Dither Removal

Bart M. ter Haar Romeny

***Mathematica* is ideal for explaining the design of seemingly complex mathematical methods. In this article we explain the use of the 2D Fourier transform to remove unwanted dithering artifacts from images. All steps are visualized, so the reader can get a good idea of what the Fourier transform of an image looks like, the location of the origin, the artifacts and their extent, and how geometric reasoning works in the Fourier domain. The method leads to a marked clean up of images deteriorated by dither.**

■ Image Dithering

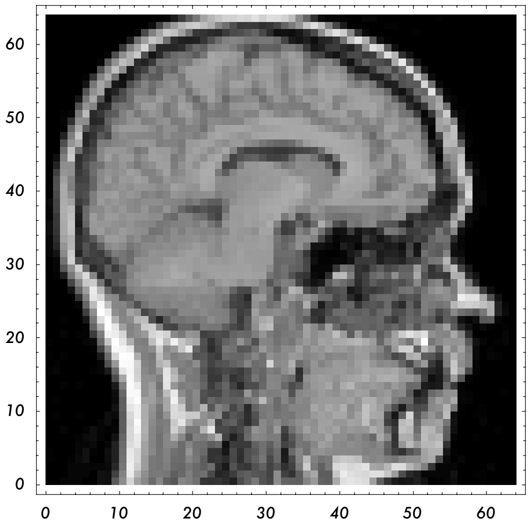
Image dithering is often used for commercial print products like newspapers and magazines. Many printers (e.g., laser and inkjet printers) can only put a series of single black dots on paper. Intensity is then regulated by the local density of the dots. This process is called dithering. However, when a dithered image is scanned, the interaction between the discrete scanning process and dithering may seriously degrade the image. Dithering and scanning are often both periodic, leading to periodic interference artifacts. These can be removed by proper filtering in the spatial 2D Fourier domain.

We first give an example of how dithering can be applied to an image. It is simplest to replace each pixel with a small array of random black dots, with the number of dots proportional to the intensity.

Some general settings:

```
In[1]:= SetOptions[ListDensityPlot, Mesh → False,
          ImageSize → 200, AspectRatio → Automatic, PlotRange → All];
        << LinearAlgebra`MatrixManipulation`;
        SetDirectory[DirectoryName[ToFileName[
            "FileName" /. NotebookInformation[EvaluationNotebook[]]]]];
```

In[4]:=
```
url = "http://www.bmi2.bmt.tue.
    nl/image-analysis/People/BRomeny/images/";
pixels = Import[url <> "mr64.gif"][[1, 1]];
ListDensityPlot[pixels]
```



In[7]:=
```
{min, max} = {Min[pixels], Max[pixels]}
```

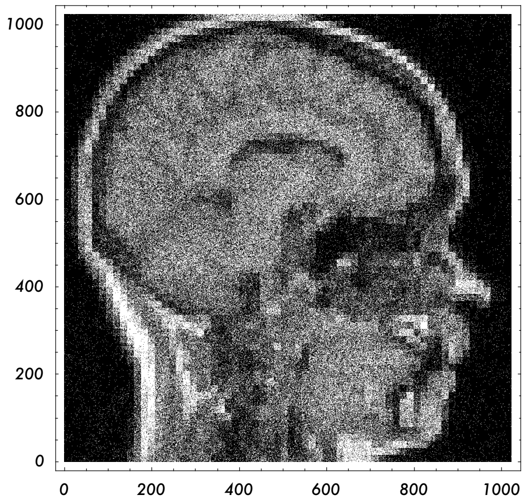
Out[7]=
```
{0, 214}
```

The dithering function creates a small 16×16 array, which enables us to make 256 different intensities.

In[8]:=
```
dither[int_] := Table[If[Random[] < int/214., 1, 0], {16}, {16}];
```

We map this onto every pixel and get an image that is 16 times larger with dithered points. The function `BlockMatrix` (available in `LinearAlgebra'Ma`⋱ `trixManipulation'`) combines the different submatrices.

