 400
Frame Width      : 601
Box Height       : 400
Box Width        : 601
New BESTBOX found: box001: Stat=646548 (0:0-600:399)
New BESTBOX found: box006: Stat=669873 (3005:0-3605:399)
New BESTBOX found: box007: Stat=740602 (3606:0-4206:399)
New BESTBOX found: box008: Stat=840570 (4207:0-4807:399)
New BESTBOX found: box010: Stat=982410 (5409:0-6009:399)
New BESTBOX found: box105: Stat=1033634 (2704:200-3304:599)
New BESTBOX found: box115: Stat=1109163 (2704:600-3304:999)
New BESTBOX found: box026: Stat=1202627 (3005:800-3605:1199)
New BESTBOX found: box125: Stat=3105509 (2704:1000-3304:1399)
New BESTBOX found: box035: Stat=3200177 (2404:1200-3004:1599)
New BESTBOX found: box036: Stat=5106683 (3005:1200-3605:1599)
New BESTBOX found: box135: Stat=7035849 (2704:1400-3304:1799)
Final BESTBOX: box135: Stat=7035849 (2704:1400-3304:1799)
centre-IMGP5095.jpg 7035849 (2704:1400-3304:1799)
Processing took 2 seconds
```

Centre box

Using simple checkfocus(1), we find the small centre box is nice and sharp, it's better than the centre of sweep7 Figure 23 because there, the best part of the image was off centre.

If we run **bestfocus(1)** with the verbose option we can see it iterate towards the best 601x400 box.

These are show in this picture.

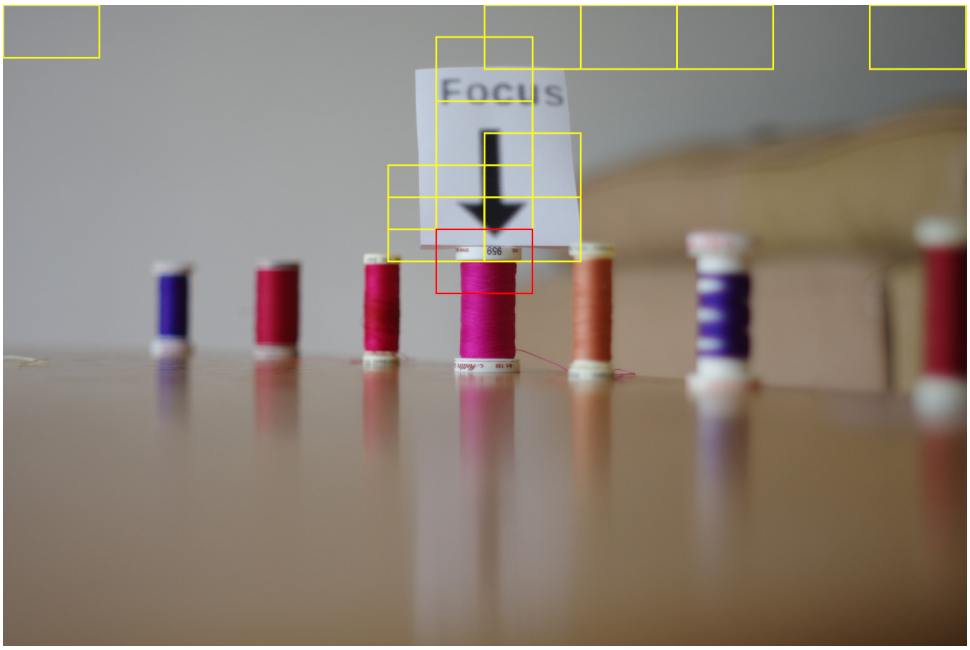
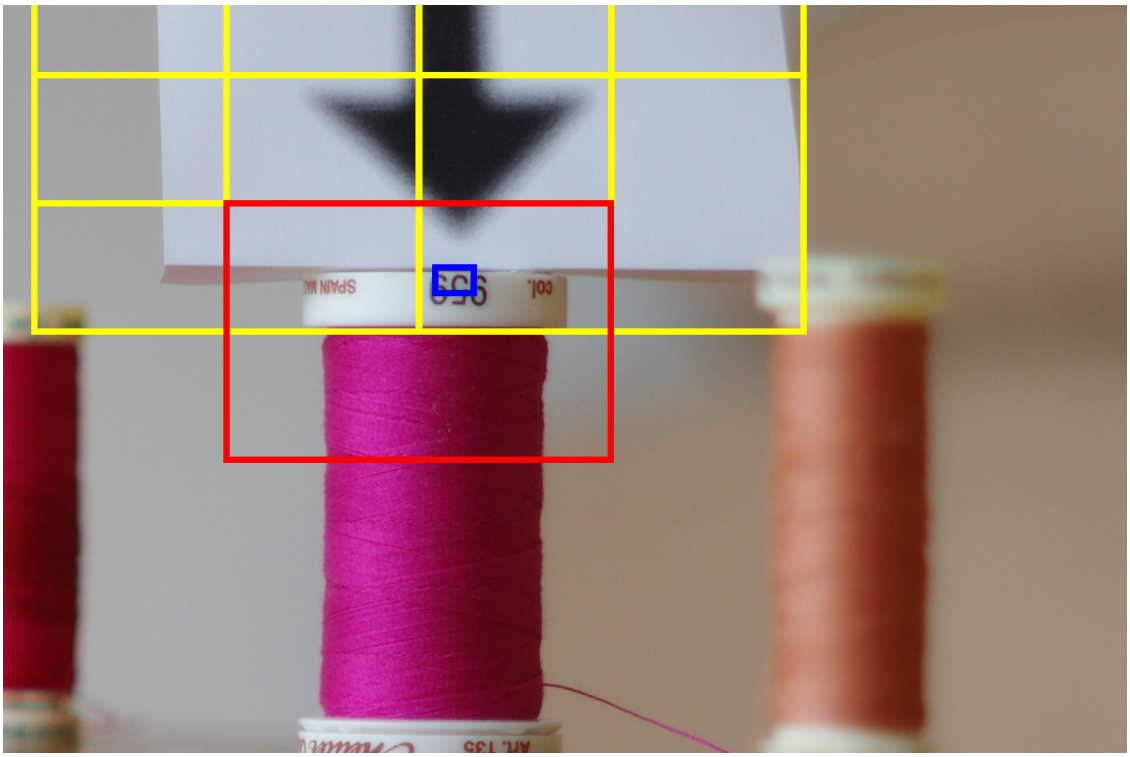



*Figure 25: centre box10 with focus boxes*

The paper label with the arrow attracted our focal point but since it was printed out of focus it was beaten by the text at the top of the cotton reel.

Repeating the run with blue or red channels selected finds the same result, just with different scores. Switching to a 100x100 grid, points out the attractiveness of the black on white text



## Version 0.2 & 0.3 enhancements

A few useabaliblity enhancements have been added in these versions.

# General

Both tools now support **-f <listlist>** this allows a list of files to be processed to be read from a filename, so you could for example generate a list using find(1) of particular files , eg newer than 5 days etc.

# checkfocus(1)

The option **–shift <no bits> | -s <nobits>** reduces the size of the numbers for the whole image (it has no effect on the centre box) **-s 1** rights right once (so divides by 2) **-s 8** shifts 8 bits (divide by 256) this makes the numbers more manageable  This may be needed by tools used later in your workflow which might use e.g. 32 or 64 bit arithmetic.

