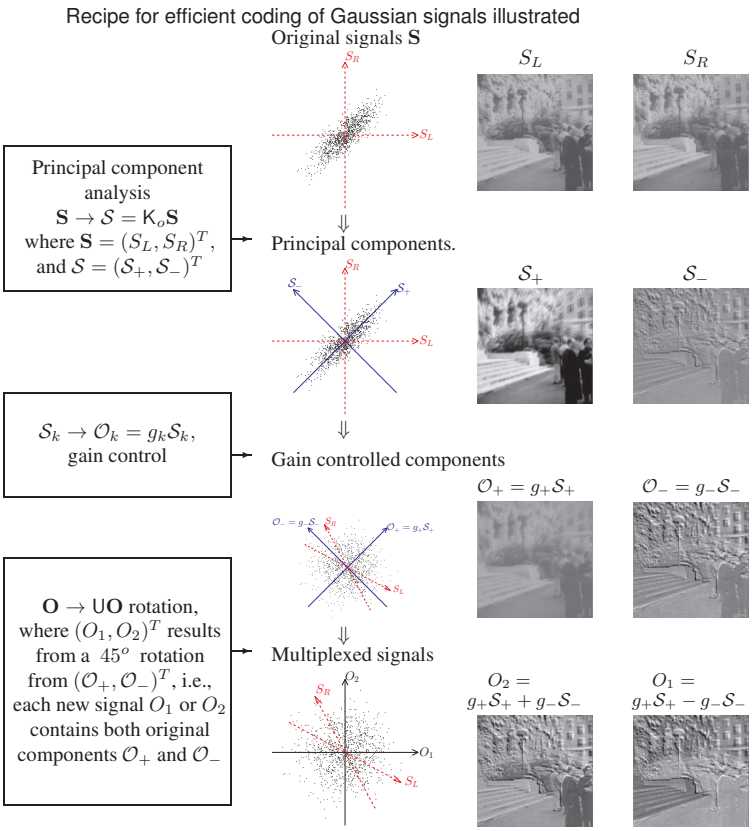

Figure 1: This is figure 3.17 in the textbook "Understanding vision: theory, models, and data". Illustration of the three conceptual ingredients in the recipe for efficient coding, using the example of stereo coding. Each scatter plot contains random samples from distribution, (top to bottom) $P(\mathbf{S} = (S_L, S_R))$, $P(\mathcal{S}_+, \mathcal{S}_-)$, $P(\mathcal{O}_+, \mathcal{O}_-)$, or $P(O_1, O_2)$. The relationship between the transformed signals $\mathcal{S}_\pm$, $\mathcal{O}_\pm$, $O_{1,2}$, and the original signals $S_{L,R}$ are evident in the directions of the axes for the transformed and the original signals in each plot. Examples of stereo images and their transforms are shown in the two rightmost columns.

Homework 2, Please submit your completed homework by a single pdf file

1. you must have completed the online videos and quizzes in the first six chapters of the online course "The efficient coding principle". Please send me a record, such as a screen shot at the online course, showing your progress status or completion status for this requirement.

2. Write a 2-5 page summary of the materials in the online course content above.

3. Practice exercises for introductory math for computational vision and initial steps of efficient coding using some programming. Some of these exercises will produce some plots like the plots in Fig. 1, which is Figure 3.17 of the textbook. You can compare your plots with these plots as you progress through the exercises.

Please do the following:

- (A): Take a pair of stereo images, one image, denoted as $S_L(x, y)$, is for the left eye, and one $S_R(x, y)$ for the right eye (you note that these two images are slightly different from each other, even though they look similar). Here x and y are the coordinates of the image pixel locations. For example $x = 1, 2, ..., 256$ and $y = 1, 2, ..., 256$. You may find some examples online, for example at this website https://www.londonstereo.com/3-D-gallery1.html. If the original images are colored, remove the color. Plot out the images, each as a luminance image.
- (B): Please normalize each image as follows. For $S_i(x, y)$, with $i = L$ or $i = R$, find S_i^{min} and S_i^{max} as the mininum and maximum of $S_i(x, y)$ across all pixel locations (x, y). Then, for $\hat{S} = 255$, do

$$S_i(x, y) \to \hat{S} \cdot (S_i(x, y) - S_i^{min})/(S_i^{max} - S_i^{min}). \tag{1}$$

  Now $0 \leqslant S_i(x, y) \leqslant \hat{S}$. Round each $S_i(x, y)$ into an integer value so that $S_i(x, y)$ should be an integer between 0 and $\hat{S}$.
- (C): For each $S_i(x, y)$, with $i = L$ or $i = R$, calculate the probability $P(S)$ for $S_i(x, y) = S$ across all (x, y). This means, for each $S = 0, 1, 2, ..., \hat{S}$, let $n(S)$ be the number of pixels (x, y) for which $S_i(x, y) = S$, and let N be the total number of pixels in S_i, then $P(S) = n(S)/N$.

