CHAPTER 14 ~ SUGGESTED SOLUTIONS (ODD)

14-1. ***Bad Pixel Correction*** The portion of a sensor's FPA shown below shows the raw counts (12-bit) for one frame of data, and contains output from a bright target against a dark (nighttime) background, noise, and other artifacts. There are two bad pixels noted: **H3** and **F6**. Apply a median filter to correct these inappropriate values.

	A	B	C	D	E	F	G	H	I	J
1	104	95	103	106	97	98	104	100	91	107
2	106	109	99	108	111	121	104	96	91	101
3	105	96	115	261	610	703	439	0	101	107
4	100	90	229	927	1908	2170	1450	504	123	93
5	105	94	401	1616	3113	3533	2419	904	161	97
6	91	105	398	1563	3033	4095	2356	869	163	109
7	105	99	207	847	1757	2017	1324	443	118	91
8	94	93	112	222	506	605	378	130	90	97
9	97	109	109	93	120	123	98	104	91	104
10	101	100	98	96	107	93	107	92	109	98

SUGGESTED SOLUTION: The median filter picks out the middle value of a list arranged in numerical order. Computer programming-wise, it is said to be faster to use integer arithmetic, so we want an odd number of values to sort. This is easily accomplished by selecting the bad pixel, itself, and its eight nearest neighbors. (The wisdom of including the bad pixels is often questioned, but see the following remarks.) For cells **H3** (apparently a dead pixel) and **F6** (a "happy" pixel), respectively ~

H3	**F6**
0	1324
91	1757
96	2017
101	2356
104	2419
123	3033
439	3113
504	3533
1450	4095

Thus we see that appropriate values to replace the two bad pixels are 104 and 2419 counts, respectively. Are these values the right ones? Don't know! But we can confidently say that they are closer to being correct than the bad values. We could say the same thing about *any* method we might use to replace the inappropriate values; it's just that the median filter method is simple to understand, easy to program, and reasonably fast.

14-3. ***Estimating Target Energy.*** The point target collected in Problem 14-1, and corrected in Problem 14-2 for dark current, was most likely a "superbolt" lightening flash lasting about 0.5 s. The data frame was thus only one of several. The advanced metsat sensor, in geosynchronous orbit, watched the event through 30 cm diameter optics in the 2.1 – 2.3 μm window band. The sensor's optical throughput was roughly 50% to a CMOS FPA with a 40% efficiency; the quantum wells per pixel were designed to be about 80,000 electrons deep. From this information, estimate the lightening bolt's radiant energy output (in band).

SUGGESTED SOLUTION: This calls for our dusting off the end-to-end equation from Chapter 10 and applying one new wrinkle, namely summing the counts on all the pixels that represent energy from the target:

$$\tilde{N}_{TGT} = \sum \tilde{N}_{PIX} = \sum_{PIXELS} A/D \left\{ \int_{BANDPASS} \frac{I_\lambda}{R^2} \tau_{ATM} A_R \tau_{OPT} \frac{\eta \mathrm{F}}{hc/\lambda} d\lambda \, \Delta t_{INT} \right\}.$$

This equation we approximate in the usual way ~

$$\tilde{N}_{TGT} \approx \sum_{PIXELS} A/D \left\{ \frac{\langle I_\lambda \rangle}{R^2} \overline{\tau_{ATM}} A_R \overline{\tau_{OPT}} \frac{\overline{\eta}\,\overline{\mathrm{F}}}{hc} \bar{\lambda} \, \Delta\lambda \, \Delta t_{INT} \right\}.$$

Since we are interested in comparing energy collected to the energy of a known source, examine the quantity $\langle I_\lambda \rangle \Delta\lambda \, \Delta t_{INT}$, and apply the definitions of power and what we mean by a spectral radiometric quantity:

$$\langle I_\lambda \rangle \Delta\lambda \, \Delta t_{INT} \approx \left\langle \frac{\Phi_\lambda}{4\pi} \right\rangle \Delta\lambda \, \Delta t_{INT} \approx \left\langle \frac{\Delta E_\lambda}{4\pi \Delta t} \right\rangle \Delta\lambda \, \Delta t_{INT} \approx \left\langle \frac{\Delta E/\Delta\lambda}{4\pi \Delta t} \right\rangle \Delta\lambda \, \Delta t_{INT} \approx \frac{\langle \Delta E \rangle}{4\pi}.$$

