Journal of UAB ECTC
Volume 17, 2018
Department of Mechanical Engineering
The University of Alabama, Birmingham
Birmingham, Alabama USA

AUTOMATED IMAGE ACQUISITION FOR VEHICLE INSPECTION

Hannah Evans, Kevin McFall, Ph.D.,
Mir Atiqullah, Ph.D.
Kennesaw State University
Marietta, Georgia, USA

ABSTRACT

This paper details the design and implementation of an automated image acquisition system. Its purpose is to autonomously detect and log images of vehicles to inspect for damage and to pinpoint when damage was dealt. This task is accomplished through interfacing pyroelectric infrared sensors and four USB cameras through Python scripts running on a Raspberry Pi on startup. The images are stored on a permanently mounted USB flash drive in a timestamped and chronologically ordered file system. Adjustable parameters include frame rate (time delay), image resolution, brightness, contrast, saturation, sensor retriggering, and sensor timeout.

KEY WORDS: *Vehicle inspection, damage detection, PIR sensors, GPIO, Automated image acquisition and archiving*

INTRODUCTION

Jack Cooper Logistics specializes in vehicle transportation and inspection. During transportation, minor vehicle damages frequently occur and must be accurately documented and cataloged, lest the burden of compensation be incorrectly placed. Currently, the company manually inspects vehicles and seeks to automate the process and keep a visual record of the vehicles in transit.

Current methods of damage detection as applied to structural health monitoring (SHM) rely on dynamic strain sensor readings' frequency response to determine the structure's integrity [1]. Similar approaches in vehicle damage detection involve embedding sensors to measure structure-borne sound, and filtering false positives caused by normal vehicle operation or changes in vehicle state through either software or hardware thresholding; these methods focus on detecting damage during normal vehicle operation as it occurs [2-5].

Other related technologies focus on retroactively assessing damages. One such system compares an input aircraft damage description to past cases to determine a resulting course of action [6]. Another system uses a still image or video input to assess known damage for the purpose of processing an insurance claim [7]. A similar system uses a support vector machine (SVM) trained on extracted image features to classify the type and magnitude of vehicle damage [8]. These methods rely on human-input data, whereas this project requires the data to be collected autonomously.

A current automated vehicle image acquisition system enlists image processing algorithms to identify a vehicle's license plate and other identifying information; this system photographs the vehicle from multiple angles, and compares the images to prior data to detect damages [9]. This system relies on specific positioning of the vehicle relative to the cameras, which is difficult to accomplish in transit, as Jack Cooper Logistics wishes to log vehicle data.

To best fit the needs of Jack Cooper Logistics, a light-proof scaffolding tunnel, as shown in Figure 1, was designed and constructed to accommodate a vehicle of maximum height 7.5 ft and maximum width 9.0 ft as it is photographed. The reason for a light-proof structure is to allow for the use of structured light to increase damage visibility. This structure houses cameras, sensors, and other necessary electronic hardware for image acquisition. These cameras are positioned within a 5.0 ft section in the center of the structure to photograph different segments of the vehicle as it passes through. The rest of this paper focuses on electronic hardware and image acquisition methods rather than mechanical design.

