Trackmate guide Documentation

Release 0.1.0

Open Microscopy Environment

May 10, 2023

Contents

TrackMate is Fiji plugin for single-particle tracking. We demonstrate how to use TrackMate and OMERO using both the User Interface and the API.

CHAPTER 1

Contents

1.1 Analyze OMERO timelapse images using the TrackMate User Interface

In this example we open in Fiji an image stored in an OMERO server and use TrackMate to analyze it.

1.1.1 Description

In this section, we show how to use TrackMate via its User Interface. The manual steps are essential to determine the suitable parameters to analyze the images. Note that when using the TrackMate User Interface, the generated tracks **cannot** be saved as ROIs into OMERO.server.

1.1.2 Setup

- Install Fiji on the local machine with the OMERO.insight-ij plugin. The installation instructions can be found in the Fiji guide.

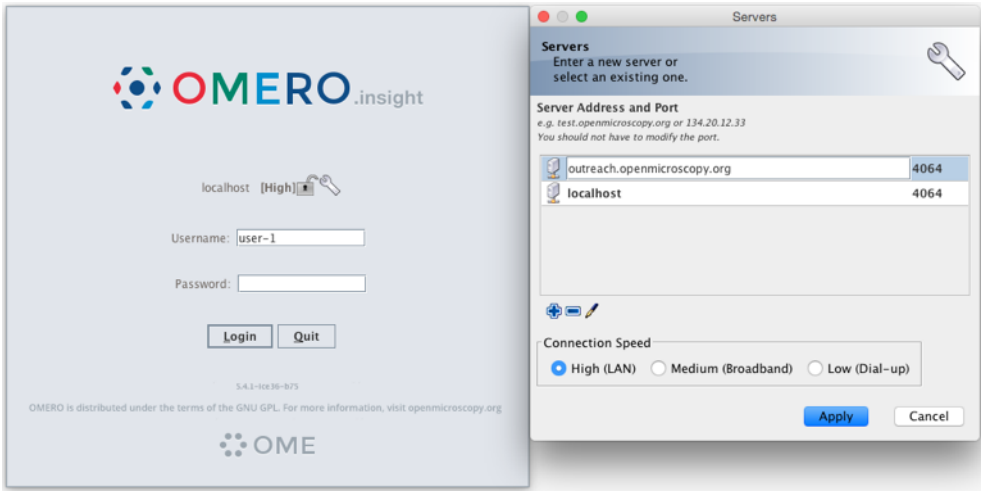
1.1.3 Resources

- We use an artificial track image https://samples.fiji.sc/FakeTracks.tif.

