

**VIAME: AN OPEN-SOURCE FRAMEWORK FOR
UNDERWATER IMAGE AND VIDEO ANALYTICS**

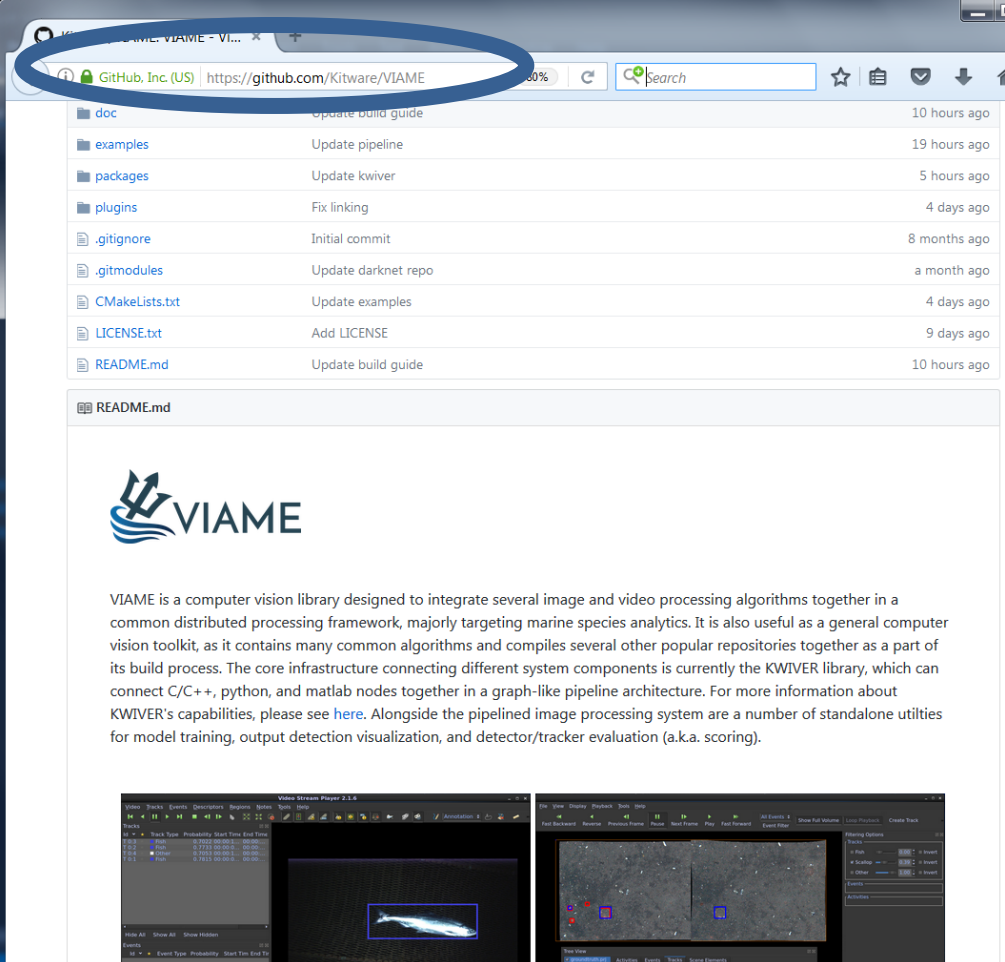
INSTALL GUIDE AND QUICK START MANUAL

VIAME

viametoolkit.org

<https://github.com/Kitware/VIAME>

- Video and Image Analytics for Multiple Environments
- Both installers (pre-built binaries) and source code are hosted on github.com
- Pre-built binaries are for users, the source code and build instructions are for developers

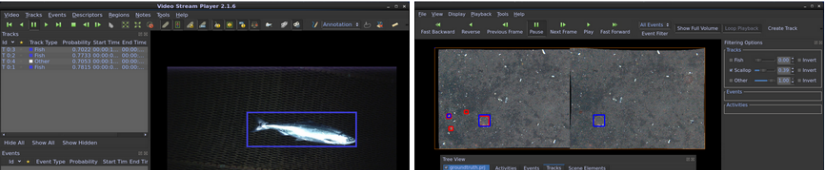


GitHub, Inc. (US) | <https://github.com/Kitware/VIAME>

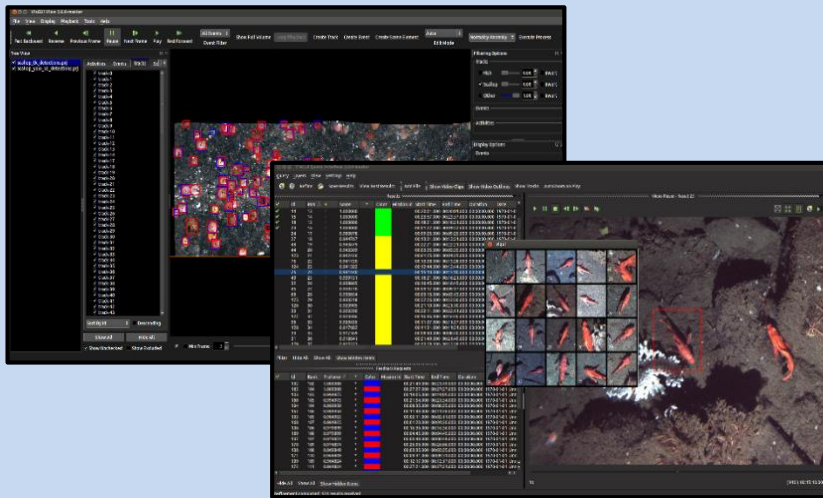
File/Folder	Commit Message	Time Ago
doc	Update build guide	10 hours ago
examples	Update pipeline	19 hours ago
packages	Update kwiver	5 hours ago
plugins	Fix linking	4 days ago
.gitignore	Initial commit	8 months ago
.gitmodules	Update darknet repo	a month ago
CMakeLists.txt	Update examples	4 days ago
LICENSE.txt	Add LICENSE	9 days ago
README.md	Update build guide	10 hours ago

VIAME

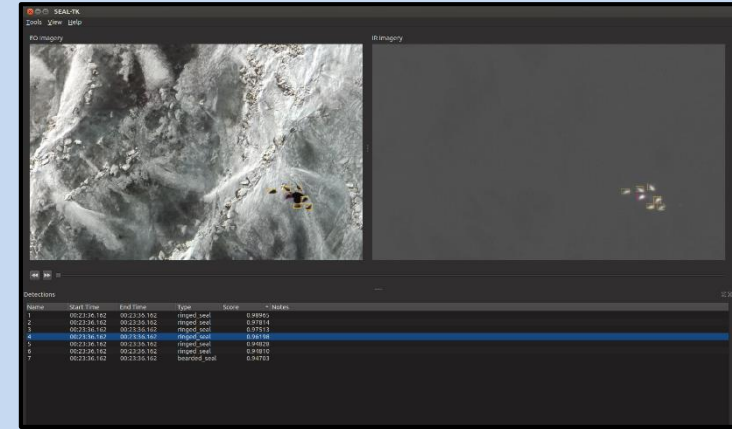
VIAME is a computer vision library designed to integrate several image and video processing algorithms together in a common distributed processing framework, majorly targeting marine species analytics. It is also useful as a general computer vision toolkit, as it contains many common algorithms and compiles several other popular repositories together as a part of its build process. The core infrastructure connecting different system components is currently the KWIVER library, which can connect C/C++, python, and matlab nodes together in a graph-like pipeline architecture. For more information about KWIVER's capabilities, please see [here](#). Alongside the pipelined image processing system are a number of standalone utilities for model training, output detection visualization, and detector/tracker evaluation (a.k.a. scoring).



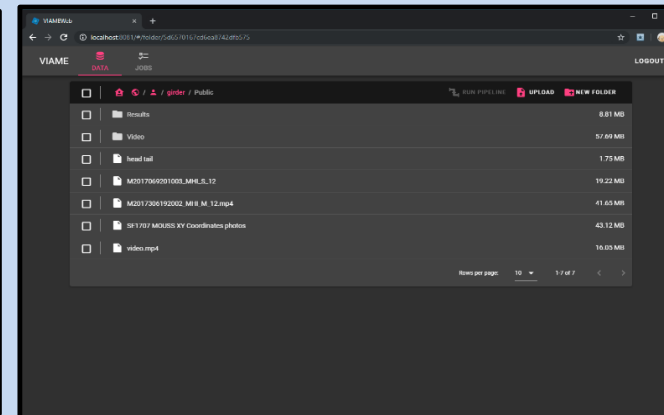
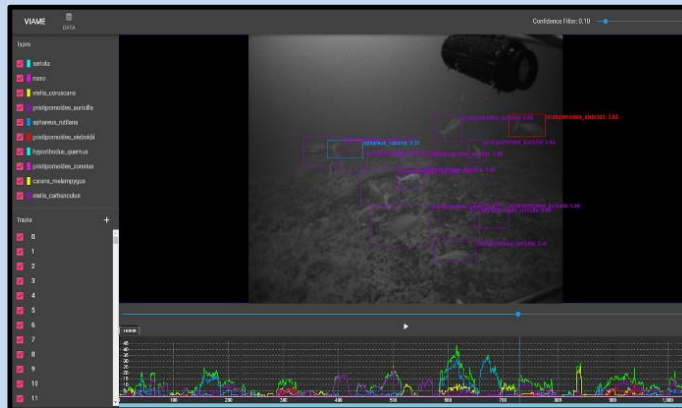
VIAME Flavors



Original Desktop Interfaces



Multi-Modality/Stereo Desktop Interface



Web Client Interface

Desktop Version Installation

Windows 7/8/10, 64-Bit

Requirements: Windows 7, 8, or 10, 64-Bit

Recommendations: NVIDIA GPU with ≥ 4 Gb Video RAM (partial image processing support),
 ≥ 8 Gb Video RAM (full image processing support)

A. Download Binaries

Goto: <https://github.com/VIAME/VIAME>

Download Correct Pre-Built Binary for your Operating System:

```
Pre-Built Binaries
To install, extract the binaries and place them in a directory of your choosing, for example C:\Program Files\VIAME on
Windows or /opt/noaa/viame on Linux. Next, set the PYTHON_INSTALL_DIR and CUDA_INSTALL_DIR variables at the top of
the setup_viame.sh (Linux) or setup_viame.bat (Windows) script in the root install folder to point to the location of your
installed Anaconda and CUDA distributions. Lastly, run through some of the examples to validate the installation.

Installation Requirements:
Anaconda3 5.2.0
CUDA 8.0.642

Installation Recommendations:
A CUDA-enabled GPU with 8 Gb or more VRAM

Linux Binaries:
VIAME v0.9.7 Ubuntu 16.04, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror1
VIAME v0.9.7 Ubuntu 16.04, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror2
VIAME v0.9.7 CentOS 7, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror1
VIAME v0.9.7 CentOS 7, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror2

Windows Binaries:
VIAME v0.9.7 Windows 7/8/10, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror1
VIAME v0.9.7 Windows 7/8/10, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror2
```

Binaries are currently large (~4Gb) due to the inclusion of multiple model files for training different methods.

