Using Pixylator (1.0 alpha)

K. Sehara

Contents

1 What is Pixylator?

In short terms, Pixylator is a hue-based object categorization system that is implemented as an ImageJ plugin. The name “Pixylator” derives from the Pixy camera (`http://charmedlabs.com/default/pixy-cmucam5/`), which reports the positions of color-coded objects at a rate of 60 samples per second (SPS). The idea and algorithm used in Pixylator plugin originate from those of Pixy camera.

The basic procedures are as follows:

1. Take movies using your favorite camera.
2. Open the movie in ImageJ (possibly convert it beforehand).
3. Run Pixylator on the movie.

2 Installation

Once you downloaded the Pixylator file and extract it (in case it is a zip file), it should be a directory containing lots of files.

1. Place the directory inside “plugins” directory of ImageJ (it should be in the same directory as in the ImageJ app itself).
2. Restart ImageJ.
3. Open any color image file on ImageJ.
4. Find the submenu “Pixylator” in the “Plugins” menu (the items should be in an alphabetical order), and select “Pixylator alpha” from it.
5. If you see the main window of the plugin, the installation is successful.

In cases either 1) you cannot recognize the “Pixylator” submenu in the “Plugins” menu, or 2) you encounter (through the process) a log output complaining about something, then it is likely the plugin is compiled in a different version of ImageJ. Then:

1. Select menu “Plugins” →“Compile and Run...”.
2. Find and select “Pixylator_alpha.java” in the “Pixylator” directory, which you have just moved in the “plugins” directory.
3. See if the main window of the plugin pops up.

Otherwise, please ask a person nearby that seems to know ImageJ plugins.

3 Basics

After opening your movie, open “Pixylator” plugin. There, you can configure the tracking.

3.1 Concepts and limitations

3.1.1 Hue-based object tracking

Pixylator (just like Pixy) detects pixels that have the specified range of **hues** (for the description of hue, refer to Wikipedia: `https://en.wikipedia.org/wiki/Hue`). It is believed to be less sensitive to the change in lightings than the brightness or the color itself.

Pixylator works as follows for each of the frames in the movie:

1. Calculate the hue for each of the pixels.
2. Collect the pixels that are within the specified range.
3. Generate the output, and go on to the next frame.

3.1.2 One object per one hue range

Note that there is an important difference between Pixy and Pixylator. Whereas the former separates individual objects with the same hue range after step 2, the latter reports the set of pixels as one single "set". In other words, for Pixylator there should be at maximum one object within one hue range.

3.1.3 Only two classes are available

Also, there are only two entries for hue ranges for the moment. This is just the simplicity and the utility reasons. The available number of entries may increase in the future.

3.2 Typical procedures

Note that the plugin is still in the alpha phase, and it can be buggy. It is **highly recommended to generate/save "mask images" (at least for some of the files; see section 4.5) and visually verify that the objects be correctly identified, and that object locations be as expected**.

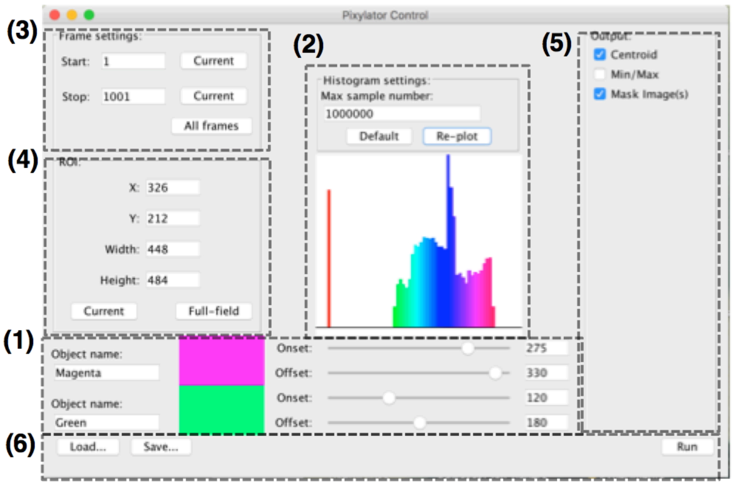
1. Open the image/movie file on ImageJ.
2. Open Pixylator plugin.
3. *(Optional)* Load existing configuration, if any.
4. Configure frames to track.
5. Configure the ROI to track.
6. *(Optional)* Re-plot the histogram and re-configure hue ranges.
7. *(Optional)* Rename the object names as you like.
8. Configure the output to generate.

