BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Detection, Extraction and Measurement of the Contour and Circumference of the Metacarpal Bones in X-rays of the Human Hand

Matej Otčenáš

1 Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 Namespace Documentation

3.1 main Namespace Reference

Functions

- def main ()

3.1.1 Detailed Description

```
main.py: Main module for running the contour detection and bone measurement algorithm.
```

3.1.2 Function Documentation

3.1.2.1 main() `def main.main ( )`

```
Main function starts an algorithm
```

3.2 Modules.canny Namespace Reference

Functions

- def Canny (img_mask)

3.2.1 Detailed Description

```
canny.py: Module uses basic approach for contour extraction using OpenCV
functions such as threshold or Canny edge detection algorithm.
```

3.2.2 Function Documentation

3.2.2.1 Canny() `def Modules.canny.Canny (`
`img_mask )`

```
Method computes edges of given image mask using right threshold and Canny edge
detection algorithm.

Parameters
----------
img_mask : list
    Input image mask

Returns
-------
array
    matrix of an image
```

3.3 Modules.config Namespace Reference

Functions

- def TestMeta ()
- def TestROI ()
- def TrainMeta ()
- def TrainROI ()

Variables

- int MODEL_USE1 = 1
- int MODEL_USE2 = 1

3.3.1 Detailed Description

```
config.py: Configuration module for training and inference. Values are specific
for use of Detectron2 library.
https://detectron2.readthedocs.io/en/latest/tutorials/getting_started.html
https://github.com/facebookresearch/detectron2

NOTE:
    Models were primary trained on Google Colab(https://colab.research.google.com/)
    due to high quality of GPU utilization.
```

3.3.2 Function Documentation

3.3.2.1 TestMeta() `def Modules.config.TestMeta ( )`

```
Function for inference on given dataset of hands based on the trained model. Coming from
'TrainMeta()' function.

Returns
-------
str, str, str, str, int, int, float
    set of multiple parameters for model prediction
```

3.3.2.2 TestROI() `def Modules.config.TestROI ( )`

```
Function for inference on given dataset of ROI of third metacarpal bone
based on the trained model. Coming from
'TrainROI()' function.

Returns
-------
str, str, str, str, int, int, float
    set of multiple parameters for model prediction
```

3.3.2.3 TrainMeta() `def Modules.config.TrainMeta ( )`

```
Function for training custom model based on given image annotations using pretrained
deep neural network (DNN) called Mask RCNN for instance segmentation. Models
utilize COCO(common objects in context) large-scale detection dataset.
Training is provided on full x-ray image of human hand, where the third metacarpal is
the only important.



Returns
-------
str, str, str, str, int, int, int, float, int, int
    set of multiple parameters for model configuration
```

3.3.2.4 TrainROI() `def Modules.config.TrainROI ( )`

```
Function for training custom model based on given image annotations using pretrained
deep neural network (DNN) called Mask RCNN for instance segmentation. Models
utilize COCO(common objects in context) large-scale detection dataset.
Training is provided on region of interest (ROI) of detected bones where
the bone width is shortest.

Returns
-------
str, str, str, str, int, int, int, float, int, int
    set of multiple paremeters for model configuration
```

3.3.3 Variable Documentation

3.3.3.1 MODEL_USE1 `int Modules.config.MODEL_USE1 = 1`

3.3.3.2 MODEL_USE2 `int Modules.config.MODEL_USE2 = 1`

3.4 Modules.roi Namespace Reference

Functions

- def ROI (boxes)

3.4.1 Detailed Description

```
roi.py: Module obtains bounding box points coordinates.
```

3.4.2 Function Documentation

3.4.2.1 ROI()

```
def Modules.roi.ROI (
    boxes )
```

```
Function extracts top left and bottom right point from given bounding box.

Parameters
----------
boxes: array
    Bounding box of processed bone.

Returns
-------
int, int, int, int
    set of coordinates as separate numbers
```

3.5 Modules.train Namespace Reference

Classes

- class Trainer

3.5.1 Detailed Description

```
train.py: Module for training the model based on custom
dataset (annotated metacarpal bones or annotated ROI).
```

3.6 run Namespace Reference

Variables

- ap = argparse.ArgumentParser()
- string arg = 'python -W ignore main.py'
- args = vars(ap.parse_args())
- help
- input_dir = args["input"]
- output_dir = args["output"]
- required
- start_image = args["name"]

3.6.1 Detailed Description

```
run.py: This module starts the entire third metacarpal edgde detection and measurement program.
```

