

Semi automatic CVAT pipeline for data annotation

Julien Audet-Welke¹

²CERVO Brain Research Centre, Quebec City, Canada October 2024

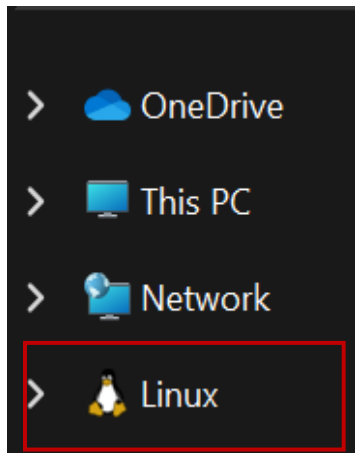
Overview

This package of scripts is designed to make adjusting tracks on a deep learning model (Precision Track) which is trained on animals as simple as possible. It streamlines the process of correcting automatically generated keypoints and bounding box annotations using CVAT (Computer Vision Annotation Tool). This program is an open-source data annotation tool that offers an SDK and an API which were used to create the pipeline described in this document. This pipeline is only compatible with Windows computers and has not been developed for Mac or Linux.

How to install Python

For the packages to function properly you will need to download Python which comes with pip install. Here is the link to install python: <https://www.python.org/downloads/>.

How to install Windows Subsystem for Linux (WSL)




Some of the components of CVAT need Linux to run on your machine so you will need to install WSL which is available at the following link: <https://ubuntu.com/desktop/wsl>.

Once it is installed, if you go into your system folder you should see a little Linux icon should appear. It should look like the left image. Next, go to the start menu, search for WSL and double click the app to start it. Once the terminal opens, enter this command:

```
julien@LAPTOP-8QJM408V:~$ pip install ubuntu
```

This will install the Linux operating system that is required. Next you can close the terminal and begin the CVAT installation tutorial.

Important remark

For this entire document I am using a custom command prompt that will have a different visual appearance than the standard Windows command prompt. This visual overlay does not change the commands in any way. Simply enter the command I write starting from the two right pointing arrows: 