B. Uninstall Previous Versions

Only perform this step if you have a previously installed version.

Typically located at C:\Program Files\VIAME

Remove this directory, optionally backing it up until you validate your new installation

Ubuntu 16.04/18.04 64-Bit, CentOS 7 64-Bit

Requirements: Ubuntu 16.04 or CentOS 7

Recommendations: NVIDIA GPU with ≥ 4 Gb Video RAM (partial image processing support),
 ≥ 8 Gb Video RAM (full image processing support)

Goto: <https://github.com/VIAME/VIAME>

Download Correct Pre-Built Binary for your Operating System:

```
Pre-Built Binaries
To install, extract the binaries and place them in a directory of your choosing, for example C:\Program Files\VIAME on
Windows or /opt/noaa/viame on Linux. Next, set the PYTHON_INSTALL_DIR and CUDA_INSTALL_DIR variables at the top of
the setup_viame.sh (Linux) or setup_viame.bat (Windows) script in the root install folder to point to the location of your
installed Anaconda and CUDA distributions. Lastly, run through some of the examples to validate the installation.

Installation Requirements:
Anaconda3 5.2.0
CUDA 8.0.642

Installation Recommendations:
A CUDA-enabled GPU with 8 GB or more VRAM

Linux Binaries:
VIAME v0.9.7 Ubuntu 16.04, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror1
VIAME v0.9.7 Ubuntu 16.04, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror2
VIAME v0.9.7 CentOS 7, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror1
VIAME v0.9.7 CentOS 7, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror2

Windows Binaries:
VIAME v0.9.7 Windows 7/8/10, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror1
VIAME v0.9.7 Windows 7/8/10, 64-Bit, GPU Enabled, CUDA 8.0, Python 3.6, Mirror2
```

Binaries are currently large (~4Gb) due to the inclusion of multiple model files for training different methods.

Only perform this step if you have a previously installed version.

Typically located at /opt/noaa/viame

Remove this directory ('rm -rf /opt/noaa/viame'), optionally backing it up until you validate your new installation

To optionally backup, open terminal:

```
cd /opt/noaa
mv viame viame-bckup
```

After completing installation remove old version:

```
rm -rf /opt/noaa/viame-bckup
```

Desktop Version Installation

C. Install Dependency – NVIDIA Drivers

Only perform this step if you don't have CUDA or appropriate NVIDIA drivers installed ahead of time and are using GPU-enabled binaries.

Drivers can be found here:

<https://www.nvidia.com/Download/index.aspx?lang=en-us>

(Version 410.48 or above is required for installation)

Or alternatively get CUDA (installing CUDA is no longer required, even though it use to be, only the drivers are, but they are included in CUDA):

*Windows 7, unlike 8 and 10, requires some updates and service packs installed, e.g. [KB2533623](#) alongside drivers or else you will get errors using GPU-dependent code.

Only perform this step if you don't have CUDA or appropriate NVIDIA drivers installed ahead of time. Drivers can be found here:

https://linuxhint.com/ubuntu_nvidia_ppa/ (Ubuntu)

<https://www.nvidia.com/Download/index.aspx?lang=en-us> (Other)

(Version 410.48 or above is required for installation)

Or alternatively get CUDA (CUDA is no longer required, only the drivers are, even though it use to be, but they are included in CUDA):

Goto: <https://developer.nvidia.com/cuda-toolkit-archive>

The best way to install the drivers depends on your operating system. We recommend using package managers (like the above PPA for Ubuntu) when able, but if that fails falling back to one of NVIDIA's standalone installers.

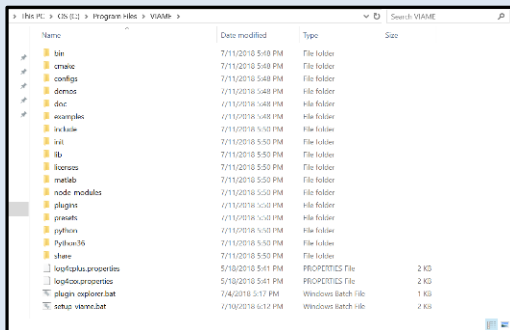
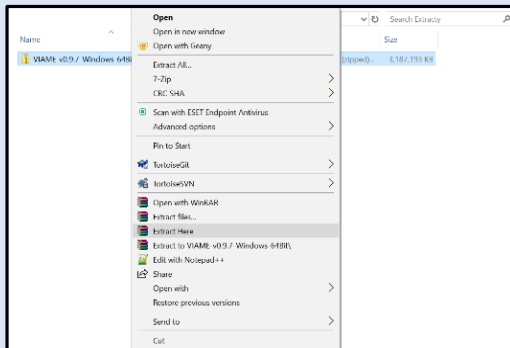
Desktop Version Installation

D. Extract Downloaded VIAME Binaries

Choose an installation directory for VIAME.

We recommend C:\Program Files\VIAME, from here on out this will be known as [viame-install]

Extract the binaries from step A, for example if using winrar select the below, or alternatively 'Extract All' using the default windows option:



Installation Complete

Choose an installation directory for VIAME.

We recommend /opt/noaa/viame, from here on out this will be known as [viame-install]

Extract the binaries from step A, for example, right click on the downloaded .tar.gz binaries file and click "Extract Here"

The alternative is to untar the file on the command line type:

```
tar -xvf VIAME-v*-Ubuntu-64Bit.tar.gz
```

Navigate to the folder with the extracted 'viame' folder and move it to [viame-install], for example:

```
mkdir -p /opt/noaa/  
mv viame /opt/noaa
```

The contents of the folder should look like the below.



Depending on your system, you may need to get permission to modify your install directory (e.g. /opt/noaa/viame)

Installation Complete

Platform Add-Ons

Linux Desktop Binaries:

VIAME v0.9.13 Ubuntu 16.04/18.04, 64-Bit, GPU Enabled, CUDA 9.0, Python 3.6, Mirror1
VIAME v0.9.13 Ubuntu 16.04/18.04, 64-Bit, GPU Enabled, CUDA 9.0, Python 3.6, Mirror2
VIAME v0.9.13 RHEL/CentOS 7, 64-Bit, GPU Enabled, CUDA 9.0, Python 3.6, Mirror1
VIAME v0.9.13 RHEL/CentOS 7, 64-Bit, GPU Enabled, CUDA 9.0, Python 3.6, Mirror2

Windows Desktop Binaries:

VIAME v0.9.14 Windows 7*/8/10, 64-Bit, GPU Enabled, CUDA 9.0, Python 3.6, Mirror1
VIAME v0.9.14 Windows 7*/8/10, 64-Bit, GPU Enabled, CUDA 9.0, Python 3.6, Mirror2
VIAME v0.9.9.7 Windows 7*/8/10, 64-Bit, CPU Only, Python 3.6, Mirror1
VIAME v0.9.9.7 Windows 7*/8/10, 64-Bit, CPU Only, Python 3.6, Mirror2

*Windows 7 requires some updates and service packs installed, e.g. KB2533623.

Web Applications:

[Experimental Online Annotator](#)

Optional Patches:

[MOUSS Model Set 1 \(Deep 7 Bottomfish\) Add-On, All OS](#)
[MOUSS Model Set 2 \(Deep 7 Bottomfish\) Add-On, All OS](#)
[MOUSS Sample Project, All Linux](#)
[Arctic Seals Models Add-On, All Linux](#)
[HabCam Models \(Scallop, Skate, Flatfish\) Add-On, All OS](#)
[Alternative Generic Detector for IQR Add-On, All OS](#)
[Low Memory GPU \(For 4+ Gb Cards\) Add-On, All OS](#)

Standalone Tools:

[Sea Dual Display GUI, Windows 7/8/10, 64-Bit](#)

Note: To install Add-Ons, copy them into your install. To use project files extract them into your working directory of choice.

Quick Run Instructions

Add-ons contain extra model files for particular problems, extra custom project files (see preceding slides) and standalone applications for specific problems.

To install model files, extract them in a similar fashion to the installation binaries and then copy them into your install tree (make sure to match the high level folder, e.g. don't copy the VIAME folder in the patch to VIAME/VIAME, in the install make sure that the contents of the 'VIAME' patch folder go in the 'VIAME' install folder in matching directories).

Models in them will then show up in the GUIs or also be runnable on the command line.

To install project files extract in directory of choice, to install custom executables instruct then run the launch scripts in the high-level directory of the package.

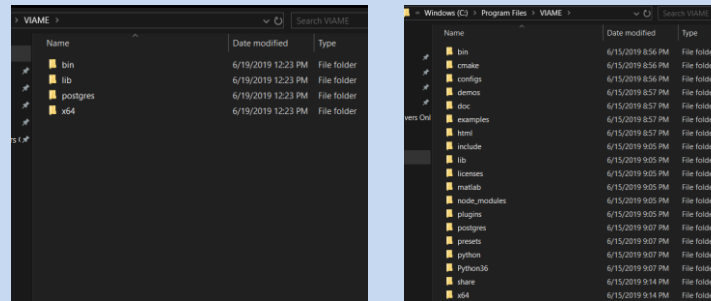
Common Install Issues

```
C:\Program Files\VIAME\examples\annotation_and_visualization>launch_annotation_interface.bat
'python.exe' is not recognized as an internal or external command,
operable program or batch file.
C:\Program Files\VIAME\examples\annotation_and_visualization>_
```

Your python install directory is incorrect, see slide 7.

```
RuntimeError: CUDA out of memory. Tried to allocate 26.63 MiB (GPU 0; 2.00 GiB total capacity; 1.29 GiB already allocated; 18.87 MiB free; 55.08 MiB cached)
Press any key to continue . . .
```

Your GPU isn't very powerful, or maybe you're running too many things at once, or have an old process which has stuck around. Maybe try logging out and in again if you suspect the later.



Your install looks like the left on windows, not the right. Something went wrong and your download and extraction of the binaries was only partially complete (Low disk space? Dropped connection?), redownload and re-extraction is recommended.

```
libpython3.6m.so.1.0: cannot open shared object file: No such file or directory
OR
Unable to load shared library "E:\VIAME\lib\kwiver\processes/modules_python.dll
OR
Exception caught: There is no such process of type 'resnet_descriptors' in the registry,
```

You probably installed the wrong version of Anaconda, see slide 6.

If you get any run-time issues mentioning failing to import numpy as well this is usually the case. 8

Ways to Run System Components

[viame-install] = Where you installed the software to (C:\Program Files\VIAME, /opt/noaa/viame)

(1) Running examples in the [viame-install]/examples sub-folder

The full manual for these is available here: <https://viame.readthedocs.io/en/latest/>

And are mirrored in the comments of the github repository here: <https://github.com/VIAME/VIAME/tree/master/examples>

Both the manual and corresponding examples are broken down by functionality (annotation, object detection, measurement, etc...). This method can be useful if you just want to run one component in a standalone capacity without much effort.

