Miguel Millan

Image Processing and Motion Detection Using OpenCV

# Goal:

With this research I’d like to develop a deeper understanding of image processing, and I will be using OpenCV as the basis for my understanding since many methods and algorithms are already implemented in this Library.

The result of this research should be an easily understood procedure for implementing motion detection.

# Findings:

List of referenced sites:

* <https://corochann.com/basic-image-processing-tutorial-1220.html>
* <https://github.com/vdumoulin/conv_arithmetic>
* <http://www.ee.surrey.ac.uk/CVSSP/Publications/papers/KaewTraKulPong-AVBS01.pdf>
* <http://www.zoranz.net/Publications/zivkovic2004ICPR.pdf>
* <http://www.zoranz.net/Publications/zivkovicPRL2006.pdf>
* <https://goldberg.berkeley.edu/pubs/acc-2012-visual-tracking-final.pdf>
* <https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/>
* <https://www.pyimagesearch.com/2015/06/01/home-surveillance-and-motion-detection-with-the-raspberry-pi-python-and-opencv/>
* <https://en.wikipedia.org/wiki/Gaussian_blur>
* <https://www.ancient-asia-journal.com/articles/10.5334/aa.06113/>
* <https://www.docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=gaussianblur#cv2.GaussianBlur>

We can represent colored images as 3-dimensional arrays (where 1st dimension is for height, 2nd is for width, and the 3rd dimension is for color channel (such as RGB)), or gray scale images as 2-dimensional arrays (height, width).

Image.shape => (height, width, channel)

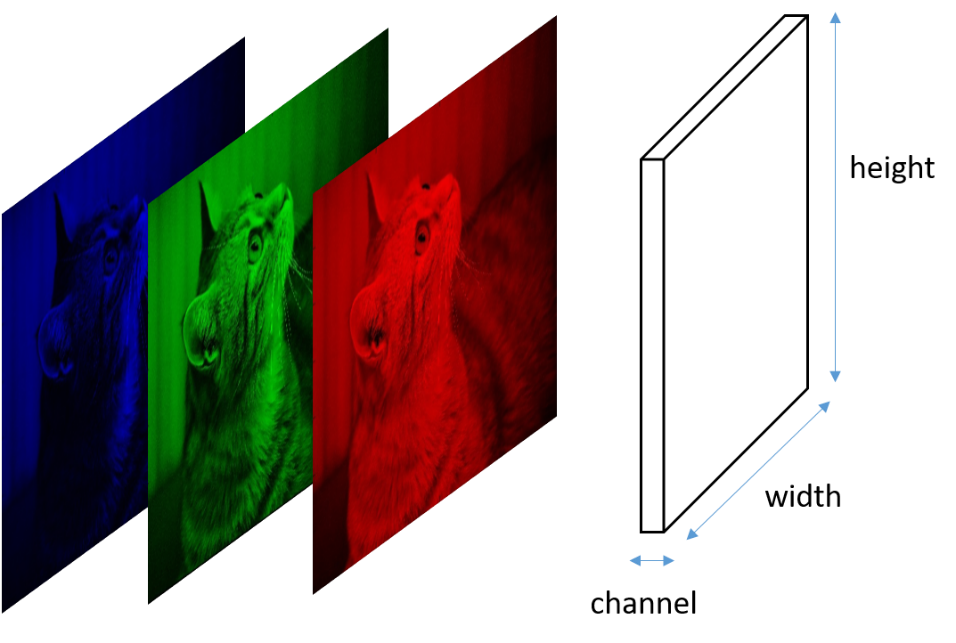
Grey\_image.shape => (height, width)

RGB represents colors as 8-bit integers from 0-254.

“Note that openCV version 3 reads the image color in the order B, G, R. However, matplotlib deals with the image color in the corder R, G, B.”

We can convert 3-dimensional colored images to 2-dimensional gray scale images using:

Gray\_image = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

Understanding the meaning of “channel” is important in deep learning: 

The above is an image array (Height, width, channel), where channel corresponds to the RGB (or in the case of OpenCV BGR) color representative integers (0 = R, 1= G, 2=B).

## Some concepts to be aware of:

“In image processing, a **Gaussian blur** (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss). It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales—see scale space representation and scale space implementation.”  
 -- Wikipedia

**“Thresholding** is a process of converting a grayscale input image to a bi-level image by using an optimal threshold. The purpose of thresholding is to extract those pixels from some image which represent an object (either text or other line image data such as graphs, maps). Though the information is binary the pixels represent a range of intensities. Thus the objective of binarization is to mark pixels that belong to true foreground regions with a single intensity and background regions with different intensities.”   
--ancient-asia-journal

## Moving on to motion detection using Background subtraction (as our means to detect change):

Essentially, we’ll need to have a camera taking a continuous stream of data and try to detect a change occurring by calculating the difference between the average of previous frames and the current frame and seeing if the difference is above a desired threshold.

