CMPSC 370
Introduction to Artificial Intelligence
Fall 2016
Janyl Jumadinova

Lab 4 Part 1
4 October, 2016
Due: 18 October by 2:30 pm
This is a team lab (of two or three).

Objectives

To learn how to use a computer vision `OpenCV` software and a Python programming language for the problem of video surveillance. To correctly install and set up OpenCV on a Raspberry Pi computer and properly connect and use the Raspberry Pi camera. To correctly apply OpenCV motion detection algorithms on stand-alone videos in a computer and continuous detection with Raspberry Pi and a camera.

Overall Goals and Structure

This laboratory assignment is divided into two 2-week parts. The stated deliverables for each part of the lab are to be submitted by the due date. This lab will consist of three separate (3 lab grades): one for the tasks in each part of the lab and one for the overall system design, experimentation and analysis. The second part of the lab will be given on October 18 and the completed, well-documented lab with the results of your experiments is due on November 1.

During the first portion of the lab, you will learn how to use OpenCV for motion detection and complete all of the appropriate installations on your Raspberry Pi. The first portion of the lab is built on the tutorials by Adrian Rosebrock.

1. Basic Motion Detection and Tracking

You will experiment with OpenCV and Python by going through the given programs and explanations. If you are not familiar with a Python programming language, you may refer to Python library documentation at `https://docs.python.org/2/library/`. You will also find additional introductory materials on Python in the `lab4` directory in the shared repository.

You will first build basic motion detection and tracking system for surveillance using computer vision techniques. The provided program works with pre-recorded videos and you will be developing this system on a desktop or a laptop. There are many algorithms in OpenCV for performing motion detection, tracking, and analysis. In motion detection we generally assume that the background is mostly static, this way we can model it and then monitor for big changes. Many of the algorithms in OpenCV (Gaussian Mixture Model, Bayesian Model) use sophisticated methods to segment the background from the foreground. Since our goal is to perform motion detection and tracking on a Raspberry Pi, we will start with a very simple approach.

(a) Understand and explore the given program

In the `cs370f2016-share/lab4/part1/basic-motion-detection` directory, you will find `MotionDetector` program and two videos to use with this program. As you read the explanations of the program below, map each step to the lines in the program.

- The first few lines import the necessary packages. If the `imutils` package, which is a set of convenience functions to make basic image processing tasks easier, does not work for you, you can install it locally via pip: `pip install imutils`.
- Next, we parse the command line arguments. The `--video` argument is optional. It simply defines a path to a pre-recorded video file that you can detect motion in. If you do not supply a path to a video file, then OpenCV will utilize your webcam to detect motion.
- Then, we define `--min-area`, which is the minimum size (in pixels) for a region of an image to be considered actual motion (small regions of an image could change substantially without motion, likely due to noise or changes in lighting conditions). That's why we define a minimum size of a region to combat and filter out these false-positives.
- The next few lines handle grabbing a reference to the camera object. In the case that a video file path is not supplied, we take a reference to the webcam. And if a video file is supplied, then we create a pointer to it.
- Next, we declare and initialize a variable called `firstFrame`, which stores the first frame of the video file/webcam stream. We make an assumption that the first frame of our video file will contain no motion and just background - therefore, we can model the background of our video stream using only the first frame of the video.
- Now we can start looping over each of the frames. A call to `camera.read()` returns a 2-tuple. The first value of the tuple is grabbed, indicating whether or not the frame was successfully read from the buffer. The second value of the tuple is the frame itself.
- We also define a string named `text` and initialize it to indicate that the room we are monitoring is "Unoccupied". If there is indeed activity in the room, we can update this string.
- The `break` statement (to get out of the loop) is used in the case that a frame is not successfully read from the video file.
- Now we can start processing our frame and preparing it for motion analysis. We first resize it down to have a width of 500 pixels - there is no need to process the large, raw images straight from the video stream. We also convert the image to grayscale since color has no bearing on our motion detection algorithm. Finally, we apply Gaussian blurring to smooth our images. It is important to understand that even consecutive frames of a video stream will not be identical! Due to tiny variations in the digital camera sensors, no two frames will be 100% the same as some pixels will most certainly have different intensity values. However, we need to account for this and apply Gaussian smoothing to average pixel intensities across an 11 x 11 region. This helps smooth out high frequency noise that could hinder the motion detection algorithm.