1.1.4 Step-by-Step

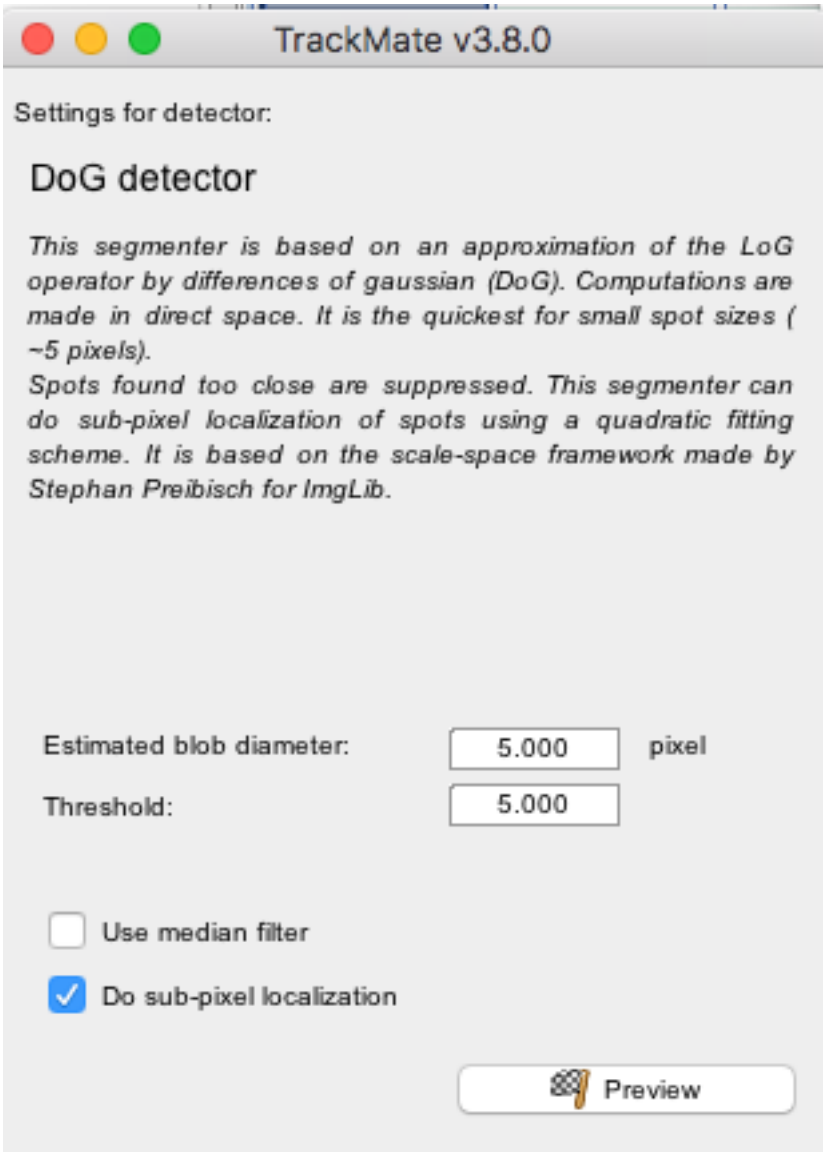
1. Launch Fiji.
2. Go to *Plugins* > *OMERO* > *Connect* To OMERO. This will show a login screen where you can enter the name of the server to connect to, the username and password. The OMERO plugin will allow you to browse your data in a similar manner to OMERO.web.

3. In the OMERO login dialog, click the wrench icon  and then add the server address in the dialog. By default, only "localhost" is listed. Click on the *plus* icon to add a new line to the list and type into the line the server address.
4. Click *Apply*.

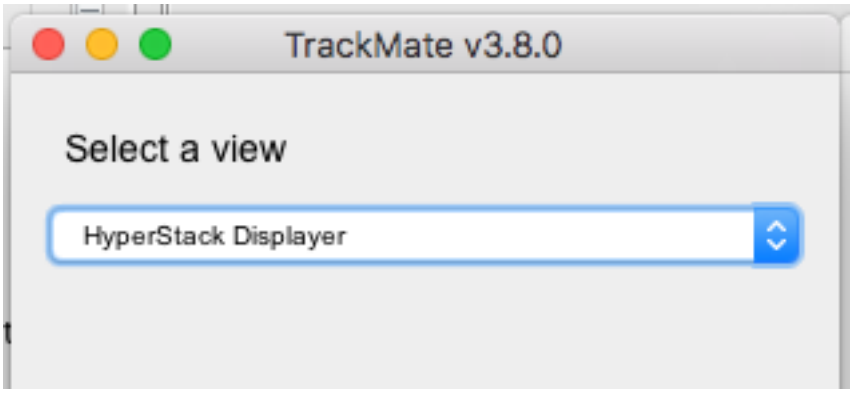


5. Enter your credentials and click *Login*.
6. Browse the Dataset where your image to be tracked is located, e.g. *artificial-trackmate*.
7. Double-click on the *FakeTracks.tif* Image to open it in Fiji.
    1. Make sure it is opened using the `Hyperstack` viewer. This option depends on what you did last time when using your Fiji.
    2. In case you are not sure, Open `Plugins > Bio-Formats > Bio-Formats importer` and then select any image on your local computer to open it in Fiji. In the popup dialog, in the top left dropdown menu.