9. *(Optional)* Save the current configuration.
10. Run tracking.
11. Inspect/save the results.
12. If you have another image/movie to process, open it and go on.

You can find how to do each step for the rest of this document.

4 Overview of the plugin GUI

Below is the main panel for the plugin.



Frame settings:
Start: 1
Stop: 1001

Histogram settings:
Max sample number:
1000000

ROI:
X: 326
Y: 212
Width: 448
Height: 484

Output:
☑ Centroid
☐ Min/Max
☑ Mask Image(s)

Object name:
Magenta
Onset: 275
Offset: 330

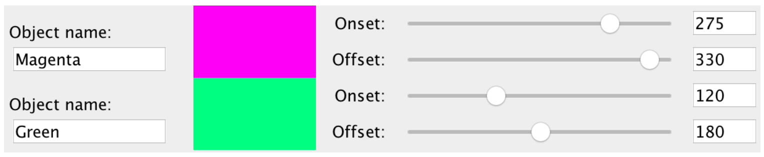
Object name:
Green
Onset: 120
Offset: 180

Several components can be seen.

1. **Mask control**: configuration for hue ranges to track.
2. **Histogram**: a hue histogram of the movie file.
3. **Frame settings**: configuration of which frames are to be used.
4. **ROI settings**: configuration of the region to be used.
5. **Output settings**: configuration of what output to generate.
6. **Button panel**: a set of actions for the plugin.

4.1 Mask Control

This is the core configuration of Pixylator. The Mask control interface is found near the bottom of the plugin.


Object name: Magenta — Onset: 275 — Offset: 330

Object name: Green — Onset: 120 — Offset: 180

On the leftmost of the Mask Control interface, you can find text fields labeled as “Object name: ”. Here you can set the name of this object category to something more understandable to you.

Object names are used as the column names when the output (section 4.5) is generated.

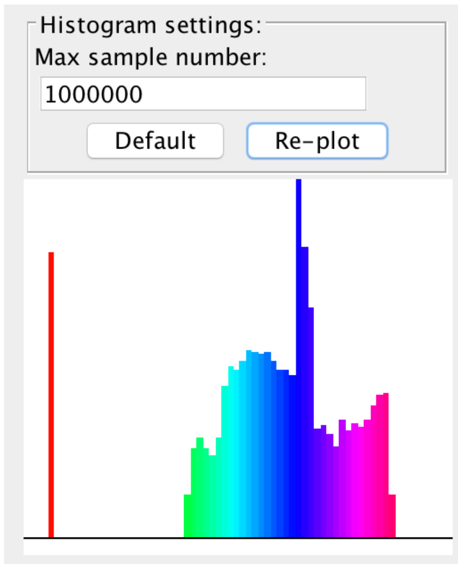
Next to the object names, you can see colored rectangles. This is the representative color for the hue range at the moment. As you change the hue range (see below), the color changes accordingly.

Note that the representative colors are used to draw masks when ”mask images” are generated for the output (section 4.5).

The sliders and text fields that lie on the center of the window is for control of hue ranges. For each hue range, hue values larger than “onset” and smaller than “offset” are picked up. You can refer to the histogram (section 4.2) above for approximate color of the hue.

4.2 Histogram

Once you open the plugin, Pixylator tries to plot the hue histogram of the current image (i.e. the frontmost image that is open). It may take a while the process is done.



