10 Appendices 1 – IDL scripts for image processing

In this appendix, examples are given of the main types of image processing scripts developed for calculations in the chapters 4 to 6. Each script is followed by an example of the accompanying parameter file.

10.1 Appendix 1.1 - Calculation of cover percentage, diversity, edge and fragmentation metrics

```
pro cover_div_frag040130
; This program should be applied to land-cover data in ERDAS 7.5 (.gis) format
;  or similar formats like CHIPS, assuming single band
; It is meant to complement outputs from Moving Windows Fragstats (a la GAF)
; Input : images, list with image and moving windows data in the following format :
;			(once)
;			no. of images
;			number of land cover classes
;			initial window size, increase in winsize, no. of diff. windows, initial step, increase in step
(once)
;			(then for each image)
;			filename (.gis file)
;			For each image: headerlength (no. of pixels to be skipped), cols, rows, pixelsize
;			(no. of classes of interest)
;			(outfile - created automatically in this version)
;
;  Output are comma separated ASCII (.csv) files with each of the cover classes'
; - percentage of sublandscape area, richness'=no. of classes present in last column
; - percentage of edge pixels
; - a simple per class "edge index"
; - per class Matheron "fragmentation" indices + 'landscape Matheron index' in last column of outfile!


; Modified 4 september 2003 to read input UTM coordinates for image and output coordinates for centre
of esch cell/window
; 18 september bug fixed in block-edge-count

			others=7				; No. of other div. metrics to be calculated, pt. SIDI,
SHDI, richness
			filelist='m:\IDL_test\aak_div25m_fill2.txt' ; where the run parameters are kept
			n=0b ; number of files in list
			noclasses=0  ; read from info-file
			backval=0    ; read from info-file
			landval=1b   ; read from info-file
			inclback=0 ; read from info-file
			coverland=1 ; read from info-file
			cols=0l		; etc.
			rows=0l
			headersize=0l
			grainsize=0.0
			ws_ini=0l
			winstep=0l
			step_ini=0l
			step_incr=0l
			openr,lun3,filelist, /get_lun
			readf,lun3,n


			; Read over-all parameter(s): number of inpuit images
```

```
                    image='' ; strings for filenames

                    for inputfiles=0,n-1 do begin
                    readf,lun3, image                                    ; reading image-specific parameters
                    readf,lun3, headersize, cols, rows, grainsize, UL_E, UL_N
                    readf,lun3, noclasses, backval, landval
                    readf,lun3, inclback ; should background pixels be included in calculations (yes if
inclback <> 0)?
                    readf,lun3, coverland ; is there a class for non-background, non-forest (context/matrix)
land ?
                    readf,lun3, ws_ini, win_incr, winss, step_ini, step_incr ; initial window size, increment in
window size,
                                                                         ; number of different windows, inital
step size
                                                                         ; incenrement of stepsize with larger
window..
                    divs=others
                    wins=winss-1
                    maxwin=ws_ini+(win_incr*wins)
                     winstep=step_ini ; stepsize must be reset before each new image is processed
                    winsize=ws_ini

for rounds=1, winss do begin ; new image - varying window sizes

; create (meaningful?) NAMES for output files:

sizestr=string(winsize)
stepstr=string(winstep)
suf1='_w'+strcompress(sizestr, /remove_all)
suf2='s'+strcompress(stepstr, /remove_all)
split=str_sep(image,'.')
origimagename=split[0]
imagename=origimagename+suf1+suf2
outfile1=imagename+'_cover.csv'
outfile2=imagename+'_edgelength.csv'
outfile3=imagename+'_edgeindex.csv'
outfile4=imagename+'_matheron.csv'
outfile5=imagename+'_diverse.csv'
outfile6=imagename+'_sqp.csv'
outfile7=imagename+'_mathallmap.csv'

openr, lun, image, /get_lun  ; read input image
image_arr=bytarr(cols*rows+headersize)
readu, lun, image_arr
free_lun,lun ; Close input image

print, 'reading ',image
print, 'output to ', outfile1
print, 'output to ', outfile2
print, 'output to ', outfile3
print, 'output to ', outfile4
print, 'output to ', outfile5
print, 'output to ', outfile6
print, 'output to ', outfile7
print, 'Background value: ', backval
if (inclback EQ 0) then print, 'Background pixels ignored in Diversity calculations'
print, 'Land value      : ', landval

pixcount=headersize ; store input image as 2-D matrix
image_mtx=bytarr(cols,rows)
for rc=0,rows-1 do begin
          for cc=0,cols-1 do begin
                    image_mtx(cc,rc)=image_arr(pixcount)
                    pixcount=pixcount+1
          endfor ; cc
endfor ; rc
```

```
winsz=float(winsize)
blocksize=float(winsz*winsz)
block_cols=fix((cols-winsize+winstep)/winstep)
block_rows=fix((rows-winsize+winstep)/winstep)
geo_E=fltarr(block_cols)
geo_N=fltarr(block_rows)


x=0u
y=0u
value=0
outedgect=0l
blockedgect=0l
prob=fltarr(256)
percent=0.0
MI=0.0

covercount=lonarr(256)
edgecount=lonarr(256)
edgelengths=fltarr(block_cols, block_rows, noclasses+3)
countpct=intarr(block_cols, block_rows, noclasses+3) ; array for percentage of cover & richenss (no. of
classes in window)
edgepct=intarr(block_cols, block_rows, noclasses+1)  ; array for simple edge ratio
edgeprop=intarr(block_cols, block_rows, noclasses+1) ; array for edge to covertype area ratio
MIA=fltarr(block_cols, block_rows, noclasses+1)  ; Matheron Index Array
sqp_mtx=fltarr(block_cols, block_rows,2)               ; Squareness of Patches Array
divind=fltarr(block_cols, block_rows,divs)           ; matrix for various diversity metrics

; Define output coordinates
outsize=winsize*grainsize
outstep=winstep*grainsize
UL_E_out=UL_E+((outsize-outstep)/2)
UL_N_out=UL_N-((outsize-outstep)/2)
for east=0,block_cols-1 do Geo_E(east)=UL_E_out+outstep*(east+0.5)
for north=0,block_rows-1 do Geo_N(north)=UL_N_out-outstep*(north+0.5)

for a=0,(block_rows-1) do begin  ; calculation starts, runs through blocks - a counts rows (Y values)
        print, 'img', inputfiles+1,' / ',n,'Iteration ',rounds,', ws: ',winsize, ' step: ', winstep,', now analysing
row ',a+1,' of', block_rows
        print, 'blocksize :', blocksize, ' pixels = ', blocksize*grainsize*grainsize/10000,' ha'
        aa=(block_rows-1)-a ; lowerleft coordinate system - better for Surfer import! ignored for the
moment!!
        for b=0,(block_cols-1) do begin                 ; = overlapping windows - b counts columns (X values)
                for c=0,255 do covercount(c)=0 ; reset covercounter
                for ee=0,255 do edgecount(ee)=0 ; reset edgecounter
                blockedgect=0.0
                outedgect=0.0
                totedgect=0.0
                isobject=0
                notback=0

                ; count of object - "outside window" edges (only) for combined M and SqP values :
                for xo=1,winsize-2 do begin ; counting along outer rows:
                        blockedgect=blockedgect+(image_mtx(b*winstep+xo,a*winstep) NE landval)
*(image_mtx(b*winstep+xo,a*winstep) NE backval); counting for top row in block
                        blockedgect=blockedgect+(image_mtx(b*winstep+xo,a*winstep+winsize-1) NE
landval) *(image_mtx(b*winstep+xo,a*winstep+winsize-1) NE backval); counting for bottom row in block
