

Spot: cDNA Microarray Image Analysis Users Guide

Richard Beare and Michael Buckley

October 11, 2004

CSIRO Mathematical and Information Sciences
Locked Bag 17, North Ryde, NSW 2113
Michael.Buckley@csiro.au, Richard.Beare@csiro.au

Contents

1	Introduction	4
2	Quick Start	5
3	Tutorial	6
3.1	Creating and editing batch information	10
3.1.1	Creating an image list	10
3.1.2	Editing an image list	11
3.1.3	Creating a parameters file	11
3.1.4	Editing a parameters file	12

3.1.5	Creating a template	12
3.1.6	Editing a template	12
4	Detailed description of concepts	13
4.1	Sources of images	13
4.2	Image batches	14
4.3	Analysis steps	14
4.4	Imview	15
4.5	runSpots	16
4.6	Spot quantification	18
4.7	Seeded region growing vs globally optimal geodesic active contours.	20
4.8	Four channel microarrays	20
4.9	Result objects	21
4.10	exploreSpots	21
5	Technical details	25
5.1	Prefiltering	25
5.2	Background measurements	26
5.3	Quality measures	26
5.4	findSpots	27

5.5	Customization	27
5.5.1	SpotDefault options	27
5.5.2	Exporting data	30
5.6	Programming	30
5.6.1	Loading images and batch information	31
5.6.2	Image combination	31
5.6.3	Detecting distortion	31
5.6.4	Correcting distortion	31
5.6.5	Registration	32
5.6.6	Grid finding	32
5.6.7	Grid creation	32
5.6.8	Segmentation	33
5.6.9	Doughnut segmentation	33
5.6.10	Background measurement	33
5.6.11	Quantification	33
5.6.12	Image display, overlays, saving and interaction	34
5.7	Image operations	34
5.8	Image manipulation	35
A	Image batch file information	36

A.1 The Image Name File	36
A.2 The Parameter File	37
A.3 The Template File	38

1 Introduction

Spot is a package for analysis of cDNA microarray images. It produces measurements of spots and background from which further analysis can be done. It is usable only from the R statistical environment. This document provides an introduction to microarray image analysis and the **Spot** package.

Spot has two roles. The first is as a simple tool which anyone can use to carry out analysis of microarray images. A simple graphical user interface makes it easy for the novice user to begin analysis of microarray images, examine the results of the analysis and export data to other packages. The functionality of this tool can be customized by more experienced users. The second role is as a tool for researchers in the field of microarray analysis. The experienced R user can take advantage of the modular image analysis API to produce customized image analysis procedures or experiment with novel analysis ideas.

This document begins a brief description for impatient users. This is followed by a more detailed description of the workflow and important concepts needed by typical users in the form of a tutorial. A more detailed description of important features, such as `exploreSpots` and arrays with more than two channels, is given in Section 4. Information for advanced users and programmers is in Section 5.

This document supercedes the original online documentation associated with the first version of **Spot**.

2 Quick Start

This section assumes that `Spot` has been correctly installed. The installation procedure is described on the web site <http://spot.cmis.csiro.au/spot/spotinstall.php>. Once the installation is successfully carried out you should execute the following commands inside `R` to begin using `Spot`.

Start `R`, then execute the following commands from the `R` prompt.

```
> library(Spotbeta)
```

IMPORTANT NOTICE

```
1' COPYRIGHT Commonwealth Scientific and Industrial  
Research Organisation ('CSIRO'), Australia 2001.
```

```
All right in this Software are reserved to CSIRO.  
You are only permitted to have this Software in  
your possession and to make use of it if you have  
agreed to a Licence Agreement with CSIRO.
```

```
All enquiries for the use of this Software should  
be referred to:
```

```
Dr Michael Buckley  
CSIRO Mathematical and Information Sciences  
Locked Bag 17, North Ryde, NSW 1670, Australia  
email: Michael.Buckley@csiro.au  
phone: +61 2 9325 3209  
fax: +61 2 9325 3200
```

```
Spot Version 2.0
```

```
Attaching package 'Spotbeta':
```

```
The following object(s) are masked from package:base :
```

```
aperm as.array
```

A quick check to see that the installation is correct

```
Spot> imview.new()
```

If no imview window appears then there was probably a problem with installation. Check spot.cmis.csiro.au/spot/spotfaq.php for common problems and solutions.

```
Spot> runSpots()
```

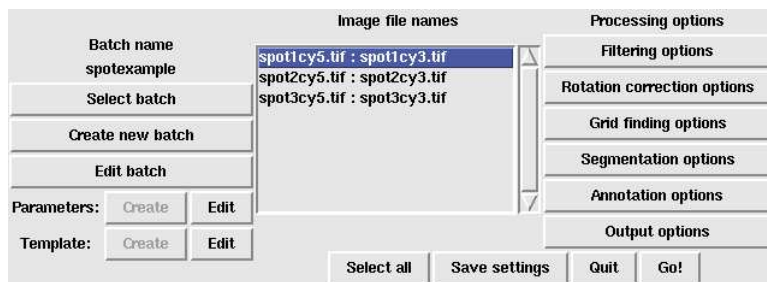
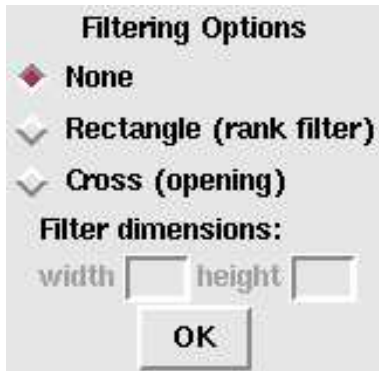


Figure 1: The runSpots window

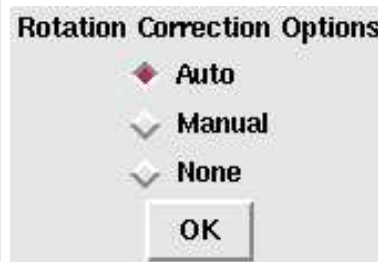
This command runs the graphical interface (see Figure 1) to the image analysis procedures.

