APPENDIX A. TECHNICAL DETAIL OF SPATIAL SUBSETTING ALGORITHM

INTRODUCTION – POINT SPREAD FUNCTIONS, PIXELS, SPATIAL SAMPLING, AND OFFSETS BETWEEN DIFFERENT SPATIAL DATA

In this Appendix, we want to be more precise about the relationship between the 1-km data and the ¼-km data than we were in the body of the text. We begin by discussing Point Spread Functions (PSFs), which represent the angular weighting that relates a radiance pattern to the signal sensed by a particular channel of a radiometer. For fairly precise work, we summarize the PSF of a given instrument by a centroid position and an ellipse that represents the angular boundary within which ninety-five percent (or some other reasonable fraction) of a particular pixel's signal originates. Next, we discuss the spatial pattern that these ellipses make when they are projected onto a surface of reference, such as a geoid that describes the Earth's surface (or the top of the atmosphere at some reference altitude). What we want from this disucssion is a standardization of nomenclature, in which we approximate the mathematical complexities of the real sampling by creating a roughly rectangular grid of cells. The center of each cell is located at the centroid of the ellipse for that pixel. This conceptual framework allows us to think of pixels as being organized into scan lines and pixels within each line.

With this conceptual framework, we can formulate the precise relationship between pixels with different spatial resolutions. We can also describe sampling algorithms that extract a fraction of the pixels in the original images and place them in the arrays that contain the data in the subset. As we have seen in the body of this text, we need an algorithm that can identify which pixels from the original data belong in the lower resolution subset. We also need algorithms that will let us relate pixels in the subset at ¼-km resolution to pixels in the 1-km subset – or to the original imager data set. These indexing algorithms are important because we are trying to arrange the higher resolution data so that 4 X 4 arrays of ¼-km pixels are spatially matched with single pixels of 1-km data (or aggregates of ½-km data).

POINT SPREAD FUNCTIONS

With a fair degree of generality, the measurement of the radiance leaving the top of the atmosphere involves having an instrument integrate the radiance's spatial pattern over the field-of-view of the instrument. For careful work, we would probably start by describing the distribution of radiance arriving at the instrument aperture as an angular function with respect to the optical axis. The light typically bounces through the optics of the instrument, arriving at the detector for a particular channel, where the wavelengths of interest are (mostly) absorbed. For thermal instruments, such as CERES, an increment of light energy slightly increases the temperature of the detector flake that absorbs it. For quantum instruments, such as MODIS, the increment of light energy alters the flow of electrons within the semi-conductor material in the detector array. Regardless of the detection mechanism, the instrument electronics integrates the increment of signal over a small, but finite length of time, during which the instrument moves along the orbital track – and perhaps changes slightly in angle.

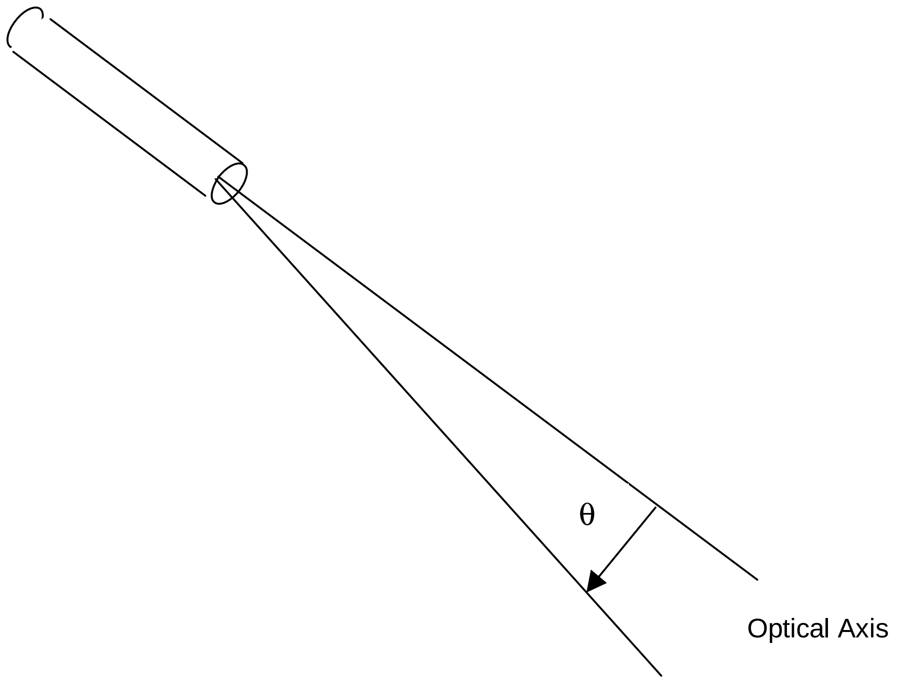

Figure 1. Definition of Polar Angle, q, used to describe the direction from which light approaches the telescope with respect to the optical axis. This figure does not show the reference direction that quantifies the azimuthal angle definition in this spherical polar coordinate system.

