Solutions For Homework #2

Problem 1:[10 pts]

(a) The file *hawaiidem* consists of $927 \times 375 = 681345$ 1-byte elements. The following MATLAB code arranges these elements into an image matrix of 927 lines by 735 columns

```
fid = fopen('hawaiidem','rb');
data = fread(fid, inf, 'uint8');
fclose(fid);
im = reshape(data, [735 927]); im = im';
```

Figure 1 displays a grayscale image of a Digital Elevation Model (DEM) of the Big Island of Hawaii.

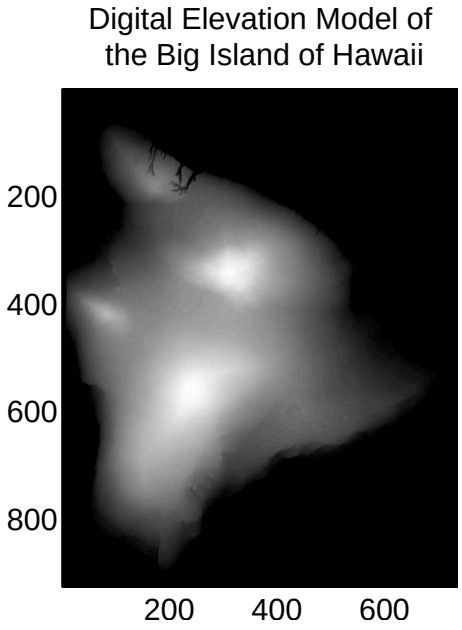


Figure 1:

The MATLAB code that displays this image is as follows

```
imagesc(im);
colormap gray;
caxis([0 255]);
axis image;
```

This code ensures that the grayscale display is set such that pixel value zero is assigned to pure black while pixel value 255 is set to pure white. Intermediate pixel values are assigned to shades of gray. Figure 2 shows a contour map of Hawaii. The two highest peaks, Mauna Loa and Mauna Kea, are shown in the figure. Evidently, Mauna Kea - at pixel (344,317) - has an altitude of 4160 meters, while Mauna Loa - at pixel (555, 234) - is an altitude of approximately 4140 meters. The horizontal distance, in pixels,

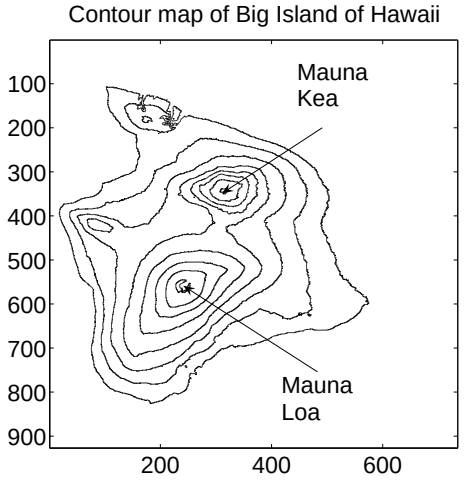


Figure 2:

between Mauna Kea and Mauna Loa is computed as follows:

$$\Delta = \sqrt{(344 - 555)^2 + (317 - 234)^2} = 226.74 \text{ pixels} \qquad (1)$$

Since the pixel spacing is 180 meters, then the horizontal distance between Mauna Loa and Mauna Kea is $180 \times 226.74 = 40,812$ meters or about 40.8 km.

(b) The shaded relief map is shown in Figure 3 We observe from Figure 3 that

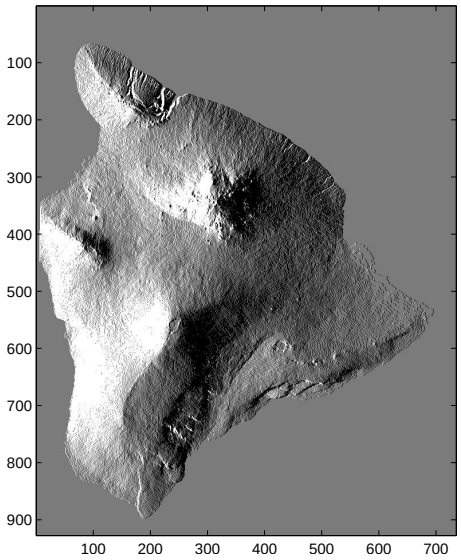


Figure 3:

the topography seems illuminated from the left. This is because of the difference equation we applied to the DEM, which is as follows

$$\text{DEM}_{shaded}(i,j) = \text{DEM}(i,j) - \text{DEM}(i,j+1) \tag{2}$$

that is, we take finite differences across the columns. We note that for topographic slopes that increase towards the right, the formula above yields positive numbers. In comparison, for slopes that decrease towards the left, the finite differences yield negative numbers. The positive slopes appear bright, while negative slopes show up as dark.