That is, our approximate solution reduces to ~

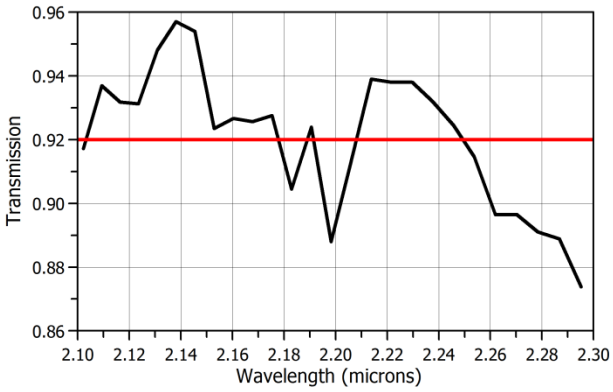
$$\tilde{N}_{TGT} \approx \sum_{PIXELS} A/D \left\{ \frac{\langle \Delta E_{PIX} \rangle}{4\pi R^2} \overline{\tau_{ATM}} A_R \overline{\tau_{OPT}} \frac{\overline{\eta}\,\overline{\mathrm{F}}}{hc} \bar{\lambda} \right\}.$$

Now for a little hand-waving. We believe that a CMOS FPA essentially uses 100% of its surface area for capturing photons, so $\mathrm{F} \approx 1$ for all pixels. BUT, we will also suppose the sensor's optics are nearly diffraction limited, resulting in only 84% of the target's image energy being within the pixels we have found. Thus our sensor equation can be solved as ~

$$\langle \Delta E_{TGT} \rangle \approx \frac{4\pi hcR^2}{0.84 \overline{\tau_{ATM}} A_R \overline{\tau_{OPT}} \overline{\eta} \bar{\lambda}} D/A \left\{ \sum_{PIXELS} \tilde{N}_{PIX} \right\}.$$

This is for the one frame of collected data, of course, and is only for the bandpass of the sensor. (Another little piece of hand-waving is the last term ~ it can be either the conversion of each pixel's digital output to electrons and then sum, or the other way around: sum, then convert.)

To start filling in the numbers, let's suppose that the distance from sensor to target is on the order of 40,000 km – something greater than geosynchronous altitude above the equator, but less than looking at the North Pole. The atmospheric transmission in the 2.1 – 2.3 μm band is good, and averages to about 0.92 (see plot at right). The collecting aperture is



$$A_R = \frac{\pi D^2}{4} = \frac{(\pi)(0.3\,\mathrm{m})^2}{4} \approx 0.071\,\mathrm{m}^2 .$$

For the sum of the counts on the pixels, we refer back to Problem 14-2 where we removed dark current from the collection (no background was assumed to be present in the nighttime scene). We found 40 or 41 pixels with what we believed to be target energy after also accounting for random noise. There is still the matter of the two pesky bad pixels, but applying the same method as Problem 14-1, we replace them with estimates using the median filter, as shown here, noting the "dead" pixel has no apparent target energy on it ~

	A	B	C	E	D	F	G	H	I	J
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	19	0	0	0	0
3	0	0	0	166	502	598	347	0	0	0
4	0	0	135	825	1816	2068	1360	398	29	0
5	0	0	304	1525	3018	3436	2325	808	58	0
6	0	0	292	1468	2937	2325	2248	761	61	0
7	0	0	113	744	1665	1915	1225	353	0	0
8	0	0	20	113	412	503	272	40	0	0
9	0	0	0	0	18	23	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

The sum of counts on target pixels is therefore $\sum_{PIXELS} \tilde{N}_{PIX} \approx 37,245$. We recall from the previous problem, however, that the noise component of each pixel – after subtraction of the dark frame – is about ± 8.5 counts per pixel, up to at most ± 15 counts. This being the case, when we add together the outputs of 40 pixels, we expect that our summed result may have an error (noise) of approximately

$$\mathcal{N}_{TGT} \approx \pm\sqrt{40}\cdot 8.5 \approx \pm 54 \text{ counts.}$$