The option **-F[<rows:cols>] | --fudge[=<row:col>]** defines a “fudgebox”. This requires a bit of explantation but first note the syntax, the rows and columns are optional. If they are used with the shortform no space is allowed, with the longform an equals sign is required. If they are not given the values of 12000:8000 will be used.

How does a “fudgebox” work?  . Imagine you take a photo at 3000x2000 resolution then you retake that image at 6000x4000. Both are equally “good” . The 2nd one will always come out better because it has **more** good points of focus. If you intend to enlarge this image then that’s fine. The bigger image is better. However if you intend to scale the image down that may not be the case. If you have a bunch of mixed size images how do you compare a pin sharp small image with a fuzzy big image. It’s actually quite hard to decide, it depends on how you intend to use the image. This attempts to “fudge” the numbers for the whole image scores (not the centre-box)  to what they would be for the fudgebox size image, e.g. 12000:8000. It can only do this for images smaller than the fudgebox. The default (12000x8000) is 96MP , so right now is probably big enough, so you can just use -F for the moment.

Now if you play with this feature you may decide “*this is nonsense, just use the full image score in all cases*”, “*this is nonsense, just use the centre-box scores*” or “*this is obviously the way to compare images, why is it not the default?*” Rest assured somebody else will hold one of the other positions. See what works for you/your workflow.

The options **-o <0-255> --overexposed=<0-255>  -u<0-255>  --underexposed=<0-255>** are slightly misnamed **dark** and **light** might be better terms. They set upper and lower bounds on the channel (default green) values at or above that value are considered overexposed below the lower setting are underexposed. The number of pixels under and over exposed is counted and the two percentages are output following the centre-box scores .

The defaults are -o 230 and -u 25 . Values of 255 and 0 might be better choices depending on your source.  Computer graphics, yes; Digital DSLR probably, scanned slides/negatives no. These just serve a quick check of image quality. I had one image with 100% overexposed, yep it was simply a white box a dust check frame that got saved in error.

*Drawing 4: Over and under exposed scores*