(c) The shaded relief map is shown in Figure 4 We observe from Figure 4 that the topography seems illuminated from the bottom. This is because of the difference equation we applied to the DEM, which is as follows

$$\text{DEM}_{shaded}(i,j) = \text{DEM}(i,j) - \text{DEM}(i-1,j) \tag{3}$$

Problem 2:[10 pts]

In this problem, we use the perspective projection equations given in the class notes to project the three-dimensional distribution of shaded relief intensities on

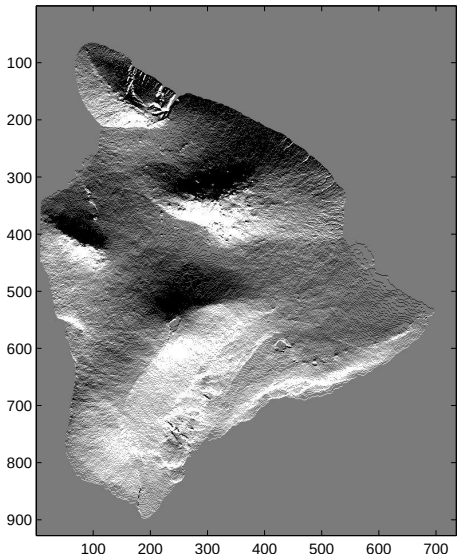


Figure 4:

a 2D plane. First, however, we need to define the (x, y, z) coordinate axis of the 3D object - in this case, topographic slopes from shaded relief - and the (u, v) coordinate axis of the image plane. Figure 5 defines these coordinate axes. From Figure 5 we observe that the $x - z$ plane coincides with the (u, v) image plane. Also shown is the location of the observer, located at height h above the $x - y$ (ground) plane, on the $y-$axis, i.e. at location $(0, -d, h)$. The equations for perspective projection are as follows

$$\begin{aligned} u &= \frac{dx}{y+d} \\ v &= h + \frac{d(z-h)}{y+d} \end{aligned} \tag{4}$$

Equation set 5 provides the appropriate mapping between a point (x, y, z) in object space and the corresponding point (u, v) in the image space (a plane). We calculate the (x, y, z) object coordinates from the pixel indices and elevations $(i, j, \text{DEM}(i, j))$ from the Digital Elevation Model (DEM) data of Hawaii we formed in Problem 1.

Since we are asked to provide two projections, 90 degrees apart, we will need to assign the $x-$ and $y-$ axes to either the row or column dimension of the DEM matrix. By attaching the $x-$ axis to the row dimension of the DEM matrix, in

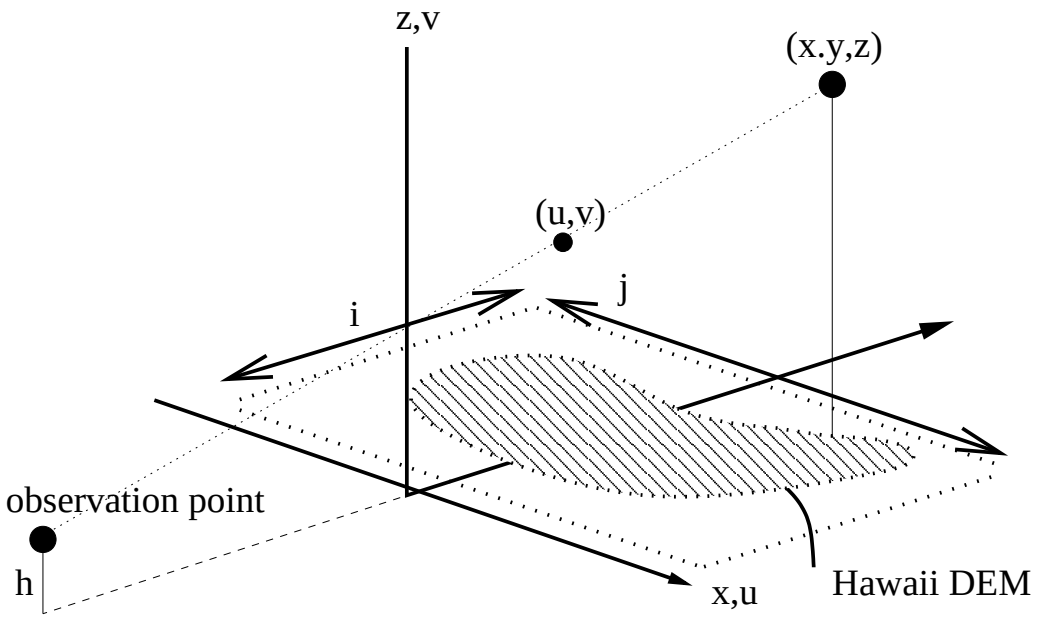


Figure 5: (x, y, z) coordinate axis of the object (shaded relief map of the Big Island) and (u, v) coordinate axis of the image plane

one case, and subsequently to the column axis in the other, we can perform the two perspective projections 90 degrees apart. We denote the row and column dimensions of the DEM matrix as i and j respectively. Note that the contents of the DEM matrix, the height values dem(i, j), are always assigned to the $z-$ axis. Thus, the relevant mappings between these various coordinate axes are

$$\text{Perspective Projection 1:} \quad (x, y, z) = (\Delta i, \Delta(j - \frac{735+1}{2}), 20 \times \text{DEM}(i, j)) \tag{5}$$

$$\text{Perspective Projection 2:} \quad (x, y, z) = (\Delta j, \Delta(i - \frac{937+1}{2}), 20 \times \text{DEM}(i, j)) \tag{6}$$

where $\Delta = 180$ meters, the stated pixel spacing. From the equations above and Equation 5, we can easily achieve the mapping

$$(i, j, \text{DEM}(i, j)) \rightarrow (u, v) \tag{7}$$

Finally, we need to compute the image plane indices (k', l') from the calculated (u, v) coordinates. This is simply done as follows

$$(k', l') = \lceil (\frac{u}{\Delta}, \frac{v}{\Delta}) \rceil \tag{8}$$

Note that, as stated above, (k', l') will not necessarily be positive as (u, v) can taken on negative values. Negative array indices are disallowed by most programming languages, but the fix is quite simple. One only needs to shift the image indices (k', l'),

$$k = k' + k_{min} + 1, \quad l = l' + l_{min} + 1 \tag{9}$$

The MATLAB code implementing the above is given at the back. Figure 6 and Figure 7 show the two projections, 90 degrees apart, of the three-dimensional distribution of shaded relief of Hawaii,