  Plot out $P(S)$ vs S for each image S_i.
- (D): For each $S_i(x, y)$, with $i = L$ or $i = R$, calculate the pixel entropy

$$H(S_i) = -\sum_{S_i} P(S_i) \log_2 P(S_i). \tag{2}$$

  Please note that, if for some values S you have $P(S) = 0$, in such a case $P(S) \log_2 P(S) = 0$. Your computer will complain if you try to calculate $\log_2 P(S)$ when $P(S) = 0$. So omit these S values with zero $P(S)$ when doing the sum above.

  Write out what $H(S)$ is for each image S_i.
- (E): Calculate the joint probability $P(S_1, S_2)$ of $S_L(x, y) = S_1$ and $S_L(x, y) = S_2$ for $S_1 = 0, 1, 2, ..., \hat{S}$ and $S_2 = 0, 1, 2, ..., \hat{S}$. This means, for each possible pair of values (S_1, S_2), go across all pixels (x, y) to find $n(S_1, S_2)$ as the number of pixels satisfying $S_L(x, y) = S_1$ and $S_R(x, y) = S_2$. Then $P(S_1, S_2) = n(S_1, S_2)/N$.

  Plot out $P(S_1, S_2)$ (which is the joint probability distribution function) versus S_1 and S_2.
- (F): Calculate joint entropy as

$$H(S_1, S_2) = -\sum_{S_1, S_2} P(S_1, S_2) \log_2 P(S_1, S_2) \tag{3}$$

  Write out the value for $H(S_1, S_2)$.

- (G): Calculate mutual information between corresponding pixels in the two images as

$$I(S_1; S_2) = \sum_{S_1, S_2} P(S_1, S_2) \log_2 \frac{P(S_1, S_2)}{P(S_1)P(S_2)} = H(S_1) + H(S_2) - H(S_1, S_2). \quad (4)$$

  Write out the value for $I(S_1, S_2)$.

- (H): Calculate the redundancy between the left and right eye images as

$$\text{Redundancy} = \frac{H(S_1) + H(S_2)}{H(S_1, S_2)} - 1. \quad (5)$$

- (I): So far, you have done (B)-(H) when the highest pixel value is $\hat{S} = 255$. Now, repeat (B)-(H) for $\hat{S} = 127, 63, 31, 15, 7, 3, 1$. In other words, you can take $\hat{S} = 2^n - 1$ for $n = 1, 2, 3, ..., 8$ (so that $\hat{S} = 255$ when $n = 8$), so that you use n bits to present each image pixel. Plot $H(S_i)$, $H(S_1, S_2)$, $I(S_1; S_2)$ and Redundancy as functions of n. Also, please plot out the two images for each n value, and see if they make sense.

- (J): Let us go back to take $\hat{S} = 255$, so that each $S_i(x, y)$ pixel is represented by 8 bits. For each image $S_i(x, y)$, let

$$\bar{S}_i = \sum_{x,y} S_i(x, y)/N \quad (6)$$

  be the average value of $S_i(x, y)$ across all the image pixels. Then shift the pixel value

$$S_i(x, y) \to S_i(x, y) - \bar{S}_i \quad (7)$$

  so that each image should now have a zero mean value.

