

Sapphire アプリマニュアル

はじめに

遺伝子研究においては、世代交代に要する時間や飼育の容易さ、遺伝子情報の既知性などから、ショウジョウバエがモデル生物として頻繁に用いられています。主に、導入した遺伝子やその操作の影響が、ショウジョウバエ各個体のライフイベント（孵化・蛹化・羽化・生命活動停止、など）に及ぼす影響が調べられていますが、多個体同時飼育されたショウジョウバエ集団におけるライフイベント転換点を、個体ごとに検出するには、現在目視による作業となっており、これには膨大な人的・時間的コストを伴います。

そのため、多個体同時飼育下におけるショウジョウバエ集団のタイムラプス画像から、各個体ごとの全ライフステージ転換点を自動検出するアルゴリズムを提供し、その結果を可視化するとともに、ユーザーフレンドリーな可視化カスタマイズや、精度のチューニングを提供するソフトウェアの開発が待望されています。

本ドキュメントは、それらを実現した「ショウジョウバエ全ライフステージ個体別自動判定システム」(Drosophila Individual Activity Monitoring and Detection System; DIAMonDs)のソフトウェア部分に関するマニュアルです。

DIAMonDsは、撮像のためのハードウェアを含めたシステム全体の呼称であり、ライフイベント検出のためのアルゴリズムとそれを実装したソフトウェア部分が「Sapphire」という名称となっています。

Sapphireは、ショウジョウバエを用いた遺伝子研究現場において使用することを想定し開発されたものであり、「Dash」というフレームワークで動作しています。ブラウザベースのため動作が軽く、幅広い環境で使用できるというのが特徴です。

以下に、本ソフトウェアの使用方法を記します。

推奨環境

Ubuntu 16.04

Google chrome（ウェブブラウザ）

Python 本体

- Python (3.6.2)

導入手順

- Python 環境の構築
 - Python のインストール
 - 仮想環境作成
 - 必要なモジュールのインストール
- データの準備
 - オリジナル画像
 - マスクファイルの作成
 - config.json の作成
- 推論作業
 - inference.py を使って推論
 - make_CF_signal.py を使ってシグナルを作成
- Sapphire の起動

Python 環境の構築

Sapphire は Python で作成されているため、動作にはまず Python のインストールが必要です。Ubuntu にはデフォルトで Python がインストールされていますが、仮想環境構築の利便性から Anaconda（または Miniconda）をインストールすることを推奨します。

Python のインストール

Anaconda（または Miniconda）をインストールすることで Python をインストールすることができます。

Anaconda:

<https://www.anaconda.com/distribution/>

 Anaconda Python/R Distribution - Free Download • www.anaconda.com

Miniconda:

<https://docs.conda.io/en/latest/miniconda.html>

Anaconda 仮想環境の作成

まずは anaconda で仮想環境を作成します。

この作業は Sapphire の利用には必須ではありませんが、Anaconda を利用した Python コーディングでは一般的かつ便利なので、ここでは作成方法についても記載します。

まず端末（ターミナル）アプリを開きます。

次に、仮想環境のつくり方についてです。

下にならされているコマンドを打ち込みましょう。

```
conda create -n sapphire python=3.6
```

(このとき、「sapphire」が仮想環境の名前となります。そして、「python」の後に続く数字は、この仮想環境で使うpythonのバージョンを指定しています。今回(sapphire ver0.3.0) はpython=3.6以上のバージョンを指定してください)

ENTER キーを押して実行すると、次のようになるので、y キーを打ち込んで、ENTER キーを押してください。(y キーの代わりにn キーとすることで、作業を中断できます)

```
(base) r37-89-24-133:~ Masasya$ conda create -n sapphire python=3.7
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.7.10
  latest version: 4.7.12
Please update conda by running
```

```
-----
setuptools-42.0.2 | 645 KB
-----
Total: 801 KB

The following NEW packages will be INSTALLED:

ca-certificates  pkgs/main/osx-64::ca-certificates-2019.11.27-0
certifi          pkgs/main/osx-64::certifi-2019.11.28-py37_0
libcxx           pkgs/main/osx-64::libcxx-4.0.1-hcfea43d_1
libcxxabi        pkgs/main/osx-64::libcxxabi-4.0.1-hcfea43d_1
libedit          pkgs/main/osx-64::libedit-3.1.20181209-hb402a30_0
libffi           pkgs/main/osx-64::libffi-3.2.1-h475c297_4
ncurses          pkgs/main/osx-64::ncurses-6.1-h0a44026_1
openssl          pkgs/main/osx-64::openssl-1.1.1d-h1de35cc_3
pip              pkgs/main/osx-64::pip-19.3.1-py37_0
python           pkgs/main/osx-64::python-3.7.5-h359304d_0
readline         pkgs/main/osx-64::readline-7.0-h1de35cc_5
setuptools       pkgs/main/osx-64::setuptools-42.0.2-py37_0
sqlite           pkgs/main/osx-64::sqlite-3.30.1-ha441bb4_0
tk               pkgs/main/osx-64::tk-8.6.8-ha441bb4_0
wheel            pkgs/main/osx-64::wheel-0.33.6-py37_0
xz               pkgs/main/osx-64::xz-5.2.4-h1de35cc_4
zlib             pkgs/main/osx-64::zlib-1.2.11-h1de35cc_3

Proceed ([y]/n)? y

Downloading and Extracting Packages
setuptools-42.0.2 | 645 KB | ##### | 100%
certifi-2019.11.28 | 156 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

以上で仮想環境の構築は完了です。

※ ここからの話はmac環境を前提に進めております。Linuxなどでは少し違う表示の仕方になっていると思いますが、やり方などに大きな影響はありません。ここに記載されているのと同じ方法で作業を進めることができます

次に、つくった仮想環境に移動します（デフォルトの(base)環境から、先ほどつくった(sapphire)環境に移動）。次のコマンドを打ち込んでください。

```
source activate sapphire
```

すると、(base) → (sapphire)に切り替わります。

以下参照画像 ↓

```
(base) r37-89-24-133:~ Masasya$ source activate sapphire
(sapphire) r37-89-24-133:~ Masasya$
```

必要なモジュールのインストール

Sapphire 起動に必要な Python モジュールをインストールする方法についてです。

必要なモジュールは必要なモジュールをご覧ください。

Sapphire のバージョンによって各モジュールのバージョンが異なりますのでご注意ください。

モジュールのインストール方法

モジュールのインストールは `conda install` コマンドで行います。

(例) Numpy をインストールする

```
conda install numpy
```

(例) Numpy と Scipy をインストールする (複数同時にインストールする)

```
conda install numpy scipy
```

(例) Numpy (Ver. 1.15.0) をインストールする (バージョン指定してインストールする)

```
conda install numpy==1.15.0
```

`conda install` で Dash 関連のモジュールをインストールするとき

Anaconda のリポジトリには Dash 関連のモジュールがありません (2019年12月9日現在)。

目的のモジュールが管理されているリポジトリ (チャンネル) を手動で指定する必要があります。

(例) `dash`、`dash-core-components`、`dash-html-components` モジュールを `conda-forge` チャンネルからインストールする

```
$ conda install -c conda-forge dash dash-core-components dash-html-components
```

※2019/12/15追記：`derickl` チャンネルでは提供されなくなりました。

—(例) `dash-auth` モジュールを `derickl` チャンネルからインストールする

```
$ conda install -c derickl dash-auth
```

※ `conda-forge` チャンネルで提供されるようになりました。

~~dash-table のみ現時点 () では pip でのインストールにだけ対応している。~~