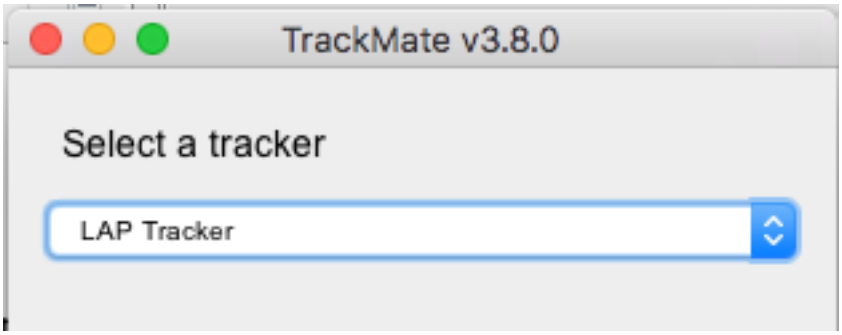
    3. Select `Hyperstack` and click `OK`. Then, close and re-open the *FakeTracks.tif* Image from OMERO.
8. Go to `Plugins > Tracking > TrackMate`.
9. Click `Next` in the first dialog that pops up.
10. A new dialog pops up indicating to select a detector. Select the `DoG detector`. Click `Next`.
    1. Set the `Estimated blob diameter` to `5.0`
    2. Set the `Threshold` to `5.0`



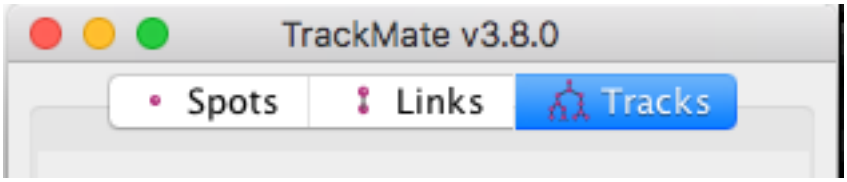
11. Click the `Preview` button to find spots. 4 spots should be found.
12. Click the `Next` button several times until you get to the `Select a view` dialog.
13. Select the `HyperStack Displayer view`



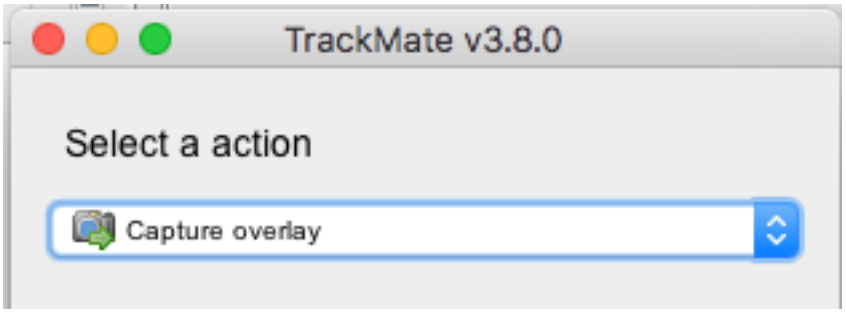
14. Click the `Next` button twice until you get to the `Select a tracker` window. Select the `LAP Tracker`.



15. Click the `Next` button several times until a dialog with three tabs pops up
16. Select the `Tracks` tab to display the `Tracks`.



17. Click `Next` until you get to `Select an action` dialog.
    1. Select the option Capture overlay
    2. Click Execute.



18. Click `OK` in the following dialog, leaving the defaults (the whole stack will be captured).
19. New image appears. Select it. This new Image can then be imported as an OME-TIFF using `Plugins > OMERO > Save Image(s) to OMERO`. The tracks will be part of the generated OME-TIFF and the time-points will be captured as z-sections.

1.2 Analyze OMERO timelapse images using the TrackMate API

In this example we open an image stored in an OMERO server and use the TrackMate API to analyze it. One of the advantages of this approach over the User interface workflow is that the generated track **can** be saved as OMERO ROIs.

1.2.1 Description

First, we will show how to use the TrackMate API and the Scripting editor of Fiji.

We will show:

- How to connect to OMERO using the JAVA API.
- How to retrieve an image.
- How to open the image using Bio-Formats.
- How to create a TrackMate model using its API.
- How to save the tracks as polygon ROIs in OMERO.

1.2.2 Setup

- Install Fiji on the local machine with the OMERO.insight-ij plugin, version 5.5.10 or higher. The installation instructions can be found at here.