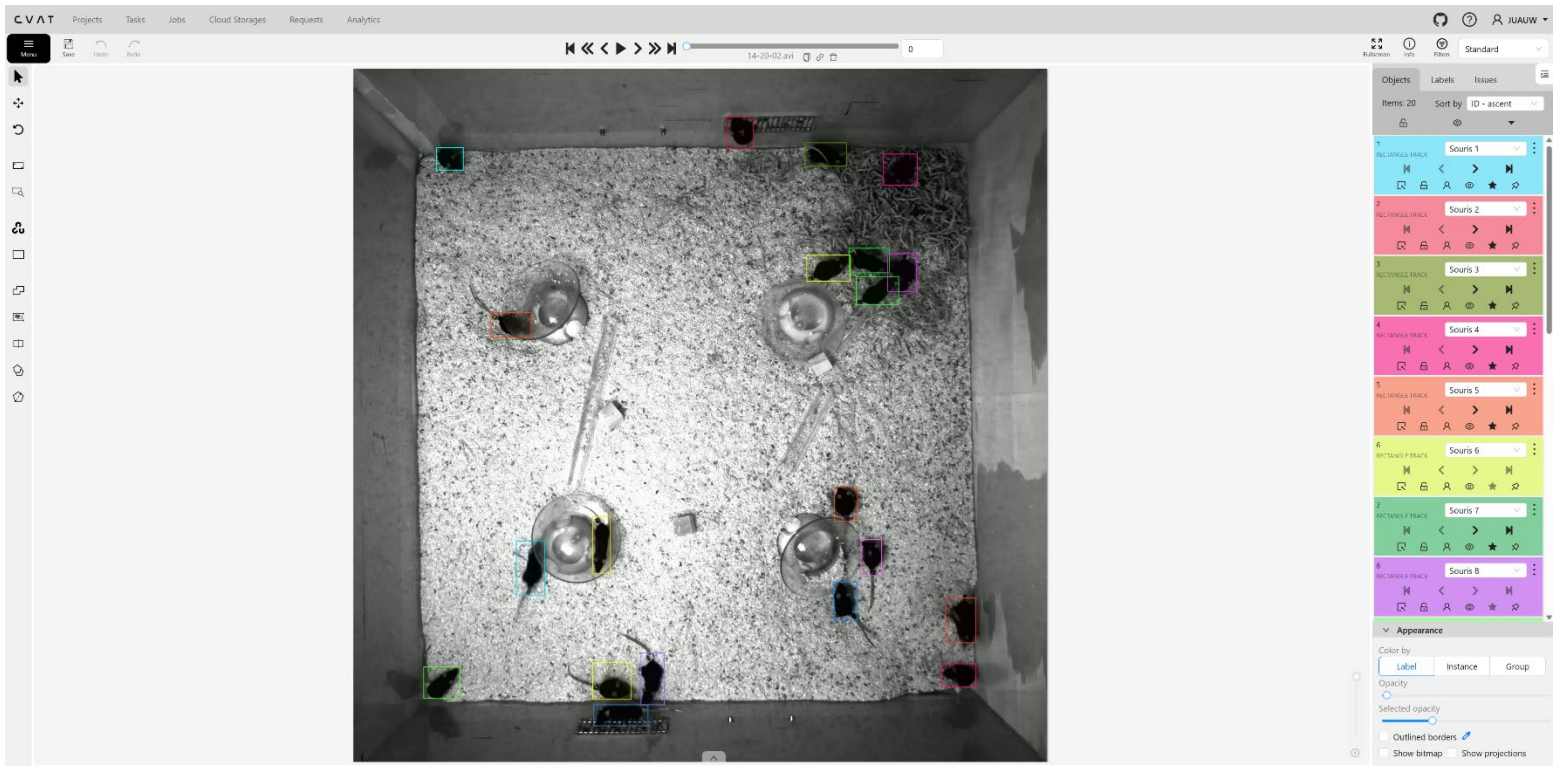
Local CVAT installation guide

YouTube tutorial: <https://www.youtube.com/watch?v=hyvNFuc06qQ>

Summary of the tutorial:

1. Install Git (file manager for code)
2. Install Docker (CVAT runs on a container)
3. Copy the CVAT repository on your machine (git clone)
4. Initialise docker
5. Install WSL
6. Install CVAT on docker
7. Create a username and password
8. Launch CVAT by typing localhost:8080 into a web browser

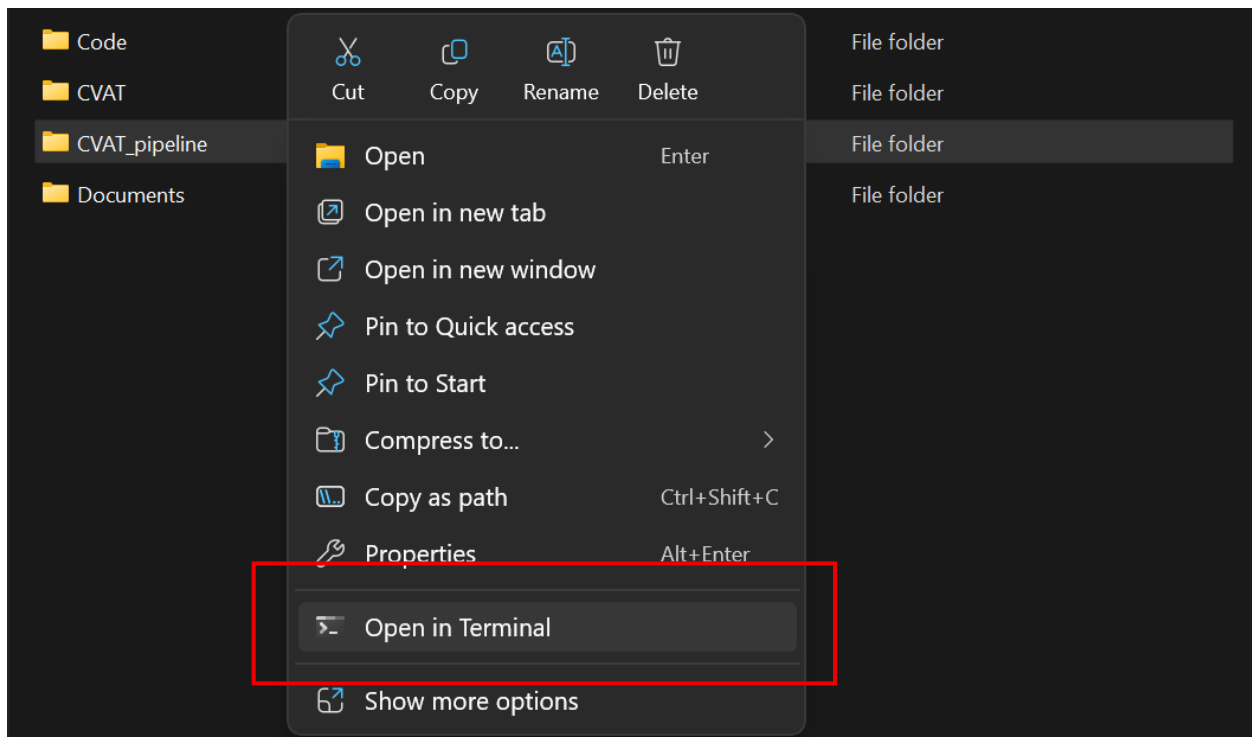
CVAT work environment example



Link to the GitHub repository: https://github.com/juauw/CVAT_pipeline.git

Step by step guide

Once you have CVAT up and running locally you can download the GitHub repository that was linked above. Follow the same steps as the CVAT download tutorial to load the repository on your computer. The first step is to open your terminal in the copied CVAT folder (CVAT_pipeline). Here is what that should look like:



