Motion Detection Arduino

***Author**: Luka Zoric*
***Student Number**: s5027193*
***E-mail**: luka.zoric@griffith.edu.au*

Abstract

Branch of Artificial Intelligence that deals with creating algorithms and systems that mimic human's ability to make sense of its surroundings is called Computer Vision.
It is usually used for face recognition, identification, verification, emotion analysis and crowd analysis. Additionally, it is great for Optical Character Recognition, Object Recognition, 3-D Printing and Image capture and also has many interesting applications.
This paper describes a simple system that explores part of computer vision called motion detection.

Introduction

Problem discussed will be related to motion detection. Motion detection is used in many ways and its applications are very useful. This system discussed in this paper will explore motion detection in a modest way. The system will receive live camera feed as an input and every time movement is detected, output signal will be created. This output is transferred to Arduino and is presented visually. This technology is simple but very useful. It can be used for intruder alarms, automatic ticket gates, automatic doors, hand dryers, self-driving cars and many more.

Previous Work

Motion detection is something people have been working on for a long time as its fairly simple and straight forward. Just find something different in the frame, decide if it's different enough to be classified as the object, and send signal accordingly. When doing my research, it was easy enough to find a lot of information around motion detection as most of examples follow the same procedure. This includes setting the initial frame, getting the next frame and calculating the intensity difference between them. Then there were numerous methods for finding if difference classifies object presence. For motion detection, decent amount of work has been done but there was a lack of examples of movement detection and this is where my challenge in this assignment was. Only thing that was obvious was the method of measuring if change in frames was changing. I had to compute difference between previous frames and next frames incrementally to detect if object was moving.

Technical Approach

In overview, python script will be running and sending the information to the Arduino, which will light up LEDs according to the output from the script.

Firstly, Python script will be discussed. Modules used are 'cv2' and 'serial'. Cv2 module is used for functions used for filtering video data. Serial module was used to connect Arduino to python. The system can be said to have two functionalities – motion detection and object presence detection.

Data from video camera was imported using VideoCapture() function. Arduino was linked to the python script using Serial() function. Initial variables that were set were static_back that was declared as None and was used to detect object presence, motion and presence flags (no_motion_count, in_frame) were declared as false (0).

From this point on, 'while' loop is created. In the first iteration, static_back variable is set to the first frame of the video capture. For every next iteration of the loop, frame from the video capture was recorded continuously. Frame variable is an array of 3D vectors that represent RGB colors of the pixel. As difference between the new frame and initial frame had to be computed, it is easier and more efficient to perform this operation using intensity of the frame. For this reason, gray frame was introduced (black and white version for the current frame). This photo was blurred using Gaussian Blur filter. Using this data, similar filtering is used for computing two previously mentioned functionalities.

First, most basic functionality is object presence detection. First step is to compare new frame to the initial frame. This is done using the absolute difference of the two frames. This have good enough result for object detection when combined with the threshold of the resulting image. This threshold

image clearly displayed new object in white pixels, and background was displayed in black pixels. To find those white pixels, cv2 function findCountours() has been used to detect white areas in the frame. This function also returned list of coordinates of the blobs created with white pixels. For each one of these blobs, rectangle coordinates were computed and stored. If area of the blobs (or number of white pixels) was above threshold value, rectangle with coordinates taken from the blobs has been drawn onto the original frame and printed to the screen. Additionally, motion flag (in_frame) was updated to 1 and information has been passed to serial output.

Process of motion detection is exactly the same as previous functionality until after contour detection. Motion detection will work in a way where after some number of frames, it resets the background image used in absolute difference calculation (is object in the frame). If, after reset, number of white pixels (blobs) is under certain threshold, do not change anything. Otherwise, if there is movement, which will cause display of white pixels due to absolute difference between the current frame and newly set frame, no_motion_count variable responsible for recording number of frames with which movement is detected will start incrementing.