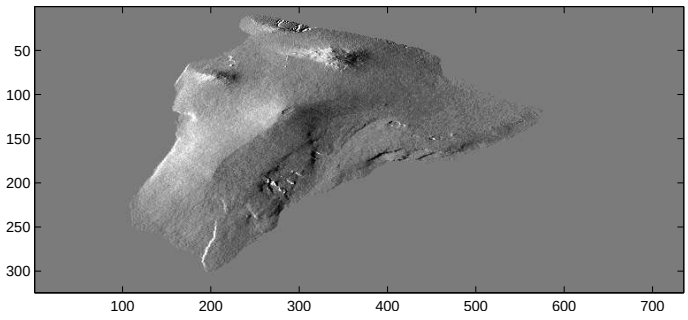


Figure 6: Perspective projection 1. $d = 120000$, $h = 100000$

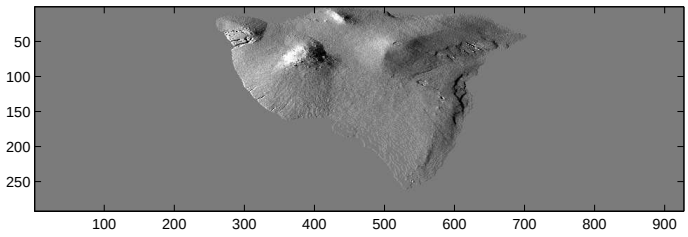


Figure 7: Perspective projection 2. $d = 120000$, $h = 100000$

Problem 3:[10 pts]

This problem differs from the previous one only in the definition of the imaging geometry. The geometry of pinhole camera projection is shown in Figure 8 This different geometry of course leads to a different set of projection equations,

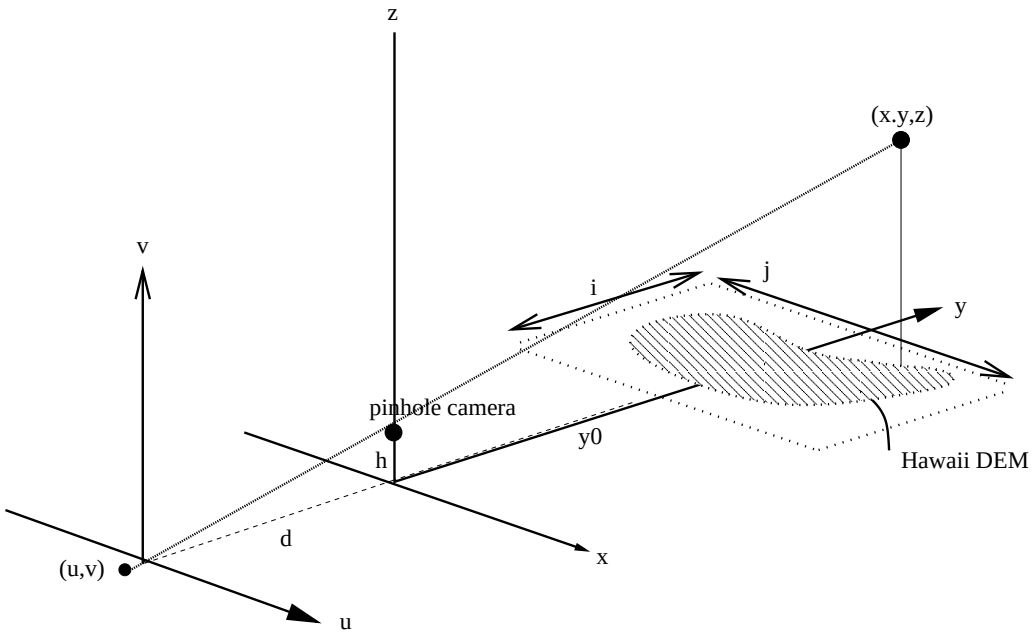


Figure 8:

$$u = \frac{-dx}{y + y_0} \tag{10}$$
$$v = h + \frac{-d(z-h)}{y + y0}$$

Equation set 11 is slightly different from the set of equations given in class notes, in that we have allowed for a vertical displacement of the pinhole. We see from Figure 8 that the pinhole is located at coordinates $(x, y, z) = (0, 0, h)$. Comparing Equation set 11 with Equation set 5, we note that pinhole projection requires an additional parameter y_0, which denotes the distance of the image plane from the location of the pinhole. This difference aside, we perform pinhole camera projection in exactly the same way as perspective projection in Problem 2. We apply the same mapping between DEM indices and elevations to (x, y, z) in object space,

$$\text{Pinhole Projection 1:} \quad (x, y, z) = (\Delta i, \Delta(j - \frac{735+1}{2}), 20 \times \text{DEM}(i, j)) \tag{11}$$

$$\text{Pinhole Projection 2:} \quad (x, y, z) = (\Delta j, \Delta(i - \frac{937+1}{2}), 20 \times \text{DEM}(i, j)) \tag{12}$$

where $\Delta = 180$ meters, and subsequently use Equation set 11 to find the corresponding image plane coordinates (u, v). Deriving image array indices (k, l)

from the calculated (u, v) coordinates proceeds analogously as in Problem 2. Figure 9 and Figure 10 show the two pinhole camera projections, 90 degrees apart, of the three-dimensional distribution of shaded relief of Hawaii, We note that the