This will open the terminal in the correct folder and should allow all the scripts to work. Next step to running the pipeline is to install all the dependencies and packages of the environment. You can do this by writing the following in your terminal (opened in the CVAT_pipeline folder):

```
>> pip install -r requirements.txt
```

Afterwards you can go ahead with starting the workflow. It splits into different parts depending on what your input data is. In the case of keypoints, you will need to generate an SVG file. An SVG file is an image file that contains the shape of a given skeleton and all the names of each point. It is essential to have the SVG file to be able to do keypoint annotation.

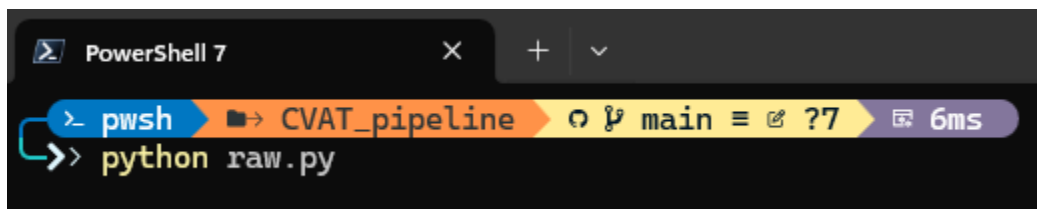
Keypoints pipeline

Here are the steps to generate the SVG file:

1. You will need a `metadata.py` file that contains the skeleton information for your animal
2. Next you will execute the `raw.py` script directly or via `auto_task_SDK.py`
3. The script will generate a file named `skeleton.svg` which you will input into CVAT later

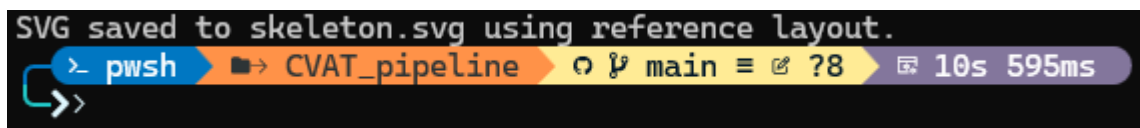
Details about **raw.py**

This script loads the `metadata.py` file and uses it as a reference to build an SVG image. By default, it uses mouse keypoints extracted from the metadata, but it can be expanded to fit any animal. Here is how to execute it directly:



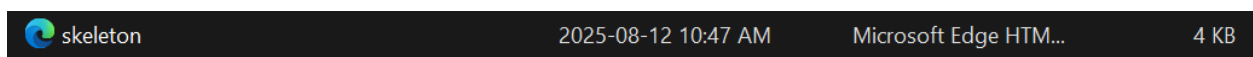
```
PowerShell 7
> pwsh -c "python raw.py"
python raw.py
```

If the command is successful, the command line should display this:



```
SVG saved to skeleton.svg using reference layout.
> pwsh -c "python raw.py"
python raw.py
```

And it should look like this in your folder browser:



The SVG will be generated into the same folder as the script. This SVG will be put into a field in CVAT once the **auto_task_SDK.py** script has been executed.

At this point in the workflow, it is important to know if you want to annotate keypoints or bounding boxes in CVAT. If you want to annotate keypoints you need to execute the **convert_json_keypoints.py** script. Here is how this script works and how to execute it. Note that if you don't have any annotations and are annotating from scratch you can skip this script and go directly to **auto_task_SDK.py**

Details about **convert_json_keypoints.py**

This script takes a file named `metadata.py` and a `.csv` or `.json` input file and produces a file named **converted_keypoints.json** or **converted_keypoints.xml** which is compatible with the CVAT keypoints import module. Here is how to execute the script directly:

```
> pwsh -> CVAT_pipeline -> main 2ms
>> python convert_json_keypoints.py input_path metadata.py --video_path "video1.mp4" --format "json"
```

For this command the mandatory arguments are the **input_path** and **metadata.py**. The **input_path** refers to the name of the .csv or .json annotation file which should be placed in the same folder as this script. The annotation file needs its name and the extension in the command prompt. The **--video_path** optional argument needs to be used only when the input format is CSV. The argument should be the name of the video corresponding to the CSV annotation file. It is used to calculate the resolution of the frame which is needed to create the correct annotations. The video should also be in the same folder as the **convert_json_keypoints.py** script. The **-format** optional argument is used to select the desired output format. The options are xml or json. Here is what the console should display if the script ran correctly (for the example above):