Without descending too far into the mathematical details, we can describe the relationship between the digital count in the measurement stream from a particular pixel and the angular distribution of light with respect to the optical axis as

$$D = \int d\Omega \; PSF(\Omega) * I(\Omega)$$

In this expression, *D* is the digital number (or CERES count) in the data stream. $d\Omega$ is a solid angle increment (if q is the polar angle with respect to the optical axis and f is the azimuthal angle, then $d\Omega = dq \; \sin q \; df$). Figure 1 shows the relationship between the telescope optical axis and a light ray arriving there. *PSF*(Ω) is the Point Spread Function, and *I*(Ω) is the incident radiance, which varies with direction. The *PSF* is typically a bivariate Gaussian, i.e., a hill-like shape where the size of the FOV is representable as an ellipse centered on the peak of the hill. Figure 2 provides a schematic of this function. It is hard to make square or hexagonal *PSF*'s. To do so requires sharply focusing optics, very rapidly responding detectors, and no detector FOV motion. Derivation or measurement of PSF's is beyond the scope of this Appendix.

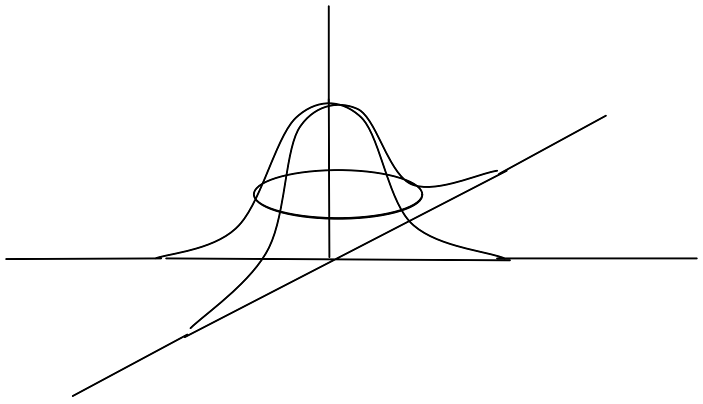

Figure 2. Point Spread Function for a Single Pixel in the optical axis coordinate system, where distance from the original is proportional to q**.**

SPATIAL SAMPLING PATTERNS

By design, most imaging instruments align their pixels in rectangular arrays. The design includes alignment of the array axes with respect to the orbital motion of the satellite over the Earth. Typically, one axis of the array is perpendicular to the orbital motion. We call the pixels that line up with each other in this direction a *scan line*. Scan lines then follow one another in a sequence along the orbit. Figure 3 shows a three-dimensional representation of the geometry of a single scan line of imager data with respect to the orbital motion. The pixels are represented as ellipses defined by the PSF, in accordance with the discussion in the previous section of this Appendix.

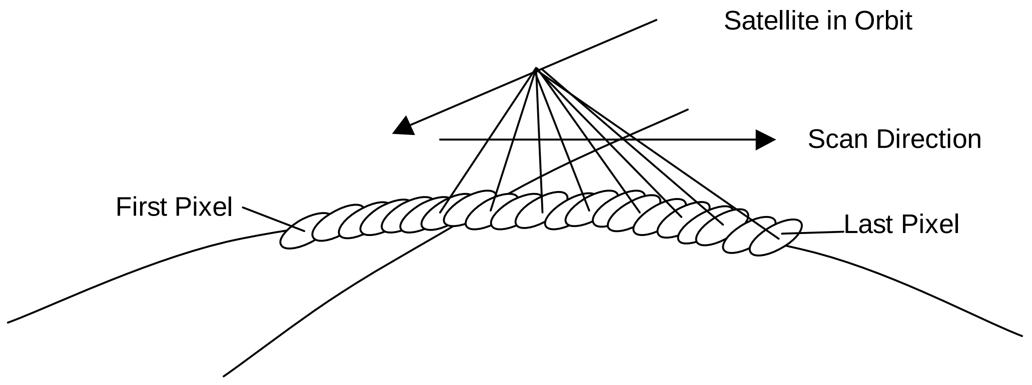

Figure 3. Description of Single Scan Line Geometry, showing the Satellite motion in orbit and the pixels lined up in the scan direction. This schematic view also indicates the first pixel in the scan line and the last pixel in the line.

To some extent, the geometric description is schematic. The real imaging instruments may have scans that go from right-to-left (when facing in the direction of satellite motion) or left-to-right. The discrete mathematics we provide shortly is straightforward to adopt to the geometric description. In addition, the instrument may use linear detector arrays to provide data, where a single scan of the instrument provides multiple scan lines (in the same spectral band). From the perspective of this description, what matters is that we number scan lines uniquely. Thus, we want the description to have sequentially numbered scan lines.

The geometric description in Figure 3 is convenient for representing images, in which we want to identify the upper-left corner of the image with pixel 1 of scan line 1. We assume that the scan lines in a single spectral band have a constant number of pixels, *NP*. We also assume that the number of scan lines in an image is not larger than *Max_Scan_Lines*. Figure 4 is an image extracted from the MODIS image gallery, showing the Mississippi Delta region during daylight. While the image is rectangular, the sides of the rectangle are not aligned with respect to parallels of latitude and meridians of longitude. In that frame of reference, the image would be slightly skewed, with the satellite orbital path passing from north-northeast to south-southwest. In terms of pixel and scan line indexing scheme, pixel 1 in scan line 1 is located at the upper left corner of the image. Pixel *NP* in scan line *Max_Scan_Lines* is located at the lower right corner of the image.