                endfor; xo
                for yo=1,winsize-2 do begin ; counting along outer columns:
                        blockedgect=blockedgect+(image_mtx(b*winstep,a*winstep+yo) NE landval)
*(image_mtx(b*winstep,a*winstep+yo) NE backval); counting for left column in block
                        blockedgect=blockedgect+(image_mtx(b*winstep+winsize-1,a*winstep+yo) NE
landval) *(image_mtx(b*winstep+winsize-1,a*winstep+yo) NE backval); counting for right column in block
                endfor; yo
                blockedgect=blockedgect+2*(image_mtx(b*winstep,a*winstep) NE landval)
*(image_mtx(b*winstep,a*winstep) NE backval) ; top left corner of block
```

```
		blockedgect=blockedgect+2*(image_mtx(b*winstep,a*winstep+winsize-1) NE landval)
*(image_mtx(b*winstep,a*winstep+winsize-1) NE backval) ; top right corner of block
		blockedgect=blockedgect+2*(image_mtx(b*winstep+winsize-1,a*winstep) NE landval)
*(image_mtx(b*winstep+winsize-1,a*winstep) NE backval) ; bottom left corner of block
		blockedgect=blockedgect+2*(image_mtx(b*winstep+winsize-1,a*winstep+winsize-1) NE
landval) *(image_mtx(b*winstep+winsize-1,a*winstep+winsize-1) NE backval) ; bottom right corner of
block

		for d=0,(winsize-1) do begin	; counting inside window
			for e=0,(winsize-1) do begin
				x=(b*winstep+e)
				leftx=(x-1)
				rightx=(x+1)
				y=(a*winstep+d)
				upy=(y-1)
				downy=(y+1)
				value=image_mtx(x,y)			; reading of pixel value
				covercount(value)=covercount(value)+1	; THIS is where the actual counting
takes place - directly in the array

; "Internal edges" (between all LC types):

; "Landscape edges" (forest - background)
				isobject=((value NE landval)*(value NE backval))
				notback=(value NE backval)
				if coverland NE 0 then begin ; the 'object of structural interest' is anything not
matrix or background (e.g. all forest):
				if (e GT 0) then outedgect=outedgect+(isobject*((image_mtx(leftx,y) EQ
landval)+(image_mtx(leftx,y) EQ backval)))  ; checks for edges in horizontal direction
				if (e LT winsize-1) then outedgect=outedgect+(isobject*((image_mtx(rightx,y)
EQ landval)+(image_mtx(rightx,y) EQ backval)))
				if (d GT 0) then outedgect=outedgect+(isobject*((image_mtx(x,upy) EQ
landval)+(image_mtx(x,upy) EQ backval)))    ; checks for edges in vertical direction
				if (d LT winsize-1) then
outedgect=outedgect+(isobject*((image_mtx(x,downy) EQ landval)+(image_mtx(x,downy) EQ backval)))
				endif else begin ; the 'object of structural interest' is anything not background
(e.g. all land):
				if (e GT 0) then outedgect=outedgect+(notback*(image_mtx(leftx,y) EQ
backval)) ; checks for edges in horizontal direction
				if (e LT winsize-1) then outedgect=outedgect+(notback*(image_mtx(rightx,y)
EQ backval))
				if (d GT 0) then outedgect=outedgect+(notback*(image_mtx(x,upy) EQ
backval)) ; checks for edges in vertical direction
				if (d LT winsize-1) then outedgect=outedgect+(notback*(image_mtx(x,downy)
EQ backval))
				endelse
			endfor ;e
		endfor ;d

; INDEX CALCULTATION:
		richn=0s
		richslot=0s
		shannon_f=0.0
		shannon_l=0.0
		simpson_f=0.0
		simpson_l=0.0
		sqp_obj=0.0
		sqp_land=0.0
		landscpix=float(blocksize-covercount(backval)) ; Greater than 0 if there in this window are
pixels different from background
		forestpix=float(landscpix-covercount(landval)) ; Greater than 0 if there in this window are
pixels different from non-forest land
		if (landscpix GT 0) then forestfraction=(forestpix/landscpix) else forestfraction=0
		sum_pf=0.0
		sum_pl=0.0
		totel=0.0
```

```
          for f=0,noclasses-1 do begin          ; only go through the classes that are defined and
meant for output

               proportion=0.0 ; to be used for this class in this window
               prop_forest=0.0
               prop_land=0.0
               edgelength=0.0
               present=float(covercount(f))             ; coverdata from array of counts
               ; to be used for sevaral indices
               if (inclback NE 0) then proportion=float(present/blocksize) else if (landscpix GT 0)
then proportion=float(present/landscpix)
               if (f EQ backval) then proportion=proportion*(inclback NE 0); background-
proportion set to zero if flag is up
               richn=richn+(present GT 0)               ; checks for presence of pixel value =
land cover type
               countpct(b,a,f)=round(100*proportion)
               edge=float(edgecount(f))              ; edgedata written to matrix for output
               edgelength=edge*grainsize
               totel=totel+edgelength
               edgelengths(b,a,f)=edgelength            ; real world edge-length
               edgepct(b,a,f)=round(100*(edge/blocksize)) ; edge realtive to TOTAL AREA in
window
               edgeprop(b,a,f)=round(100*(edge/present))  ; edge relative to AREA of the
CLASS within the window

               ; the original MATHERON index calculated and written to matrix - per class:
               if (present GT 0) then MI=float(edgecount(f)/(sqrt(present)*sqrt(blocksize))) else
MI=-0.1
               MIA(b,a,f)=10*MI
               rif=f ; to be used for where to insert extra values

               if (present GT 0) then begin
                    if (forestpix GT 0) then prop_forest=(f NE landval)*(f NE backval)
*float(present/forestpix) else prop_forest=0
                    prop_land=(f NE backval)*float(present/landscpix)
               end

               ; the classic diversity metrics are summed:
               if (prop_forest GT 0) then begin
                    shannon_f=shannon_f+(prop_forest*alog(prop_forest))
                    simpson_f=simpson_f+prop_forest^2
                    sum_pf=sum_pf+prop_forest
               end
               if (prop_land GT 0) then begin
                    shannon_l=shannon_l+(prop_land*alog(prop_land))
                    simpson_l=simpson_l+prop_land^2
                    sum_pl=sum_pl+prop_land
               end

          endfor;   f - same output cell, LC classes was run through

          countpct(b,a,noclasses)=round(100*forestfraction) ; forest fraction written to array
          if landscpix EQ 0 then begin
            edgedens_land=0
            edgedens_block=0
          endif else begin
            edgedens_land=totel*10000/(landscpix*grainsize^2)
            edgedens_block=totel*10000/(blocksize*grainsize^2)
          endelse
          edgelengths(b,a,noclasses)=totel
          edgelengths(b,a,noclasses+1)=edgedens_land
          edgelengths(b,a,noclasses+2)=edgedens_block
          richslot=noclasses+2              ; outputs richness = "species number"
          countpct(b,a,richslot)=richn
          divind(b,a,0)=richn

          totedgect=outedgect+blockedgect
```

```
			; the 'non-empty' criterion:
			if (landscpix GT 0) then begin
countpct(b,a,noclasses+1)=round(100*(landscpix/blocksize)) ; landscape fraction to array
					; now calculate forest-non-forest (landscape) Matheron index
					mathout=10*float(totedgect/(sqrt(forestpix)*sqrt(landscpix)))
			endif else begin mathout = -1
			endelse

			MIA(b,a,rif)=mathout ; aggregated Matheron written to matrix

			; calculate "Squareness of Patches", sensu Frohn(1998), index for forest-nonforest:
			if outedgect GT 0 then sqp_obj=1-(4*(sqrt(forestpix))/totedgect) else sqp_obj=1
			if sqp_obj LT 0 then sqp =0