In[9]:=
```
dithered = Map[dither, pixels, {2}];
ListDensityPlot[BlockMatrix[dithered]]
```



There are many sophisticated ways to dither an image. For a good overview see [1]. One of the best methods is the Floyd–Steinberg dithering method, based on error-diffusion [2]. The following graphic is an example.

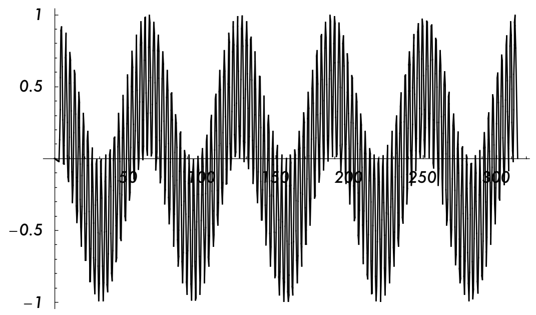
In[11]:=
```
Show[
  Import["http://www.webstyleguide.com/graphics/graphics/7.09.gif"]]
```

Full-color image
dithered
to two colors

■ Scanning a Dithered Image

A problem occurs when the dithering mask is not made up of randomly placed black-and-white pixels but has some regularity. The dithered image then contains many highly regular spatial frequencies, which may give rise to interference patterns when the image is sampled with a regular scanner, which likely has a different sampling frequency. The result is a range of extra harmonics (i.e., frequencies that are the result of the interference). They come as the products of the frequencies involved. An example of the period of the frequencies that emerge follows.

```
In[12]:= Plot[Sin[x] Sin[1.1 x], {x, 0, 100 π}]
```



This occurs because the product signal is actually the addition of the sum and the difference of the base frequencies.

```
In[13]:= TrigReduce[Sin[x] Sin[1.1 x]]
```

Out[13]= $\frac{1}{2}$ (Cos[0.1 x] - Cos[2.1 x])

The lower frequencies are rarely a problem. The higher frequencies can be removed by proper low-pass filtering.

■ Dither Removal by Filtering

A “large” dithered and scanned input image follows (3045 × 1017 pixels, 1.9MB). Note the periodic artifacts.

```
In[14]:= im = Import[url <> "AIA0019E-tiger.gif"];
         Show[im, ImageSize → 300]
```

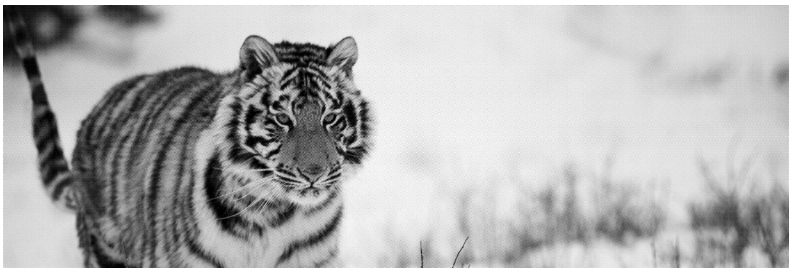


Figure 1. Image of a tiger, produced by printing a stock-art photo with an 85 lines per inch dot screen printer and then scanning.

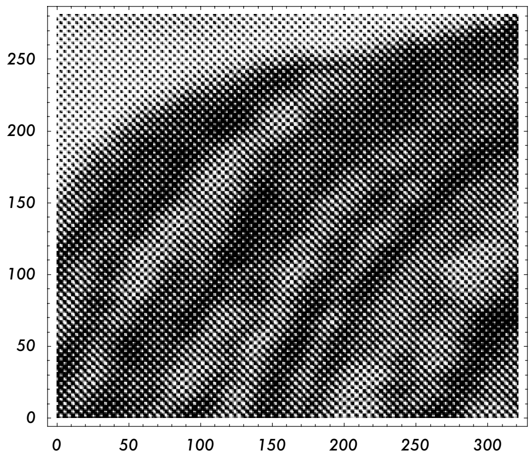
The image is imported as a structure, in which the pixels and the header data are embedded. The pixels can be extracted as the [[1,1]] element.

```
In[16]:=  pixels = im[[1, 1]];
          Dimensions[pixels]

Out[17]=  {1017, 3045}
```

To develop the method, we do not need the whole image. For speed and lower memory use, we take a subimage. We can see the periodic dithering more clearly by zooming in.