Furthermore, since the output of the focal plane is reported to 12-bit depth (2^{12} = 4096), evident from the "happy" pixel, and the pixel well depth is given to be 80,000 electrons, the "digital-to-analog" reverse conversion is therefore $D/A \rightarrow \frac{80,000}{4096}$, but with a "quantization

error" of approximately 19 or 20 electrons (one bit) per pixel. Assuming a random distribution of counts in individual pixels (i.e., number of electrons high or low in the top-most bin), taken together the total of the summed pixel output may have an expected quantization error of

$$\Delta N \approx \pm\sqrt{40}\cdot 19.5 \approx \pm 123 \text{ electrons} \approx \pm 6 \text{ counts}.$$

Thus we see that the uncertainty in our Pixel Sum is dominated by the focal plane noise more than the quantization error, although it amounts to only about $\frac{54}{37,245} \approx 0.15\%$ because we have a very strong target signal in this Problem. (We comment here that the uncertainty introduced by fixing the bad pixels may, in fact, be the largest source of error, although we have no way of knowing that.)

To complete the calculation , we now plug in the numbers:

$$\left\langle \Delta E_{TGT} \right\rangle \approx \frac{(4\pi)\left(6.63\times10^{-34}\,\text{J}\cdot\text{s}\right)\left(3\times10^{8}\,\text{m}\cdot\text{s}^{-1}\right)\left(4\times10^{7}\,\text{m}\right)^{2}}{(0.84)(0.92)\left(0.071\,\text{m}^{2}\right)(0.5)(0.4)\left(2.2\times10^{-6}\,\text{m}\right)}\left[\frac{80,000}{4096}\right]\{37,245\} \approx 1.2\times10^{5}\,\text{J}.$$

14-5. ***Principal Component Analysis***[1] Spreadsheet file Chapter 14 ~ Suggested Problems DATA gives ten seconds of data collected on two pixels, called xi and yi, looking at a constant background. (i is an index.) The variation in the data is probably due to sensor jitter and some noise. Apply Principal Components Analysis to find the nominal background.

SUGGESTED SOLUTION: The computational method for calculating Principal Components (PCs) is not given in the text, so the following is a step-by-step procedure for two-dimensional data. Since this is a two-dimensional problem, we will find two PCs. The student will find computerized techniques for handling greater dimensional data in other books on multivariate statistics, as well as numerous web sites.

1. Find the average, or mean, of variables (pixel values) xi and yi. Call them X and Y (at the bottoms of their respective columns on the Suggested Solutions (ODD) worksheet).
2. Calculate the *variance* (difference) between each xi and X, and between each yi and Y. (See Columns E and F on the worksheet.)
3. Calculate the squares of the variances, $(xi - X)^2$ and $(yi - Y)^2$, and also the *covariance* $(xi - X)(yi - Y)$. (See Columns H, I, and J on the worksheet.)
4. Find the sums of $(xi - X)^2$, $(yi - Y)^2$, and $(xi - X)(yi - Y)$ (at the bottoms of their respective columns).
5. Note that this data set has $n = 101$ entries (for times $t = 0.0(0.1)10.0$), so divide the three sums by $n - 1 = 100$ (below the sums).

[1] Unfortunately the calculation of principal components for an $n \times n$ focal plane – or section thereof – is too difficult a computational problem to assign as homework. This problem considers only a two pixel array to demonstrate the method. The student is encouraged to dig into any modern book on multivariate statistics to learn the full implementation and to explore the computer algorithms necessary to make it tractable.

