Decode-PAINT: 9-color decoding sample dataset user guide

Before starting analysis

1. Download Picasso following instructions (as of 10/10/2022, ver 0.4.11)
    - https://github.com/jungmannlab/picasso
2. Modify Picasso Render source code
    - picasso>gui>render.py (NOT render.py in
    - In ***def align(self)***
    -
```python

    def align(self):
        if len(self._picks) > 0:
            shift = self.shift_from_picked()
            print("Shift {}".format(shift))

            # 10/10/2022 added
            for i in range(len(shift)):
                for j in range(len(shift[i])):
                    print(shift[i][j], end=',')
                print('\n')


            sp = lib.ProgressDialog("Shifting channels", 0, len(self.locs), self)
            sp.set_value(0)
            for i, locs_ in enumerate(self.locs):
                locs_.y -= shift[0][i]
```
    -
```python
            self.update_scene()
        else:
            max_iterations = 4
            iteration = 0
            convergence = 0.001  # Thhat is 0.001 pixels ~0.13nm
            shift_x = []
            shift_y = []
            shift_z = []
            display = False

            # 10/10/2022 added
            shift_x_combined = []
            shift_y_combined = []

            progress = lib.ProgressDialog("Aligning images..", 0, max_iterations, self)
            progress.show()
            progress.set_value(0)

            for iteration in range(max_iterations):
                completed = "True"
                progress.set_value(iteration)
                shift = self.shift_from_rcc()
                sp = lib.ProgressDialog("Shifting channels", 0, len(self.locs), self)
                sp.set_value(0)
```

- ```python
              shift_y.append(np.mean(temp_shift_y))
              if len(shift) == 3:
                  shift_z.append(np.mean(temp_shift_z))
              iteration += 1
              self.update_scene()

              # 10/10/2022 added
              if len(shift_x_combined) > 0:
                  shift_x_combined = [x1 + x2 for (x1, x2) in zip(shift_x_combined, temp_shift_x)]
                  shift_y_combined = [y1 + y2 for (y1, y2) in zip(shift_y_combined, temp_shift_y)]
              else:
                  shift_x_combined = temp_shift_x
                  shift_y_combined = temp_shift_y

              # Skip when converged:
              if completed:
                  break

          progress.close()
          # Plot shift etc
          if display:
  ```
- ```python
              plt.subplot(1, 1, 1)
              plt.plot(shift_x, "o-", label="x shift")
              plt.plot(shift_y, "o-", label="y shift")
              plt.xlabel("Iteration")
              plt.ylabel("Mean Shift per Iteration (Px)")
              plt.legend(loc="best")
              fig1.show()

          # 10/10/2022 added
          for i in range(len(shift_x_combined)):
              print(shift_x_combined[i], end=',')
          print('\n')
          for j in range(len(shift_y_combined)):
              print(shift_y_combined[j], end=',')
          print('\n')

      @check_pick
      def combine(self):
          channel = self.get_channel()
          picked_locs = self.picked_locs(channel, add_group=False)
          out_locs = []
          r_max = 2 * max(
              self.infos[channel][0]["Height"], self.infos[channel][0]["Width"]
          )
          max_dark = self.infos[channel][0]["Frames"]
  ```
- In ***def _undrift_from_picked_coordinate(self, channel, picked_locs, coordinate):***

```python
            drift_mean = np.ma.average(drift, axis=0, weights=1 / msd)
            drift_mean = drift_mean.filled(np.nan)

            # Linear interpolation for frames without localizations
            def nan_helper(y):
                return np.isnan(y), lambda z: z.nonzero()[0]

            nans, nonzero = nan_helper(drift_mean)
            drift_mean[nans] = np.interp(nonzero(nans), nonzero(~nans), drift_mean[~nans])

            # 10/10/2022 added
            drift_init = drift_mean[0]
            drift_mean -= drift_init

            return drift_mean

        def _undrift_from_picked(self, channel):
            picked_locs = self.picked_locs(channel)
            status = lib.StatusDialog("Calculating drift...", self)

            drift_x = self._undrift_from_picked_coordinate(channel, picked_locs, "x")
            drift_y = self._undrift_from_picked_coordinate(channel, picked_locs, "y")
```