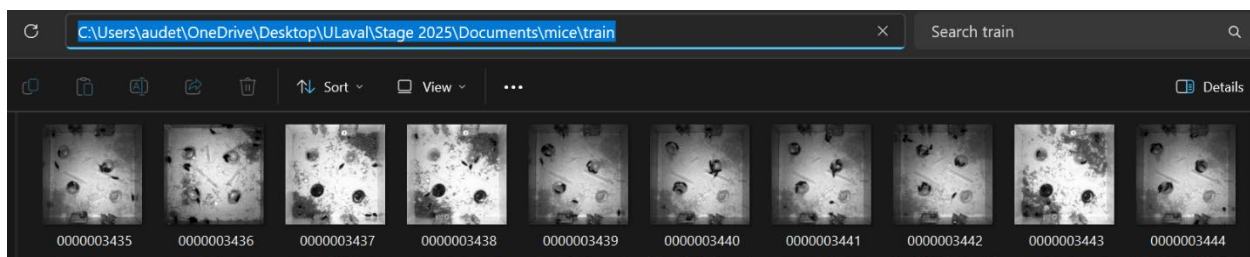
```
COCO Keypoints 1.0 file generated successfully: 'converted_keypoints.json'
> pwsh -> CVAT_pipeline -> main 1s 945ms
>>
```

The script will output a file named **converted_keypoints.json**, this file will be uploaded manually into a CVAT task once the **auto_task_SDK.py** script has been executed. The alternate xml output file is only used if there is a chance that the tracks have switched in the annotations and there needs to be a verification completed since COCO keypoints do not support persistent tracks in CVAT.

Details about **auto_task_SDK.py**

This script is central to the keypoints annotation pipeline. It creates a task in CVAT and automatically imports the images associated with the task. The annotations must be uploaded manually because of a limitation in CVAT. There must only be one video in **-folder** argument. This script also executes **raw.py** automatically. Note that for this script and the **auto_kpts_bbox.py** script you can rename the task using the **-task-name** argument. Here is how to use the program:

1. You need to enter your username and password into the command line
2. You can choose to enter a task name, otherwise it will be "Keypoint annotation"
3. Select the folder where your images et videos are and copy its path:



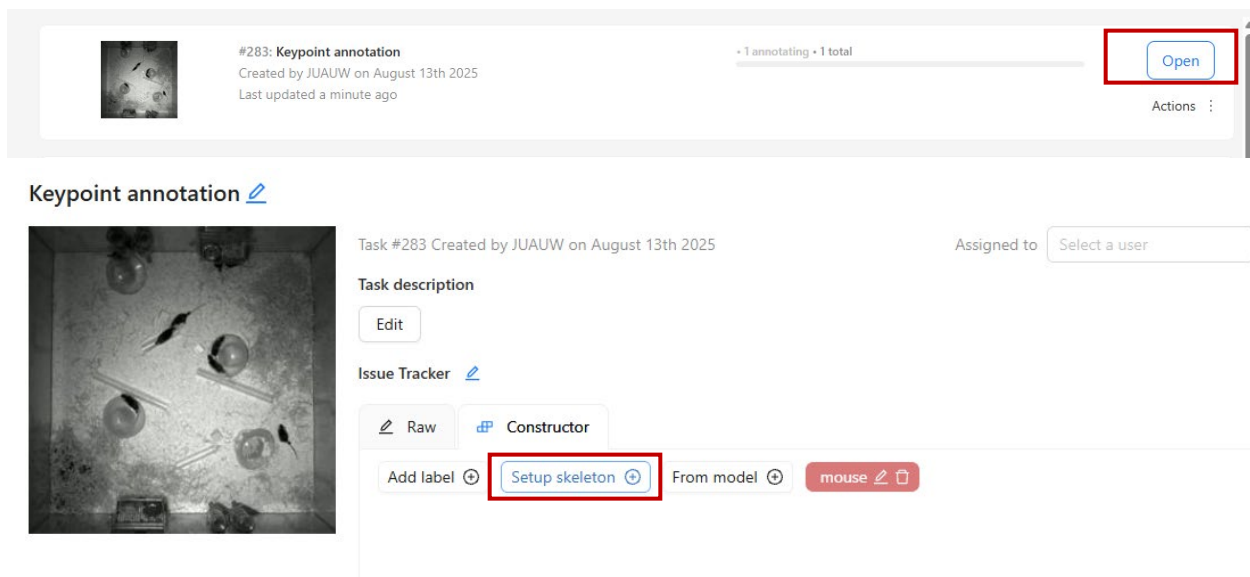
Here is what the input into the command line should look like:

```
> pwsh -> CVAT_pipeline -> main 75 2ms 100% 13,11:20
>> python auto_task_SDK.py --username "JUAUW" --password "Yetisb130^^" --folder "C:\\Users\\audet\\OneDrive\\Desktop\\ULaval\\Stage 2025\\CVAT_pipeline"
```