			if outedgect GT 0 then sqp_land=1-(4*(sqrt(landscpix))/totedgect) else sqp_land=1
			if sqp_land LT 0 then sqp_land =0
			; write to matrix:
			sqp_mtx(b,a,0)=sqp_obj
			sqp_mtx(b,a,1)=sqp_land

			; now close diversity indices:
			if (richn GT 1) then divind(b,a,1)=-shannon_f else divind(b,a,1)= 0
			if ((richn GT 1) and (forestpix GT 0)) then divind(b,a,2)=1-simpson_f else divind(b,a,2)=
0
			if (richn GT 1) then divind(b,a,3)=-shannon_l else divind(b,a,3)= 0
			if (richn GT 1) then divind(b,a,4)=1-simpson_l else divind(b,a,4)= 0
			divind(b,a,5)=sum_pf
			divind(b,a,6)=sum_pl
			; to give higher values (tow. 1) of SIDI /Simpson's for more diverse compositions

			endfor ;b - next block (next colum)
endfor ;a - next line of blocks (next row)

; end of counting/calculation sequence

; start output sequence

openw,lun,outfile1, /get_lun		; output results for each window cell = ASCII line
print, 'now writing cover results'
for aaa=0,(block_rows-1) do begin	; count through rows - increase Y values
	aaah=(block_rows-1)-aaa; modified Y coordines for 'lower left style'
	for bbb=0,(block_cols-1) do begin		;
		outline1=''
		for classes=0, noclasses+2 do begin
		    outline1=outline1+strcompress(countpct(bbb, aaa, classes))+', '
		endfor; classes
		outline1=outline1+strcompress(bbb)+',
'+strcompress(aaah)+','+string(Geo_E(bbb))+','+string(Geo_N(aaa))
		printf, lun, outline1
	endfor ;bbb

endfor ;aaa				; useful for e.g. Surfer(R)

free_lun,lun  ; _cover written
; Column (noclasses)   : Forest fraction (of landscape)
; Column (noclasses+1) : Landscape fraction (of entire window)
; Column (noclasses+2) : Land cover class Richness
; Column (noclasses+3) : Image X-coordinate
; Column (noclasses+4) : Image Y-coordinate
; Column (noclasses+5) : UTM X-coordinate
; Column (noclasses+6) : UTM Y-coordinate

openw,lun,outfile2, /get_lun		; output results for each window cell = ASCII line
print, 'now writing edge count results'
print, '...writng header line'
outline0=''
writtenclasses=0b
```

```
for head=0,noclasses-1 do begin
        if (writtenclasses GT 0) then outline0=outline0+', '
        outline0=outline0+'cl'+strcompress(head, /remove_all)
        writtenclasses=writtenclasses+1
endfor
outline0=outline0+', total, ED_land, ED_block, X, Y, X_geo, Y_geo'      ; Header line!
printf, lun, outline0

for ccc=0,(block_rows-1) do begin       ; writings to files..
ccch=(block_rows-1)-ccc
        for ddd=0,(block_cols-1) do begin               ;
                outline2=''
                for edgeclasses=0, noclasses+2 do begin
                        outline2=outline2+strcompress(edgelengths(ddd, ccc, edgeclasses))+', '
                endfor ;edgeclasses
                outline2=outline2+strcompress(ddd)+', '+strcompress(ccch)+', '+string(Geo_E(ddd))+',
'+string(Geo_N(ccc))
                printf, lun, outline2 ;
        endfor ;ddd

endfor ;ccc
free_lun,lun ; _edgepct written

openw,lun,outfile3, /get_lun                            ; output results for each window cell = ASCII line
print, 'now writing edge proportion results'
for eee=0,(block_rows-1) do begin       ; writings to files..
        eeeh=(block_rows-1)-eee
        for fff=0,(block_cols-1) do begin               ;
                outline3=''
                for edgepclasses=0, noclasses do begin
                    outline3=outline3+strcompress(edgeprop(fff, eee, edgepclasses))+', '
                endfor ;edgepclasses
                outline3=outline3+strcompress(fff)+', '+strcompress(eeeh)
                printf, lun, outline3 ; write array for this window to output file
        endfor ;fff

        ; plus block coordinates
endfor ;eee     useful for e.g. Surfer(R)
free_lun,lun ; _edgeindex written

openw,lun,outfile4, /get_lun                            ; output results for each window cell = ASCII line
print, 'now writing Matheron results'
for ggg=0,(block_rows-1) do begin       ; writings to files..
        gggh=(block_rows-1)-ggg
        for hhh=0,(block_cols-1) do begin               ;
                outline4=''
                for edgeMIclasses=0, noclasses do begin
                        outline4=outline4+strcompress(MIA(hhh, ggg, edgeMIclasses))+', '
                endfor ;edgeMIclasses
                outline4=outline4+strcompress(hhh)+', '+strcompress(gggh)+', '+string(Geo_E(hhh))+',
                '+string(Geo_N(ggg))
                printf, lun, outline4 ;, hhh,', ', gggh   ; write array for this window to output file
        endfor ;hhh                                                     ; plus block coordinates
endfor ;ggg

free_lun,lun ; Matheron written

openw,lun,outfile5, /get_lun                            ; output results for each window cell = ASCII line
print, 'now writing Diversity results'
; Column 1(A) : Class richness
; Column 2(B) : SHDI object
; Column 3(C) : SIDI obejct
; Column 4(D) : SHDI landscape (object+matrix)
; Column 5(E) : SIDI landscape (object+matrix)
; Column 6(F) : coversum forest = forest mask
; Column 7(G) : coversum landscape = land mask
for iii=0,(block_rows-1) do begin       ; writings to files..
```

```
        iiih=(block_rows-1)-iii
        for jjj=0,(block_cols-1) do begin                ;
                outline5=''
                for divindtypes=0, divs-1 do begin
                        outline5=outline5+strcompress(divind(jjj, iii, divindtypes))+', '
                endfor ; divindtypes
                outline5=outline5+strcompress(jjj)+', '+strcompress(iiih)+', '+string(Geo_E(jjj))+',
                '+string(Geo_N(iii))
                printf, lun, outline5 ;, hhh,', ', gggh   ; write array for this window to output file
        endfor ; jjj
endfor ; iii

free_lun,lun ; Diversities written

openw,lun,outfile6, /get_lun                          ; output results for each window cell = ASCII line
print, 'now writing SQP results'
outline0='SqP_object, SqP_land, X_Image, Y_Image, X_Geogr, Y_Geogr' ; Header line!
printf, lun, outline0
for kkk=0,(block_rows-1) do begin       ; writings to files..
        kkkgb=(block_rows-1)-kkk
        for lll=0,(block_cols-1) do begin              ; write array for this window to output file:
                outline6=strcompress(sqp_mtx(lll,kkk,0))+', '+strcompress(sqp_mtx(lll,kkk,1))+', '
                outline6=outline6+strcompress(lll)+', '+strcompress(kkkgb)+', '+string(Geo_E(lll))+',
                '+string(Geo_N(kkk))
                printf, lun, outline6
        endfor ; lll
endfor ; kkk
free_lun,lun ; SqP values written

; openw,lun,outfile7, /get_lun ; output 'total'Matheron index as ASCII image
print, 'writing Matheron map'
for mmm=0,(block_rows-1) do begin       ; writings to files..
                outline7=''
                ; mmma=(block_rows-1)-mmm - no inversing of y-values here!
                for nnn=0, (block_cols-1) do begin                    ;
                                outline7=outline7+strcompress(MIA(nnn,mmm,noclasses))+', '
                endfor ; nnn
;               outline7=outline7+strcompress(MIA(block_cols-1,mmm,noclasses))
;               printf, lun, outline7
endfor; mmm

; free_lun,lun ; 'Total M map' written
; end of output sequence

winstep=winstep+step_incr; ready with next stepsize
winsize=winsize+win_incr ; ready with next windowsize
winsize=fix(winsize)

endfor ; rounds - to next winodw/step size

endfor ;inputfiles - to next image

free_lun,lun3 ; close parameter file
print, 'finito'

end
```