3 Tutorial

A collection of small arrays is available from spot.cmis.csiro.au/spot/demodownload/SpotExamples.zip. There are three arrays in this set, and each array is stored in a pair of images. Sample batch, parameter and template files are also provided.



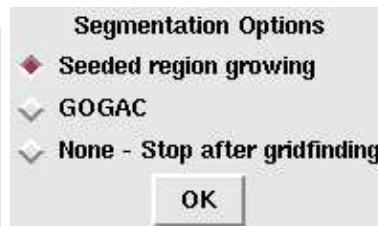
(a)



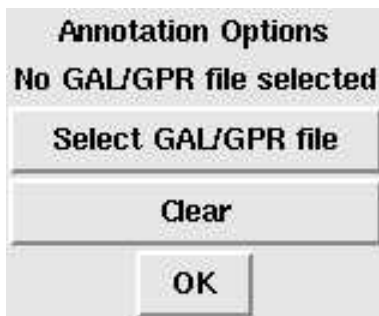
(b)



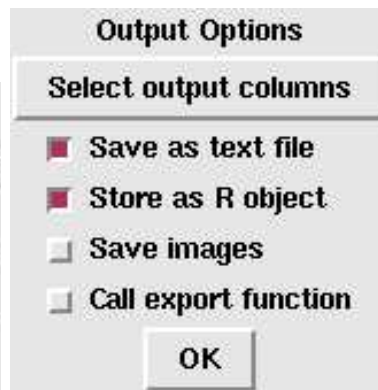
(c)



(d)



(e)



(f)

Figure 2: Processing options dialog boxes

- Download the SpotExamples archive by clicking on the link above and unzip/extract it to produce the SpotExamples directory. You can put this wherever you like – e.g. on your desktop or wherever you normally work.
- Start R. (Life is easier if you start R in the SpotExamples directory, but it isn't essential. In windows you can change the working directory using the “File->Change dir..” menu option.)
- Execute `library(Spot)` in R. You can put this in your `.Rprofile` file if you want it to start automatically.
- Execute `runSpots()` in R. This starts the `runSpots` window.
- Begin using the predefined template information. Press the **Select Batch ...** button and select *images.spotexample*.
- Notice that the list of images making up the batch appear in the central list-box, “Image file names”, with the first image highlighted. Also note that the **Edit** button for both **Parameters** and **Template** is active. This means that the template and parameter files exist. If they did not exist the **Create** buttons would be active and **Go!** would not be.
- We will begin with a run that only does grid finding. This will give an idea of the sort of problems we may need to deal with. Go to the **Segmentation Options** dialog) and check the **None - Stop after gridfinding** (see Figure 2 option).
- Click the **Go!** button. The grid finding process is started and the main application window is disabled. You will be prompted for the location of the top left corner of the grid. Selecting the top left grid point is the way in which a previously defined template is applied to the current image. The software will attempt to find the best position for the template within a small range of the point you select. The process of creating a template is discussed later.

It is important to be able to enter points in imview. When a point is entered a circle appears on the image. There are a number of ways of doing this. The simplest is to click the left mouse button while pressing the shift key. The second is to enter the “Add point mode” which means that points will be added whenever the left button is clicked. You can enter “Add point mode” from the toolbar (Transform->Toolbar menu

or shift-t) or from the Transform->pointfile->Add point mode menu option (ctrl-g).

You can remove the most recent point selection by pressing alt-d.

More details about imview are available in Section 4.4.

Press **OK** when you have selected the position.

- It is often useful to use gamma correction in imview (transform->histogram->contrast-brightness menu) to make the spots clearer.
- Once the grid overlay appears examine it closely to confirm that the grid lines pass through the spots.
- Try the other two image pairs in the batch by selecting them individually and pressing **Go!** (Note that you deselect an image by clicking on it). You will notice that the third image pair has a significant level of rotation that is automatically corrected. The rotation correction and grid finding options are all controlled via the **Rotation correction** and **Grid finding options** respectively.
- Now select a different option in the **Segmentation Options** dialog – begin using **Seeded region growing**. Now go to the **Output Options** dialog and check **Save as text file**, **Store as R object** and **Save images** and rerun beginning with the first array (select it in the image list). You will need to reindicate the grid origin positions. Quit **runSpots** (by clicking the **Quit** button) after this process is complete.
- The SpotExamples directory should contain some new files. The files ending in a **.spot** suffix are text files containing the measurements of the arrays. The **Save as text file** option controls the creation of these files. There are also some image files (with a **.jpg** suffix) displaying the grid finding and segmentation results.
- If you examine the image files you will notice that the third array was rotated. This rotation was corrected automatically. The correction is controlled by the **Auto** option in the **Rotation Correction Options** dialog.
- Execute **ls()** in R. You will see an object **spotexamples**. This a list of length 3, and each list element is an matrix of measures. The **Store as R object** option controls the creation of this object.

- Repeat the process using **GOGAC** option in the **Segmentation Options** dialog. This uses a different segmentation process that might produce more reliable results on some images.
- **Annotation options** are also likely to be of interest — if you have a GAL file (or .GPR file) associated with the images then it can be used to provide gene IDs and descriptions.

3.1 Creating and editing batch information

In the preceding tutorial three pieces of information in the form of data files were provided in the `SpotExamples` directory. These provided

- The image list which defines the image files making up the “batch”. (The `images.spotexample` file.)
- The structural parameters: block layout (e.g. 4×4), spot layout (e.g. 18×20) and some tolerance information. (The `parameters.spotexample` file.)
- “Template” information primarily describing print-tip deformation. (The `template.spotexample` file.)

