Image/Video processing logs

4th March 2024

# Disclaimer:

Currently, the idea regarding the algorithm (how often pictures and video should be taken, and should image even be taken due to concern regarding synchronization) still not finalized yet, and findings from data size and viability in here will help to decide that. Current assumption is a picture every 5 minute, a 30 second video every 30 minutes, for 16 hours a day, 7 days a week which totals around 80.724GB, thus needing a weekly data transfer/backup.

# Object/Scene detection preliminary testing:

Taking inspiration from LibrePhotos Scene detection: [💡 Feature Comparison | LibrePhotos](https://docs.librephotos.com/docs/user-guide/features/) [Introduction | LibrePhotos](https://docs.librephotos.com/docs/intro/) which uses [MIT Places Database for Scene Recognition](http://places.csail.mit.edu/). Maybe I can take some frames from video using tutorial and guides like [Video Data Processing with Python and OpenCV - YouTube](https://www.youtube.com/watch?v=AxIc-vGaHQ0) which also uses ffmpeg which I’m now familiar with thanks to Algorithm part. Other than that, image captioning also fits the description [OLeoghain/image\_caption-lstm: Image Captioning Model Implemented in PyTorch using CNN followed by LSTM (github.com)](https://github.com/OLeoghain/image_caption-lstm).

Sidenote: a great suggestion for future improvement is to have accelerometer to detect when to take videos to avoid shaky video and more importantly a GPS sensor module so we can have geolocation metadata.

# Stitching and alignment preliminary test:

Using visual test via Virtual Desktop SBS player, it seems sync is pretty fine, and 15fps is borderline acceptable, for some reason I cant get it to 24/30 fps anymore even though I got it once first time before.. Other than that, the alignment is horrible, one of the camera had different angle and need me to cross eye to see properly, but the 3d effect is surely there! So need to redesign and reprint the mount for sure!

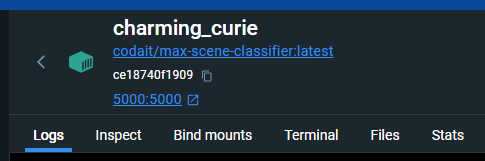
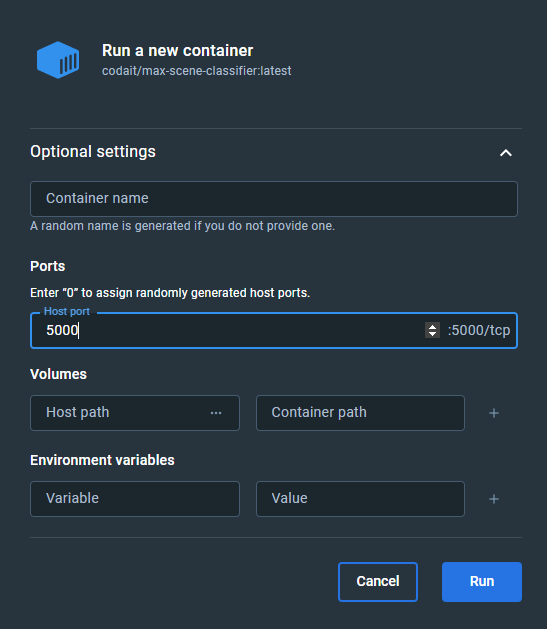
# Places365 testing:

This is so old, and the github pages seems to be full of unresolved issues, I tried both locally and docker but to no avail, ig main culprit is that it uses Python2, and even then for some reason, I cant get python2 to work with torch etc, ig I skipped proper installation, anyway, it seems like I found more promising alternative (more recent and better installation guide): [IBM/MAX-Scene-Classifier: Image classifier for physical places/locations, based on the Places365-CNN Model (github.com)](https://github.com/IBM/MAX-Scene-Classifier)

