User Guide for DDC

By Christopher H. Bohrer

This is a user guide for DDC, which uses the pair-wise distances to correct for blinking in SMLM images. In this user guide I discuss the two parameters that you must define to use DDC and the inner workings of the code, which was initially written in Matlab (Feel free to rewrite or use any of the ideas put forth in the code). I start by showing you how to organize your data and then discuss some initial plots. I also show you how to split up your images if you have a large number of localizations within your images and then how to combine them again. Second, I discuss the two parameters you need to define and how to easily determine them, given just your data. Finally, I show you how to run DDC and how to look at the results.

# Organizing Your Data and Qualitative Blinking

In order to use DDC you need to have your data organized in a particular way. You will need to store the localizations within a cell called **LocalizationsFinal**, where **LocalizationsFinal{1}** contains the **[x, y]** coordinates of the localizations within the first image (and so on). You will also need the corresponding frames of the data for each image and store them in a cell called **Frame\_Information**. That is **Frame\_Information{i}** are the frames for the localizations in **LocalizationsFinal{i}**. If you want an example as to how your data should be organized, there is a .**mat** file named **Example\_3d\_2darkstate\_random\_data.mat**, which contains some ‘3d’ simulation data organized in the correct manner that has yet to be analyzed. This simulation data contains 100,000 localizations distributed throughout a 25000 nm^3 volume. The distribution of these localizations is not homogeneous and instead contains clusters on a similar size scale as the localization precision, approximately 15% of the localizations are within the clusters and the rest are randomly distributed throughout the volume. The photo-kinetics of the fluorophores is the same as the 2-dark state fluorophores simulated within the main text.

# Splitting Up Images with a Large Number of Localizations

If you have a large number of localizations within one image (greater than 6000 localizations) DDC will likely converge very slowly (depends on your computer). This is because DDC needs to calculate the pairwise distance distribution between all localizations and the number of distances increases exponentially with the number of localizations. The dataset that I have provided contains 100,000 localizations within the first image.

*{Note: Another alternative is to only use distances between localizations that are within a certain distance, and though the math works out fine for this, this methodology has not been extensively tested yet by us.}*

To get around this issue, I will show you how to split up your image so you can then analyze the individual sections in parallel. To split up an image go to the script **Split\_UP\_Image.m**. This will split up your image so that the individual images within the cell **LocalizationsFinal**, will have around *min\_loc*, which in this case is set to 4000 within the script. The script goes through and splits the image into sections containing (approximately) the correct number of localizations. It then adds a “buffer” zone around each image of ~200 nm so that there are no boundary effects when combining the images back together. {Note: This “buffer” zone may need to be modified based off of the resolution of the experiment.}

If you click run in the **Split\_UP\_Image.m** you will be able to select your data. {*Note: The data in the large image should still have the two cells containing the data* ***LocalizationsFinal*** *and* ***Frame\_Information****, also there is also a cell which contains the true\_localizations which will be used to evaluate the performance of* ***DDC****.*}

When you click **RUN** you will then have the opportunity to click on your data and the script will run. The script will then save a new **.mat** file with the same name as the data with the word SPLIT out front and also generate some 3d scatter plots colored through time. This script should take about 5 min to run on this data set, as there is a bit of an optimization step within the script.

An example of one of the individual images in a 3d scatter plot is shown in **Fig. 1** and is colored through time. You can see small punctate clusters of the same color, which is indicative of blinking. You can also see some of the real clusters that have multiple colors within the clusters.

|  |
| --- |
| Figure : Here is a sample of the localizations from the example dataset, colored through time. As you can see there are small punctate clusters of the same color which is indicative of blinking. Also, you can see within the image small clusters that have multiple colors, these are actual clusters with a higher density of true localizations. |

If you want to combine images into a big one go to the script **Combine\_Images.m**. After you have run DDC and you have your analyzed data. You can click run with **Combine\_Images.m**. This will produce a new analyzed data file with the same name, but with the name combined out front. We will show you how to do this at the end of this user guide.

# Defining the Two Parameters Used By DDC

DDC needs two parameters to work properly, 1. the bin resolution of the SMLM experiment and 2. The frame difference where steady state is reached (*N*). I will describe both of these parameters and how to input them into DDC so that your images are processed properly.

### Determining N

In order for DDC to work it needs to be able to obtain the “true pairwise distance distribution” (TPDD) of the underlying molecules. See the main text for how this was derived and how it is utilized within DDC. To determine the TPDD you will need to determine when the pairwise distance distribution for the localizations (separated by a certain frame difference) reaches a steady state. The name of this parameter is *N.* To determine *N* for a dataset, go to the script named **Determine\_N.m**.

To determine *N* we are going to plot , which is the summed difference between the cumulative distribution function of the pairwise distance distribution at a certain frame difference and the cumulative distribution function of the pairwise distance distribution at a frame difference of 1. {*Note: there is one parameter you need to define within the script, NFTP. This is simply the highest where will be calculated, we put this as 200 frames for this dataset. If you are looking at the data for the first time it is better that you pick larger rather than smaller values.*} After running this script and clicking on the **Split up Images** you will generate a plot like the one below {*Note: this will take about 2min to run*}:

|  |
| --- |
| Figure : Here is a plot showing the change in the pairwise distance distributions at various frame differences, relative to when the frame difference is equal to 1. You can clearly see that Z reaches a steady state after about 70 frames. Therefore, a good estimation for N is any value above this, to be safe when applying DDC we will set N equal to 140 frames. |

In **Fig. 2** you can see that the pairwise distance distributions approach a steady state after about 70 frames. When the pairwise distance distributions reach a steady state, the pairwise distance distribution between these localizations separated by that frame difference is approximately equal to the TPDD. Based off of **Fig. 2** any *N* above 70 (frames) would be fine to determine the true pairwise distance distribution. Though to be safe, within this user guide we set *N* to be 140 (frames). (Once again it is better to go larger rather than smaller for almost all of DDC.