```
$ pip install dash-table
```

※ Anaconda 仮想環境では `pip install` コマンドを使用して Python モジュールをインストールすることも可能ですが、**仮想環境に不具合が生じる可能性がありますので推奨しません。** 可能な限り `conda install` コマンドを使用することをおすすめいたします。

参考：

<https://insilico-notebook.com/conda-pip-install/>

 conda installとpip installの違い。機能の比較など【Python】 | In-slico Notebook • insilico-notebook.com

必要なモジュール

Sapphire の動作に必要なモジュールは以下の表を参照してください。

Sapphire のバージョンによって必要なモジュールのバージョンが異なりますので注意してください。

また各モジュールのバージョンは、開発時に動作確認を行った時のバージョンです。

記載のバージョンよりも上位または下位バージョンのモジュールでも動作することがあります。

Sapphire ver. 0.1.0以前

- dash (0.30.0)
- dash-auth (1.1.2)
- dash-core-components (0.41.0)
- dash-html-components (0.12.0)
- dash-table (3.1.6)
- dash-renderer (0.13.2)
- numpy (1.15.4)

- pandas (0.23.4)
- Pillow (5.2.0)
- plotly (3.2.0)
- scipy (1.1.0)

Sapphire ver. 0.2.0

- dash (0.38.0)
- dash-auth (1.3.2)
- dash-core-components (0.43.1)
- dash-html-components (0.13.5)
- dash-table (3.5.0)
- dash-renderer (0.19.0)
- numpy (1.15.4)
- pandas (0.23.4)
- Pillow (5.2.0)
- plotly (3.2.0)
- scipy (1.1.0)

Sapphire ver. 0.3.0

モジュール名	動作確認済みのバージョン	備考
changefinder	0.3	<code>pip install</code> でのみインストール可能
cupy	9.0	<code>cupy</code> のバージョンは <code>CUDA</code> のバージョンと同じです。 <code>tensorflow-gpu</code> のバージョンと依存関係があるので注意が必要です。
cupy	7.6.0	
dash	0.43.0	<code>dash</code> のみインストールすれば、 <code>dash-core-components</code> 、 <code>dash-html-components</code> 、 <code>dash-</code>

		<code>renderer</code> 、 <code>dash-table</code> 、その他関連モジュールが自動でインストールされます。
<code>dash-auth</code>	1.3.2	<code>pip install</code> でのみインストール可能
<code>dash-core-components</code>	0.48.0	
<code>dash-html-components</code>	0.16.0	
<code>dash-renderer</code>	0.24.0	
<code>dash-table</code>	3.7.0	
<code>keras</code>	2.2.4	
<code>numpy</code>	1.16.5	
<code>pandas</code>	0.25.1	
<code>Pillow</code>	6.1.0	
<code>plotly</code>	3.2.0	
<code>scipy</code>	1.3.1	
<code>tensorflow-gpu</code>	1.9.0	<code>tensorflow-gpu</code> をインストールすれば、 <code>cuda-toolkit</code> 、 <code>cudnn</code> 、その他関連モジュールが自動でインストールされます。
<code>tqdm</code>	4.32.1	

データの準備

オリジナル画像

スキャナで撮像した画像ファイルは、グレースケール画像（カラー画像ではない画像）である必要があります。

また解像度が高すぎる場合、現仕様ではニューラルネットワークが入力として受け取れないため、注意が必要です（一つのウェルのピクセル数が、縦 56 ピクセル以下、横 56 ピクセル以下である必要があります）。

マスクファイルの作成

マスクファイルは、スキャナによって撮像されたオリジナル画像を各ウェルごとの画像に分割する際に必要な情報です。

マスクファイルは Sapphire を利用して作成可能ですので、まず Sapphire を起動してください。

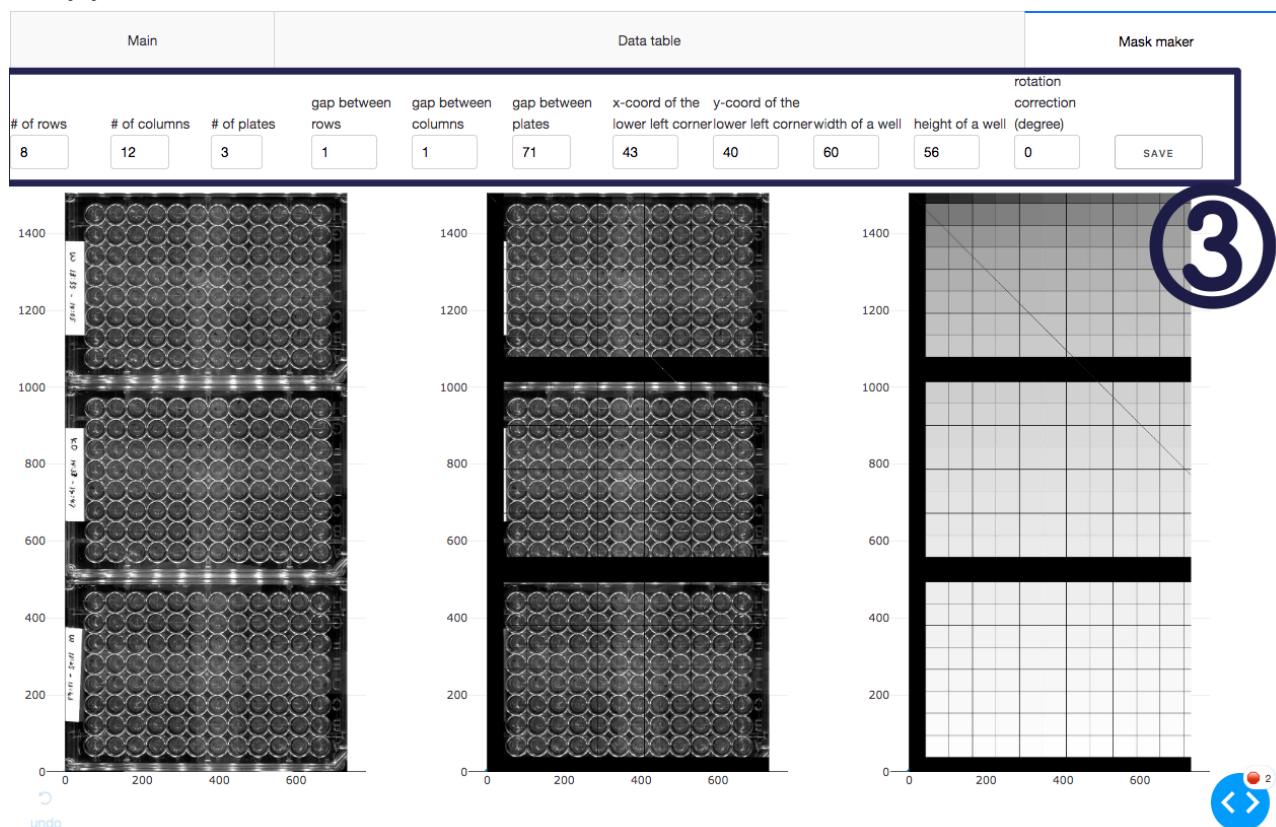
マスクファイル作成の手順

1. データセットを選択する
2. 画面に表示された画像上でウェルをひとつ枠取りをする
3. 新たに右側に表示された画像を見ながら、パラメータを変化させ、チューニングを行う
4. maskが各ウェルごときちんと切り出されているか確認をする
5. **SAVE** ボタンを押して、パラメータを保存する

各手順について(以下の画像を参考に↓)

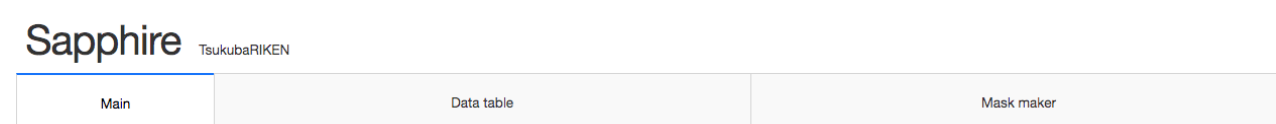


参考画像「Mainのレイアウト」



参考画像「Mask makerのレイアウト」

1. データセットを選択する



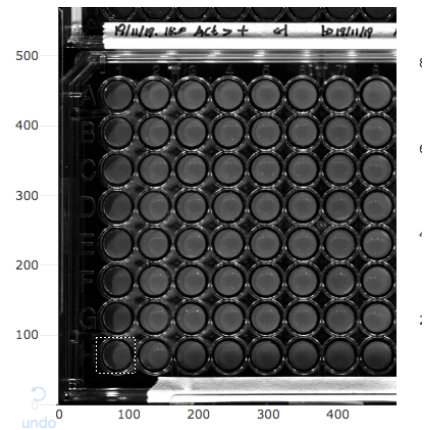
Sapphireのタブ

デフォルトで選択されているMainタブで、データセットを選びます。
参考画像(2)のところで、データセットを選択します。
そして、一番右側のMask makerタブをクリックします。

2. 画面に表示された画像上でウェルをひとつ枠取りをする

画像の読み込みが終わると、先ほど選んだデータセットの1番目の画像が表示されます。
左側に読み込まれた画像が表示される。
画像左下のウェルを、右クリックして点線で囲む。

(何度でも囲い直すことが可能)



左下のウェルを囲む

3.新たに右側に表示された画像を見ながら、パラメータを変化させ、チューニングを行う

参考画像(3)のところでパラメータ変更可能。

# of rows	# of columns	# of plates	gap between rows	gap between columns	gap between plates	x-coord of the lower left corner	y-coord of the lower left corner	width of a well	height of a well	rotation correction (degree)	
<input type="text" value="8"/>	<input type="text" value="12"/>	<input type="text" value="3"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="71"/>	<input type="text" value="54"/>	<input type="text" value="23"/>	<input type="text" value="54"/>	<input type="text" value="61"/>	<input type="text" value="0"/>	<input type="button" value="SAVE"/>

各パラメータについて

番号	パラメータ	備考
1	# of rows	1 プレーットのウェルの行数
2	# of columns	1 プレーットのウェルの列数
3	# of plates	プレーットの個数
4	*gap between rows	ウェル間の列のすき間の幅
5	*gap between columns	ウェル間の行のすき間の幅
6	gap between plates	プレート間のすき間の幅
7	x-coord of the lower left corner	点線で囲んだ左下端のx座標
8	y-coord of the lower left corner	点線で囲んだ左下端のy座標
9	width of a well	1 プレーットの横辺の長さ
1 0	height of a well	1 プレーットの縦辺の長さ
1 1	rotation correction (degree)	全体の傾き

(*)マークがついているパラメータはほとんど変更しない。

Mask を作る際のコツ

方法1. 過去のMaskのパラメータ値を参考にする

方法2. 最初の枠取りを丁寧に行う

4.maskが各ウェルごときちんと切り出されているか確認をする

確認の仕方

確認1. 右上のウェルがmaskによって、きちんと枠取りされているか

確認2. (他の確認方法を聞く)

5.SAVE ボタンを押して、パラメータを保存する

参考画像(3)のパラメータの右側にあるSAVE ボタンを押す。

すると次の画像のようなダイアログが出てくる。Maskの上書きがされますがよろしいでしょうか、という内容が書かれているので、それでもよければ、OK ボタンを押しましょう。その後、保存が完了したことを知らせるダイアログも出てきます。



ダイアログ その1



ダイアログ その2

データベースのファイルと階層構造

階層構造

xxx：ファイル

xxx/：ディレクトリ

Data/

