

VisionTag

Documentation

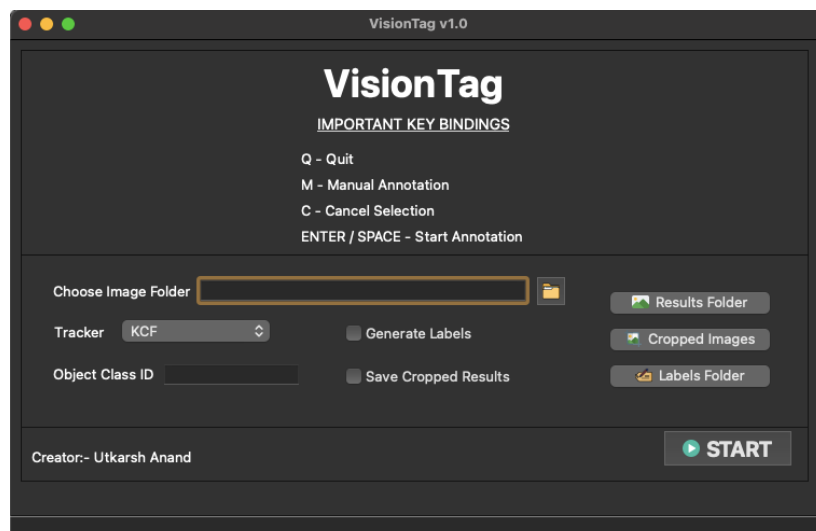
Author: - [Utkarsh Anand](#)

Version: - 1.0.0

Queries: - utkarshanand221@gmail.com

1. OVERVIEW: -

VisionTag is a GUI application made using **OpenCV** & **PyQt5** library in Python. It uses Computer Vision techniques to automate the annotation process of images for training Deep Learning models. The application provides a graphical user interface (GUI) to annotate images in a more user-friendly manner.



2. CODE DESCRIPTION:-

Class: **MyGUI()**

This class is the main class of the application and implements the GUI functionality of the application. It has the following main components:

- Loads the main.ui file which is a UI file created in Qt Designer.
- Connects different buttons in the UI to their respective functions.
- Contains functions to perform annotation, open results, open cropped images, open labels and choose folder.

Function: ***drawBox()***

This function draws a bounding box on the image. It takes the image and the bounding box coordinates as input and returns the image with the bounding box drawn on it.

Function: ***get_index ()***

This function returns the index of an image from its filename. It takes the string (filename) as input and returns the integer index of the image.

Function: ***getTrackerType()***

This function returns the tracking algorithm to be used. It takes the string *tracker_type* as input and returns the tracker algorithm.

Function: ***text_generator()***

This function generates the label text file. It takes the number of the image, its bounding box, and the result path of the label text file as input.

Function: ***popup()***

This function displays a message box with a given message. It takes the message to be displayed as input.

Function: ***execute()***

This function is the main function to execute the annotation process. It performs the following steps:

- Clears the Result and Labels folders.
- Loads the images from the specified directory.
- Initializes the tracking algorithm.
- Performs tracking on each image and saves the bounding box information in a text file.

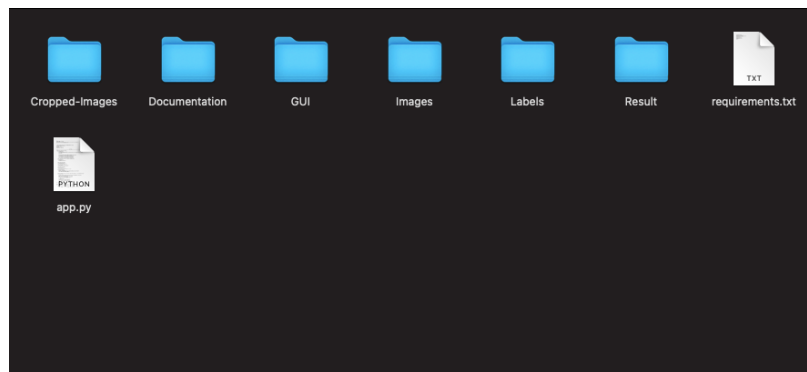
Global Variables

The following global variables are used in the code:

- ***path***: the path of the directory containing the images.
- ***tracker_type***: the tracking algorithm to be used.
- ***obj_id***: the object id for which tracking is to be performed.
- ***label***: a flag to specify whether to save the bounding box information in a text file.
- ***crop***: a flag to specify whether to crop the image or not.

