Abstract

Image processing relies involves the manipulation of images, which can have a variety of representations. Image representations and the operations that can be done on them are introduced. True color and indexed color representations, pixel structure, neighborhoods and connected components are defined. Examples of connected component location using IDL are demonstrated.
What is an Image?

Possible definitions

• An optical reproduction of an object formed by a lens or mirror

• A set of values of a function corresponding to a particular subset of a domain

• An array of numbers

• ...
Representation System

• We assume a physical world that can produce real images.

• We assume an abstract world of mathematical objects.

• A given real image can be described symbolically by specifying a mathematical object.

• The mathematical objects can be represented by data structures that can be manipulated in computers, transmitted over communication links, and displayed on output devices.
The boxes represent operations and the arrows represent data structures.

Data and operations are application dependent.

Image processing algorithms use general methods in application specific tasks. How shall we describe and represent algorithms to achieve the maximum flexibility and power?
Image Processing

We use image processing to

- Enhance the visual qualities of an image
- Extract information from an image (or set of images)

Both uses of image processing are based on mathematical operations, which may be implemented in physical devices or computer algorithms.
Image Algorithms

An image algorithm can be either:

• A mathematical transformation that operates on mathematical objects

• A computational transformation that operates on data structures

The mathematical form is expressed with equations and the computational form is expressed with computer programs.
Image Representations

The three representations

- Physical
- Mathematical
- Data Structure

are all valid forms for an image. All can represent an observation from the physical world.

In this course we will work with the mathematical and data structure forms. The transformations out of and back into the real world are done by application specific sensor and display systems.
Elements of Image Processing

- Color image Processing
- Image Restoration
- Image Enhancement
- Image Acquisition
- Wavelets and Multiresolution Processing
- Compression
- Morphological Processing
- Segmentation
- Representation & Description
- Object Recognition

Knowledge base
Image

Scalar images are typically displayed by interpreting each array element as a gray value. The surface plots shown on the next page were constructed from a portion of the image shown below.
A 2D signal can also be defined on a continuous or a discrete domain. Digital images can be represented by data structures over a discrete 2D domain.
Images as Data Structures

Image data structures typically contain descriptive information contained in some kind of header and the digital information needed to reconstruct an image array.

Common image formats: TIFF, JPEG, GIF, WMF, BMP, EPS, PNG, Raw

Specialized image formats: ENVI, AVIRIS, FITS, ...

Image processing systems typically provide tools to read and write a selection of image formats.

IDL provides functions and procedures to read and write many image formats. Notable exceptions: GIF and compressed TIFF (because of LZW license restrictions.)
IDL Demonstration

fname='c:\rsi\user\img\imgdata\LENA.tif'
A=Read_Tiff(fname)
sz=size(A)
window,/free,xsize=sz[1],ysize=sz[2]
A=reverse(A,2)
TV,A
wshow
Images as Arrays

An image can be represented as an array of pixels. An image of width \( w \) pixels and height \( h \) pixels is equivalent to an array \( A \) of size \( w \times h \) pixels.

The pixel in position \([x, y]\) has value \( A[x, y] \).

IDL always allocates and references data in row-major format. That is, the subscript order is [column,row], which corresponds to the \([x, y]\) format that is natural for images.
True Color Image
Color Planes

Original

Red

Green

Blue
Indexed Color Image

Palette

Zoomed Section
Objects and Structure

- Objects are often recognized by their spatial structure.

- Algorithms are needed to find spatial structure in images.

- Other algorithms can make use of information about spatial structure to identify objects and make measurements of important properties.

- A basic process in analyzing spatial structure is finding connected components.

- Connected components are often associated with objects or parts of objects.
Region Boundaries

How many white regions are there in this image?

The black pixels apparently divide the image into regions by forming region boundaries.

The number of white regions actually depends upon the way in which we define pixel neighbors.

If white pixels touch only on their sides then there are three white regions. If they touch on their corners and sides then there is one white region.
Region Boundaries

How many black objects are in this image?

How many white regions?

The number depends upon how individual pixels connect.

Pixel connection is defined in terms of *neighborhoods*
Neighborhoods

The neighborhood of a pixel is the set of pixels in the image that it touches.

The 4-neighbors of $p$ touch $p$ on the sides (dark pixels)

The $d$-neighbors of $p$ touch $p$ on the corners (shaded pixels)

Note: The pixel $p$ is not a member of its neighborhood.
Neighborhoods

The 8-neighbors of p touch p on either the sides or the corners.