```
└─ [Your dataset name 1]/
  │   └─ inference/
  │   └─ original/
  │       └─ [目視判定ファイル]
  │   └─ blacklist.csv
  │   └─ config.json
  │   └─ grouping.csv
  │   └─ mask.npy
  │       └─ mask_params.json
└─ [Your dataset name 2]/
...
└─ [Your dataset name N]/
```

Data/

データベースのルートディレクトリ。

Sapphire はこのディレクトリ以下に配置されている各データセットの解析結果を読み込みます。

inference/

推論結果を格納するフォルダです。

original/

オリジナルのタイムラプス画像を格納するフォルダです。

すべてのタイムラプス画像のファイル名は撮像時間順に連番である必要があります。

[目視判定ファイル] は目視判定結果を記述するファイルです。

目視判定ファイルが存在する場合、ライフイベントごとにその判定結果を記したcsvファイルを選択し、読み込むことができます。（2018年11月1日現在、目視判定ファイルはcsvのみ対応しています。）

csvの行列は、撮像されたショウジョウバエ集団のウェルの配置に対応しており、対応する行列のセル内数値は、イベントが生起したタイミング（何枚目の画像）となっています。

pupariation.csv：「蛹化」の目視判定結果

eclosion.csv：「羽化」の目視判定結果

将来的には、目視データが存在しないデータを扱っていきますが、目視データが存在する場合には、それらを読み込み、自動判定の精度を見積もることができます。また、目視データが存在しない場合でも、撮像環境（ショウジョウバエ集団画像の取得環境）が同一であれば、その精度を保証するためのチューニングとして用いることも想定されます。

[目視判定ファイル] は以下のいずれかのファイル名である必要があります。

- pupariation.csv
- eclosion.csv
- death.csv

blacklist.csv

ブラックリスト。

Survival Ratio や Consistency の評価から除外したいウェルを指定します。

オリジナル画像のウェルの位置と各セルが対応している整数値テーブルです。

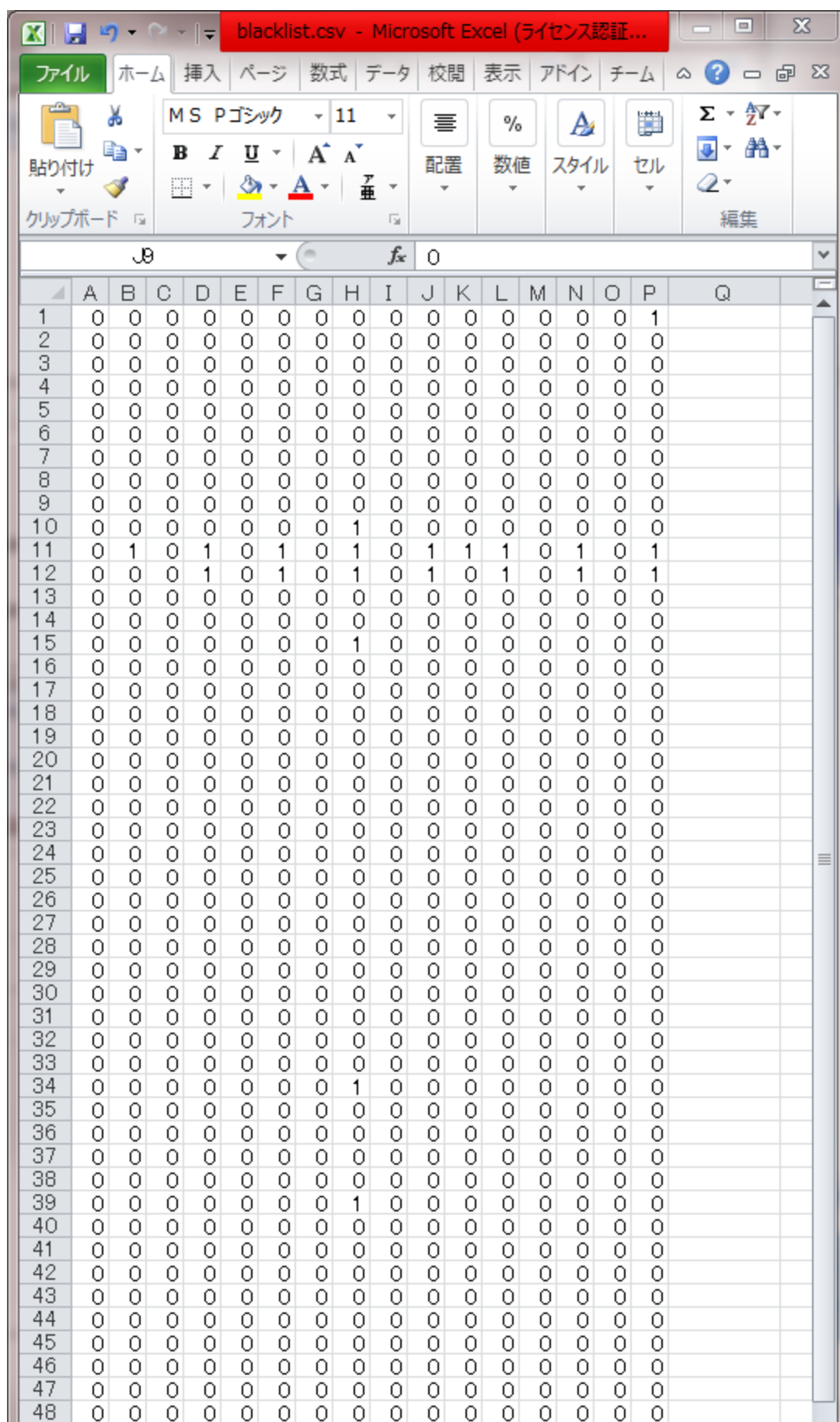
各セルの要素は評価から除外するかしないかを意味します。

例：

- 空のウェル
- 1 ウェル内に個体が 2 匹入っているウェル

0：解析に含めるウェル

1：解析に含めないウェル





例：blacklist.csv

config.json

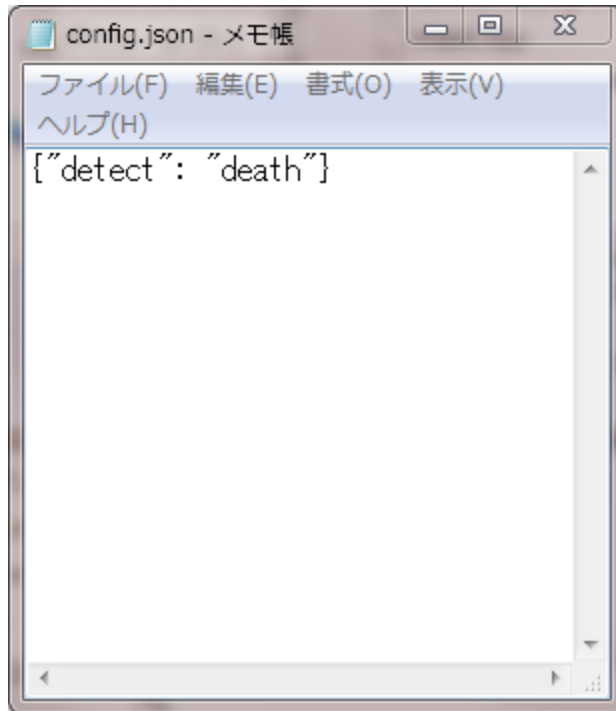
データセットのパラメータ設定用ファイル。

今の所パラメータは detect の1つのみです。

detect には以下の4種の文字列のうちどれか1つを設定します。

detect	意味
pupariation	蛹化検出用データセット
eclosion	羽化検出用データセット
pupa&eclo	蛹化・羽化検出用データセット
death	死亡検出用データセット

ファイル例：



死亡検出用データセットの config.json ファイル

grouping.csv

グループ定義用のファイル。

オリジナル画像のウェルの位置と各セルが対応している整数値テーブルです。

各セルの要素はグループ番号を意味します。

グループ番号は1から始まり、連番である必要があります。

※0 はグループの分類には使えません。どのグループにも属していないウェル（空のウェルなど）を指定するために使います。

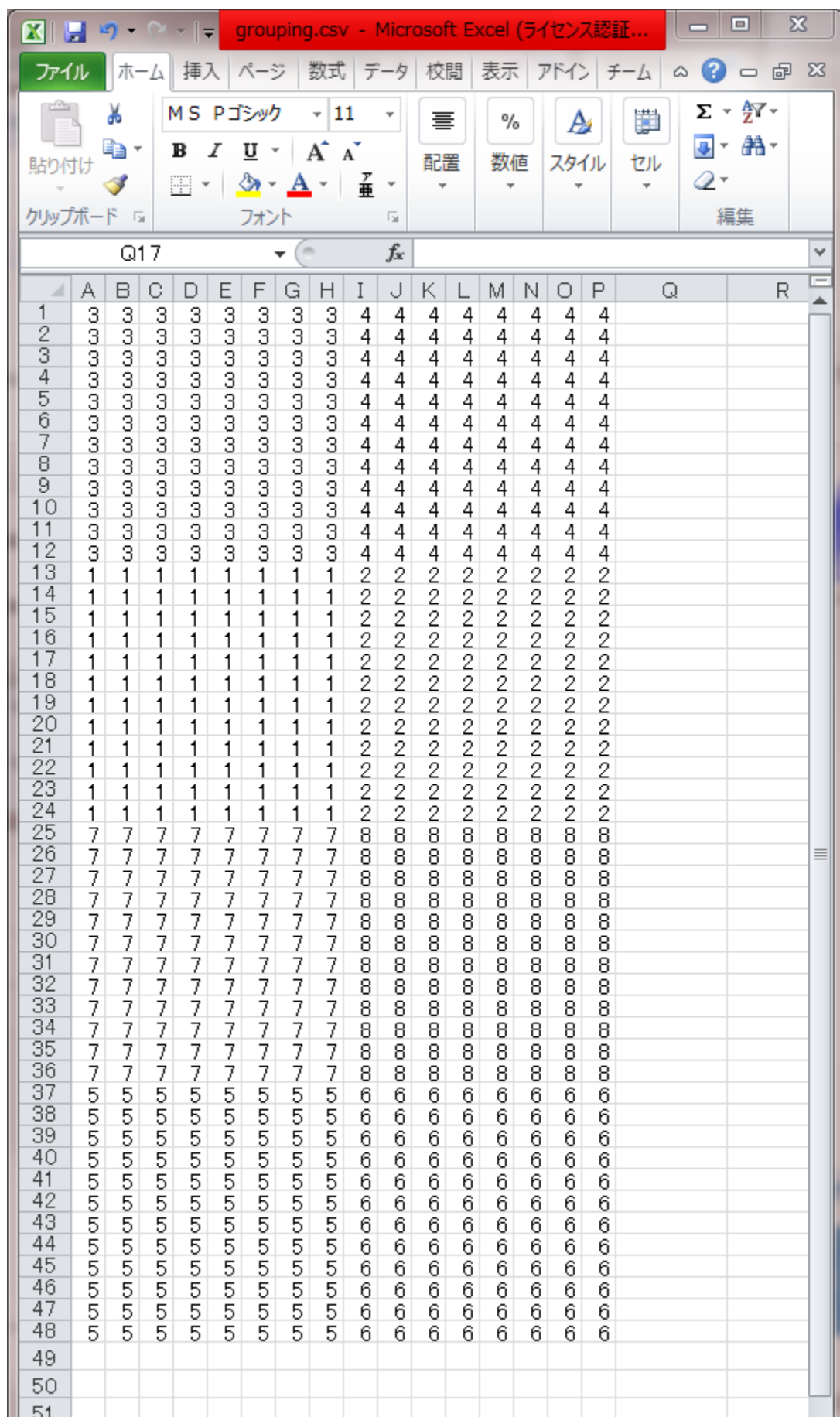
0：どのグループにも属していないウェル

1：グループ1 に属しているウェル

2：グループ2 に属しているウェル

3：グループ3 に属しているウェル

...





例：grouping.csv (8グループ)

mask.npy と mask_params.json

マスクファイル。

推論作業

データセットと訓練済みネットワークの場所をコピーして貼り付ける

まず、コードが集められたフォルダの場所に移動していることが前提なので、その確認をしてください。(ターミナルを新しく開き直すなどすると、仮想環境も、移動した場所もデフォルトの状態にリセットされてしまいます)

推論

推論を実行するには `inference.py` という Python スクリプトを実行します。

必要なモジュールをインストールした仮想環境でスクリプトを実行すると推論が開始されます。

推論時には推論対象となるデータセットのパスと、推論に使用する訓練済みネットワークがあるパスの2つを引数として指定する必要があります。

(例)

以下のようなデータセットフォルダと訓練済みネットワークがある状況で推論を実行します。

データセット：`/home/deepstation/datasets/dataset1`

訓練済みネットワー

ク：`/home/deepstation/trained_network/adult/profile1/network_file.h5`

以下のコマンドを実行すると推論が開始されます。