3.6.2 Variable Documentation

3.6.2.1 ap `run.ap = argparse.ArgumentParser()`

3.6.2.2 arg `string run.arg = 'python -W ignore main.py'`

3.6.2.3 args `run.args = vars(ap.parse_args())`

3.6.2.4 help `run.help`

3.6.2.5 input_dir `run.input_dir = args["input"]`

3.6.2.6 output_dir `run.output_dir = args["output"]`

3.6.2.7 required `run.required`

3.6.2.8 start_image `run.start_image = args["name"]`

3.7 setup Namespace Reference

Variables

- ext_modules
- name

3.7.1 Detailed Description

```
setup.py: Setup module for creating shared object libraries and C files for precompiling the program to faster
```

3.7.2 Variable Documentation

3.7.2.1 ext_modules `setup.ext_modules`

3.7.2.2 name `setup.name`

4 Class Documentation

4.1 Modules.train.Trainer Class Reference

Public Member Functions

- def __init__ (self, COCO_NAME, COCO_ANNOTS, MODEL_TYPE, WEIGHT_PATH, MAX_ITER_META, NUM_WORKERS, IMS_PER_BATCH, BASE_LR, BATCH_SIZE_PER_IMAGE, NUM_CLASSES)
- def run (self)

Public Attributes

- BASE_LR
- BATCH_SIZE_PER_IMAGE
- COCO_ANNOTS
- COCO_NAME
- IMS_PER_BATCH
- MAX_ITER_META
- MODEL_TYPE
- NUM_CLASSES
- NUM_WORKERS
- WEIGHT_PATH

4.1.1 Detailed Description

```
Description
-------
Trainer class including method for training the model.

Methods
-------
run()
    Starts the training process
```

4.1.2 Constructor & Destructor Documentation

4.1.2.1 __init__()

```
def Modules.train.Trainer.__init__ (
        self,
        COCO_NAME,
        COCO_ANNOTS,
        MODEL_TYPE,
        WEIGHT_PATH,
        MAX_ITER_META,
        NUM_WORKERS,
        IMS_PER_BATCH,
        BASE_LR,
        BATCH_SIZE_PER_IMAGE,
        NUM_CLASSES )
```

```
Parameters
----------
COCO_NAME: str
    Image annotations exported to '.json' file in COCO format (https://roboflow.com/formats/coco-json)
COCO_ANNOTS: str
    Name of the annotated images folder
MODEL_TYPE: str
    Loading model zoo configuration, using  ResNet and FPN(Feature Pyramid Networks) backbone
WEIGHT_PATH: str
    Loading pre-trained weights based on model zoo
    (https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md) for instance segmentation
MAX_ITER_META: int
    Number of iterations
NUM_WORKERS: int
    Number of parallel data loading workers
IMS_PER_BATCH: int
```

```
    Number of images per batch across all machines (depends on number of GPUs), each GPU will see 2 images per
BASE_LR: float
    Hyperparameter that controls how much to change the model in response to the
    estimated error each time the model weights are updated (it has big impact for resulting model)
BATCH_SIZE_PER_IMAGE: int
    Number of samples(images) that will be propagated through the network
NUM_CLASSES: int
    Number of thing classes for R-CNN
```

4.1.3 Member Function Documentation

4.1.3.1 run()

```
def Modules.train.Trainer.run (
     self )
```

```
Method for starting the trainig process including configuration for Detectron2
```

4.1.4 Member Data Documentation

4.1.4.1 BASE_LR
`Modules.train.Trainer.BASE_LR`

4.1.4.2 BATCH_SIZE_PER_IMAGE
`Modules.train.Trainer.BATCH_SIZE_PER_IMAGE`

4.1.4.3 COCO_ANNOTS
`Modules.train.Trainer.COCO_ANNOTS`

4.1.4.4 COCO_NAME
`Modules.train.Trainer.COCO_NAME`

4.1.4.5 IMS_PER_BATCH
`Modules.train.Trainer.IMS_PER_BATCH`

4.1.4.6 MAX_ITER_META
`Modules.train.Trainer.MAX_ITER_META`

4.1.4.7 MODEL_TYPE `Modules.train.Trainer.MODEL_TYPE`

4.1.4.8 NUM_CLASSES `Modules.train.Trainer.NUM_CLASSES`

4.1.4.9 NUM_WORKERS `Modules.train.Trainer.NUM_WORKERS`

4.1.4.10 WEIGHT_PATH `Modules.train.Trainer.WEIGHT_PATH`

The documentation for this class was generated from the following file:

- train.py

Index