When analysing your own images you need to create these data files and sometimes you will need to edit them. `Spot` provides tools for doing this and this section shows you how to use them to create and edit each of these three data files.

Now let’s create the batch information for `SpotExamples` from scratch.

3.1.1 Creating an image list

- Select **Create new batch**.
- Select the folder which the images are stored in and enter a new name for the batch e.g. “myExamp”. (do not include the “images” prefix)

- A **Spot:question** window will be launched. You will then be asked whether the batch contains separate or combined **tiff** images. Some files contain multiple channels — we call these combined **tiff** images. The example contains one channel per image file, so select **separate**.
- You will then be prompted for the channel names. This choice is arbitrary, and we'll select **Cy3/Cy5**.
- The image batch window will now appear. Select **Add**. You will be asked to select a Cy3 image. Select "spot1cy3.tif". You will then be prompted to add a Cy5 image. Select "spot1cy5.tif".
- Select **OK**. You have now created a batch containing 1 array. You can add the others later if you want to.
- The newly created batch is selected automatically.

3.1.2 Editing an image list

The **Edit batch** button will be highlighted once you have selected a batch. Pressing it allows you to edit the selected batch using the same dialog that was used for creation.

3.1.3 Creating a parameters file

- Select **Parameters: Create**. This button allows you to create a parameter file for the *currently selected batch*.
- The parameters file contains information about grid and block geometry as well as grid search tolerance. The first two define the number of spots in each block and the second two define the number of blocks per slide. The example slides have 18 spot rows and 24 columns per block. There is 1 row of blocks and 4 columns per slide.

Don't change the tolerance unless spots are very close to the edge of the image (set the tolerance to something small if that is the case). Tolerance settings are not used if either of the **Mark grid corner** options in the **Grid finding** dialog are selected.

3.1.4 Editing a parameters file

The **Parameters: Edit** button will be active if a parameters file for the currently selected batch exists. You can use it to change values in the parameters file using the same dialog that was used for creation.

3.1.5 Creating a template

- The **Template: Create** button will start the process of creating a template for the currently selected batch.
- The template captures information about the position of blocks relative to one another and the block size. This information is generated by interacting with the image in imview. The idea is to use the imview point selection mode to mark the top left corner of each block of one array. This provides information about offset between blocks that may be caused by bent printing pins. You will need to guess when the spot in question is missing. You will also need to mark the bottom right of the last block (this indicates the block size). You can enter the point-file mode in imview by selecting the Transform->toolbar menu option. Then select the “Add point mode” button from the toolbar. When you click on the imview image a circle should appear.

Other useful tips include zooming on an area of interest to accurately locate the spot and using the Transform->histogram->contrast-brightness option to carry out gamma correction and make spots visible.

This process can be tedious but is important to carry out accurately. If you make a mistake near the end you can try editing the location values directly in the “Edit template” window.

3.1.6 Editing a template

The **Template: Edit** button will allow you to edit a template if it already exists. There are two ways of doing this. One will delete the existing template and recreate it using the process just described. The second will allow

you to edit the coordinate values directly using the R data editor. Choose between these options using **Delete** or **Edit** buttons when prompted.

The **Edit** option uses the R data editor. Close the editor window when you have changed the values you are interested in and the result will be saved.

The batch has now been set up.

4 Detailed description of concepts

4.1 Sources of images

cDNA chips are printed using robotic printers and each chip contains a number of grid blocks. These grid blocks are arranged in rows and columns. Each grid block contains a number of spots. A number of chips are usually printed in a run and have the same arrangement of grid blocks and spots. We shall see later that information about the relative position of grid blocks is important for the analysis of these images. The relative positions of blocks is dependent on the relative positions of pins in the printer used to create the chip.

A chip is processed and scanned by an instrument such as the Axon GenePix microarray scanner to produce an image file or files. The scanner will usually produce two images, each of which corresponds to a different fluorescent wavelength (new scanners are able to produce four channel images — see Section 4.8). These two images are often stored in the same image file (e.g. Axon scanners typically store both channels in the same `tiff` file, and there may also be some lower resolution preview images). It is also possible that a scanner may produce separate image files for each fluorescent channel. All functions in `Spot` assume that these images are registered and have the same dimensions. If this is not the case then some tools are available to correct the problem and they will be discussed in Section 5.8.

4.2 Image batches

`Spot` represents one or more cDNA chip images as a *batch*. The cDNA chips in a batch must come from the same printer at similar times because the geometry information that is essential to the analysis is printer dependent. A batch may also represent a single experiment or part of an experiment.

A batch consists of 3 pieces of information

1. Image file names.
2. Parameter information.
3. Template information

If the batch is called “experimentA” then this information is stored in files called `images.experimentA`, `parameters.experimentA` and `template.experimentA`.

Section 3 gave an example of creating and editing batch information. Full details about the file formats are in Appendix A.

4.3 Analysis steps

The image analysis of cDNA images requires a number of steps. The goal of the process is to provide a number of measures for each spot in the images. These measures may vary, but are likely to include spot intensity in each fluorescent band and background measures in each band. In order to produce these measures it is necessary to segment each spot — i.e define which pixels belong to a given spot. In order to do this `Spot` carries out the following steps, some of which are optional (and can be switched on or off as required).

1. Load images.
2. Combine images. The two fluorescent channels are combined so that segmentation can be carried out. If segmentation was carried out independently on the two channels the measurements would be biased.