  Now, the correlation between $S_i(x, y)$ and $S_j(x, y)$ across pixels is

$$\begin{aligned} \mathsf{R}^S_{ij} &\equiv \langle S_i S_j \rangle & (8) \\ &= \frac{\sum_{x,y} S_i(x, y) S_j(x, y)}{\sum_{x,y} 1} & (9) \\ &= \frac{\sum_{x,y} S_i(x, y) S_j(x, y)}{N}. & (10) \end{aligned}$$

  So you can get a 2×2 matrix R^S with elements R^S_{ij} for $i = 1, 2$ and $j = 1, 2$. Please note that these are correlation, not correlation coefficients (which have a magnitude no larger than one).

$$\mathsf{R}^S = \begin{pmatrix} \mathsf{R}^S_{11} & \mathsf{R}^S_{12} \\ \mathsf{R}^S_{21} & \mathsf{R}^S_{22} \end{pmatrix} \quad (11)$$

  The diagonal elements, R^S_{11} and R^S_{22} of this matrix are the variance of pixel values in each monocular image, and the off-diagonal values are the covariance between the two monocular images. Please write out this matrix value.

- (K): Give a scatter plot of $S_L(x,y)$ versus $S_R(x,y)$. This means, start with a plot with horizontal and vertical axes, plot a point at location $(S_L(x,y), S_R(x,y))$, with the value of $S_L(x,y)$ and $S_R(x,y)$ on the horizontal and vertical axes respectively, for each pixel (x,y) in images $S_L(x,y)$ and $S_R(x,y)$. Compare your plot with one in Figure 1, and see if they look similar.
- (L): Calculate the eigenvalues and eigenvectors of the 2×2 matrix R^S. Plot each eigenvector as a vector in the scatter plot you obtained above in (K), and observe how each eigenvector is related to the character of this scatter plot of data, and observe how each eigenvalue is related to the variance of the data projected onto each eigenvector.
- (M): Define

$$\mathcal{S}_+(x,y) = \frac{1}{\sqrt{2}}(S_L(x,y) + S_R(x,y)) \tag{12}$$

$$\mathcal{S}_-(x,y) = \frac{1}{\sqrt{2}}(-S_L(x,y) + S_R(x,y)) \tag{13}$$

  Now $\mathcal{S}_+(x,y)$ and $\mathcal{S}_-(x,y)$ are two new images. Plot them out. Their pixel values are the projections of the original data $(S_L(x,y), S_R(x,y))$ onto the eigenvectors, or they are the principal components of the data.
- (N): Calculate the 2×2 correlation matrix with elements

$$\mathsf{R}^{\mathcal{S}}_{ij} \equiv \langle \mathcal{S}_i \mathcal{S}_j \rangle \tag{14}$$

  with $i = +$ or $-$ and $j = +$ or $-$. Write out the correlation matrix

$$\mathsf{R}^{\mathcal{S}} = \begin{pmatrix} \mathsf{R}^{\mathcal{S}}_{++} & \mathsf{R}^{\mathcal{S}}_{+-} \\ \mathsf{R}^{\mathcal{S}}_{-+} & \mathsf{R}^{\mathcal{S}}_{--} \end{pmatrix} \tag{15}$$

  and verify that the off diagonal elements $\mathsf{R}^{\mathcal{S}}_{+-} = \mathsf{R}^{\mathcal{S}}_{-+} \approx 0$ in comparison to the diagonal elements. Reflect on what this means. Do a scatter plot of data points $(\mathcal{S}_+(x,y), \mathcal{S}_-(x,y))$ using all pixels (x,y), and observe how the matrix $\mathsf{R}^{\mathcal{S}}$ reflect the character of data in the scatter plot. Observe the variance of $\mathcal{S}_+(x,y)$ versus the variance of $\mathcal{S}_-(x,y)$. Which one has a larger variance?
- (O): Give gains g_+ and g_- to $\mathcal{S}_+(x,y)$ and $\mathcal{S}_-(x,y)$, respectively, to create the gain controlled images

$$\mathcal{O}_+(x,y) = g_+ \mathcal{S}_+(x,y) \tag{16}$$

$$\mathcal{O}_-(x,y) = g_- \mathcal{S}_-(x,y) \tag{17}$$

  Let the gain values be

$$g_+ = \frac{1}{\sqrt{\mathsf{R}^{\mathcal{S}}_{++}}}, \quad g_- = \frac{1}{\sqrt{\mathsf{R}^{\mathcal{S}}_{--}}}. \tag{18}$$

  Plot the two images $\mathcal{O}_+(x,y)$ and $\mathcal{O}_-(x,y)$.