```
python inference.py /home/deepstation/datasets/dataset1 /home/deepstation/trained_network/adult/profile1/network_file.h5
```

※コマンドと各引数の間には必ずスペースを入れてください。

※GPU（GTX1080Ti）使用時は概算でオリジナル画像1枚あたり1秒かかります。

上記の例のように `python` コマンドに対して `inference.py` とデータセットフォルダのパス（`/home/deepstation/datasets/dataset1`）、訓練済みネットワークのパス（`/home/deepstation/trained_network/adult/profile1/network_file.h5`）をスペースで区切って記述し実行します。

make_CF_signal.py を使ってシグナルを作成

ChangeFinder（CF）を使い変化点検出を行います。

CF を実行するには `make_CF_signals.py` を実行します。

`make_CF_signals.py` の実行には、「変化点検出の対象となる活動量シグナルデータのパス」と、「検出したいイベント」の2つの引数が必要です。

（例）

以下の活動量シグナルに対して蛹化時刻を変化点検出として検出する場合

シグナルパ

ス：`/home/deepstation/datasets/dataset1/inference/larva/target_dir/signals.npy`

以下のコマンドを実行すると CF シグナルが作成されます。

```
python make_CF_signals.py /home/deepstation/datasets/dataset1/inference/adult/target_dir/signals.npy pupariation
```

`make_CF_signals.py` が検出可能なイベントは以下の 3 種類です。

イベントの種類	<code>make_CF_signals.py</code> の第2引数値	CF によって検出されるポイント
蛹化	pupariation	立ち下がり
羽化	eclosion	立ち上がり
死亡	death	立ち下がり

蛹化検出の場合、CF は活動量シグナルの立ち下がりポイントを検出します。

なお、CF を実行してもイベントがうまく検出されない場合は、第3引数である r 値を調整することで改善される可能性があります。

(例)

`r = 0.009` で CF を実行する場合

以下のように、3つ目の引数に r 値 (0.009) を明示します。

```
python make_CF_signals.npy /home/deepstation/datasets/dataset1/inference/larva/target_dir/signals.npy pupariation -r0.009
```

※ r 値は $0 < r < 1.0$ で指定してください

r 値は撮像画像の総フレーム数に応じて調整すると改善が見込めます。

r 値と総フレーム数の対応は下の表を参考にしてください。

総フレーム数	r 値
1,000 未満	0.009
$1,000 \leq r < 4000$	0.003
$4,000 \leq r < 10,000$	0.001
10,000 以上	0.0006

Sapphire の利用

Sapphire はウェブアプリケーションでアプリの利用自体は基本的には任意のウェブブラウザから可能です（Google Chrome でのみ動作確認済み）。

アプリでは3つのタブがあり、各タブが提供する主な機能は以下のとおりです。

メインタブ

- スキャナで撮像したオリジナル画像の閲覧
- 推論結果の閲覧
- ウェルごとの活動量シグナル、自動イベント判定結果の閲覧

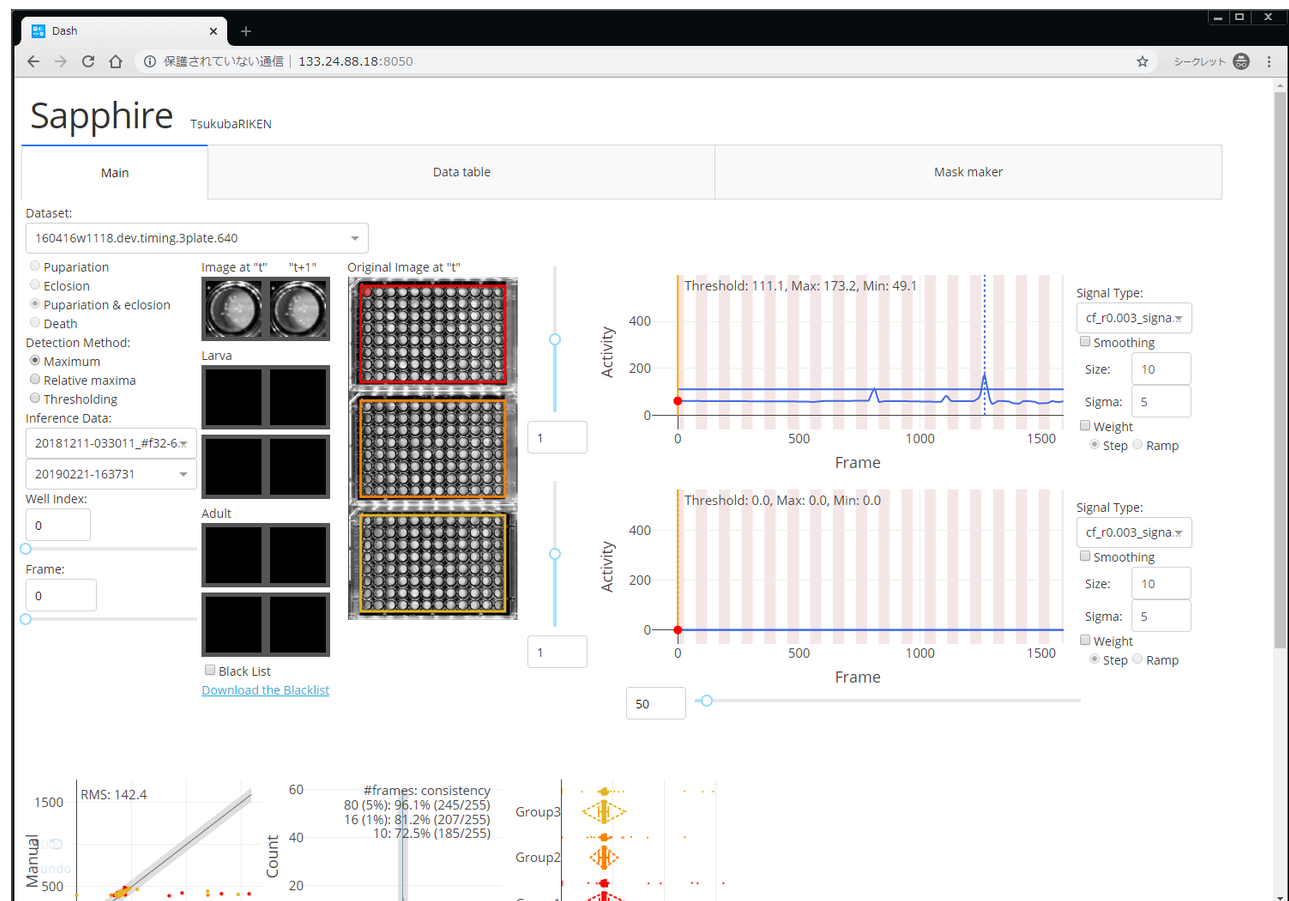
データテーブルタブ

- すべてのウェルの自動イベント判定結果の閲覧とデータのダウンロード

マスク作成タブ

- マスクファイルの作成

スクリーンショット：



利用方法

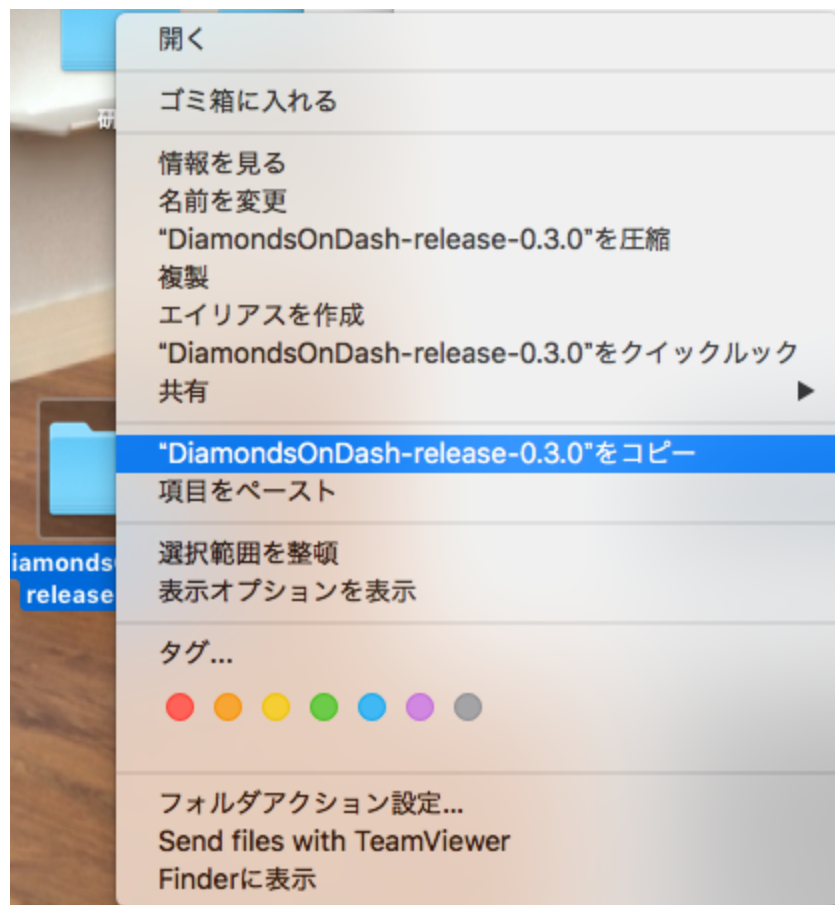
Sapphire を利用するにはまず sapphire.py という Python スクリプトを実行し、Sapphire を起動します。起動後、ブラウザ（Google Chrome 推奨）を起動し、指定のアドレスにアクセスすることで Sapphire の利用が可能になります。

Sapphire の起動方法

そのコードをターミナル上で実行するには、そのコードがある場所に移動する必要があります。

簡単な方法として、コードが集められたフォルダの場所を参照するという方法があります。

ここでは、「DiamondsOnDash-release-0.3.0」というフォルダに「sapphire.py」があるので、そのフォルダを左クリックし、コピーを選択します。下の写真参照



※ Macの場合

コピーをすることにより、フォルダの場所を取得できます。取得した場所をそのままターミナルに貼り付けることで、移動することができます。下の画像のように、「DiamondsOnDash-release-0.3.0」に移動できていることがわかります。コードのあるフォルダの場所に移動することで、「sapphire.py」を実行できるようになりました。