We have now found the covariance coefficients among the variables: x with itself, y with itself, and x with y. (Note that the covariance of x with y, c_{xy}, is the same as the covariance of y with x, c_{yx}, because the multiplication is commutative.) Assuming the student is familiar with matrix, or linear, algebra, these may be expressed in a *covariance matrix* ~

$$\mathbf{C} = \begin{pmatrix} c_{xx} & c_{xy} \\ c_{yx} & c_{yy} \end{pmatrix} = \begin{pmatrix} 32.574 & 32.365 \\ 32.365 & 34.013 \end{pmatrix}.$$

6. Find the characteristic values (eigenvalues) of the covariance matrix; that is, find values of λ such that $\mathbf{CX} = \lambda\mathbf{X}$, where $\mathbf{X}$ is a general two-pixel vector $\mathbf{X} = \begin{pmatrix} x \\ y \end{pmatrix}$:

$$(\mathbf{C} - \lambda\mathbf{I})\mathbf{X} = \begin{pmatrix} c_{xx} - \lambda & c_{xy} \\ c_{yx} & c_{yy} - \lambda \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 0.$$

To solve this equation with anything other than the trivial solution x = y =0, the coefficient matrix must be "singular". That is, its determinant (called the "characteristic equation") must be equal to zero:

$$\det \begin{pmatrix} c_{xx} - \lambda & c_{xy} \\ c_{yx} & c_{yy} - \lambda \end{pmatrix} = (c_{xx} - \lambda)(c_{yy} - \lambda) - c_{xy}c_{yx}$$

$$= \lambda^2 - (c_{xx} + c_{yy})\lambda + (c_{xx}c_{yy} - c_{xy}c_{yx}) = 0.$$

Using the standard quadratic formula, the solutions (calculated in the worksheet) are

$$\lambda = \frac{(c_{xx} + c_{yy}) \pm \sqrt{(c_{xx} - c_{yy})^2 + 4c_{xy}c_{yx}}}{2} \approx 65.667 \text{ and } 0.920\,.$$

These characteristic values tell us the relative weights, or importance, of the PCs. Obviously, one of the two components for this problem dominates the other; it is the *principal* component of the data set, giving the most significant relationship between the data dimensions (the two pixels).

7. Find the characteristic vectors (**E** for eigenvectors) for the characteristic values:

For $\lambda_1 \approx 65.667$:

$$\begin{pmatrix} 32.574 - 65.667 & 32.365 \\ 32.365 & 34.013 - 65.667 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} -33.093x_1 + 32.365y_1 \\ 32.365x_1 - 31.654y_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

These two equations are, in fact, the same (they are not linearly independent), so there is no unique solution to them. The best we can do is

$$y_1 = \frac{33.093}{32.365} x_1\,.$$

The choice of x_1 is thus arbitrary, and a reasonable choice is $x_1 = 1$. It is also common to choose values such that $\sqrt{x^2 + y^2} = 1$, i.e., it is a unit vector. Thus ~

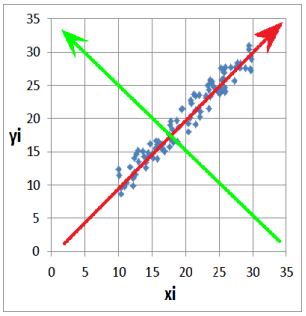
$$\mathbf{E}_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1.022 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 0.691 \\ 0.707 \end{pmatrix}$$

Similarly for $\lambda_2 \approx 0.920$:

$$\begin{pmatrix} 32.574 - 0.902 & 32.365 \\ 32.365 & 34.013 - 0.902 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 31.672x_2 + 32.365y_2 \\ 32.365x_2 + 33.111y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

$$y_2 = \frac{-31.672}{32.365} x_2,$$

$$\mathbf{E}_2 = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -0.979 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 0.715 \\ -0.699 \end{pmatrix}$$



Having found the characteristic vectors, we illustrate them at right, plotted atop the data set. We clearly see that our original data set has the most variability along the $\mathbf{E}_1$ direction (red), while there is very little variation in the $\mathbf{E}_2$ direction (green). Of course, we may have surmised this already from the fact that $\lambda_1 \approx 71.4\lambda_2$. The impact of this is that our data set can be *principally* represented by reducing it to only the *one* dimension. Since the other dimensional variation is small, leaving it out will not represent much loss of information.