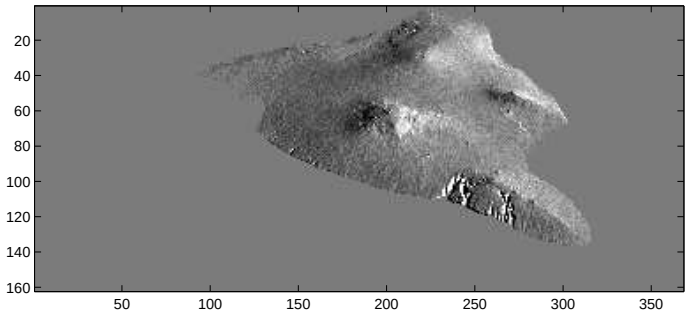


Figure 9: Perspective projection 1. $d = 600000$, $y_0 = 60000$, $h = 100000$

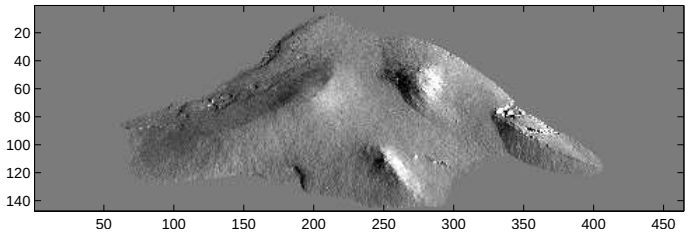


Figure 10: Perspective projection 2. $d = 60000$, $y_0 = 60000$, $h = 100000$

MATLAB implementation of the pinhole camera projection is analogous to the implementation of the perspective projection, as in Problem 2. The only difference is the set of equations used for projection.

Problem 4:[10 pts]

(a) We implement a Gaussian random number generator by adding together a large number of uniformily distributed random variables. This follows from the Central Limit Theorem, which claims that the probability distribution function (pdf) of the sum of independent random variables approaches, in the limit, a Normal distribution. We generate a number of random variables,

x_k, uniformily-distributed between 0 and 1 and form the sum as follows

$$Y' = \sum_{k=1}^{N} x_k; \tag{13}$$

Let us denote $<>$ as the statistical expectation operator. Since the random variables x_k are independent, we know that

$$\begin{aligned} < x_k >= 0.5 \quad &\Rightarrow \quad < Y' >= 0.5N \\ var(x_k) = 1/12 \quad &\Rightarrow \quad < Y' >= \frac{N}{12} \end{aligned} \tag{14}$$

where var() denotes variance. Thus, we see that the sum Y possesses a mean of 0.5N and variance $\frac{N}{12}$. Consequently,

$$Y = \frac{Y' - 0.5N}{\sqrt{N/12}} \tag{15}$$

yields a Gaussian-distributed random variable Y with zero mean and unit variance. In this problem, we sum a sample of 1000 instances of a uniformily-distributed random variable to produce one instance of a Gaussian r.v. We subtract the sample mean from that sum, and normalize the result by the sample standard deviation.

A complex Gaussian random variable c is defined as

$$c = x + jy \tag{16}$$

where x, y are independent, zero-mean, unit-variance Gaussian random variables. We apply our Gaussian random number generator twice to generate one instance of x and y. We repeat this 10000 times to generate a sample of 10000 complex Gaussian numbers, $\{c_k\}_{k=1}^{1}0000$.

We then form intensities $I_k = c_k c_k^* = |c_k|^2$ from the 10000-long sample of complex Gaussian numbers. We know that I is exponentially distributed and, therefore, its mean is equal to its standard deviation. Theoretically, the mean intensity is calculated as follows

$$< I >=< (x + jy)(x - jy) >=< x^2 > + < y^2 >= 2 \tag{17}$$

which allows us to form the pdf, $f_I(r)$ of intensities I,

$$f_I(r) = \lambda e^{-\lambda r}, \;\; \lambda = \frac{1}{< I >} = 0.5 \tag{18}$$

	mean	standard deviation
empirical	1.6698	1.6625
theoretical	2	2

We will compare this theoretically calculated distribution with the histogram of intensity values obtained from the 10000-long sample $\{I_k\}_{k=1}^{1}0000$. Figure 11 shows this comparison. Table compares the theoretical values for

