Motion Detecting Camera

Jack Myers

1 Neccessary Software

OpenCV: Here is a link to the installation
imutils: Type 'pip install imutils' without the quotes in the command line

2 Necessary Hardware

Raspberry Pi Camera Module V2 - 8 Megapixel,1080p

3 What Does It Do?

The purpose of this software is to detect motion and record video when it detects motion. To do this I use the OpenCV software to process video from the raspberry pi camera. OpenCV is extremely powerful and there are many ways one can achieve the desired result, however, I found the background subtractor method to be the most reliable. Using cv2.createBackgroundSubtractorMOG2() to create a background subtractor, I then apply this background subtractor to each frame of a live stream. The output, called the mask, is a black, white, and gray image where the background (the unmoving parts) is black, the foreground is white, and the shadows are gray. Using a binary threshold, I eliminate the shadows from the mask and we are left with a black and white image. I also use morphology methods to eliminate noise. From this we can use OpenCV to draw contours around the white parts of the mask. However, even after the attempted noise reduction, white spots can still appear and so there is a limit to how small a contour can be for it to be counted. If there are one or more contours, the frame is considered to have motion and is saved to the disk. In addition to saving video, the program can also time stamp motion and save it to a text file. For example, if motion is detected on 06-12-2018 at 13:47:23 for 35.345 seconds, it will save this as one line in the text file. Each successive detection of motion will add lines to the text file. The format of the video or text files are MM-DD-YYYY_N.avi or .txt where N is the Nth file saved on that day, that way files saved on the same day of the same type aren't overwritten.
Upon running mcamera.py, a window will appear displaying a live feed from the raspberry pi camera. In the bottom left corner is a clock that displays the current time in 24 hour format.
Keyboard Input:

<d>: toggle debugging; when debugging, bounding boxes are drawn and the mask is displayed in a separate window. Debugging is displayed in upper right when enabled.
<r>: toggle recording; when recording, video is saved when motion is detected(Note that when debugging is enabled, the saved video has bounding boxes). Recording is displayed in upper left when enabled.
<t>: toggle time keeping, when enabled motion times are stored in a text file(Note that recording does not need to be enabled). Time in lower left corner turns green when enabled.
<q>: Quits the program.

4 Results

The resulting program can be used for surveillance, motion traps for nature videography, gathering data on busiest times of day in public space, and much more. The most interesting application I thought of, which is why I added the ability to save time stamps, is gathering data. For example, the camera and raspberry pi could be placed at the entrance to the campus library. Using the time stamped text file, the user could average the duration of motion detected for each hour every day over many days. The user could then plot a histogram to see what time the library is most busy. This can be expanded to days that the library is most busy and so on. This time stamp along with the available video recording provides a lot of data and a visual of what is going on with that data allowing for a multidude of social and scientific experiments.
In the .tar file for this project are three other files besides mcamera.py: 06-12-2018_1.avi, 06-12-2018_1.txt, and mask.avi. The first is the video file, where the camera is set up in my house and it shows me searching for my wallet, leaving to get food, and finally coming back and turning off the recording. Notice that when motion is detected there are, in general, green bounding boxes around the areas of motion. This indicates that the debugging mode was enabled while it was recording. You may also notice that when I enter through the door there are many bounding boxes where no motion is taking place. This is due to the drastic change in lighting when opening the door throwing off the algorithms for background subtraction. This goes to show that this software is not reliable for any sort of tracking, however, because drastic lighting changes are usally accompanied by some moving object, this program is pretty dependable when it comes to motion detection. The second file is a text file that contains all of the time stamps from the video associated with the first file. The last of the three files, mask.avi, is a sample video showing what my arm looks like when the background subtractor is applied, noise is reduced, and shadows are removed. To view the .avi files, type omxplayer and then the file you'd like to play.

5 Links That I Used To Help Me Learn

PyImageSearch
OpenCV Python Tutorial