Image Analysis with Rapid and Accurate 2D Gaussian Fitting

Stephen M. Anthony[a] and Steve Granick[a,b]

Departments of Chemistry,[a] Materials Science and Engineering,[b] and Physics[b]

University of Illinois, Urbana, IL 61801

Supporting Information

Included below are sample Matlab implementations of the weighted regression algorithm developed in this paper. Gauss2dcirc.m applies a 2-dimensional circular Gaussian fit, while Gauss2dellipse applies a 2-dimensional elliptical Gaussian fit where the major and minor axes are aligned with the x and y coordinates of the pixels. Both programs assume that preprocessing has been applied such that the background offset has already been subtracted.

```
function [xc,yc,Amp,width]=gauss2dcirc(z,x,y,noiselevel)

%[xc,yc,Amp,width]=gauss2dcirc(arr,x,y,noiselevel)
%
%GAUSS2DCIRC.m attempts to find the best 2D circular Gaussian fit for the
%selected region of the image.
%
%Copyright 2008 Stephen M. Anthony, U. Illinois Urbana-Champaign
%
%This program is free software: you can redistribute it and/or modify
%it under the terms of the GNU General Public License as published by
%the Free Software Foundation, either version 3 of the License, or
%(at your option) any later version.
%
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%GNU General Public License for more details.
%
%You should have received a copy of the GNU General Public License
%along with this program.  If not, see <http://www.gnu.org/licenses/>
%
%INPUTS:    Z:          The subregion of the image to be fit, with each
%                       element giving the value of one pixel.
%           X:          This matrix or vector should be the same size as
%                       z, with each element giving the x-coordinate of
%                       the corresponding pixel.
%           Y:          This matrix or vector should be the same size as
%                       z, with each element giving the y-coordinate of
%                       the corresponding pixel.
%           NOISELEVEL: The standard deviation of the background noise.
```

```
%
%OUTPUTS:   XC,YC:      The center of the Gaussian in x and y
%           W:          The width of the Gaussian.
%           A:          The amplitude of the Gaussian.

%Convert to column form, in case it is not already
x=x(:); y=y(:);
Z=z(:)+1e-15;
%The miniscule offset on Z has no significant effect, but is a shortcut to
%ensure that no element of Z equals 0. The probability that an element of
%z equals 0 is not insignificant (as we may be working with integers, the
%probability than an element equals exactly -1e-15 is negligible.

%Compute the weighting. This is approximate, as we are working in the log
%scale, where the noise will be asymmetric. Bounds were placed on this to
%avoid logs of negative numbers, but for any case the bound triggers, the
%weight will be quite low anyway.
noise=log(max(Z+noiselevel,1e-10))-log(max(Z-noiselevel,1e-20));
wght=(1./noise);
wght(Z<=0)=0;

n=[x y log(Z) ones(size(x))].*(wght*ones(1,4));
d=-(x.^2+y.^2).*wght;
a=n\d;
%In the least squares sense, a was selected such that
%   n*a = d
%or the best possible match thereof.

%Extract the desired values
xc = -.5*a(1);
yc = -.5*a(2);
width=sqrt(a(3)/2);
Amp=exp((a(4)-xc^2-yc^2)/(-2*width^2));
```

```
function [xc,yc,Amp,wx,wy]=gauss2dellipse(z,x,y,noiselevel)

%[xc,yc,Amp,wx,wy]=gauss2dellipse(z,x,y,noiselevel)
%
%GAUSS2DCIRC.m attempts to find the best 2D circular Gaussian fit for the
%selected region of the image.
%
%Copyright 2008 Stephen M. Anthony, U. Illinois Urbana-Champaign
%
%This program is free software: you can redistribute it and/or modify
%it under the terms of the GNU General Public License as published by
%the Free Software Foundation, either version 3 of the License, or
%(at your option) any later version.
%
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%GNU General Public License for more details.
%
%INPUTS:    Z:          The square or rectangular subregion of the image to
%                       be fit, with each element giving the value of one
%                       pixel.
%           X:          This matrix or vector should be the same size as
%                       z, with each element giving the x-coordinate of
%                       the corresponding pixel.
%           Y:          This matrix or vector should be the same size as
%                       z, with each element giving the y-coordinate of
%                       the corresponding pixel.
%           NOISELEVEL: The standard deviation of the background noise.
%
%OUTPUTS:   XC,YC:      The center of the Gaussian in x and y
%           WX:         The width of the Gaussian, in the X direction
%           WY:         The width of the Gaussian, in the Y direction
%           A:          The amplitude of the Gaussian.

%For elliptical Gaussians, if z is a square region around the brightest
%pixel, then only a rectangular sub-region contains significant signal if
%the widths are not identical. Apply a first pass, excluding rows and
%columns which do not contain signal.
cols=find(mean(z,1)>4*noiselevel/sqrt(size(z,2)));
rows=find(mean(z,2)>4*noiselevel/sqrt(size(z,1)));

%Extract the relevant portions
z=z(rows,cols);
x=x(rows,cols);
y=y(rows,cols);

%Convert to column form, if it wasn't already
x=x(:); y=y(:);
Z=z(:)+1e-15;
%The miniscule offset on Z has no significant effect, but is a shortcut to
%ensure that no element of Z equals 0. The probability that an element of
%z equals 0 is not insignificant (as we may be working with integers, the
%probability than an element equals exactly -1e-15 is negligible.
```

```
%Compute the weighting. This is approximate, as we are working in the log
%scale, where the noise will be asymmetric. Bounds were placed on this to
%avoid logs of negative numbers, but for any case the bound triggers, the
%weight will be quite low anyway.
noise=log(max(Z+noiselevel,1e-10))-log(max(Z-noiselevel,1e-20));
wght=(1./noise);

%Apply a threshold, to exclude points where the signal-to-noise ratio is
%too low to contribute usefully
wght(Z<=noiselevel)=0;


n=[ones(size(x)) x x.^2 y y.^2].*(wght*ones(1,5));
d=log(Z).*wght;
a=n\d;
%In the least squares sense, a was selected such that
%   n*a = d
%or the best possible match thereof.

%Extract the desired values
wx=sqrt(-.5/a(3));
wy=sqrt(-.5/a(5));
xc=a(2)/(-2*a(3));
yc=a(4)/(-2*a(5));
Amp=exp(a(1)-a(3)*xc^2-a(5)*yc^2);
```