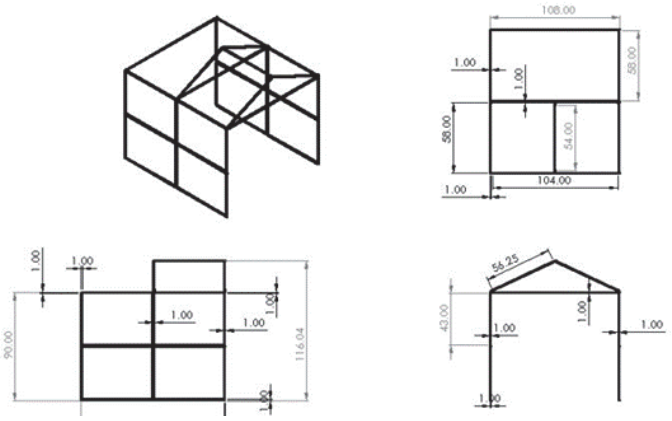

Figure 1: 2D CAD Drawing of Scan Gate Structure

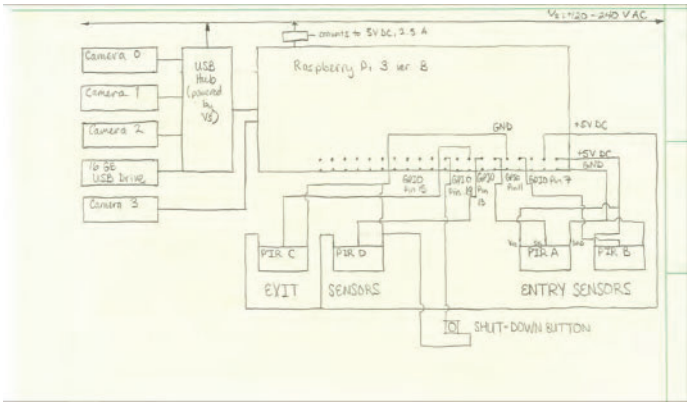

Figure 2: Sketch of GPIO Block Diagram

ELECTRONIC SYSTEM DESIGN

The central automation task is designing a system to sense the presence of a vehicle and taking photographs as it passes through. These photographs should be of a high quality and, in total, completely photograph the vehicle. This system is represented by the block diagram in Figure 2.

To easily facilitate GPIO sensor communication with code to control cameras, a Raspberry Pi 3 was chosen as the processor. This selection limits the sensors able to be used for detection; any sensors must be digital sensors with a high signal of 3.3 V; this effect may also be achieved through using a voltage level shifter to limit the output of a 5 V digital sensor.

Complying with these constraints, pyroelectric infrared (PIR) sensors were chosen to accomplish vehicle detection. Two sensors are placed at the entrance, and two sensors are placed at the exit. Initially, the multiple sensors were intended to decrease missed detections. However, in implementation, the issue was oversensitivity rather than undersensitivity. Therefore, the sensor inputs were subjected to an "and" operation in the code, as follows:

```
def entrySensor():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(11,GPIO.IN)
    GPIO.setup(7,GPIO.IN)
    sense = GPIO.input (11) & GPIO.input(7)
    return sense
def exitSensor():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(15,GPIO.IN)
    GPIO.setup(13,GPIO.IN)
    motion = GPIO.input(15) & GPIO.input(13)
    return motion
```

The sensors themselves were enclosed in open cylinders to limit the detection radius and limit false detections. This setup is demonstrated in Figure 3.

To ensure completion of vehicle image acquisition, PIR sensor retriggering must be considered. The sensors can be set to retrigger, continuing to output a high signal as motion continues to be detected, or to not retrigger, returning to low signal soon after the initial detection, regardless of additional motion. In the typical case, in which a large vehicle is passing through the 5.0 ft scan area, the exit sensors should be set to retriggering mode in order to completely capture the vehicle. This parameter, however is easily adjustable on the sensor modules themselves.

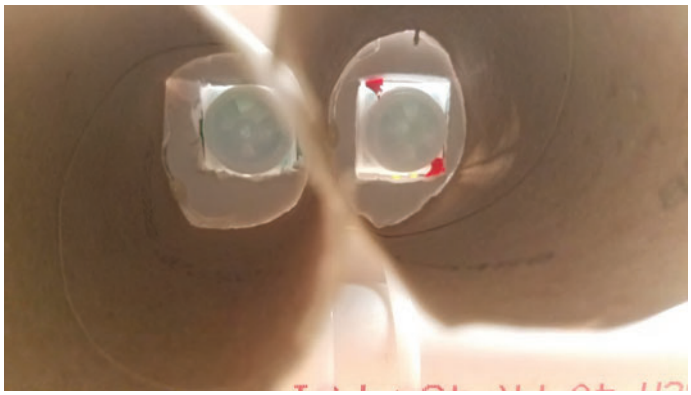

Figure 3: PIR Sensors Lenses with Physically Limited Sensing Radius

Another factor to consider when contemplating retriggering settings is sensor timeout. If set to retrigger, the time required for the PIR sensor to send a low voltage signal after motion has ceased can be modeled by the following equation[1]:

$$T_x = 24576 \times (R_{10} + R_{TIME}) \times C_6$$

C_6 equals 0.01 µF and R_{10} equals 10 kΩ added to R_{TIME}, the resistance determined by the "time" potentiometer, as shown in Figure 4. To compute the minimum timeout, the time potentiometer must be set to its minimum of $TIME = 0\ \Omega$, making the total resistance 10 kΩ. The minimum timeout, then, equals 2.5 seconds. Assuming pictures are taken at a rate of 4 frames per second, this minimum timeout accounts for around 10 extraneous images. However, acquiring extraneous images, in most cases, is preferable to prematurely ending the process, making retriggering more frequently the better option.