- These modification allows Picasso Render to:
  - Align/register localizations to the initial frame (frame 0)
  - Show XY registration shift coordinates

3. Install picasso

```
> python setup.py install
```

Sample 1: 9-color decoding (Fig. 1n)

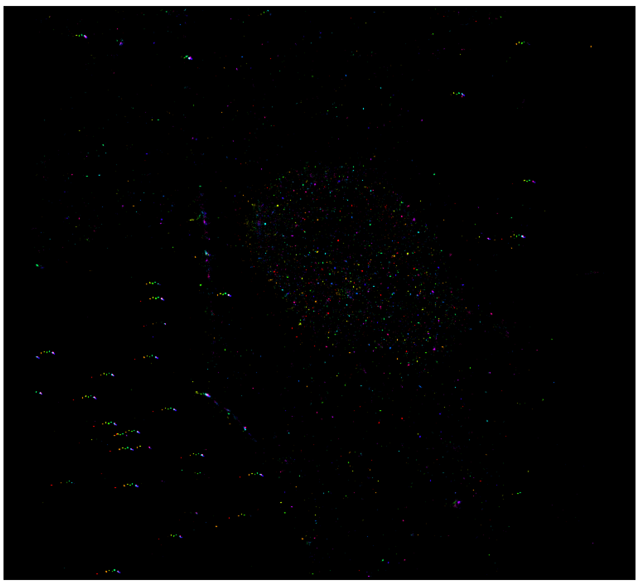
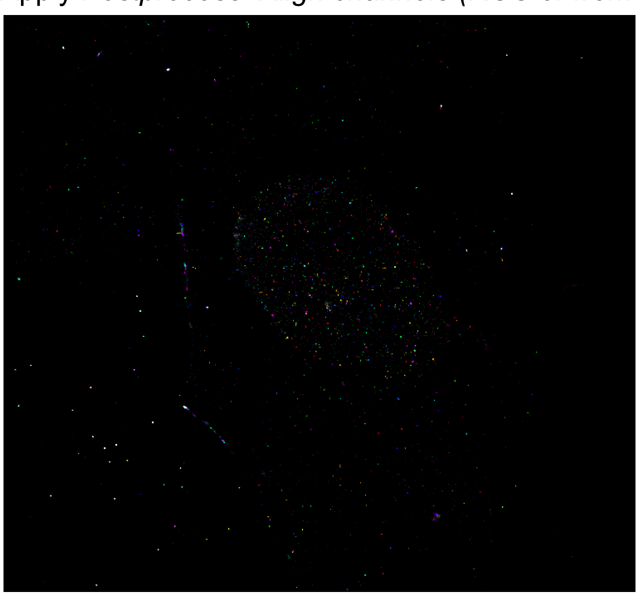
Step 1. Process raw localization data

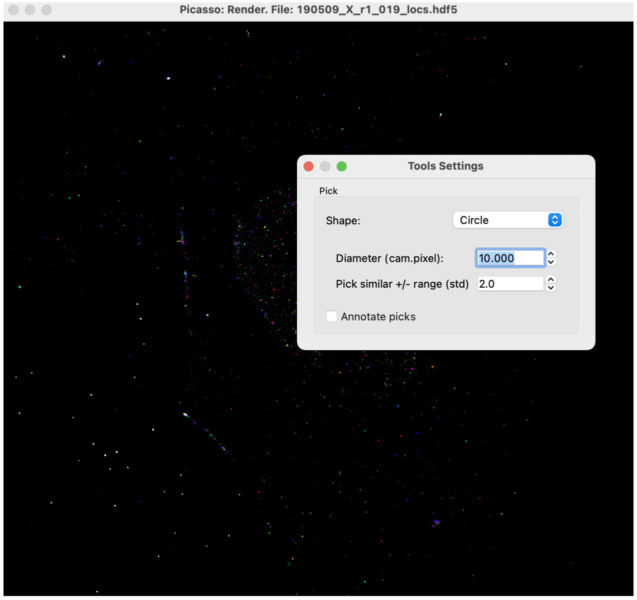
- Raw image file: 190509_X_r1_021.nd2
  - The file is a concatenation of multiple z stacks:
    - Frames 1–100: z = 0 nm (glass surface)
    - Frames 101–10100: z = 0 nm
    - Frames 10101–10200: z = 0 nm
    - Frames 10201–20200: z = 170 nm
    - Frames 20201–20300: z = 0 nm
    - Frames 20301–30300: z = 340 nm
    - Frames 30301–30400: z = 0 nm
    - Frames 30401–40400: z = 510 nm
    - …
    - Frames 90901–91000: z = 0 nm
    - Frames 91001–101000: z = 1530 nm
  - The file needs to be split into z stack series for localization with Picasso Localize