In terms of data structures, we now assume that an image like the one in figure 4 is representable in terms of an array of a single type of numbers, typically 16-bit signed integers. If we use C array indexing conventions, so that the right-most index of an array advances more rapidly, we can define an image as an array, such as

```
Max_Scan_Lines_1km : constant := 1015; -- 1-km MODIS
NP_1km             : constant := 2030; -- 1-km MODIS
type Single_Band_1km_Image is array
  (1 .. Max_Scan_Lines_1km, 1 .. NP_1km) of signed_16_bit_integer;
```

This convention stores the pixels in a particular scan line consecutively.

RELATIONSHIP OF PIXELS IN SUBSETS TO FULL-RESOLUTION PIXELS

For CERES purposes, there are two important resolutions in MODIS data: 1-km and ¼-km. While the original MODIS data have been obtained using three resolutions, 250 m, 500 m, and 1 km, the CERES team expects to use the MODIS team's aggregation of 250 m data (for channels 1 and 2) and 500 m data (for channels 3, 4, 5, 6, and 7) to 1 km. The major difficulty lies in the relationship between these 1 km data and the 250 m data in channel 1. We note that there may be offsets in the position of pixels between these two resolutions.

To define a subset of 1 km data that reduces data volume, we want to choose every other pixel in every other scan line. The algorithm is straightforward, as we show in Listing 1.