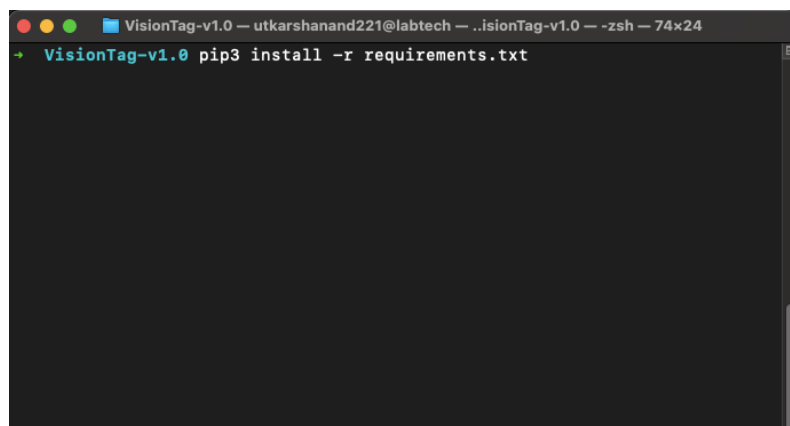
3. STEPS TO INSTALL THE APPLICATION:-

- Extract the Zip File and Navigate into the VisionTag Directory.



- Install the required Python libraries using the following command in the terminal.

```
$ pip3 install -r requirements.txt
```



4. BEFORE RUNNING THE APPLICATION:-

1. Video to frames.py (OPTIONAL)

SKIP IF YOU ALREADY HAVE YOUR IMAGES READY

- You can use this python script to convert your input video into a series of frames.
- Simply, put the full file path of your video into the **vid_capture** instance.

```
1 import cv2
2
3 vid_capture = cv2.VideoCapture('Drone-Videos/7.mp4') # Path of the vid
4 if (vid_capture.isOpened() == False):
5     print("Error opening video file")
6 else:
7     i = -1
8     while(vid_capture.isOpened()):
9         i = i+1
10        ret, frame = vid_capture.read()
11        if ret == True:
12            path = f"Images/frame{i}.png"
13            cv2.imwrite(path, frame)
14            print(path)
15        else:
16            print("Process Completed!")
17            break
18
19 vid_capture.release()
20
```

- Run the python file in the terminal using the following command.

```
$ python3 video_to_frames.py
```

- After execution, your frames will be stored in the **Images** Folder.

2. IMAGES NAMING FORMAT:-

Make sure that all of you images are named in the following format.

frame<index>.png

Example:- frame0.png, frame1.png, frame2.png

Default indexing starts from **0** and default extension used is **.png**.

NOTE:- By default, **frame0.png** will be used as the reference image to initially annotate the object. To use any other image, accordingly change the reference image path in the **app.py** file.

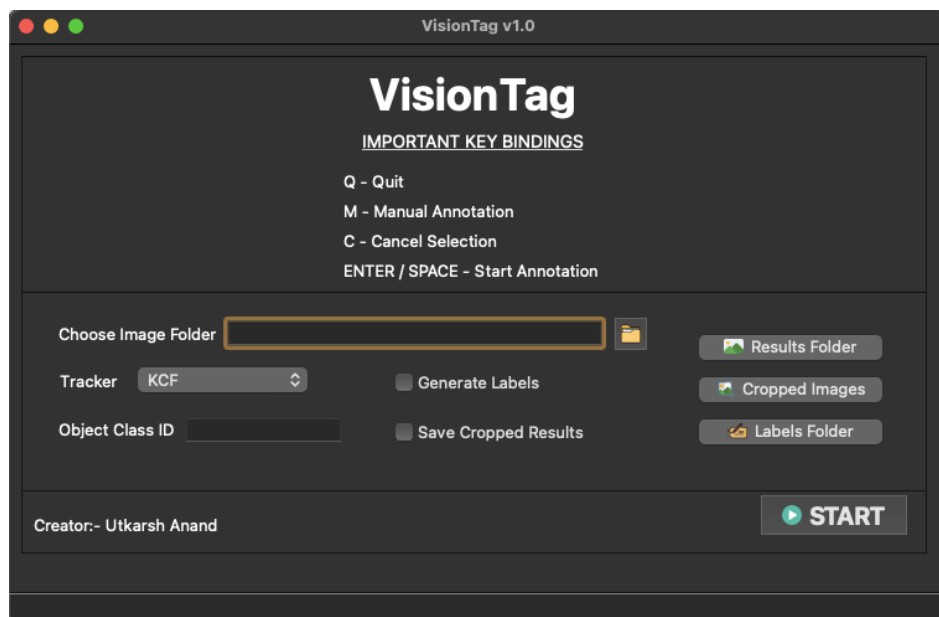
NOTE:- If your images are not named using the convention given above, application will throw errors. If you want to use other naming conventions or if you want to use other image format other than PNG, you have to change the code in the **app.py** file accordingly.


