Code is shown in blue

Requirements:

* Installation of Python (>= 3.6)
  + Preferably Anaconda + Spyder
* Installation of all other packages (TrackPy etc.)
  + Spyder issues an error in the console when a package is missing.
  + These can then be installed in the Anaconda Promt.
  + It is helpful here to search the package online https://pypi.org/)NanoObjectDetection Modul
* Download a Github release (<https://github.com/RonnyFoerster/NanoObjectDetection/releases>)
  + Copy the folder NanoObjectDetection into the directory of the Python packages
  + E.g.: C:\ProgramData\Anaconda3\Lib\site-packages\NanoObjectDetection
  + Import package by import NanoObjectDetection as nd

In general, the data should be sparse enough, sothat the captured particles are individual (well separated from each other) for at least 100 frames. The higher the framerate the easier this is. A longer track leads to a smaller confidence interval in the final diameter estimation.

Example:

* Code is shown in blue
* **The module contains several examples which run fully automatically. However, they can also be executed block by block (in nd.Tutorial.py) in order to learn, debug and optimize**
  + **nd.Tutorial.RandomWalk ()**
  + **nd.Tutorial.MonaARHCF ()**
  + **nd.Tutorial.StefanARHCF ()**
  + **nd.Tutorial.HeraeusNanobore()**
* For a new set of data: Copy the parameter json file (for example: “default\_json.json” or “tutorial\_50nm\_beads.json” in the new folder.
* Open the json file and fill in the parameters as good as you can
  + The parameters are explained in default\_json.py
  + Be aware that python uses „\\“ instead of „\“ for path- and file directories
  + Each line of a key has to be finished with a comma (,), except the last entry of each block.
    - "StationaryObjects": {

"Analyze fixed spots": 0,

"Min distance to stationary object": 0

}

* + Enter the path of the new directory in „SaveFolder“ and „SaveProperties“ ( + beispielsweise „\\parameters.json“)
    - "data\_file\_name": "Z:\\Data\\TorstenWieduwilt\\190322\_test\_heraeus\\100nist\_1000\_50fps\_1.tif",
  + If you dont know your parameter you should use the default value
    - You can delete a key completely and python will reintroduce this key with the default value, if you have "DefaultParameterJsonFile": "default",
  + For the Nanobore the following Parameters are important:
    - "Shape": "round" , because the fiber is round
    - "SimulateData": 0, to not generate artificial data
    - "Remove\_CameraOffset": 1, because the camera has offset
    - "Remove\_Laserfluctuation": 0, it would be nice to remove the laser fluctuations. However, for this at least a couple of dozen of particles are required which do not fit in a nanobore (in contrast to a ARHCF)
    - "Help": "ROI": 1, "Bead brightness": 1, "Bead size": 1, for opening the help to adjust new experimental parameters in case of a new specimen or changed experimental parameters (Fiber, Camera, Objective etc.)
    - "Analyze fixed spots": 1, because partikels to attached to the fiber in the Nanobore
    - Dift: "Apply": 1, because drift occurs
    - "Do transversal drift correction": 0, because you need a lot of particles to do that, which do not fit in a nanobore (in contrast to a ARHCF)
    - "EstimateHindranceFactor": 1, because the particle diameter is not neglectable comparing the the fiber diameter.
  + The Json file can be changed (and saved) at any time, because the parameters are read in at any block. This makes debugging and optimizing easier.
* Opening of MainCode.py in the Python Editor (e.g. Spyder)
  + Inserting the full path of the parameter json file:
    - ParameterJsonFile = "Z:\\Datenauswertung\\Torsten\_Wieduwilt\\190322\\100nist\_1000\_50fps\_1\\100nist\_1000\_50fps\_1.json"
  + Execute the code blockwise (use Shift + Enter)
    - As soon as the json file is optimized the code can run completely at once.
* The help opens in nd.AdjustSettings.FindSpot(rawframes\_rot, ParameterJsonFile)
  + „ROI“
    - Shows a maximum projection.
    - Choose a region of interest (ROI). A ROI is computed much faster than the entire dataset. Thus, parameters can be optimized faster. This is vaild for frames, too.
    - The ROI can be inserted in the json.
    - A huge dataset can be evaluated best, if the parameters are first optimized on a ROI with a few hundred frames.
  + "Bead brightness"
    - The image of the particle detection (successful detection in red circles) is saved. The image must be opened in the displayed folder.
    - The help ask if the detection is good (yes (y) or no(n))
      * If not, the question arises what is wrong:
        + 1 - Bright (isolated) spots not recognized
        + 2 - Spots where nothing is to be seen
        + 3 - Bright (non-isolated) spots not recognized but you would like to have them both
        + 4 - Much more recognized spots than I think I have particles
      * Insert the number of the problem (1-4). The problem displays which parameter needs to changed in which direction.
      * Then the routine runs again.
    - If the help is not required anymore, the help parameter can be set to 0.
  + "Bead size"
    - The histogram of the subpixel accuracy should be flat (noise allowed). Is the estimated particle size to small the histogram has a dip in the middle at 0.5.
      * The help routine ask again if the user is satisfied or not
    - The plot of the images can be in an external window or inside the consol. This can be set in the console using:
      * External window: %matplotlib qt
      * Consol: %matplotlib inline
* The remaining code runs without any help. Not all parameter have a parameter estimation, so that the algorithm produces results if a new specimen or component is in. Than the parameters need to be optimized by hand.