[1] IC datasheet and schematic. On page 3, the equation is provided for the PIR sensor from the manufacturer : https://cdn-learn.adafruit.com/assets/assets/000/010/133/original/BISS0001.pdf

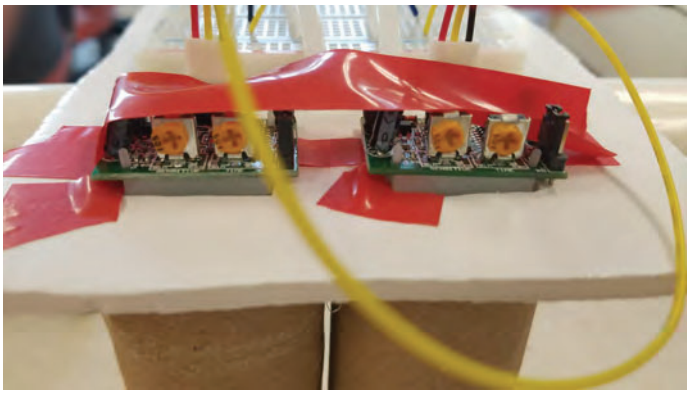

Figure 4: PIR Sensor Potentiometers: Sensitivity and Time

This system uses four USB cameras to completely photograph vehicles: one camera on the left, right, front, and back, angled to capture the front, back, and roof of the vehicle.

The selected cameras' maximum image resolution is 1920×1080 pixels. However, taking higher resolution images often leads to a timeout error from the Raspberry Pi. Therefore, the resolution was decreased to 1280×720 pixels. Additionally, field of view (FOV) plays a role in camera selection. A wider FOV captures more of the vehicle per image. The cameras used in this prototype have a 180° FOV.

The cameras were interfaced in the code using the OpenCV3 library for Python, as follows:

```
def cheese(cap0,cap1,cap2,cap3,path,itr):
    ret0,frame0 = cap0.read()
    ret1,frame1 = cap1.read()
    ret2,frame2 = cap2.read()
    ret3,frame3 = cap3.read()
    if itr > -1:
        itr = str(itr)
        os.chdir(path)
        cv2.imwrite(itr+ '-cam0.jpg',frame0)
        cv2.imwrite(itr+ '-cam1.jpg',frame1)
        cv2.imwrite(itr+ '-cam2.jpg',frame2)
        cv2.imwrite(itr+ '-cam3.jpg',frame3)
        os.chdir('/home/pi')
```

The central tasks were initializing video streams and capturing images with a reasonable time delay. A 0.25 s delay was appropriate throughout testing.

Additional OpenCV tasks involved adjusting and calibrating camera properties. The cameras used allow for adjusting resolution, brightness, contrast, and saturation. These parameters are adjustable in the code and can be fine-tuned based on the system's environment. In practical implementation, however, the parameter changes do not take effect for the first set of images. Therefore, during a calibration period on system boot, a set of pictures is taken so that the delay in parameter updates does not affect any of the vehicle images.

The system stores all images taken on a USB flash drive. Therefore, it is necessary to set a permanent mount point for the flash drive in use[2]. This method requires specifying the device UUID, so if multiple flash drives are to be used with the system, each device must be specified. However, once the device is specified, the Raspberry Pi can automatically recognize it. Figure 5 demonstrates the USB ports' configuration.

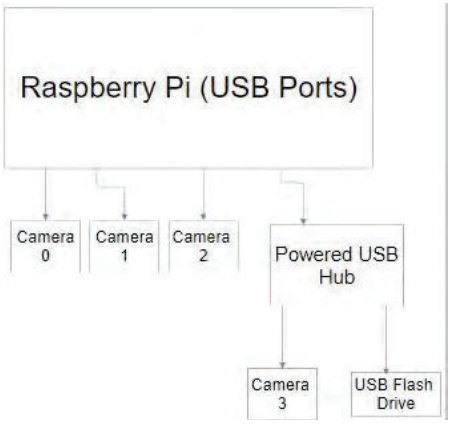

Figure 5: USB Port Block Diagram

An organized folder system, as coded below, not only separates images from separate sessions, but organizes images based on time acquired.

```
today = datetime.datetime.now().strftime('%Y-%m-%d')
path = 'media/usb/' + today
if not os.path.isdir(path):
    os.makedirs(path)

    while not entry:
            entry = entrySensor()
    entryTime=datetime.datetime.now().strftime
                    ('%Hhr-%Mmin-%Ssec')
       car = path + '/' +entryTime
       os.makedirs(car)
```

This system consists of folders stamped with the current date that are generated on boot which contain subfolders stamped with the time of each vehicle's entry that are generated as the entry PIR sensors are triggered. These timestamps, however, will only be accurate given an internet connection. Without an internet connection, the Raspberry Pi will count time from the last time recorded on it. Therefore, regardless of timestamp accuracy, the images will appear in chronological order.

