exeley INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS

Article | DOI: 10.21307/ijssis-2020-007

Issue 1 | Vol. 13 (2020)

A novel approach to label road defects in video data: semi-automated video analysis

Jakob Thumm[1,*], Johannes Masino[1], Martin Knoche[2], Frank Gauterin[1] and Markus Reischl[1]

[1]Karlsruhe Institute of Technology, Karlsruhe, Germany.

[2]CEO of N3T in Whangarei, New Zealand.

*E-mail: jakob.thumm@student.kit.edu

The paper was edited by Djilali Kourtiche.

Received for publication December 5, 2019.

Abstract

Road defects like potholes have a major impact on road safety and comfort. Detecting these defects manually is a highly time consuming and expensive task. Previous approaches to detect road events automatically using acceleration sensors and gyro meters showed good results. However, these results could be significantly improved with additional usage of image analysis. A large, labeled image data set is required for training and validation. This paper presents a method to automate parts of the labeling task. The method is based on a simple two step approach: at first, an unsupervised algorithm detects possible events based on the acceleration data and filters those video sequences with defects. Second, a human operator decides based on the short video sequences if the event was due to an existing road defect and labels the corresponding area in an image.

Keywords

Pothole detection, Road defect detection, Image labeling, Acceleration data processing, Sensor fusion, Intelligent sensor systems.

Road defects cause increased vehicle repairs and maintenance[1], crashes, death[2], and cost a substantial amount of money globally. Road maintenance in OECD countries averages at about 1.5% of GDP (OCED, 2019). Rather than being the ambulance at the bottom of the cliff and reacting to existing road defects, it is suggested to perform regular road inspections and shift from a reactive to a preventative road maintenance regime.

In this paper we suggest a pragmatic, affordable method to detect and map road defects in a semi-automated fashion. Karlsruhe Institute of Technology (KIT) has developed an elegant method (Masino et al., 2017) to create training data sets of images for road damage classification[3]. The efficiency gains in terms of accuracy and time savings of event labeling have operational and commercial benefits.

To detect potholes on rural roads in Northland[4] (NZ) at present, a highly trained road safety engineer travels the 5,000 km long road network every month at an average traveling speed of 40 km/h or less. It can take a team of two road inspectors up to three and a half days to mark up all road defects on 12 km of a rural road like Otaika Valley Road (OVR)[5]. With the suggested process

[1]In New Zealand: up to $7,000 per truck per year (company information from Fonterra, NZ biggest company 2019).

[2]India: 3,597 people had been killed and 25,000 injured in 2017 owing to crashes caused by potholes alone (*The Guardian* 24/07/2018).

[3]The analysis has been largely done in Python. The code and sample data can be found: https://zenodo.org/record/3384989.

[4]Northland is the biggest geographical region of New Zealand covering about 5% of the overall area. It has a population of about 180,000 inhabitants. It has 5,000 km of roads, 3,000 km are rural, gravel roads.

[5]Otaika Valley Road (OVR) is a 12 km rural road in Northland, New Zealand. It carries over 800 truck movements per day. For more detail: see http://n3t.kiwi/motivation.

a vehicle can take an inertial measurement unit (IMU) and camera technology as payload and travel at 80 km/h and achieve similar inspection results.

Our IMU and video data show for a newly sealed, heavily loaded road like OVR, that a pothole can form within a few months. Therefore, detection and remedial work resulting in preventative road maintenance constitutes a huge commercial opportunity for cost savings and road safety improvements.

Keeping roads safe is not only good for the people living and working in the area, but also has substantial economic and financial benefits. If road unevenness can be detected early and remedied, these causes of crashes can be reduced by preventative road maintenance. The New Zealand business case is seen as a proxy for similar situations in other countries around the world.

Literature review

There exists an extensive amount of studies, which present methods and systems to detect road hazards automatically. One of the most cited and famous systems is 'Pothole Patrol', published by MIT researchers in 2008 (Eriksson et al., 2008). The authors utilized an inertial sensor placed in the vehicle to detect potholes with a filter-based data processing pipeline. They managed to cover 2,492 distinct kilometers during their 10 days driving period. Overall, the detector misidentified road features as having potholes in less than 0.2% of the time, while 90% of reported detections contain actual road anomalies in need to repair. Most of studies employing inertial sensors have mainly tried to detect potholes or the overall road roughness. There are several approaches in recent literature to use camera systems to detect further types of road defects besides potholes, such as cracks or patches (Radopoulou and Brilakis, 2016). Hereby, supervised learning is the most prominent method to process the images from camera systems since it needs less training data compared to unsupervised learning. For example, Wu et al. (2014), Xu et al. (2008), and Zhou et al. (2006) applied artificial neural network algorithms to detect cracks on the road, which are trained with annotated images of road hazards. Radopoulou and Brilakis (2016) used semantic texton forests (STF), supervised learning algorithms, to segment road hazards based on their texture, layout, and context. Support vector machines (SVM) represent another supervised classification method for road condition monitoring, applied by Lin and Liu (2010).