5. STEPS TO RUN THE APPLICATION :-

- a. Open Terminal and make sure that you are in the VisionTag directory.
- b. Run the app.py file in the terminal using the following command

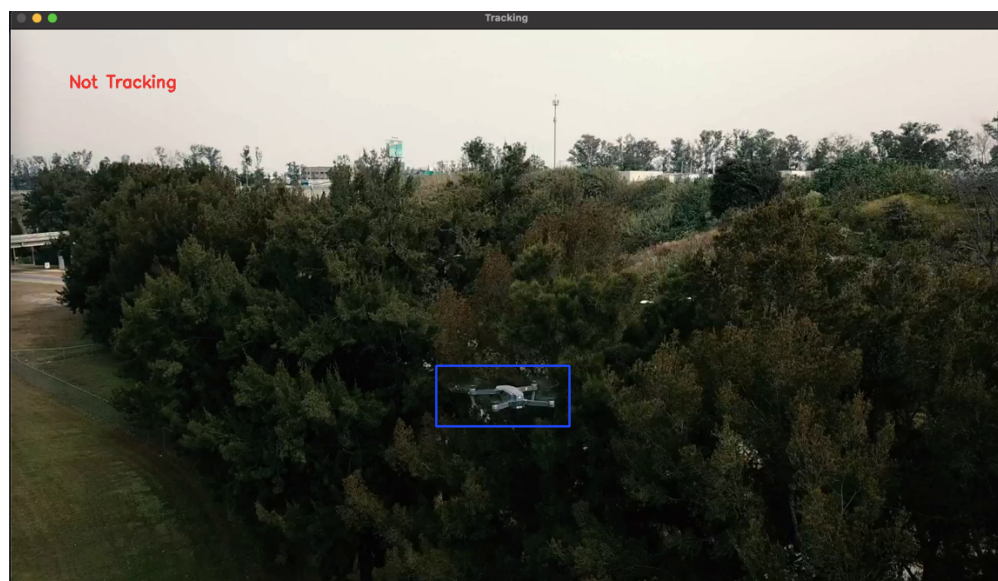
\$ python3 app.py

- c. The VisionTag GUI will appear.



- d. Click on the  Icon to select the location of the folder where the input images are stored. You can also use the Images folder in the VisionTag directory to store your input images.
- e. Next, Select the tracker which you want to use for your annotation process. The different type of available trackers are:-
 - i. KCF (Default & Recommended)
 - ii. MOSSE
 - iii. CSRT
 - iv. BOOSTING
 - v. MEDIANFLOW
 - vi. TLD

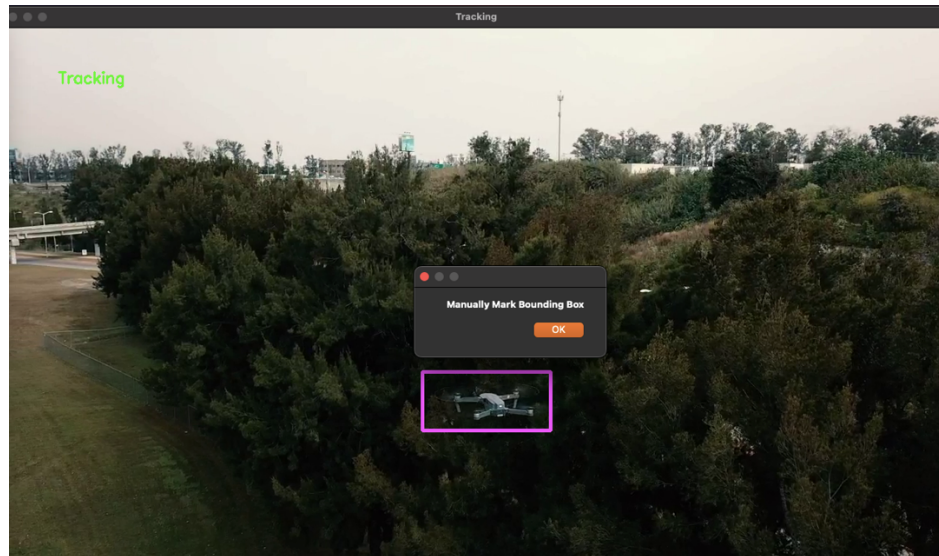
- f. Next, enter the object class ID of your object. For Deep Learning training purpose, each class of object has to be given a unique ID. Enter whatever ID you want to appear in the labels generated for your images.
- g. Next, Select the “**Generate Labels**” box if you want to generate labels for your images. Labels will be generated in the YOLO format and will be stores in the **Labels/** folder.
- h. Next, Select the “**Save Cropped Results**” box if you want to save the cropped image of the object from each frame. Images will be stored in the **Cropped-Images/** folder.
- i. Finally, Click on the **START** button. As soon as you click on the **START** button, your Reference image will appear and it will ask you to initially manually annotate the object in the image.



After Annotating the object using the cursor, press **ENTER / SPACE** to start the annotating process.

You can also press “**C**” on your keyboard if you don’t want to annotate your reference image. In that case, you Annotation process will exit and you would have to start again.

- j. During the annotation process, If you want to **Quit** the annotation process at any given point of time, you can simple Press and Hold the “**Q**” key on your keyboard and your annotation process and exit back to the Main HomeScreen.
- k. If the tracker looses the object, it will automatically ask you to manually annotate the image again



If at any given point of time, you feel that the tracker is loosing the object or that if it is not working properly, you can **manually annotate** the image again and re-initialize the tracker. To do so, just Press and Hold the “**M**” key on the keyboard during the annotation process to stop it and manually annotate the image.

After manually annotating the images, similar to above, either press the **ENTER / SPACE** key to start the process or “**C**” to end it.

- j. Once the Annotation process is complete, the Resultant Images can be found in the **Results/** folder.