```
$checkfocus IMGP4934.jpg
IMGP4934.jpg 666184535 420920329 245264206 90422724 47896857 42525867 0 18
```

In Drawing 4 18% of the pixels are underexposed (on the green channel). A “bad image” may have 98% of underexposed pixels. This was the dark image I’d dismissed [Drawing 1] but turned out to be well focussed.

# Back at the start

Oh yes that poppy on page1, a red centre section in focus, trips up my camera autofocus

```
$ ../../checkfocus '/home/graeme/src/check_focus/doc/MGP4188-rt.jpg'
/home/graeme/src/check_focus/doc/MGP4188-rt.jpg 53280333 27025243 26255090 4636062 2187306 2448756 0 14

$ ../../checkfocus -r '/home/graeme/src/check_focus/doc/MGP4188-rt.jpg'
/home/graeme/src/check_focus/doc/MGP4188-rt.jpg 57606816 31025118 26581698 6272471 2863523 3408948 1 2

$ ../../checkfocus -b '/home/graeme/src/check_focus/doc/MGP4188-rt.jpg'
/home/graeme/src/check_focus/doc/MGP4188-rt.jpg 42597818 22707184 19890634 2007549 963482 1044067 0 23

$ ../../checkfocus -g '/home/graeme/src/check_focus/doc/MGP4188-rt.jpg'
/home/graeme/src/check_focus/doc/MGP4188-rt.jpg 53280333 27025243 26255090 4636062 2187306 2448756 0 14
```

The focus checks get higher scores roughly in the ratio 1:2:3 , red being the highest, but only for the centre bit, with the red poppy.

# Worked example

On 26th Aug 2016 I attended a "track day" in Oxfordshire. I took a number of photos:

Right now I have 310 jpegs in my rejected pile, 247 digital negatives and 58 images I've tagged as the "best" and backed up. The "best" are in a different directory.  I set my camera to record both "jpeg" and "raw". If I needed a "high speed burst" I temporally turned off the "raw" copy. So all raw images have a processed jpeg doppelganger but the reverse is not always true.

```
~/Pictures/2016/20160826$ ls -1  *.JPG   | wc
    310     310    8986
~/Pictures/2016/20160826$ ls -1  *.DNG   | wc
    247     247    7163
~/Pictures/2016/20160826$ ls -l ../../Photos/Graeme/DigitalPhotos/20160826-best/*.[jJ][pP][gG] | wc
     58     522    7262
```

Now with these tools I can't process the .DNG (raw) files but may use these if the corresponding jpeg is only deficient in terms of exposure.

First I make a list of all the files I plan to process:

```
~/Pictures/2016/20160826$ ls -1 ../../Photos/Graeme/DigitalPhotos/20160826-best/*.[jJ][pP][gG] *.[jJ]
[pP][gG] > /tmp/all-images
~/Pictures/2016/20160826$ wc /tmp/all-images
  368   368 13464 /tmp/all-images
```

368, sounds about right.

So, just consider the overall focus