- (P): Do a scatter plot of data $(\mathcal{O}_+(x,y), \mathcal{O}_-(x,y))$, and calculate the correlation matrix $\mathsf{R}^{\mathcal{O}}$ with matrix element

$$\mathsf{R}^{\mathcal{O}}_{ij} \equiv \langle \mathcal{O}_i \mathcal{O}_j \rangle \tag{19}$$

with $i = +$ or $-$ and $j = +$ or $-$. Write out the correlation matrix

$$\mathsf{R}^{\mathcal{O}} = \begin{pmatrix} \mathsf{R}^{\mathcal{O}}_{++} & \mathsf{R}^{\mathcal{O}}_{+-} \\ \mathsf{R}^{\mathcal{O}}_{-+} & \mathsf{R}^{\mathcal{O}}_{--} \end{pmatrix} \tag{20}$$

You should see that $\mathcal{O}_+(x,y)$ and $\mathcal{O}_-(x,y)$ are not correlated with each other, but have roughly the same variance. This is because we used the gains g_+ and g_- which are for whitening.

- (Q): Construct two new images $O_1(x,y)$ and $O_2(x,y)$ from $\mathcal{O}_\pm(x,y)$ as follows

$$O_1(x,y) = \frac{1}{\sqrt{2}}(\mathcal{O}_+(x,y) + \mathcal{O}_-(x,y)) \tag{21}$$

$$O_2(x,y) = \frac{1}{\sqrt{2}}(\mathcal{O}_+(x,y) - \mathcal{O}_-(x,y)) \tag{22}$$

and plot them out. Also, do a scatter plot of data $(O_1(x,y), O_2(x,y))$. You should see that $O_1(x,y)$ and $O_2(x,y)$ are not correlated with each other, but have roughly the same variance.

- (R): Construct the 2×2 correlation matrix with elements

$$\mathsf{R}^{O}_{ij} \equiv \langle O_i O_j \rangle \tag{23}$$

with $i = 1$ or 2, and $j = 1$ or 2, for the matrix

$$\mathsf{R}^{O} = \begin{pmatrix} \mathsf{R}^{O}_{11} & \mathsf{R}^{O}_{12} \\ \mathsf{R}^{O}_{21} & \mathsf{R}^{O}_{22} \end{pmatrix} \tag{24}$$

Relate this matrix with the scatter plot in (Q).

- (S): In step (J), you have the images $S_L(x,y)$ and $S_R(x,y)$. Add noise $N_L(x,y)$ and $N_R(x,y)$ to them, such that

$$S_L(x,y) \to S_L(x,y) + N_L(x,y) \tag{25}$$

$$S_R(x,y) \to S_R(x,y) + N_R(x,y) \tag{26}$$

At each pixel location (x,y), the noise $N_L(x,y)$ and $N_R(x,y)$ are independent of each other, and is a random number uniformly distributed in the range between $-N_{max}$ and N_{max}, i.e., $-N_{max} \leqslant N_L(x,y) \leqslant N_{max}$ and $-N_{max} \leqslant N_R(x,y) \leqslant N_{max}$. Noise in different pixels should be independent random numbers in this range. Choose $N_{max} = \sqrt{(\mathsf{R}^S_{11} + \mathsf{R}^S_{22})}/2$ (or you can choose a larger or smaller N_{max}). Plot out the noisy images $S_L(x,y)$ and $S_R(x,y)$. With these noisy images, repeat steps (B)-(R) and observe what happens and reflect on the result. You can play with the value of N_{max} and see different effects by different N_{max} values, and try to reflect why.

