**README FOR EXAMPLE.M**

Authors: Carlas .S. Smith, Sjoerd Stallinga, Keith .A. Lidke, Bernd Rieger, David Grünwald

Citation: Please cite the following publication in case you use the software:

**Carlas S. Smith et. al. Probability-based particle detection that enables threshold-free and robust in vivo single molecule tracking**

This helps us to ensure financial support for our projects and allow further development of this software. Thank you

Conditions of use: The University of Massachusetts has developed and is the owner of the image processing routines distributed in this package, hereafter called SOFTWARE. The SOFTWARE is free for non-commercial use by students and staff in universities or non-profit research institutes. Redistribution of SOFTWARE or parts thereof in any form is not permitted. This SOFTWARE 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.

Requirements: This code requires the toolbox DIPimage [1] and **CUDA 6.5** [2] to be installed. The windows watchdog that will stop the GPU execution if not disabled set [3]: **TdrLevel=0; TdrDelay=180**

[1] <https://developer.nvidia.com/cuda-toolkit-65>

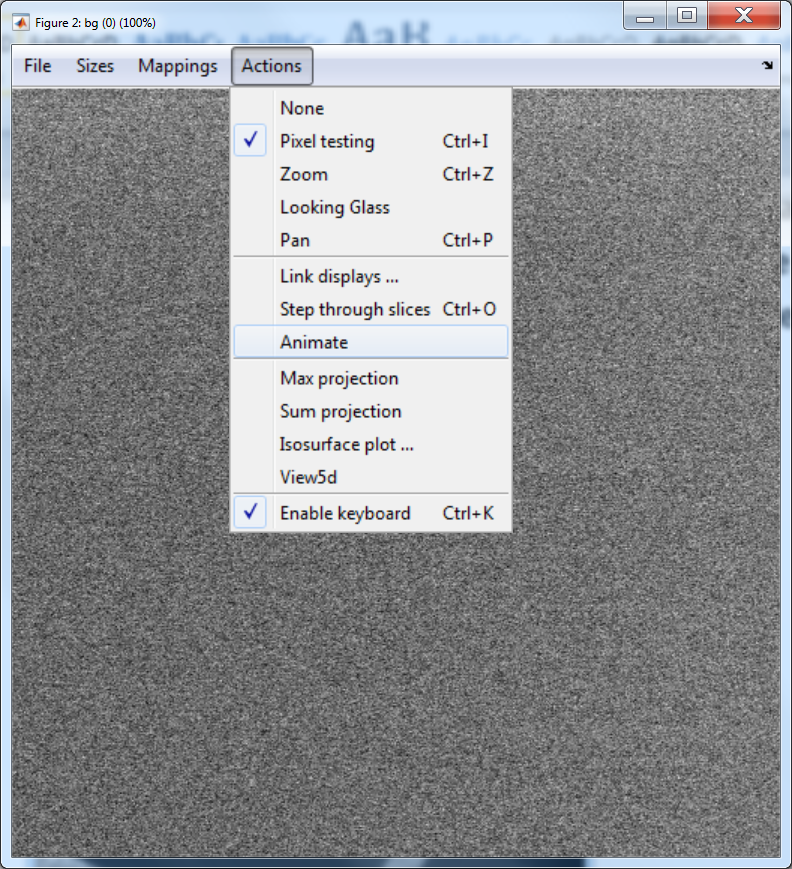
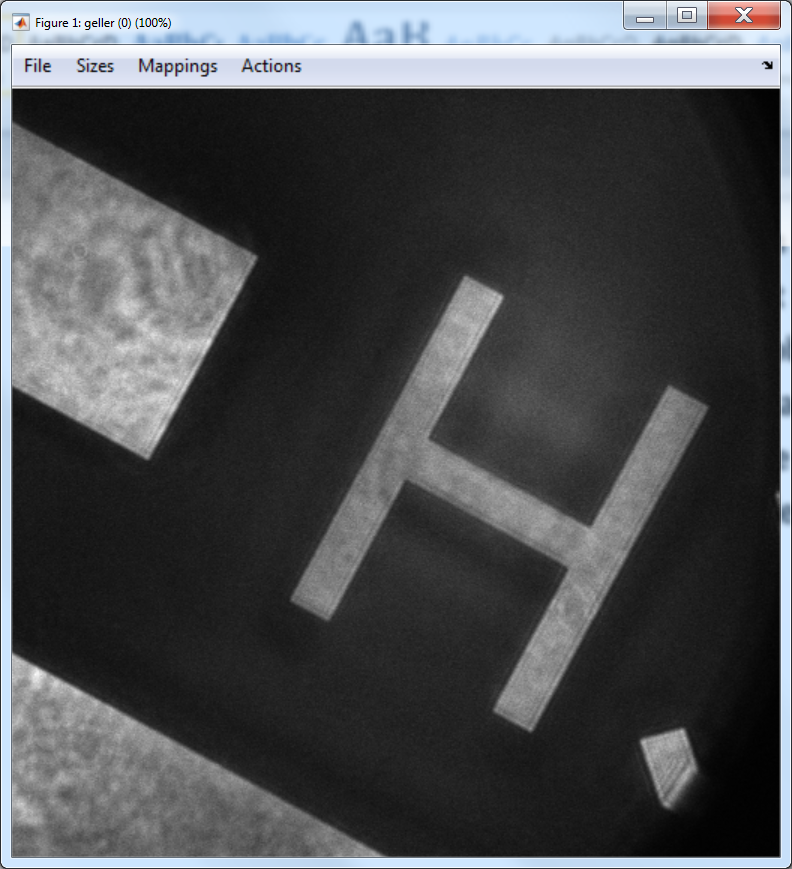
[2] <http://www.diplib.org/download>