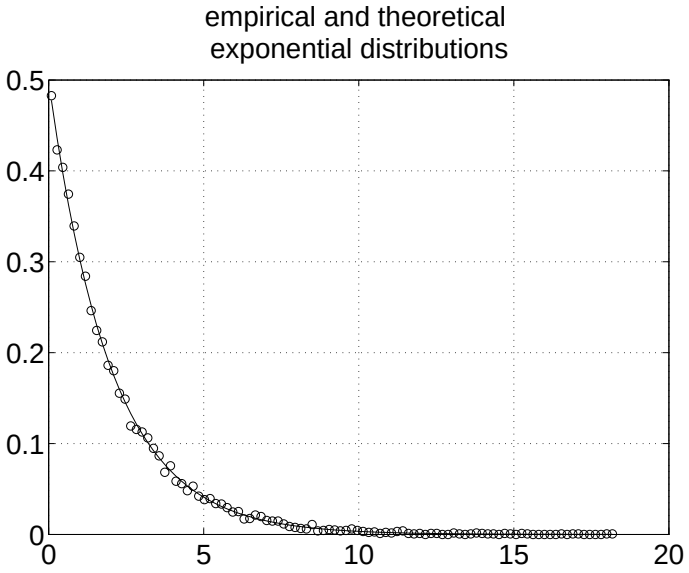


Figure 11:

the mean and standard deviation of the exponential distribution with the empirical values obtained directly from the 10000-long sample of intensities. We see that the values are in good agreement.

(b) We form the amplitudes $\{A_k\}$ from the 10000-long sample of complex Gaussian r.v. instances $\{c_k\}$ as follows

$$A_k = \sqrt{c_k c_k^*} = |c_k| \tag{19}$$

	mean	standard deviation
empirical	1.2508	0.6597
theoretical	1.2533	0.6551

We know that the amplitudes A_k are Rayleigh distributed. The amplitude random variable A is the square root of intensity random variable I from Part (a). The intensity I is exponentially-distributed, with paramater $\lambda = 0.5$, as we saw in Part (a). Hence, we can calculate the theoretical value for the mean of amplitude A as follows

$$\begin{aligned} < A >=< \sqrt{I} > &= \int_0^\infty \sqrt{r}\lambda e^{-\lambda r} \\ &= \frac{1}{2}\sqrt{\frac{\pi}{\lambda}} \end{aligned} \tag{20}$$

With $\lambda = 0.5$, we see that $< A >= 1.2533$. The theoretical standard deviation of the amplitude random variable A is calculated as follows

$$\begin{aligned} \text{var}(A) &= < A^2 > -(< A >)^2 \\ &= < I > - \left(\frac{1}{2}\sqrt{\frac{\pi}{\lambda}}\right)^2 \\ &= \frac{4-\pi}{4\lambda} \end{aligned} \tag{21}$$

Again, with $\lambda = 0.5$, we see that the theoretical standard deviation $= \sqrt{\text{var}(A)} = 0.6551$. Table compares the theoretical values for the mean and standard deviation of the Rayleigh distribution with the empirical values obtained directly from the 10000-long sample of amplitude. The theoretical value for the mean of the Rayleigh-distributed amplitudes now allows us to construct the pdf, as follows

$$f_A(a) = \frac{ae^{-\frac{a^2}{2s^2}}}{s^2} \quad \text{where } s = \sqrt{\frac{2}{pi}} < A > \tag{22}$$

Finally, we compare the theoretical Rayleigh pdf constructed above with the histogram of amplitudes obtained directly from the 10000-long sample

$\{A_k\}_{k=1}^{1}0000$. This comparison is shown in Figure 12

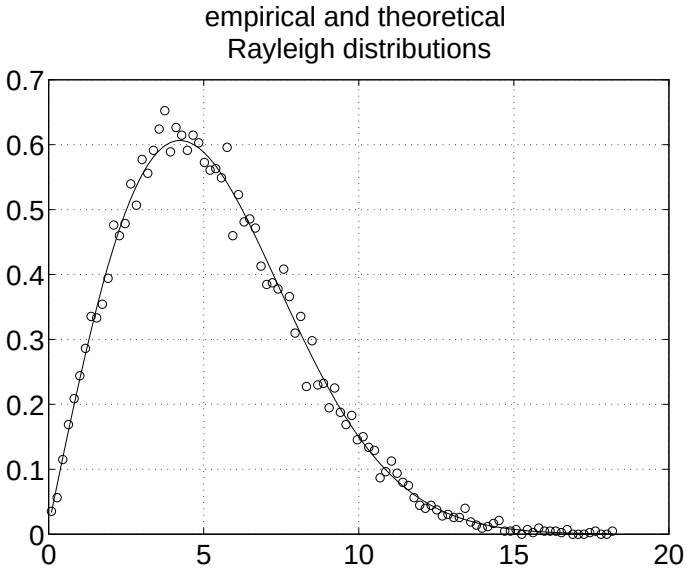


Figure 12:

Problem 5:[10 pts]

(a) Figure 13 shows the linearly transformed image *image1* from Problem Set 1. For details of the linear transformation, we refer the reader to the solutions to Problem Set 1. We denote the transformed image in Figure 13 by I.

Figure 13:

$I(i,j)$ refers to the ij pixel value of the image. We take this pixel value to represent the variance of a complex Gaussian random variable $c(i,j)$. The ij-th entry in a simulated 1-look coherent image L is then the magnitude squared of the complex random quantity $c(i,j)$.

$$\begin{aligned} L(i,j) &= |c(i,j)|^2 \\ &= \left(\sqrt{\frac{I(i,j)}{2}}x(i,j)\right)^2 + \left(\sqrt{\frac{I(i,j)}{2}}y(i,j)^2\right) \end{aligned} \tag{23}$$

where $x(i,j), y(i,j)$ are zero-mean, mutually independent, unit variance Gaussian-distributed random variables. We can use the Gaussian random number generator from Problem 4 to produce two arrays of random numbers (zero-mean, unit variance Gaussian distributed) x and y, with the same dimensions as the transformed image, from which we can simulate the 1-look coherent image by applying the equations in Equation set 24.