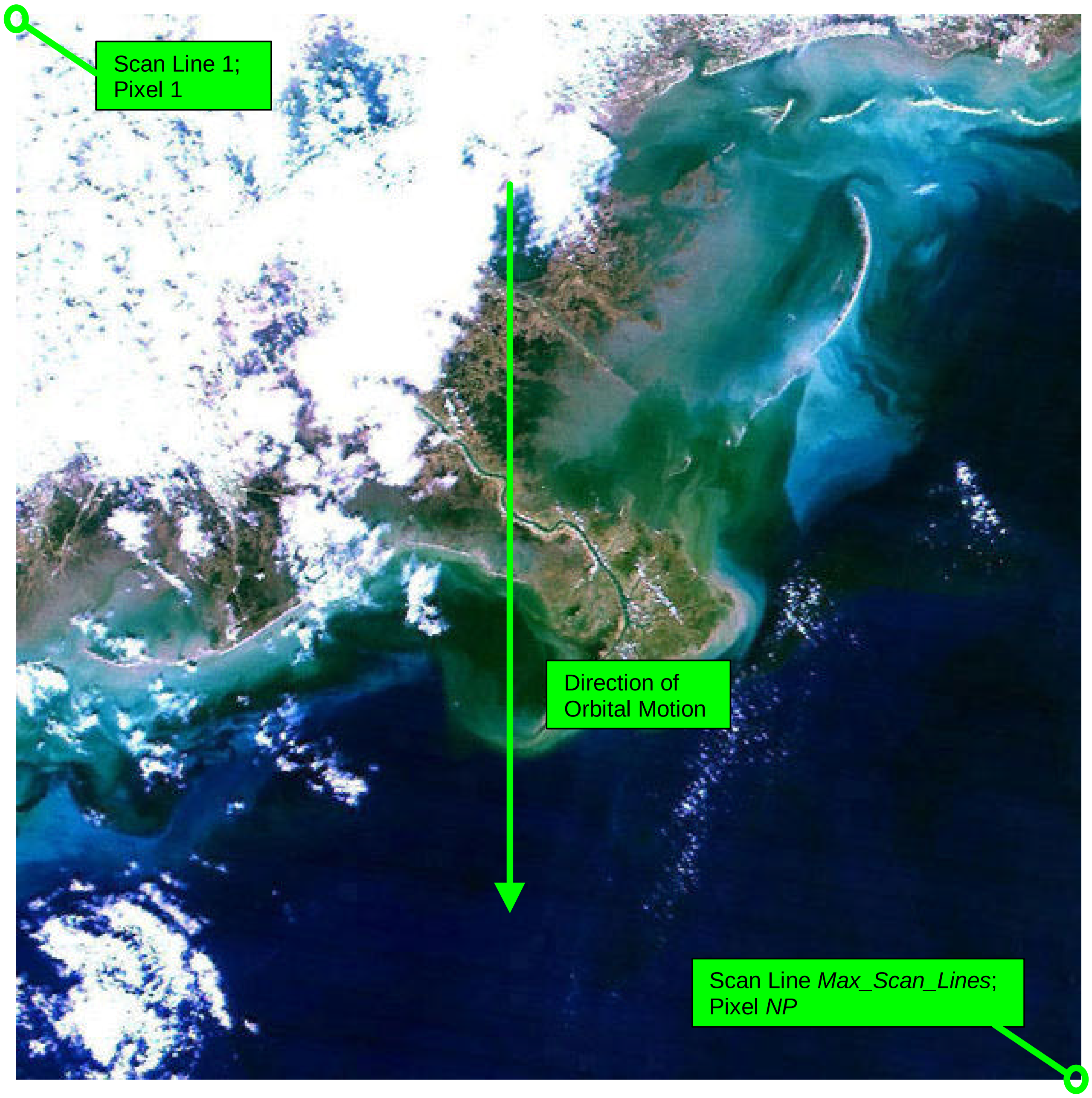

Figure 4. False-Color MODIS Image of Mississippi Delta (MODIS1000001_md.jpg) obtained from the MODIS image gallery (http://modis.gsfc.nasa.gov…). Pixel 1 in scan line 1 is at the upper left of the image; pixel *NP* in scan line *Max_Scan_Lines* **is at the lower right.**

```
------------------------------------------------------------------
-- Type Definitions
------------------------------------------------------------------
Max_Scan_Lines_1km   : constant := 1015; -- 1-km MODIS
NP_1km               : constant := 2030; -- 1-km MODIS
type Single_Band_1km_Image is array
  (1 .. Max_Scan_Lines_1km, 1 .. NP_1km) of signed_16_bit_integer;
MSS_SL_1km           : constant := Max_Scan_Lines_1km/2;
MSS_NP               : constant := NP_1km/2;
type Single_Band_SS_1km_Image is array
  (1 .. MSS_SL_1km, 1 .. MSS_NP) of of signed_16_bit_integer;

------------------------------------------------------------------
-- Variables
------------------------------------------------------------------
MODIS_1km_Image      : Single_Band_1km_Image;
Subsampled_1km_Image : Single_Band_SS_1km_Image;

…

------------------------------------------------------------------
-- Code for Subsampling
------------------------------------------------------------------

for Subsampled_Scan_Line in 1 .. MSS_SL_1km loop
  MODIS_1km_Scan_Line := 2*(Subsampled_Scan_Line – 1) + 1;
  For Subsampled_Pixel in 1 .. MSS_NP loop
    Subsampled_1km_Image(Subsampled_Scan_Line, Subsampled_Pixel)
      := MODIS_1km_Image(MODIS_1km_Scan_Line,
                         2*(Subsampled_Pixel – 1) + 1);
  end loop;
end loop;
```

Listing 1. Ada-like Description of the Data Structures and Sampling Algorithm that would sample a 1-km MODIS array to produce the subset for the CERES CID input.

From a programming standpoint, Listing 1 carefully identifies the arrays as being 16-bit signed integers that start each index at a value of 1. By using a sampling in the subset array indices, it is possible to avoid complex indexing logic. The correspondence between MODIS 1-km image scan lines and the scan lines in the subsample, or between the pixels in these two images is indicated in the code. Table 1 provides some numerical examples of this correspondence, as well as the general relationships between the two indexing sequences. Note that if actual code that builds the subsets uses arrays starting with zero instead of one, then these relationships need to be adjusted accordingly.

Table 1. Index Correspondances Between 1-km MODIS Images and CERES 1-km Subsetted Images.

Scan Line Correspondence	
MODIS 1-km Scan Line Index	CERES 1-km Subsetted Line Index
1	1
3	2
5	3
7	4
9	5
…	…
`2*(SSL – 1) + 1`	`SSL`
`SL`	`((SL – 1)/2) + 1`
Pixel Correspondence	
MODIS 1-km Pixel Index	CERES 1-km Subsetted Pixel Index
1	1
3	2
5	3
7	4
9	5
…	…
`2*(SP – 1) + 1`	`SP`
`P`	`((P – 1)/2) + 1`

RELATIONSHIP OF PIXELS IN ¼-KM SUBSETS TO 1-KM PIXELS

The algorithm we just presented does not provide the subset of ¼-km pixels that we need in order to understand the spatial variability within the 1-km footprints. What we want are subsets of a ¼-km resolution image in which each subset covers (most) of a 1-km pixel. Unfortunately, the ¼-km image may not be exactly collocated with the 1-km image. Potentially, there is an offset in the scan-line direction and another in the along-track direction. (In a pessimistic mood, one might worry about the possibility that the two arrays are slightly skewed with respect to each other. We will not consider this possibility further in this discussion – out of the expectation that the instrument designers were well aware of this concern and have carefully designed it out of the devices in orbit.)

The algorithm to create a subset of four-by-four ¼-km pixel blocks from the original ¼-km MODIS images is more complex than the algorithm we just presented. Most of this complexity arises from the desire to use these blocks rather than a variant of the previous algorithm. Geometric offsets add to this algorithmic complexity. In addition, we need to specify algorithms that relate a ¼-km pixel to the 1-km pixels in the subset, to the 1-km MODIS data, and to the ¼-km original image – as well as the inverse relationships. In the following paragraphs, we develop the appropriate algorithm and provide some calculational results that allow us to check the relationships when they have been programmed to create the subset. Our approach will be based on using simple, discrete functions to relate different pixel index relationships to each other. Once we have such a relationship for one dimension, we can apply the same logic to the second dimension.