Parameter file:

```
3
m:\divind\LC_Vends\AAK25LND.RST
0, 3120, 3600, 25, 522000, 6405000
25, 99, 1
0
1
20, 40, 6, 20, 40
```

```
m:\divind\LC_Vends\AAK25NAT.RST
0, 3120, 3600, 25, 522000, 6405000
18, 99, 1
0
1
20, 40, 6, 20, 40
m:\divind\LC_Vends\AAK25FOR.RST
0, 3120, 3600, 25, 522000, 6405000
6, 99, 1
0
1
20, 40, 6, 20, 40
```

10.2 Appendix 1.2 – Patch counting in M-W

```
pro patchcount_mw031129
; mw = moving windows = multiple classes

; 17/8 2001: now works with images up to about 1000*1000 pixels,  for bigger ones -> too slow.
; 18/8 2001: moving windows implemented
; april 2003: multiple window/step sizes implemented
; september 2003 - UTM-georef. option added
; november 2003 - header row added, total NP now minus background patches

filelist='m:\idl_ting\patchcount\nj_lcp_themes_bw.txt' ;  ; pointing to "parameter file", where the run
parameters are stored
n=0b ; number of files in list
image=''
noclasses=0
incols=0l
inrows=0l
headersize=0l
ws_ini=0l
win_incr=0l
winstep=0l
step_ini=0l
step_incr=0l

openr,lun3, filelist, /get_lun
readf,lun3, n

for inputfiles=0,n-1 do begin ; needs modifications to work with >1 files

readf,lun3, image; name of input image
readf,lun3, headersize, incols, inrows, grainsize, UL_E, UL_N
readf,lun3, landscvalue, backval ; pixelvalue for landscape-class, resp. background
readf,lun3, ws_ini, win_incr, winss, step_ini, step_incr ; initial window size, increment in window size,
                                                          ; number of different windows, inital step size
                                                          ; incenrement of stepsize with larger window..
print, 'input image ',image
print, 'columns : ', incols, ' rows: ', inrows

wins=winss-1
winstep=step_ini ; stepsize must be reset before each new image is processed
winsize=ws_ini

openr, lun, image, /get_lun  ; read input image
imagesize=incols*inrows+headersize
image_arr=bytarr(imagesize)
readu, lun, image_arr

print, 'reading input:   ',image

histotal=lonarr(256) ; histogram for entire image
```

```
for byt=0,255 do histotal(byt)=0; reset count array - using pixel value from image array as index in
historgram table
for his=headersize, imagesize-1 do histotal(image_arr(his))=histotal(image_arr(his))+1
types=0b
for bytt=0,255 do if histotal(bytt) GT 0 then types = types + 1; counts number of different types
print, 'land cover types (different pixel values) : ', types
histotab=lonarr(types, 2)
actualtype=0b
for bytval=0,255 do begin
	if histotal(bytval) GT 0 then begin
		histotab(actualtype,0)=bytval
		histotab(actualtype,1)=histotal(bytval)
		if (bytval EQ backval) then backslot=actualtype
		print, ' type ',bytval,' : ',histotal(bytval)
		if (bytval NE backval) then maxtype=bytval
		actualtype=actualtype+1
	endif ; histotab
endfor; bytval

pixcount=headersize			; store input image as 2-D matrix
wholeimage_mtx=bytarr(incols,inrows)
for rc=0,inrows-1 do begin
	for cc=0,incols-1 do begin
		wholeimage_mtx(cc,rc)=image_arr(pixcount)
		pixcount=pixcount+1
	endfor ; cc
endfor ; rc

for rounds=1, winss do begin ; new image - varying window sizes

; create (meaningful?) NAMES for output files:
sizestr=string(winsize)
stepstr=string(winstep)
suf1='_w'+strcompress(sizestr, /remove_all)
suf2='s'+strcompress(stepstr, /remove_all)
split=str_sep(image,'.')
origimagename=split[0]
imagename=origimagename+suf1+suf2
outfile1=imagename+'np_geo.csv'

print, 'output to  ',outfile1

winsz=float(winsize)

blocksize=float(winsz*winsz)
block_cols=fix((incols-winsize+winstep)/winstep)
block_rows=fix((inrows-winsize+winstep)/winstep)
geo_E=fltarr(block_cols)
geo_N=fltarr(block_rows)

pns=intarr(block_cols, block_rows, types+1); defines array for results

rowpatch=0l

; Define output coordinates
outsize=winsize*grainsize
outstep=winstep*grainsize
UL_E_out=UL_E+((outsize-outstep)/2)
UL_N_out=UL_N-((outsize-outstep)/2)
for east=0,block_cols-1 do Geo_E(east)=UL_E_out+outstep*(east+0.5)
for north=0,block_rows-1 do Geo_N(north)=UL_N_out-outstep*(north+0.5)

;START OF MOVING WINDOWS :

for a=0,(block_rows-1) do begin ; calculation starts, runs through blocks - a counts rows (Y values)
print, 'img', inputfiles+1,' / ',n,', ws: ',winsize, ' step: ', winstep,', now analysing row ',a+1,' of', block_rows
print, 'blocksize :', blocksize
```

```
rowpatch=0

for b=0,(block_cols-1) do begin   ; = overlapping windows possible  - b counts columns (X values)
                                  ; extract sub-image for patch-counting:

image_mtx=bytarr(winsize,winsize)
for rc=0,winsize-1 do begin
        for cc=0,winsize-1 do begin
                image_mtx(cc,rc)=wholeimage_mtx((b*winstep+cc),(a*winstep+rc))
        endfor ; cc
endfor ; rc

totpatch=0l

for typenr=0,types-1 do begin ; inside each output cell, run through "patch types"

        landval=histotab(typenr,0) ; classtype/pixel value to count patches for!

        patch_mtx=intarr(winsize,winsize)          ; for storage of assigned patch-number values of each
        pixel
                                                   ; X-Y cordinate system, upper left corner = 0,0
        patchcount=1l
        cols=winsize
        rows=winsize

        ; PATCH COUNTING STARTS:
        foundn=0b

        ; first (horisontal) row - with nothing above:
        for pccol=0, cols-2 do begin ; presence check:
                if (image_mtx(pccol,0) NE landval) then patch_mtx(pccol,0)=0 else begin
                        patch_mtx(pccol,0)=patchcount  ; and if nothing to the right, increase patch number:
                        if (image_mtx(pccol+1,0) NE landval) then patchcount=patchcount+1
                endelse
        endfor; pccol
        ; checking last pixel in first row:
        if (image_mtx(cols-1,0) NE landval) then patch_mtx(cols-1,0)=0 else begin
                patch_mtx(cols-1,0)=patchcount
                patchcount=patchcount+1
        endelse

        ; then for the rest of the (horisontal) rows of the matrix
        for pcrow=1, rows-1 do begin
        ; (1) for first pixel in each row:
        if (image_mtx(0,pcrow) NE landval) then patch_mtx(0,pcrow)=0 else begin; presence check
                patch_mtx(0,pcrow)=patchcount
                foundn=0; no negihbours this far
                ; compare with pixel above:
                if (image_mtx(0, pcrow-1) EQ landval) then begin
                        patch_mtx(0,pcrow)=patch_mtx(0, pcrow-1)
                        foundn=1
                endif
                ; compare with pixel above-right:
                if (image_mtx(1, pcrow-1) EQ landval) then begin
                        patch_mtx(0,pcrow)=patch_mtx(1, pcrow-1)
                        foundn=1
                endif; for cols except the rightmost
        endelse

        ;  (2) then for the rest of the pixels in the row, except the last
        for pccol=1, cols-2 do begin
                if (image_mtx(pccol,pcrow) NE landval) then patch_mtx(pccol,pcrow)=0 else begin;
                presence check
                        foundn=0
                        if (image_mtx(pccol-1, pcrow-1) EQ landval) then begin; compare with pixel above
                   left
                                patch_mtx(pccol,pcrow)=patch_mtx(pccol-1, pcrow-1);
```

```
                    foundn=1
            endif  ; above-left
            if (image_mtx(pccol, pcrow-1) EQ landval) then begin ; compare with pixel above
                    patch_mtx(pccol,pcrow)=patch_mtx(pccol, pcrow-1)
                    foundn=1
            endif ; above
            if (image_mtx(pccol+1, pcrow-1) EQ landval) then begin; compare with pixel above-
            right - exception for last pixel in each row