Figure 15 shows the resulting simulated 1-look coherent image. The grayscale display that produced this image is set such that pixel values at 255 or greater are mapped to pure white while values 0 or less are mapped to pure black.

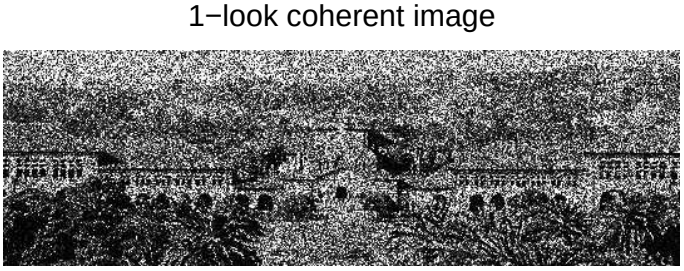


Figure 14:

(b) To simulate a 10-look coherent image, we average ten instances of 1-look coherent images of which Figure 14 in Part (a) is one realization

$$L_{10\ look}(i,j) = \sum_{k=1}^{10} L^{(k)}(i,j) \tag{24}$$

where $L^{(k)}$ denotes the k-th realization of a 1-look coherent image. The resulting image is shown in Figure 15 We know from Problem 4 that each

10−look coherent image

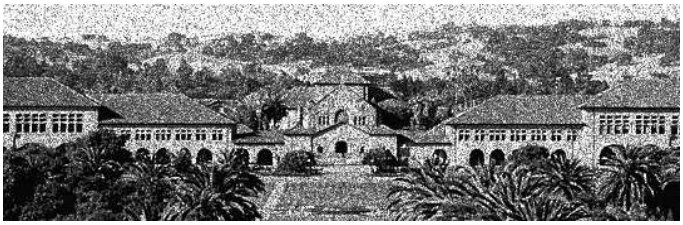
[ht] 

Figure 15:

1-look coherent image in the sum above is exponentially-distributed (it is an intensity image, formed from complex Gaussian r.vs). The expected value of this simulated 1-look coherent image is exactly the original transformed image in Figure 13, as can be seen from the following

$$\begin{aligned} < L(i,j) > &= < \left(\sqrt{\frac{I(i,j)}{2}} x(i,j) \right)^2 + \left(\sqrt{\frac{I(i,j)}{2}} y(i,j)^2 \right) > \quad (25) \\ &= I(i,j) \end{aligned}$$

The average of 10 exponentially-distributed intensity images above approximates the expected 1-look coherent image. This approximation improves with more realizations (draws) in the averaging. Nevertheless, even by averaging 10 realizations of simulated 1-look coherent images, the resulting image seems to approach the mean, the original image, as can be observed in Figure 15.

(c) From our results in Part (b), we would expect that averaging a large number of 1-look images would result in an image that looks quite similar to the original image in Figure 13. Analogous to Part (b), we average one hundred 1-look coherent images, the result of which is shown in Figure 16

(d) Theoretically, we would need an infinite number of realizations for the resulting to sum to converge on the statistical expectation of the exponentially-distributed speckled images. However, as we can see from Figure 16, the average of 100 draws looks quite similar to the original Figure 13.

Problem 6:[10 pts]

100−look coherent image

Figure 16:

(a) The highly speckled image of the Stanford quad, *lab2prob6data*, is shown in Figure 17. We average adjacent pixels to reduce speckle noise. We form a 2-look averaged image $L^{(2)}(i,j)$ as follows

$$L^{(2)}(i/2, j/2) = L^0(i,j) + L^0(i+1,j) + L^0(i,j+1) + L^0(i+1,j+1) \quad (26)$$

where $i = 0, 2, 4, ...2560 \;\; j = 0, 2, 4, ..., 2048$. The resulting 2-look intensity image is shown in Figure 18. This image is qualitatively less "grainy" than the original, Figure 17, which is a result of speckle reduction through averaging adjacent pixels. Furthermore, we see that 2-look image is one-quarter the size of the original. In Problem 5, we reduced speckle noise by averaging together multiple realizations of exponentially-distributed intensity images, the mean of which is the original intensity map. We noted that by averaging a sufficiently large number of realizations, the resulting intensity distribution approaches its statistical average. In most practical cases, however, we are usually given only one speckled intensity image, i.e. one realization. To improve detectability, then, we approximate the averaging of realizations (looks) by summing together adjacent pixels. In doing this, we assume that adjacent pixels represent independent draws of one random variable. So, in the above, we have assumed that pixels $L^0(i,j), L^0(i+1,j), L^0(i,j+1), L^0(i+1,j+1)$ are different draws of the same exponentially-distributed intensity pixel. A further consequence of averaging adjacent pixels to reduce speckle noise is the inevitable reduction in resolution, as is evident from the 2-look image in Figure 18.