```
~/Pictures/2016/20160826$ sort -k2 -nr /tmp/all-focus  > /tmp/overall-focus
~/Pictures/2016/20160826$ head -10 /tmp/overall-focus
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-153506-IMGP8126-rt2.jpg 5213981622 2491804632
2722176990 59927475 27979595 31947880 3 7
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-115435-IMGP7955.JPG 4121432654 1992064844 2129367810
32945181 12928617 20016564 17 5
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-153012-IMGP8114.JPG 3527931061 1090538775 2437392286
4255665 1193276 3062389 0 24
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-114655-IMGP7900-gimped.JPG 3070937719 1245506916
1825430803 34415200 9562030 24853170 14 16
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-112358-IMGP7815.JPG 2616852716 1023947254 1592905462
45190042 19603821 25586221 10 8
../../Photos/Graeme/DigitalPhotos/20160826-best/IMG_20160826_102257.jpg 2529761897 1084704379 1445057518
17840853 7381407 10459446 0 17
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-113323-IMGP3601.JPG 2451754744 1367940273 1083814471
11063898 5189512 5874386 2 0
20160826-120348-IMGP8074.JPG 2412350029 1150037191 1262312838 33751040 11017946 22733094 0 1
20160826-153007-IMGP8113.JPG 2367175835 701289640 1665886195 28984623 3572396 25412227 0 31
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-113618-IMGP3610-rt.jpg 2332172269 1147203823
1184968446 27111359 12094646 15016713 0 6
```

Generally reassuring, best 7 overall focus are already in my "best" folder.

```
~/Pictures/2016/20160826$ tail -10 /tmp/overall-focus
20160826-115832-IMGP3620.JPG 102009759 36522367 65487392 7679222 2446794 5232428 0 9
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-152158-IMGP8097.JPG 95687921 48472890 47215031
3196703 1378581 1818122 0 15
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-152129-IMGP8095.JPG 84903897 46298496 38605401
772766 399654 373112 8 5
20160826-120310-IMGP3627.JPG 81370057 30236085 51133972 3883665 1640713 2242952 0 10
20160826-201330-DUST.JPG 80849808 6923292 73926516 0 0 0 98 1
20160826-152747-IMGP8106.JPG 61025737 34479900 26545837 457406 239832 217574 22 3
20160826-152741-IMGP8105.JPG 59359558 25402206 33957352 316150 166013 150137 11 5
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-152144-IMGP8096.JPG 57883999 30696578 27187421
3856577 2758512 1098065 0 18
20160826-120254-IMGP3624.JPG 49841359 24323251 25518108 1154224 390877 763347 0 0
../../Photos/Graeme/DigitalPhotos/20160826-best/20160826-120301-IMGP3625.JPG 49801288 24288081 25513207
1204118 409985 794133 0 0
```

Among the worst (focus) is 20160826-201330-DUST.JPG , which is, as it sounds, a dust check image, note also it's 98% overexposed.
Now I did my categorization of these images a few years back, I don't want to really move files around, so to emulate doing this the first time, I'll link all the files into a single directory (hard links, so no space is used)

```
/data/tmp$ cat /tmp/all-paths | xargs -I{} ln {} .
```

..I'll rerun the **checkfocus** in this directory:

```
/data/tmp$ checkfocus -f all-images > all-focus

/data/tmp$ sort -n -k8  all-focus > all-overexpose
/data/tmp$ sort -n -k9  all-focus > all-underexpose