3. Prefiltering. Image noise may upset the segmentation procedure. If prefiltering routines are defined then they are applied to the combined image. (Advanced procedure)
4. Detect grid distortion (optional). An analysis is carried out (using some grid block information) to determine the distortion of the grids of spots. This distortion may be rotation due to slight misplacement during scanning or printing or shear due to problems with the robotic printer. If the distortion is above a certain threshold the combined image is corrected. It is also possible to manually correct distortion from the GUI.
5. Grid finding. The grids are located using information from the parameter and template file. This search depends on the grids being aligned with the image borders. The starting point for the search may be specified manually from the GUI.
6. Segmentation. A choice of two segmentation methods are available — seeded region growing (SRG) and globally optimal geodesic active contour (GOGAC). The relative merits of the two are discussed in [Section 4.7](#)
7. Invert distortion correction. The inverse of the distortion correction that was applied to the combined image is applied to the segmentation mask.
8. Measurements. A variety of measurement of the original images are taken using the segmentation masks.

4.4 Imview

Image display is carried out using `imview`. `Imview` is also capable of limited user interaction and `Spot` makes use of this to mark points of interest. This interaction is via the pointfile mechanism. There are a number of ways of doing this. The simplest is to click the left mouse button while pressing the shift key. The second is to enter the “Add point mode” which means that points will be added whenever the left button is clicked. You can enter “Add point mode” from the toolbar (Transform->Toolbar menu or shift-t) or from the Transform->pointfile->Add point mode menu option (ctrl-g).

Points may be grouped by clicking the right mouse button. “Alt-d” deletes the last point or grouping action.

Inview can also apply useful transformations (try the “Transform” menu) to the image, such as gamma correction and image colour maps which can aid in viewing dark spots.

Zooming is also available. This is essential when very large array images are being processed. The simplest zoom mechanism uses the middle button of a 3 button mouse to highlight the area of interest. Zooming is also available using the menu or toolbar.

Another useful feature is the ability to adjust the transparency of an overlay. This can make examining spots using `exploreSpots` easier. Transparency is adjusted from the “Edit->User Preferences” menu.

4.5 `runSpots`

The graphical interface to analysis functionality of Spot is executed using the following command in R

```
Spot> runSpots()
```

This will start the application shown in [Figure 1](#).

The buttons on the left hand side of the window are for creating and editing the control files associated with a batch of images. The list box in the center shows the names of images in a batch and allows the user to select which arrays should be processed.

The right hand side of the window controls the way in which arrays are processed.

The detailed descriptions are

- **Select batch.** The batch we are interested in processing is selected using a file selection dialog. Once a batch is selected the image list will appear in the central list box.
- **Create** and **Edit** buttons. If the parameters and templates have not been set then the **Create** buttons will be active. If they have been set then the **Edit** buttons will be active. **Go!** will only become active once the batch, templates and parameters all exist.
- **Image filenames** is a list of the image files referenced by a batch. Items selected from this list are processed when **Go!** is pressed.
- **Filtering options** allows a user to apply filters to a combined image. The filters may be able to remove some artifacts that upset the segmentation. If the filter kernel sizes are set to be larger than the expected spot size then many spots will disappear.
- **Rotation correction options** controls the way in which grid distortion is corrected. The manual rotation correction requires a user to enter 2 pairs of points that indicate the orientation of the grid — it may be useful if the automated procedure fails.
- **Grid finding options** controls the way in which grid location is determined. The **Mark grid corner (small search)** option allows a small search (10 pixels) around the location indicated by the user. This seems quite reliable. The **Mark grid corner (no search)** places the grid exactly as indicated by the user while the **Large search** uses the tolerance parameters in the parameters file and the location of the template. This is only useful if the arrays are in very similar positions in the image.
- **Segmentation options** allows the user to select between different types of spot segmentation (see Section 4.7 for details). The **None - stop after grid finding** option causes the process to exit before segmentation begins. This allows the operator to examine the placement of the grid and confirm whether the results look sensible – i.e whether grid lines are aligned with spot rows and columns. The grid finding process is relatively quick so it is convenient to examine the results of this step rather than wait for the segmentation results.
- **Annotation options.** Allows a GAL file or GPR file to be selected. Gene IDs and descriptions are extracted from this file and included in the output object.

- **Output options.** The measures reported by the analysis and the form of the report are set here. Section 4.6 describes the different statistics that are available. The check boxes allow the data to be stored as text files, as objects in the R global environment, passed to export functions and arrange for the saving of images for verification purposes. The **Save as text file** option produces text files named “batchname_XX.spot”, where “batchname” is the name of the batch and “XX” is the number of the array in the batch.

Details about the R objects are in Section 4.9 and the export function is described in Section 5.5.

- **Select all** selects all images in a batch.
- **Save settings.** The checkbox settings are saved in a file named `.runSpotDefaults` which is used next time the GUI is run.
- **Quit.** Quit the GUI and return to the R console.
- **Go!.** Begin processing arrays.

4.6 Spot quantification

The measures reported by the analysis procedure are specified by the control shown in Figure 3. You should take care modifying these — it is possible that subsequent processing steps may depend on your choice here. The meanings are as follows

- **Location Info.** Spot index, grid row, grid column, spot row and spot column.
- **Shape Info.** Spot perimeter, circularity, area, image x position, image y position. Circularity is $(4 \times \pi \times \text{area})/(\text{perimeter}^2)$
- **Derived Info.** The log ratios of the red and green channels. The inputs are the red and green median values and the “morph” background estimates.
- **Quality Measures.** If SRG segmentation is used then the circularity is used as the quality measure. If GOGAC segmentation is used then the integrated edge strength is used as the quality measure.



Figure 3: Set output columns

- **Morph Background.** Morphological background estimates for each channel. There are two sets of data - morph and morph.close.open. The former is computed using a morphological opening and tends to under estimate the background. The latter closes small holes first and is a better estimate.
- **Median.** Median of foreground for each channel.
- **Mean.** Mean of foreground for each channel.
- **IQR.** Interquartile range of foreground for each channel.
- **SD.** Standard deviation of foreground for each channel.
- **Background Median.** Median of background values for each channel.
- **Background Mean.** Mean of background values for each channel.
- **Background IQR.** Interquartile range of background for each channel.
- **Background SD.** Standard deviation of background for each channel.

