Real-time Large-scale Traffic Sign Detection and Recognition

Brief about the project, TODO lists, misc.

**Abbreviations**

TSDR - Traffic Signs Detection and Recognition

# Quick tutorial on code usage (GitHub)

1. The main folder is »findROI«
2. Open findROI/config.m and edit the following lines (you can change others too, but these are essential):
   * param.general.findROIalgorithm – select an algorithm for aROI ('dummy' == fixed ROI); the lattest version is 'smartyColor5\_4'
   * param.general.folderSource – a path to a folder with images (a full dataset is not included in the repository due to size limits). For testing, you can set this to '../data/original' and use only included images).
   * param.general.parallelNumWorkers – number of parallel jobs. Set this to 1 to use one core only.
3. Run and evaluate detector on the selected image(s), few examples:  
   * runAndEvaluate(44, 1, 'none');   
     process image 0000044.jpg, show the final result (second argument), and do not store the results (third argument)
   * runAndEvaluate([1, 44, 119], 0, 'all');   
     process a sequence of images, do not show results, and save bounding boxes, evaluation statistics, and image of ROIs in the folder that is specified in config.m (param.general.folderResults)
4. Process all the data in the source folder – »production mode«:  
     
   runAndEvaluate([], 0, 'results');   
   process all images in param.general.folderSource folder, do not interactively show any results, save bounding boxes with evaluation results. It is recommended to set the number of parallel jobs (param.general.parallelNumWorkers) to 2 more

# Goals

* highly accurate TSDR (mAP > 90%) in real time (>20 FPS?)
* harnessing the power of GPUs (2-3 cards in parallel)
* using large-scale DFGTSD database (provided by Slovenian company DFG Consulting) with 200 categories of traffic signs and ~7000 FHD images
* Benchmark developed pipeline on other databases too (GTSDB, GTSRB, BTS, STSD, LISA, ...)

# Dataset

* DFGTSD database: <https://www.vicos.si/Downloads/DFGTSD>  
  Sizes of images (KB) in this repository are different from Aleksej’s
* Some of the selected images are on GDrive in folder ***data***
* Annotations (v1.1) are in folder ***data/annotations***
* Note: there is original database (with captured images in real world = default) and augmented one with artificially generated images (beneficial for CNN training).

Some datasets we have found through the papers reading:

* Lim2017, Vienna traffic rules (KR-D, KR-N, DE-D): <https://figshare.com/articles/Traffic_Sign_Recognition_Testsets/4597795/1>
* Hasan Fleyeh repository: <http://users.du.se/~hfl/>

# Literature

Some of the most important (?) papers are in the folder ***literature***.

The starting point are papers *Avramovic2018* and *Tabernik2019*.

# Sitemap of repository (GDrive)

* **data**
  + **annotations**
    - **augmented:** ground truth for dataset with augmentation
    - **default:** no augmentation. JSON files were parsed and saved to Matlab format (eg. train.json.mat). We have also joined train and test parts into one for convenience (joined\_train\_test.mat).
  + **masks**: binary masks as the result of image preprocessing and thresholding
    - **RGB\_Aleksej**: initial results by Aleksej (using SaliencyDetection.m)
  + **original**: small subsample of DFGTSD dataset
  + **preprocessed\***: results from preprocessing step with histogram equalization (heq) and color constancy (cc).
  + **tileImages\***: original and preprocessed images are put together for easier comparison and thresholds adjustment (using Matlab colorThresholder app).
* **docs**
* **findROI:** prototype code for finding regions of interest (areas with traffic signs) and helpers for reading annotations, image tiling, etc.
  + **sandbox**: scripts with tests and experimentation.
* **literature**

# Proposed approach

Two-stages approach:

1. fast and coarse detection of traffic signs -> extraction of K patches/ROIs (708 x 708px)
2. YOLO-based TSDR on extracted ROIs only and using K GPUs, one for each ROI

## 1. findROI

TODO

INPUT  
(RGB image)

PREPROCESS

Histogram Eq.

(CLAHE)

Color constancy

???

HSV THRESHOLDING

RGB to HSV

Fixed / adaptive limits for H, S, and V

Thresholds are defined for specific colors (red, blue, yellow, green, brown, white, black)

Mind the theory (Fleyeh2017): S >= 0.5, V >= 0.2

output: binary mask for each color

1. INPUT: RGB image   
   size 1920 x 1080 x 3 uint8 or