```
(sapphire) r37-89-24-133:~ Masasya$ cd /Users/hatamasaya/Desktop/DiamondsOnDash-release-0.3.0
(sapphire) r37-89-24-133:~ Masasya$
```

そして、「sapphire.py」を実行します。実行するためのコマンドが次の通りです

```
python sapphire.py
```

しかし、仮想環境に、手順2の通りにモジュールを入れただけでは、「sapphire.py」を実行するためのモジュールが足りていようです。以下のような文章が出てきます

```
(sapphire) r37-89-24-133:DiamondsOnDash-release-0.3.0 Masasya$ python sapphire.py
Traceback (most recent call last):
  File "sapphire.py", line 23, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
(sapphire) r37-89-24-133:DiamondsOnDash-release-0.3.0 Masasya$
```

```
ModuleNotFoundError: No module named 'pandas'
```

「'pandas'という名前のモジュールが入っていないですよ」というエラー文です。これを解決するために、もう一度モジュールを入れる作業をします。デフォルトでは以下のモジュールが足りていませんでした。

	モジュール名	コマンド(conda or pip)
1	pandas	conda コマンド利用可能
2	scipy	conda コマンド利用可能

ここでは、「conda」でインストールするように統一しているので、「conda」でインストールを進めます。このとき、バージョンの指定はありません。上記二つとも「conda」でインストール可能なので、同時に入れてしまいましょう。

これら全てをインストールすることで、「sapphire.py」をきちんと実行できます。

4.実行できたら、表示されたURLを検索サイトに貼り付ける

「sapphire.py」を実行することができたら、次のような画面になります