[3] <https://msdn.microsoft.com/en-us/library/windows/hardware/ff569918(v=vs.85).aspx>

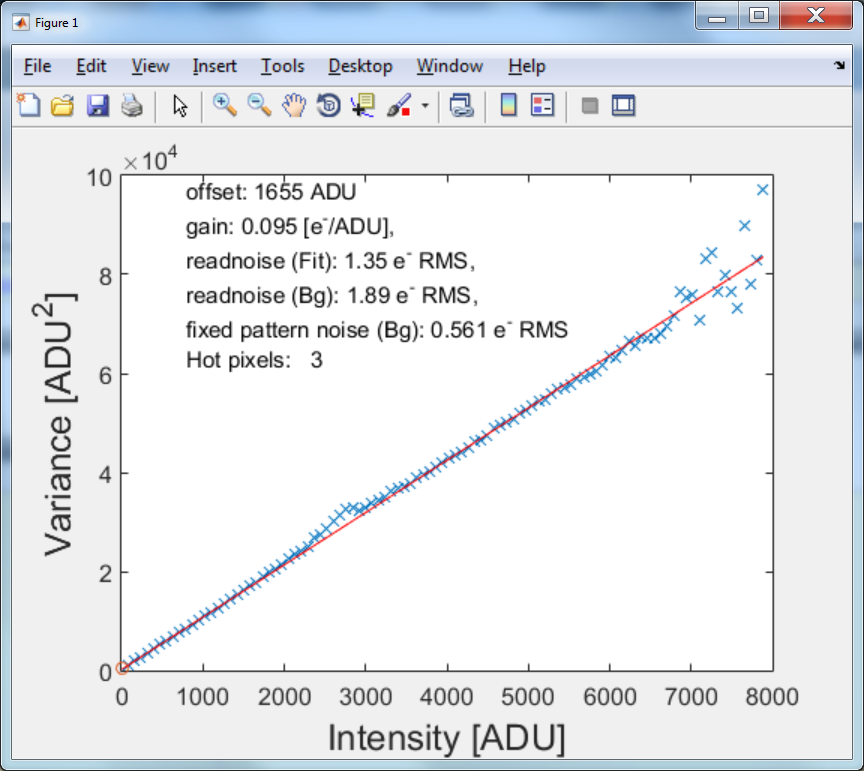
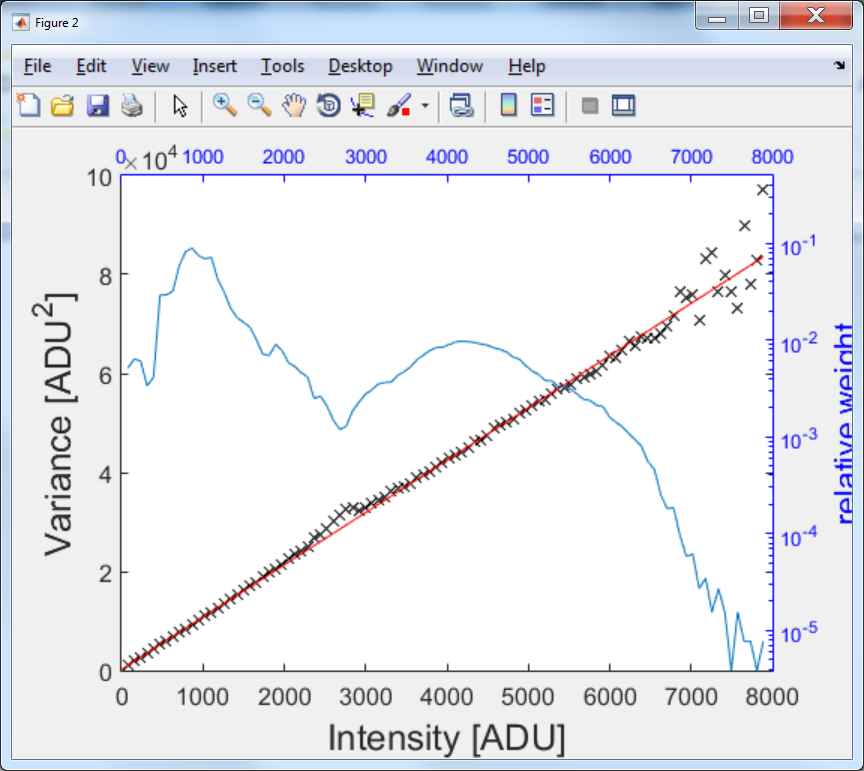
This document describes how the steps work to obtain a set of detections and localizations usable for a PALM/STORM/Tracking experiment. If you have any problems please send an email to: carlas.smith@umassmed.edu.