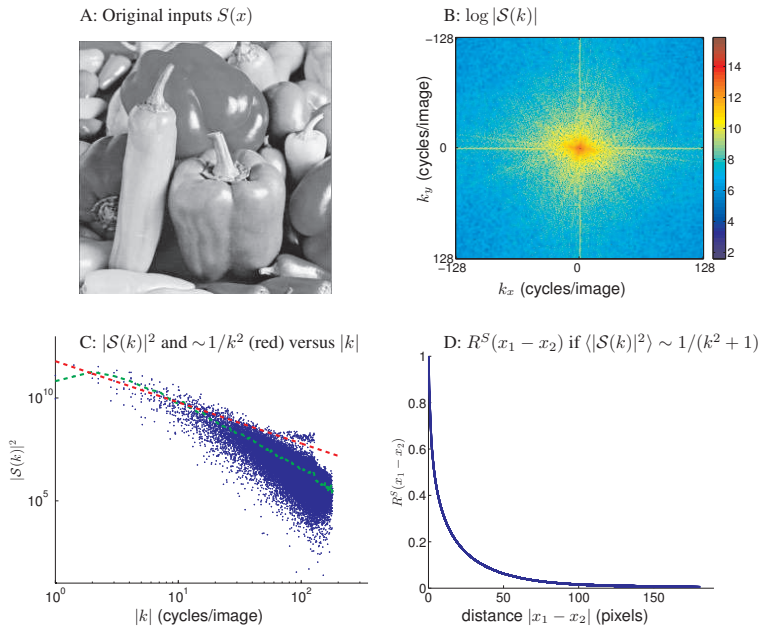

Figure 2: This is figure 3.18 in the textbook "Understanding vision: theory, models, and data". A: An example visual input $S(x)$ (also denoted as S_x) in space x. The image has 256×256 pixels. B: Fourier transform $\mathcal{S}(k)$ of image $S(x)$, visualized as $\log|\mathcal{S}(k)|$ versus the 2D spatial frequency $k = (k_x, k_y)$. In the text, $\mathcal{S}(k)$ is also denoted as $\mathcal{S}_k$. C: Power $|\mathcal{S}(k)|^2$ versus $|k|$ for input $S(x)$ in A; red line: $\sim 1/|k|^2$; green curve: the average of $|\mathcal{S}(k)|^2$ over a local range of k for the input image in A. D: Spatial correlation $R^S(x_1 - x_2)$ assuming $\langle|\mathcal{S}(k)|^2\rangle = 1/(k^2 + k_o^2)$ for low cut-off frequency $k_o = 1$ cycle/image.

4. Practice exercises for efficient spatial coding. This exercise is doing things similar to making plots in those in Figures 2, 3, and 4, which are Figures 3.18, 3.19, and Figure 3.22 of "Understanding vision: theory, models, and data". Please compare your plots with the corresponding ones in the figures to get some understanding.

   Please do the following:

   - (A): Take a photograph of a natural environment. This image is denoted as $S(x, y)$. Here x and y are the coordinates of the image pixel locations. For example $x = 1, 2, ..., 256$ and $y = 1, 2, ..., 256$. You may find some examples online, or one of your own photographs. An example is the image of peppers in Figure 2. If the original image is colored, remove the color. Plot out the image as a luminance image.
   - (B): Transform the image to make it zero mean for convenience. Hence, get the pixel value average $\hat{S} = \sum_{x,y} S(x, y)/N$, where N the total number of pixels in this image. Then make $S(x, y) \rightarrow S(x, y) - \bar{S}$. Now $S(x, y)$ has zero mean value across the image.
   - (C): Observe how neighboring pixels are correlated with each other in their luminance values. For example, if you have one pixel at (x, y) and another at $(x + d, y)$, these two

pixels are horizontally displaced by a distance d. Calculate

$$R^S(d) = \sum_{x,y} S(x,y)S(x+d,y)/N', \tag{27}$$

where N' is the total number of pairs of pixels that you include in your sum $\sum_{x,y}$. Note that some pixels (x,y) are too close to the edge of the images that $(x+d,y)$ in not included in the image. This $R^S(d)$ approximates the correlation as a function of d. Try to calculate for a number of different d values, and see how $R^S(d)$ decays with d. Take a moment to consider what this entails for efficient coding of such images. (Many programming languages have routines for calculating correlations, e.g., in Matlab, you have xcorr().)

- (D): If you have not yet done it in your previous exercise, calculate a Fourier power spectrum. Let us practice just a simplified version. Let L be the range of x values of your image pixels (x,y). Then, take Fourier frequency k as one of these discrete values,

$$k = n \cdot 2\pi/(L), \text{using } (L+1) \text{ integer values } n = 0, 1, 2, ..., L, \tag{28}$$

(this k is in the unit of radian/pixel) and calculate

$$\mathcal{S}_c(k) = \sum_{x,y} S(x,y)\cos(kx) \tag{29}$$

$$\mathcal{S}_s(k) = \sum_{x,y} S(x,y)\sin(kx). \tag{30}$$

Then the signal power for this Fourier frequency k is (up to a scale constant)

$$|\mathcal{S}(k)|^2 = [\mathcal{S}_c(k))^2 + [\mathcal{S}_s(k)]^2 \tag{31}$$

Plot out (use log-log plot to see the functional character more easily) $|\mathcal{S}(k)|^2$ as a function of k. For typical photographs of natural scenes, $|\mathcal{S}(k)|^2$ decays with k as $1/k^2$. To see the results clearly, use log scale on both the vertical and horizontal axes, like in the lower left panel in Figure 2.