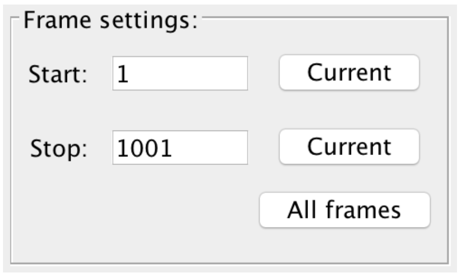
By having a hue histogram, you can get a hint on what hue range to pick up as the object (note that the hue of the object often shows up as a mode in the histogram). The hue value is aligned with the slider for Mask control (section 4.1), so as to be easy to reflect the visual inspection results on the mask control.

The panel on the top of the histogram is the control for plotting. Because building a histogram is a heavy load for the plugin, it offers a way to perform sub-sampling of the image/movie file in a way the number of sampling not exceed "Max sample number". The number goes back to the default value when you click on the "Default" button.

You can change the range of pixels collected for generating histogram using Frame (section 4.3) or ROI (section 4.4) settings as well. In any case, make sure to click on "Re-plot" button to update the histogram.

4.3 Frame Settings

On the top-left of the window, you can specify the controls for restricting frames used for tracking.


Frame settings:

Start: 1

Stop: 1001

There are mainly two inputs: "Start" and "Stop". Here, "Start" denotes for the first frame included for tracking, and "Stop" for the last frame included for tracking. The frame number starts from 1. Thus, if you want to track 10 frames after the first 20 frames, then "Start" is 21 and "Stop" is 30.

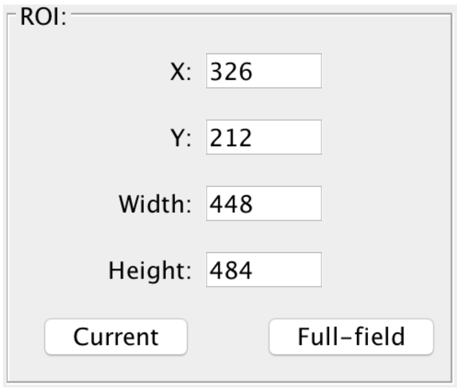
Note that the selection of frames can affect histogram (section 4.2) generation as well. If you press "Re-plot" button after changing Frame settings, the resulting histogram will only include pixels from the specified frames.

If you click on "Current" button associated with either of the inputs, the current frame (the one that is visible on the image window) is selected as the corresponding input.

A click on "All frames" let the plugin process all the images.

4.4 ROI Settings

On the middle-left of the window, you can find the ROI (for "region of interest") settings to specify the region used for tracking.


ROI:

X: 326

Y: 212

Width: 448

Height: 484

Restriction of ROI has both good and bad sides. Good things include 1) you can save time for computation, and 2) the data is less likely to get contaminated from unwanted object sources. On the other hand, arbitrary ROI selection may "cut" the object as it moves in the movie. **You need to be very careful that the ROI cover all of what you want.**

For the moment, the only way to specify the ROI is by using a rectangle: X and Y coordinates of the origin, the width, and the height.

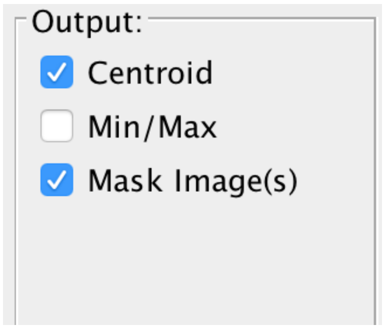
The coordinate system follows the convention used in ImageJ. If you place the cursor somewhere on the image window, X and Y coordinates will be displayed on the space under the ImageJ tool bar (where lots of tool icons lie). Further, if you use the rectangular selection tool and perform a drag on the image window, you are likely to see the "w" (stands for "width") and "h" (height) as well.

When you make a selection, and verified that the rectangle cover all the pixels of interest, you can click on "Current" button of ROI settings. Then the coordinate data is transferred into Pixylator.

The button "full-Field" sets the ROI back to the full field of view.

4.5 Output Settings

On the right-hand side of the window is Output settings.