If the command has run successfully, it should display this result:

```
Running raw.py ...
SVG saved to skeleton.svg using reference layout.
Connected to CVAT server.
Extracting frames from video: video1.mp4
Uploading 1301 extracted frames.
Uploading data: 100%| 719M/719M [01:08<00:00, 11.1MB/s]
Task 'Keypoint annotation' created with ID: 283
```

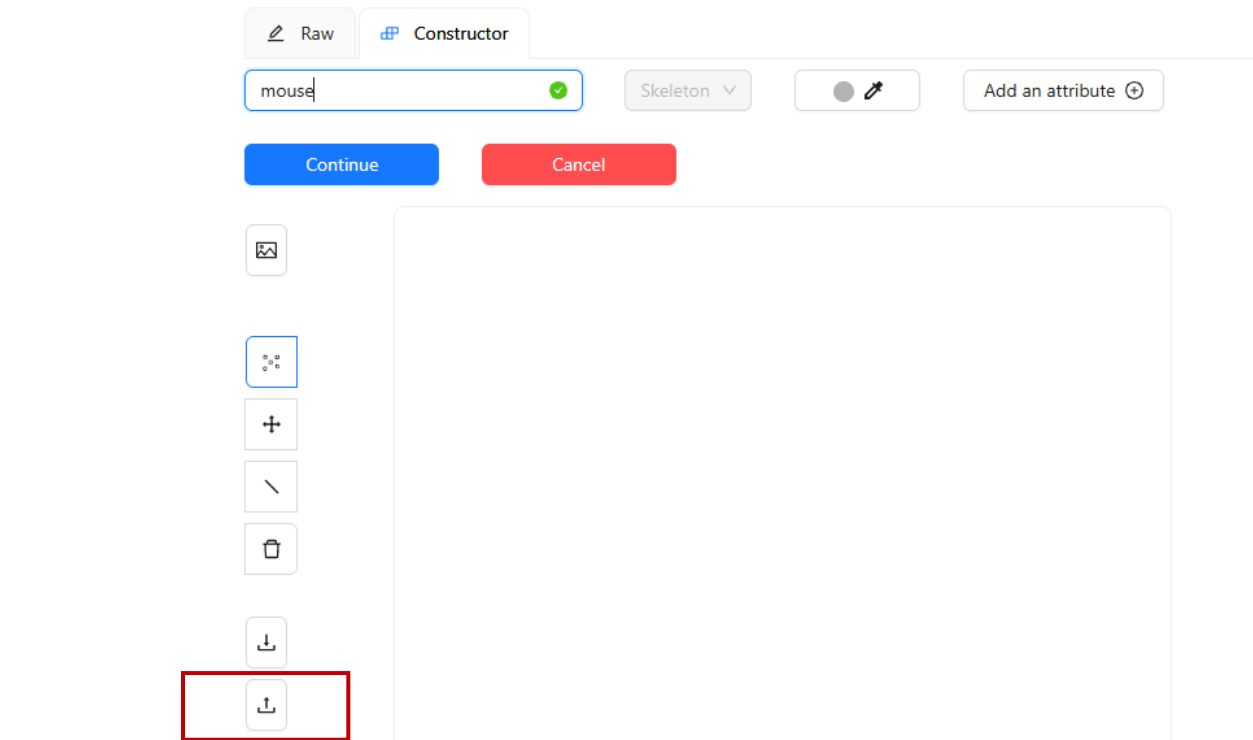
4. Once this step is complete you will find the task in the CVAT UI where you will have to upload the SVG and the annotations manually. Here are the steps shown visually:



The screenshot shows the CVAT web interface. At the top, a task card for '#283: Keypoint annotation' is displayed, created by JUAUW on August 13th 2025. The 'Open' button on this card is highlighted with a red rectangle. Below this, the task details page for 'Keypoint annotation' is shown. On the left is a video frame with keypoint annotations. On the right, the 'Task description' section includes an 'Edit' button. The 'Issue Tracker' section has a link. The 'Raw' and 'Constructor' tabs are visible, with 'Constructor' being the active tab. In the 'Constructor' tab, the 'Setup skeleton' button is highlighted with a red rectangle. Other buttons like 'Add label', 'From model', and 'mouse' are also visible.