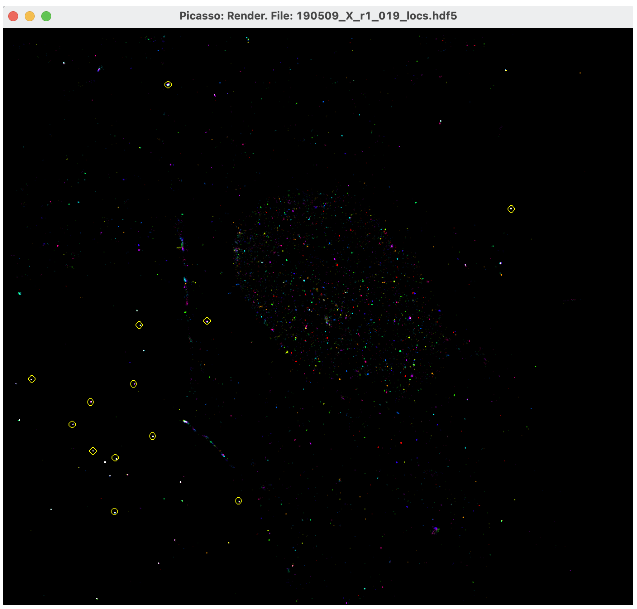
1. Process raw images with Picasso Localize to localize single molecules
   - Box Size: 17
   - Convergence Criterion: 0

- Max. Iterations: 0
- Min. Net Gradient: 7500
- PSF file for astigmatism 3D analysis: 190509_psf.yaml

- Output files: ./locsfile/raw_locs/190509_X_r1_0##_locs.hdf5
  - 190509_X_r1_001, 003, 005, … : Imaging fiducial markers on the surface for further registration
  - 190509_X_r1_002, 004, 006, … : Actual localization data in 170 nm steps

2. Register the surface fiducial marker localization files (190509_X_r1_001, 003, 005, …_locs.hdf5) to generate a text file with drift registration coordinates
   - Open all surface fiducial marker localization files 190509_X_r1_001, 003, 005, …_locs.hdf5 in Picasso Render
   - 
   - Apply *Postprocess>Align channels (RCC or from picked)* without any picks
   - 
   - Manually pick ~10 fiducial markers in the image using *Tools>Pick*
   - (Optional:) Use *Tools>Pick similar* to pick more fiducials automatically if there are many