Output:

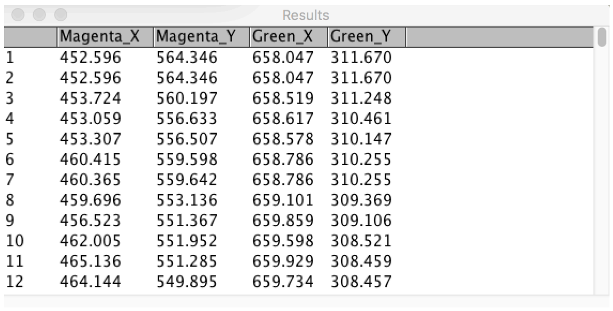
[✓] Centroid

[ ] Min/Max

[✓] Mask Image(s)

4.5.1 Parameter output

Enabling “Centroid” makes the plugin generate a table, and write centroid values for each of the object classes per each frame.

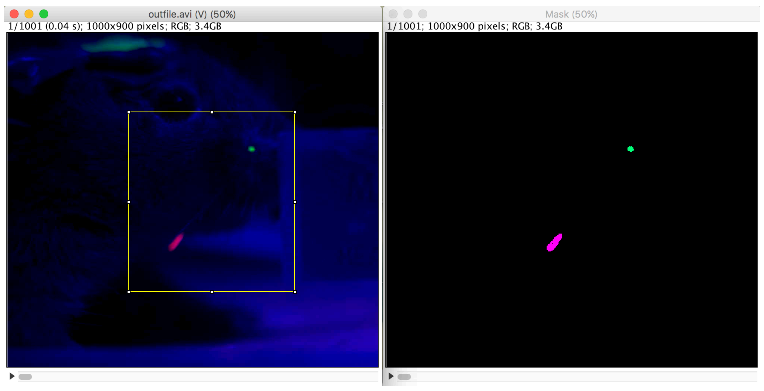


	Magenta_X	Magenta_Y	Green_X	Green_Y
1	452.596	564.346	658.047	311.670
2	452.596	564.346	658.047	311.670
3	453.724	560.197	658.519	311.248
4	453.059	556.633	658.617	310.461
5	453.307	556.507	658.578	310.147
6	460.415	559.598	658.786	310.255
7	460.365	559.642	658.786	310.255
8	459.696	553.136	659.101	309.369
9	456.523	551.367	659.859	309.106
10	462.005	551.952	659.598	308.521
11	465.136	551.285	659.929	308.459
12	464.144	549.895	659.734	308.457

Enabling “Min/Max”, on the other hand, writes the bounds of X- and Y-coordinates for each of the object classes. Both value types are written on a single Results window.

4.5.2 Mask output

When you enable “Mask Image(s)”, a whole new set of images are created. You can save the generated images in as many options as ImageJ provide.


outfile.avi (V) (50%)
1/1001 (0.04 s); 1000x900 pixels; RGB; 3.4GB
Mask (50%)
1/1001; 1000x900 pixels; RGB; 3.4GB

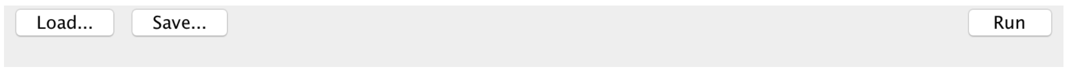
The highlighted area on the mask file represents the region (a set of pixels) whose hue value was within the specified range. The mask color is specific to the hue value range you used.

As you change the frames of the image set, you can see how Pixylator detected the objects in the specified frames.

Note that, even when you specified Pixylator to work within some small ROI (as in the case in the figure; ROI settings can be found at section 4.4), the mask image is created in the same size as the original movie. As such, the coordinate system is reconverted to the original one (in the parameter output as well).

4.6 Actions

On the panel at the bottom of the window, there are several actions that Pixylator can take.



- **"Run" button**: runs the tracking using the current configurations.
- **"Save..." button**: saves the current configurations to a file.
- **"Load..." button**: loads the configurations that are saved into a file before.