```
(sapphire) r37-89-24-133:DiamondsOnDash-release-0.3.0 Masasya$ python sapphire.py
Running on http://127.0.0.1:8050/
Debugger PIN: 816-725-162
* Serving Flask app "Sapphire" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
Running on http://127.0.0.1:8050/
Debugger PIN: 452-545-802
```

赤枠で囲まれた部分が、「sapphire」を起動するために必要なURLです。URLをコピーするなどして、検索エンジン(Google Chrome など)に打ち込みます。検索すると、「sapphire」を起動させることができます。

ブラウザで Sapphire にアクセスする

ブラウザを起動してアドレスバーに下記 URL を入力してアクセスしてください。

<http://localhost:8050/>

(または <http://127.0.0.1:8050/>)

メインタブ

オリジナル画像、推論結果、自動イベント判定結果を閲覧するためのタブです。

Main
Data table
Mask maker

Dataset:
160416w1118.dev.timing.3plate.640

Pupariation
Eclosion
Pupariation & eclosion
Death

Detection Method:
Maximum
Relative maxima
Thresholding

Inference Data:
20181211-033011_#f32-6
20190221-163731

Well Index:
0

Frame:
0

Image at "t"
"t+1"

Larva

Adult

Black List
[Download the Blacklist](#)

Original Image at "t"

Activity

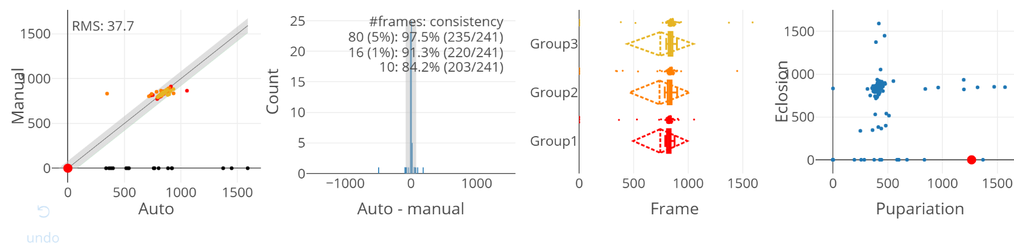
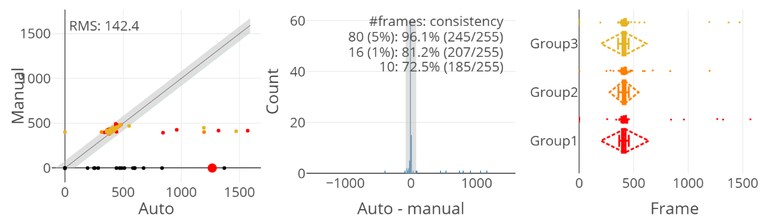
Activity

Threshold: 111.1, Max: 173.2, Min: 49.1

Signal Type:
cf_r0.003_signa.
Smoothing
Size: 10
Sigma: 5
Weight
Step
Ramp

Threshold: 0.0, Max: 0.0, Min: 0.0

Signal Type:
cf_r0.003_signa.
Smoothing
Size: 10
Sigma: 5
Weight
Step
Ramp



データセット

Dataset:

160416w1118.dev.timing.3plate.640

- ☐ Pupariation
- ☐ Eclosion
- ☒ Pupariation & eclosion
- ☐ Death

表示したいデータセット（フォルダ）を選択します。

選択したデータセットの種類によってラジオボタンが選択されます。

選択される項目は config.json ファイルの `detect` 変数の内容と対応します。

フォルダ内データは主に、同時飼育下におけるショウジョウバエ集団のタイムラプス画像ですが、解析をすすめるとともに、セグメンテーションの訓練結果や推論結果、信号データなども保存されます。同一フォルダ内データであるかによって、対応関係が識別されます。

イベント判定方法

Detection Method:

- ☒ Maximum
- ☐ Relative maxima
- ☐ Thresholding

推論済みデータ

Inference Data:

20181211-033011_#f32-6.▼

20190221-163731 ▼

「推論」を行った結果のファイルです。

推論とは、（ヒトが作成した）予め答えがわかっている教師データを用いて訓練を行い、その後（訓練後）に新規データで（幼虫あるいは成虫のセグメンテーションを）行った結果のファイルです。

訓練に用いた教師データの数や内訳によって結果が異なるので、複数存在する場合があります。

訓練に用いた教師データの内訳には、以下のように名称が与えられています。

- normal : 明らかに個体が存在することが判別しやすいもの（を「訓練」に用いた）。
- noisy : ノイズなどにより、一見しただけは個体が判別しにくいもの（を「訓練」に用いた）。
- empty : （死角にいる？などの理由により）一見しただけでは、個体がいないように見えるもの（を「訓練」に用いた）。

2018年11月1日の時点では、主に上記三種を様々な割合で混合させて訓練を行ったネットワークを用いて推論を行いました。

ウェル、フレーム

Well Index:

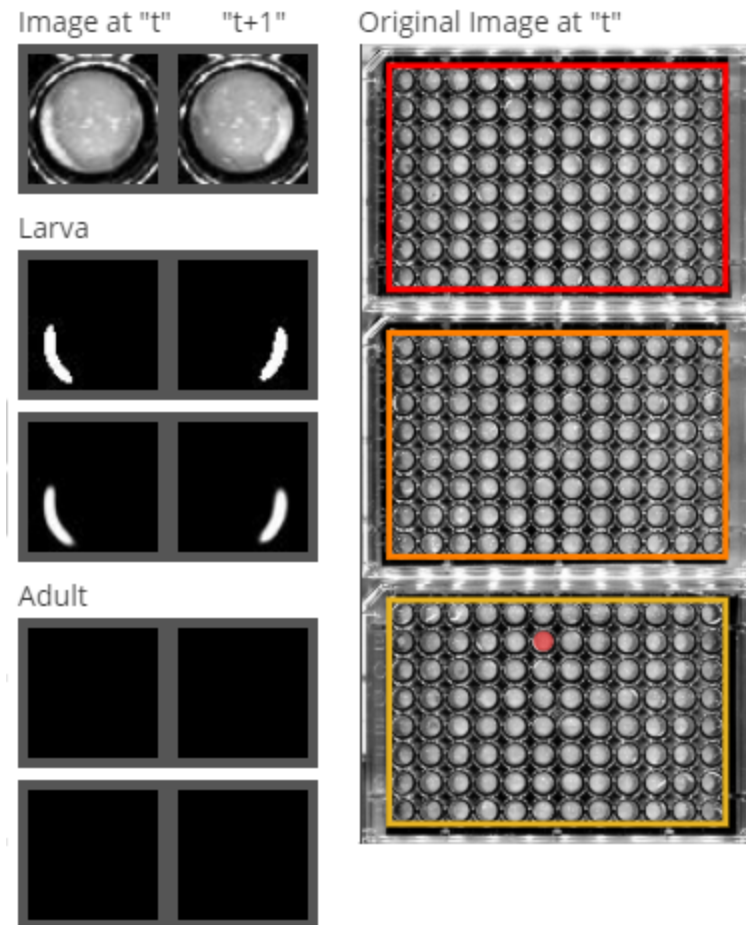


Frame:



注目するウェル番号、タイムステップを選択します。

時刻 t と $t+1$ のウェル画像とラベル



選択されたパラメータに対応した各画像が表示されます。

画像は、指定したウェルおよびタイムステップ(t)に対して、変化を比較するために(t+1)の画像を並べて表示します。左右の列がそれぞれのタイムステップに対応しています。

また、最上段は原画像、二段目はセグメンテーションの結果「個体（の一部）」と判定されたピクセルをラベルした画像となります（0 or 1の値をとります）。

三段目は、セグメンテーションを行った結果算出される「そのピクセルが個体（の一部）である確率」です（0-1の値をとります）。この確率がある値（2018年11月時点では、0.5）以上の場合、そのピクセルは「個体（の一部）」とラベルされます。そのようにラベルされた結果が二段目に表示されています。

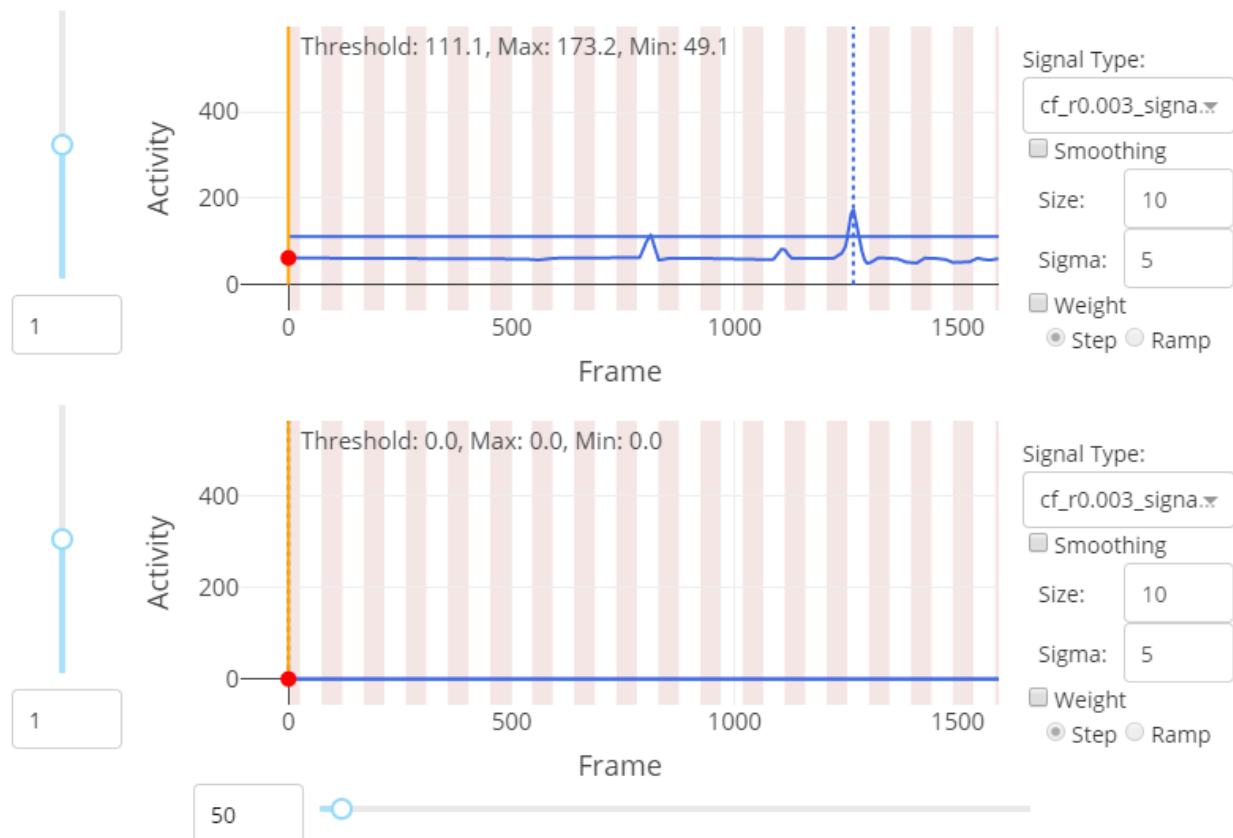
ブラックリスト

☐ Black List

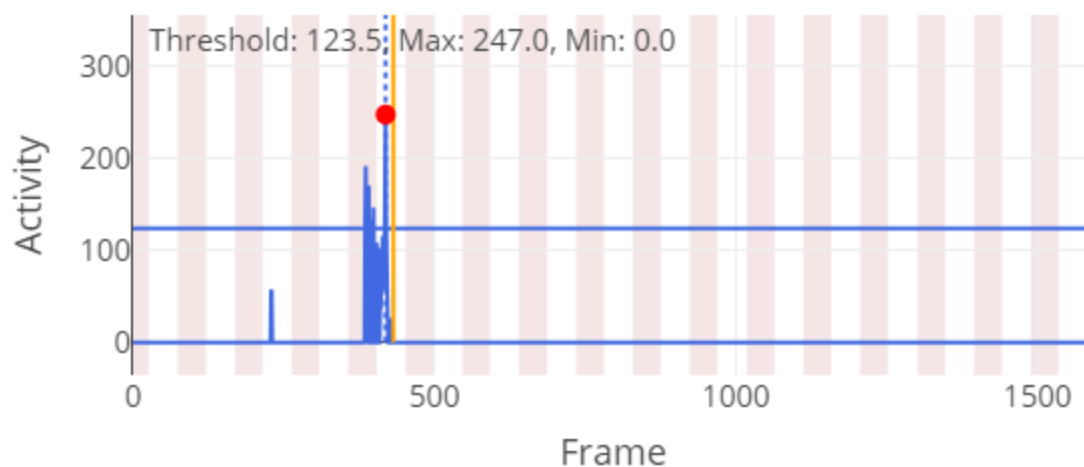
[Download the Blacklist](#)

チェックボックスにチェックを入れると対象ウェルが解析から除外されます。
リンクをクリックすると全ウェル分のチェック情報が入った CSV ファイルがダウンロードされます。

活動量シグナル表示部分



活動量シグナル



オレンジ : 目視判定によるイベントの検出点（目視判定のcsvファイルが存在し、それを読み込んだ場合にのみ描画）

青点線 : 自動判定 1（セグメンテーションにより得られた差分信号が、設定された閾値を横切ったタイミングにより算出される）によるイベントの検出点

緑点線 : 自動判定 2（輝度値差分により得られた差分信号が、設定された閾値を横切ったタイミングによって算出される）によるイベントの検出点

信号の種類

Signal Type:

signals.npy ▼

スムージング

☐ Smoothing

Size: 10

Sigma: 5

重み付け

☐ Weight
☒ Step ☐ Ramp

閾値調整（スライダーバー）



「セグメンテーション結果の差分から得られた信号」と、「輝度値の差分から得られた信号」という2つの信号それぞれに対して任意の閾値を設定することができます。

（信号の色と対応）

閾値の変更により、下段の要約図（「自動・目視判定の比較散布図」および「自動・目視判定誤差のヒストグラム」）もリアルタイムで更新されます。

セグメンテーション結果による信号（青色）に関しては、デフォルトでは、「m: 全ウェルの信号平均」「s: 全ウェルの信号から得られた標準偏差」として、閾値を、「 $m + \alpha \times s$ （ α はパラメータ）」で設定しており、ユーザーは、 α を $-5.0 \sim +10.0$ の範囲で設定することができる仕様になっています。また、全ウェル一律同じ閾値を適用しています（2018年11月1日時点）。

輝度値による信号（緑色）に関しては、信号のy軸と対応して閾値そのものをスライダーバーによって設定します。

重み付けのための中間点

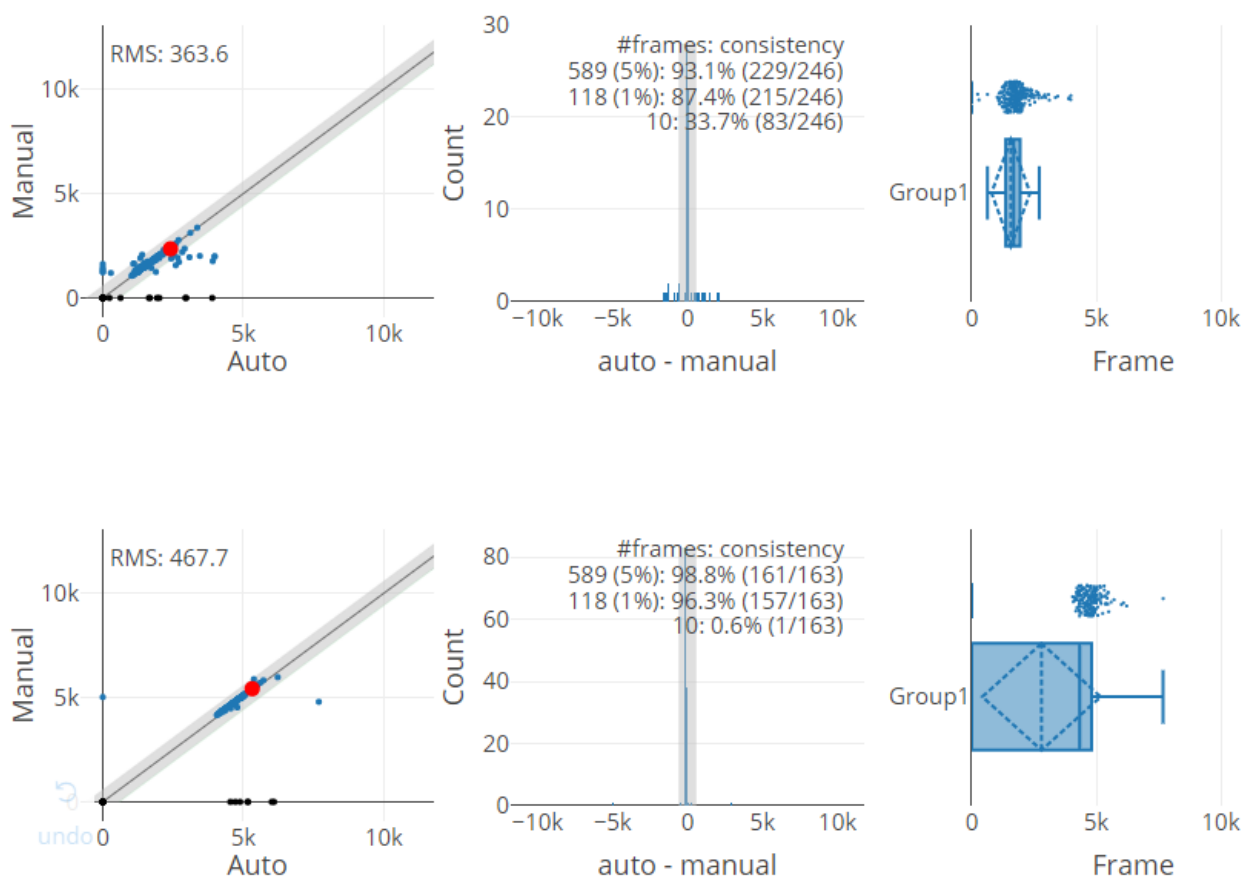