- 

- 

- Apply *Postprocess>Align channels (RCC or from picked)* with picks
- Copy the last three lines, which are x, y, and z drift against the first file

```
Shift (array([ 0.        , -0.01879561, -0.00735031, -0.02074205, -0.04966956,
        0.05836098, -0.01163047, -0.00270014, -0.05915131,  0.06729843]), array([ 0.        , -
0.10741499, -0.07970728, -0.03024503, -0.09572253,
       -0.14047242, -0.17934642, -0.15648159, -0.15292683, -0.20726581]), array([   0.        ,
  -55.18399124,  -69.64136286,  -93.7948873 ,
        -98.1653928 , -114.03902056, -110.25880659,  -98.2442982 ,
        -79.40068285,  -81.789624  ]))
0.0,-0.018795609808876197,-0.007350308811874168,-0.020742053899448373,-0.049669561767950635,0.0
5836098312865948,-0.011630471143871543,-0.0027001443435437644,-0.0591513099847361,0.06729842601
34399,

0.0,-0.10741499299183485,-0.07970728334039448,-0.030245030298828 99,-0.09572253413498 4,-0.140472
4171385168,-0.17934641512110827,-0.15648158844560384,-0.15292683066800233,-0.207265807967 6329,

0.0,-55.18399124145506,-69.64136285781862,-93.79488730430602,-98.16539279818534,-114.0390205621
7191,-110.25880658626555,-98.24429820179941,-79.4006828546524,-81.78962399959563,
```

- Create a text file with the following lines with comma (,) delimitor: 190509_r1_drift.txt

        - Line1, file names to be undrifted
        - Lines 2–4, the three lines above: x, y, and z shift
    - 190509_r1_drift.txt

```
1 190509_X_r1_002_locs,190509_X_r1_004_locs,190509_X_r1_006_locs,190509_X_r1_008_locs,190509_X_r1_010_locs,190509_X_r1_012_locs,190509_X_r1_014_locs,190509_X_r1_016_locs,190509_X_r1_018_locs,190509_X_r1_020_locs
2 0.0,-2.65508419573,-4.96590882987,-3.72427673992,-5.11782836467,-3.37165220082,-2.4585662514,-1.86706850529,-2.83761903644,-3.69600827936
3 0.0,12.6123923302,16.9455328703,20.6623250723,23.9738042831,28.0911989331,29.6879745632,31.1352251887,29.3304789171,29.094566606
4 0.0,-41.8267650604,-51.8772878647,-81.8539729595,-91.797389257,-108.208583069,-90.1446902633,-90.8802564859,-70.562548542,-75.8468509197
```

3. Undrift localization files (190509_X_r1_002, 004, 006… _locs.hdf5) using **undrift.py**
    - Copy 190509_r1_drift.txt to the folder containing data to be undrifted: ./locsfile/raw_locs/
    - Run **undrift.py**
    - `> python undrift.py -f ./sample_9col/locfiles/raw_locs/190509_r1_drift.txt`
    - Output files: ./locsfile/undrifted/190509_X_r1_0##_locs_undrifted.hdf5

4. Further undrift each localization file using Picasso Render built-in function (*Postprocess>Undrift by RCC* and or *Postprocess>Undrift from picked*)
    - Output files: ./locsfile/aligned/190509_X_r1_0##_locs_undrifted_render.hdf5

5. Merge all files into one single 3D localization file using **concatenate.py**
    - Run **concatenate.py**
    - `> python concatenate.py -d ./sample_9col/locfiles/aligned/ -s 170 -o 190509_X_r1_merged`
    - Output file: 190509_X_r1_merged.hdf5

6. Link localizations using Picasso Render built-in function (Postprocess>Link localizations)
    - Output file: 190509_X_r1_linked.hdf5

7. Crop the localization data to 256 x 256 px using **crop.py**
    - Run **crop.py**
    - `> python crop.py -f ./sample_9col/locfiles/190509_X_r1_linked.hdf5 -x 230 -y 220 -s 256`
    - Output file: 190509_X_r1_cropped.hdf5

