Package ‘wuepix’

November 8, 2018

Type Package

Title Using Computer Vision to Count Pedestrians

Version 1.0.1

Author Jeroen Staab

Maintainer Jeroen Staab <jstaab@posteo.de>

Description R-Package for using computer vision to count persons and other objects in webcam imagery. Wuepix is ment to aid geographers counting visitors of touristic destinations as natural reservers protected areas. The package is part of the authors master thesis.

License CC BY-SA 4.0

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

Suggests knitr, rmarkdown, Rdpack

Imports tidyverse, dplyr, purrr, readr, stringi, foreach, doParallel,
jpeg, SDMTools, git2r, magick

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

CD_list *Change Detection*

Description

Detect changes using image differencing for a list of images. Includes parallel processing.

Usage

```
CD_list(img.list, ...)
```

Arguments

`img.list`	file path to images.
`...`	Arguments passed to CD_single().

Value

Classification result. Here work is in progess...

CD_single *Change Detection*

Description

Change Detection

Detect changes between two images using image differencing

Usage

```
CD_single(file.now, file.old, Min = 0.2, Max = 1, predictions = NULL,
  extend = NULL, plot = FALSE, method = "diff")
```

Arguments

`Min` Threshold for positive classification

`Max` Threshold for positive classification

`predictions` dir path to where to store prediction images

`extend` DEPECATED! Used to crop images. Has been moved to a seperate preprocess step.

`method` Select change detection method. "ratio" Image Rationing. "diff" Image Differencing, absolute changes in both directions. "diff+" Image Differencing, positive changes only.

`now` Path to first image

`old` Path to second image

Value

Classification result. Here work is in progess...

`fun_Aggregation` *Aggregate Time-series*

Description

Aggregate Time-series

Usage

```
fun_Aggregation(Timestamp, Variable, T_scale = "hour")
```

Arguments

`T_scale` Timeinterval. See ?lubridate::floor_date()

Value

Dataframe including summed and mean variable per Timeinterval

GTD_list *Sample Ground Truth Data*

Description

Manually asses number of persons in multiple images.

Usage

```
GTD_list(img.list)
```

Arguments

`img.list` file path to image, also known as ‘now‘.

Value

numeric vector with number of persons.

GTD_single *Sample Ground Truth Data*

Description

Manually asses number of persons in a single image.

Usage

```
GTD_single(img)
```

Arguments

`img` file path to image, also known as ‘now‘.

Value

numeric vector with number of persons.

Author(s)

Jeroen Staab

`GTD_truePositives` *Benchmark Pedestrian Detection*

Description

Accuracy Assesment for Object-Based Classifiers as in DALAL etal. 2005 p. 888

Usage

```
GTD_truePositives(GTD, PRD)
```

Arguments

`GTD`	Numeric vector of Ground-Truth-Data, as returned by GTD_list()
`PRD`	Numeric vector of prediction values, as returned by hog_list() and yolo_list()

Value

Returns datafram with following colums: FalseNegative, TruePositives, FalsePositive, MissRate, FalsePositvesPerWindow, correlation

`hog_install` *How to install HOG-Descriptor?*

Description

hog_list() depends on a functional OpenCV installation. This is how I installed it on the LSFE workstation (Linux). OS-specific

OpenCV: `sudo apt install python-opencv`

Package Manager: `sudo apt install python-pip`

HOG Dependency: `pip install imutils`

CUDA GPU: `sudo apt-get install nvidia-cuda-dev nvidia-cuda-toolkit nvidia-nsight`

Usage

```
?hog_install()
```

See Also

http://docs.opencv.org/trunk/df/d65/tutorial_table_of_content_introduction.html

hog_list *Detect pedestrians using HOGDescriptor*

Description

Detect objects using HOG+SVM (implemented in OpenCV) in all Files/Images of 'path'

Usage

```
hog(img.folder)
```

Arguments

`winStride` Window stride. It must be a multiple of block stride.

`padding` Not implemented yet!

`Mscale` Numeric. Allows multi-scale detection. Coefficient of the detection window increase.

`resize` Numeric factor resizing image in integrated pre-processing step. E.g. 2 will double the image extent. People should be 100 pixels high.

`predictions` dir path to where to store prediction images. Must end with "/".

`img.folder` Path to (preprocessed) image archive

Details

Python and OpenCV have to be installed. Tested on Linux only.

Further ideas: [A] Add more 'hog.detectMultiScale' parameters: winStride=(4, 4), padding=(8, 8), scale=1.05) [B] Save predictions.png to a folder

Value

Numeric vector with number of detected persons.

Author(s)

Jeroen Staab

`JPEG_grayscale` *Convert RGB to Grayscale*

Description

Convert RGB img to Grayscale. Default is mixing the three bands equaly. Use camera specific weights if possible. Stardot red=0.3 green=0.59 blue=0.11

Usage

```
JPEG_grayscale(img, red = 1/3, green = 1/3, blue = 1/3)
```

Arguments

`img`	Three layered raster object.
`red`	Calibration weight for red.
`green`	Calibration weight for green.
`blue`	Calibration weight for blue.