8. Form the transformation matrix that will take you from the original data set to a new one composed of the PCs. The transformation matrix is composed of the characteristic vectors, arranged in order of importance in columns from left to right:

If you want to retain all of the variability in the data, then for this two-dimensional problem, the transformation matrix is

$$\mathbf{T}_2 = \begin{pmatrix} \mathbf{E}_1 & \mathbf{E}_2 \end{pmatrix} = \begin{pmatrix} 0.691 & 0.715 \\ 0.707 & -0.699 \end{pmatrix}$$

But if you want to reduce the data to one dimension, eliminating the second, the transformation matrix is

$$\mathbf{T}_1 = \begin{pmatrix} \mathbf{E}_1 & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0.691 & 0 \\ 0.707 & 0 \end{pmatrix}$$

9. Take the *transpose* of the transformation matrix, which is an arrangement of the characteristic vectors in horizontal rows instead of vertical columns:

$$\mathbf{T}_2^{\mathrm{T}} = \begin{pmatrix} 0.691 & 0.707 \\ 0.715 & -0.699 \end{pmatrix} \quad \text{and} \quad \mathbf{T}_1^{\mathrm{T}} = \begin{pmatrix} 0.691 & 0.707 \\ 0 & 0 \end{pmatrix}.$$

10. Finally, calculate your data set transformed into its PCs by [matrix] multiplying[2] your transposed transformation matrix by the original data vectors (represented as column vectors):

$$\mathbf{X}_{PC} = \mathbf{T}^{\mathrm{T}}\mathbf{X}\,.$$

For example, if we want to retain all of the information and keep the data set two-dimensional, the first data point, $\mathbf{X} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 15.138 \\ 14.013 \end{pmatrix}$, becomes

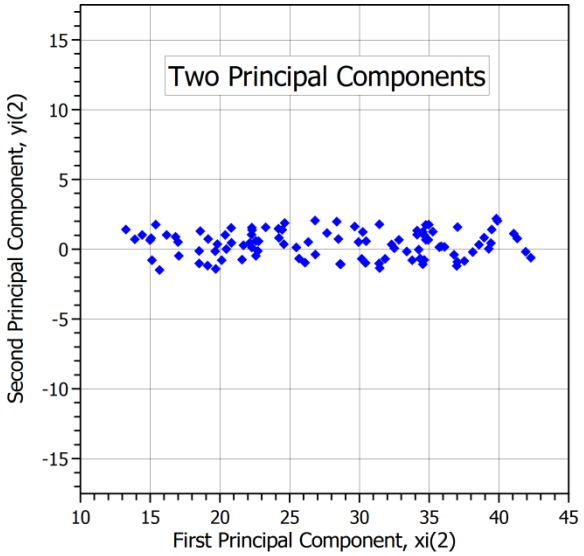
$$\mathbf{X}_{PC}^{(2)} = \begin{pmatrix} x_1^{(2)} \\ y_1^{(2)} \end{pmatrix} = \begin{pmatrix} 0.691 & 0.707 \\ 0.715 & -0.699 \end{pmatrix}\begin{pmatrix} 15.138 \\ 14.013 \end{pmatrix} = \begin{pmatrix} 20.368 \\ 1.029 \end{pmatrix}$$

where the superscript "(2)" reminds us it is the two-PC transformed data point.

Continuing with the example, the first data point transformed into ONE PC, which is the lion's share of information, is

$$\mathbf{X}_{PC}^{(1)} = \begin{pmatrix} x_1^{(1)} \\ 0 \end{pmatrix} = \begin{pmatrix} 0.691 & 0.707 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 15.138 \\ 14.013 \end{pmatrix} = \begin{pmatrix} 20.368 \\ 0 \end{pmatrix}.$$

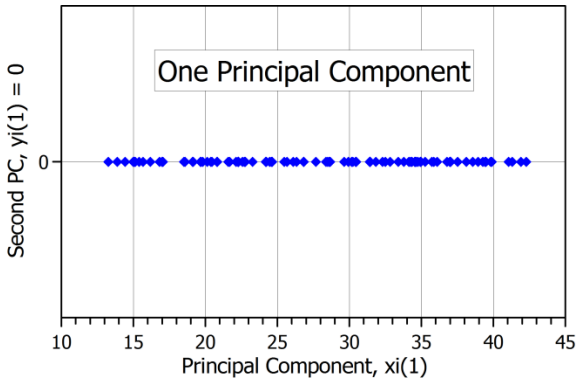
Calculation of the two-dimensional and one-dimensional PCs for this problem's data set are in the worksheet, columns O and P, and column R, respectively. (Note also the values shown in column S, which will be discussed below.)



