Visionkit

人間画像認識

# -\*- coding: utf-8 -\*-

import time

import sys

import os

import traceback

import json

import subprocess

import paramiko

import scp

import datetime

# visionkitのIPアドレス、インターフェース、ユーザー名、パスワード

class VK(object):

\_\_VK\_IP = '127.0.0.1'

\_\_VK\_PORT = 22

\_\_VK\_USER = 'pi'

\_\_VK\_PASSWD = 'raspberry'

def \_\_init\_\_(self, ip, port, user, passwd):

self.\_\_VK\_IP = ip

self.\_\_VK\_PORT = port

self.\_\_VK\_USER = user

self.\_\_VK\_PASSWD = passwd

# SSH接続する

def \_\_ssh\_connect(self, host, port, user, passwd):

ssh = paramiko.SSHClient()

ssh.load\_system\_host\_keys()

ssh.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

ssh.connect(host, username=user, password=passwd, port=port, timeout=20, look\_for\_keys=False)

return ssh

# SSH切断する

def \_\_ssh\_disconnect(self, ssh):

ssh.close()

# scpでファイル転送する(ローカル->リモート)

def \_\_put\_file(self, ssh, f\_file, t\_file):

with scp.SCPClient(ssh.get\_transport()) as rcp:

rcp.put(f\_file, t\_file)

return True

# scpでファイル転送する(リモート->ローカル)

def \_\_get\_file(self, ssh, f\_file, t\_file):

with scp.SCPClient(ssh.get\_transport()) as rcp:

rcp.get(f\_file, t\_file)

return True

# sshでコマンドを実行する

def \_\_remote\_exec(self, ssh, cmd):

stdin, stdout, stderr = ssh.exec\_command(cmd)

out = ''

err = ''

for o in stdout:

out = out + o

for e in stderr:

err = err + e

return out, err

# コマンドを実行して、標準出力を結果として返す

def \_\_do\_cmd(self, ssh, cmd, opt):

# 実行コマンドを組み立てる

c = cmd +' ' + ' '.join(opt)

# 実行

o, e = self.\_\_remote\_exec(ssh, c)

return o, e

def \_\_copy\_file(self, ssh, f\_file, t\_file):

# ホストとファイルを取得

f\_h = t\_h = None

if f\_file.find(':') > 0:

f\_h, f\_f = f\_file.split(':')

else:

f\_f = f\_file

if t\_file.find(':') > 0:

t\_h, t\_f = t\_file.split(':')

else:

t\_f = t\_file

if t\_h != None and f\_h != None:

print('リモート間のコピーはできません:(' + f\_h + ' -> ' + t\_h + ')')

return False

if t\_h == None and f\_h == None:

print('ローカル間のコピーはできません')

return False

# ディレクトリ取得

d = os.path.dirname(t\_f)

try:

# ディレクトリ作成&ファイルコピー

if t\_h != None:

if d != '' and d != '.':

# リモートにディレクトリ作成

self.\_\_do\_cmd(ssh, 'mkdir', ['-p', d])

# コピー

self.\_\_put\_file(ssh, f\_f, t\_f)

else:

if d != '' and d != '.':

# ローカルにディレクトリ作成

os.makedirs(d)

# コピー

self.\_\_get\_file(ssh, f\_f, t\_f)

except Exception as e:

print('!!ERROE!!')

print(e)

return False

return True

def vk\_put\_file(self, f\_file, t\_file):

# SSH準備

ssh = self.\_\_ssh\_connect(self.\_\_VK\_IP, self.\_\_VK\_PORT, self.\_\_VK\_USER, self.\_\_VK\_PASSWD)

# コピー

res = self.\_\_copy\_file(ssh, f\_file, self.\_\_VK\_IP + ':' + t\_file)

# SSH切断

self.\_\_ssh\_disconnect(ssh)

return res

def vk\_get\_file(self, f\_file, t\_file):

# SSH準備

ssh = self.\_\_ssh\_connect(self.\_\_VK\_IP, self.\_\_VK\_PORT, self.\_\_VK\_USER, self.\_\_VK\_PASSWD)

# コピー

res = self.\_\_copy\_file(ssh, self.\_\_VK\_IP + ':' + f\_file, t\_file)

# SSH切断

self.\_\_ssh\_disconnect(ssh)

return res

# Vision Kitのコマンドを実行して結果を受け取る

def vk\_exec(self, cmd, opt):

# SSH準備

ssh = self.\_\_ssh\_connect(self.\_\_VK\_IP, self.\_\_VK\_PORT, self.\_\_VK\_USER, self.\_\_VK\_PASSWD)

# コマンド実行

o, e = self.\_\_do\_cmd(ssh, cmd, opt)

# SSH切断

self.\_\_ssh\_disconnect(ssh)

if len(e) != 0:

print("ERROR:" + e)

return o

# 画像認識realtime

def vk\_raspistill(self,out\_filepath):

#print(out\_filepath.split('/')[-1])

filename = out\_filepath.split('/')[-1]

#print("filename:",filename)

if out\_filepath != None:

o\_file = os.path.normpath(os.path.join('data/', filename))

else:

o\_file = '\_out.jpg'

#print("o\_file:",o\_file)

out = self.vk\_exec('raspistill',['-o '+ "data/"+filename])

print("###exec raspistill func###")

pic = os.path.join('data/', filename)

t\_file = os.path.normpath(os.path.join('data/', filename))

# 認識結果画像を取得

#print("o\_file, o\_pic,out\_filepath",o\_file, out\_filepath,o\_pic)

if out\_filepath != None:

res = self.vk\_get\_file(o\_file, filename)