- **Regression Quality.** A symmetric pixel by pixel regression essentially the same as the “regression ratio” measure in GenePix Pro. The `Rsq` component can be used as an indicator of quality and the `mhat` component is a substitute for the log ratio.

4.7 Seeded region growing vs globally optimal geodesic active contours.

There are two segmentation methods provided with `Spot`. The first, seeded region growing (SRG) [1], was used in the first version of `Spot`. The second, globally optimal geodesic active contours (GOGAC)[2], is new. These two methods have quite different properties. If the images are of high contrast the results are likely to be very similar. The differences are summarised below:

- Edge strength vs region brightness. GOGAC creates a closed contour of optimal integrated edge strength. SRG adds pixels to regions in a priority driven fashion that builds regions of consistent brightness.
- GOGAC produces a quality measure (integrated edge strength). SRG doesn't.
- SRG performs simultaneous segmentation, GOGAC performs serial segmentation. This means that SRG has an inbuilt mechanism preventing regions overlapping. GOGAC works on each spot independently and any overlap needs to be corrected after the fact. This can be an issue when spots are close together.
- GOGAC is biased to producing circles while SRG is unconstrained. The more desirable approach will depend on your images.
- SRG is a lot faster.

4.8 Four channel microarrays

Some scanners are able to produce images with more than two channels of data. `Spot` is able to deal with images of this sort, but only if they are

stored as a combined image (preview images can also be included in the combined image). If an image is provided with three or more channels then the processing is basically the same, but there are some differences with the output.

Segmentation is still carried out on a combined image that is constructed using all of the channels. The response columns are named using “Ch1”, “Ch2”, “Ch3” etc instead of “R” and “G”. Logratio measurements are not reported (since it is unclear which columns should be used) and the regression based quality measure is computed using the first two channels.

4.9 Result objects

Analysis of an array produces a large matrix (or data frame if gene ID’s are included) , with one row for each spot. There are a variety of measurements that may be produced for each row, such as spot intensity in each channel, area, background intensity as well as some quality measures. The user has some flexibility about selection of measures reported by `Spot` (See Section 4.6).

`runSpots` will produce a list of arrays from a batch. This object will be placed in the global environment with a name corresponding to the batch name if the **Store as R object** option is selected.

Each element of this list is a matrix or dataframe containing the information from a single array.

If **Save as text file** is selected then a set of text files for each batch is produced, as described earlier.

4.10 exploreSpots

`exploreSpots` is a tool that allows spot segmentation results to be displayed in an interactive fashion. The **Store as R object** option must have been set for the tool to operate. The command:

```
Spot> exploreSpots(spotexample)
```

is to allow the user to examine the spot data, to view the segmentation of a subset of spots and, if required, return the subset information to R for use in further analysis. (If this option is required use a command like `SelectedSpots <- exploreSpots(spotexample)`. `SelectedSpots` will be assigned the most recently displayed subset information).

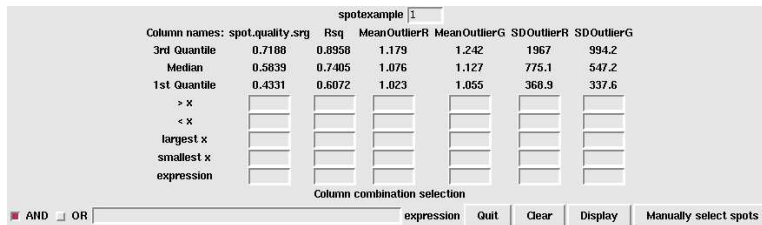


Figure 4: The `exploreSpots` window

Subset selection can be done in two ways: either by:

- entering criteria in the spreadsheet fields as described below and then pressing **Display**.
- Pressing **Manually select spots** and marking the spots of interest in the imview window.

When `exploreSpots` is first started the spot mask is not displayed. It is often useful to select the array number you are interested in and press the **Display** button. This will update the summary statistics and display the mask.

The spot segmentations are displayed as a solid pink mask over the original image. The transparency slider in the “Edit->User preferences” dialog of imview can be used to adjust the transparency of the mask so the underlying spots are visible.

The argument `spotexample` is either the list produced by `runSpots`, or a single element of that list.

Note that `exploreSpots` uses a special attribute of the output of `runSpots` that can **not** be saved and reloaded.

`exploreSpots` is set up like a spread sheet. The columns correspond to the various measures reported by `runSpots` plus some quality related measures. The rows immediately below the titles are column summaries that are computed the first time **Display** is pressed. These are global spot statistics, not statistics of the subset.

The quality measures can be interpreted as follows:

- `spot.quality.srg` or `spot.quality.gogac`: In the latter case the quality measure is an integrated edge strength. Low values indicate low edge strengths and probably a low intensity or missing spot. The value should be $1/2\pi$ for a completely featureless area. In the former case the circularity is used as a quality indicator. Low circularity values indicate spots that are far from circular.
- `area`: In many arrays the spots should be of similar size, so very large or very small spots may be an indication of problems.
- `Rsqr`: Is the R^2 from a symmetric pixel by pixel regression, essentially the same as the “regression ratio” in GenePix Pro. It should be high for clear spots. However it can also be high in the case of outlier pixels in otherwise faint spots.
- Outlier columns compare means and medians or standard deviations and inter-quartile ranges. If these values are very different then there are likely to be some outlier pixels caused by either dust, missing pixels or doughnuts.

Note that the quality measures are dependent on the outputs selected in `runSpots`. If you modify the columns reported then not all measures will be available. The quality function is currently only set up for two channel microarrays. A limited subset will be available for arrays with more than two channels — i.e. the outlier columns will not be available. We expect to customize the quality indicators for multichannel arrays as more experience with them is available.