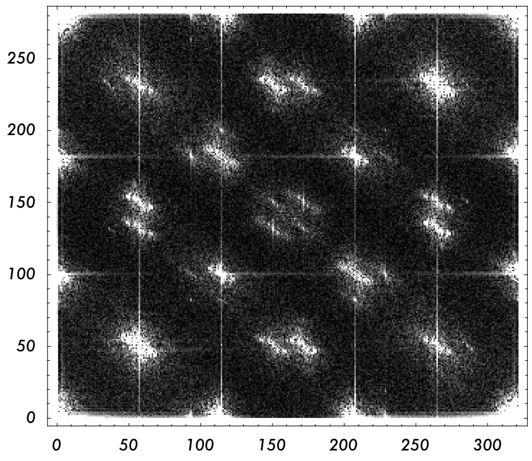
```
In[18]:=  ListDensityPlot[subim = Take[pixels, {500, 780}, {380, 700}]]
```



```
In[19]:=  {ydim, xdim} = Dimensions[subim];
```

Because the dithering has a high spatial frequency, we suspect these frequencies to be higher than most of the other frequencies in the image. We compute the Fourier spectrum of the input image and plot its absolute value. Because the origin is really large, the image is automatically scaled with a large factor, which reduces the contrast. We zoom in on a limited range (0 – 100) of spectral values.

```
In[20]:=  ListDensityPlot[Abs[fft = Fourier[subim]], PlotRange → {0, 100}]
```

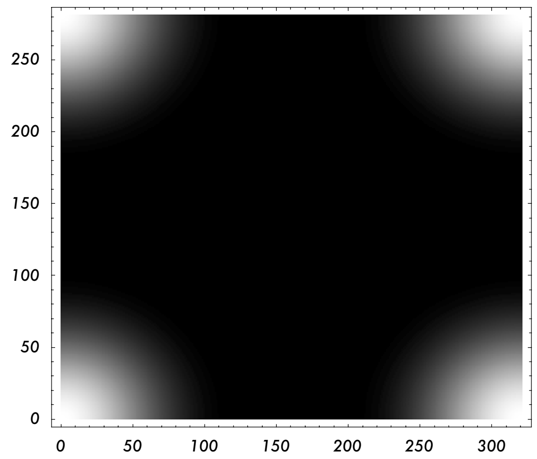


The significant image frequencies are around the origin (i.e., in the corners of the periodic spectrum plot). The marked peaks at other locations are proof of the existence of strong periodic signals due to dithering. We will use a mask that allows only frequencies near the origin to be included: a low-pass filter for spatial frequencies.

The mask is most easily created by an elliptic Gaussian weighting function in the center (different standard deviations in the *x* and *y* directions), which is then shifted over half the image size in each dimension to put the kernel at the origin of the Fast Fourier Transform (FFT) domain, which is in the lower left. Note that the Fourier domain is periodic in either dimension, so we actually see a single tile of a whole tiled 2D space in the *x* and *y* directions. The standard deviations have to be estimated from the 2D plot of the spectrum. For speed, we exploit the separability of the 2D Gaussian kernel. We first make two 1D arrays of Gaussian values and then multiply them into a matrix with `Outer` in a compiled function. This is about 140 times faster than a 2D `Table` command. `σxs` and `σys` are the standard deviations of the Gaussian in the *x* and *y* directions of the subimage.

```
In[21]:= σxs = 40.0; σys = 35.;
         mask = Compile[{}, Outer[Times,
                  Exp[-(Range[ydim] - ydim/2)^2/(2 σys^2)], Exp[-(Range[xdim] - xdim/2)^2/(2 σxs^2)]]];
         maskr = RotateLeft[mask[], Floor[{ydim/2, xdim/2}]];

In[24]:= ListDensityPlot[maskr]
```