重みの起点を選択するためのスライダーおよびスピンドボックスです。

重み付けチェックボックスを有効にした際に使用します。

ここで選択したフレームを起点として、ステップ荷重であれば 0 と 1 が切り替わり、ラン
プ荷重であれば荷重の値が増加し始めます。

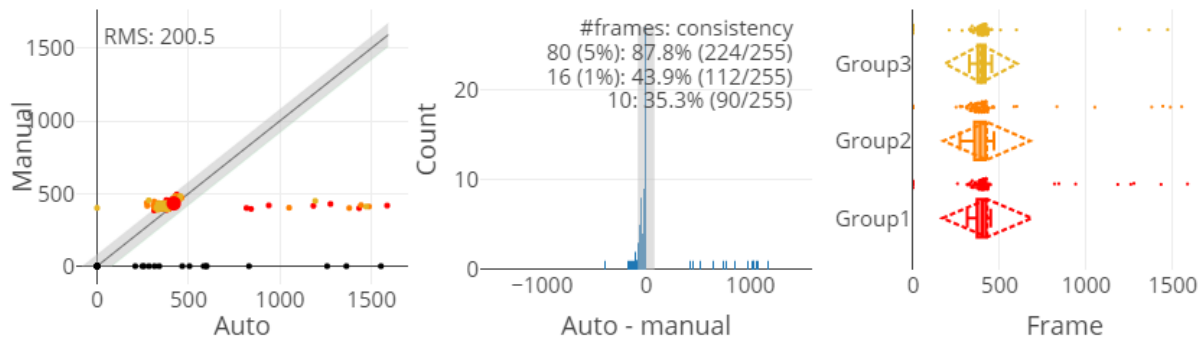
イベントの検出結果



目視判定と自動判定の比較

各シグナルに対して閾値処理を行い、イベント検出を行った結果を表示します。

目視判定と自動判定の比較（散布図、ヒストグラム）



右2つ : 自動判定1 (セグメンテーションの差分信号を用いて判定されたイベントタイミング) と目視判定の比較

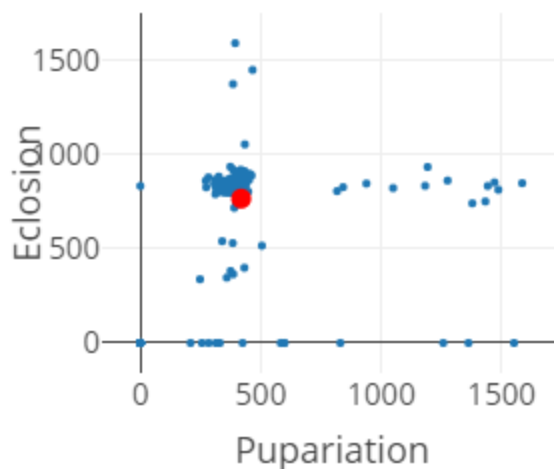
左2つ : 自動判定2 (輝度の差分信号を用いて判定されたイベントタイミング) と目視判定の比較

赤点 : 現在注目しているウェル

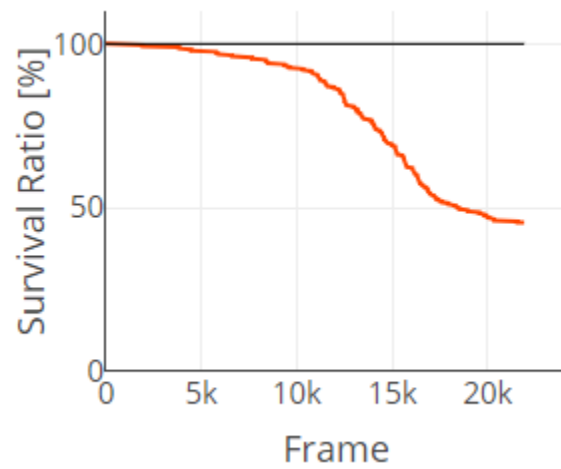
緑枠 : 目視判定と自動判定の誤差が全フレームの5%以内のもの

#frames:consistency: 目視判定と自動判定との誤差が全フレームの5%、1%、10フレーム以内のものの割合

蛹化時刻 vs 羽化時刻



サバイバルカーブ



スクリーンショット

データテーブルタブ

Main	Data table	Mask maker
------	------------	------------

Timestamp

Image name	Create time
0001.jpg	2016-04-16 12:13:11
0002.jpg	2016-04-16 12:27:04
0003.jpg	2016-04-16 12:42:20
0004.jpg	2016-04-16 12:57:36
0005.jpg	2016-04-16 13:12:53
0006.jpg	2016-04-16 13:28:08
0007.jpg	2016-04-16 13:43:24
0008.jpg	2016-04-16 13:58:40
0009.jpg	2016-04-16 14:13:55
0010.jpg	2016-04-16 14:29:11
0011.jpg	2016-04-16 14:44:28
0012.jpg	2016-04-16 14:59:43
0013.jpg	2016-04-16 15:14:59
0014.jpg	2016-04-16 15:30:15
0015.jpg	2016-04-16 15:45:31
0016.tif	2016-04-16 16:00:47

[Download](#)

Event timings of larva (manual)

	A	B	C	D	E	F	G	H	I	J	K	L
0	409	440	426	435	415	410	405	399	388	408	384	
424	447	398	415	424	407	398	403	427	445	0	394	
455	392	0	0	419	428	392	426	405	0	417	412	
398	398	493	403	416	436	412	394	406	421	395	411	
396	396	433	428	410	441	410	431	393	417	446	433	
398	396	400	433	403	420	416	428	412	403	412	0	
413	0	425	427	427	435	399	418	448	412	393	0	
407	0	433	400	400	455	420	410	404	413	417	440	
405	395	0	396	406	0	412	407	459	415	428	420	
0	408	436	0	431	417	407	427	403	401	415	398	
424	428	413	424	0	413	418	458	419	436	484	428	
0	0	403	434	0	0	400	0	394	480	413	424	
417	430	412	428	429	407	418	0	0	435	418	424	
416	411	402	396	424	417	417	420	435	438	416	410	
418	445	420	0	437	441	420	392	408	419	422	430	
421	409	435	415	417	471	422	429	401	422	402	409	
412	440	434	407	409	407	419	404	432	453	416	427	
401	460	388	421	410	432	0	423	421	414	425	460	
430	0	411	417	411	422	0	412	414	459	435	435	
392	398	447	430	415	427	0	403	430	421	416	430	
406	395	432	397	410	0	0	431	407	426	450	413	
0	407	391	436	404	470	0	432	416	420	406	412	
404	454	411	406	413	410	0	418	437	434	421	433	
449	441	410	435	417	435	0	412	0	419	401	425	

[Download](#)

Event timings of larva (auto)

	A	B	C	D	E	F	G	H	I	J	K	L
1206	408	438	423	434	411	406	443	396	384	393	382	
422	445	396	385	423	398	394	400	425	441	0	386	
449	398	0	0	410	423	394	423	401	0	396	403	
391	391	437	399	411	420	404	843	402	412	369	406	
387	393	438	427	406	435	409	426	388	398	444	430	
390	394	395	431	400	416	414	367	408	445	405	0	
403	0	423	901	413	386	340	1320	423	408	386	0	
404	257	429	456	430	432	412	402	395	404	1571	436	
404	389	676	394	392	471	401	403	445	413	415	419	
0	399	428	239	410	414	405	411	398	399	413	387	
421	415	388	423	0	405	400	456	414	435	482	426	
0	288	379	432	585	0	304	443	390	473	361	406	
404	425	411	423	420	370	370	835	601	394	411	421	
413	403	1196	392	407	414	412	412	431	435	412	404	
404	441	415	487	434	438	415	388	405	401	420	429	
417	407	428	406	415	470	415	422	390	419	315	405	
410	437	431	403	407	405	411	391	428	450	414	424	
396	476	376	418	409	428	195	416	416	412	382	452	
424	1370	406	415	390	416	0	399	402	453	414	432	
387	379	445	413	413	422	0	392	424	412	401	428	
404	393	416	392	414	599	0	428	405	419	442	411	
0	388	390	431	401	552	0	394	414	358	406	401	
390	413	408	350	400	403	0	407	435	422	415	431	
1194	430	1472	433	412	431	0	411	511	412	0	421	

[Download](#)

Event timings of adult (manual)

	A	B	C	D	E	F	G	H	I	J	K	L
0	807	832	836	825	804	801	835	802	797	803	773	