1.2.3 Resources

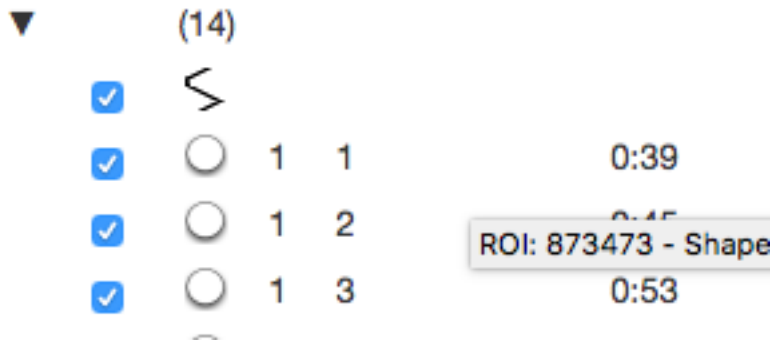
- We will use a timelapse image available at DV/iain/438CTR
- Script: Groovy script for tracking timelapse images - `tracking.groovy`.

1.2.4 Step-by-Step

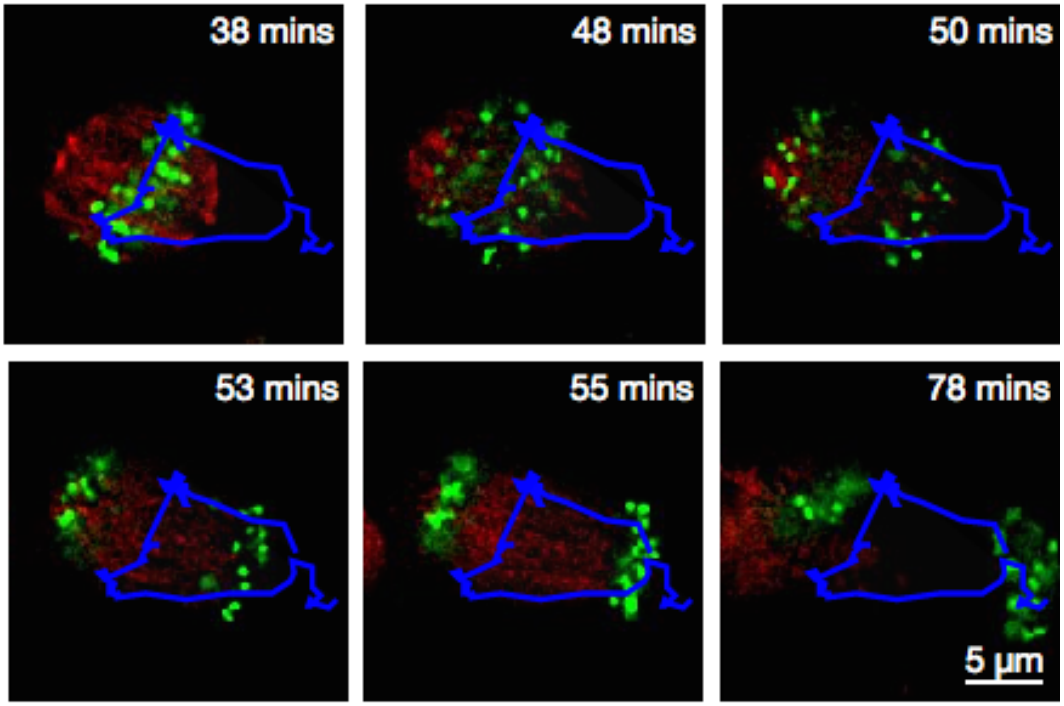
In this section, we go through the steps required to analyze the data. The script used in this document is `tracking.groovy`. The script follows similar steps than the ones used via the User Interface, see *Analyze OMERO timelapse images using the TrackMate User Interface*.

One advantage of the scripting approach is that we can save the generated tracks as ROIs in an OMERO server.

1. Launch Fiji.
2. Open `File > New > Script...`.
3. Select `Groovy` as a language.
4. Copy the content of `tracking.groovy` in the text area.
5. A dialog will pop up. Enter the credentials to connect to the server and select an Image. If you are not using websockets i.e. no `wss` in front of the host name, the port value needs to be changed to `4064`.
6. Click Run.
7. Go back to OMERO.web to visualize the tracks. Double-click on the image in OMERO.web to open it in OMERO.iviewer.
8. Click on the ROI tab and observe that you now have ROIs under which there are Shapes. Each ROI is a collection of shapes. The ROI corresponds to a track in Trackmate. There is always one polyline shape in each ROI which represents the track. The other, elliptical shapes in the same ROI represent the tracked spots.