                    patch_mtx(pccol,pcrow)=patch_mtx(pccol+1, pcrow-1)
                    foundn=1
            endif   ;  above-right
            if (image_mtx(pccol-1, pcrow) EQ landval) then begin ; compare with pixel to the left
                    patch_mtx(pccol,pcrow)=patch_mtx(pccol-1, pcrow)
                    foundn=1
            endif   ; left
            if (foundn EQ 0) then begin
                    patchcount=patchcount+1
                    patch_mtx(pccol,pcrow)=patchcount
            endif ; no neighbours
    endelse; case of pixel in the landscape category
endfor; pccol

; (3) checking last pixel in row:
if (image_mtx(pccol,pcrow) NE landval) then patch_mtx(pccol,pcrow)=0 else begin; presence
check
    foundn=0
    if (image_mtx(pccol-1, pcrow-1) EQ landval) then begin; compare with pixel above left
            patch_mtx(pccol,pcrow)=patch_mtx(pccol-1, pcrow-1);
            foundn=1
    endif  ; above-left
    if (image_mtx(pccol, pcrow-1) EQ landval) then begin ; compare with pixel above
            patch_mtx(pccol,pcrow)=patch_mtx(pccol, pcrow-1)
            foundn=1
    endif ; above
    if (image_mtx(pccol-1, pcrow) EQ landval) then begin ; compare with pixel to the left
            patch_mtx(pccol,pcrow)=patch_mtx(pccol-1, pcrow)
            foundn=1
    endif
    if (foundn EQ 0) then begin
            patchcount=patchcount+1
            patch_mtx(pccol,pcrow)=patchcount
    endif
endelse; case of pixel in the landscape category

endfor; pcrow

; end of preliminary 'classification'

patches=lonarr(patchcount+2); checks presence of patches before/after filtering
for reset=0,patchcount+1 do patches(reset)=0

;trick1 - to avoid filter being affected by backgrond pixels:
for aa=0,winsize-1 do begin
    for bb=0,winsize-1 do begin
            if (image_mtx(bb,aa) NE landval) then patch_mtx(bb,aa)=patchcount+1
            patches(patch_mtx(bb,aa))=patches(patch_mtx(bb,aa))+1
            ; plus counting for initial histogram of patch 'areas'
    endfor; bb
endfor; aa

change=0l
runs=0l
filter_mtx=intarr(winsize,winsize)          ; for storage of assigned patch-number values of each
pixel
                                            ; X-Y cordinate system, upper left corner = 0,0
ul_corner=intarr(4)
top_row=intarr(6)
```

```
ur_corner=intarr(4)
leftside=intarr(6)
kernel=intarr(9)
rightside=intarr(6)
ll_corner=intarr(4)
bottom_row=intarr(6)
lr_corner=intarr(4)

;FINDING MINIMUM PATCH NUMBERS for coherent patches (8-directions):

cols=winsize
rows=winsize

repeat begin

    change=0
    runs=runs+1

    ul_corner=[patch_mtx(0,0),patch_mtx(0,1),patch_mtx(1,0),patch_mtx(1,1)]
    filter_mtx(0,0)=min(ul_corner)

    for top=1,cols-2 do begin
        top_row=[patch_mtx(top-1,0), patch_mtx(top-1,1),patch_mtx(top,0),
        patch_mtx(top,1), patch_mtx(top+1,0), patch_mtx(top+1,1)]
        filter_mtx(top,0)=min(top_row)
    endfor

    ur_corner=[patch_mtx(cols-2,0),patch_mtx(cols-2,1),patch_mtx(cols-1,0),patch_mtx(cols-
    1,1)]
    filter_mtx(cols-1,0)=min(ur_corner)

    for down=1, rows-2 do begin
        leftside=[patch_mtx(0,down-1),patch_mtx(1,down-
        1),patch_mtx(0,down),patch_mtx(1,down),
        patch_mtx(0,down+1),patch_mtx(1,down+1)]
        filter_mtx(0,down)=min(leftside)
        for across=1,cols-2 do begin
            kernel=[patch_mtx(across-1,down-1),patch_mtx(across,down-
            1),patch_mtx(across+1,down-1),patch_mtx(across-
            1,down),patch_mtx(across,down),patch_mtx(across+1,down),
            patch_mtx(across-
            1,down+1),patch_mtx(across,down+1),patch_mtx(across+1,down+1)]
            filter_mtx(across,down)=min(kernel)
        endfor ; across

        rightside=[patch_mtx(cols-2,down-1),patch_mtx(cols-1,down-1),patch_mtx(cols-
        2,down), patch_mtx(cols-1,down),patch_mtx(cols-2,down+1),patch_mtx(cols-
        1,down+1)]
        filter_mtx(cols-1,down)=min(rightside)
    endfor ; down

    ll_corner=[patch_mtx(0,rows-2),patch_mtx(0,rows-1),patch_mtx(1,rows-
    2),patch_mtx(1,rows-1)]
    filter_mtx(0,rows-1)=min(ll_corner)

    for bottom=1,cols-2 do begin
        bottom_row=[patch_mtx(bottom-1,rows-2), patch_mtx(bottom-1,rows-
        1),patch_mtx(bottom,rows-2), patch_mtx(bottom,rows-1),
        patch_mtx(bottom+1,rows-2), patch_mtx(bottom+1,rows-1)]
        filter_mtx(bottom,rows-1)=min(bottom_row)
    endfor

    lr_corner=[patch_mtx(cols-2,rows-2),patch_mtx(cols-2,rows-1),patch_mtx(cols-1,rows-
    2),patch_mtx(cols-1,rows-1)]
    filter_mtx(cols-1,rows-1)=min(lr_corner)

    ;count changes in landscape pixels:
```

```
		for aa=0,cols-1 do begin
			for bb=0,rows-1 do begin
				if (image_mtx(bb,aa) EQ landval) then begin
					if not(filter_mtx(bb,aa) EQ patch_mtx(bb,aa)) then change=change+1
				end ; if
			endfor ; bb
		endfor ; aa

		;swap before going back:
		for aa=0,cols-1 do begin
			for bb=0,rows-1 do begin
				patch_mtx(bb,aa)=filter_mtx(bb,aa)
			endfor; bb
		endfor; aa

		;trick2 - to avoid influence of patches spreading over background:
		for aa=0,cols-1 do begin
			for bb=0,rows-1 do begin
				if not(image_mtx(bb,aa) EQ landval) then patch_mtx(bb,aa)=patchcount+1
			endfor; bb
		endfor; aa

	endrep until (change EQ 0)

	; check possible patches for existence after filtering


	for reset=0,patchcount+1 do patches(reset)=0

	for aa=0,cols-1 do begin
		for bb=0,rows-1 do begin
			patches(patch_mtx(bb,aa))=patches(patch_mtx(bb,aa))+1
		endfor; bb
	endfor; aa

	;count number of different patches in (sub)landscape:
	count_filtered=0l
	for cc=0,patchcount do begin
		count_filtered=count_filtered+(patches(cc) GT 0)
	endfor; cc

	totpatch=totpatch+count_filtered

	pns(b,a,typenr)=count_filtered ; storing result from this window/block

endfor  ; typenr; next land cover type

totpatch=totpatch-pns(b,a,backslot)

pns(b,a,typenr)=totpatch
rowpatch=rowpatch+totpatch

endfor  ;b - next block (next colum)

print, 'Row ',a,' Column ',b, ', Summed no. of Patches: ', rowpatch
endfor   ;a - next line of blocks (next row)

; END MOVING WINDOWS

openw,lun,outfile1, /get_lun                    ; output results for each window cell = ASCII line
print, 'now writng header line'
outline0=''
writtenclasses=0b
for head=0,255 do begin
	if (histotal(head) GT 0) then begin
		if (writtenclasses GT 0) then outline0=outline0+', '
		outline0=outline0+'cl'+strcompress(head)
```

```
                    writtenclasses=writtenclasses+1
          end; if
endfor ; head
outline0=outline0+', total, X, Y, X_geo, Y_geo'
printf, lun, outline0

print, 'now writing patch numbers'
for aaa=0,(block_rows-1) do begin       ; count through rows - increase Y values
          aaah=(block_rows-1)-aaa; modified Y coordines for 'lower left style'
          for bbb=0,(block_cols-1) do begin              ;
                    outline1=''
                    for classes=0, types do begin
                              outline1=outline1+strcompress(pns(bbb, aaa, classes))+', '
                    endfor; classes
                    outline1=outline1+strcompress(bbb)+', '+strcompress(aaah)+','+string(Geo_E(bbb))+',
                    '+string(Geo_N(aaa))
                    printf, lun, outline1                         ; write array for this window to output file
          endfor ;bbb                                             ; plus block coordinates
endfor  ;aaa

free_lun,lun  ; _np written

winstep=winstep+step_incr ; ready for next stepsize
winstep=fix(winstep)
winsize=winsize+win_incr ;  ready for next window size
winsize=fix(winsize)

endfor ; rounds - to next winodw/step size

endfor ;inputfiles - to next image

free_lun,lun3  ; close parameter file

print, 'THE END'

end
```