- Since we need to model the background of our image, we make the assumption that the first frame of the video stream contains no motion and is a good example of what our background looks like. If the `firstFrame` is not initialized, we store it for reference and continue on to processing the next frame of the video stream.

- Given the static background image (modeled through `firstFrame` variable, we can now actually perform motion detection and tracking. We use `firstFrame` variable to calculate the difference between the initial frame and subsequent new frames from the video stream. Calculating the difference between two frames is a simple subtraction, where we take the absolute value of their corresponding pixel intensity differences as : $delta = |background\_modelcurrent\_frame|$

- To show regions of the image that only have significant changes in pixel intensity values, we apply thresholding to the `frameDelta`, such that if the delta is less than 25, we discard the pixel and set it to black (i.e. background), and if the delta is greater than 25, we set it to white (i.e. foreground).

- Now, given this thresholded image, we apply contour detection to find the outlines of these white regions. We loop over each of the contours, where we filter the small, irrelevant contours. If the contour area is larger than our supplied `--min-area`, we draw the bounding box surrounding the foreground and motion region. We also update our text status string to indicate that the room is "Occupied".

- Finally, we draw the room status on the image in the top-left corner, followed by a timestamp (to make it feel like "real" security footage) on the bottom-left. The results allow us to visualize if any motion was detected in our video, along with the frame delta and thresholded image so we can debug our program.

(b) Apply the given program to your own videos

Now you need to record at least **two** of your own short videos and apply the basic motion detection and tracking program from the previous step to them. You will likely need to tune the values for `cv2.threshold` and the `--min-area` argument to obtain the best results for your lighting conditions.

Take a screenshot of your program running each of your videos.

2. OpenCV and Python: Raspberry Pi Set Up

(a) Setting up Raspberry Pi

You need to have a monitor, keyboard and a mouse to use a raspberry pi. Alternatively, you can use your laptop's screen, keyboard and a mouse (or a trackpad). The external monitor must have HDMI interface and USB keyboard and mouse, or you can use HDMI to VGA converter for most of the lab machines. If you use lab machines, please make sure to connect everything back to the computer after you finished using them for your Raspberry Pi. Follow the following steps to set up your pi:

- Plug in the keyboard, mouse and monitor cables.
- Plug in the USB power cable to your raspberry pi.
- The raspberry pi should now boot, and a window with a list of different operating systems that you can install should appear. An operating system should be already installed on Raspberry Pis, but if yours doesn't, you can install Raspbian (click the box next to Raspbian or another OS you wish to use and click on `Install`).
- The default login for Raspbian is **username: pi** with the **password: raspberry**.
- There is also a GUI that you can use with a raspberry pi. To load GUI type `startx`.
- You are now ready to use your raspberry pi!

To be able to use your laptop with a raspberry pi, you may follow many tutorials online (search for "connect raspberry pi to laptop"). For example, you may follow instructions on the website `https://www.raspberrypi.org/blog/use-your-desktop-or-laptop-screen-and-keyboard-with-your-pi/` to connect to your Raspberry Pi through your laptop.

(b) Setting up OpenCV on the Raspberry Pi

In order to use OpenCV on a Raspberry Pi with camera, you will need to correctly install OpenCV and all relevant packages. OpenCV 2.4 that works with Python 2.7 is already installed on all Alden machines. If you would like to install it correctly on your laptop, or if you decide to use a newer version of OpenCV and Python, and to install it on a Raspberry Pi, use the installation instructions on:
`http://www.pyimagesearch.com/opencv-tutorials-resources-guides/`
During our next laboratory session, I will walk around and check on the correct set up of OpenCV on the Raspberry Pis.

(c) Using Raspberry Pi Camera on the Raspberry Pi computer

Follow the tutorial on `https://www.raspberrypi.org/learning/getting-started-with-picamera/worksheet/` to learn how to use and then experiment with Raspberry Pi camera. Take a snapshot of your camera functioning correctly.

Required Deliverables

This assignment invites you to submit electronic versions of the following deliverables through your Bitbucket repository (`cs370f2016-<your-username>/lab4/part1/`).

1. Two screenshots from running the basic motion detection on two different videos (your own).
2. A demonstration of a correct OpenCV set up on a Raspberry Pi.
3. A screenshot of the functioning Raspberry Pi camera.