Understanding the Geometric Relationships Between ¼-km Subsets and 1-km Subsets

What we want from the ¼-km data is the ability to check on the spatial variability within a 1-km pixel. This means that we want to center the ¼-km block on the 1-km pixel. It is easy to see that if we have a linear offset, ΔO, between the ¼-km pixels and the 1-km pixels, then we have the same offset between the desired ¼-km subset arrays and the 1-km pixels. As a result, we can simplify the subsetting problem to one that deals with offsets in array indices. We do not propose to deal with the complexities of resampling (where we would have to know the MODIS PSF and would have a much more complex computation to provide a rigorous resampling-deconvolution algorithm).

We need to provide a more careful definition of geolocation offsets. Let us consider first an offset in the along-track direction. If the 1-km pixels are offset in the positive along-track direction (so the 1-km pixel's center is moved a distance ΔO in the increasing along-track direction), we can define an index offset, O, as

$$O = \mathrm{INT}(\Delta O \,/\, 250\mathrm{m})$$

In words, if the 1-km pixels are slid 250m down the orbit track, the offset is 1; if they slide 500 m down the track, the offset is 2, and so on. We expect the pixels we choose from the ¼-km data to respond to the pixel index offset, in the sense that if O is 2, we need to select the subset pixels from two ¼-km scan lines further down the image.

Likewise, if the linear offset is in the cross-track direction, then we take a positive index offset as moving the ¼-km pixels we want to sample in the direction of the increasing index. In other words, for a positive offset in the 1-km data the ¼-km samples we want to extract are to the right of where they would be in Figure 4 if there were no offset in that direction.

To derive the relationship between the original ¼-km pixels and the subsetted ¼-km pixels, we can consider two one-dimensional arrays of pixel indexes. The first array includes the indexes for the original data, and therefore starts at 1 and increments sequentially through the remaining indices. The second array has the same size, but puts nothing in the unselected elements and an appropriate index in the selected ones. In the latter case, the selected cell with the smallest array index has a 1 in its element. The second selected cell gets a 2, the third a 3, and so on. In the deep interior of the array, the elements will have sequences of four successive elements that are part of the subset, followed by four empty elements, then four more successive elements, and so on.

Table 2 illustrates this notion for no offset, for an offset of +2 pixels, and for an offset of –2 pixels. The left pair of columns shows a relatively simple subsetting pattern – groups of four pixels, with the indexes of the first selected element in a group increasing by four. The middle pair of columns shows what happens to the selection when the 1-km pixels are shifted down the satellite track by two pixels. The pattern is similar, but offset. In the rightmost pair of columns, the first 1-km pixel has moved partially above the ¼-km data. Thus, we only have two samples from the ¼-km data that provide information about the first 1-km pixel. Thereafter, the "block of 4" pattern resumes.

Table 2. Illustration of Pixel Indices for Original ¼-km Image and for Subsets with Three Different Offsets.

Original	Subset w No Offset	Original	Subset w +2 Offset	Original	Subset w -2 Offset
1	1	1		1	1
2	2	2		2	2
3	3	3	1	3	
4	4	4	2	4	
5		5	3	5	
6		6	4	6	
7		7		7	3
8		8		8	4
9	5	9		9	5
10	6	10		10	6
11	7	11	5	11	
12	8	12	6	12	
13		13	7	13	
14		14	8	14	
15		15		15	7
16		16		16	8
17	9	17		17	9
18	10	18		18	10
19	11	19	9	19	
20	12	20	10	20	

What we need is an algorithmic description that will give us these patterns. We have two alternatives: use the original index and compute the subset index (with a 0 output from the computation identifying which pixels should not enter the subset), or use the subset index as the base and compute the original index that enters the subset. The latter is computationally preferable because there are only half as many elements in the array and because we do not have to insert a conditional statement in the loop to check for elements that are not subset – provided we can come up with a sufficiently simple algorithm. Perhaps perversely, we start with by showing how the algorithm should work if the MODIS ¼-km pixel indexes are used to count position in the array. Then, we use the CERES subset indexes for this purpose.

Subsetting Functions for Indexing in the Original ¼-km MODIS Image

If we ignore the comments on computational efficiency, we can build an algorithm that will loop through the ¼-km data for all of the lines (or all of the pixels) – giving the proper index for the subsetted image where the data should be pulled into the subset and zero otherwise. To build the algorithm in terms of the original MODIS index, we note that the effect of the offset is to allow us to create an index array whose value is simply offset from the original index.

Let us start by defining the function

```
$\Delta$ : if $O >= 0$ then
        $\Delta = 1$;
    else
        $\Delta = 1 - \mathrm{MOD}(\mathrm{ABS}(O), 8)$;
    end if;
```

Then, define

$$X = MI - O - 1$$

To extract appropriate values from X, we form a comb-like filter by first computing

$$R = 7 - \mathrm{MOD}(X, 8)$$

and then

$$U = \mathrm{TRUNC}(R/4)$$

Next, let

$$T = 4*\mathrm{TRUNC}(X/8)$$

By writing

$$CI = U*(T + Q + \Delta)$$

we have defined a function that computes the ¼-km subset indexes, *CI*, from the original ¼-km indexes, *MI*.

Table 3a provides the sequences for the various variables we have defined when the offset is 0. Table 3b shows these sequences when the offset is +2; 3c when it is –2. Based on experience with a spreadsheet form of this computation, we must keep the offset between –4 and +4 in order for it to work correctly. This should be a satisfactory range given the design of the instrument.