You may like to average $|\mathcal{S}(k)|^2$ over multiple images (make all images the same size for convenience) to see the trend of $|\mathcal{S}(k)|^2$ versus k more clearly.

- (E): The steps in (D) only looked at Fourier frequencies for a Fourier wave that is vertical, so the frequency k is along the horizontal or x direction. However, the Fourier wave can be in any direction. So the frequency k should be a vector with a horizontal component k_x and a vertical component k_y, so that the magnitude $k = sqrtk_x^2 + k_y^2$. Then, for each image $S(x,y)$, which span horizontally in x and vertically in y, one can obtain its Fourier component $\mathcal{S}(k)$ for vector frequencies k. Usually, your programming language has a routine for Fourier transform, try to use it to get $\mathcal{S}(k)$. For example, Matlab has a function fft2() for (2-dimensional image) Fourier transform and

another function ifft2() for the inverse Fourier transform. In your matlab code, represent your image $S(x, y)$ as a 2-d matrix, let us call it S. Then fft2(S) should give you an output matrix which is $\mathcal{S}(k)$ as a function of k, with k_x and k_y. $\mathcal{S}(k)$ is usually expressed as a complex number with its real and imaginary components as real($\mathcal{S}(k)$) and imag($\mathcal{S}(k)$), and magnitude as abs($\mathcal{S}(k)$). Representing your $\mathcal{S}(k)$ also as a matrix, let us call it $\mathcal{S}$, then ifft2($\mathcal{S}$) does the inverse Fourier transform and it should give you back your original image S. Try it. However, to get the plots as in Figure 2B and 2C, you need to know what frequency k is associated with each matrix element from the output of fft2(S), this may depend on the convention of your programming language, and you need to dig it out. In matlab, it is done as follows.

```
%— Let us call the image Simage in your matlab code.
[N1, N2] = size(Simage); % N1 and N2 are the number of pixels spanning vertically and horizontally, so Simage is an N1 × N2 matrix.
SFourier = fft2(Simage); % SFourier is the Fourier components, it is also an N1 × N2 matrix,
kx = 0:N2-1; kx = min([kx; N2*ones(1, N2) - kx]); kx = ones(N1, 1)*kx; % Now kx is the kx value for the SFourier matrix elements.
ky = 0:N1-1; ky = min([ky; N1*ones(1, N1) - ky]); ky = ky'*ones(1, N2); % Now ky is the ky value for the SFourier matrix elements.
k = sqrt(kx.*kx + ky.* ky); % k is the magnitude of k for the SFourier matrix elements.
Simage_back = ifft2(SFourier); % inverse Fourier transform on SFourier, shoud give you the original image back.
figure(201); clf;
subplot(2, 3, 1); imagesc(Simage); axis square; colormap(gray); axis off; colorbar;
title('original image');
subplot(2, 3, 2); imagesc(log(abs(SFourier))); axis square; colormap(gray); axis off; colorbar;
title('log of magnitudes of the Fourier components');
subplot(2, 3, 3); imagesc(Simage_back); axis square; colormap(gray); axis off; colorbar;
title('After the inverse Fourier transform of the Fourier Components');
subplot(2, 3, 4); imagesc(kx); axis square; colormap(gray); axis off; colorbar; title('kx');
subplot(2, 3, 5); imagesc(ky); axis square; colormap(gray); axis off; colorbar; title('ky');
subplot(2, 3, 6); imagesc(k); axis square; colormap(gray); axis off; colorbar; title('magnitude of k');
```

- (F): in (D) and (E), you may find that your power spectrum, the $|\mathcal{S}(k)|^2$ versus k, does not decay with k in a way expected when k is very large. This may be because your image is noisy, so that $|\mathcal{S}(k)|^2$ include the power of the white noise. However, if your image is not noisy enough, or to practice seeing the effect of noise on $|\mathcal{S}(k)|^2$, let us make the image noisy by adding noise as follows

$$S(x, y) \to S(x, y) + N(x, y) \tag{32}$$

  in which $N(x, y)$ is a noise image. For each pixel (x, y), make $N(x, y)$ a zero-mean random nubmer $-N_{max} \leqslant N(x, y) \leqslant N_{max}$ with a maximum magnitude N_{max}. Make N_{max} large enough so that the noise is obvious. Plot out this noisy image. You can see an example of such a noisy image in Figure 4A. Once you get this noisy image, repeat (D) to plot $|\mathcal{S}(k)|^2$ versus k, and see whether $|\mathcal{S}(k)|^2$ versus k behaves like the blue curve in Figure 4B, and try to understand the result.