Experiments

Object presence detection

This project was simple and therefore experimentation couldn't have been complicated. Firstly, let's take a look at object presence detection in action.

Initial frame

Let's assume this was the initial frame. Figure below displays an example.

Figure 1: random location – kitchen table

As the object walks into the frame, detection starts to work. If we look at the absolute difference frame before the object entered the screen and after it entered the screen, we can obviously see how intensity of the pixels where our object is much greater, and object is clear enough. See below how objects are easily recognizable in figures 2 and 3:

Figure 2: intensity difference frame without objects present

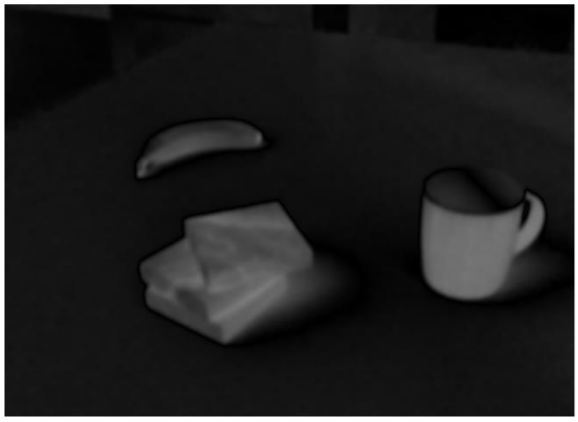

Figure 3: intensity difference frame with objects present

If we perform threshold on absolute difference photo, object presence is detected with white pixels in a very clear manner. See figure 4 below:

Figure 4: threshold frame objects present

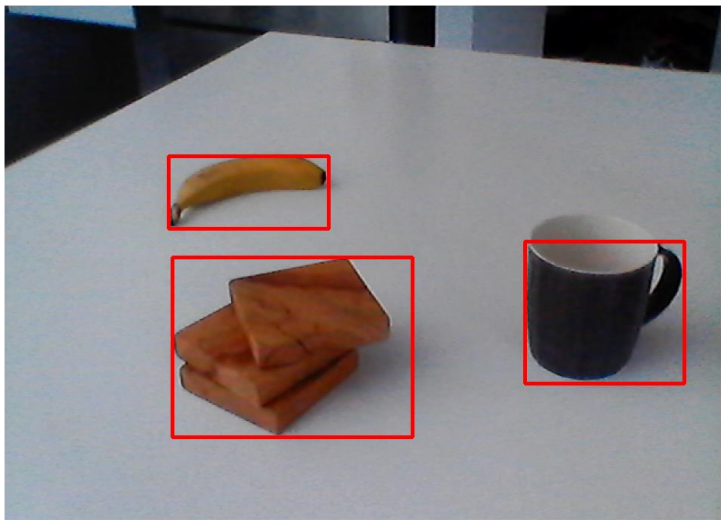

Figure 5: detected objects

As previously mentioned, these white blobs were detected, coordinates obtained and labeled accordingly. I used red rectangles as markings. Detection is displayed on the figure 5.

Motion detection

With motion detection, it was easy enough to move around number which delay motion detection flag. This particular problem could have been solved with time, however, time limit in frames was much easier to be programmed and worked well enough. Only problem was that if computer testing was performed on performed slowly. In case of this happening, detection was never noted, or object could move slowly and never get caught. To fix this only requirement was to reduce number of required delay frames (no_motion_count) and then compute the difference and decide on motion flag.

Conclusion

This assignment was created fairly simply within the time frame proposed in the initial proposal handed in prior to this report. The initial idea was to explore possibilities for motion detection and object presence. Most important challenges were overcoming, software was built, and it is clear that its use is versatile. Signal can not only be sent to Arduino to be displayed as a flashing LED, but also as a speaker for security reasons, signal can be sent to start any kind of doors opening when object presence is detected in front of it. In conclusion, even though not ground-breaking technology, motion detection and object presence is proven to be useful in many different ways.

References

N/A