4.7 Troubleshooting

Some of the problems that one may frequently encounter.

4.7.1 ImageJ does not open the movie file

Today, many movie files (even AVI files) are encoded in codecs such as h264, and ImageJ cannot open them.

You need to first decode them to obtain "raw AVI files" for ImageJ to read them. Install and use tools such as **ffmpeg**(`https://ffmpeg.org/`). Make sure to specify "raw" AVI, and avoid any compression or encoding.

4.7.2 I don't know what hue range I should choose

This is the process that takes you most of the time. Below are some tips.

Start from simple Do not use 3600-frame movie first, but start from one representative single-frame (or up to 50 frames or so) image file. Alternatively, use Frame settings (section 4.3) to restrict the frames that you process.

Use standard colors In a single-frame image, specify the ROI that contains only the object (no background) (section 4.4), and generate histogram (section 4.2); by doing so several times, you will get an intuition about what hue value you are aiming for.

Verify what is obtained As you run several test runs using small-frame image samples, always generate "Mask images" (section 4.5), and compare the results with your visual inspection:

- Is there any extra detection other than what you target? If so, try restricting the hue range (section 4.1) and/or restricting the ROI (section 4.4).

- Is there enough pixels detected for tracking on each frame? If so, you may need to widen the hue range.
- Do the output centroid coordinates correspond nicely with the target object? There may be small "pixel islands" that you may miss to detect, and that can cause errors on object tracking.

4.7.3 Hue histogram does not show up (or appear) properly

- Check that the progress bar in the ImageJ toolbar is not working; if the number of pixels/frames is very large, drawing the histogram can take up to minutes. Change the "Max sample numbers" and re-plot in those cases (section 4.2).
- Try clicking the "Re-plot" button.
- Try resizing the plugin window (although it is unlikely, repainting of the histogram is somehow stuck).
- Check that the Frame/ROI settings are correct (sections 4.3 and 4.4). If you re-use the plugin in the same ImageJ session, it is possible that the previous configurations contaminate the working of histogram.

4.7.4 Things (histogram and/or tracking) are really slow

It can be the problem of the computer/OS, but assume here that it is the problem of the plugin. Here are what one can take:

- Make the ROI smaller (but without cutting the objects to be tracked; see section 4.4).
- Disable "Mask Image(s)" output (section 4.5).
- (For histograms only) reduce "Max sample number" (section 4.2).

4.7.5 Too few results are generated as the output

If you re-use the plugin in the same ImageJ session, it is possible that the previous configurations contaminate the working of tracking. Check that Frame/ROI settings are correct (see sections 4.3 and 4.4).

4.7.6 Cannot save the "Mask images"

If you deal with a movie that has many/large frames, it is likely that saving process hit the theoretical maximal size of a TIFF file (2 GB in most cases).

One of the best solutions is to split the image set into several stacks. You can do it by using "Image" →"Stacks" →"Tools" →"Make Substack..." (Alternative way is to use "Stk" button on the ImageJ toolbar, then select "Make Substack..." from it). By checking "Delete slices from original stack", you can successfully dissociate a set of frames from the rest. Then you can save the individual stacks as separate TIFF files.

In some cases, the split files can be merged back by compressing the frames (this is especially true for the case of Masked images). Use tools such as Libtiff. More specifically, a command called tiffcp (or TiffCP) takes care of both concatenating and compressing at once.

- **for Windows** Install LibTiff.Net (`https://bitmiracle.com/libtiff/`), and run TiffCP on the command prompt (cmd.exe).
- **for Mac** Retrieve the source file and build it yourself (follow e.g. `http://mac-dev-env.patrickbougie` Alternatively, you can install libtiff through package manager such as Homebrew (`https://brew.sh/`)
- **for Linux** In many cases libtiff tools are already installed (check with "which tiffcp"), but in case not, check for "libtiff" in the package manager.

4.7.7 ImageJ runs out of memory

Change the volume of memory for ImageJ and restart the app; increase the available amount of memory at "Edit" →"Options" →"Memory & Threads...".