(2) Project files located in [viame-install]/configs/prj-(linux or windows)/*

[RECCOMENDED]

Useful for bulk processing data, as well as training object detection and tracking models with three to four different techniques. Contains run scripts for multiple functions in the same folder. Can be easily copied to a new directory, e.g. your computer's desktop (or anywhere else of choice) and run from there.

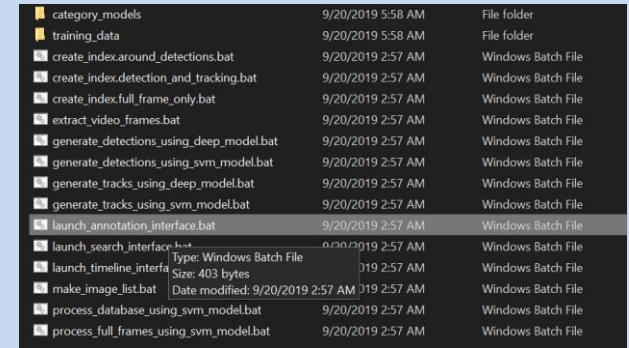
How to Run Assorted Functionality

On Windows:

All examples and project file launchers are .bat scripts

You just need to double click them to run them

(sometimes a security dialogue will pop up but just ignore them)



On Linux:

You may need to know 3 command line commands:

'**bash**' - for running commands, e.g. 'bash run_annotation_gui.sh' which launches the application

'**ls**' - for making file lists of images to process, e.g. 'ls *.png > input_list.txt' to list all png image files in a folder

'**cd**' - go into an example directory, e.g. 'cd annotation_and_visualization' to move down into the annotation_and_visualization example directory. 'cd ..' is another useful command which moves one directory up, alongside a lone 'ls' command to list all files in the current directory.

Alternatively, you can make .sh scripts double-clickable via:

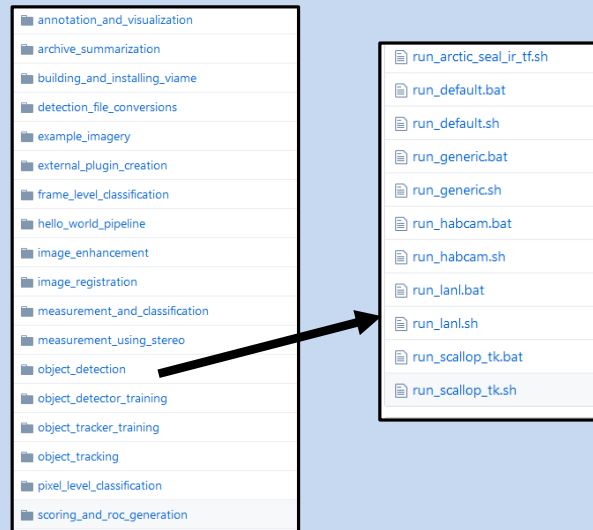
<https://askubuntu.com/questions/138908/how-to-execute-a-script-just-by-double-clicking-like-exe-files-in-windows>
to not have to deal with launching things from the terminal.

As a second alternative you can make shortcuts to your VIAME annotation GUI and common scripts:

<https://askubuntu.com/questions/854373/how-to-create-a-desktop-shortcut>

How to Run From Examples Folder

- (1) Go to the examples directory
- (2) Choose your example
- (3) Read the corresponding entry in either the manual of example webpage
- (4) Run the example scripts

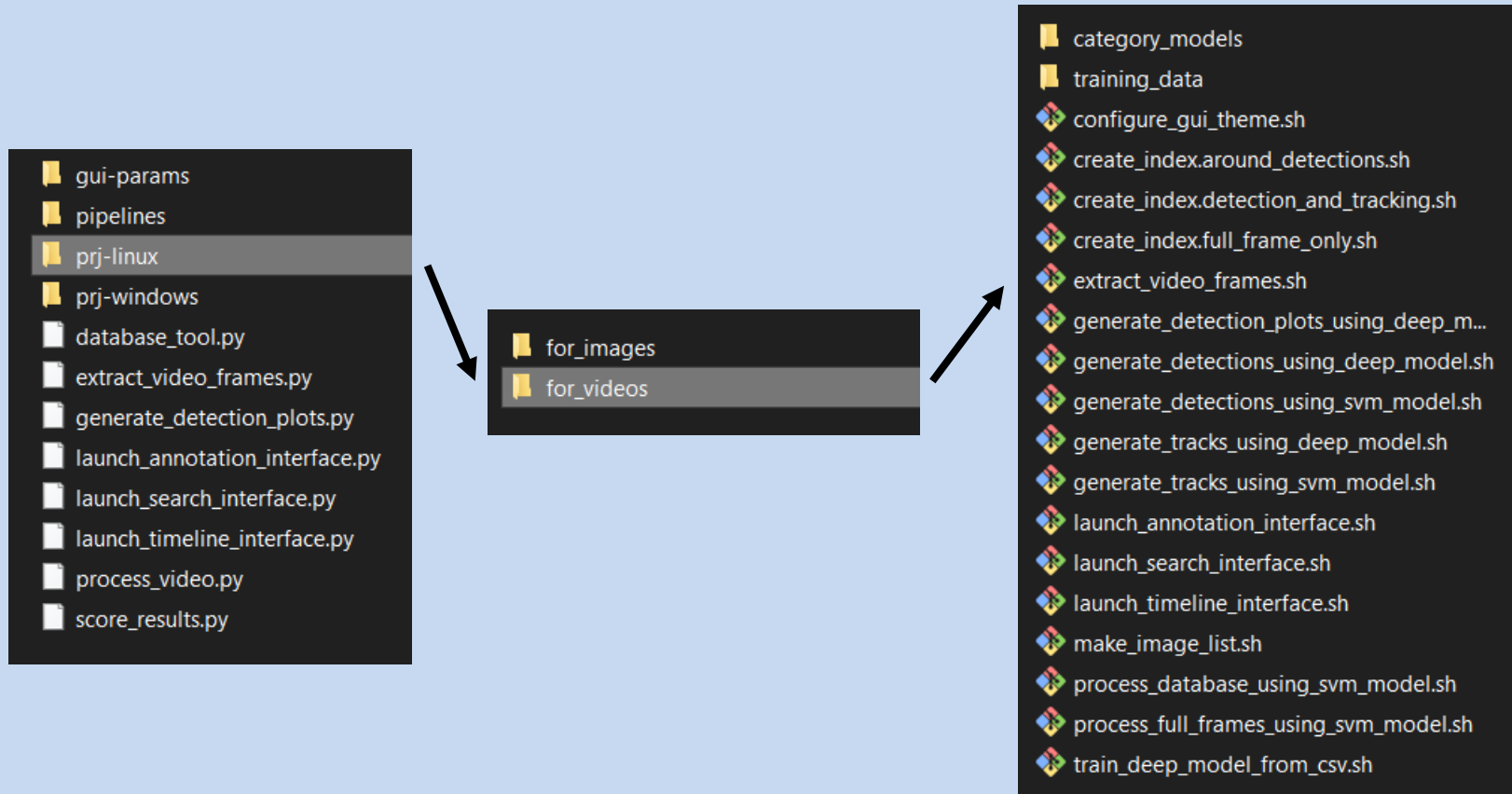


In some examples knowing how to make image lists are also helpful for some examples. There are scripts showing how to do this in the project files which can be edited in notepad++ if necessary on Windows. On Linux running 'ls [path]*.png > input_list.txt' will generate a list as well, for a directory of .pngs in 'path'. You can also mount network drives and do your processing off them.

Example run files can also be run from different directories if you open up the launch script in a notepad editor and change the 'VIAME_INSTALL' path to point to your VIAME installation directory.

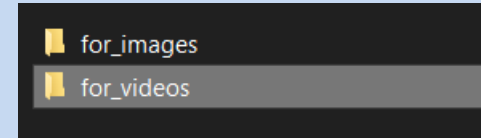
Using Project Files

Project files are broken down based on operating system (Windows vs Linux/Mac) then class, in [viame-install]/configs:



Lot of launchers here, but more on that later....

Using Project Files

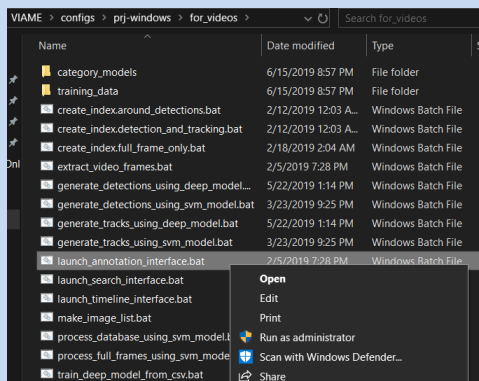


for_images - is designed to process a single list of imagery at time
(except for training detectors which can be on multiple lists)

for_videos - is designed to process either a folder of videos, a single video, or a folder of folders of images

To use them copy them to a folder you want to work out of, hopefully one with a bit of disk space if you plan on doing model training.

Possible requirement: if you installed VIAME to a non-default location, in order to run any of the scripts you will need to change VIAME_INSTALL at the top of your run script to point to your installation, i.e.:



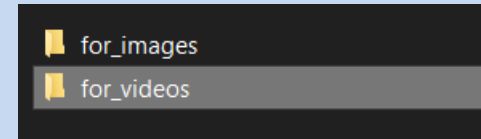
```
launch_search_interface.bat - Notepad
File Edit Format View Help
@echo off

REM Path to VIAME installation
SET VIAME_INSTALL=C:\Program Files\VIAME

REM Setup paths and run command
CALL "%VIAME_INSTALL%\setup_viame.bat"

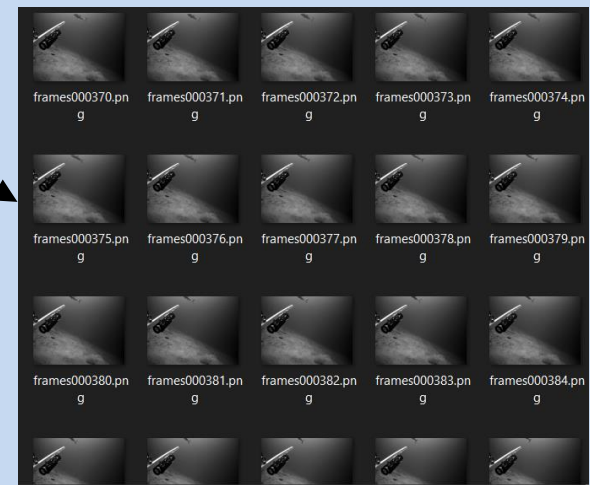
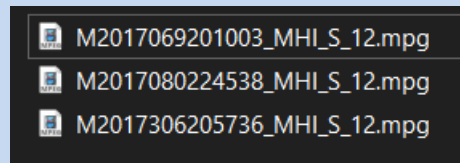
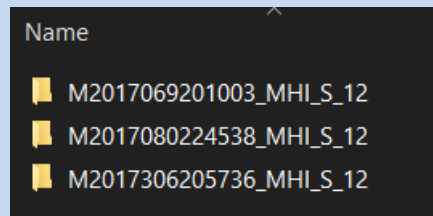
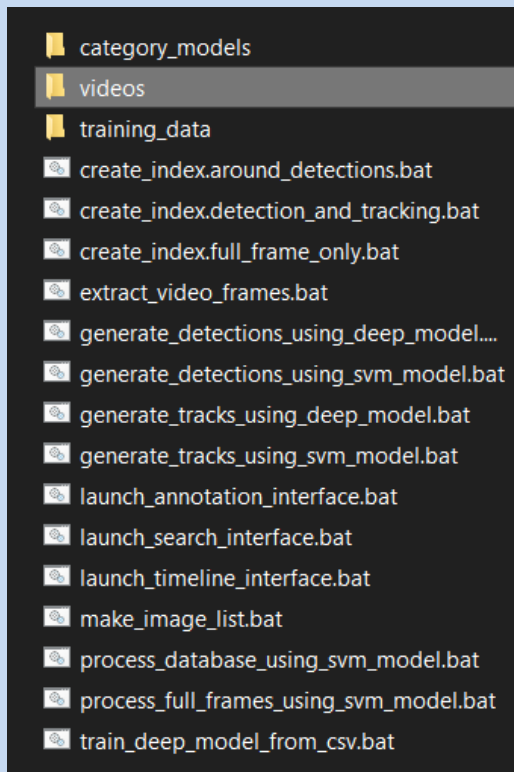
python.exe "%VIAME_INSTALL%\configs\launch_search_interface.py" ^
  -qp pipelines\query_retrieval_and_iqr.pipe
```

Using Project Files

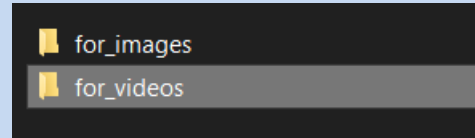


The video project file processes a folder of videos called 'videos' in the project directory by default. You can alternatively let 'videos' be a shortcut to a network drive or other non-local location (e.g. an external hard drive). Or have the project file be on the alternative drive or network.

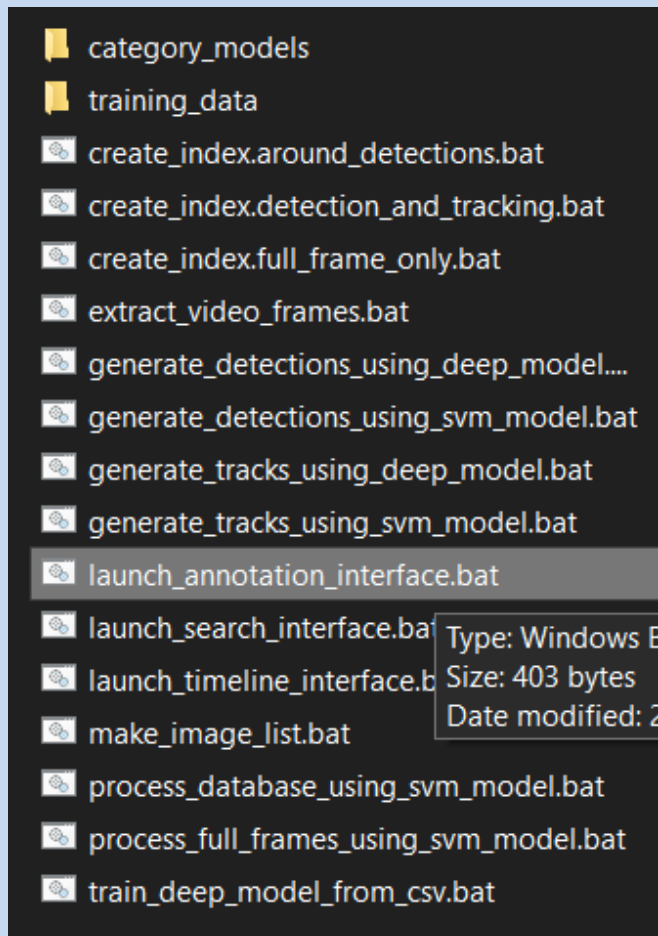
'videos' can also be a directory of directory of images



Using Project Files

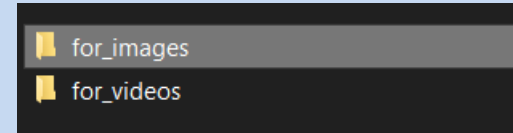


Also in project files (for videos) are frame rates to process videos at, these can be changed along with the video input directory. The later can be modified to point to other locations on disk or network as desired.

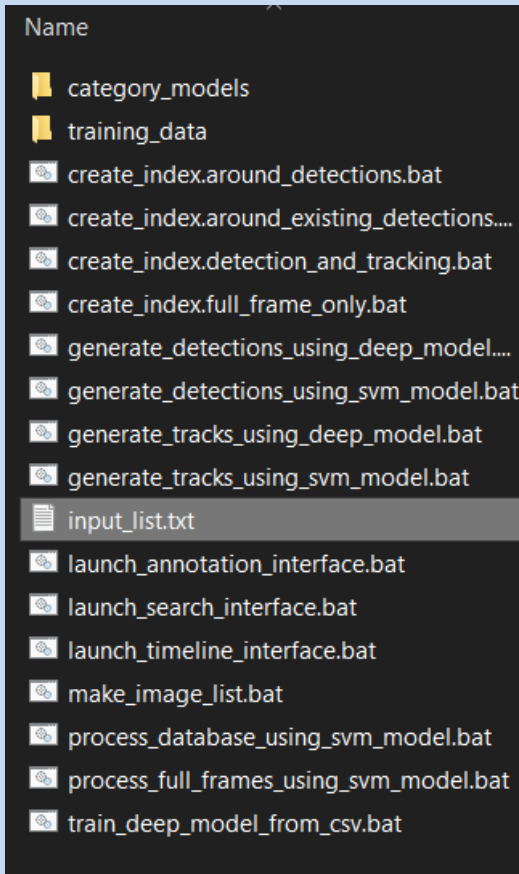


```
change.log x make_image_listbat x launch_annotation_interface.bat x
1 @echo off
2
3 REM Path to VIAME installation
4 SET VIAME_INSTALL=C:\Program Files\VIAME
5
6 REM Processing options
7 SET INPUT_DIRECTORY=videos
8 SET CACHE_DIRECTORY=database
9 SET FRAME_RATE=5
10
11 REM Setup paths and run command
12 CALL "%VIAME_INSTALL%\setup_viame.bat"
13
14 python.exe "%VIAME_INSTALL%\configs\launch_annotation_interface.py"
15
```

Using Project Files

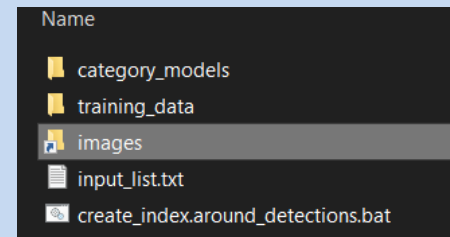


The image project file processes a list of images called “input_list.txt” at either train (model generation) or test time (model application). This file points to imagery to process, 1 file per line.



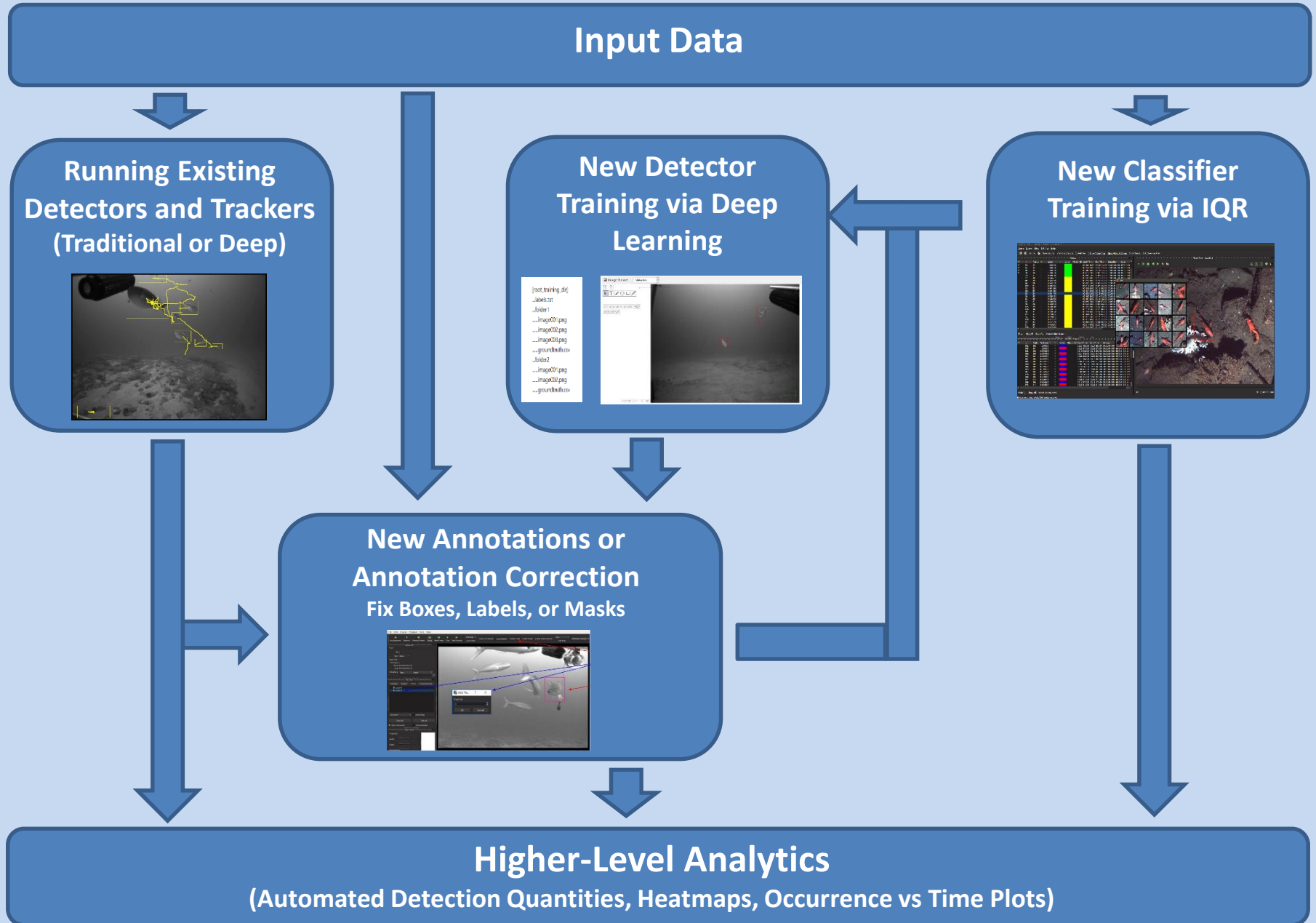
```
change.log x make_image_list.bat x
1
2 SET INPUT_DIRECTORY=images
3
4 dir /s/b "%INPUT_DIRECTORY%\*" > input_list.txt
5
```

On windows, the ‘make_image_list.bat’ command can be used to make this list if you put your images in a folder called ‘images’ then double click make_image_list.bat; similarly on Linux the ‘ls’ command can be used. The glob pattern can also be changed from ‘*’ to ‘*.png’ to list images of only a certain type and not all files in the directory.