Once we have *CI*, we can compute the equivalent indices for 1-km CERES subset samples by using the following algorithm:

```
Δ′ : if $O >= 0$ then
          $\Delta' = 1$;
     else
          $\Delta' = 1 - \mathrm{MOD}(\mathrm{ABS}(O), 4)$;
     end if;
```

followed by

$$C\_1\mathrm{km} = U*\mathrm{TRUNC}[(CI - \Delta')/4] + 1$$

and

$$M\_1\mathrm{km} = U*\{2*\mathrm{TRUNC}[(CI - \Delta')/4] + 1\}$$

Note that we can use this algorithm in a sequence such as that shown in Listing 2.

Table 3a. Illustration of Intermediate Variables When *O* = 0.

In this case, $\Delta = 1$ and $\Delta' = 1$

MI	X	R	U	T	CI	C_1km	M_1km
1	0	7	1	0	1	1	1
2	1	6	1	0	2	1	1
3	2	5	1	0	3	1	1
4	3	4	1	0	4	1	1
5	4	3	0	0	0	0	0
6	5	2	0	0	0	0	0
7	6	1	0	0	0	0	0
8	7	0	0	0	0	0	0
9	8	7	1	4	5	2	3
10	9	6	1	4	6	2	3
11	10	5	1	4	7	2	3
12	11	4	1	4	8	2	3
13	12	3	0	4	0	0	0
14	13	2	0	4	0	0	0
15	14	1	0	4	0	0	0
16	15	0	0	4	0	0	0
17	16	7	1	8	9	3	5
18	17	6	1	8	10	3	5
19	18	5	1	8	11	3	5
20	19	4	1	8	12	3	5
21	20	3	0	8	0	0	0
22	21	2	0	8	0	0	0
23	22	1	0	8	0	0	0
24	23	0	0	8	0	0	0
25	24	7	1	12	13	4	7
26	25	6	1	12	14	4	7
27	26	5	1	12	15	4	7
28	27	4	1	12	16	4	7
29	28	3	0	12	0	0	0
30	29	2	0	12	0	0	0
31	30	1	0	12	0	0	0
32	31	0	0	12	0	0	0
33	32	7	1	16	17	5	9
34	33	6	1	16	18	5	9
35	34	5	1	16	19	5	9

Table 3b. Illustration of Intermediate Variables When *O* = 2.

In this case, $\Delta = 1$ and $\Delta' = 1$

MI	X	R	U	T	CI	C_1km	M_1km
1	-2	1	0	0	0	0	0
2	-1	0	0	0	0	0	0
3	0	7	1	0	1	1	1
4	1	6	1	0	2	1	1
5	2	5	1	0	3	1	1
6	3	4	1	0	4	1	1
7	4	3	0	0	0	0	0
8	5	2	0	0	0	0	0
9	6	1	0	0	0	0	0
10	7	0	0	0	0	0	0
11	8	7	1	4	5	2	3
12	9	6	1	4	6	2	3
13	10	5	1	4	7	2	3
14	11	4	1	4	8	2	3
15	12	3	0	4	0	0	0
16	13	2	0	4	0	0	0
17	14	1	0	4	0	0	0
18	15	0	0	4	0	0	0
19	16	7	1	8	9	3	5
20	17	6	1	8	10	3	5
21	18	5	1	8	11	3	5
22	19	4	1	8	12	3	5
23	20	3	0	8	0	0	0
24	21	2	0	8	0	0	0
25	22	1	0	8	0	0	0
26	23	0	0	8	0	0	0
27	24	7	1	12	13	4	7
28	25	6	1	12	14	4	7
29	26	5	1	12	15	4	7
30	27	4	1	12	16	4	7
31	28	3	0	12	0	0	0
32	29	2	0	12	0	0	0
33	30	1	0	12	0	0	0
34	31	0	0	12	0	0	0
35	32	7	1	16	17	5	9

Table 3c. Illustration of Intermediate Variables When *O* = -2.