Parameter file:

```
3
m:\divind\LC_Vends\LCP25LND.RST
0, 3120, 3600, 25, 522000, 6405000
1, 99
40, 40, 5, 40, 40
m:\divind\LC_Vends\LCP25NAT.RST
0, 3120, 3600, 25, 522000, 6405000
1, 99
200, -40, 5, 200, -40
m:\divind\LC_Vends\LCP25FOR.RST
0, 3120, 3600, 25, 522000, 6405000
1, 99
200, -40, 5, 200, -40
```

10.3 Appendix 1.3 – Spatial degradation of binary maps

```
pro binary_degprop_030317
; reads list of files to spatially degrade, plus maximal degradation factor
; then degrades each file to a number of cell sizes and writes to output files
; this version for ERDAS 7.5 (.gis) files with 128 bytes header
; assumes one-byte pixels !
; NO OVERLAP in this version.
; Input, list in the following format:
;    number of files to treat (and then for each file)
;    name and path of input image file
;    columns and rows in input image
;    maximum degrade factor
;    cut or threshold value of cover percentage for inclusion in output image

filelist='c:\NCN\IDL_ting\AIS\degraster.txt'
nfiles=0b ; number of files in list  alt. read, nfiles, prompt='number of files : '
openr, lun3, filelist, /get_lun
readf, lun3, nfiles
print, nfiles, ' files to treat'

infile=''
rows=0l
cols=0l

for i=0,nfiles-1 do begin   ; goes through input files in list

        readf, lun3, infile
        readf, lun3, cols, rows
        readf, lun3, maxdegrade
        readf, lun3, cut

        openr, lun, infile, /get_lun  ; read input image
        image_arr=bytarr(cols*rows+128)
        readu, lun, image_arr

        print, 'reading ',infile

        pixcount=128l                              ; store input image as 2-D matrix
        image_mtx=bytarr(cols,rows)
        for rc=0,rows-1 do begin
                for cc=0,cols-1 do begin
                        image_mtx(cc,rc)=image_arr(pixcount)
                        pixcount=pixcount+1
                endfor ; cc
        endfor ; rc

        for degfactor=2,maxdegrade do begin

                degcols=0
                degrows=0
                degcols=long(fix(cols/degfactor))
                degrows=long(fix(rows/degfactor))
                deg_img=bytarr(degcols,degrows)
                prop_img=bytarr(degcols,degrows)
                degsize=0.0
                degsize=float(degcols*degrows)

                ; start of actual degradation:
                for k=0,(degrows-1) do begin     ; going through blocks = output cells/pixels
                        for l=0,(degcols-1) do begin
                                cellsum=0.0
                                average=0
                                result=0b
                                for n=0,degfactor-1 do begin            ; collect/sum values over output cell
```

```
                        for p=0,degfactor-1 do begin
                            x=l*degfactor+n
                            y=k*degfactor+p
                            cellsum=cellsum+(image_mtx(x,y) GT 0) ; add to sum
                        endfor ; p
                    endfor ; n
                    average=float(cellsum/(degfactor*degfactor))
                    proppct=average*100
                    deg_img(l,k)=1*(proppct GT cut)
                    prop_img(l,k)=proppct
                endfor ; l
            endfor ; k

            print, 'finished degrading ', infile, ' with deg. factor', degfactor, 'with threshold', cut

            deg_arr=bytarr(degsize)
            for r_row=0,(degrows-1) do begin                ; back to array for export - import
                for s_col=0,(degcols-1) do begin
                    deg_arr((r_row*degcols)+s_col)=deg_img(s_col,r_row)
                endfor ; s_col
            endfor ; r_row

            prop_arr=bytarr(degsize)
            for r_row=0,(degrows-1) do begin                ; back to array for export - import
                for s_col=0,(degcols-1) do begin
                    prop_arr((r_row*degcols)+s_col)=prop_img(s_col,r_row)
                endfor ; s_col
            endfor ; r_row

            dsuffix='.d_'+strcompress(degfactor, /remove_all)
            psuffix='.p_'+strcompress(degfactor, /remove_all)
            degoutfile=infile+dsuffix                       ; write to export files
            propoutfile=infile+psuffix
            openw,lun,degoutfile, /get_lun
            writeu,lun,deg_arr
            free_lun,lun
            openw,lun,propoutfile, /get_lun
            writeu,lun,prop_arr
            free_lun,lun

            print, 'output to ', degoutfile, propoutfile

        endfor ; degfactor

endfor  ; i

print, 'THE END'

end
```

Parameter file:

```
1
c:\NCN\IDL_ting\AIS\aak_natt.lan
1560, 1800
60
40
```

10.4 Appendix 1.4 – Spatial degradation of thematic maps

```
pro degrweight040106
; FOR DEGRADATION OF LAND COVER (Choropleth) MAPS and similar data...
; reads list of files to spatially degrade and weight file (if available)
; then degrades each file to a number of cell sizes and writes to output files
; this version for ERDAS 7.5 (.gis) files with 128 bytes header
; assumes one-byte pixels !

filelist='m:\idl_ting\deg_jord.txt'
nfiles=0b ; number of files in list  alt. read, nfiles, prompt='number of files : '
openr, lun3, filelist, /get_lun
readf, lun3, nfiles
print, nfiles, ' files'
infile=''
rows=0l
cols=0l
header=0l
minclass=0l
maxclass=0l
weightfile=''
degfactor=0l
suffix=''

for i=0,nfiles-1 do begin   ; goes through input files in list
        readf, lun3, infile
        readf, lun3, header, cols, rows, maxclass
        readf, lun3, mindegrade, maxdegrade
        readf, lun3, weightfile
        openr, lun, infile, /get_lun  ; read input image
        image_arr=bytarr(cols*rows+header)
        readu, lun, image_arr
        print, 'read ',infile

        pixcount=header              ; store input image as 2-D matrix
        image_mtx=bytarr(cols,rows)
        for rc=0,rows-1 do begin
                for cc=0,cols-1 do begin
                        image_mtx(cc,rc)=image_arr(pixcount)
                        pixcount=pixcount+1
                endfor ; cc
        endfor ; rc
        ; convert to include histogram creation, check for min-max values ?

        for degfactor=mindegrade,maxdegrade do begin ; start new degradation size
                cwtab=fltarr(2,256) ; table for class weights
                count=0
                classno=0
                classwt=0.0
                for j=0,255 do begin ; reset cw-table (all classes equal weight)
                        cwtab(0,j)=j
                        cwtab(1,j)=1
                endfor ;

                if not (weightfile eq 'x') then begin
                        openr, lun1, weightfile, /get_lun   ; assign name to weight-tablefile
                        while not eof(lun1) do begin
                                readf, lun1, classno, classwt
                                cwtab(0,count)=classno
                                cwtab(1,count)=classwt
                                count=count+1
                        endwhile
                endif

                dc=0
                dr=0
```

```
        dc=long(fix(cols/degfactor))
        dr=long(fix(rows/degfactor))
        deg_img=bytarr(dc,dr)
        histotab=fltarr(2,256)
        select=0
        degsize=0l
        degsize=long(dc*dr)
        ; start of actual degradation
        for k=0,(dr-1) do begin     ; going through blocks = output cells/pixels
            for l=0,(dc-1) do begin
              for m=0,maxclass do begin ; reset histogram
                histotab(0,m)=0
                histotab(1,m)=0
              endfor ; m
              for n=0,degfactor-1 do begin              ; build histogram for block
                  for p=0,degfactor-1 do begin
                    x=l*degfactor+n
                    y=k*degfactor+p
                    histotab(0,image_mtx(x,y))=histotab(0,image_mtx(x,y))+1 ; incr value
                  endfor ; p
              endfor ; n
              maxwtd=0.0
              select=0
              for q=1,maxclass do begin                 ; find highest value = output
                  histotab(1,q)=histotab(0,q)*cwtab(1,q) OBS - ignore counts of zero-values!
                  if (histotab(0,q) gt maxwtd) then begin
                      maxwtd=histotab(0,q)
                      select=q
                  endif
              endfor ; q
              deg_img(l,k)=select
            endfor ; l
        endfor ; k

        degfactorprint=round(degfactor)
        print, 'finished degrading ', infile, ' with deg. factor', degfactorprint

        deg_arr=bytarr(degsize)
        for r_row=0,(dr-1) do begin                     ; back to array for export - import
        for s_col=0,(dc-1) do begin
        deg_arr((r_row*dc)+s_col)=deg_img(s_col,r_row)
        endfor ; s
        endfor ; r
        suffix='.d_'+strcompress(degfactorprint, /remove_all)
        outfile=infile+suffix                           ; write to export file
        openw,lun,outfile, /get_lun
        writeu,lun,deg_arr
        free_lun,lun

        print, 'output to ', outfile

    endfor ; degfactor - next (smaller) image
endfor ; i

print, 'THE END'
end
```