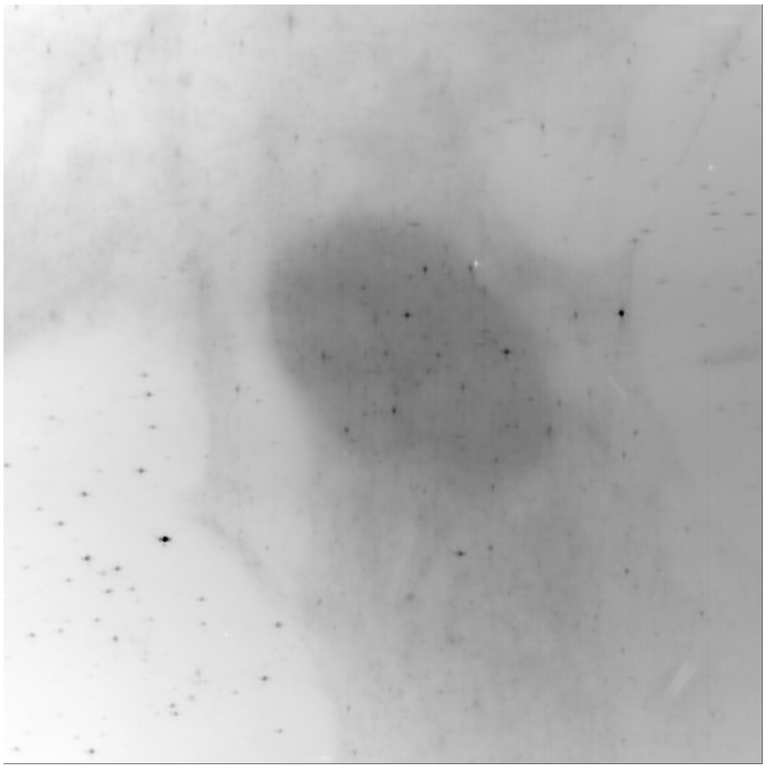
8. Process localization data with HDBSCAN using **hdbscanlocs.py**
    - Run **hdbscanlocs.py**
        - Set HDBSCAN parameter 'minimum cluster size' to 30
    - `> python hdbscanlocs.py -f ./sample_9col/190509_X_r1_cropped.hdf5 -c 30`
    - Output file: ./hdbscan/hdbscan_30_None_0/190509_X_r1_hdbscan_30_None_0.hdf5

Step 2. Process SABER diffraction limited image

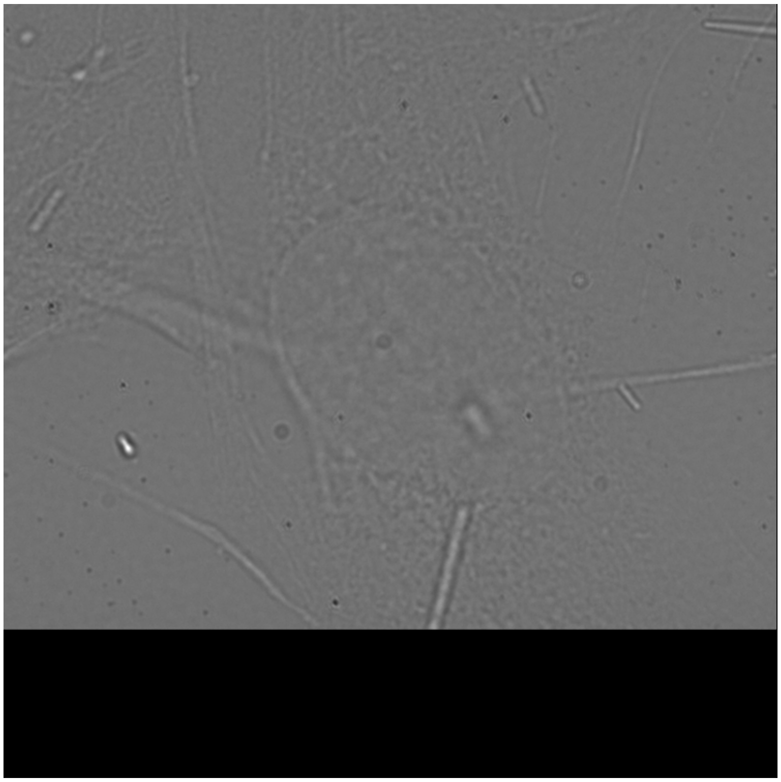
- Raw image file: ./diff/190509_saber1_r1.tif, 190509_saber2_r1.tif, 190509_saber3_r1.tif

