Subject: registering images, shift(/nowrap)
Posted by ronn on Mon, 11 Nov 2002 13:51:57 GMT
View Forum Message <> Reply to Message

Hello All,

I was wondering if anyone has a routine out there that allows you to shift an array without wrapping? The shift routine works great but anything that gets pushed off on one edge gets wrapped around to the other. I just want to throw this wrapped part away. A simple 1D example is

in = [1,2,3,4,5]
print, shift(in,2)
IDL> 4,5,1,2,3

what I want is

print, shiftNoWrap(in,2)
IDL> 0,0,1,2,3

Of couse, it really needs to work on a 2D image....

Thanks,
Ronn

--
Ronn Kling
KRS, inc.
email: ronn@rlkling.com
"Application Development with IDL"ï¿½ programming book updated for IDL5.5!
"Calling C from IDL, Using DLM's to extend your IDL code"!
"Power Graphics with IDL, A Beginner's Guide to Object Graphics", NEW BOOK!
http://www.rlkling.com/

Subject: Re: registering images, shift(/nowrap)
Posted by Martin Downing on Thu, 14 Nov 2002 23:02:17 GMT
View Forum Message <> Reply to Message

Ron,
I wrote this a while back, allows edge values to be extrapolated or inserts a value for missing. Definitely handy for image processing.
regards
Martin