/data/tmp$ tail -10 all-underexpose
./20160826-153216-IMGP8120.JPG 294933632 78002712 216930920 3114110 727310 2386800 0 28
./20160826-153007-IMGP8113.JPG 2367175835 701289640 1665886195 28984623 3572396 25412227 0 31
./20160826-112332-IMGP7814.JPG 160892432 68125760 92766672 6272969 1117463 5155506 0 35
./20160826-151811-IMGP8084.JPG 660278181 274272694 386005487 4962213 1527428 3434785 0 38
./20160826-152456-IMGP8098.JPG 238816498 79699470 159117028 13188821 4446847 8741974 6 39
./20160826-151843-IMGP8085.JPG 298709047 94130802 204578245 12034061 3662256 8371805 0 43
./20160826-153231-IMGP8121.JPG 361641690 84892830 276748860 2803151 748549 2054602 0 45
./20160826-152557-IMGP8102.JPG 130855516 51293558 79561958 911590 394659 516931 0 51
./20160826-113618-IMGP3610.JPG 106180852 47922794 58258058 6581949 2430995 4150954 0 53
./20160826-152624-IMGP8103.JPG 112316010 54837816 57478194 15349091 7269780 8079311 1 65
/data/tmp$ tail -10 all-overexpose
./20160826-152747-IMGP8106.JPG 61025737 34479900 26545837 457406 239832 217574 22 3
./20160826-153604-IMGP8129.JPG 444386862 156146417 288240445 7606059 3576450 4029609 23 1
./20160826-153426-IMGP8125.JPG 494451843 170622327 323829516 3268223 1196235 2071988 27 7
./20160826-151643-IMGP8081.JPG 176019432 80782623 95236809 2086638 955091 1131547 29 0
./20160826-151937-IMGP8088.JPG 462458368 183108693 279349675 2410454 1428223 982231 29 0
./20160826-152657-IMGP8104.JPG 175578430 70060421 105518009 3286714 1763650 1523064 29 0
./20160826-152530-IMGP8101.JPG 166421885 61528920 104892965 983698 391278 592420 31 0
./20160826-153415-IMGP8124.JPG 464523436 167043145 297480291 2235625 913112 1322513 35 4
./20160826-152953-IMGP8112.JPG 435166434 231823640 203342794 5667459 2702398 2965061 58 0
./20160826-201330-DUST.JPG 80849808 6923292 73926516 0 0 0 98 1
```

This uses the default values of underexposure (25) and over exposure (230). I'm going to make an executive decision, if 29% is under or over exposed (by this definition) then it's "bad". So we get:

```
/data/tmp$ cat is-under
./20160826-153007-IMGP8113.JPG 2367175835 701289640 1665886195 28984623 3572396 25412227 0 31
./20160826-112332-IMGP7814.JPG 160892432 68125760 92766672 6272969 1117463 5155506 0 35
./20160826-151811-IMGP8084.JPG 660278181 274272694 386005487 4962213 1527428 3434785 0 38
./20160826-152456-IMGP8098.JPG 238816498 79699470 159117028 13188821 4446847 8741974 6 39
./20160826-151843-IMGP8085.JPG 298709047 94130802 204578245 12034061 3662256 8371805 0 43
./20160826-153231-IMGP8121.JPG 361641690 84892830 276748860 2803151 748549 2054602 0 45
./20160826-152557-IMGP8102.JPG 130855516 51293558 79561958 911590 394659 516931 0 51
./20160826-113618-IMGP3610.JPG 106180852 47922794 58258058 6581949 2430995 4150954 0 53
./20160826-152624-IMGP8103.JPG 112316010 54837816 57478194 15349091 7269780 8079311 1 65

/data/tmp$ cut -d' ' -f1 is-under
./20160826-153007-IMGP8113.JPG
./20160826-112332-IMGP7814.JPG
./20160826-151811-IMGP8084.JPG
./20160826-152456-IMGP8098.JPG
./20160826-151843-IMGP8085.JPG
./20160826-153231-IMGP8121.JPG
./20160826-152557-IMGP8102.JPG
./20160826-113618-IMGP3610.JPG
./20160826-152624-IMGP8103.JPG


/data/tmp$ mkdir under
/data/tmp$ mv $(cut -d' ' -f1 is-under) under/
/data/tmp$ ls under/
20160826-112332-IMGP7814.JPG  20160826-152557-IMGP8102.JPG
20160826-113618-IMGP3610.JPG  20160826-152624-IMGP8103.JPG
20160826-151811-IMGP8084.JPG  20160826-153007-IMGP8113.JPG
20160826-151843-IMGP8085.JPG  20160826-153231-IMGP8121.JPG
20160826-152456-IMGP8098.JPG
```

And likewise for overexposed:

```
/data/tmp$ mkdir over
/data/tmp$ mv $(cut -d' ' -f1 is-over ) over
```

Another group of photos to sift out. A few images were taken at a high ISO, I'd use lower ISO versions in preference.