Example link to image directory stored on external drive

3 Core Possible Detector Training Workflows



3 Core Detection Training Workflows

Workflow #1: Traditional Deep Learning Model Generation from Scratch.

Process:

- (a) Load up Imagery in Annotator
- (b) Annotate Imagery Manually
- (c) Export Detection or Tracks Files
- (d) Repeat for as Many Sequences as Possible
- (e) Run Model Training
- (f) Evaluate Model Performance
- (g) Repeat (d) thru (f) as Desired on Detector Fail Cases
 - Focusing additional annotation on sequences with the most errors

Pros:

- Models perform better than most other solutions when trained with enough training data

Cons:

- Requires a large amount of training data and user time to generate it

3 Core Detection Training Workflows

Workflow #2: Traditional Deep Learning Model Generation with Partial Automation

Process:

- (a) Load up Imagery in Annotator
- (b) Run an Automated Detector
 - Can be IQR based
 - Can be default model or other pre-trained detector
 - Can be a user generated deep detector
- (c) Correct and Export Detection or Tracks Files
- (d) Repeat for as Many Sequences as Desired
- (e) Run Model Training
- (f) Evaluate Model Performance
- (g) Repeat (b) thru (f) as Desired on Detector Fail Cases

Pros:

- Can speed up annotation if automated detector is decent enough

Cons:

- If automated detector is poor it can take more effort to correct automated outputs instead doing annotations from scratch

3 Core Detection Training Workflows

Workflow #3: IQR (Video Search with Adjudication) for Rapid Model Generation

Process:

- (a) Create searchable index for a video archive
- (b) Launch search GUI
- (c) Use search GUI to generate IQR (.svm) models
- (d) Save models to category directory
- (e) Evaluate models

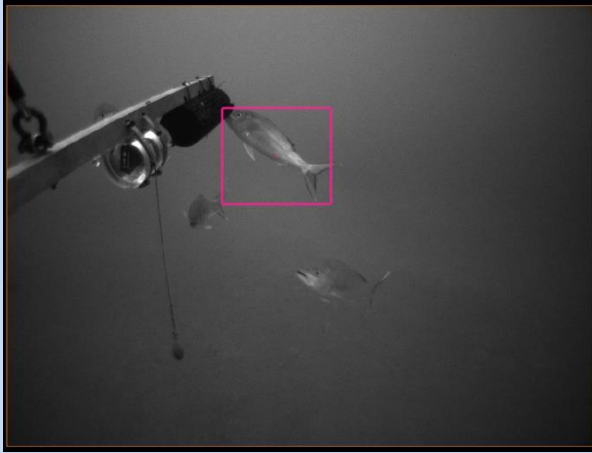
Pros:

- Can be done with very little user effort, mostly computer runtime
- Can be used to rapidly generate models for new classes

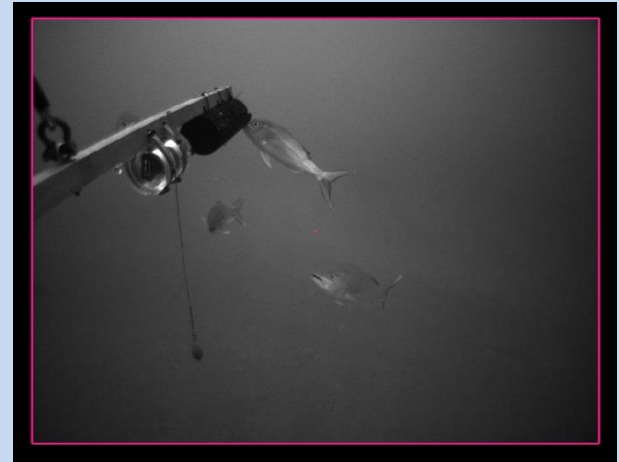
Cons:

- GUI generally crashes after about 6 iterations due to memory issues
- Models generally not as good as deep models trained on enough training data (but can be better for cases with not a lot of training data, often, depends on the exact case)

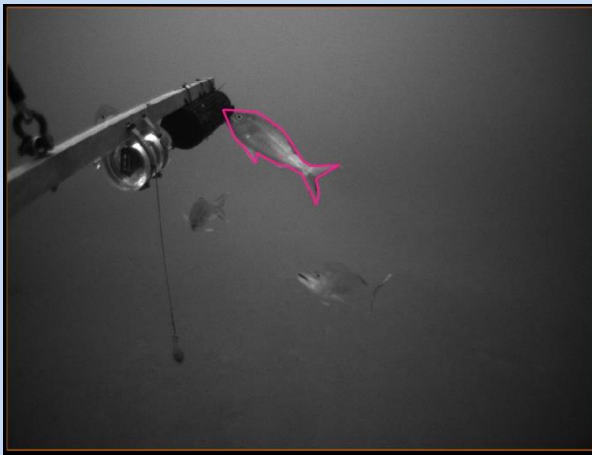
Types of Annotation and Detection Models



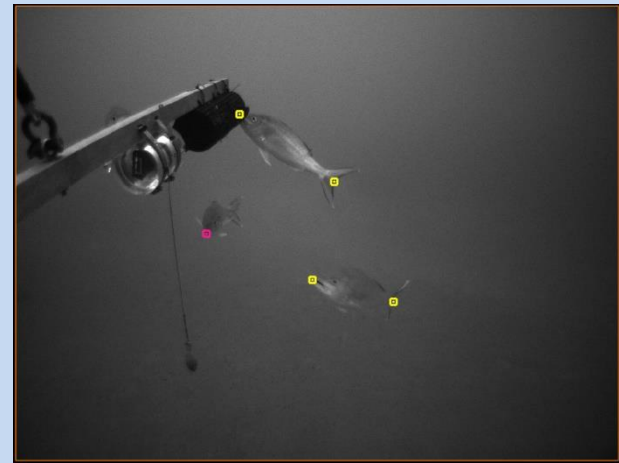
Box-Level



Frame-Level



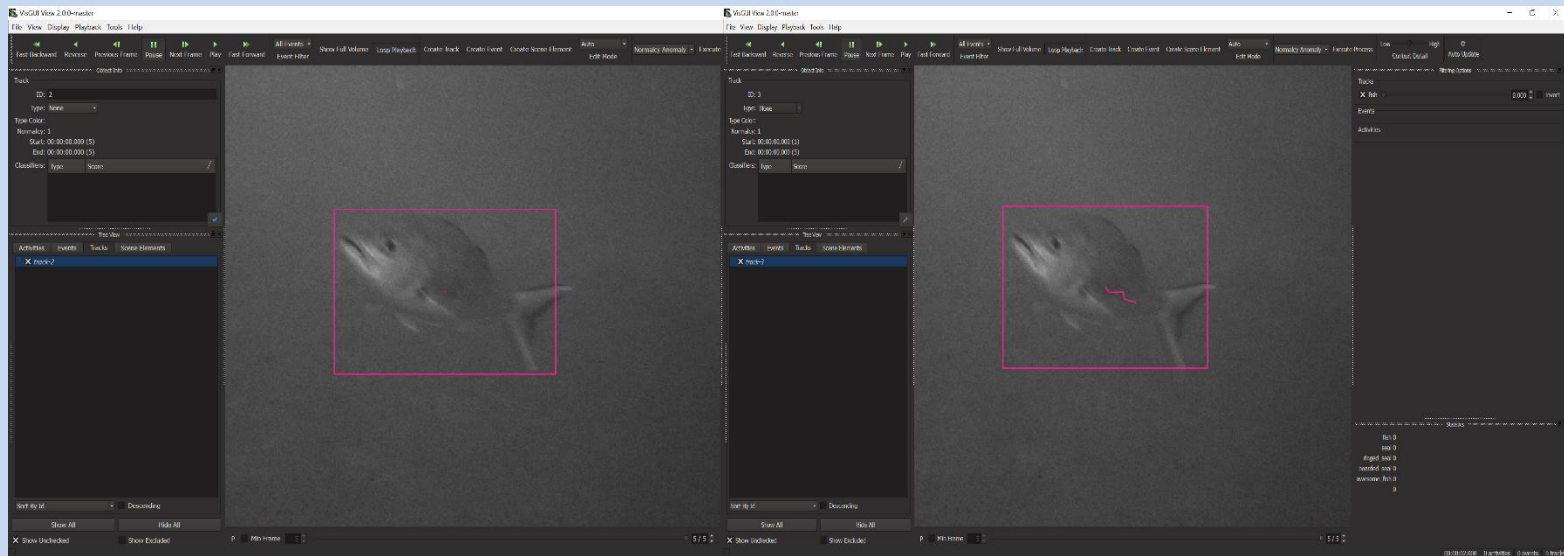
Pixel-Level



Keypoints

Detections vs Tracks

Detections and tracks are synonymous across examples and user-interfaces. A track is a (temporal) sequence of single-frame detections, but a detection can also be viewed as a track with just a single state.