- 5 channels: 488, 565, 647, DAPI, and bright field

- Registration reference file: 190509_X_r1_021.nd2, Frame 1–100 (the first set of z = 0 nm (glass surface) image)

1. Generate a reference image for registration
    - Open ./diff/190509_X_r1_021.nd2, Frame 1–100 with ImageJ/Fiji
    - Generate the average projection
    - Invert the intensity
    - Output file: ./diff/AVG_190509_X_r1_021.tif
    - 
    - The black spots outside of the cell are fiducial gold nanoparticles

2. Register SABER images to localization data based on fiducial markers visible in the bright field images
    - Open ./diff/190509_saber1_r1, r2, r3.tif and find the surface plane in the bright field channel

- 
- Find a few gold nanoparticles observed in both bright field and registration reference images
- Register the SABER images based on fiducial markers manually or using Fiji Plugin
- Generate max Z projections
- Merge all SABER images into one single file: ./diff/190509_X_r1_saber.tif

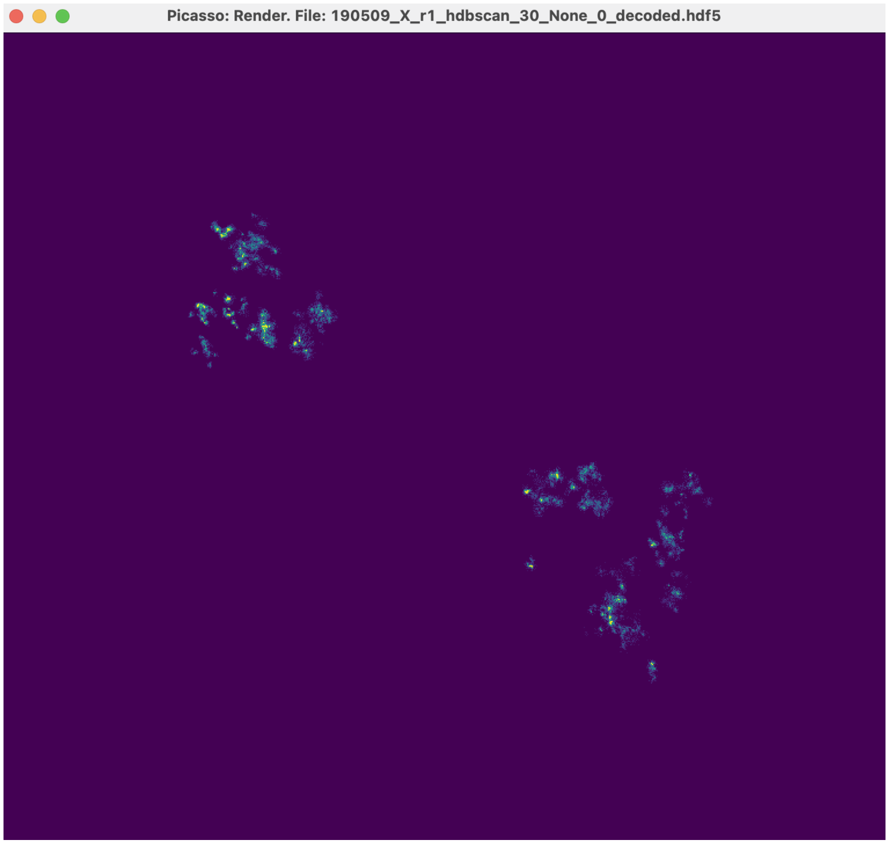
3. Generate a SABER binary mask file
    - Crop 190509_X_r1_saber.tif to align it to the localization file (256 x 256 px, origin = (220, 230))
    - Apply Fiji *Image>Adjust>Threshold>Yen*
    - Output file: ./diff/190509_X_r1_saber_mask.tif

