

USB Camera

[usbwebcamera.py]

usbwebcamera.py

```
import cv2
import threading
import time
import logging
import os

logger = logging.getLogger(__name__)
archive_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'archive')

thread = None

class UsbWebCamera:

    def __init__(self, fps=20, video_source=1):
        logger.info("Initializing usb camera class with {fps} " + str(fps) + " and video_source={ " + str(video_source) + "}")
        self.fps = fps
        self.video_source = video_source
        self.camera = cv2.VideoCapture(self.video_source)
        print( "1: width: {}, height : {}".format(self.camera.get(3), self.camera.get(4) ) )
        #width=self.camera.get(cv2.CAP_PROP_FRAME_WIDTH)
        #height=self.camera.get(cv2.CAP_PROP_FRAME_HEIGHT)
        #print( "2: width: {}, height : {}".format(width, height) )
        self.camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        self.camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

        # We want a max of 5s history to be stored, thats 5s*fps
        self.max_frames = 5*self.fps
        self.frames = []
        self.isrunning = False

    def run(self):
        logging.debug("Perparing Usb Camera thread")
        global thread
        if thread is None:
            logging.debug("Creating thread")
            thread = threading.Thread(target=self._capture_loop, daemon=True)
            logger.debug("Starting thread")
```

```

        self.isrunning = True
        thread.start()
        logger.info("Thread started")

def _capture_loop(self):
    dt = 1/self.fps
    logger.debug("Observation started")
    while self.isrunning:
        v,im = self.camera.read()
        #im = cv2.flip(im, 1)
        #im = cv2.flip(im, 0)
        if v:
            if len(self.frames)==self.max_frames:
                self.frames = self.frames[1:]
            self.frames.append(im)
        time.sleep(dt)
    logger.info("Thread stopped successfully")

def stop(self):
    logger.debug("Stopping thread")
    self.isrunning = False

def get_frame(self, _bytes=True):
    if len(self.frames)>0:
        if _bytes:
            img = cv2.imencode('.png',self.frames[-1])[1].tobytes()
        else:
            img = self.frames[-1]
    else:
        with open(archive_path+"/not_found.jpeg","rb") as f:
            img = f.read()
    return img

```

🔄Revision #2

★Created 30 December 2023 13:52:10 by Hyeon Su Ryu

✎Updated 30 December 2023 14:23:53 by Hyeon Su Ryu