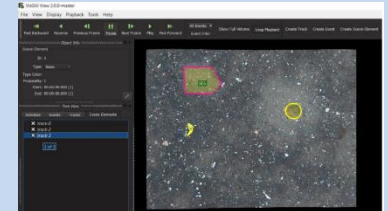
Detection

Track

How to Perform Annotation

(for workflows #1 and #2)

Annotation should be performed in the format of what you want your classifier to output. For example, if you only care about full frame properties, you should only annotate full frame properties



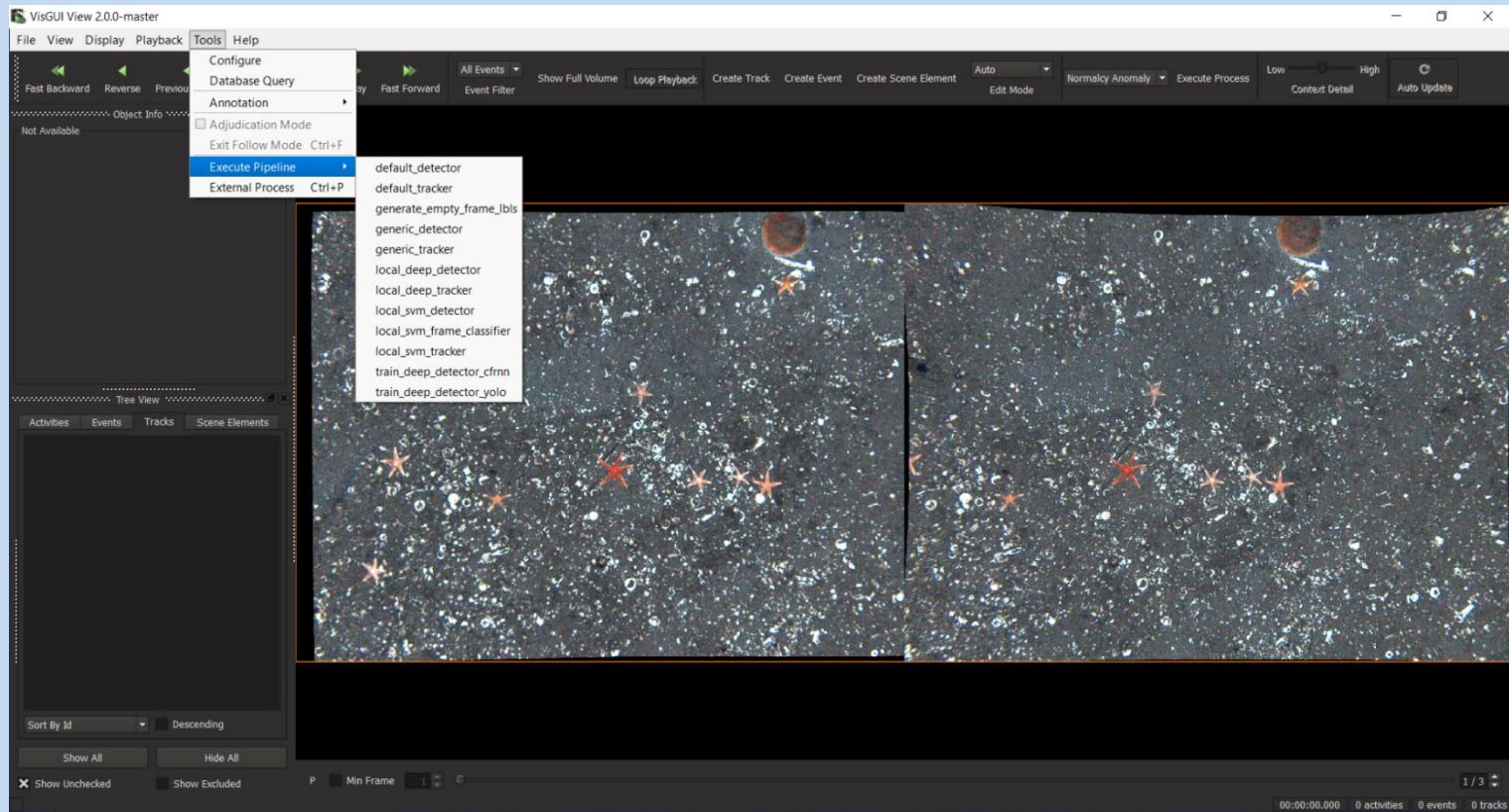
If you want to perform species-level classification, you need to assign types to your detections or tracks, the type being the species or subspecies identifier. Annotating at a finer level is usually best, as later you can then train on the merging of lower-level categories, but still have the option to train at the finest level.

A full annotation guide and annotation example videos can be found below, we recommend reading through them and learning how to use the annotator:

https://viame.readthedocs.io/en/latest/section_links/annotation_and_visualization.html

In the project file, just like in the examples, 'launch_annotation_gui.bat' launches the annotator.

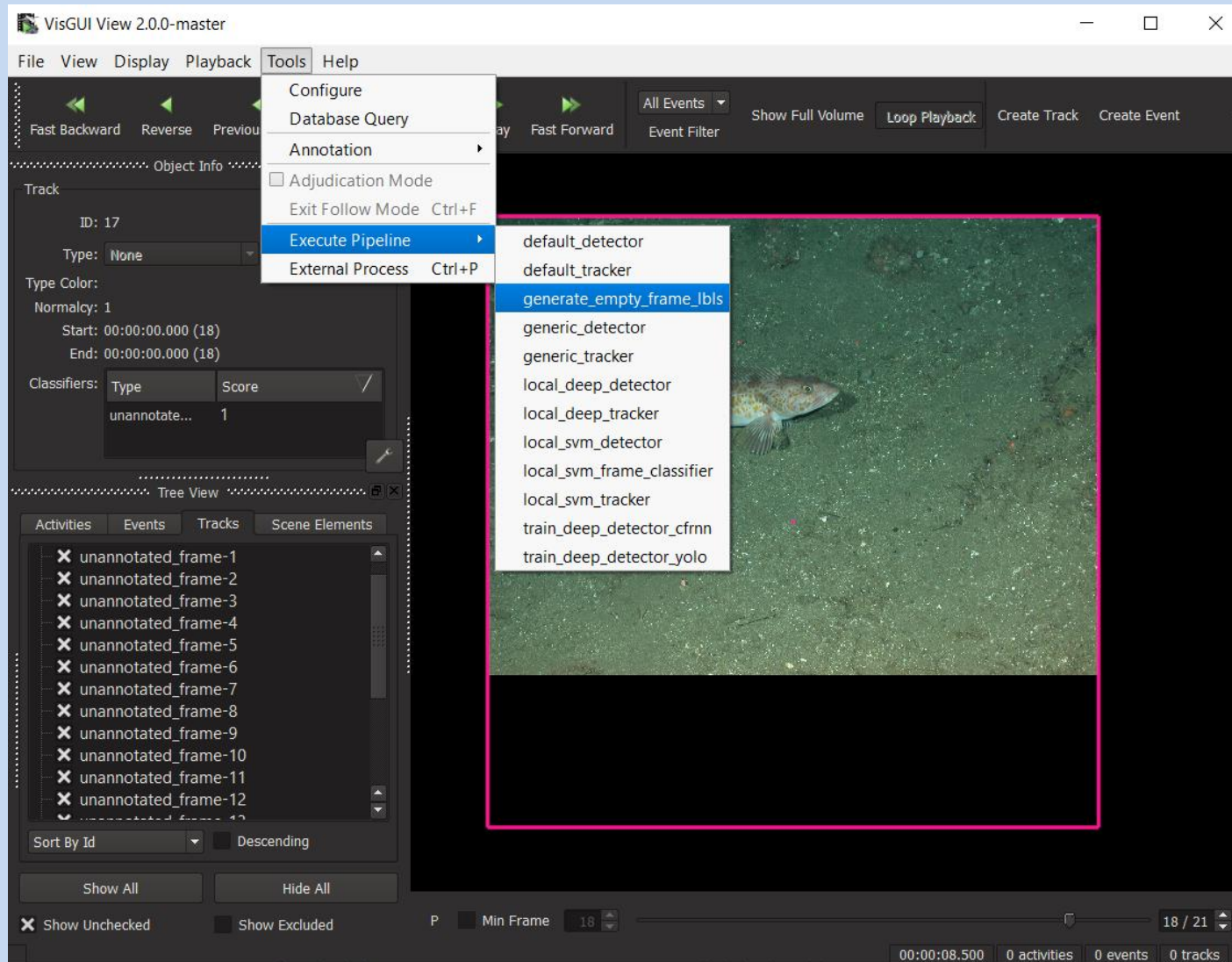
Running Trained Detector Pipelines



Detector pipelines can be run from the GUI on a single sequence or video of data via the Tools->Execute Pipeline->[pipeline] option, as shown above; results then appear in the GUI.

Detector pipelines can be run from the project file in batch over multiple sequences and then the result of each sequence opened in the annotation GUI automatically

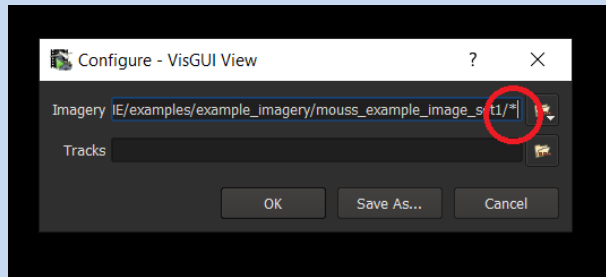
Running Trained Detector Pipelines



Empty full frame labels can be generated via this pipeline

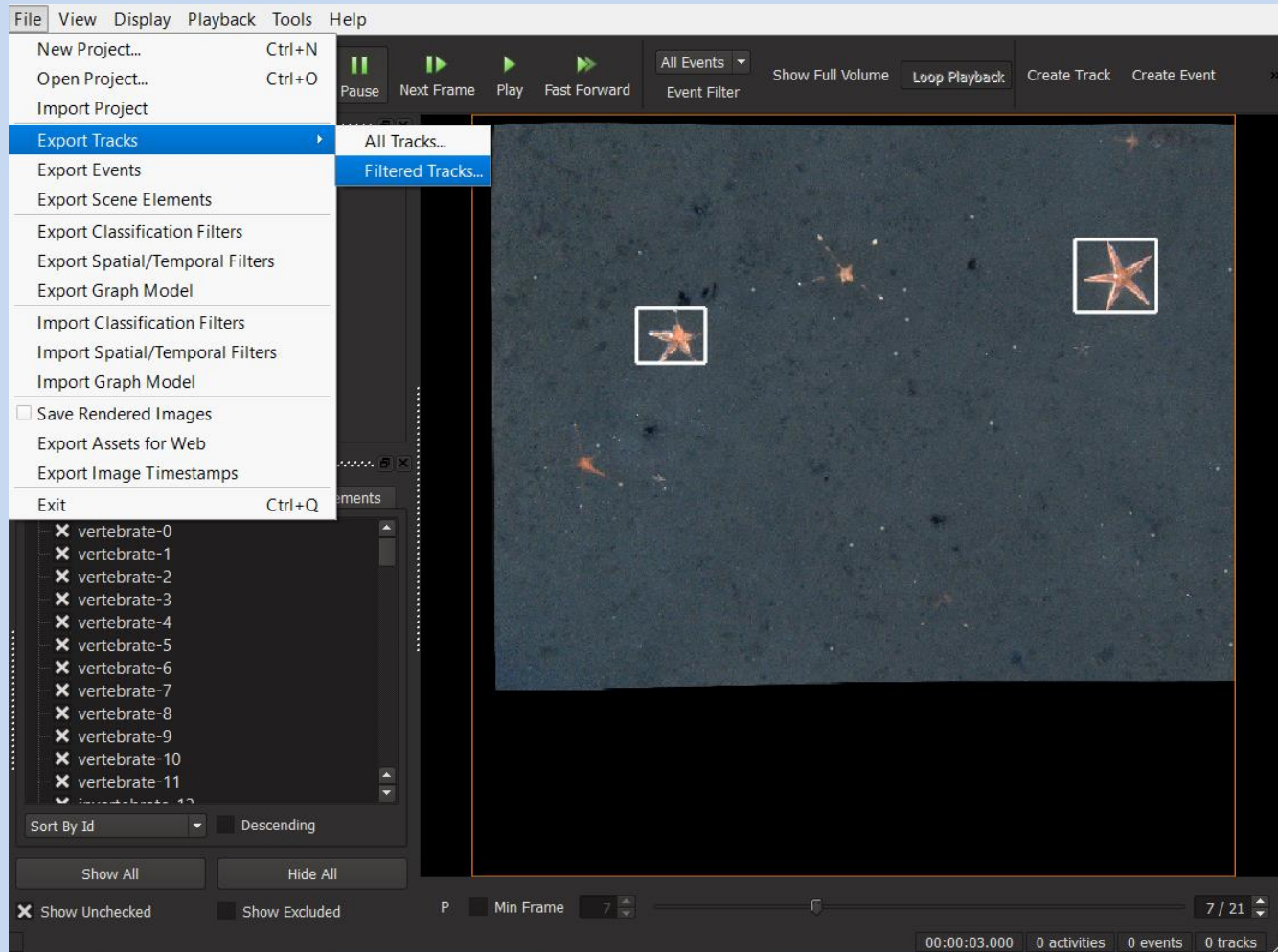
Annotation Tips n' Tricks

- When you select a directory of images to process a glob pattern appears in the dialogue (below) as “*”. “*” means try to open any file in the directory, but if you have non-images in a directory it can also be helpful to put “*.png” or “*.sgi” to only load images of a known output extension. This can also be used to select only the left or right camera if you have stereo data in the same folder.