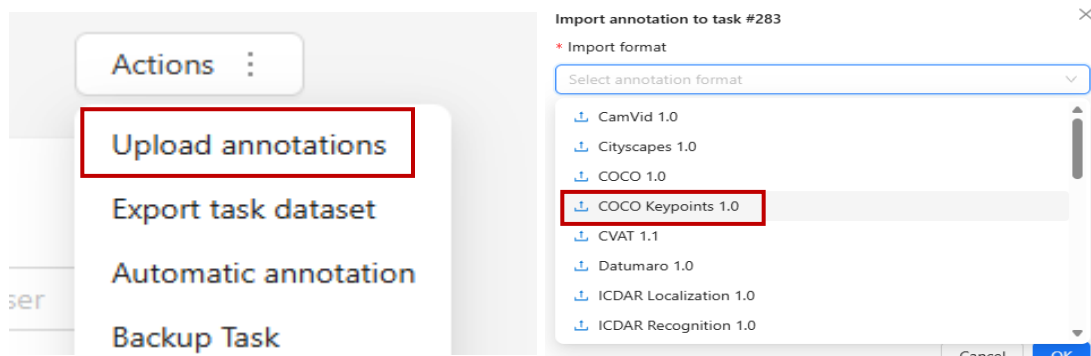
5. You will need to delete the label and create a new one, it needs to keep the same name. Next you need to upload the SVG:

Note: If you labelled your task differently the task will not be named “Keypoint annotation”



Note: If you are starting annotation from scratch, you can go back to tasks and click on the Job hyperlink to get started directly with annotations.

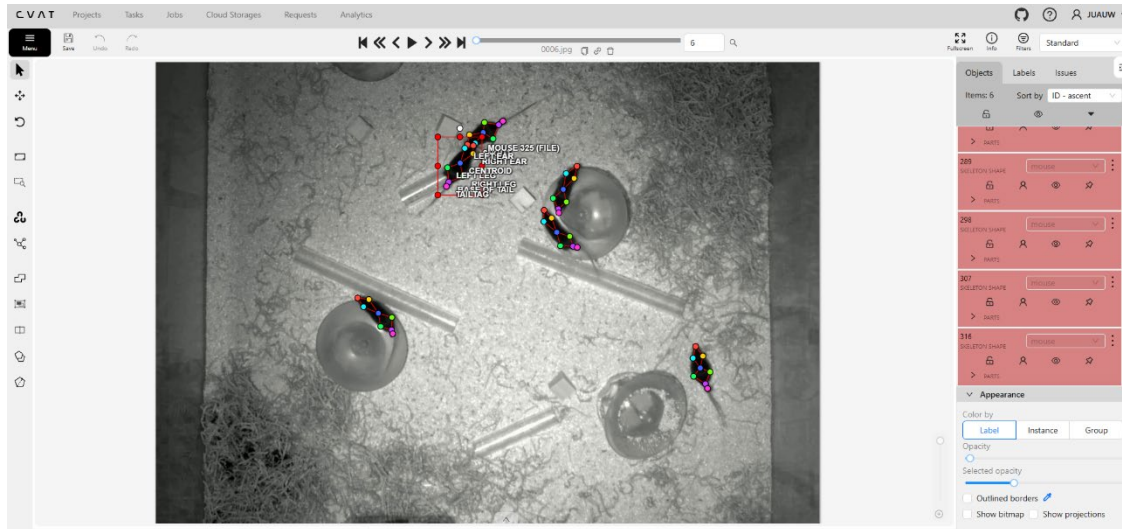
6. A skeleton will show up once you upload the SVG that is in the CVAT_pipeline folder. Once this is done you can click continue and go back to the task. Next you will upload the file named **converted_keypoints.json**:



7. This is the page where the annotations will upload:



8. Once the annotations are uploaded the interface should look like this:



At this stage you can annotate keypoints from scratch or modify existing keypoints, so they are more precise. Once the annotation is completed you will need to export the annotations manually since the CVAT SDK does not currently support exports.

How to export keypoints or bounding boxes

The process for exporting keypoints and bounding boxes are the same except for the export format. For keypoints the format to select is **COCO Keypoints 1.0** and for bounding boxes the format is **COCO 1.0**. It is very important to name the keypoints folder **corrected_keypoints** and the bounding box folder **corrected_bbox**. Here is a visual representation of the process:

Export task #274 as a dataset

* Export format

COCO Keypoints 1.0

Save images

Custom name

corrected_keypoints.zip

Use default settings

CancelOK

Cloud StoragesRequestsAnalytics

Export AnnotationsTask #274

Started by JUAUW on Aug 13th 25, 13:25

Expires on Aug 14th 25, 13:26

Finished

100.00%

COCO Keypoints 1.0

Download

The final step of the keypoints pipeline is to use the **auto_kpts_bbox.py** script to convert the corrected or manually created keypoints into bounding boxes automatically if no bounding box annotations are present for the current dataset.