Parameter file:

```
1
m:\mapinfo\Vendsyssel\jord\degrade\jordtype.rst
0, 3120, 3600, 10
2, 40
x
```

* * *

10.5 Appendix 1.5 – Per-window averaging of continuous field value images

pro MW_average040202coord_realidr

```
; This program should be applied to land-cover data in ERDAS 7.5 (.gis) format
; or similar formats like CHIPS, assuming single band
; Input : images, list with image and moving windows data in the following format :
; (once)
; no. of images
; number of land cover classes
; initial window size, increase in winsize, no. of diff. windows, initial step, increase in step (once)
; (then for each image)
; filename
; For each image: headerlength (no. of pixels to be skipped), cols, rows, pixelsize
; (no. of classes of interest)
; (outfile - created automatically in this version)
;
; Output are comma separated ASCII (.csv) files with each of the cover classes'
; - percentage of sublandscape area, richness'=no. of classes present in last column

; Created  21/10 2003, based on older script (2001-2003) for extraction of spatial metrics in moving
windows
; Modified 22/10 2003 to read input UTM coordinates for image and output coordinates for centre of esch
cell/window
; Modified 22/10 2003 to output Idrisi header (v 2.0, = .doc file) along with binary (.rst) image
; 24/10 outputs area proportion file, calc on area corresp. to output window, for use as mask
; NB. Check settings for file extensions in Idrisi/Environment

filelist='m:\IDL_ting\CLC_avg2coord.txt'; where the run parameters are stored
n=0b ; number of files in list
noclasses=0  ; read from info-file
backval=0    ; read from info-file
inclback=0   ; read from info-file
subst=0
cols=0l                                    ; etc.
rows=0l
headersize=0l
grainsize=0.0
ws_ini=0l
winstep=0l
step_ini=0l
step_incr=0l

openr,lun3,filelist, /get_lun
readf,lun3,n       ; Read over-all parameter(s): number of input images
image='' ; initiating strings for filenames

for inputfiles=0,n-1 do begin   ; read list of ID-images, valuetables and outputimages (to be implemented)
readf,lun3, image                  ; reading image-specific parameters
readf,lun3, headersize, cols, rows, grainsize, UL_E, UL_N
readf,lun3, noclasses, backval, inclback, subst ; should background pixels be included in calculations
(yes if inclback <> 0)?
; if yes then use assigned (substitute) number for the background pixels in calculations of average
values
if (inclback GT 0) then print, 'Background value pixels included in calculations'
print, 'Background value is : ',backval
readf,lun3, ws_ini, win_incr, winss, step_ini, step_incr ; initial window size, increment in window size,
                                                    ; number of different windows, inital
                                                    step size
                                                    ; incenrement of stepsize with larger
window..
wins=winss-1
```

```
maxwin=ws_ini+(win_incr*wins)

winstep=step_ini ; stepsize must be reset before each new image is processed
winsize=ws_ini

for rounds=1, winss do begin ; new image - varying window sizes

sizestr=string(winsize)  ; Create names for output files
stepstr=string(winstep)
suf1='x_w'+strcompress(sizestr, /remove_all)
suf2='s'+strcompress(stepstr, /remove_all)
split=str_sep(image,'.')
origimagename=split[0]
imagename=origimagename+suf1+suf2
outfile1=imagename+'_avg.csv'
outfile1img=imagename+'_avg.rst'
outfile1prop=imagename+'_incl.rst'
outfile1doc=imagename+'_avg.doc'
outfile1pdoc=imagename+'_incl.doc'
outfile2=imagename+'_cov.csv'


openr, lun, image, /get_lun  ; read input image to memory
print, 'Now reading ',image
image_arr=fltarr(cols*rows+headersize)
readu, lun, image_arr
free_lun,lun ; Close input image

print, 'output to ', outfile1

pixcount=headersize                    ; store input image as 2-D matrix
image_mtx=fltarr(cols,rows)
for rc=0,rows-1 do begin
        for cc=0,cols-1 do begin
                image_mtx(cc,rc)=image_arr(pixcount)
                pixcount=pixcount+1
        endfor ; cc
endfor ; rc

block_cols=0l
block_rows=0l
winsz=float(winsize)
blocksize=float(winsz*winsz)
block_cols=fix((cols-winsize+winstep)/winstep)
block_rows=fix((rows-winsize+winstep)/winstep)
x=0u
y=0u

; Define output coordinates
geo_E=fltarr(block_cols)
geo_N=fltarr(block_rows)
outsize=winsize*grainsize
print,'Calculation window size : ',outsize
outstep=winstep*grainsize
print,'Output window size : ',outstep
UL_E_out=long(UL_E+((outsize-outstep)/2))        ; Upper Left corner coordinates of output image
UL_N_out=long(UL_N-((outsize-outstep)/2))
LR_E_out=long(UL_E_out+outstep*block_cols)   ; Lower Right corner coordinates of output image
LR_N_out=long(UL_N_out-outstep*block_rows)
for east=0,block_cols-1 do Geo_E(east)=UL_E_out+outstep*(east+0.5) ; writing coordinates for each
output pixel to array
for north=0,block_rows-1 do Geo_N(north)=UL_N_out-outstep*(north+0.5) ; for use when .csv file is
imported to Surfer-grid

; Define and reset count parameters
value=0.0
minimum=99.99
```

```
maximum=0.0
prob=fltarr(256)
percent=0.0
avg_mtx=fltarr(block_cols, block_rows)  ; average value for outout cells
prop_mtx=fltarr(block_cols, block_rows)  ; proportion of non-background in area corresp. to output cells
average=0.0

for a=0,(block_rows-1) do begin ; calculation starts, runs through blocks - a counts rows (Y values)

	aa=(block_rows-1)-a ; lowerleft coordinate system - better for Surfer import! ignored for the
	moment!!
	for b=0,(block_cols-1) do begin 		; = overlapping windows - b counts columns (X values)
		inclpix=0.0
		cellsum=0.0
		for d=0,(winsize-1) do begin	; counting inside window - d counts rows
			for e=0,(winsize-1) do begin ; counting inside window - e counts columns
				x=(b*winstep+e)
				y=(a*winstep+d)
				value=image_mtx(x,y)
				if (inclback GT 0) then begin
					if (value EQ backval) then value=subst
					cellsum=cellsum+value; adds to sum
					inclpix=inclpix+1
				endif else begin if (image_mtx(x,y) NE backval) then begin
					cellsum=cellsum+value; adds to sum
					inclpix=inclpix+1
					endif
				endelse
				; covercount(value)=covercount(value)+1	; THIS is where the actual
				counting takes place - directly in the array
			endfor ;e
		endfor ;d

		startbox=fix((winsize-winstep)/2)
		endbox=fix((winsize-winstep)/2+(winstep-1))
		inclsum=0.0
		inclprop=0.0
		for i=startbox, endbox do begin
			for j= startbox, endbox do begin
				x=(b*winstep+j)
				y=(a*winstep+i)
				value=image_mtx(x,y)
				if (value NE backval) then inclsum=inclsum+1
			endfor ; j
		endfor ; i
		inclprop=inclsum/(winstep*winstep)
		prop_mtx(b,a)=inclprop

		; INDEX CALCULTATION:
		richn=0s
		richslot=0s
		average=0.0

		if (inclpix GT 0) then average=(cellsum/inclpix) else average=0 ; average value in block
		calculated as floating point number
		avg_mtx(b,a)=average		; and stored in matrix for subseq. output
		if (average LT minimum) then minimum=average
		if (average GT maximum) then maximum=average

	endfor ;b - next block (now go to next colum)
endfor ;a - next line of blocks (now go to next row)

; Finished Moving-Windows, start output

openw,lun,outfile1, /get_lun				; output results for each window cell = ASCII line
print, 'now writing average values to .csv'
for aaa=0,(block_rows-1) do begin	; count through rows - increase Y values
```

```
aaah=(block_rows-1)-aaa                          ; modified Y coordines for 'lower left style'
for bbb=0,(block_cols-1) do begin                ;
outline1=''
outline1=outline1+strcompress(avg_mtx(bbb, aaa))+', '
outline1=outline1+strcompress(bbb)+', '+strcompress(aaah)+', '+string(Geo_E(bbb))+',
'+string(Geo_N(aaa))
printf, lun, outline1 ;                  ; write array for this window to output file
    endfor ;bbb
endfor ;aaa


free_lun,lun  ; _averages written to file

block_col2=long(block_cols)
block_row2=long(block_rows)
outsize=long(block_col2*block_row2)
avg_arr=fltarr(outsize)
prop_arr=fltarr(outsize)

for r_row=0,(block_row2-1) do begin                        ; back to arrays for export - import
   for s_col=0,(block_col2-1) do begin
                                    avg_arr((r_row*block_col2)+s_col)=avg_mtx(s_col,r_row)
                                    prop_arr((r_row*block_col2)+s_col)=prop_mtx(s_col,r_row)
                  endfor ; s_col
endfor ; r_row

openw,lun,outfile1img, /get_lun ; write to export file
print, 'now writing average values to binary image (.rst) file'
writeu,lun, avg_arr
free_lun,lun

openw,lun,outfile1prop, /get_lun ; write to export file
print, 'now writing land proportion values to binary image (.rst) file'
writeu,lun, prop_arr
free_lun,lun

openw,lun,outfile1doc, /get_lun ; write to export file
print, 'now writing average parameters Idrisi documentation (.doc) file'

printf,lun,'file title  : '
printf,lun,'data type   : real'
printf,lun,'file type   : binary'
outline='columns     :'+strcompress(block_cols)
printf,lun, outline
outline='rows        :'+strcompress(block_rows)
printf,lun, outline
printf,lun,'ref. system : utm-32n'
printf,lun,'ref. units  : m'
printf,lun,'unit dist.  : 1.0000000'
outline='min. X      :'+strcompress(UL_E_out)
printf,lun, outline
outline='max. X      :'+strcompress(LR_E_out)
printf,lun, outline
outline='min. Y      :'+strcompress(LR_N_out)
printf,lun, outline
outline='max. Y      :'+strcompress(UL_N_out)
printf,lun, outline
printf,lun,'pos''n error : unknown'
outline='resolution  :'+strcompress(outstep)
printf,lun, outline
outline='min. value  :'+strcompress(minimum)
printf,lun, outline
outline='max. value  :'+strcompress(maximum)
printf,lun, outline
printf,lun,'value units : undefined'
printf,lun,'value error : unknown'
printf,lun,'flag value  : none'
```

```
printf,lun,'flag def''n  : none'
printf,lun,'legend cats : 0'

free_lun,lun

openw,lun,outfile1pdoc, /get_lun ; write to export file
print, 'now writing average parameters Idrisi documentation (.doc) file'

printf,lun,'file title  : '
printf,lun,'data type   : real'
printf,lun,'file type   : binary'
outline='columns     :'+strcompress(block_cols)
printf,lun, outline
outline='rows        :'+strcompress(block_rows)
printf,lun, outline
printf,lun,'ref. system : utm-32n'
printf,lun,'ref. units  : m'
printf,lun,'unit dist.  : 1.0000000'
outline='min. X      :'+strcompress(UL_E_out)
printf,lun, outline
outline='max. X      :'+strcompress(LR_E_out)
printf,lun, outline
outline='min. Y      :'+strcompress(LR_N_out)
printf,lun, outline
outline='max. Y      :'+strcompress(UL_N_out)
printf,lun, outline
printf,lun,'pos''n error : unknown'
outline='resolution  :'+strcompress(outstep)
printf,lun, outline
outline='min. value  :  0'
printf,lun, outline
outline='max. value  :  1'

printf,lun, outline
printf,lun,'value units : undefined'
printf,lun,'value error : unknown'
printf,lun,'flag value  : none'
printf,lun,'flag def''n  : none'
printf,lun,'legend cats : 0'

free_lun,lun

winstep=winstep+step_incr; ready with next stepsize
winsize=winsize+win_incr ; ready with next windowsize
winsize=fix(winsize)

endfor ; rounds - to next winodw/step size

endfor ;inputfiles - go to next image

free_lun,lun3 ; close parameter file
print, 'finito'
end
```