If so, then a change has occurred, so we’ll start tracking the frames of motion (start recording the video we plan to save).

Once the motion stops, the average difference between the previous frames and the current frame should go below the threshold: at this point we can save all the motion frames to a video file and reset the motion counter (stop recording the video and save the segment of the stream we’ve been recording).

Steps:

1. Initialize camera,
2. Initialize a variable, rawCapture, to capture the raw data from the camera and represent it as a frame.
3. Initialize variables to keep track of the average, last frame’s timestamp, and a counter to track how many frames of motion have occurred since the change was detected.
4. Start capturing a continuous stream of video, looping through each frame in a for loop.
5. For each frame:
   1. Convert the frame of raw data to an array
   2. Get the current timestamp
   3. Preprocess the frame to make it easier to work with:
      1. Resize it to have width of 500 pixels
      2. Convert it to gray scale (2-d image processing is more efficient)
      3. Apply a Gaussian Blur\* to remove high frequency noise.
   4. Accumulate the weighted average\*\* between the current frame and the previous frames (this will represent our *background\_model*)
   5. Compute the delta image (delta = |background\_model – current\_frame|)
   6. Threshold\*\*\* the delta image to find regions in the image that pass the thresholding test and assign this to another threshold variable to hold this threshed frame.
   7. Dilate the threshold variable to fill in any holes in the frame.
   8. Find the contours (the regions of the image that mark the changes in the threshed frame).
   9. Loop through the contours and draw a box (on the original frame) to point out the detected changes
   10. Draw the timestamp onto the original frame
   11. If motion was detected for this frame (i.e. we found a change) then start recording the stream.
   12. If no motion was detected, which will only be noticed a few frames after the actual motion has stopped, then save the recorded segment of the video stream.

\*. Without getting into the math, a Gaussian blur reduces the image detail and blurs the image. This reduces the noise (outlier values that might skew the calculations). OpenCV has a function for this called using cv2.GaussianBlur()

\*\*. The weighted average will account for subtle changes that take place over the course of the day (like lighting changes), in other words, the background model represented in the code will dynamically adjust to match gradual changes to the background over the course of the day. Use cv2.accumulateWeighted()

\*\*\*. We must define an image threshold, adjust this value to determine how sensitive the algorithm should be to changes.

# Results:

Here are the steps we probably want to take, at least initially, simple change detection:

(literally, just the last sub-section of the Findings section)

1. Initialize camera,
2. Initialize a variable, rawCapture, to capture the raw data from the camera and represent it as a frame.
3. Initialize variables to keep track of the average, last frame’s timestamp, and a counter to track how many frames of motion have occurred since the change was detected.
4. Start capturing a continuous stream of video, looping through each frame in a for loop.
5. For each frame:
   1. Convert the frame of raw data to an array
   2. Get the current timestamp
   3. Preprocess the frame to make it easier to work with:
      1. Resize it to have width of 500 pixels
      2. Convert it to gray scale (2-d image processing is more efficient)
      3. Apply a Gaussian Blur\* to remove high frequency noise.
   4. Accumulate the weighted average\*\* between the current frame and the previous frames (this will represent our *background\_model*)
   5. Compute the delta image (delta = |background\_model – current\_frame|)
   6. Threshold\*\*\* the delta image to find regions in the image that pass the thresholding test and assign this to another threshold variable to hold this threshed frame.
   7. Dilate the threshold variable to fill in any holes in the frame.
   8. Find the contours (the regions of the image that mark the changes in the threshed frame).
   9. Loop through the contours and draw a box (on the original frame) to point out the detected changes
   10. Draw the timestamp onto the original frame
   11. If motion was detected for this frame (i.e. we found a change) then start recording the stream.
   12. If no motion was detected, which will only be noticed a few frames after the actual motion has stopped, then save the recorded segment of the video stream.

\*. Without getting into the math, a Gaussian blur reduces the image detail and blurs the image. This reduces the noise (outlier values that might skew the calculations). OpenCV has a function for this called using cv2.GaussianBlur()

\*\*. The weighted average will account for subtle changes that take place over the course of the day (like lighting changes), in other words, the background model represented in the code will dynamically adjust to match gradual changes to the background over the course of the day. Use cv2.accumulateWeighted()

\*\*\*. We must define an image threshold, adjust this value to determine how sensitive the algorithm should be to changes.