In this case, Δ = -1 and Δ' = -1

MI	X	R	U	T	CI	C_1km	M_1km
1	2	5	1	0	1	1	1
2	3	4	1	0	2	1	1
3	4	3	0	0	0	0	0
4	5	2	0	0	0	0	0
5	6	1	0	0	0	0	0
6	7	0	0	0	0	0	0
7	8	7	1	4	3	2	3
8	9	6	1	4	4	2	3
9	10	5	1	4	5	2	3
10	11	4	1	4	6	2	3
11	12	3	0	4	0	0	0
12	13	2	0	4	0	0	0
13	14	1	0	4	0	0	0
14	15	0	0	4	0	0	0
15	16	7	1	8	7	3	5
16	17	6	1	8	8	3	5
17	18	5	1	8	9	3	5
18	19	4	1	8	10	3	5
19	20	3	0	8	0	0	0
20	21	2	0	8	0	0	0
21	22	1	0	8	0	0	0
22	23	0	0	8	0	0	0
23	24	7	1	12	11	4	7
24	25	6	1	12	12	4	7
25	26	5	1	12	13	4	7
26	27	4	1	12	14	4	7
27	28	3	0	12	0	0	0
28	29	2	0	12	0	0	0
29	30	1	0	12	0	0	0
30	31	0	0	12	0	0	0
31	32	7	1	16	15	5	9
32	33	6	1	16	16	5	9
33	34	5	1	16	17	5	9
34	35	4	1	16	18	5	9
35	36	3	0	16	0	0	0

```
function CERES_250m_Subset_Scan_Line(
                MODIS_250m_Index : in     integer;
                D_AT             : in     integer)
         return integer
is
  Delta : integer;
  X     : integer;
  R     : integer;
  U     : integer;
  T     : integer;
  CI    : integer;
begin
  if D_AT >= 0 then
      Delta := 1;
  else
      Delta :== 1 – MOD(ABS(D_AT), 8);
  end if;
  X := MODIS_250m_Index – D_AT – 1;
  R := 7 – MOD(X, 8);
  U := TRUNC(R/4);
  T := 4*TRUNC(X/8);
  CI := U*(T + Q + Delta);
  return CI;
end CERES_250m_Subset_Scan_Line;

function CERES_250m_Subset_Pixel(
                MODIS_250m_Index : in     integer;
                D_CT             : in     integer)
         return integer
is
  Delta : integer;
  X     : integer;
  R     : integer;
  U     : integer;
  T     : integer;
  CI    : integer;
begin
  if D_AT >= 0 then
      Delta := 1;
  else
      Delta :== 1 – MOD(ABS(D_CT), 8);
  end if;
  X := MODIS_250m_Index – D_CT – 1;
  R := 7 – MOD(X, 8);
  U := TRUNC(R/4);
  T := 4*TRUNC(X/8);
  CI := U*(T + Q + Delta);
  return CI;
end CERES_250m_Subset_Pixel;
```

Listing 2. Ada-like Description of the Sampling Algorithm that extracts a 1/4-km MODIS array to produce the subset for the CERES CID input data. Note that Delta_Along_Track and Delta_Cross_Track should be confined to the range –3 .. +3 to work properly. This should not be a problem in practice, since the offsets are currently believed to be between 1 pixel and 2 pixels in magnitude.

```
-------------------------------------------------------------------
-- Code for Subsampling 1/4–km MODIS image to produce CERES subset
-------------------------------------------------------------------
for MI in 1 .. MSS_SL_250m loop
  Read_In_Scan_Line(MI, MODIS_250m_Image);
  CI := CERES_250m_Subset_Scan_Line(MI, Delta_Along_Track);
  if CI > 0 then
    for MI_Pix in 1 .. MSS_NP_250m loop
      CI_Pix := CERES_250m_Subset_Pixel(MI_Pix, Delta_Cross_Track);
      if CI_Pix > 0 then
        CERES_250m_Subset_Image(CI, CI_Pix)
          := MODIS_250m_Image(MI, MI_Pix);
      end if;
    end loop;
  end if;
end loop;
```

Listing 2 (cont'd). Ada-like Description of the Sampling Algorithm that extracts a 1/4-km MODIS array to produce the subset for the CERES CID input data. Note the logic required to deal with 0 index values returned by the index calculation functions, indicating that certain lines or pixels can be skipped. Because of the difference in behavior of the index calculations for positive and negative offsets, an algorithm that must accept either cannot avoid the logic indicated by the boled 'if' constructions.

In producing subsets, the algorithm does not need special logic if the indexing uses the ¼-km scan lines and pixels in the subset – avoiding the scan lines and pixels that don't have to enter the subset. In this case, we need

```
Δ : if $O >= 0$ then
      $\Delta = 1$;
  else
      $\Delta = 1 - \mathrm{MOD}(\mathrm{ABS}(O), 4)$;
  end if;
```

We then run the indexes over *CI*, computing the quantities

$$u = \mathrm{TRUNC}[(CI - \Delta)/4]$$

Next, let

$$r = \mathrm{MOD}[(CI - \Delta), 4] + 1$$

Then

$$MI = 8*d + r + O$$

To obtain the 1-km indices, we simply use

$$C\_1\mathrm{km} = \mathrm{TRUNC}[(CI - \Delta)/4] + 1$$

and

$$M\_1\mathrm{km} = 2*\mathrm{TRUNC}[(CI - \Delta)/4] + 1$$

Note that we can use this algorithm in a sequence such as that shown in Listing 3.

```
-------------------------------------------------------------------
-- Variables for Subsampling 1/4–km MODIS image to produce CERES subset
-------------------------------------------------------------------
D_AT     : integer;
D_CT     : integer;
Delta    : integer;
Delta_CT : integer;
d        : integer;
r        : integer;
CI       : integer;
CI_Pix   : integer;
MI       : integer;
MI_Pix   : integer;

…

-------------------------------------------------------------------
-- Code for Subsampling 1/4–km MODIS image to produce CERES subset
-------------------------------------------------------------------
if D_AT >= 0 then
  Delta := 1;
else
  Delta :== 1 – MOD(ABS(D_AT), 4);
end if;
if D_CT >= 0 then
  Delta_CT := 1;
else
  Delta_CT :== 1 – MOD(ABS(D_CT), 4);
end if;
for CI in 1 .. MSS_SL_250m/2 loop
  d := TRUNC((CI – Delta)/4);
  r := MOD((CI – Delta), 4) + 1;
  MI := 8*d + r + D_AT;
  Read_In_Scan_Line(MI, MODIS_250m_Image);
  for CI_Pix in 1 .. MSS_NP_250m loop
    d := TRUNC((CI_Pix – Delta_CT)/4);
    r := MOD((CI_Pix – Delta_CT), 4) + 1;
    MI_Pix := 8*d + r + D_CT;
    MODIS_250m_Subset_Image(MI, MI_Pix)
      := CERES_250m_Image(CI, CI_Pix);
  end loop;
end loop;
```