825	845	797	830	824	806	831	828	838	0	0	793	
852	807	0	0	814	850	812	818	807	0	825	825	
816	793	900	815	830	833	812	824	812	829	795	802	
794	790	846	845	824	805	827	849	829	828	853	811	
829	813	796	832	813	822	818	850	827	912	829	0	
816	0	842	844	833	0	817	841	877	817	795	0	
825	0	824	0	850	875	819	825	801	813	830	859	
836	832	0	792	800	0	816	827	867	840	863	865	
0	831	839	0	861	842	808	833	821	796	0	813	
821	857	867	827	0	822	838	850	820	856	0	842	
0	0	799	862	0	829	0	816	894	833	846		
836	840	826	833	854	819	815	0	0	841	822	839	
835	813	810	803	842	816	836	833	832	864	835	831	
814	854	830	0	851	861	831	806	821	839	823	832	
842	834	844	819	837	890	837	829	802	840	807	811	
809	867	843	821	832	805	846	801	877	873	836	835	
824	0	0	823	821	0	0	850	818	832	820	865	
846	0	827	835	831	825	0	840	819	861	0	851	
794	824	871	848	832	854	0	0	857	849	835	0	
818	826	846	790	830	0	0	863	832	0	852	813	
0	830	810	830	821	0	0	863	820	839	829	834	
863	830	836	837	830	0	849	872	0	835	859		
877	847	830	825	830	0	837	0	844	831	833		

[Download](#)

Event timings of adult (auto)

	A	B	C	D	E	F	G	H	I	J	K	L
0	813	831	836	826	803	811	839	822	803	810	790	
825	848	803	834	822	807	830	830	840	365	0	799	
855	817	0	0	816	851	811	816	809	0	826	840	
818	804	905	818	831	840	811	827	813	828	799	823	
804	803	847	843	835	1054	828	847	837	831	864	832	
849	816	800	836	813	820	823	862	825	912	839	0	
825	0	848	844	834	530	751	846	875	819	806	0	
834	0	829	803	855	876	821	820	807	733	840	847	
831	846	0	793	814	1449	824	828	871	848	871	824	
0	840	838	338	863	845	807	833	831	798	759	813	
831	869	862	825	0	821	846	857	818	861	399	843	
0	0	802	862	0	0	741	0	822	898	347	896	
935	840	831	837	785	819	813	0	0	843	831	853	
849	827	822	717	850	818	844	834	838	874	838	830	
832	867	832	540	854	859	833	811	829	847	829	835	
852	842	887	826	837	889	842	831	802	847	827	825	
809	867	853	819	830	806	850	799	878	879	842	839	
829	806	0	823	821	766	0	848	816	832	820	873	
847	0	829	845	829	824	0	844	827	864	382	862	
798	832	874	846	830	860	0	1374	857	854	840	0	
883	828	848	790	832	0	0	867	837	1591	859	821	
0	838	810	831	822	919	0	861	819	840	836	843	
837	910	830	834	838	832	0	847	873	921	843	860	
934	876	852	830	840	841	0	838	510	840	833	830	

[Download](#)

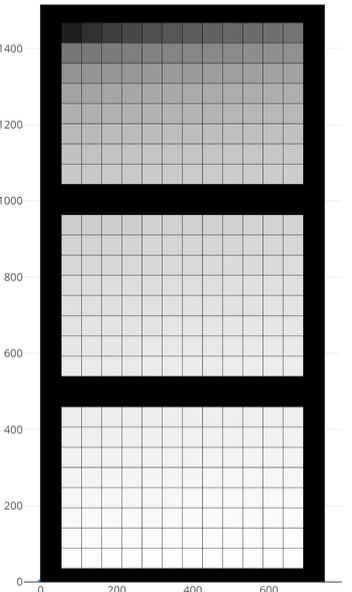
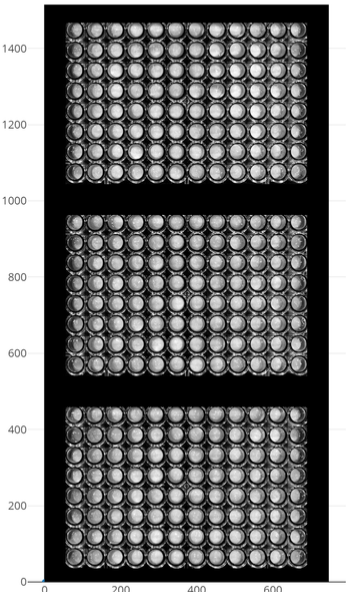
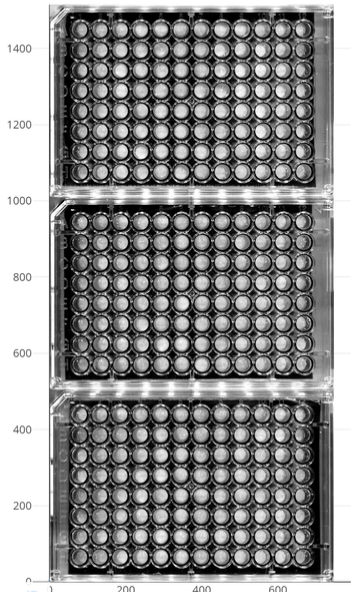
リンクをクリックすることで解析結果のダウンロードができます。

マスク作成タブ

Sapphire

TsukubaRIKEN

Main			Data table						Mask maker		
# of rows	# of columns	# of plates	gap between rows	gap between columns	gap between plates	x-coord of the lower left corner	y-coord of the lower left corner	width of a well	height of a well	rotation correction (degree)	
8	12	3	1	1	87	55	37	52	52	0	SAVE



undo

マスクファイルを作成するためのタブです。

SAVE ボタンを押すと確認ダイアログが表示されます。

OK ボタンを押すと以下2つのファイルが作成され、キャンセルボタンを押すと保存をキャンセルします。

マスクファイルが既に存在する場合は、元々あるマスクファイルのバックアップを取った後に上書き保存されます。

マスクファイル

- mask.npy
- mask_params.json

※ バックアップは SAVE ボタン押下時の日付時刻がファイル名に付加されます。
マスクファイルは1度作成し、推論を行った場合上書きしないようにしてください。
上書きしてしまうと推論結果の再現が難しくなりますので、上書きしてしまった場合は推論をやり直す必要があります。

(例) 2019年11月29日14:31:01 に SAVE ボタンを押した場合のバックアップファイル

- `mask_20191129-143101.npy`
- `mask_params_20191129-143101.json`

Q&A

Q < 画像データフォルダの名前は『original』以外でも良いのですか？

A . いいえ、必ず『original』にしてください。一字一句違っていてもいけません。

例えば、良くない例として、

ORIGINAL (全て大文字)、Original (一部大文字)、original1 , original_ (飾り付け) ...
などがあります。

Q < 画像データの名前の付け方にも、気をつけるべき点がありますか？

A . あります。画像の名前は、「番号のみ(拡張子を除いて)」です。

もう一つの注意点として、その番号の「桁数」があります。

桁数は、同一画像データフォルダ内で統一してください。

例えば、良くない例として、

(0001(4桁).jpg 0002(4桁).jpg 10000(5桁).jpg)

のようにしてしまうのは、エラーの原因となります。このような場合は、

(00001(5桁).jpg 00002(5桁).jpg 10000(5桁).jpg)

と桁数が同じになるようにしましょう。

リンク

[Dash by plotly](#)

[Dash のユーザーガイド](#)