# IBM/MAX-scene-classifier on Docker:

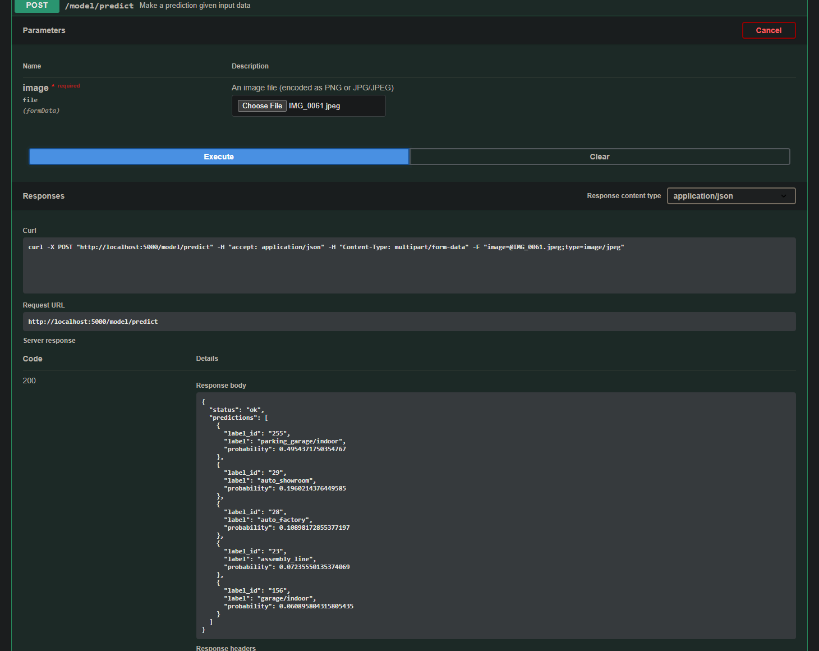
It works!!!

[codait/max-scene-classifier - Docker Image | Docker Hub](https://hub.docker.com/r/codait/max-scene-classifier)

Note: remember to put the port 5000 so it is hosted there straight away and works! 

Or else it u cant access it at <http://localhost:5000>

The image and response prediction!!!



Also got metadata tagging works using piexif, tags doesn’t work and not available to change according to the docs, the only problem now is how to tag the video, should I take some frames and tag them separately or tag the video by itself or combination of both?

8th March 8, 2024

# Stitching automation for all files in folders etc for quick testing

Done! Not so hard at all tbh especially when camera aligned correctly!! The output images from samples 8th march is sooo good, especially the morning and outside one!

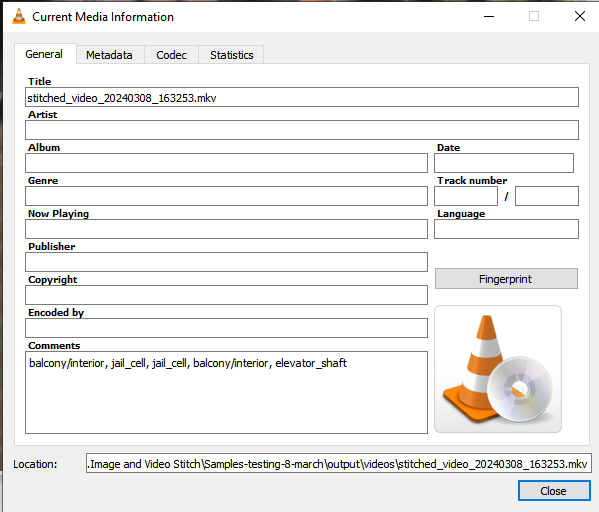
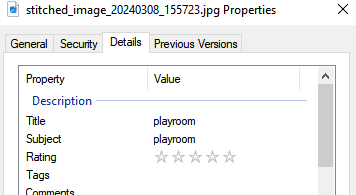
# MAX-scene-classifier automated

Automated to loop through all .jpg and .mkv instances, then write the highest prediction on description metadata for jpg using piexif and 5 prediction from evenly spaced frames from 30sec video to comments for .mkv using ffmpeg. This runs very fast, the predictions api is so good! Make sure the docker container for it is running!

Next, I should modify previous stitching code to retain the metadata from img0/vid0.

DONE! Used ffmpeg for videos and piexif for images! nice

Example:



10th March 10, 2024

Due to the fact that Godot’s built-in videostreamplayer node that I utilized for SBS video player only compatible with .ogv videos instead of .mkv, I changed the ffmpeg command to output as so, seems to work fine for now, very notable problem is the performance/speed of conversion. This is notably wayyyy slower than just stitching, understandably so as conversion to different format also is done.

Next steps: might need to multithread this so can run multiple conversion in parallel to speed it up.

Okay, apparently accessing metadata in Godot is not simple and easy, so instead I’ve opted to also make separate .json response file of the predictions.   
Remember to multithread stitching!