Details about **auto_kpts_bbox.py**

This script converts a **COCO Keypoints 1.0** file to a **COCO 1.0** file by estimating bounding boxes based on the exported files keypoints. After the script creates a new task and automatically uploads the images and annotations to CVAT. It can also directly upload bounding boxes to CVAT using the **CVAT 1.1** format. Here is what to enter in the command line (the folder path will be different for your computer):

```

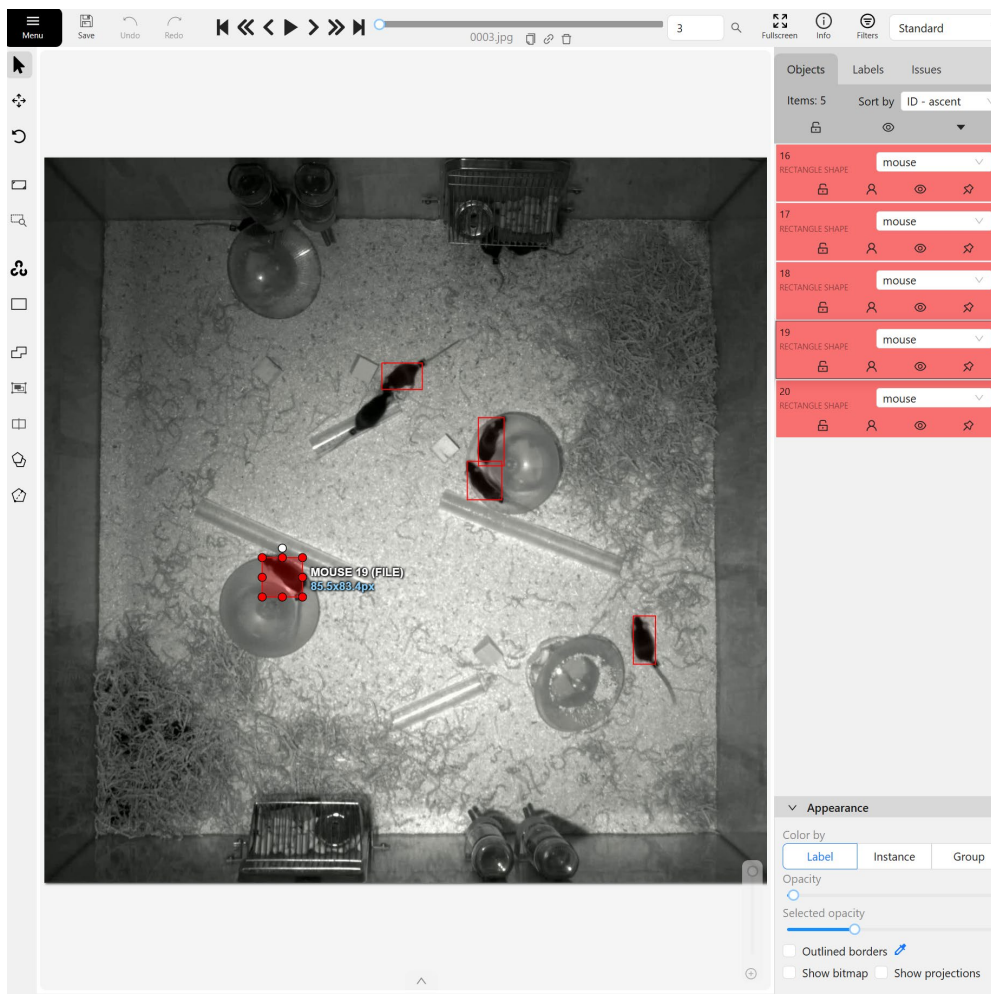
pwsh -c CVAT_pipeline main ?8 ~1 2ms
python auto_kpts_bbox.py --username "JUAUW" --password "Yetisb130^^" --folder "C:\Users\audet\OneDrive\Desktop\ULaval\Stage 2025\CVAT_pipeline\video1.mp4" --input "corrected_keypoints.json"

```

If the command was entered properly the command line should display the following:

```
Deleted zip file: corrected_keypoints.zip  
Moved and renamed file to: .\corrected_keypoints.json  
Deleted extracted folder: corrected_keypoints  
Video detected in folder: converting to frames...  
Extracted 1301 frames  
Converting COCO keypoints to COCO bounding boxes...  
Prepared COCO bbox file  
Uploading data: 100%|██████████████████████████████████████████████████████████████████████████████| 719M/719M [01:05<00:00, 11.6MB/s]  
Task created with ID: 286  
Uploading data: 100%|██████████████████████████████████████████████████████████████████████████████| 1.47M/1.47M [00:00<00:00, 6.78MB/s]  
Bounding box annotations imported successfully.  
Cleaned up extracted frames folder
```

Once this step is completed the bounding boxes created from keypoints will be ready to be corrected in CVAT. All you need to do is go to tasks, click the task named “Bounding box annotation” and click on the job. The UI will appear and will look like this:



Once you are done with correcting the bounding boxes you will export the annotation file like it was explained in the “how to export” section. This is the final step in the keypoints pipeline and all that is left is to merge the bounding boxes and keypoints in the final **merge bbox keypoints.py** script.