The spreadsheet part of the application allows the quality indicators to be queried and the spots satisfying the queries are displayed. The queries are constructed by entering values or expressions in the cells. The results of the cell computations for each column are ORed together. The results of each column are combined in a user selectable manner — AND, OR or an expression.

The row meanings are:

- $> x$. A threshold value. Selects the column values that are greater than the cell value.
- $< x$. A threshold value. Selects the column values that are less than the cell value.
- **largest** x . Select the largest x values, where x is the cell value.
- **smallest** x . Select the smallest x values, where x is the cell value.
- **expression**. This is the most powerful way of defining a selection. The idea is to enter an R expression in the cell. This expression should operate on a variable X (note that this is a capital). This expression could be a call to a function defined elsewhere.

An example might be a function that returns the highest 1% of spots for a given measure.

```
X > quantile(X, 0.99)
```

An especially useful example could be

```
X == 'GeneName'
```

in the ID column to highlight a particular gene of interest.

This function can be entered directly in a cell in the “expression” row of the table.

The column combination options are similar. The AND and OR options do the obvious thing - simply combining the column selection using logic operations. There is also an expression option that takes precedence over the other alternatives (i.e if an expression is defined then the AND or OR flags are ignored).

The expression operates on components of a list named `X`. The component names match the column names. The following expression lets us select the largest 10 red and green values (selected using cell settings described earlier) with a `spot.quality` measure greater than 0.9.

```
(X$Rmedian | X$Gmedian) & X$spot.quality
```

`exploreSpots` invisibly returns the last subset vector calculated.

The default behaviour of `exploreSpots` is to generate quality measures using the function `computeQualityMeasures`. If you want to explore the raw spot data then use `qualfun=NULL` as an argument to `exploreSpots`. Advanced users can write their own versions of `computeQualityMeasures` and pass it as an argument to `exploreSpots`.

5 Technical details

5.1 Prefiltering

It is possible that image noise or clutter can interfere with the spot segmentation process. This most commonly occurs if a small area (several) of very bright pixels appears in an otherwise uniform, but much fainter, spot. In some cases only the very bright region may be segmented. Prefiltering may help reduce the chances of this happening by allowing removal of localized “features”. Spot has two types of prefilter available — region rank based filters and open by union filters. The first replaces each pixel by the median of a surrounding kernel. The second replaces each pixel by the maximum of vertical and horizontally oriented line openings. The first is a form of smoothing that behaves the same with light and dark features while the second will only modify bright features.

Both of the filters can have large kernel sizes without a speed penalty, but if large kernel sizes are selected then spots will disappear.

5.2 Background measurements

`Spot` provides two types of background estimates. One is based on statistics of a region around the spot while the other is based on the response of various morphological filters [3].

The region used by the first method is computed by a thickening that places a doughnut around each spot. This doughnut does not overlap other spots or other doughnuts. It is possible to compute means, medians, standard deviations and interquartile ranges of pixels in the doughnut. These measures are returned in columns with names prefixed with `bg..` Advanced users can access the pixels in each region and compute other statistics.

There are two types of morphological filters available. The first corresponds to an *opening*. Background estimates based on the opening are lower and less variable than other estimates. These estimates are returned in columns labelled `morph`. The second corresponds to a combination of a *closing* followed by an *opening*. The filter removes small dark regions and is therefore less of an underestimate than the opening. However it can be dangerous if the spots are close together and the closing is too large. These estimates are returned in columns labelled `morphCloseOpen`.

Section 5.5 describes how to control the sizes of the openings and closings using the `options` mechanism.

5.3 Quality measures

There is a large and active body of research investigating quality of microarray data — and it is still an open problem. `Spot` provides some facilities that may assist by indicating the quality of segmentation.

GOGAC, as mentioned in Section 4.7, provides a measure of normalised integrated edge strength that can be a good indicator of very faint spots.

Columns `Rsq` and `mhat` are the results of a regression based measure that is very similar to the “regression ratio” measure in GenePix Pro. The `Rsq` can be interpreted as a quality indicator, however it should not be considered in

isolation from other information. The value of `Rsq` will be high for distinct spots and low for indistinct ones. Unfortunately it can also be high if there are outlier pixels.

Some other quality indicators are provided via `exploreSpots`.

The tools used to construct `Spot` can be used by advanced users to develop their own quality measures.

5.4 `findSpots`

`findSpots` is the command line interface to the analysis functionality of `Spot`. It is the function that is called by `runSpots`. `findSpots` accepts a large number of arguments and is very flexible. This flexibility can make it difficult to use. Direct use is only recommended for advanced users, who should check the online help via `?findSpots`.

5.5 Customization

5.5.1 `SpotDefault` options

A certain level of customization is available via the `options` mechanism in R.

```
Spot> options($SpotDefaults)
```

```
$fixgrid  
[1] FALSE
```

```
$crossgap  
[1] 0
```

```
$fgseedsize  
[1] 3
```

```
$templatebyid
[1] FALSE

$grid.rotation
[1] TRUE

$grid.rotation.thresh
function (para, template)
{
  xtotal <- para$ngrid.c * template$gridsize.c
  ytotal <- para$ngrid.r * template$gridsize.r
  total <- max(xtotal, ytotal)
  spotsep <- max(para$spotsep.r, para$spotsep.c)
  dd <- spotsep * 0.25
  angle <- atan2(dd, total) * 180/pi
  angle
}

$rotation.only
[1] FALSE

$grid.rotation.manual
[1] FALSE

$visualize
[1] TRUE

$saveImages
[1] FALSE

$gogac.margin
[1] 1

$SE.scale
[1] 2.5

$SE.close
[1] 3
```

```
$SmallSearchR  
[1] 10
```

```
$SmallSearchC  
[1] 10
```

```
$prefilter  
NULL
```

```
$ExportFunction  
NULL
```

```
$CombineChannels  
NULL
```

Many of these options are also set via the `runSpots` GUI, and the GUI will take precedence. Some of the less obvious parameters, such as structuring element sizes for the morphological background estimation can only be modified this way.