Maeda et al. (2018) were the first to introduce a comprehensive labeled image data set for road damage recognition. The damage types are separated into the four main classes longitudinal cracks, lateral cracks, alligator cracks, and severe damages like potholes or bumps. Additionally, the classes 'crosswalk blur' and 'white line blur' are introduced to prevent confusion between these visual effects and actual road damages. In overall, 163,664 captured images they labeled 15,435 events but only 409 (2.65%) of these events were severe damages (e.g. potholes). This low number is caused by the low likelihood of occurrence of potholes in well-developed countries. Their proposed machine vision algorithms reached an accuracy of 79% for longitudinal cracks, 93% for lateral cracks, 84% for alligator cracks, and 95% for potholes.

Angulo et al. (2019) added 1,200 potholes and other road defects from Mexico and Italy to the Japanese data set in order to reach a more balanced number of events per class. They gathered 180,345 road images containing 45,435 instances of surface damages. Their accuracy of pothole detection increased to 98% using the new image data.

Results of recent studies show that cameras in combination with supervised learning methods can precisely detect and mark defects within images. However, one problem arises throughout the literature of camera systems and road condition monitoring. Composing a training data set is typically a highly manual, time consuming, and costly process. Researchers must go through a large size of acquired video material, select images with road defects, and annotate these events. In addition to this, manually labeled data sets will only include a small proportion of severe defects like potholes, sunken manholes, and bumps due to their low likelihood of occurrence. Creating a data set for combining acceleration data and video data is even harder because the operator must verify that the road defect was hit by the tyres.

To close this research gap, we propose a method to select images with road defects from acquired video material automatically. Our approach is based on an IMU sensor in synchronised operation with a camera. First, we show a method to fuse inertial sensor and video data in order to filter them automatically and detect road defects based on the inertial sensor data. Afterwards an algorithm selects the corresponding images with the identified road defects. In the last step, the user verifies and labels the road defects in the suggested images. This semi-automated process constitutes a substantial time saving over existing methods.

Methodology

The goal of this publication is to help users create a training data set of images for a road damage

classification. The following section shows the required methodology. Figure 1 gives an overview of the main process steps. First, we import and pre-process the raw IMU sensor data. Afterwards, our program compresses the data and calculates summarizing features. The road roughness in each street segment must be assigned to a score from 0 to 1. To generate this score, we construct a comparison table from all data. After building this table, we can detect heavy road events with a score threshold.

Data acquisition and signal processing

To gather a massive amount of unlabeled data, the University of Berkeley fitted a fleet of cars with smartphones mounted to the front windows. The data were gathered in urban areas of the east coast of the USA. It contains all types of weather and both day and night drives. The smartphones gathered acceleration, gyroscope, GPS, and video data of each drive. These data are separated into 20 to 30 sec segments to lower individual file size and to protect the privacy of the drivers.