- (G): Going from the image S to its Fourier components $\mathcal{S}$ is the principal component transform on the image. For efficient coding, we need to give gain control $g(k)$ to each component $\mathcal{S}(k)$ as the second step in the efficient coding recipe (see Figure 3. This gives

$$\mathcal{O}(k) = g(k)\mathcal{S}(k). \tag{33}$$

  This $g(k)$ should be

$$g(k) = (|k| + |k_o|) \cdot \exp(-|k|^2/k_{low}^2) \tag{34}$$

in which k_o and k_{low} are parameters. The first factor $(|k| + |k_o|)$ is a whitenning filter, useful for low input noise conditions. Typically k_o is very small, you could make $k_o = 0$ or just non-zero small number. The second factor $\exp(-|k|^2/k_{low}^2)$ is a low-pass filter, to avoid sending too much noise when signal-to-noise is too low for large $|k|$. Try to make k_{low} at a $|k|$ value where the signal and power power are comparable, this is where $|\mathcal{S}(k)|^2$ stop the trend of decaying with $|k|$.

Then, you do the inverse Fourier transform on $\mathcal{O}(k)$ and plot the results. Play with different values for k_{low} and see what you get as the results. Compare with the plots in Figure 4CDE.

- (H): if you feel comfortable, try to do the inverse Fourier transform on $g(k)$ and get the shape of the receptive field for the encoding, and observe how this shape changes with your choice of k_{low}. As the image noise becomes noiser, the receptive field size increases and the inhibitory surround weakens.