Bounding box pipeline

As mentioned at the beginning of this document the pipeline has two main workflows. We will now explore the bounding box pipeline which functions in a similar way to the keypoints pipeline but is more automated and shorter.

Details about `convert_json_bbox.py`

This program takes two files that are named `metadata.py` and a `.json` or `.csv` file and produces a file named `converted_bbox.xml` which is compatible with the CVAT bounding box import module. The format produced is CVAT 1.1 so the bounding boxes can have a consistent track. Here is what to input into the command line to run the script:

```
> pwsh CVAT_pipeline main ?8 2ms
>> python convert_json_bbox.py bboxes.csv metadata.py --video video1.mp4
```

If the script executes correctly, it should display this following message in the command line:

```
Saved CVAT 1.1 XML to: converted_bbox.xml
```

The `converted_bbox.xml` file will be saved in the same folder as the script and will have a different looking icon then the `.json` files that were created previously.

The next step is to execute the `auto_kpts_bbox.py` script which has a built-in function that automatically detects the `.xml` extension and skips the keypoints to bounding box conversion. It will directly upload the **CVAT 1.1** bounding box annotation to CVAT. Here is how to execute the script in this context. The task name will be the same as the `.json` bounding boxes but you can modify it using the `--task-name` argument to avoid confusion:

```
> pwsh CVAT_pipeline main ?9 ~1 -2 6ms
>> python auto_kpts_bbox.py --username "JUAUW" --password "Yetisb130^^" --folder "C:\Users\audet\OneDrive\Desktop\ULaval\Stage 2025\CVAT_pipeline\video1.mp4" --input "converted_bbox.xml"
```

Here is what the command line should show if the script executed correctly:

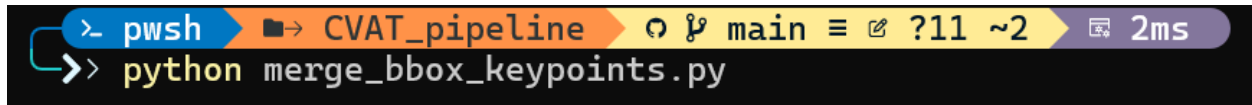
```
Video detected in folder: converting to frames ...
Extracted 1301 frames
XML detected - will import CVAT 1.1 directly
Uploading data: 100% | 719M/719M [01:14<00:00, 10.1MB/s]
Task created with ID: 289
Uploading data: 100% | 1.46M/1.46M [00:00<00:00, 5.48MB/s]
Bounding box annotations imported successfully.
Cleaned up extracted frames folder
```

Once this has executed you should be able to go into the newly created task, and the UI will show the same view as the image presented in the keypoints pipeline subsection. You will be able to correct the bounding boxes while keeping a consistent track.

Details about `merge_bbox_keypoints.py`

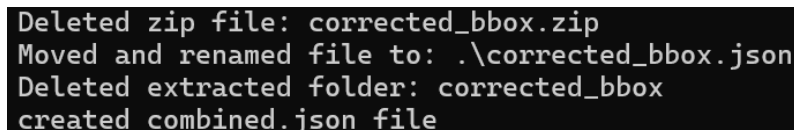
This script takes a corrected **COCO Keypoints 1.0** `.json` file and merges it with a corrected **COCO 1.0** file and makes it compatible with the Precision Track pipeline. It is important to make sure that you have both `corrected_keypoints.json` and `corrected_bbox.json` when you

executed the script. There is no configuration needed for the script, so the execution of the script is straightforward. Here is what it should look like:

A terminal window with a dark background. The title bar shows 'pwsh' in a blue box, 'CVAT_pipeline' in an orange box, and 'main' in a grey box. The command prompt is '>>'. The command entered is 'python merge_bbox_keypoints.py'. The execution time '2ms' is shown in the top right corner.

```
> pwsh CVAT_pipeline main ?11 ~2 2ms
>> python merge_bbox_keypoints.py
```

If the program executes correctly, it will display the following and the final **combined.json** file will be present in the **CVAT_pipeline** folder:

A terminal window with a dark background showing the output of the script. The text is as follows:

```
Deleted zip file: corrected_bbox.zip
Moved and renamed file to: .\corrected_bbox.json
Deleted extracted folder: corrected_bbox
created combined.json file
```

Once you have completed the pipeline, I recommend you delete all the **.json** and **.xml** files before moving on to the next annotation task. This will remove clutter from your pipeline folder. This file has all the information needed to be used by the Precision Track pipeline. With the updated keypoints and bounding boxes the Precision Track model will be able to track movement more accurately.

CVAT pipeline overview: here is a visual summary of the entire pipeline