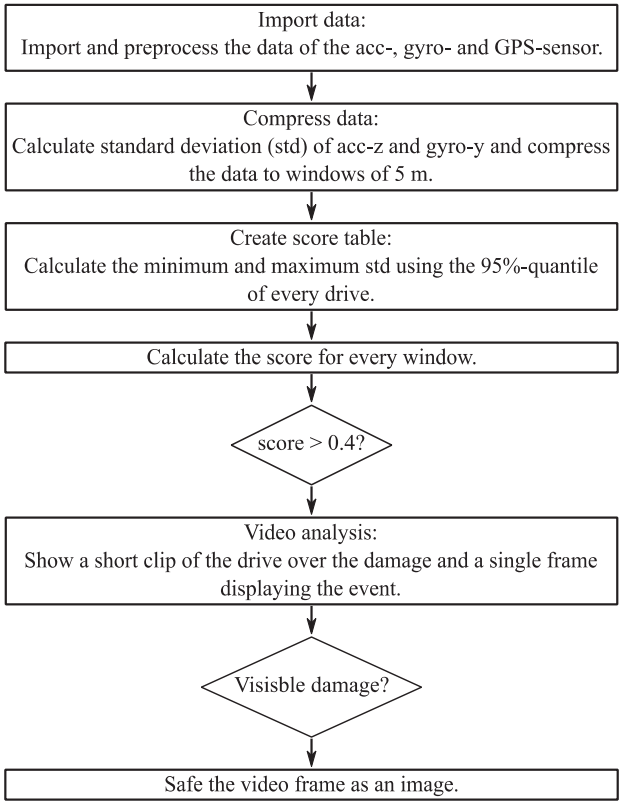

Figure 1: Overview of the program workflow.

We import the sensor data and pre-process it in order to minimize measurement inaccuracies.

First, we smoothen the acceleration data using a Savitzky–Golay filter[6]. The SG filter reduces the noise of the acceleration data and minimizes the least-square error (Li et al., 2013).

Then we reduce the noise of the raw gyroscope data with a one-dimensional median filter. Typical measurement errors like spikes were thereby removed.

The program needs to operate with the information from all three sensors. To merge the data, we interpolate the gyroscope and location data to fit the timestamp vector of the accelerometer. This is done using one-dimensional linear interpolation[7].

Ward and Iagnemma (2009) showed that the response of the vehicle to the excitation of the road depends on the driving speed. The conversion of a time series into space domain reduces this effect. To convert the data into space domain, we initially calculated the traveled distance for each time step using the following formula:

$$\Delta s = v \cdot \Delta t.$$

In the second step, we resample the data by distance using the same linear interpolation function as before. It would also be possible to calculate the space domain using the GPS coordinates, but this would lead to higher deviations due to the inaccuracy of the sensor.

Data compression

The next process reduces the amount of data and extracts important information from the oscillation profiles. Our program compresses the data into so called windows. The data compression algorithm is described in Figure 2. The program combines five meters of data (500 data points) to one new data point. Each new point gets assigned the location and speed of the segment center point. Furthermore, the routine calculates the standard deviation of the acceleration in *z*-direction (std-acc-z) and the standard deviation of the angular velocity around the *y*-axis (std-gyro-y) for every window. To guarantee that every event is captured with this method, the windows should overlap.

Score calculation

We want to estimate the road roughness by comparing the standard deviation of the *z*-acceleration

[6]SG Savitzky–Golay filter taken from the Python scikit library.
[7]1D interpolation taken from the Python numpy library.

Algorithm 1: Compress with windows

Data:

- `data`: All IMU measurement data
- `window_length`: The number of measurement points in a window
- `ov`: The overlapping factor

Result: Compressed feature data

```
overlap ← round(ov/100 · window_length);
middle_distance ← round(window_length/2);
counter ← 0;
i ← 1;
while (i + 2 · middle_distance) < size(data)) do
    start_points(counter) ← i;
    middle_points(counter) ← i + middle_distance;
    end_points(counter) ← i + window_length;
    i ← i + window_length − overlap;
    counter ← counter + 1;
end
for i = 0 : size(start_points) do
    sp ← start_points(i);
    mp ← middle_points(i);
    ep ← end_points(i);
    windows(i).timestamp ← data(mp).timestamp;
    windows(i).longitude ← data(mp).longitude;
    windows(i).latitude ← data(mp).latitude;
    windows(i).velocity ← data(mp).velocity;
    windows(i).std_acc_z ← std(data(sp:ep).acc_z);
    windows(i).std_gyro_y ← std(data(sp:ep).gyro_y);
end
return windows
```

Figure 2: Algorithm to compress the data and extract information about oscillation.

with the minimum and maximum values of all data. The algorithm in Figure 3 describes how we calculate the scoring table, including the min/max values.

Algorithm 2: Calculate score table

Data:

- `score_file_path`: The path to the score file (To be created or imported)
- `all_data_list`: A list that contains all IMU measurements compressed to windows.