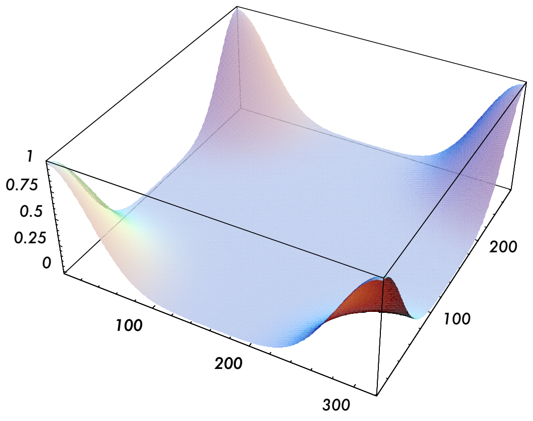


We also can inspect the amplitude spectrum and the low-pass Gaussian filter (both functions are scaled for plotting) along the interpolated diagonal positions of the rectangular image.

```
In[25]:= interpolated = ListInterpolation[afft = Abs[fft = Fourier[subim]]];
```

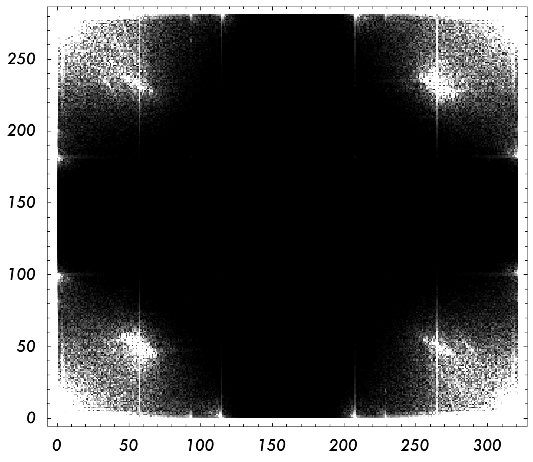
Here we view the mask as a height plot.

```
In[26]:= ListPlot3D[maskr, Mesh → False, PlotRange → All]
```



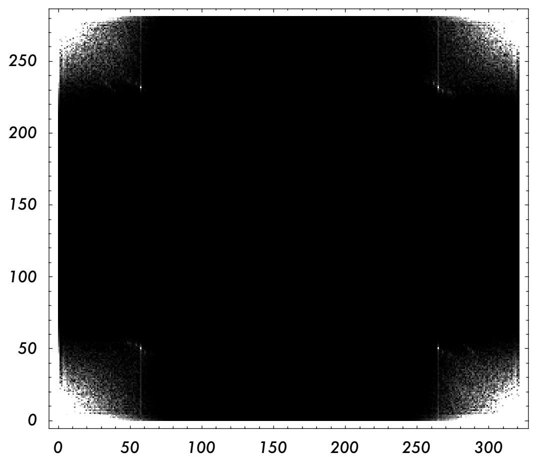
We multiply the Fourier spectrum with the mask and study it visually.

```
In[27]:= ListDensityPlot[maskr Abs[fft], PlotRange → {0, 10}]
```



We mask out the unwanted frequencies by multiplying the spectrum with the mask, and we get the resulting image by the inverse Fourier transform. The Gaussian filtering is still letting some periodic frequencies through (we only like to keep the information near the corners), so we make it steeper by raising it to the third power.

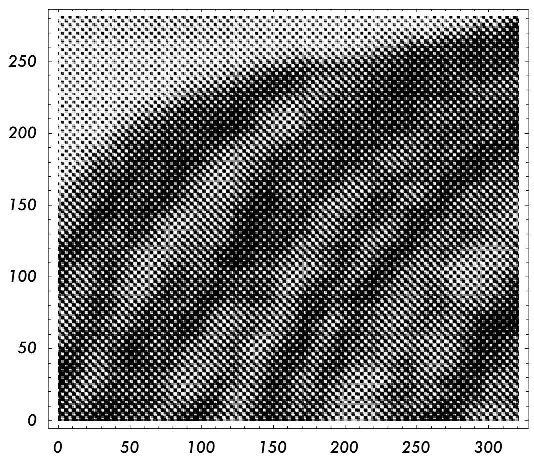
```
In[28]:=  ListDensityPlot[maskr³ Abs[fft], PlotRange → {0, 10}]
```