Options may be changed as follows.

```
Spot> myExportFunction <- function(X) {  
+   print(dim(X))  
+ }  
Spot> xx <- options()$SpotDefaults  
Spot> xx$ExportFunction <- myExportFunction  
Spot> options(SpotDefaults = xx)
```

It is only useful to set some of the other parameters if `findSpots` is being called directly from the command line.

Most of the things that can be done by modifying the options can also be achieved using the `runSpots` GUI and saving the settings.

One thing that cannot be controlled by the GUI is the function that determines whether the rotation should be corrected. If the rotation correction option is checked then an estimate of rotation is made. The estimated value

is passed to the function `SpotDefaults$grid.rotation.thresh` which determines whether correction should be made. At the moment this function attempts to determine whether the detected rotation is sufficient to move a quarter of the spot separation. In some circumstances other tests may be more appropriate so this function can be replaced in the same way as the `export` function.

5.5.2 Exporting data

The **Call export function** option in `runSpots` is a hook that allows a user defined function to be called by the GUI. This function (which is an option that can be modified as described above) can be used to implement alternative methods of saving the data, such as conversion to other R objects, entry into databases some normalization etc.

Some example functions are available at <http://spot.cmis.csiro.au/spot/contrib/>.

5.6 Programming

The functionality in `Spot` can be used to develop customized microarray image analysis procedures. This will require experience with the R language and the ability to do some investigation of the code. The documentation for most of the low level functions has not been included because it is not in the correct R format. It can be found at <http://spot.cmis.csiro.au/spot/manuals/>. Most of these capabilities discussed here are used by `findSpots`. However `findSpots` is quite complex and difficult to understand because it supports a large number of alternatives — much of the code complexity supports the flexibility.

This section will provide an overview of the higher level functional blocks. We hope this will provide a general understanding of the tools available and how they can fit together. More detailed questions can be referred to the mailing list.

5.6.1 Loading images and batch information

The high level image loading functions are `ReadImages` and `LoadImage`. The first reads the image file names from an *images.batch* file while the second actually loads the images. It contains logic to deal with `tiff` files containing multiple images. The lowest level function available is `imloadtiff`.

The batch parameter and template information is read by `ReadParameters` `ReadTemplate`.

5.6.2 Image combination

All processing steps relating to distortion estimation, grid finding and segmentation are carried out on a combined image. This image is generated using `CombineIms`. This function can also be controlled using `options`. `SpotDefaults$CombineChannels` can be set to a vector of channel indices which will be combined. This could be useful in situations where one channel is a good marker.

5.6.3 Detecting distortion

Grid distortion estimation is estimated using `estimateDistortion`. This function returns an estimate of X and Y shear as well as row and column spacing.

5.6.4 Correcting distortion

The grid distortion is corrected using `fixShear`, which uses the `Shear` function. The correction is applied to the combined image and the inverse transformation is applied to the segmentation mask.

A more general warping function `imwarpaffine` is also available.

5.6.5 Registration

Registration is not used in the existing analysis procedure. However some users have suggested that it might be useful in order to combine multiple scans of the same slide. A function `imcorrelate` has been provided that operates on a pair of images. This function performs a normalised cross correlation of the two input images. A common way of registering similar images is to estimate the shift required to maximize the correlation.

The correlation is carried out using fast fourier transforms which are limited in the choice of image dimensions. The function `imfftefflen` is also available to estimate the most appropriate image dimensions. If you do not resize the image (see Section 5.7) then you are likely to see errors like “insufficient array storage”. Use `imfftefflen` as follows:

```
Spot> imfftefflen(1100, over = FALSE, maxprime = 5)
```

```
[1] 1080
```

`imcorrelate` can return a full correlation image or a subset around the maximum value. In the second case some position information is also returned in the second component.

5.6.6 Grid finding

Grid finding is a crucial step in the segmentation procedure. Successful grid finding relies on the grids being oriented parallel to the image sides (hence the need for distortion correction). The high level function is `mk.grids` which returns lists of row and column locations.

5.6.7 Grid creation

This grid information is turned into seeds in two different ways — `GridSeeds` is used to create a seed image for seeded region growing segmentation while

in the case of GOGAC segmentation the information is used directly.

5.6.8 Segmentation

The segmentation is carried out by either `seg.srg` or `seg.gogac`. These functions produce a special mask object. This object is essentially a run length encoded mask image with some shape statistics attached. This mask is used to make measurements later on.

5.6.9 Doughnut segmentation

Another experimental function that may be of interest is `seg.doughnuts`. This function takes a segmentation mask and the combined image and breaks each spot segmentation into two. If the spot has a dark centre then this will be separated. This may be an interesting tool for some arrays.

5.6.10 Background measurement

There are two classes of background measurements in `Spot`. The first is the morphological measurements — these are computed using `bgVals.morph`. The low level functionality is provided by `imrecterode`, `imrectdilate` and `SampleGrid`. The other class employs a mask generated by *thickening* the spot segmentation (thickening is like a dilation). Pixels statistics can be generated from the mask. The thickening function is `spot.thicken` and the measurement is described next.

5.6.11 Quantification

Quantification using segmentation masks requires two steps. The first is carried out by the mask generation procedure and involves the computation of mask shape measures. These are available as an attribute to the segmentation mask — see `ShapeParams` for an example of how to deal with this type of data. The second step is to collect the pixels under each mask element

so that various statistics can be computed. The function `imfetchpixels` takes an image and a mask and returns a list of vectors. There is one vector for each mask element and each vector contains all of the pixel values under the mask. Conventional R functions can be used to compute statistics of the different regions. Note that direct application of the high level statistics function like `median` to these objects can be quite slow due to their generality — customization of these routines may prove much faster (see `spot.med` for an example). The segmentation mask is returned as part of the `SpotInfo` attribute attached to the result matrix.