(b) Figure 19 shows a 4-look image, formed by averaging blocks of 4 by 4 pixels together. We observe, particularly in areas of uniform intensity (the sky, the ground), that the variability of pixel values are greatly reduced. We

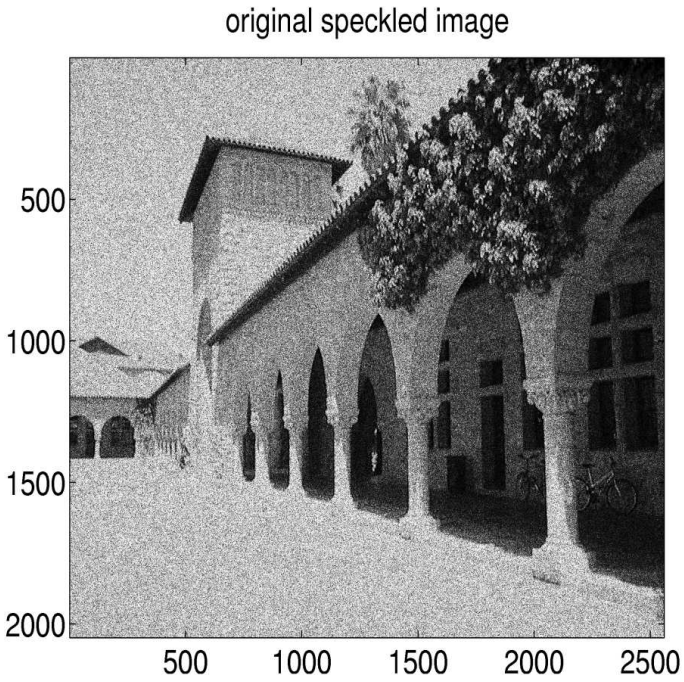


Figure 17:

also notice a blurred quality of the resulting image, indicative of the loss of resolution due to averaging.

MATLAB code for Problem 2

```
% EE 262 Spring 2005, Problem Set 2
%-----------------------------------%

% Problem 2
%-----------%

% read in data bytes
fid = fopen('hawaiidem','rb');
data = fread(fid, inf, 'uchar'); fclose(fid);

% arrange bytes into image matrix,
```

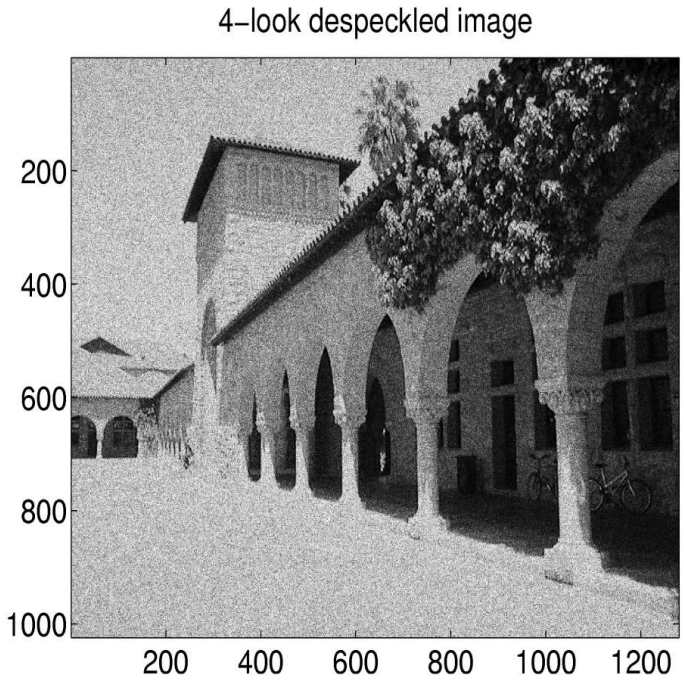


Figure 18:

```
% 927 rows x 735 columns
numrows = 927; numcols = 735;
im = reshape(data, [numcols numrows]); im = im.';

% covert data number (0-255) to heights (each data
% number represents 20 m of elevation)

im = im * 20;

% create shaded relief map of im

[im_shadedx1,im_shadedx2] = gradient(im,1);

% perspective projection
%-----------------------%

% create x1-x2-x3 grid for pixels. The two perspective projections
```