The 8-neighbor set is the union of the 4-neighbor and d-neighbor sets.

Pixels outside the image are not in the neighborhood.
Connectivity

- Two pixels are connected if they are neighbors that share a common property that defines a component. The property may be color, brightness, range of brightness values, or anything else of interest.

- Pixels may be 4-connected, d-connected, 8-connected and m-connected.

- Pixels $p$ and $q$ are 4-connected if $p$ and $q$ both have the required property and $q$ is in the 4-neighborhood of $p$.

- Pixels $p$ and $q$ are d-connected if $p$ and $q$ both have the required property and $q$ is in the d-neighborhood of $p$.

- Pixels $p$ and $q$ are 8-connected if $p$ and $q$ both have the required property and $q$ is in the 8-neighborhood of $p$. 

Connected Components

A set $S$ of pixels is a connected component if there is at least one path in $S$ that joins every pair $\{p, q\}$ of pixels in $S$. The path must contain only pixels in $S$.

A, B and C are connected components under 4-connectivity.

B and C are joined under 8 or m connectivity.

How does the kind of connectivity affect the number of holes in D?
Connected Components

Three connected components are shown in this image.

The green objects both have the same property but are not connected.
LABEL_REGION does a binary search.

The search program has found three objects.

Overlapping objects with different colors (values) are treated as a single object.
IDL LABEL_REGION PROGRAM

The LABEL_REGION function consecutively labels all of the regions, or blobs, of a bi-level image with a unique region index.

The argument for LABEL_REGION is an n-dimensional bi-level integer type array: only zero and non-zero values are considered. The result of the function is an integer array of the same dimensions with each pixel containing its region index. A region index of zero indicates that the original pixel was zero and belongs to no region. Output values range from 0 to the number of regions.

Result = LABEL_REGION( Data [, /ALL_NEIGHBORS] [, /ULONG] )

Data
A n-dimensional image to be labeled. Data is converted to integer type if necessary. Pixels at the edges of Data are considered to be zero.

ALL_NEIGHBORS
Set this keyword to indicate that all adjacent neighbors to a given pixel should be searched. (This is sometimes called 8-neighbor searching when the image is 2-dimensional). The default is to search only the neighbors that are exactly one unit in distance from the current pixel (sometimes called 4-neighbor searching when the image is 2-dimensional).

ULONG
Set this keyword to specify that the output array should be an unsigned long integer.
Sequential Search Procedure

The binary labeling algorithm can be used sequentially by applying it separately to each color.

First, find out how many colors are in the image array. This can be done by using the HISTOGRAM function.

```
ha=Histogram(A)
ia=Where(ha GT 0)
Print,ia ;Prints 11 26 180 215
Print,ha[ia] ;Prints 5816 14337 12872 216975
```

This gives us the value and count of each index (color).
Sequential Search Procedure

We will now construct a separate binary image for each color value.

\[
\begin{align*}
A1 &= \text{BytArr}(500,500); \text{ Construct a blank image} \\
k &= \text{Where}(A \ EQ \ 11) \\
A1[k] &= 1 \\
C1 &= \text{Label\_Region}(A1) \\
\text{Window,}/\text{free},Xsize=500,Ysize=500,Title='Value=11' \\
\text{TV,C1}
\end{align*}
\]

Note: The top three lines could have been done with just the single line

\[
A1 = A \ EQ \ 11
\]
LABEL_REGION does a binary search.

The search program has found the one object that has value=11.
IDL LABEL_REGION PROGRAM

LABEL_REGION does a binary search.

A2=A EQ 26
C2=LABEL_REGION(A2)
TV,C2

The search program has found the two objects that have value=26.
IDL LABEL_REGION PROGRAM

LABEL_REGION does a binary search.

\[
\begin{align*}
A_3 &= A \text{ EQ } 180 \\
C_3 &= \text{LABEL\_REGION}(A_3) \\
& \text{TV}, C_3
\end{align*}
\]

The search program has found the two objects that have value=180.

Original Image

Search Results
Searching an Indexed Color Image

Find the different objects represented by the colored blobs in this image. The image is represented by an index array and a color palette.
Find the Indexes for each Color

A=read_image(datadir+'blobs.png',rr,gg,bb)
tvlct,rr,gg,bb
window,1,xsize=256,ysize=256
tv,reverse(congrid(bindgen(16,16),256,256),2) ;Display the palette
help,A
;A BYTE = Array[400, 400]
window,2,xsize=400,ysize=400,title='Blobs Original'
tv,A
;Count the number of colors in the image
ha=histogram(A)
ka=where(ha GT 0)
print,ka ;These are the color indexes
;   10  33  186  215
print,ha[ka] ;These are the pixel counts for each color
;   4057  18313  18464  119166
;The largest count is at index 215 and corresponds to the background
;The color of the background is
print,[rr[215],gg[215],bb[215]]
; 255 255 255
Color Indexes, Cont.