EXPERIMENTAL RESULTS

The prototype was developed and set up indoors, so these spatial constrains prevented a full-scale test. The upper level of the frame was set up and suspended about 3.0 ft off the floor. To test the system, a line of chairs was rolled through the scan area. The time delay between images was 0.25 seconds with an approximate frame rate of 4 frames per second. In an average

[2] https://www.raspberrypi-spy.co.uk/2014/05/how-to-mount-a-usb-flash-disk-on-the-raspberry-pi/

test run, the "vehicle" traversed the 5.0 foot scan area in 5 frames, yielding a typical test velocity of 4 ft/s or 2.7 mph.

The test conditions did not include a light-poof scan area featuring structured light. Therefore, image clarity is demonstrated by placing small pieces of tape that matched the chairs, simulating minor vehicle damage. As shown in Figure 6, they are visible, even at the decreased resolution.

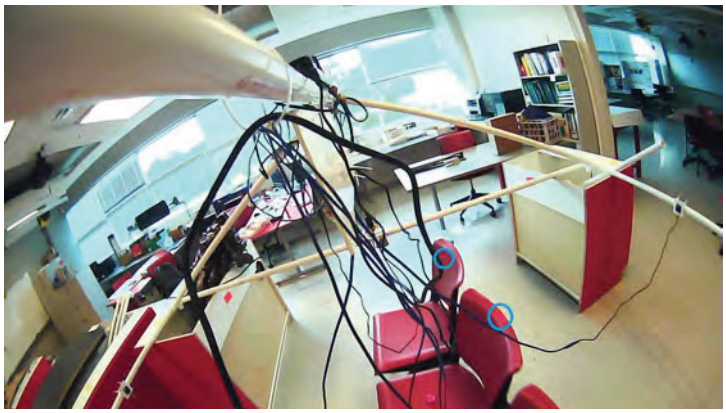

Figure 6: Test Run Images

At 2.7 mph with a 4 FPS frame rate, the system captured the passing object in its entirety. When set to non-retriggering mode, this process took place over 7 frames (1.75 s). As previously discussed, a larger vehicle would require the exit sensors to retrigger, and the process would take place over a longer period. The initial system design assumes a constant vehicle speed of 5 mph, but the system was not tested to that end. The frame rate is adjustable in the code and can be increased to facilitate a faster vehicle speed than was tested.

CONCLUSION

This system prototype proves the concept of a sensor-based photograph logging system. This system can be applied to vehicle inspection given further parameter tuning and refinements for user friendliness. Extensions of this work could include changing the system output to be a single composite image of the passing vehicle, and implementing structured light arrays to aid in detection.

ACKNOWLEDGEMENTS

This project was made possible and supported by: Andrea Amico, president of Jack Cooper Logistics for providing the project and funding, Dr. Kevin McFall for advising the electronic system aspect of the project, Dr. Mir Atiqullah for advising the project as a whole, and the KSU mechanical engineering senior design team (Duronte Carter, Jacob Kaveney, and Matthew Kliewer) for designing the mechanical structure.

REFERENCES

[1] Tondreau, G., & Deraemaeker, A. "Local modal filters for automated data-based damage localization using ambient vibrations", Mechanical Systems and Signal Processing, Volume 39, Issues 1–2, 2013, pp. 162-180, doi:10.1016/j.ymssp.2013.03.020.

[2] Baumgärtel, Hauke, Kneifel, Andre, Gontscharov, Sergei, & Krieger, Karl-Ludwig, "Investigations and Comparison of Noise Signals to Useful Signals for the Detection of Dents in Vehicle Bodies by Sound Emission Analysis", Procedia Technology, vol. 15, 2014, pp. 716-725. doi:10.1016/j.protcy.2014.09.044.

[3] Gontscharov, Sergei, et al. "Algorithm Development for Minor Damage Identification in Vehicle Bodies Using Adaptive Sensor Data Processing." Procedia Technology, vol. 15, no. 2nd International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering, 01 Jan. 2014, pp. 586-594. doi:10.1016/j.protcy.2014.09.019.
[4] Flextronics AP, "Vehicle Damage Detection and Indication", patent number 20140309805, 2014.
[5] Soles, Alexander M., Walsh, Matthew R., & Scott, Eric B. "Vehicle damage detection system", patent number 8788220, 2014
[6] Wilke, D. A., & McCarthy, D. A., "Automated damage assessment, report, and disposition", patent number 7546219, 2009.
[7] Brandmaier, Jennifer A., et al. "Feedback loop in mobile damage assessment and claims processing", patent number 8510196, 2013.
[8] Harshani, W.A. Rukshala, & Vidanage, Kaneeka. "Image processing based severity and cost prediction of damages in the vehicle body: A computational intelligence approach," Proc. 2017 National Information Technology Conference (NITC), Colombo, 2017, pp. 18-21.
[9] Webb, S. G., Johnson, I. G., & Watson, J. G., "Systems and methods for automated vehicle image acquisition, analysis, and reporting", patent number 7889931, 2011.