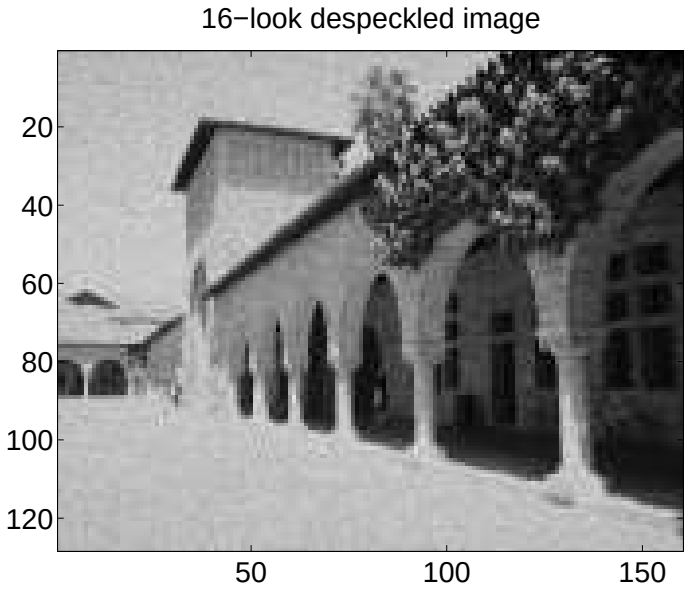


Figure 19:

```
% 90 degrees apart will be implemented by rotating the xyz
% coordinate axes.

% Projection 1: x axis = u axis = x1 axis,
%               y axis = x2 axis,
%               z axis = v axis = x3 axis;

delta = 180;     % pixel spacing in meters
[x1,x2] = meshgrid([0:numcols-1]*delta,[0:numrows-1]*delta);
x3 = im;

x = x1 - ((numcols+1)/2)*delta;
y = x2;
z = x3;

% perspective projection parameters
d1 = 120000;
h1 = 100000;
```

```
% perspective projection equations. Note: we loop from the
% points to points that are the nearest to projection plane
k = 1;
k = 1;
for rows = numrows:-1:1
  u(k,:) = x(rows,:).*d1./(y(rows,:) + d1);
  v(k,:) = h1 + (z(rows,:)-h1).*d1./(y(rows,:) + d1);
  k = k + 1;;
end

% convert u-v coordinates into image indices (i,j)
i = round(u/delta); j = round(v/delta);
j = j - min(min(j)) + 1; i = i - min(min(i)) + 1;

% create output image
im1 = zeros(max(max(i)),max(max(j)));
im1(sub2ind(size(im1),i,j))=im_shadedx1;


% Projection 2: x axis = u axis = x2 axis,
%               y axis = x1 axis,
%               z axis = v axis = x3 axis;

y = x1;
x = x2 - ((numrows+1)/2)*delta;
z = x3;

% perspective projection parameters
d2 = 120000;
h2 = 100000;

% perspective projection equations.Note: we loop from the
% points to points that are the nearest to projection plane
k = 1;
for cols = numcols:-1:1
  u(:,k) = x(:,cols).*d2./(y(:,cols) + d2);
  v(:,k) = h2 + (z(:,cols)-h2).*d2./(y(:,cols) + d2);
  k = k + 1;
```

```
end

% convert u-v coordinates into image indices (i,j)
i = floor(u/delta); j = floor(v/delta);
j = j - min(min(j)) + 1; i = i - min(min(i)) + 1;

% create output image
im2 = zeros(max(max(i)),max(max(j)));
im2(sub2ind(size(im2),i,j))=im_shadedx2;
```

MATLAB code for Problem 4

```
% EE 262 Spring 2005, Problem Set 2
%-----------------------------------%

% Problem 4
%----------%

% generate gaussian r.v by summing independent
% uniformily distributed random variables
% we choose to sum 1000 uniformily-distributed r.vs
% per realization of Gaussian r.v per channel (Real, Imag)

x = rand(1000,10000); x = sum(x);

% remove mean and normalize by standard deviation
 x = x - mean(x); real_rv = x./std(x);

x = rand(1000,10000); x = sum(x);

% remove mean and normalize by standard deviation
x = x - mean(x); imag_rv = x./std(x);

% form complex Gaussian rv
c = real_rv + sqrt(-1)*imag_rv;

mu_real = mean(real(c)); mu_imag = mean(imag(c));
```

```
var_real = (std(real(c)))^2; var_imag = (std(imag(c)))^2;
moment_3_real = mean( (real(c)-mu_real).^2 );
moment_3_imag = mean( (imag(c)-mu_imag).^2 );

% Part (a)
%---------%

% form intensities
I = c.*conj(c);

% compute histogram (100 bins)
[n1,x] = hist(I,100);

% normalize histogram counts so that
% Integral I dx ~= sum(I.*delta_x) is unity

n1 = n1./(sum(n1).*(x(2)-x(1)));

% intensity is exponentially-distributed. r.v.
% mean and standard deviation are equal, and equal to 2
% for zero-mean, unit variance complex Gaussian rvs
mean_intensity = 2; std_intensity = 2;

% form theoretical exponential pdf

lambda = (1./mean_intensity);
f1 = lambda.*exp(-x.*lambda);

% plot empirical and theoretical distribution
figure(1);plot(x,n1,'o');hold on; plot(x,f1,'r');grid on
title('empirical and theoretical exponential distributions');

% Part (b)
%---------%

% form amplitudes
A = sqrt(c.*conj(c));
```

```
% compute histogram (100 bins)
[n2,r] = hist(A,100);

% normalize histogram counts so that
% Integral A dx ~= sum(A.*delta_x) is unity

n2 = n2./(sum(n2).*(r(2)-r(1)));

% mean amplitude can be found form computing
% Integral_0^{Inf} Sqrt[r]*lambda*exp(-lambda*r) dr
% i.e. the first moment of Sqrt(Intensity) that is
% exponentially distributed. For zero-mean, unit variance
% complex Gaussian variates, Part (a) shows that the mean
% intensity has mean = 2  => lambda = 1/2
mean_amplitude = sqrt(pi/(1/2))/2;

% form theoretical Rayleigh pdf

s = sqrt(2/pi)*mean_amplitude;

f2 = r.*exp(-r.^2./2/s^2)/s^2;

% plot empirical and theoretical distribution
figure(2);plot(x,n2,'o');hold on; plot(x,f2,'r');grid on
title('empirical and theoretical Rayleigh distributions');
```