; We can get the triples for the other colors as well.
print,[rr[10],gg[10],bb[10]]
;  51    0   204
print,[rr[33],gg[33],bb[33]]
;  255   0   153
print,[rr[186],gg[186],bb[186]]
;  51 255  0
; Clearly 10 is blue, 186 is green, and 33 is the reddish color
Find the Blue Objects

;Find the blue objects
B=A EQ 10 & LB=LabelRegion(B)
hlb=histogram(LB)
klb=Where(hlb GT 0)
print,klb
; 0 1
print,hlb[klb]
; 155943 4057
window,3,xsize=400,ysize=400
tv,LB*10
Find the Green Objects

;Find the green objects
G=A EQ 186 & LG=Label Region(G)
hg=Histogram(LG)
kg=Where(hg GT 0)
print,kg
; 0 1 2
print,hg[kg]
; 141536 7708 10756
window,4,xsize=400,ysize=400
tv,(LG GT 0)*186
Find the Red Objects

;Find the red objects
R=A EQ 33
LR=Label_Region(R)
hlr=Histogram(LR)
klr=Where(hlr GT 0)
print, klr
; 0 1 2
print, hlr[klr]
; 141687 8706 9607
Window, 5, xsize=400, ysize=400
TV, (LR GT 0) * 33
Show all the Objects

;Get a palette with good color contrast
TEK_COLOR
;Look at the palette
window,6,xsize=256,ysize=256
P=reverse(bindgen(16,16),2)
P=congrid(P,256,256)
tv,P

;Show the objects in different colors
Window,7,xsize=400,ysize=400
TV,LB+LG*2+LR*3
True Color Images

A true color image uses three color planes (RGB) to represent an image. This is done in IDL using a 3D array.
Finding Colored Objects

One could use brightness in the R, G and B color planes to search for colored objects.

A better approach is to use a color model such as HSV.

http://www.efg2.com/Lab/Graphics/Colors/HSV.htm
When the selection is based on the color it is easier to use hue, which is expressed as an angle that does not depend on brightness.
Selection Algorithm

1. Calculate the HSV triple for each RGB triple in the image.
2. Set thresholds for H, S and V for desired objects.
3. Select pixels that match the criteria.
4. Use the binary LABEL_REGION function on selection mask.
5. Collect the labeled objects that have sufficient area.
6. Display the selected objects.
Load, Display and Color Convert

A = read_tiff(datadir+'peppers.tif')
A = reverse(A,3)
window,0,xsize=512,ysize=512
device,decomposed=1
loadct,0
tv,A,true=1

; Convert the color space
rr = (A[0,*,*])[*]
gg = (A[1,*,*])[*]
bb = (A[2,*,*])[*]

color_convert,rr,gg,bb,hh,ss,vv,/RGB_HSV
Find the Red Pixels

\[ kr = \text{where}((hh \ LT \ 25 \ OR \ hh \ GT \ 345) \ AND \ ss \ GT \ 0.7 \ AND \ vv \ GT \ 0.45) \]

\[ \text{print,'Number of red pixels=',n_elements(kr)} \]

\[ MR = \text{bytarr}(512,512) \]
\[ MR[kr]=1 \]
\[ \text{window,1,xsize=512,ysize=512} \]
\[ B = \text{bytarr}(3,512,512) \]
\[ B[0,*,*]=MR*A[0,*,*] \]
\[ B[1,*,*]=MR*A[1,*,*] \]
\[ B[2,*,*]=MR*A[2,*,*] \]
\[ tv,B,true=1 \]
Label the Red Components

BR=Label Region(MR,/All)
hr=histogram(BR)

khr=where(hr ge 1000)
print,'Number of red objects=',n_elements(khr[1:*])
print,'Sizes of red objects= ',hr[khr[1:*]]

Window,4,xsize=512,ysize=512
RL=bytarr(512,512)
FOR k=1,n_elements(khr)-1 DO BEGIN
  k1=where(BR EQ khr[k])
  RL[k1]=k
END

tek_color
device,decomposed=0
tv,RL
Results on Red, Green and Yellow Objects