**OUTPUT EXAMPLE.M**

It is important to calibrate the camera to obtain the precision at, which a single molecule is localized. Similarly, it is very important to calibrate the camera to obtain a false alarm probability of detection based on the calibration data: dark counts and a static object in which the intensities are represented that you will use in your experiment (typically between 1 and 10^5), both for recorded for 100-200 frames.



The output of the calibration gives two figures the first showing mean intensity versus the intensity variance and the second plot shows the same, but also showing the representation intensity freqency. The aim is to have the line fit the data exactly, so that the Poisson assumption is satisfied and the corrected image can be used for maximum likelihood fitting.



The result is a calibrated image where the counts resemble a Poisson characteristic



This image can now be used to for GLRT detection. The detections can be further filtered by use of features determined from the detections, such as the distance to the next detection or the stretched false alarm probability. The filtering is done using the function preFilterFits and results in a figure and text in Matlab’s command window.

*Allocating host and device memory...  
plan[0].Nfits 88044 (blockDim 96, gridDim 918) on Quadro K2100M*

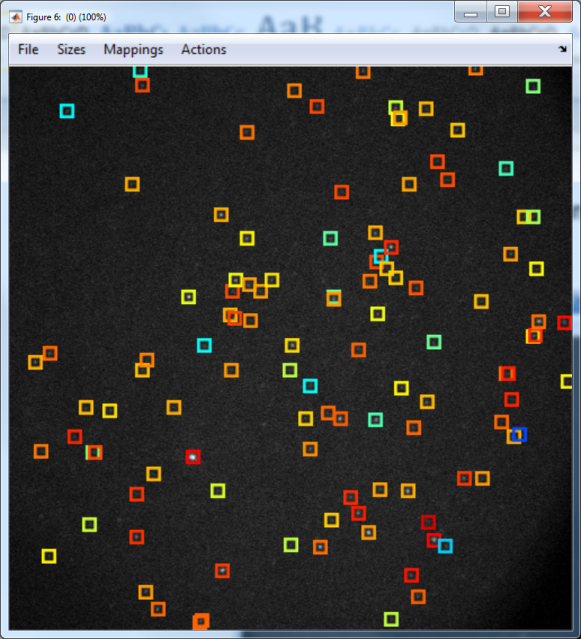
*All fits are distributed over the GPU 1 devices*

*Cleanup...*

*GPU Processing time 3188.118122 (ms) for 88044 fits*

*Eliminated 0 localizations based on detection probability*

*Eliminated 0 localizations smaller than minPixelDist*



From the command window output we observe that no detections have been eliminated based on the detection probability or on the distance between the detections (minPixelDist). The boundaries for these filters can be set in the structure paramsPreFilterFits.