### Determining the Bin Resolution

Now all you need is to determine the average bin resolution of the SMLM experiment. The bin resolution is the width of the bins for the histograms of the probability distributions, like the TPDD.

To start you will need to determine the “classic” resolution of the SMLM data. You can do this in a number of different ways and we will not discuss this in depth, as it is well discussed within

|  |
| --- |
| Figure 3: Here you can see the true pairwise distance distributions for the first 3 (Split) images using a bin width of 40nm [Equal to the “classical” resolution of the simulation data]. You can see that given the amount of data and N the TPDD using all localizations is well defined and is very close to the TPDD determined only using the True Localizations. |

the literature. For the example dataset the “classical” resolution of the SMLM data is equal to 40nm, {*1 sigma*}, you will need this value as a starting point to determine the bin resolution.

Now to get an idea as to how well the true pairwise distributions are sampled for each split-up image, you should select the script **Determine\_Res.m**. At the top of the script you should input your “classical” resolution, as this will be a starting point for determining the bin resolution for DDC. {*Note: the “classical” resolution is normally adequate for defining the bin widths of the probability distributions used within DDC, that is as long as there is a sufficient number of localizations.*} You will also have to input *N* at the top of the script, as the TPDD needs to be generated.

Upon running this script, you will see two plots of the TPDD’s for the first 3 (split) images using all of the localizations and only the true localizations. If you are using **DDC** on actual data you will not be able to generate the bottom plot, as you will not know the actual true localizations. In **Fig. 3** You can see that the two plots are approximately the same and that the probability distribution is well defined. This suggests that the “classical” resolution in this case is sufficient to create the probability distribution. Generally, you do not want to go lower than the “classical” resolution of the experiment.

Here, when I state you need to find the resolution, you do not need a super accurate measurement of this value as its exact value is not super important to DDC. In DDC the resolution parameter defines the bin size of the pairwise distance distribution and should be large enough that there are a sufficient number of distances within each bin.

We are now finished finding the two parameters that DDC needs, (*N=140 and Res=40nm)*.

# Running DDC

Now that you have the two parameters that **DDC** needs, we can run the program on the split-up images. To run **DDC** go to the script **run\_script\_DDC.m**, here there will be 3 values that you will need to set.

1. For the variable **Resolution** set it equal to the bin resolution that was determined (40)
2. For the variable **N\_f** set it equal to 140 as determined above (*N*)
3. For the variable **Stepper**, set it equal to 300, {*Note: This is equal to the number of steps that the Markov Chain Monte Carlo will go after reaching a new maximum of the Likelihood.*}

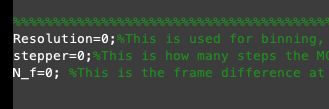


Figure :These are the 3 values you need to define

Now that you have everything set up all you need to do is click **RUN** and then you have to select your split image file and **DDC** will start working. First, **DDC** will complete **STEP 1** and will alert you of its progress. This should take a small amount of time to finish the first step.

The second step depends upon your hardware and runs in parallel. For 6 cores, **DDC**, took about a day to finish its run. {*Note: if you do not have time to wait for this to run for this example, the analyzed data is stored in* ***Analyzed\_Time06-Jun-2019\_Split\_Example\_3d\_2darkstate\_random\_data.mat***}

The output during a run looks like the following:



|  |
| --- |
| Figure 5: The performance improvement in the image error by performing the MCMC within DDC. As you can see there is a very significant improvement in terms of the relative image error. |

The Score is the relative score that is described within the main text and the MLE is the difference between the likelihood of the subset of true localizations determined by DDC and the actual likelihood of the True Localizations (If the true localizations are known), otherwise it is just the log of the likelihood.

When **DDC** is finished the final localizations with their frames will be stored in the following two cells: 1. **Final\_Loc\_Blinking\_Corrected** and 2. **Final\_Frame\_Blinking\_Corrected**.

# Combining the Split Images

Now that we have run **DDC** and have the blinking corrected split images, we are going to recombine the images together. To do this go to the script **Combine\_Images.m**, and just like the others click run and then click on the analyzed data. This will save a file with the name *Combined* out front. You can then look at the localizations as you see fit. The localizations will be stored in two arrays **Combined\_Image** and **Combined\_Image\_Frame.**

# Evaluating Performance of DDC on Split Images

To illustrate the effectiveness of the maximization of the likelihood in DDC, we will plot the relative Image Errors before and after the MCMC phase space search. This data was saved out during the run of DDC. To look at the effect of the MCMC go to the script **Evaluate\_Your\_Data\_USER\_GUIDE.m.** Then load in the *Analyzed\_file* that DDC originally output and then click run. It will generate a figure like that in Fig. 5 here which shows there is a dramatic improvement in the Image Error after maximizing the likelihood.