All algorithms are implemented in Python and were from the ndparse algorithms.py source code, or as outlined from ndod. Following the pipeline as outlined in ndod, I started with a 2D TIFF image screencap conversion of Fear200 from ndviz (coordinates: 2264, 991, 0).

To generate the initial classifier, I used the heads-up display (GUI interface) for Ilastik to manually select bright points and their respective backgrounds. By playing with the functions, I managed to get an h5 probability output. I've preliminarily outlined this part of the code, but this part mostly is user-input driven.

For our input, we need to take the .nii NIFTI images and convert them into a useable file format for Ilastik. After downsampling the image using a Docker with ndreg/ndio, I took the downsampled .nii and made TIFFs of each individual plane. The code is in a Jupyter Notebook linked here, I was able to convert each individual plane into a TIFF because by default the .nii files have three dimensions. The first dimension is the plane, while the second is the row/the third is the column. Thus, in order to get (x, y, b) arrays with x, y being coordinates in 2D space (like we have from CSV outputs) and b referring to the grey-scale brightness value, I've outlined the code below.

Algorithm 1 Convert .nii file to TIFF images

```
Require: Raw 3D greyscale .nii file
Ensure: .nii file conforms to (pln, row, col), where the first element is the planes, the second is the
  rows, the third is the columns. The top left rows/columns are thus the darkest, while the bottom
  right are the brightest (greyscale).
  function NIITOTIFF($(p_i, x_i, y_i)$, for $i \in [n]$)
  (1) Load file using nibabel
  (2) Use getdata command from nibabel to store nii data as memmap array
  (3) Find range of planes (from 0 to k)
  (4) Convert each plane into a TIFF
      for $j := 0, \ldots, k$ do
          Convert data at $j$ plane (eg: data[j]) into a numpy array using np.asarray().
          Use scipy.toimage on the converted numpy array and save the output as a TIFF.
      end for
  end function
```

Now, we have TIFF's of each individual plane. We now can pass it into Ilastik to generate a classifier using the GUI.

Algorithm 2 Generate Ilastik Classifier on Subset of Input Data

```
Require: 1 representative sample subset of original image data (one representative TIFF plane).
Ensure: The user manually selects a subset of the data (some representative region) and uploads
  the data to Ilastik by using the gui interface.
  function CLASSIFIER(raw TIFF of representative plane)
  (1) Select at least two distinct training labels (eg: background, bright point)
  (2) Check to see if results are up to par.
  (3) Repeat until reasonable output is shown.
  (4) Save as classifier .ilp for later use with headless display
  end function
```

Algorithm 3 Run Pre-Trained Ilastik Classifier on Input Data

```
Require: .ilp classifier trained on previous data, from above.
  function RUNNING PRE-TRAINED ILASTIK CLASSIFIER(new data)
  (1) Load Ilastik Headless Display
  (2) Run .ilp on all new images
      for $k, l := 0, \ldots, n$, where $n$ is the total number of planes generated do
          Run headless ilastik on predefined .ilp on all $k$ TIFFs.
      end for
  end function
```