Next all the detections are fitted twice once with a fixed PSF width () and once the width of the PSF is fitted as well. Both these fits are performed using our GPU fitting routine. The fits can again be filtered based on featured obtained from these fits. The filtering is done using the function filterFits and results in a figure and text in Matlab’s command window. The boundaries for these filters can be set in the structure paramsFilterFits.

*Allocating host and device memory...*

*plan[0].Nfits 5223 (blockDim 96, gridDim 55) on Quadro K2100M*

*All fits are distributed over the GPU 1 devices*

*Cleanup...*

*GPU Processing time 177.780254 (ms) for 5223 fits*

*Allocating host and device memory...*

*plan[0].Nfits 5223 (blockDim 96, gridDim 55) on Quadro K2100M*

*All fits are distributed over the GPU 1 devices*

*Cleanup...*

*GPU Processing time 187.113072 (ms) for 5223 fits*

*Eliminated 37 localizations based on localization precision*

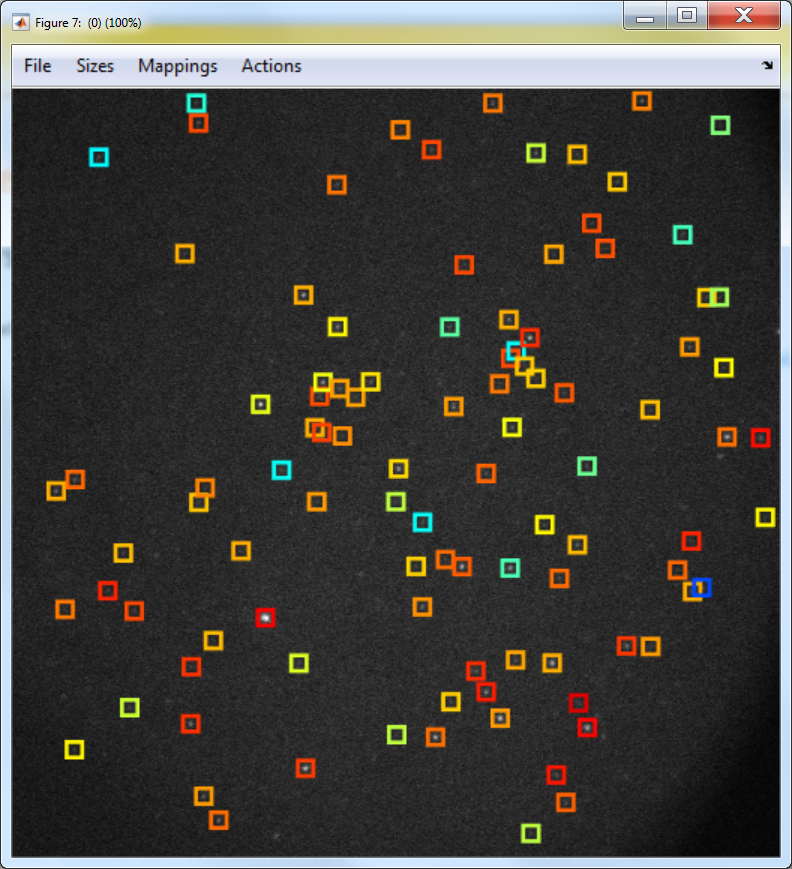
*Eliminated 35 localizations based on probability of false positive*

*Eliminated 31 localizations based on photons*

*Eliminated 0 localizations based on bg*

*Eliminated 0 localizations based on location*

*Eliminated 299 localizations smaller than minPixelDist*



Here we observe that 37 fits are eliminated based on localization precision, 35 based on probability of false alarm, 31 based on the number of photons, 0 on the number of photons per pixel background fluorescence, 0 fits are eliminated because the fitting routine found a location outside of the subregion in which the fit was executed and 299 fits where eliminated because the distance between the positions of the single molecules was too small (minPixelDist).

**OVERVIEW ADDITIONAL EXAMPLES**

Camera Calibration

1. For all MLE solutions the gain of the EMCCD camera has to be calibrated so that the noise model accurately approximates the assumed noise model. An example is shown in /Camera Calibration Example/camCal.m.

