Filter Code Documentation

Project Code

Subhransu S. Bhattacharjee

August 2023

This document provides a brief description with motivation for the filters developed for the Epic Kitchens dataset [1] processing for training the NERF.

1 Hands Filter

1.1 Motivation

The motivation of this code is to filter a set of images and select only those that contain human hands to a tolerable standard such that it does not occlude the environment too much. The code uses the *ultralytics: Yolov8* object detection model to detect persons in the images and calculates the ratio of the bounding box area of the detected person to the total image area. Images with a ratio below a certain threshold are selected and copied to a destination folder.

1.2 Description

The code consists of two main functions: *person area ratio* and *main*. The first function, *person area ratio*, calculates the ratio of the bounding box area of a detected person to the total image area. It takes in an image path, a model path, and an optional total area as input. The function loads a YOLO object detection model [4] from the given model path and uses it to predict objects in the image. It then filters the detection boxes by class value 0 (person) and calculates the area of the first-person box. If no person box is found, the function returns 0. The function then calculates the total image area if not provided, and returns the ratio of the bounding box area to the total image area.
The second function, *main*, iterates over images in a source directory and calculates their person area ratio using the *person area ratio* function. It checks if the ratio is below 0.35 and copies images with a ratio below this threshold to a destination directory. The function also includes code to create the destination directory if it does not exist, calculate the total image area for the first image in the source directory, and display a *tqdm* progress bar while filtering images.
This code provides a solution to filter images based on their content, using an object detection model to detect persons in images and calculate their bounding box area ratio. The code can be easily adapted to filter images based on other criteria or using different object detection models.

2 Frustrum Overlap Filter

2.1 Design Choice

The design choices made in developing this code include using an overlap threshold to determine which frames to remove, using a target ratio to determine how many frames to keep, and using different methods to calculate and reduce overlap. The JSON file provides camera pose metadata in quaternion format as provided in the Epic fields method in the Epic kitchen dataset [5].
The choice of an appropriate overlap threshold depends on several factors, such as the desired level of redundancy in the data and how much common field of view is required between adjacent frames. A high threshold will result in fewer frames being removed, while a low threshold will result in more frames being removed. This is a parameterised

empirical choice. A high target ratio will result in more frames being kept, while a low target ratio will result in fewer frames.

2.2 Description

This code selects frames from images based on camera frustums' overlap. A camera frustum is the visible volume of a camera, defined by its intrinsic and extrinsic parameters. The code has two functions: *calculate frustum overlap* and *select frames*.
calculate frustum overlap computes the overlap ratio between two frustums using the projection method. It projects the frustums' corners onto a common image plane using a camera matrix *K*, and finds the intersection of their bounding boxes. The overlap ratio is the intersection area divided by the first frustum's area [3].
select frames reads data from a JSON file and extracts camera and image information. It calculates the camera matrix *K* from the camera parameters, and the camera frustums from the image poses. The image poses consist of a quaternion *q* and a translation vector *t*, which are converted to a transformation matrix that transforms the frustums from camera space to world space. The function then removes frames with high overlap until it reaches a target number of frames, using *calculate frustum overlap*.
The code also processes multiple JSON files and copies the selected frames to a destination folder. It calls *select frames* for each file and copies the selected frames to a sub-directory with the same name as the JSON file.

3 Post Processing

3.1 Dark Filter

This code removes dark images from a specified directory. It takes in a directory path and a threshold value as input. The code iterates over all image files in the directory and calculates the average pixel value of each image using the *numpy* library. If the average pixel value is below the given threshold, the image is considered dark and is removed from the directory. The code also prints the name of each removed image to the console. In the example usage, the code removes dark images from the *images* directory with a threshold value of 50.

3.2 Fit

This code resizes and centre-crops images in a specified directory. It takes in a directory path and a size tuple as input. The code iterates over all image files in the directory and calculates the centre crop coordinates for each image using the PIL library. The image is then cropped to the calculated coordinates, resized to the given size using the LANCZOS resampling filter [2], and saved back to the directory
The code also prints the name of each resized and centre-cropped image to the console.

References

[1] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, , Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 130:33–55, 2022.

[2] Claude E. Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology and Climatology*, 18(8):1016 – 1022, 1979.

[3] Nikita Glushkov and Emiliya Tyuleneva. Projection matrices and related viewing frustums: new ways to create and apply. *arXiv preprint arXiv:2009.06516*, 2020.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[5] Vadim Tschernezki, Ahmad Darkhalil, Zhifan Zhu, David Fouhey, Iro Larina, Diane Larlus, Dima Damen, and Andrea Vedaldi. EPIC Fields: Marrying 3D Geometry and Video Understanding, 2023.