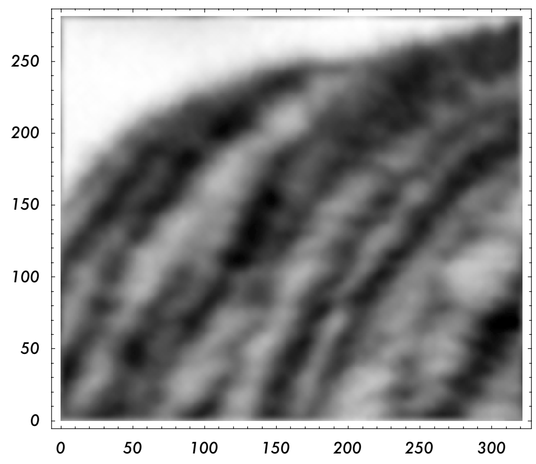


```
In[29]:=  restored = Abs[InverseFourier[fft maskr³]];
```

The result is a markedly improved image, without the disturbing dithering frequencies.

```
In[30]:=  ListDensityPlot /@ {subim, restored}
```





The dither is clearly removed, and details, such as the local hair bundles, can now clearly be discriminated.

The following reads the whole image, processes the data in the Fourier domain (note that σ_x and σ_y are proportionally rescaled), and writes the restored image to disk, in this case as a GIF file.

```
In[31]:= pixels = im[[1, 1]];
         {ydim, xdim} = Dimensions[pixels];
         fft = Fourier[pixels];
                                Dimensions[pixels]
         {σy, σx} = {σys, σxs} ------------------ ;
                                Dimensions[subim]
         mask = Compile[{}, Outer[Times,

                      (Range[ydim] - ydim/2)^2            (Range[xdim] - xdim/2)^2
                Exp[- ------------------------], Exp[- ------------------------]]]];
                              2 σy^2                              2 σx^2
         maskr = RotateLeft[mask[], Floor[{ydim/2, xdim/2}]];
         restored = Abs[InverseFourier[fft maskr^3]];

In[38]:= Display["restored.gif", ListDensityPlot[
           restored, ImageSize → Reverse[Dimensions[restored]],
           DisplayFunction → Identity], "GIF"];
```

■ Conclusion

This example shows the use of Fast Fourier Transform to remove periodic artifacts from an image. Despite the large size of the input image, the numerical performance of *Mathematica* is good.

■ References

[1] R. L. Adler, B. P. Kitchens, M. Martens, C. P. Tresser, and C. W. Wu, "The Mathematics of Halftoning," *IBM Journal of Research and Development*, **47**(1), 2003 pp. 5–15 www.research.ibm.com/journal/rd/471/adler.pdf.

[2] R. Floyd and L. Steinberg, "An Adaptive Algorithm for Spatial Grey Scale," *SID International Symposium Digest of Technical Papers,* San Jose, CA: The Society for Information Display (SID), 1975 pp. 36–37.

B. M. ter Harr Romeny, "Dither Removal," *The Mathematica Journal*, 2012.
dx.doi.org/10.3888/tmj.10.2-9.

About the Author

Bart M. ter Haar Romeny received an M.S. in applied physics from Delft University and a Ph.D. in 1983 from Utrecht University, The Netherlands. Ter Haar Romeny served from 1983 to 2001 as principal physicist and associate professor in the Department of Radiology and Image Sciences Institute at Utrecht University. He is now a full professor in the Department of Biomedical Engineering at Eindhoven University of Technology. His interests are medical image analysis and multiscale computer vision, including its mathematical foundations and clinical applications. A focus on the human visual system gives biological inspiration for mathematics. He is the author of a popular, interactive tutorial book on perceptually inspired multiscale image analysis written in *Mathematica*.

Bart M. ter Haar Romeny
Eindhoven University of Technology
Department of Biomedical Engineering
Biomedical Image Analysis
Den Dolech 2, WH2.106
NL-5600 MB Eindhoven, The Netherlands
B.M.terHaarRomeny@tue.nl
bmia.bmt.tue.nl/people/bromeny/index.html