- VIAME provides 3 main pre-trained pipelines: a default, generic, and motion pipeline.
 - Default: Detect vertebrate and invertebrates using a default per-image CNN
 - Generic: Put boxes around all arbitrary objects in an image using a CNN
 - Motion: Put boxes around all moving targets using traditional approaches
- These models can be run from the GUI in either per-frame detection-only or tracking form
- Other pipelines runnable from the GUI allow the running of local (user-trained) models or ones generated from IQR rapid model generation. VIAME add-ons add more runnable models from this dialogue as well.
- Pixel level classification via polygons currently only works with scene element annotations right now, track polygons are broken

Annotation Tips n' Tricks



When saving out detections/tracks they can be saved out either as filtered tracks or non-filtered tracks. Filtered tracks will have the threshold you set in the filtering options enabled and is generally the way to go if correcting detectors.

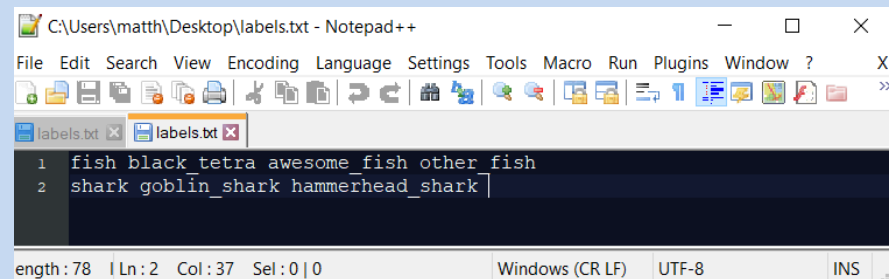
3 Core Detection Training Workflows

In workflows #1 and 2 model training is performed via saving your annotations and source data (or creating shortcuts to it) into the 'training_data' folder in a project file. The folder structure required for this is shown in the below example and also below:

https://github.com/VIAME/VIAME/tree/master/examples/object_detector_training

Project files use the same data partition strategy, just in their 'training_data' folders, then the train_deep_model_[arch].sh/.bat scripts can be used to train models in bulk, where [arch] is the desired detection architecture.

```
[root_training_dir]
...labels.txt
...folder1
.....image001.png
.....image002.png
.....image003.png
.....groundtruth.csv
...folder2
.....image001.png
.....image002.png
.....groundtruth.csv
```



```
C:\Users\matth\Desktop\labels.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X
labels.txt x labels.txt x
1 fish black tetra awesome fish other fish
2 shark goblin_shark hammerhead_shark
length: 78 | Ln: 2 Col: 37 Sel: 0 | 0 Windows (CR LF) UTF-8 INS
```

The labels.txt contains 1 line per desired output category and a list of synonyms on the same line, e.g. the above would produce a detector which produces 2 outputs: fish and shark, but including the other categories in those super-categories.

Supported Detector Architectures

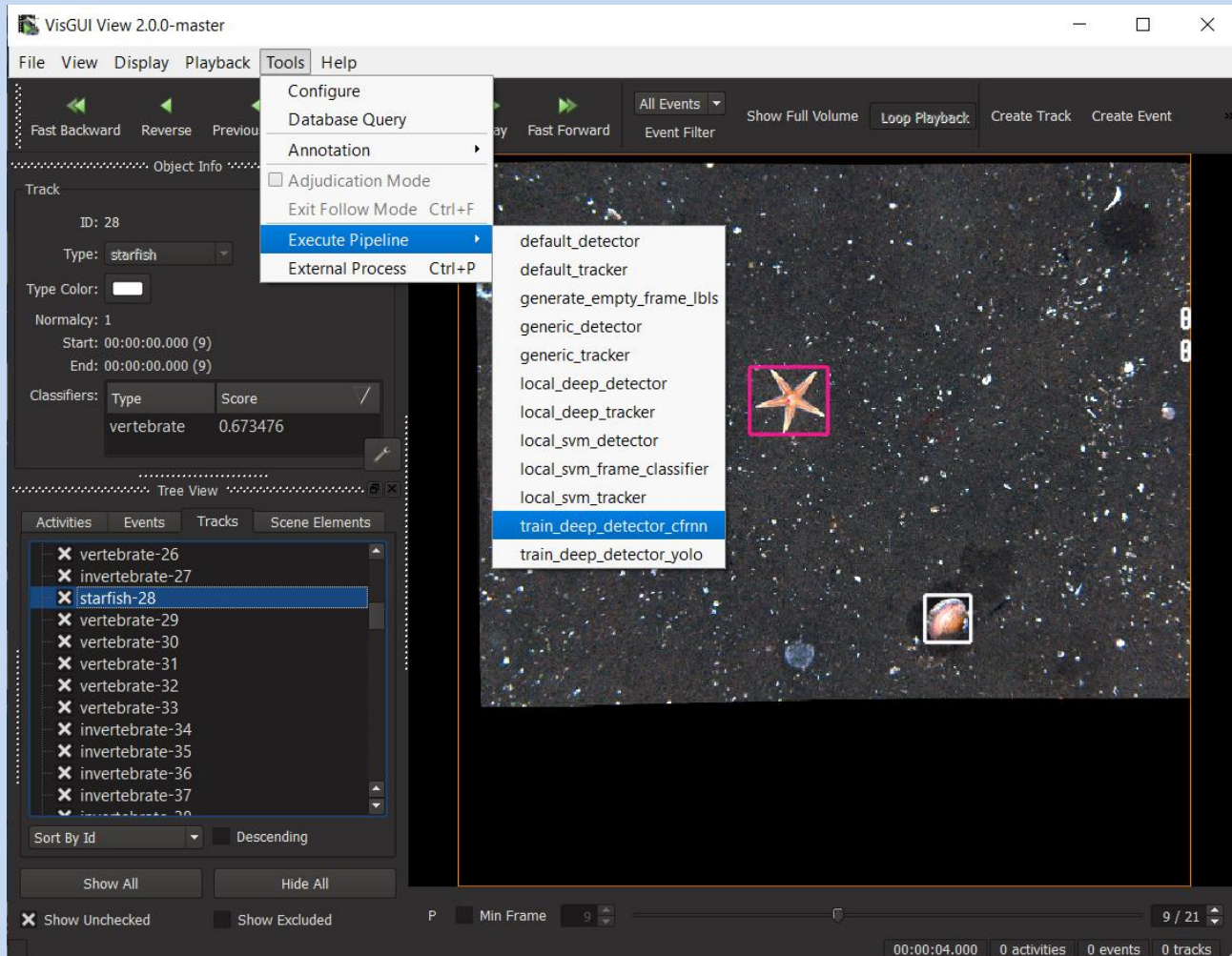
There are several detection and classification models in the system:

- YOLOv2, Gridded YOLOv2, YOLOv3, Gridded YOLOv3, YOLO-WTF
- Cascade Faster RCNN, Mask Faster RCNN
- Scallop-TK
- ResNet50 Full Frame Classifier
- SVM Models on Top of ResNet50 Late Features
- AdaBoost on top of Manual Features for Pixel Classification

Cascade FRNN (CFRNN) is typically more accurate than YOLO, but slower and requires much more annotations to train successfully. The SVM models require the least number of annotations to train successfully, but is dependent on the off-the-shelf generic object detector producing decent results for your problem. It can often be hard to know which method is best without looking at the problem and trying different techniques.

The with temporal features (WTF) variants are typically better at detecting small movers in the background and worse at using color as a discriminator between categories, but they are still experiment. Unlike the others, Mask RCNN and AdaBoost methods also performs pixel-level classification.

Training from Annotation GUI



Models can be trained from the annotation GUI, but currently only on a single sequence, unlike the examples and project files. It will use all loaded data and all categories in the sequence.

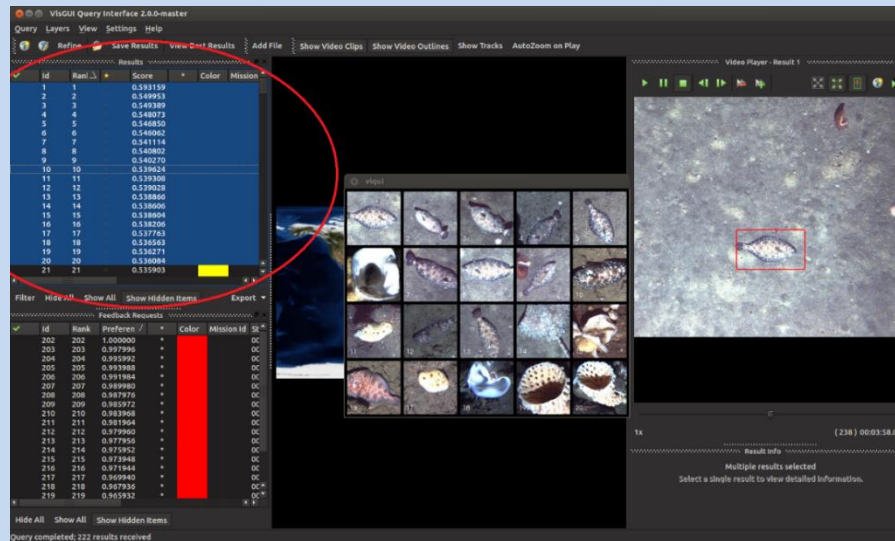
3 Core Detection Training Workflows

Method #3: Rapid Model Generation (IQR) Example Manuals:

https://viame.readthedocs.io/en/latest/section_links/search_and_rapid_model_generation.html

