Streaming Webcam Input with OpenCV

Confirming input from Logitech HD Pro Webcam C920 (connected to the beagle)

Dependencies:

Hardware Used: This section is primarly for the BeagleY-AI and Logitech HD Pro Webcam C920.

It has also been tested with the Logitech Webcam C500 so in theory, this should work with any usb webcam.

Software Used (Download instructions are provided when necesarry): fswebcam, v4l2 (video for linux 2)

Steps

1: Confirm Connection

We will check which file(s) the usb webcam is on with V4L2 and take the actual picture with fswebcam.

Install the following drivers:

```
(target)$ sudo apt update && sudo apt upgrade
(target)$ sudo apt install fswebcam
(target)$ sudo apt install v4l2-ctl
```

Note: ffmpeg is another option, but it didn't work for us

List the devices:

```
(target)$ v4l2-ctl --list-devices
```

It should be listed under HD Pro Webcam C920. (or your webcam of choice)

Example output:

```
e5010 (platform:e5010):
    /dev/video2
```

```
wave5-dec (platform:wave5-dec):
    /dev/video0

wave5-enc (platform:wave5-enc):
    /dev/video1

HD Pro Webcam C920 (usb-xhci-hcd.5.auto-1.3):
    /dev/video3 # Keep note of the ones under /dev/video$(n)
    /dev/video4 # These are the ones we want
    /dev/media0
```

2. Take a Photo

Let's create a folder to keep our photos:

```
(target)$ mkdir ~/webcam_pics/
```

Then you can choose to directly pass args:

```
(target)$ fswebcam -d /dev/video3 ~/webcam_pics/webcam.jpg
```

Or make a config file for easier option customization:

```
(target)$ nano ~/.fswebcam.conf
```

Copy the following text into your config file:

```
# Save as ~/.fswebcam.conf
device /dev/video0
resolution 640x480
skip 5          # Skip first 5 frames (avoid dark/glitchy images)
jpeg 85         # JPEG quality (1-100)
save /home/debian/webcam_pics/webcam.jpg
timestamp       # Optional: Add timestamp overlay
```

Passing the file to fswebcam as an arg:

```
(target)$ fswebcam -c ~/.fswebcam.conf
```

3. View the Photo (Copying to Host)

Note: There is a GUI for the beagle-y-ai, so it's possible to connect the board to a monitor and view the image directly.

Use the Secure Copy Protocol from the host to receive photos.

```
(host)$ scp debian@192.168.7.2:/home/debian/webcam_pics/webcam.jpg ~/Pictures/beagle_webcam
```

Note: using `~` in place of `home/debian/` may not work

It's possible to sync the directories if you want to avoid using scp everytime.

```
(host)$ rsync -avz debian@beagleboard-ip:/home/debian/webcam_pics/ ~/Pictures/webcam_pics/
```

Using OpenCV

1. Get OpenCV on the Target

```
(target)$ sudo apt update && sudo apt upgrade
(target)$ sudo apt install libopencv-dev
```

2. Get OpenCV on the Host

Note: This may take a while, have a movie ready or something

Get dependencies, you should have all of these anyways.

```
(host)$ sudo apt install cmake g++ build-essential
```

Install OpenCV

Note: For the INSTALL PROJECT SECTION, using -j$(nproc) uses all your cores and may crash your device if using a computer with less processing power.

```
# GET FROM REPO
(host)$ git clone https://github.com/opencv/opencv.git
(host)$ cd opencv

# BUILD PROJECT
```

```
(host)$ mkdir build && cd build
(host)$ cmake -D CMAKE_BUILD_TYPE=Release \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules \
    -D BUILD_EXAMPLES=OFF \
    -D BUILD_TESTS=OFF \
    -D BUILD_PERF_TESTS=OFF \

# INSTALL PROJECT
(host)$ make -j$(nproc)     # My laptop crashed with nproc, consider using less processes
(host)$ sudo make install
(host)$ sudo ldconfig
```

3. Run a Simple Example

Try running this simple snippet. If all goes well, it should download the image

```
#include <opencv2/imgcodecs.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/videoio.hpp>

#include <iostream>

// Do you remember your paths?
#define WEBCAM_PATH1 "/dev/video3"
#define WEBCAM_PATH2 "/dev/video4"

// Don't use "~" over /home/debian/
// Make sure /home/debian/webcam_pics exists
#define IMG_PATH "/home/debian/webcam_pics/VideoCaptureImg.jpg"

int main() {
    cv::Mat img;
    cv::VideoCapture cap(WEBCAM_PATH1, cv::CAP_V4L2);
    if (!cap.isOpened()) {
        std::cerr << "Unable to connect to camera" << std::endl;
    }

    // Clear buffer + garbage read dead frames
    for (int i = 0; i < 5; ++i) {
        cap.read(temp);
    }
    // The actual read
    if (!cap.read(temp)) {
        std::cerr << "Failed to read from webcam");
    }
    cv::imwrite(IMG_PATH, temp);
}
```