Value

Singe layer raster object.

`JPEG_histStrecht` *Histogram Stretching*

Description

Stretch values between 0 and 1 as in JPEG convention. Attention, use this function for plotting only (highlights contrast). But further processing (i.e. Change Detection) may be limited due altered values.

Usage

```
JPEG_histStrecht(img)
```

Arguments

`img`	A raster object.

Value

same as input, but ranged between 0 and 1 (nummeric).

JPEG_plot *Plot a JPEG*

Description

Simply plot an image.

Usage

```
JPEG_plot(img, main = NULL)
```

Arguments

img	A raster object.
main	Optional. An overall title for the plot.

Details

Images can be loaded as raster object using ‘JPEG::readJPEG()‘.

Author(s)

Jeroen Staab

ROI_draw *Inspect a Region of Interest*

Description

Draw a region of interest.

Usage

```
ROI_draw(img)
```

Arguments

img	A raster object.

Details

Insepct a region of interest by drawing a polygone. See OS-specific ?locator() for how to finish drawing. Minimum three vertex points are required.

roi <- InspectROI(jpeg::readJPEG("../Testbild.jpg"))

To visualize roi use histROI() or get it’s stats with summary().

histROI(roi)

Value

numeric dataframe with digital numbers of selected pixels.

See Also

histROI

ROI_hist *Inspect a Region of Interest*

Description

Ggplot histogramm of a region of interest.

Usage

```
ROI_hist(roi)
```

Arguments

roi	A numeric dataframe as returned by InspectROI().

yolo_install *Install YOLO Automatically*

Usage

```
yolo_install(yolo.inst)
```

Arguments

yolo.inst	directory for installation. Will be created

Details

this function wrapped the install procedure (1-3) while renaming the directory from ‘darknet‘ to ‘basename(yolo.inst)‘ and additional run a test to check whether installation succeeded.

1. ’git clone https://github.com/pjreddie/darknet’
2. ’cd darknet’
3. ’make’

during installation ‘Makefile‘ will be opened, to finetune the installation, eg. multithreading (OPENMP=1) or GPU processing (GPU=1), off by default..

after successfull installation it will place ‘yolo.inst‘ in ‘paste0(system.file(package = "wuepix"), "/exec/yolo_inst.txt")‘

yolo_list *Object Detection using YOLO*

Description

detect people using YOLO+CNN (Linux C++), in multiple images. Unfortunately it is not possible to store the predictions here, but it is significant faster on large image archives.

Usage

```
yolo_list(img.list, logfile = "yolo_detections.txt")
```

Arguments

`img.list`	file path to images.
`logfile`	file path to where to store detailed list of classification results.

Value

numeric number of detected persons.

See Also

yolo_single

yolo_Read *Read YOLO Output File*

Description

Read and clean YOLO output file, as saved to working directory by yolo_list()

Usage

```
yolo_Read(file = "yolo_detections.txt")
```

Arguments

`file`	path to output "yolo_detections.txt" file.

yolo_single *Object Detection using YOLO*

Description

detect people using YOLO+CNN (Linux C++), in a single image.

Usage

```
yolo_single(img, logfile = "yolo_detections.txt",
  predictions = "YOLO_Predictions/")
```

Arguments

`img`	file path to image, also known as ‘now‘.
`logfile`	file path to where to store detailed list of classification results.
`predictions`	dir path to where to store prediction images

Details

depends on a working YOLO installation! See ‘yolo_install()‘ and rerun ‘yolo_update()‘ after updating this package (Places yolo.inst in package directory)

single processing allows storing ‘predictions‘ (images with bounding boxes). Since these can be very insightful, you might want to ‘sapply()‘ this function instead of ‘yolo_list()‘. However because then the wights have to be loaded repetitively (~10 seconds) this slows down processing.

it’s recommended avoid spaces in the paths (also in working directory).

an idea for further work on this package would be to actually wrap YOLO into R (e.g. using Rccp).

Value

numeric number of detected persons.

Author(s)

Jeroen Staab

References

\insertRefredmon2016yolo9000wuepix https://pjreddie.com/darknet/yolo/

Examples

```
yolo_single(img)
sapply(img.list, yolo_single)
```

`yolo_update` *Update YOLO*

Description

Update YOLO

Usage

```
yolo_update(yolo.inst)
```

Arguments

`yolo.inst`	directory of YOLO installation.

Details

since YOLO is under active development this function wraps the update procedure (1-2) and test it.

1. 'git pull'

2. 'make'

during installation ‘Makefile‘ will be opened, to fine tune the installation, e.g. turning on multi-threading or GPU processing.

after successfull update it will place ‘yolo.inst‘ in ‘paste0(system.file(package = "wuepix"), "/exec/yolo_inst.txt")‘

Index