```
/data/tmp$ exiftool -p '$FileName  ISO $EXIF:ISO' *.jpg *.JPG > ISO
/data/tmp$ sort ISO -n -k3  > by-ISO
/data/tmp$ tail -10 by-ISO
20160826-153216-IMGP8120.JPG  ISO 200
20160826-153249-IMGP8122.JPG  ISO 200
20160826-153533-IMGP8127.JPG  ISO 200
20160826-153604-IMGP8129.JPG  ISO 200
20160826-152040-IMGP8092.JPG  ISO 280
20160826-152001-IMGP8089.JPG  ISO 800
20160826-152022-IMGP8091.JPG  ISO 1100
20160826-153426-IMGP8125.JPG  ISO 1100
20160826-153506-IMGP8126.JPG  ISO 1100
20160826-152007-IMGP8090.JPG  ISO 1600
```

Now in a similar fashion, take the best 100 overall focus

```
/data/tmp$ tail -100 all-overall-focus > is-overall-focus
/data/tmp$ mkdir in-focus
/data/tmp$ mv $(cut -d' ' -f1 is-overall-focus ) in-focus
mv: cannot stat './20160826-153007-IMGP8113.JPG': No such file or directory
```

That "cannot stat" is correct. That image is both "well focused overall" and underexposed, so we rejected it before(due to exposure). If we'd done this in the opposite order we'd have added an underexposed image to our good set. Would I want to do this? Maybe, I have the negatives after all. So for my working practices maybe I want to do this in to opposite order, or maybe I want more subdivisions? Anyhow I'll assume not for this example.

```
/data/tmp$ grep 20160826-153007-IMGP8113.JPG is*
is-overall-focus:./20160826-153007-IMGP8113.JPG 2367175835 701289640 1665886195 28984623 3572396 25412227 0
31
is-under:./20160826-153007-IMGP8113.JPG 2367175835 701289640 1665886195 28984623 3572396 25412227 0 31
```

Now what about images where just the centre was "good" , now we're working with the original set, so we'd expect many good overall good focus images to also have a good centre (so won't be found here anymore, we've already moved them), but we might have some nice blurred backgrounds.

```
/data/tmp$ sort -n -k5  all-focus > all-centre-focus
/data/tmp$ tail -100 all-centre-focus > is-centre-focus
/data/tmp$ mv $(cut -d' ' -f1 is-centre-focus ) in-focus
mv: cannot stat './20160826-115346-IMGP7937.JPG': No such file or directory
...
59 lines of "cannot stat"
...
```

so looks like we added (moved) 41 files that had a good centre focus but overall were not good.

```
ls in-focus/ | wc
    140     140    4076
```

So we have 140 images selected as "generically good"

If I take a quick look at the images left, turns out I have a number like these:

*Figure 26: Images with small area of focus(bad)*

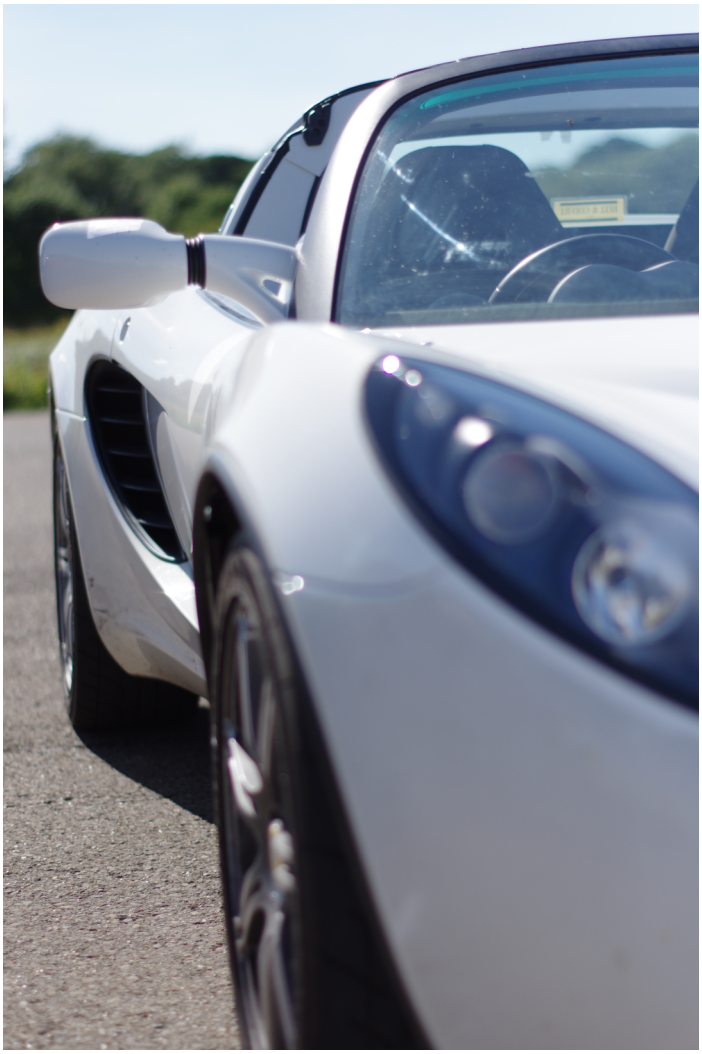


*Figure 27: Image with small area of focus (good)*

Clearly I was trying for some arty shots, with a nice BOKEH. One problem here is only a small areas is in focus. So we try bestfocus:

```
/data/tmp$ bestfocus *.JPG *.jpg > best-focus
/data/tmp$ sort -n -k2 best-focus > best-10x10-focus
```

Then I make an "arbitrary decision" that "good enough" is > 40,000,000. I can do this because all my images are the same size, so the focus boxes are all the same size 601:400 (so a score of 40 million in a 601x400 box).

```
/data/tmp$ mkdir in-box-focus
/data/tmp$ mv $(cut -d' ' -f1 best-10x10-box-focus ) in-box-focus

/data/tmp$ mkdir high-ISO
/data/tmp$ mv $(cut -d' ' -f1 is-high-ISO ) high-ISO/

/data/tmp$ tree-image-count .
185,185,0: /data/tmp
140,140,0: /data/tmp/in-focus
21,21,0:   /data/tmp/in-box-focus
9,9,0:     /data/tmp/under
7,7,0:     /data/tmp/over
5,5,0:     /data/tmp/high-ISO
```

So we've chopped the large set of images into smaller boxes. We should just view some thumbnails of "in-focus" and drag in the ones with good composition into a separate folder next look in "in-box-focus"

## So how to use the tools:

## Depends how you work:

### Wedding Photographer,

Huge stack of group photos, with bride and groom in the centre.

Use **checkfous(1),** move the under and overexposed images in the manner shown above. Sort by just the focus centre box (so -k5 on sort) move the low score x% (half?, 25%?) aside, depends how many shots you take, if you bracket etc.  Then review the remaining set for best version of "Bride's family", "grooms family" etc. Or maybe just upload this set for client review.

### Nature photographer

(e.g birds) shoots in jpeg, motor drive mode (fast as possible) Image has lots of out of focus background and a small sharp (bird) almost anywhere in the image typically 1000+ shots in a single session.

> *Run bestfocus in default setting, sort by score in best box.*
> *...go have a cup of tea ...*
> *review images in order given by sort (hint rename images to add score at front of name).*

### Landscape photographer

Possibly set the under and overexposure setting to 0,255 since images might be very bright or dark, then check for 0% of both (true under/overexposure) use **checkfocus(1)** and sort based on the overall score, or possible change the centre box to be top half of image and use that.

### Macro photography

use **bestfocus(1),** with a small grid , eg 100x100 instead of 10x10 . If speed becomes an issue set the focus box to something small, 12x12.

# Appendix

## Visualising focus boxes

A simple use of imagemagick can let you visualise what is happening:

```
$ bestfocus drone-focused.jpg
drone-focused.jpg 34763448 (448:108-575:179)
$ convert drone-focused.jpg -fill none -stroke red -strokewidth 10 -draw "rectangle 448,108,575,179" output.jpg
$ eog output.jpg
```

*Figure 28: output.jpg*