Our two-dimensional PCs are plotted at left, where we show them to the same scale on the two axes. We note that our data have roughly the same appearance as our original, except for being rotated to the new axes, and the first PC shows the majority of the variation in the data, ostensibly $\dfrac{65.667}{65.667 + 0.920} \approx 98.6\%$ of it, while the second PC is the other 1.4%. This is clear justification that we could neglect the second PC and just choose to represent our data set with only a single PC.

[2] Recall when multiplying matrices, the number of columns of the first matrix must be the same as the number of rows of the second matrix. Here, for example, the first matrix is 2 × 2, and the second – a column vector – is 2 × 1. The result has the number of rows as the first matrix and the number of columns of the second. 2 × 1 in this case.

Before we show the one-dimensional PC, the student has the right to ask "what has become of the other dimension?" Since we have reduced the dimensionality of our data from two to one, apparently, all yi(1) =0 (shown in column S in the spreadsheet). Accordingly, we can now show a plot of the one-dimensional PC at right. Here we see all the variation in the collected data in a single dimension, but we understand that we have lost, at most about 1.4% of the information we may have worked so hard to collect.



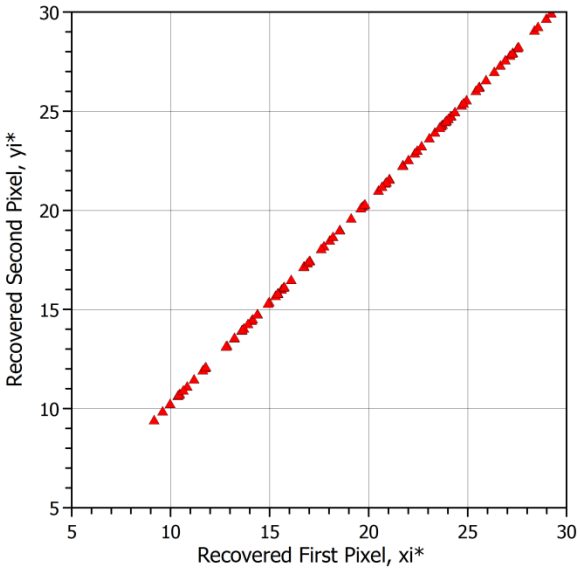
OK, so now are we done with this demonstration of PCs problem? Not quite … we now need to go back to "what are the values we're supposed to use on the pixels?" That is, since we have two real pixels, we need real numbers to put on them. To do this step, we return to our PC transformation in Step 10, above, and "undo" it by left-multiplying by the inverse of the transpose of the transformation matrix:

$$\mathbf{X}^{*} = \left(\mathbf{T}^{\mathrm{T}}\right)^{-1}\mathbf{T}^{\mathrm{T}}\mathbf{X} = \left(\mathbf{T}^{\mathrm{T}}\right)^{-1}\mathbf{X}_{PC} = \mathbf{T}\mathbf{X}_{PC}\,.$$

where we are using the notation **X*** to indicate recovered data from PC-reduced data. The last step in this equation comes from the fact that our transformation matrix was formed from orthogonal characteristic vectors (eigenvectors), thus the inverse of the transpose (Step 9) is the original transformation matrix itself (Step 8). For example, the first data point now becomes

$$\mathbf{X}^{*} = \begin{pmatrix} 0.691 & 0 \\ 0.707 & 0 \end{pmatrix}\begin{pmatrix} 20.368 \\ 0 \end{pmatrix} = \begin{pmatrix} 14.074 \\ 14.400 \end{pmatrix}.$$

The inverse transformation (back to pixel data) has been done in columns U and V of the spreadsheet, and now we can see what it looks like (and we're done with this Problem):

