Practical Basics to Connected Filters

Michael Wilkinson

1 Introduction

In this practical we familiarize ourselves with the basic morphological filters, and move on to connected filters, and extensions to connectivity. In this introduction we introduce a number of useful MatLab commands. Before you start you need to copy some files to your home directory. The easiest way is to select "run command" from the start menu, and type

```
konsole
```

as command to run (note, NO typo). After this enter

```
cp ~csg113/asciA11-1.zip .
```

on the command line (**do not** forget the full stop at the end!!) After this, we type

```
unzip asciA11-1.zip
```

followed by

```
cd asciA11
```

To make everything work properly, type

```
source labsetup
```

You can now start MatLab with the command

```
matlab &
```

For non-Linux users: the `&` ensures you can run other commands from the same command console. reading images in MatLab is done using the command:

```
I = imread('m55.tif');
```

after which the variable `I` contains the image from file `m55.tif`. Obviously you can use different file names than `m55.tif` and different variable names than `I`.

To display the above image type

```
imshow(I);
```

If no figure is open, a new figure containing the image is created. If a figure is open, this is overwritten by the new image. To display an image without overwriting an existing figure, simply type

```
figure;
```

before the `imshow` command, if desired on the same line:

```
figure; imshow(I);
```

To store a processed image, type

```
imwrite(I,'new.tif');
```

in which `new.tif` can be replaced by any other name. MatLab automatically recognizes the required image file type by the extension. The basic morphological commands are

```
J = imdilate(I, se);
```

and

```
J = imerode(I, se);
```

with `I` the input image, and `se` the structuring element (S.E.). Various disk-shaped structuring elements can be created using:

```
se = strel('disk',r);
```

with r the desired radius. Other shapes are available as well. See the MatLab help facility for more details. Pixel-wise supremum or infimum of two images of the same shape is computed using

```
A = max(I, J);
```

and

```
A = min(I, J);
```

respectively.

Frequently we wish to check some logical predicate on an image. Assume we wish to test whether an image is equal to another at all points. To do this we might type:

```
A == B
```

which only works if `A` and `B` have the same shape. This returns not a single boolean(or logical in FORTRAN parlance), but a logical matrix of the same shape as `A` and `B`. For use in `if`-statements and the like, we need to reduce this to a single value. In the case above, the `all` function is needed. Supposing the images are 2-D, the correct boolean value is obtained by

```
all(all(A == B))
```

The two calls to `all` are needed because each call reduces the dimensionality of the result by one. The `all` function performs a logical *and* on all the boolean values in each row or column it processes. The similar `any` function performs a logical `or`.

In the following assignments you will be required to write MatLab functions (or m-files). A few examples are available in the `asciAll` work directory. Several text editors are available to write such files (and religious battles are fought over which is best (`emacs` of course)).

In all the assignments below, feel free to use any images related to your own research.

2 Assignments

Exercise 1: Openings-by-reconstruction Are four astronomical images available: `M55.tif`, `M81M82.tif`, `M81.tif`, and `hya.tif`. The first is a globular cluster, the second contains two galaxies, the third a single galaxy and the fourth a comet. In all cases stars also litter the field of view. One aim of this kind of processing is removal of the stars without distorting the objects. A further image `trui.tif` is a portrait. Read into variables with suitable names.

You can now test several techniques to process these image.

a.). Perform the basic morphological operations (erosion, dilation, opening, closing) on each image, using different structuring elements (vary shape and size).

b.). Using the basic operations available, implement the opening by reconstruction in your own MatLab function. Apply this to the same images. Which methods work best on the task of star-removal and why?

c.). Implement the closing by reconstruction using the notion of duality (i.e. the laziest way). Apply the opening and the closing by reconstruction to `trui`, using disk-shaped S.E. with different radii r. How does the image change as a function or r

Exercise 2: Area Opening

An area opening and closing are available in the set of files your received. To invoke the area opening, type, e.g.:

```
I = areaopen(m55,3);
```

for an area opening of image `m55` with an area threshold $\lambda = 3$. The dual closing is available as `areaclose`. Both methods use standard connectivity.

a.). Again try to remove stars without distorting the other objects, using the standard area opening and closing. Are there differences with opening by reconstruction?

b.). Try the effect of these filters (with large λ values) on `trui`. How does this compare to the case of reconstruction, and explain the differences.

Exercise 3: Changing the Connectivity

A mask-based version of the area opening is available using the command:

```
I = areaopenDI(m55,mask,3);
```

in which the variable `mask` is an image with the same dimensions as the first parameter, and contains a modified version of the input image, for clustering or partitioning connectivity. You can create these in any way, but a straightforward method is by using openings and closings by balls or disks, e.g. through the commands `closebyball` or `openbyball`.

a.). For the problem of star removal, try different types of masks, at different scale settings of the ball or disk radius. What are the main differences with standard connectivity?

b.). Try filtering `trui` using area openings at different λ-settings with different masks. Explain the differences.

Exercise 4: Attribute Filters

A more general attribute filter (though using standard connectivity) is provided through the command:

```
I = maxtreefilter(m81,12,1.5);
```

This particular setting preserves elongated (or rather non-compact) structures in the image in a scale-invariant way. The first parameter is the input image, the second indicates the attribute to be used, and the third is the attribute threshold λ. The full list of attributes is

0	Area A
1	Area of min. enclosing rectangle
2	Length of diagonal of min. enclosing rectangle
3	Cityblock perimeter P_1
4	Cityblock complexity (P_1/A)
5	Cityblock simplicity (A/P_1)
6	Cityblock compactness $(P_1^2/(4\pi A))$
7	Large perimeter P_2
8	Large compactness $(P_2^2/(4\pi A))$
9	Small perimeter
10	Small compactness $(P_3^2/(4\pi A))$
11	Moment of Inertia I
12	Elongation: I/A^2
13	Mean X position
14	Mean Y position
15	Jaggedness: $A * P_1^2/(8\pi^2 I)$
16	Entropy
17	Lambda-max (Max.child gray level - current gray
18	Gray level

As can be seen there are several approximations to perimeters available.

a.). What is the effect of the elongation filter on the stellar images?

b.). Try other attributes on a range of images, to test the different effects they produce.

Exercise 5: Make an "Alternating Sequential Filter"

An alternating sequential filter (ASF), computes a sequence of alternating openings and closings with increasing scale parameters, e.g.

```
I = trui;

I = areaopen(I,2);
I = areaclose(I,2);
I = areaopen(I,3);
I = areaclose(I,3);
.....
```

Note that the increment in area threshold λ need not be one.

a.). Create a MatLab function for an ASF with area openings and closings

b.). Take image `trui` and add noise using

```
I = imnoise(trui,'salt & pepper');
```

c.). Remove the noise using your ASF. Which are the best settings?

Exercise 6: Compute a Pattern Spectrum (optional)
Using a similar approach to the above, create a function to compute area pattern spectra. To compute the sum of grey levels, of image `I` use the statement

```
s = sum(sum(I));
```

Function `sum` is invoked twice for the same reasons as `all` described above. Note that the step size need not be one (need not be constant).

a.). Compute the area pattern spectra for the astronomical images. Plot them using

```
plot(S);
```

with `S` your spectrum. Explain the results.

b.). Try using clustering connectivity. Would some spectra be suitable to detect galaxies or clusters?