Parameter file:

```
1
c:\ncn\geodata\AIS\divind\IDRIS\250m\CLCDKHEM.rst
0, 1208, 1480, 250, 441000, 6408000
100, 0, 0
20, 10, 4, 4, 2
```

11 Appendix 2 - Software used during the study

The programs are listed alphabetically, when appropriate the reference to the entry in the list of References is given.

Fragstats for Windows: Calculation of spatial metrics from raster images at patch, class and landscape level. Academic freeware, maintained at University of Massachusetts, Amherst. Version 3.3, 2002. Available through project web site: http://www.umass.edu/landeco/research/fragstats/fragstats.html

Hovey's Idrisi MapWalker (Hovey 1998): Smoothed averaging of raster images. Freeware, currently not available for download. Created for Research Branch, Ministry of Forests, Revelstoke, British Columbia, Canada, by Fred Hovey who can be contacted by e-mail at: ursus_soft@yahoo.com

IDL (Research Systems Inc. 1999): Interactive Data Language - implementation of matrix/image processing calculations. Version 5.2.1. Commercial software, company site: www.rsinc.com

Idrisi (Eastman 1997): GIS/image processing. Reclassification, ranking, Moran's I etc. Version 2.010, compiled 1998. Commercial software, educational discounts, information at: http://www.idrisi.clarku.edu

MapInfo Professional: Gridding of vector data to raster format (using the "Vertical Mapper" extension). Version 7.0, 2002. Commercial software, manufactured by Clark Labs, educational discounts, information at: http://www.mapinfo.com/

Microsoft Office for Windows 2000 package: commercial software, manufactured by Microsoft. Version SR-1 (9.0.3821) Product web site: http://www.microsoft.com/uk/office/ Includes

MS Excel: Used for basic statistics, drawing graphs

MS Access: Literature database

MS Power Point: Illustrations (diagrams, text)

Paint Shop Pro: Illustrations (images). Version 7.04. Commercial software, manufactured by Jasc Software, product web site: http://www.jasc.com/products/paintshoppro/

SILVICS (Satellite Image Land Vegetation Integrated Classification System): Topographic normalisation, ortho-rectification, image segmentation. Freeware, developed by Niall McCormick under contract to JRC-SAI for the Irish Forest Inventory and Planning System Project. Available from http://eurolandscape.jrc.it/forest/silvics/

Surfer (Keckler 1997): Import and display of GRID-files. Version 6.04 (Win32). Commercial software, manufactured by Golden Software. Information at: http://www.goldensoftware.com/products/surfer/surfer.shtml

WinChips (Hansen 2000): image processing, statistics, arithmetic operations. Version 4.7, January 2000. Available from http://www.geogr.ku.dk/chips/index.htm

All web sites were accessed between 1 and 3 March 2004.