9. Play the timelapse video in OMERO.iviewer.
10. Go to the Info tab, and in the Open with: line click on OMERO.figure. In OMERO.figure, add the Tracks and ellipses to the panel by selecting the appropriate ROIs in the Labels tab of OMERO.figure.



Script's description

Import packages needed:

```
import java.awt.Color
import java.util.ArrayList

import omero.gateway.Gateway
import omero.gateway.LoginCredentials
import omero.gateway.SecurityContext
import omero.gateway.facility.BrowseFacility
import omero.gateway.facility.ROIFacility
import omero.gateway.model.EllipseData
import omero.gateway.model.PolylineData
```

(continues on next page)

(continued from previous page)

```
import omero.gateway.model.ROIData
import omero.log.SimpleLogger
import omero.model.PolylineI
import static omero.rtypes.rstring

import ij.IJ

import fiji.plugin.trackmate.Spot
import fiji.plugin.trackmate.Settings
import fiji.plugin.trackmate.Model
import fiji.plugin.trackmate.SelectionModel
import fiji.plugin.trackmate.TrackMate
import fiji.plugin.trackmate.detection.DetectorKeys
import fiji.plugin.trackmate.detection.DogDetectorFactory
import fiji.plugin.trackmate.tracking.sparselap.SparseLAPTrackerFactory
import fiji.plugin.trackmate.tracking.LAPUtils
import fiji.plugin.trackmate.visualization.hyperstack.HyperStackDisplayer
import fiji.plugin.trackmate.features.spot.SpotContrastAndSNRAnalyzerFactory
import fiji.plugin.trackmate.features.spot.SpotIntensityAnalyzerFactory
import fiji.plugin.trackmate.features.track.TrackSpeedStatisticsAnalyzer
```

Connect to the server. It is also important to close the connection again to clear up potential resources held on the server. This is done in the `disconnect` method:

```
def connect_to_omero() {
    "Connect to OMERO"

    credentials = new LoginCredentials()
    credentials.getServer().setHostname(HOST)
    credentials.getUser().setUsername(USERNAME.trim())
    credentials.getUser().setPassword(PASSWORD.trim())
    simpleLogger = new SimpleLogger()
    gateway = new Gateway(simpleLogger)
    gateway.connect(credentials)
    return gateway
}

def disconnect(gateway) {
    gateway.disconnect()
}
```

Load the image from the server:

```
def get_image(gateway, image_id) {
    "Retrieve the image"
    browse = gateway.getFacility(BrowseFacility)
    user = gateway.getLoggedInUser()
    ctx = new SecurityContext(user.getGroupId())
    return browse.getImage(ctx, image_id)
}
```

Read the binary data using Bio-Formats:

```
def open_image_plus(host, port, username, password, group_id, image_id) {
    "Open the image using the Bio-Formats Importer"

    StringBuilder options = new StringBuilder()
    options.append("location=[OMERO] open=[omero:server=")
    options.append(host)
    options.append("\nuser=")
    options.append(username.trim())
    options.append("\nport=")
    options.append(port)
    options.append("\npass=")
    options.append(password.trim())
    options.append("\ngroupID=")
    options.append(group_id)
    options.append("\niid=")
    options.append(image_id)
    options.append("] ")
    options.append("windowless=true view=Hyperstack ")
    IJ.runPlugIn("loci.plugins.LociImporter", options.toString())
}
```

Create a tracking model using the TrackMate API:

```
def create_tracker(imp) {
    "Create the trackmate model for the specified ImagePlus object"
    // Instantiate model object
    model = new Model()

    // Prepare settings object
    settings = new Settings()
    settings.setFrom(imp)
    // Configure detector
    settings.detectorFactory = new DogDetectorFactory()
    settings.detectorSettings.put(DetectorKeys.KEY_DO_SUBPIXEL_LOCALIZATION, true)
    settings.detectorSettings.put(DetectorKeys.KEY_RADIUS, new Double(2.5))
    settings.detectorSettings.put(DetectorKeys.KEY_TARGET_CHANNEL, 1)
    settings.detectorSettings.put(DetectorKeys.KEY_THRESHOLD, new Double(5.0))
    settings.detectorSettings.put(DetectorKeys.KEY_DO_MEDIAN_FILTERING, false)
    // Configure tracker
    settings.trackerFactory = new SparseLAPTrackerFactory()
    settings.trackerSettings = LAPUtils.getDefaultLAPSettingsMap()
    settings.trackerSettings['LINKING_MAX_DISTANCE'] = new Double(10.0)
    settings.trackerSettings['GAP_CLOSING_MAX_DISTANCE'] = new Double(10.0)
    settings.trackerSettings['MAX_FRAME_GAP'] = 3

    // Add the analyzers for some spot features
    settings.addSpotAnalyzerFactory(new SpotIntensityAnalyzerFactory())
    settings.addSpotAnalyzerFactory(new SpotContrastAndSNRAnalyzerFactory())

    // Add an analyzer for some track features, such as the track mean speed.
    settings.addTrackAnalyzer(new TrackSpeedStatisticsAnalyzer())
    settings.initialSpotFilterValue = 1

    // Instantiate trackmate
    trackmate = new TrackMate(model, settings)
    ok = trackmate.checkInput()
```