# 削除

res = self.vk\_exec('rm -f', [t\_file])

return out

# 画像認識

def vk\_recog\_image(self, pic, o\_pic):

# os.path.join()では/が入っているとそこがrootになってしまうので、/で分解して結果のリストを分解して渡す

t\_file = os.path.normpath(os.path.join('CQ/MyServer/VK/', \*pic.split('/')))

if o\_pic != None:

o\_file = os.path.normpath(os.path.join('CQ/MyServer/VK/', \*o\_pic.split('/')))

else:

o\_file = '\_out.jpg'

# コピー

res = self.vk\_put\_file(pic, t\_file)

# 認識

out = self.vk\_exec('~/AIY-voice-kit-python/src/examples/vision/object\_detection.py', ['--input '+ t\_file, '--output ' + o\_file])

# 認識結果画像を取得

if o\_pic != None:

res = self.vk\_get\_file(o\_file, o\_pic)

# 削除

res = self.vk\_exec('rm -f', [t\_file])

return out

# stream boxing

def vk\_box\_stream(self,picname):

#vk\_output\_file = "human.jpg"

#vk\_output\_file = os.path.normpath(os.path.join('data/',picname ))#faces.jpg

#pi\_output\_file = os.path.normpath(os.path.join('',picname ))

# ??

out = self.vk\_exec('~/AIY-voice-kit-python/src/examples/vision/face\_detection\_camera.py', [])

# ?????????

#if pi\_output\_file != None:

#res = self.vk\_get\_file(vk\_output\_file, pi\_output\_file)

# ??

#res = self.vk\_exec('rm -f', [vk\_output\_file])

return out

# face boxing one time

def vk\_box\_face(self,picname):

vk\_output\_file = "human.jpg"

#vk\_output\_file = os.path.normpath(os.path.join('data/',picname ))#faces.jpg

pi\_output\_file = os.path.normpath(os.path.join('',picname ))

# ??

out = self.vk\_exec('~/AIY-voice-kit-python/src/examples/vision/face\_detection\_rectangle.py', [])

# ?????????

if pi\_output\_file != None:

res = self.vk\_get\_file(vk\_output\_file, pi\_output\_file)

# ??

res = self.vk\_exec('rm -f', [vk\_output\_file])

return out

# 画像認識

def vk\_recog\_face(self,picname):

vk\_output\_file = "faces.jpg"

#vk\_output\_file = os.path.normpath(os.path.join('data/',picname ))#faces.jpg

pi\_output\_file = os.path.normpath(os.path.join('',picname ))

# 認識

out = self.vk\_exec('~/AIY-voice-kit-python/src/examples/vision/face\_camera\_trigger.py', [])

# 認識結果画像を取得

if pi\_output\_file != None:

res = self.vk\_get\_file(vk\_output\_file, pi\_output\_file)

# 削除

res = self.vk\_exec('rm -f', [vk\_output\_file])

return out

# 猫が写っているか判定

'''

object\_detection.pyの出力は以下の形式。

Object #0: kind=CAT(2), score=0.871475, bbox=(75, 166, 485, 351)

Object #1: kind=CAT(2), score=0.557364, bbox=(329, 144, 266, 329)

kind=に、PERSON,CAT,DOGのように認識したオブジェクトが表示される。

認識されない場合は何も出力されない。

'''

def vk\_is\_cat(self, pic, result):

# 認識処理

res = self.vk\_recog\_image(pic, result)

# 猫かどうか確認

if 'kind=CAT' in res:

return True

else:

return False

def vk\_is\_human(self,pic,result):

# 認識処理

res = self.vk\_recog\_image(pic, result)

print(res)

# humanかどうか確認

if 'kind=PERSON' in res:

return True

else:

return False

def vk\_human\_detect(self,interval):

n=0

N=10

while(True):

now\_time = str(datetime.datetime.now().strftime("%H{H}%M{M}%S{S}").format(H="h",M="m",S="s"))

#print(now\_time)

#filename

picname = now\_time+"test.jpg"

#if there is a human face,photograph it and save img to pi\_output\_file

out=vk.vk\_recog\_face(picname)

pi\_output\_file = os.path.normpath(os.path.join('',picname ))

#test its human or not

res = vk.vk\_is\_human(pi\_output\_file, './'+now\_time+ 'result.jpg')

if(res==True):

print("is human!",res)

time.sleep(interval)

else:

print("is not human!",res)

time.sleep(interval)

if(n==N):

break

# コマンドラインモード、テストモード

if \_\_name\_\_ == "\_\_main\_\_":

#vk = VK('raspberrypi.local', 22, 'pi', 'raspberry')

vk = VK('192.168.11.4', 22, 'pi', 'raspberry')

#vk = VK('192.168.11.4', 22, 'pi', '123456')

print("--"\*20)

print("-----------connect to visionkit--------------")

print(vars(vk))

print("---------visionkit connect success!-------------")

print("--"\*20)

"""

picname="yutatest.jpg"

out=vk.vk\_recog\_face(picname)

print(out)"""

#vk.vk\_human\_detect(1)

vk.vk\_human\_detect(interval=1)

#res = vk.vk\_is\_human()

#print(res)

#res = vk.vk\_is\_cat('./people.jpg', './people\_result.jpg')

#print(res)

"""res = vk.vk\_is\_cat('./cats.jpg', './cats\_result.jpg')

print(res)

res = vk.vk\_is\_cat('./cat2.jpg', './cat2\_result.jpg')

print(res)

res = vk.vk\_is\_cat('./human.jpg', './human\_result.jpg')

print(res)

res = vk.vk\_is\_cat('./dog.jpg', './dog\_result.jpg')

print(res)

res = vk.vk\_is\_cat('./city.jpg', './city\_result.jpg')

print(res)"""