This is a substantial change from the previous version of Spot where all statistics were computed in the C code. Now masks can be used to extract pixel values in R objects.

5.6.12 Image display, overlays, saving and interaction

Image display capabilities are provided using `imview`. `imview` runs as a separate process and communications are via a socket and/or shared memory. This can be a little unreliable at times. `imview` can display images and overlays — the commands are `imview` and `imview.overlay`. `overlay.from.label` shows how to create an overlay image from a mask.

A single color image can also be created by using `imoverlay` to combine a greyscale and a mask with a colormap. The result can be saved using `imsavetiff` or `imsavejpeg`.

`imview` is also capable of limited user interaction (such as that used to create the template information). The main function used for this sort of interaction is `imview.getpointfile`.

5.7 Image operations

Images in `Spot` are not arrays or matrices — they are special C objects. However facilities exist so that they can be operated on as arrays. Standard array subsetting notation should work, as should most arithmetic, logical and mathematical operations.

Arrays can be converted to images using `as.image` and images to arrays using `as.array`.

5.8 Image manipulation

`Spot` is not intended to be an image manipulation package, but a number of tools are available that may help in some circumstances. `Spot` assumes that each channel of an array is the same size (i.e. the same image dimensions) and that the spots are registered. If a scanner produces channels independently then these requirements may not be satisfied. The function `Shift` is available to shift and crop batches of images. This function is not usually required because most scanners now produce images in the correct form. However it should be useful as a template for developing customized manipulation functions.

The warping functions `Shear` and `imwarpaffine` are also available for more complex manipulations.

References

- [1] R. Adams and L. Bischof. Seeded region growing. *IEEE transactions on pattern analysis and machine intelligence*, 16:641–647, 1994. [20](#)
- [2] B. Appleton and H. Talbot. Globally optimal geodesic active contours. submitted to *JMIV* September 2002, revised October 2003. [20](#)
- [3] Y. H. Yang, M. J. Buckley, and T. P. Speed. Analysis of cdna mcicroarray images. *Briefings in Bioinformatics*, 2(4), 2001. [26](#)

Appendix

A Image batch file information

This section describes in more detail the content of the files containing the batch information. These files can now be created using graphical tools so there is little need to edit them by hand. This section is purely for reference purposes. R facilities, such as `read.table`, `write.table`, `dget` and `dput` are used to read and write the files, so it is important to get the syntax correct if you choose to edit them by hand.

A.1 The Image Name File

To begin with, each batch of microarray data needs a name. We will use the name “array1”.

The file `images.array1` specifies the image pairs comprising the batch “array1”.

This looks like

```
R G
array1.R1.tif array1.G1.tif
array1.R2.tif array1.G2.tif
array1.R3.tif array1.G3.tif
etc.
```

Each line gives the names of a pair of images. These image files, as well as the file “images.array1” itself, need to be in the directory where `Spot` is run. Note that under Windows it is possible to change the working directory while running `Spot` using the “Change dir” entry in the “File” menu. Currently only TIFF image format is supported.

The first row in the image name file is a column label. Besides “R”/”G” the following label pairs may be used: “Red”/”Green”, and “Cy5”/”Cy3”. Lower

case versions may also be used and the order of columns may be reversed. The second and subsequent rows of the file contain names of the image files, one pair per line.

The columns of the image name file should be separated by “white space” - any number of spaces and TAB’s.

Spot also supports TIFF images which contain both the Red and Green channels, as produced for example by GenePix. In this case the image name file should contain one column only, with no column labels. An example image name file for this type of image follows.

```
array1.combined1.tif  
array1.combined2.tif  
array1.combined3.tif
```

Spot is capable of analysing arrays with arbitrary numbers of channels. However these channels must be in a single image file using the format just described.

A.2 The Parameter File

This is a file named **parameters.array1** which contains basic structure and shift tolerance information. A sample is as follows.

```
structure(list(nspot.r = 18, # Number of rows of spots per grid  
              nspot.c = 24, # Number of columns of spots per grid  
              ngrid.r = 1, # Number of rows of grids per image  
              ngrid.c = 4, # Number of columns of grids per image  
              tolerance.r = 20, # Top/bottom translation tolerance  
              tolerance.c = 20),# Left/right translation tolerance  
          .Names = c("nspot.r","nspot.c", "ngrid.r", "ngrid.c",  
                    "tolerance.r", "tolerance.c"  
          ))
```

This format is R's format for ASCII representation of R objects. Spaces and TABS are unnecessary, but improve readability. The “#” character and any text following it on a line are treated as a comment and ignored.

The meanings of the entries in this file are given in the comments in the sample file.

A.3 The Template File

The template file is named **template.array** and describes the position of the top leftmost spot of each grid block as well as the block size. The important information in this file is actually the relative position of grid blocks, because the absolute position is either supplied by the user or estimated. When creating the template file the user is prompted to click on the top left of each block (which provides the grid block position information) and the bottom right of the last block (which provides the grid block size information). This information is stored in a file using R's ASCII format. Our example looks like this

```
structure(list(r = c(136, 136, 137, 136), # row coordinates
              c = c(325, 777, 1226, 1678), # column coordinates
              gridsize.r = 306, gridsize.c = 414), # block size
          .Names = c("r", "c", "gridsize.r", "gridsize.c"))
```

There are two options if you wish to edit either the template or parameters files and for some reason do not want to use the graphical tools. The first is to directly edit the text, but be aware that the syntax is important and that getting it wrong will result in errors from R. The other alternative is to modify the objects inside R and then resave the object using `dput`. This is actually what the graphical tools do.