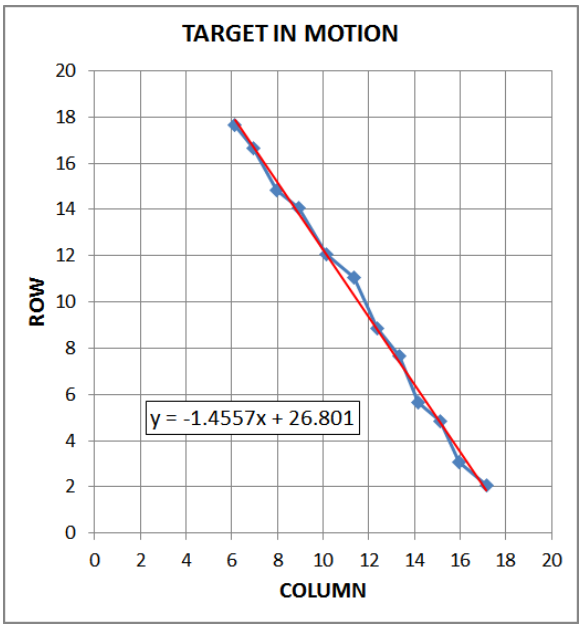


14-7. ***Tracking a Target*** On the DATA worksheet for this problem, you will find thirteen partial frames of data taken by a large area surveillance sensor of a moving target. The frames have been corrected for bad pixels, bias and gradients, and noise, and the background has been suppressed. The frames were taken one second apart, and each pixel has a GSD of approximately 800 × 800 m. Assume the orientation of the frames is aligned with the Earth's geographic coordinates, North up. Estimate the speed and heading of the target. Are these really the target's true velocity and heading? What other information might you need to know?

SUGGESTED SOLUTION: The companion Suggested Solutions (ODD) spreadsheet gives the details of the centroid calculations for Frames 2 – 13. (We can tell from what we expect the approximate shape of the target's PSF to be that it is only partially showing on the edge of the focal plane in Frame 1 – which is Murphy's Law.) The counts (i.e., energies collected) in each row/column are summed, then multiplied (weighted) by their respective row/column number. The sum of the sums of the counts is then divided into the sum of the weighted products. The results of the computation are summarized in this table, including the sum of target counts found in each frame (proportional to the energy collected within the target's PSF). The last column gives the distance (in pixels) between successive centroids (to be discussed later) ~

	--- CENTROID ---			
FRAME	ROW	COL	COUNTS	Δs
1	UNK	UNK	UNK	N/A
2	17.65	6.15	48380	-----
3	16.65	6.95	50868	1.28
4	14.85	7.95	54017	2.06
5	14.05	8.95	50955	2.79
6	12.05	10.15	49618	2.33
7	11.05	11.35	55277	1.56
8	8.85	12.35	55275	2.42
9	7.65	13.35	64068	1.56
10	5.65	14.15	57157	2.15
11	4.85	15.15	57790	1.28
12	3.05	15.95	65327	1.97
13	2.05	17.15	65954	1.56

When we plot our results, we can see the path the target's centroid has taken across our FPA. It appears somewhat jerky probably because of some jitter in the sensor's pointing control. Assuming the apparent path should be a straight line, and to smooth it out, we can estimate the motion using MSExcel's Add Trendline feature, giving us a best-fit linear equation. (The equation's notation is "x" = column and "y" = row. Remember to check the equation to ascertain it actually approximates your data! You might need more decimal places.)



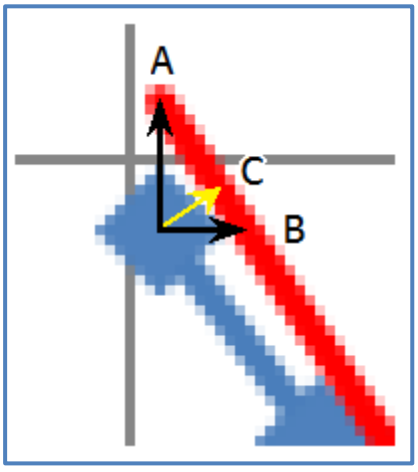
The apparent speed of a target moving across a focal plane was discussed at some length in Problems 13-7 through 13-10 where we did not see the computed centroids as given here. Here we assume that speed across the focal plane (in pixels per second) is directly proportional to apparent ground speed, since we have that one pixel represents a GSD of 800 meters.