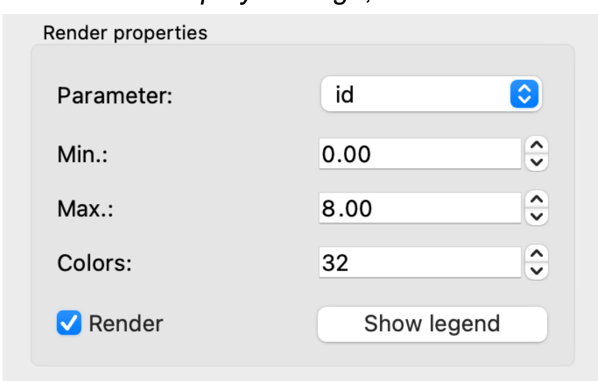
Step 3. Decode localizations

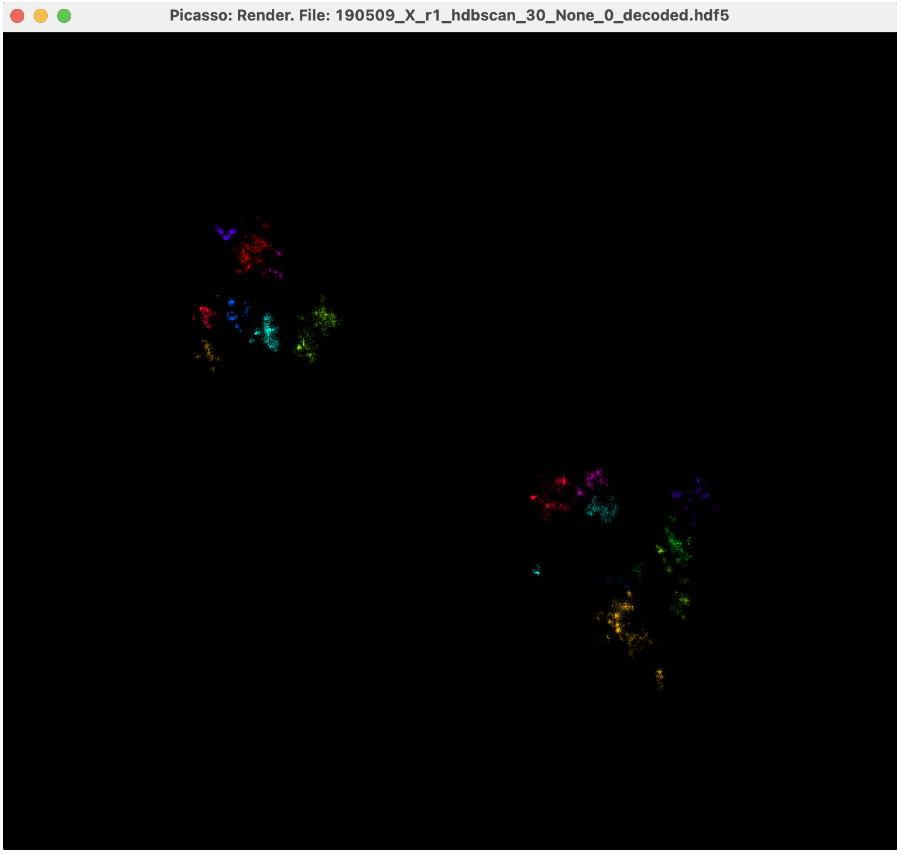
1. Run **decode.py**
    - Input files:
        - Localization: ./hdbscan/hdbscan_30_None_0/190509_X_r1_hdbscan_30_None_0.hdf5
        - SABER mask: ./diff/190509_X_r1_saber_mask.tif

```
python decode.py -f
./sample_9col/hdbscan/hdbscan_30_None_0/190509_X_r1_hdbscan_30_None_0.hdf5 -m
./sample_9col/diff/190509_X_r1_saber_mask.tif
```

- Output file: ./hdbscan/hdbscan_30_None_0/decoded/190509_X_r1_hdbscan_30_None_0_decoded.hdf5

2. Open the decoded file in Picasso Render
   - 

   - Go to *View>Display settings*, set Parameter to ‘id’, and check Render
   - 

- 
Picasso: Render. File: 190509_X_r1_hdbscan_30_None_0_decoded.hdf5