(continues on next page)

(continued from previous page)

```
    if (!ok) {
        print(str(trackmate.getErrorMessage()))
        return null
    }

    ok = trackmate.process()
    if (!ok) {
        print(str(trackmate.getErrorMessage()))
        return null
    }
    // Display the results on top of the image
    selectionModel = new SelectionModel(model)
    displayer = new HyperStackDisplayer(model, selectionModel, imp)
    displayer.render()
    displayer.refresh()
    // The feature model, that stores edge and track features.
    fm = model.getFeatureModel()
    space_units = model.getSpaceUnits()
    time_units = model.getTimeUnits()
    return model
}
```

Convert the tracks into OMERO ROIs:

```
def convert_tracks(model, dx, dy) {
    "Convert the tracks into OMERO objects"
    rois = new ArrayList()
    tracks = model.getTrackModel().trackIDs(true)
    tracks.each() { track_id ->
        track = model.getTrackModel().trackSpots(track_id)
        roi = new ROIData()
        rois.add(roi)
        points = ""
        track.each() { spot ->
            sid = spot.ID()
            // Fetch spot features directly from spot.
            x = spot.getFeature('POSITION_X')/dx
            y = spot.getFeature('POSITION_Y')/dy
            r = spot.getFeature('RADIUS')
            z = spot.getFeature('POSITION_Z')
            t = spot.getFeature('FRAME')
            // Save spot as Point in OMERO
            ellipse = new EllipseData(x, y, r, r)
            ellipse.setZ((int) z)
            ellipse.setT((int) t)
            // set trackmate track ID and spot ID for later
            ellipse.setText(track_id+':'+sid)
            // set a default color
            settings = ellipse.getShapeSettings()
            settings.setStroke(Color.RED)
            roi.addShapeData(ellipse)
            points = points + x + ',' + y + ' '
        }
        // Save the track
        points = points.trim()
```

(continues on next page)

(continued from previous page)

```
        polyline = new PolylineI()
        polyline.setPoints(rstring(points))
        pl = new PolylineData(polyline)
        // set a default color
        settings = pl.getShapeSettings()
        settings.setStroke(Color.YELLOW)
        roi.addShapeData(pl)
    }
    return rois
}
```

Save the converted tracks back to the OMERO server:

```
def save_rois(gateway, rois, image_id) {
    roi_facility = gateway.getFacility(ROIFacility)
    user = gateway.getLoggedInUser()
    ctx = new SecurityContext(user.getGroupId())
    results = roi_facility.saveROIs(ctx, image_id, user.getId(), rois)
}
```

In order the use the methods implemented above in a proper standalone script, **Wrap it all up**:

```
gateway = connect_to_omero()
exp = gateway.getLoggedInUser()
group_id = exp.getGroupId()

image = get_image(gateway, image_id)
open_image_plus(HOST, PORT, USERNAME, PASSWORD, group_id, image_id)
imp = IJ.getImage()
dx = imp.getCalibration().pixelWidth
dy = imp.getCalibration().pixelHeight
trackmate_model = create_tracker(imp)
if (trackmate_model == null) {
        print("unable to create the trackmate model")
} else {
        omero_rois = convert_tracks(trackmate_model, dx, dy)
    save_rois(gateway, omero_rois, image_id)
    print("done")
}
disconnect(gateway)
```

CHAPTER 2

Contribute

Changes to the documentation or the materials should be made directly in the omero-guide-trackmate repository.