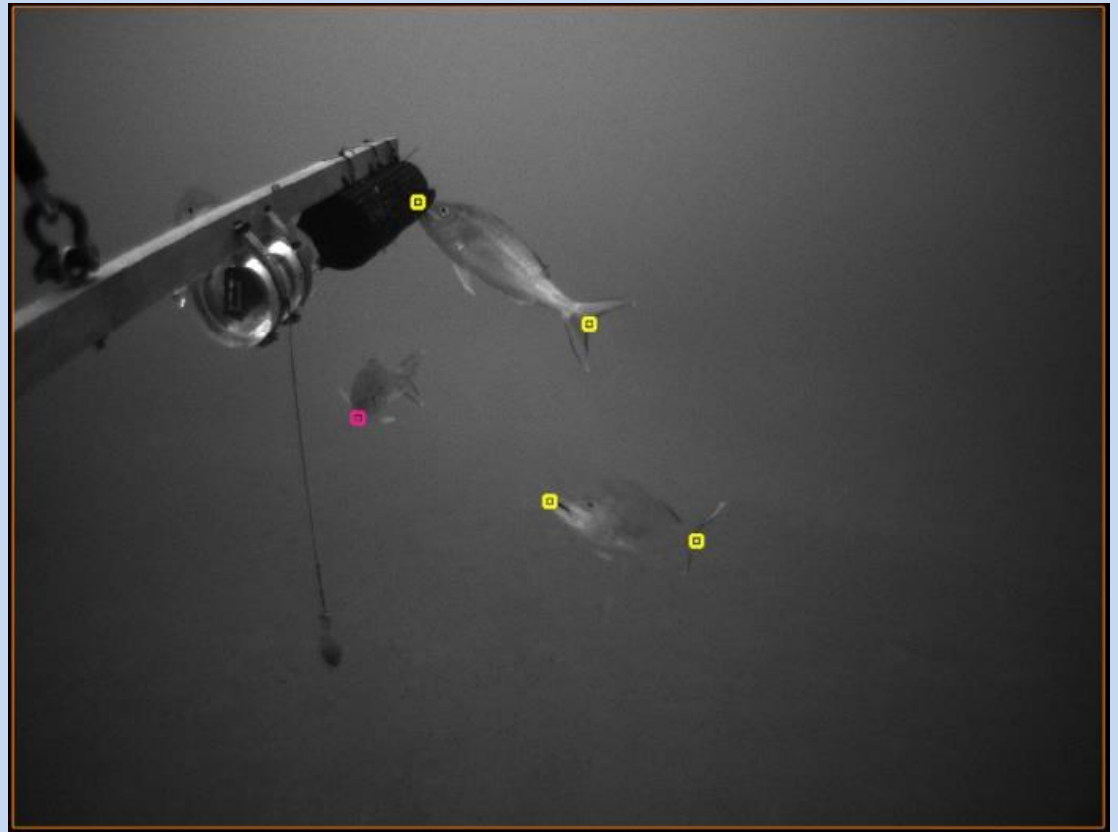
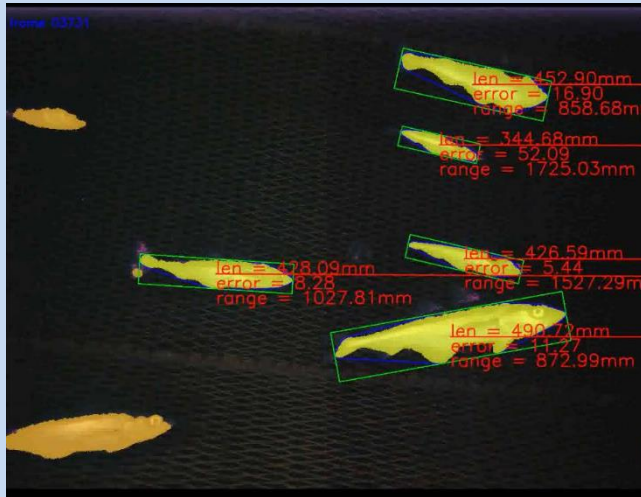
https://github.com/VIAME/VIAME/tree/master/examples/search_and_rapid_model_generation/viame_ingest

The above manuals correspond to the examples folder, which are similar to the project file. The project file contains scripts for creating indexes and models both on bounding box detections, bounding box tracks, and full frame classifiers with similar names.



IQR uses active learning to progressively fine-tune a model towards a target query via progressive rounds of supplied user feedback

Stereo Techniques



The system contains two methods for performing stereo measurement: a simple segmentation approach and between learned fixed points. To annotate points to train the fixed-point classifier, small boxes should be placed where the **center** of the box is on the fixed key-point of interest. Keypoints should be given consistent names, e.g. 'head' or 'tail'.

Running Trained Models

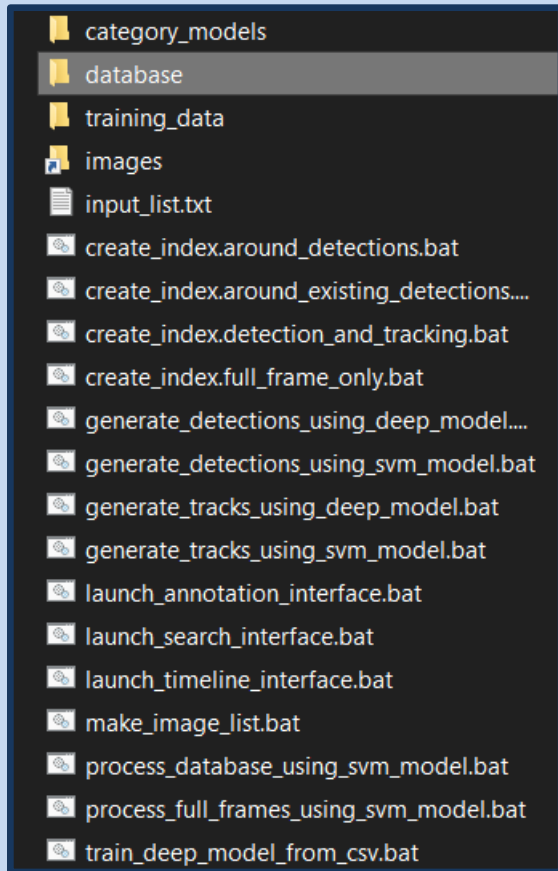
All trained output models get put in the 'category_models' folder regardless of which technique you are using. The 'deep_training' and 'augmented_images' folders contain intermediate outputs that can be deleted when finished, although 'deep_training' will output models during the training process that can also be used if the training processes are terminated early.

All of the 'generate' scripts in the project file use the detection models in the category_model to produce automatically computed results. They are divided based on whether or not they produce tracks or just per-frame detection, and whether or not they're applying deep models to data or svm (IQR) models.

These scripts look for all .svm files or a detector.pipe file which get produced by training scripts. Note: older versions of the tool did not generate detector.pipe files at training time, so if you need help upgrading them ping the developers email.

After using the generate scripts on a larger archive of videos, whenever you launch the annotation GUI it will also ask you if you want to load the detections or tracks you computed on the new data.

Running Trained Models



Outputs for all generation scripts (e.g. computed track files) get put in the 'database' folder for every sequence or video.

For 'for_image' project files the 'input_list.txt' file is similarly consumed for the generate_* scripts.

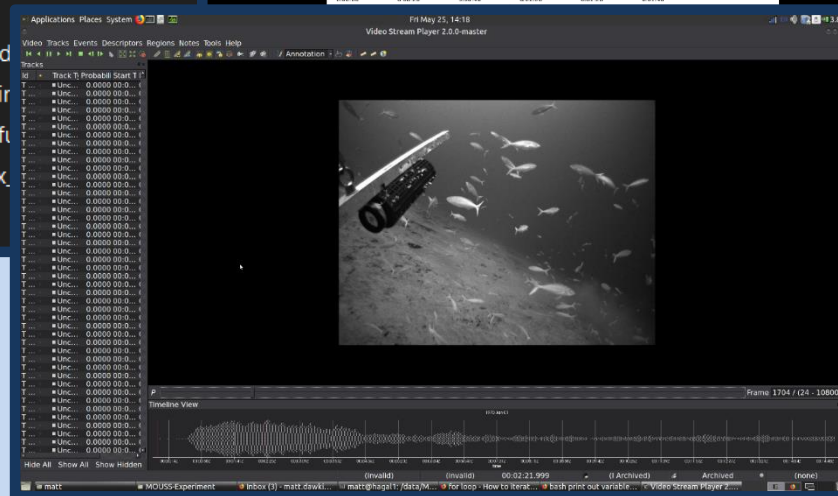
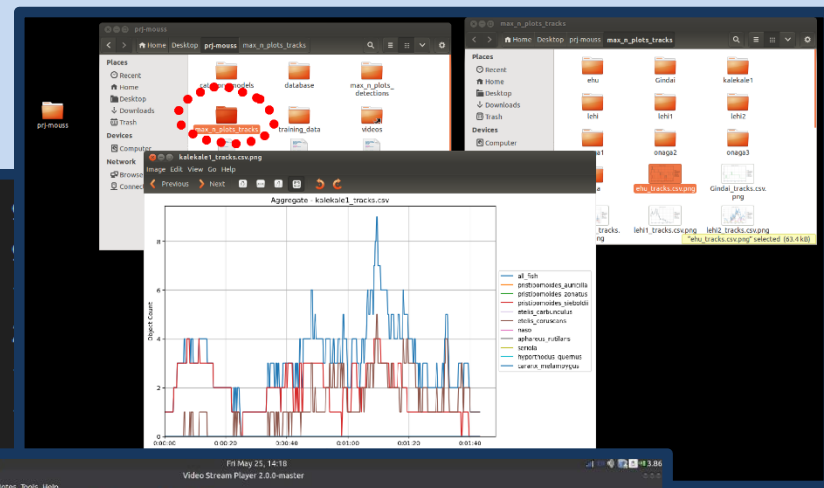
For 'for_videos' project files every folder or video file in the 'videos' folder is consumed

Outputs can then be opened either in the annotation GUI new project dialogue or via the launch script which will query which stored computed results in the database to load.

Running Trained Models

Lastly, there are many features that can be added to custom project files not in this document. One example is the 'MOUSS Project' which outputs per-species plots and Max-N counts for each input video using the trained models, but there are others as well. The 'launch_timeline_viewer.bat/sh' option also allows for this in a custom GUI.

- category_models
- training_data
- videos
- configure_gui_theme.sh
- launch_annotation_interface.sh
- launch_search_interface.sh
- launch_timeline_interface.sh
- process_videos.detection_and_tracking.no_ind
- process_videos.detection_and_tracking.timeline
- process_videos.detection_and_tracking.with_f
- process_videos.detection_only.timeline_index
- train_deep_model_from_csv.sh



For Questions:

viame.developers@gmail.com