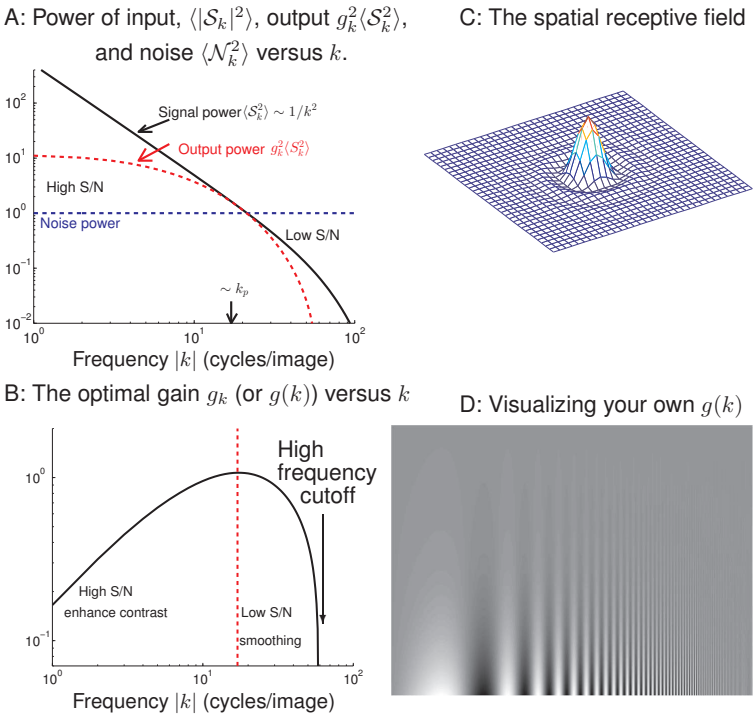

Figure 3: This is figure 3.19 in the textbook "Understanding vision: theory, models, and data". Illustration of how input statistics determine optimal encoding in space. A: The power spectra of the input signal $\langle\mathcal{S}_k^2\rangle = 500/(|k|^2 + k_o^2) \cdot (M(k))^2$, the output $g^2(k)\langle\mathcal{S}_k^2\rangle$, and white noise $\langle\mathcal{N}_k^2\rangle = 1$, with $k_o = 0.1$ and $M(k) = \exp[-(k/100)^2]$ is the modulation transfer function of the eye. At $k = k_p$, $\langle\mathcal{S}_k^2\rangle = \langle\mathcal{N}_k^2\rangle$. B: The optimal $g(k)$ (in arbitrary units) by equation (**??**), given input $\langle\mathcal{S}_k^2\rangle/\langle\mathcal{N}_k^2\rangle$ in A, when $\frac{2\lambda}{(\ln 2)\langle N_o^2\rangle} = 60$. The actual curve plotted in B is the effective gain $\tilde{g}(k) = g(k)M(k)$ (which should be proportional to the contrast sensitivity curves observed in experiments). Note that $g(k)$ peaks around $k = k_p$, and $g(k) \propto k$ for small k. C: The shape of the receptive field $K(x) \sim \int \tilde{g}(k)e^{ikx}dk$, determined by $\tilde{g}(k)$. The size of the receptive field should be roughly $1/k_p$. All frequencies k are in units of cycles/image. D: An image of gratings with spatial frequency k increasing from left to right, and contrast increasing from top to bottom. The boundary between invisible and visible contrast regions demonstrates the human contrast sensitivity function $g(k)$. Picture courtesy of Mark Georgeson, 2013.

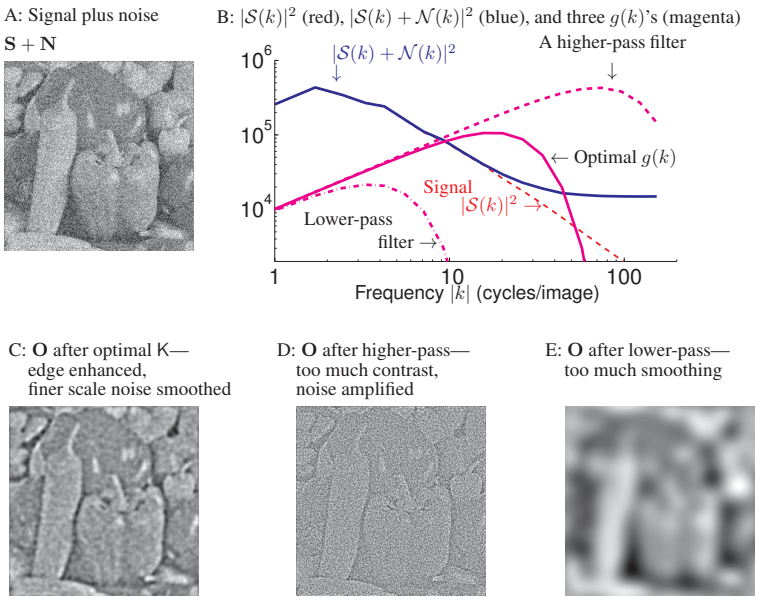

Figure 4: This is figure 3.22 in the textbook "Understanding vision: theory, models, and data". Signal transform by optimal and non-optimal codings of spatial visual inputs. A: Image $\mathbf{S}$ with added white noise $\mathbf{N}$. B: Signal power $|\mathcal{S}(k)|^2$ (red-dashed, partly covered by the blue curve), total input power $|\mathcal{S}(k)+\mathcal{N}(k)|^2$ (blue), and three filter sensitivity functions $g(k)$ (magenta) as functions of frequency k. (The vertical axis has an arbitrary scale.) Solid magenta curve: optimal sensitivity $g(k)$, which peaks near k_p where $|\mathcal{S}(k) + \mathcal{N}(k)|^2$ starts to deviate from $|\mathcal{S}(k)|^2$ significantly. For comparison, sensitivity curves with higher and lower-pass characteristics are shown as magenta dashed and magenta dash dotted, respectively. C: Response $\mathbf{O} = \mathsf{K}(\mathbf{S} + \mathbf{N})$ after optimal filtering K with the optimal sensitivity curve $g(k)$. Here, contrast in the image (i.e., edges) is enhanced for low k, where $g(k)$ increases with k; but the image is smoothed for high k, where $g(k)$ decreases with k, to avoid transmitting too much input noise at finer spatial scales. D and E: Outputs $\mathbf{O}$ when the filters are higher and lower pass, respectively, as depicted in B. Gray scale values shown in A, C, D, and E are normalized to the same range.