Listing 3. Ada-like Description of the Sampling Algorithm that builds a 1-km blocked subset from a 1/4-km MODIS image array. Note that we do not require branching statements in the core of the algorithm in order to avoid scan lines or pixels that should not appear in the subset. Some care is still required, since Delta_AT and Delta_CT should be confined to the range –3 .. +3 to work properly. This should not be a problem in practice, since the offsets are currently believed to be between 1 pixel and 2 pixels in magnitude.

Tables 4a, 4b, and 4c provide numerical test cases for this alternate indexing scheme. Note the lack of 0 entries in the CI column, indicating that using CI does not index MI rows or pixels when these will not enter the1/4-km subset. These algorithms are written under the assumption that the arrays start with 1. If they are being translated into C, come revisions will be necessary in order to ensure proper relationships between the four possible image indexes.

Table 4a. Illustration of Indexing Using CI as the independent variable with *O* = 0.

In this case, Δ = 1

CI	d	r	MI	C_1km	M_1km
1	0	1	1	1	1
2	0	2	2	1	1
3	0	3	3	1	1
4	0	4	4	1	1
5	1	1	9	2	3
6	1	2	10	2	3
7	1	3	11	2	3
8	1	4	12	2	3
9	2	1	17	3	5
10	2	2	18	3	5
11	2	3	19	3	5
12	2	4	20	3	5
13	3	1	25	4	7
14	3	2	26	4	7
15	3	3	27	4	7
16	3	4	28	4	7
17	4	1	33	5	9
18	4	2	34	5	9
19	4	3	35	5	9
20	4	4	36	5	9
21	5	1	41	6	11
22	5	2	42	6	11
23	5	3	43	6	11
24	5	4	44	6	11
25	6	1	49	7	13
26	6	2	50	7	13
27	6	3	51	7	13
28	6	4	52	7	13
29	7	1	57	8	15
30	7	2	58	8	15
31	7	3	59	8	15
32	7	4	60	8	15
33	8	1	65	9	17
34	8	2	66	9	17
35	8	3	67	9	17

Table 4b. . Illustration of Indexing Using CI as the independent variable with *O* = 2.

In this case, $\Delta = 1$

CI	d	r	MI	C_1km	M_1km
1	0	1	3	1	1
2	0	2	4	1	1
3	0	3	5	1	1
4	0	4	6	1	1
5	1	1	11	2	3
6	1	2	12	2	3
7	1	3	13	2	3
8	1	4	14	2	3
9	2	1	19	3	5
10	2	2	20	3	5
11	2	3	21	3	5
12	2	4	22	3	5
13	3	1	27	4	7
14	3	2	28	4	7
15	3	3	29	4	7
16	3	4	30	4	7
17	4	1	35	5	9
18	4	2	36	5	9
19	4	3	37	5	9
20	4	4	38	5	9
21	5	1	43	6	11
22	5	2	44	6	11
23	5	3	45	6	11
24	5	4	46	6	11
25	6	1	51	7	13
26	6	2	52	7	13
27	6	3	53	7	13
28	6	4	54	7	13
29	7	1	59	8	15
30	7	2	60	8	15
31	7	3	61	8	15
32	7	4	62	8	15
33	8	1	67	9	17
34	8	2	68	9	17
35	8	3	69	9	17

Table 3c. . Illustration of Indexing Using CI as the independent variable with *O* = -2.

In this case, $\Delta = 1$

CI	d	r	MI	C_1km	M_1km
1	0	3	1	1	1
2	0	4	2	1	1
3	1	1	7	2	3
4	1	2	8	2	3
5	1	3	9	2	3
6	1	4	10	2	3
7	2	1	15	3	5
8	2	2	16	3	5
9	2	3	17	3	5
10	2	4	18	3	5
11	3	1	23	4	7
12	3	2	24	4	7
13	3	3	25	4	7
14	3	4	26	4	7
15	4	1	31	5	9
16	4	2	32	5	9
17	4	3	33	5	9
18	4	4	34	5	9
19	5	1	39	6	11
20	5	2	40	6	11
21	5	3	41	6	11
22	5	4	42	6	11
23	6	1	47	7	13
24	6	2	48	7	13
25	6	3	49	7	13
26	6	4	50	7	13
27	7	1	55	8	15
28	7	2	56	8	15
29	7	3	57	8	15
30	7	4	58	8	15
31	8	1	63	9	17
32	8	2	64	9	17
33	8	3	65	9	17
34	8	4	66	9	17
35	9	1	71	10	19