A preliminary calculation of $\Delta s = \sqrt{(\Delta\text{row})^2 + (\Delta\text{column})^2}$, shown in the last column of the table, suggests that this target is neither accelerating nor decelerating. At least there is enough irregular change in the values from frame to frame that no clear trend is noted. We therefore will make the assumption the apparent speed (along a straight path) is supposed to be constant, and may be calculated, in engineering units, as an average speed:

$$\bar{v} \approx \frac{\Delta\text{pixels}}{\Delta\text{time}}\left(\frac{800\text{ m}}{\text{pixel}}\right) \quad \left[\frac{\text{m}}{\text{s}}\right].$$

We learned from the previous problems (Chapter 13) the minimum error in this estimate will be -1.4557when we take the end-points to be as far apart as possible, meaning values from Frame 2 and Frame 13. Taking the centroids for those frames, and noting the time difference is 11 seconds, we get ~

$$\bar{v} \approx \frac{\sqrt{(2.05-17.65)^2 + (17.15-6.15)^2}}{11\,\text{s}}\left(\frac{800\text{ m}}{\text{pixel}}\right) \approx 1388\,\text{m}\cdot\text{s}^{-1}.$$



This being an estimate, one can always ask "Is there a better answer?" The answer is "maybe" by considering this: We have reasonable assurance that the trendline is close to being an accurate depiction of the actual (apparent) track, but the end-points we just used do not lie exactly on the trendline. Can we make use of the trendline somehow?

Looking at the figure at left, which is a magnification of the trendline passing near the Frame 2 centroid, we see that we could use the trendline equation in two different ways. If

we believe the column part of Frame 2's centroid is correct, then the trendline gives us an adjusted row value at A:

$$row_2 = -1.4557(6.15) + 26.801 \approx 17.85 .$$

But on the other hand, if we believe the row part of Frame 2's centroid is correct, then the trendline gives us an adjusted column value at B:

$$column_2 = \frac{17.65 - 26.801}{-1.4557} \approx 6.29 .$$

However, a better solution is to apply the logic of the trendline, namely that it is a best-fit to the data constructed in a statistical least-squares fashion. That is, the trendline passes next to each data point so as to minimize the sum of the variances of its closest approach to each point. This is depicted as C in the figure, which is seen to be closer than A or B. To find C, we calculate the equation of a line passing through the data point perpendicular to the trendline, then solve the two equations (new equation and trendline equation) simultaneously. The details of this are left as an exercise for the student, and the results for Frame 2 and Frame 13 adjusted centroids are

	--- ADJUSTED ---	
Frame	ROW	COL
2	17.71	6.24
13	1.98	17.05

Now we can make a better (our best?) estimate of the target's apparent speed ~

$$\bar{v} \approx \frac{\sqrt{(1.98 - 17.71)^2 + (17.05 - 6.24)^2}}{11\,\text{s}} \left(\frac{800\,\text{m}}{\text{pixel}} \right) \approx 1388\,\text{m}\cdot\text{s}^{-1} .$$

WELL! Our estimate did not improve for this data set, but the method is recommended for "best" results.

Finally, we need to compute the target's apparent heading (direction of travel). We could make sure the plot above is drawn to scale and measure it with a protractor, or we could suppose the trendline equation is correct. In the latter case, the leading coefficient, the slope, is the tangent of the angle the line makes with the *x*-axis. Inspection of the data makes sure we get the direction right (the target appears to be moving southeast, if we assume the rows and columns are aligned with Earth's longitude and latitude, respectively). Thus:

$$\theta = \tan^{-1}(-1.4557) \approx -55.5^\circ .$$

Heading, however, is usually referenced to North, we need to add 90° to get ~

TH =90° + 55.5° ≈ 145.5° (clockwise from North).

COMMENT: All of the above should be approximately correct *if* the target is moving on the ground, or at least at a constant altitude. (Note the target's speed is about Mach 4.0 at sea level.) The increase in target energy noted in our centroid calculations may indicate that it is climbing through the atmosphere ~ the atmospheric attenuation is decreasing. This is indeterminate from these data, so we would need some corroborating evidence to confirm this.