Result: A table with the min and max values for different speed sections

```
if is file(score_file_path) then
    score_table ← load(score_file_path);
else
    # Create new score table;
    score_table.speeds ← 5, 10, 15, 20, 25, 100;
    score_table.min_acc_z ← infinity(1, 6);
    score_table.min_gyr_y ← infinity(1, 6);
    score_table.max_acc_z ← zeros(1, 6);
    score_table.max_gyr_y ← zeros(1, 6);
speed_ranges ← [0, score_table.speeds];
for each drive in window_data do
    speed_vec ← drive.speed;
    for i = 1 : size(speed_ranges) do
        ids ← find(speed_vecspeed_ranges(i-1) & speed_vec < speed_ranges(i));
        min_acc_z ← percentile(drive.std_acc_z(ids), 3);
        max_acc_z ← percentile(drive.std_acc_z(ids), 97);
        score_table.min_acc_z(i-1) ← min(min_acc_z, score_table.min_acc_z(i-1));
        score_table.max_acc_z(i-1) ← max(max_acc_z, score_table.max_acc_z(i-1));
        min_gyr_y ← percentile(drive.std_gyro_y(ids), 3);
        max_gyr_y ← percentile(drive.std_gyro_y(ids), 97);
        score_table.min_gyr_y(i-1) ← min(min_gyr_y, score_table.min_gyr_y(i-1));
        score_table.max_gyr_y(i-1) ← max(max_gyr_y, score_table.max_gyr_y(i-1));
save_file(score_table, score_file_path);
return score_table;
```

Figure 3: Algorithm to calculate the score comparison table.

The most important feature for the classification of road damages is the speed of the vehicle. The amount of vibration differs greatly with changing speeds.

To handle this dependency, we separate the data into sections of speed. The algorithm uses the entries from every drive to calculate the minimum and maximum values of the standard deviations. It is important that the entries are divided into their speed sections. Since there can be heavy outliers, the program sets the maximum entry to the 97% quantile of the standard deviation. The score table likewise includes the standard deviation of the gyroscope sensor data. The process is repeated for every ride and every section of speed.

The program is now able to calculate the specific score of a single window using the score table. We estimate the score by comparing the sensor data of a window with the minimum and maximum values of the corresponding speed section in the score table. The score is limited to lay in between 0 and 1. We determine the final score out of the mean of the acceleration and the gyroscope score. Figure 4 displays the required steps for the score estimation.

Implementation

We use the methods presented in Chapter Methodology to implement an automatic data processing routine. Our Python code (Thumm and Masino, 2019) can be downloaded under the following link: https://zenodo.org/record/3384989. There is also an example data set to test the program.

Algorithm 1: Get score for data

Data:

- `drive`: The data to calculate the score on
- `score_table`: The score table including min and max scores for every speed range

Result: The score for each data point

```
score_vec ← zeros(size(drive, 1), 2);
speed_ranges ← [0, score_table.speeds];
speed_vec ← drive.speed;
for i = 1 : size(speed_ranges) do
    ids ←
     find(speed_vec >= speed_ranges(i-1) & speed_vec < speed_ranges(i));
    acc_z ← drive.std_acc_z(ids);
    gyr_y ← drive.std_gyro_y(ids);
    s_min_a ← score_table.min_acc_z(i-1);
    s_max_a ← score_table.max_acc_z(i-1);
    s_min_g ← score_table.min_gyr_y(i-1);
    s_max_g ← score_table.max_gyr_y(i-1);
    score_a ← (acc_z−s_min_a)/(s_max_a−s_min_a);
    score_a(score_a > 1) ← 1;
    score_a(score_a < 0) ← 0;
    score_vec(ids,0) ← score_a;
    score_g ← (gyr_y−s_min_g)/(s_max_g−s_min_g);
    score_g(score_g>1) ← 1;
    score_g(score_g<0) ← 0;
    score_vec(ids,1) ← score_g;
final_score ← mean(score_vec, axis=1);
return final_score;
```

Figure 4: Algorithm to calculate the final score for each window.

The program automatically imports a folder of raw data, pre-processes it and extracts special events for the end user. The human operator now only needs to view a short video sequence of the drive over the event to determine whether it was a road defect and label the data accordingly.

Data structure

The sensor data are in a json file format and contain the following features:

- Acceleration in *x*, *y* and *z*-direction with a sampling rate of 50 Hz.
- Gyroscope: angular velocity around the *x*, *y* and *z* axis with a sampling rate of 50 Hz.
- GPS: longitude, latitude, and speed with a sampling rate of 1 Hz.

Each ride segment has one sensor data json file and a corresponding video file that matches its length. The video data have a sampling rate of 30 frames per second (fps).

Data import

The json file gets imported into Python and the data pre-processed as described in Chapter Data acquisition and signal processing. As parameters of the Savitzky–Golay filter, we choose a window length of 11 and a polynomial order of 5. The window length of the median filter is 5 m. The algorithm resamples the data into the space domain with a frequency of $100\,\mathrm{m}^{-1}$.

