
Subject: Re: What is like CONGRID, but averages on reduction?

Posted by [thompson](#) on Wed, 11 Oct 1995 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

grunes@news.nrl.navy.mil (Mitchell R Grunes) writes:

> Subject: What is like CONGRID, but averages on reduction?

> I need to reduce the size of an array by an arbitrary (non-integral)
> factor, using an averaging algorithm. Is there such a routine
> in IDL and/or PV-WAVE?

> When I display an image that is larger than the window, I use:

> A=REBIN(A,XSZ,YSZ)
> or
> A=CONGRID(A,XSZ,YSZ)

> The problem with REBIN is that XSZ and YSZ must be integral factors
> of the size of A. This means that I often can not take advantage
> of the whole screen. But it is able to average things down (as
> long as I don't set /SAMPLE), so bright and dark points don't get
> lost.

> The problem with CONGRID is that it sometimes misses bright or
> dark spots altogether. For example:

> A=INTARR(9,9)
> A(3,3)=1
> PRINT,CONGRID(A,4,4)

> or

> PRINT,CONGRID(A,4,4,/INTERP)
> misses the bright point, because CONGRID can interpolate or use
> nearest neighbor, but cannot average.

> Thanks in advance.

I think this should do what you want.

Bill Thompson

```
=====
=====
FUNCTION REDUCE,ARRAY,I_REDUCE,J_REDUCE
;
;+
; NAME:
; REDUCE
; PURPOSE:
```

```

; Reduce an array by box averaging to a more useful size.
; CATEGORY:
; CALLING SEQUENCE:
; A_NEW = REDUCE( ARRAY [, I_REDUCE [, J_REDUCE ]])
; INPUTS:
; ARRAY = array to be reduced.
; OPTIONAL INPUT PARAMETERS:
; I_REDUCE = Size of box in the first dimension to average over. If
;   not passed, then the procedure selects what it thinks is
;   a suitable value.
; J_REDUCE = Size of box in the second dimension. If not passed, then
;   it has the same value as I_REDUCE.
; COMMON BLOCKS:
; None.
; SIDE EFFECTS:
; None.
; RESTRICTIONS:
; The input ARRAY must have one or two dimensions. The variables must
; not be of type string.
; PROCEDURE:
; The REBIN function is used to reduce ARRAY.
; MODIFICATION HISTORY:
; William Thompson Applied Research Corporation
; May, 1987 8201 Corporate Drive
; Landover, MD 20785
;-
;
ON_ERROR,2
S = SIZE(ARRAY)
;
; Check the number of dimensions of ARRAY.
;
N_DIM = S(0)
IF N_DIM EQ 1 THEN BEGIN
  NX = S(1)
  NY = NX
END ELSE IF N_DIM EQ 2 THEN BEGIN
  NX = S(1)
  NY = S(2)
END ELSE BEGIN
  PRINT, '*** Array must have one or two dimensions, name= ARRAY, procedure REDUCE.'
  RETURN, ARRAY
ENDELSE
;
; If only two parameters were passed, then reduce ARRAY by the same amount in
; the two dimensions. If ARRAY only has one dimension, then JJ is ignored.
;
IF N_PARAMS(0) EQ 2 THEN BEGIN

```

```

IF I_REDUCE LT 2 THEN BEGIN
  PRINT, '*** Variable must be GE 2, name= I_REDUCE, procedure REDUCE.'
  RETURN, ARRAY
ENDIF
II = FIX(I_REDUCE)
JJ = II
;
; If all three parameters were passed, then reduce ARRAY by different amounts
; in the two dimensions. If ARRAY only has one dimension, then JJ is ignored.
;
END ELSE IF N_PARAMS(0) EQ 3 THEN BEGIN
  IF (I_REDUCE > J_REDUCE) LT 2 THEN BEGIN
    IF J_REDUCE GT I_REDUCE THEN BEGIN
      PRINT, '*** Variable must be GE 2, name= J_REDUCE, procedure REDUCE.'
    END ELSE BEGIN
      PRINT, '*** Variable must be GE 2, name= I_REDUCE, procedure REDUCE.'
    ENDELSE
    RETURN, ARRAY
  ENDIF
  II = FIX(I_REDUCE) > 1
  JJ = FIX(J_REDUCE) > 1
;
; If only ARRAY was passed, then calculate an optimum reduction factor. Don't
; reduce the array if one dimension is less than four.
;
END ELSE BEGIN
  N_MAX = NX > NY
  N_MIN = NX < NY
  IF N_MIN LT 4 THEN BEGIN
    PRINT, '*** Unable to reduce array, name= ARRAY, procedure REDUCE.'
    RETURN, ARRAY
  ENDIF
;
; First try reducing by either the square root of the smallest dimension or
; the largest dimension over sixty, whichever is smaller. In the latter case,
; this will try to make the resulting size of the larger dimension to be on
; the order of sixty pixels.
;
  I = FIX(SQRT(N_MIN) < (N_MAX/60)) > 2
  II = I
  JJ = II
;
; Keep increasing the reduction factor by one until either the square root of
; the larger dimension or half the smaller dimension is reached. With each
; reduction factor, find the number of elements that would be not be included
; in the reduction. Use the reduction factor with smallest number of
; remaining elements. The remainder can never be greater than the total
; number of elements.

```

```

;
;
BEST = N_ELEMENTS(ARRAY)
WHILE ((I LE SQRT(N_MAX)) AND (I LT (N_MIN/2))) DO BEGIN
;
; Calculate the number of elements remaining.
;
;
N = FIX(N_MAX/I)
REMAIN = N_MAX - N*I
IF N_DIM EQ 2 THEN BEGIN
N = FIX(N_MIN/I)
R2 = N_MIN - N*I
REMAIN = REMAIN*(N_MIN - R2) + N_MAX*R2
ENDIF
;
; If the number of elements remaining is smaller than the best value found so
; far, then set II and JJ to the current reduction factor.
;
;
IF REMAIN LT BEST THEN BEGIN
BEST = REMAIN
II = I
JJ = II
ENDIF
;
; If no elements remain, then stop looking. Otherwise increase I by one and
; loop.
;
IF BEST EQ 0 THEN GOTO,FOUND_BEST
I = I + 1
ENDWHILE
ENDELSE
;
; Reduce the array.
;
;
FOUND_BEST:
MX = FIX(NX/II)
MY = FIX(NY/JJ)
IF N_DIM EQ 1 THEN BEGIN
A = ARRAY(0:II*MX-1)
A = REBIN(A,MX)
END ELSE BEGIN
A = ARRAY(0:II*MX-1,0:JJ*MY-1)
A = REBIN(A,MX,MY)
ENDELSE
;
;
RETURN,A
END

```