GPU GLRT Detection

1. Detection on Simulated Data
   1. GLRT results in an binary image with positive pixels an example is shown in /Single Molecule Detection/exampleDetectionCluster.m
   2. An example on simulated dStorm data: /Single Molecule Detection/dstormSimDetection.m
   3. Creation of the Receiver Operating Curve (ROC); As shown in the supplementary information the detection performance can be judged using a ROC in: /Single Molecule Detection/limitDistributionGLRTWithTheory.m an example is given how to construct such curves.

GPU single molecule fitting

1. Single molecule fitting on simulated data is shown in: /Single Molecule Fitting/gpuGaussMLEv3Test.m

Workflow

1. All detection and fits can be filtered using features determined from them. An example of how to incorporate these filtering steps into an experiment is presented in: Example.m.

**FUNCTIONS**

1. **The first version of the software for the article *Probability-based particle detection that enables threshold-free and robust in vivo single molecule tracking*, by Carlas .S. Smith, Sjoerd Stallinga, Keith .A. Lidke, Bernd Rieger, David Grünwald**

Method: [coords,dectectionPar,cutProcess] = LLRMapv2(process,PSFSigma,minPixels,

compReduction,iterations,split,maxCudaFits)

Example: Example.m.

PARAMETERS:

process: single molecule data (corrected for gain and dark counts)

PSFSigma: sigma of diffraction limited PSF in pixels

minPixels: minimal size of detection cluster. minPixels = [] obbits

calculation of detection features dectectionPar =[]

compReduction: computational complexity reduction level in standard

deviations from mean. This rejects regions with small change of single

molecules, based on wavelet filtering. compReduction = [] results in

computation of GLRT on all pixels.

significance: false discovery rate

iterations: number of iterations for MLE algorithm.

split: split clusters using watershed algorithm

maxCudaFits: maximum numbers of CUDA fits on single GPU device. If

n CUDA enables devices are ready 1xn vector is expected.

DEFAULTS:

minPixels = 0;

compReduction = 2;

significance = 0.05;

split = false;

maxCudaFits = 1000000;

OUTPUTS:

coords: matrix with detection coordinates.

dectectionPar

.circularity: circularity of cluster using P2A feature.

.pH1: detection probability of cluster.

.clusterSize: cluster size

.ll: labeled clusters

.hh: binary image containing raw detections

.pfa\_adj: false discovery rate corrected false positive probability   
 cutProcess: cropped image with the actually test pixels.

1. **The version 3 of the software for the article *Fast, single-molecule localization that***

***achieves theoretically minimum uncertainty* by Carlas S. Smith, Nikolai Joseph, Bernd Rieger & Keith A.Lidke.**

Method:

[P,CRLB,LL] = gpuGaussMLEv3(ROIStack,PSFSigma,Iterations, fittype,display,maxCudaFits);

Example: Example.m

PARAMETERS:

ROIStack: stack of single molecule data create

PSFSigma: depending on fittype scalar or vector.

Iterations: number of iterations for GPU routine

The method of fitting is given by the fittype variable.

fittype

fittype=0: Fits (Photons,Bg) under H1 and (Bg) under H0 given .

fittype=1: Fits (x,y,bg,Photns) under H1 and (Bg) under H0 given .

fittype=2: Fits (x,y,bg,Photons,) under H1 and (Bg) under H0.

fittype=3: Fits (x,y,bg,Photons,z) under H1 and (Bg) under H0 given ,.

fittype=4: Fits (x,y,bg,Photons,,) under H1 and (Bg) under H0 .

fittype=5: Fits (x,y,z,bg,Photons, ,.) under H1 and (Bg) under H0.

display: boolean to display output.

maxCudaFits: maximum amount of CUDA fits per GPU enables device.

OUTPUTS:

P is the N x M matrix of found parameters

CRLB is the N x M matrix of found parameters of Cramer-Rao Lower Bound calculated

variances for each parameter.

The CRLB is now calculated internally using an LU decomposition method for inverting

the Fisher information matrix.

The center of mass estimation is now used for all models as a starting guess for the x,y positions.

This should result in fewer iterations to achieve convergence.

LL is the 4 x M matrix of calculated likelihood values likelihood

ratio, non-centrality parameter, P\_FA, PD, respectively.

where N is the number of fits and M is the number of fitted variables (i.e. 6 for

fittype=4).

COMMENT

This code requires the NVIDA CUDA 6.5 Driver and Toolkit to be installed.