; %M_FILE: h:\martin\idl\ra\math\mrd_shift.pro %
; %M_CREATED: 18/09/2001 10:39:26 %

```
; %M_MODIFIED: 18/09/2001 13:16:42 %
; %M_AUTHOR: MRD %
function mrd_shift, a, sa, MISSING=MISSING, EXTRAPOLATE=EXTRAPOLATE
;+
; MRD_SHIFT
; enhanced SHIFT function
;
; a - the array (upto 4 dim)
; sa - shift array [sx,sy,..]
; This enhanced function deals with data wrapping
; i.e. the data shifted in from array edges, in three possible ways:
; (default): data WRAPs (this is SHIFT's behaviour)
; MISSING: use this value in cells shift in from outside the dataset
; EXTRAPOLATE: ie use nearest data value (replicates original edge)
;
; restictions: none known
; Author: MR Downing 17/9/2001
;-
a1 = shift(a,sa)
s = size(a)

if s(0) gt 4 then message, "not implemented above 4 dimentions

if keyword_set(EXTRAPOLATE) then begin
 dim = 1
 if s[0] ge dim and n_elements(sa) ge dim then begin
  sh =  sa[dim-1] mod s(dim)
  for i = 0, (abs(sa[dim-1]) < s(dim) ) -1 do begin
   if sh ge 0 then begin
    a1[i,*,*,*] = a1[sh,*,*,*]
   endif else begin
    a1[s(dim)-1-i,*,*,*] = a1[s(dim)+sh-1,*,*,*]
   endelse
  endfor
 endif
 dim = 2
 if s[0] ge dim and n_elements(sa) ge dim then begin
  sh =  sa[dim-1] mod s(dim)
  for i = 0, (abs(sa[dim-1]) < s(dim) ) -1 do begin
   if sh ge 0 then begin
    a1[*,i,*,*,*] = a1[*,sh,*,*,*]
   endif else begin
    a1[*,s(dim)-1-i,*,*,*] = a1[*,s(dim)+sh-1,*,*,*]
   endelse
  endfor
 endif
 dim = 3
 if s[0] ge dim and n_elements(sa) ge dim then begin
```

```
  sh =  sa[dim-1] mod s(dim)
  for i = 0, (abs(sa[dim-1]) < s(dim) ) -1 do begin
   if sh ge 0 then begin
    a1[*,*,i,*,*,*] = a1[*,*,sh,*,*,*]
   endif else begin
    a1[*,*,s(dim)-1-i,*,*,*] = a1[*,*,s(dim)+sh-1,*,*,*]
   endelse
  endfor
 endif
 dim = 4
 if s[0] ge dim and n_elements(sa) ge dim then begin
  sh =  sa[dim-1] mod s(dim)
  for i = 0, (abs(sa[dim-1]) < s(dim) ) -1 do begin
   if sh ge 0 then begin
    a1[*,*,*,i,*,*,*] = a1[*,*,*,sh,*,*,*]
   endif else begin
    a1[*,*,*,s(dim)-1-i,*,*,*] = a1[*,*,*,s(dim)+sh-1,*,*,*]
   endelse
  endfor
 endif

endif else if defined(MISSING) then begin
 dim = 1
 if s[0] ge dim and n_elements(sa) ge dim then begin
   sh = -s(dim) > sa[dim-1] < s(dim)
   if sh gt 0 then begin
    a1[0:sh-1,*,*,*]=MISSING
   endif else if sh lt 0 then begin
    a1[s(dim)+sh:s(dim)-1,*,*,*]=MISSING
   endif
 endif
 dim = 2
 if s[0] ge dim and n_elements(sa) ge dim then begin
   sh = -s(dim) > sa[dim-1] < s(dim)
   if sh gt 0 then begin
    a1[*,0:sh-1,*,*]=MISSING
   endif else if sh lt 0 then begin
    a1[*,s(dim)+sh:s(dim)-1,*,*]=MISSING
   endif
 endif
 dim = 3
 if s[0] ge dim and n_elements(sa) ge dim then begin
   sh = -s(dim) > sa[dim-1] < s(dim)
   if sh gt 0 then begin
    a1[*,*,0:sh-1,*]=MISSING
   endif else if sh lt 0 then begin
    a1[*,*,s(dim)+sh:s(dim)-1,*]=MISSING
   endif
```

```
 endif
 dim = 4
 if s[0] ge dim and n_elements(sa) ge dim then begin
   sh = -s(dim) > sa[dim-1] < s(dim)
   if sh gt 0 then begin
    a1[*,*,*,0:sh-1]=MISSING
   endif else if sh lt 0 then begin
    a1[*,*,*,s(dim)+sh:s(dim)-1]=MISSING
   endif
 endif
endif

return , a1
end


pro test, vol, s1=s,shift_arr=sh_arr, byte=byte

if undefined(sh_arr) then sh_arr = [20,10,45]

if undefined(vol) then begin
 vol = dist(s,s)
 if keyword_set(byte) then vol = bytscl(vol)
 if keyword_set(double) then vol = double(vol)
 vol = rebin(vol,s,s,s, /sample)
endif
help, vol
t0= systime(1)
vol1 = shift(vol, sh_arr)
t1= systime(1)
print, "shift: done in ", t1-t0, " sec"
t0= systime(1)
vol1 = mrd_shift(vol, sh_arr, /ext)
t1= systime(1)
print, "mrd_shift /ext: done in ", t1-t0, " sec"
t0= systime(1)
vol1 = mrd_shift(vol, sh_arr, missing=-1)
t1= systime(1)
print, "mrd_shift mis: done in ", t1-t0, " sec"
t0= systime(1)
vol1 = mrd_shift(vol, sh_arr)
t1= systime(1)
print, "mrd_shift: done in ", t1-t0, " sec"

end

--
---------------------------------------
```

Martin Downing,
Clinical Research Physicist,
Grampian Orthopaedic RSA Research Centre,
Woodend Hospital, Aberdeen, AB15 6LS.

"ronn kling" <ronn@rlkling.com> wrote in message
news:B9F5202A.78C0%ronn@rlkling.com...
> Hello All,
>
> I was wondering if anyone has a routine out there that allows you to shift
> an array without wrapping? The shift routine works great but anything
that
> gets pushed off on one edge gets wrapped around to the other. I just want
to
> throw this wrapped part away. A simple 1D example is
>
> in = [1,2,3,4,5]
> print, shift(in,2)
> IDL> 4,5,1,2,3
>
> what I want is
>
> print, shiftNoWrap(in,2)
> IDL> 0,0,1,2,3
>
> Of couse, it really needs to work on a 2D image....
>
> Thanks,
> Ronn
>
>
> --
> Ronn Kling
> KRS, inc.
> email: ronn@rlkling.com
> "Application Development with IDL" programming book updated for IDL5.5!
> "Calling C from IDL, Using DLM's to extend your IDL code"!
> "Power Graphics with IDL, A Beginner's Guide to Object Graphics", NEW
BOOK!
> http://www.rlkling.com/
>
>