The data get compressed to windows as shown in Chapter 2.2. To guarantee good coverage of the street, we set the windows length to 5 m and the overlapping factor to 66%. The compressed data are saved with the corresponding video file names for later use.

Calculate road roughness score

The score table, presented in Chapter Data compression, is build using the speed ranges shown in Figure 5.

The created table helps to compare the sensor data with all previous data. Hence, we can assign a score to each window. Figure 6 displays the minimum and maximum values of the score table for an example data set. It is noticeable, that the *z*-acceleration value is high for medium speeds. The IMU vibration in the car increases with higher speeds. Typically, higher speed roads have fewer road defects and the vibration peaks are therefore lower. A similar pattern can be seen in the gyroscope data. The pitch *y*-gyroscope values at low speeds are higher than the corresponding vertical *z*-acceleration values. A possible explanation for this is the pitch movement a car makes when harsh cornering or braking. These movements directly lead to a higher *y*-angle velocity.

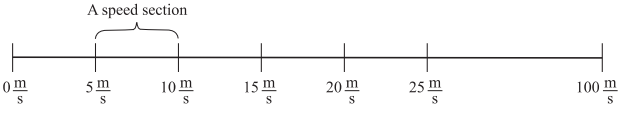

Figure 5: Arrangement of speed sections.

Video data analysis

After the data pre-processing, the video sequences can be analyzed. We use a threshold to distinguish the special events from normal driving. Every window with a score higher than a threshold *T* indicates a possible road defect.

To reduce the number of incorrect images presented to the operator, the program filters out every event with a speed lower than 3 m/s. If the vehicle velocity is lower than this threshold, the algorithm often detects harsh braking or acceleration as a road defect event.

Since the goal of the video analysis is to generate training data for an image classification, night drives should not be considered. This is simply done by measuring the mean grey level of the event image and filtering out all events with a value lower than 55 on a grey scale from 0 to 255.

The overlapping windows lead to multiple detections of the same event. We decided to only show the first event occasion, if the score value is higher than the threshold multiple times in a row.

The goal is to show the operator an image of the road defect. This cannot be done using the same image, as where the sensors measured an event, because the car is positioned over the road defect at this point. To get the correct camera image, the program must show a video frame, that lies some frames before the event occurred in the sensor data. Therefore, the routine calculates the number of frames by going back 9 meters in the video data using the fps of the video footage and the current speed of the vehicle:

$$n_{\text{frames}} = \frac{9\ m \cdot fps}{v_{\text{curr}}}.$$

To better detect the damage cause, the program shows a short video sequence of the drive over the obstacle. After this, the user decides if the image contains a noticeable road defect and saves the event image.

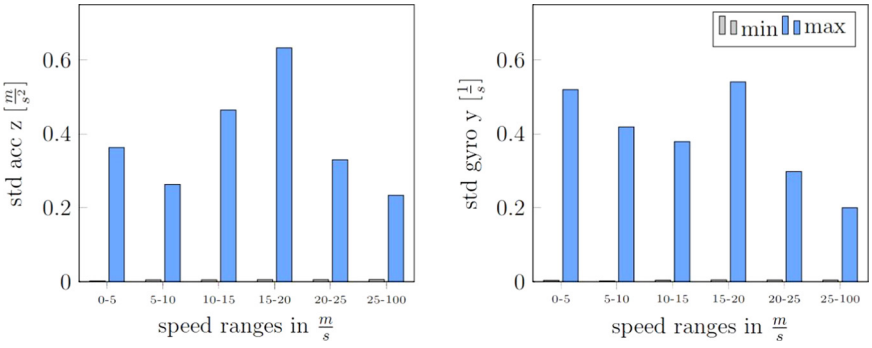

Figure 6: Score table entries for the minimum and maximum standard deviation of *z*-acceleration.

Mark damages on images

To label the events in the images, we use the external program 'labellmg' (Lin, 2020). In Figure 7 you can see the image annotation process with 'labellmg'. In this example, we identified two manholes and one sunken patch.

Results

Three test series are compared to show the effects of the threshold *T* (see Table 1). The test consists of 74 km driven distance in 163 min of video data. The evaluation of one image takes approximately 20 sec because the operator has to watch a short clip of the drive over the road defect and label the event afterwards. The operator was able to detect 47 true road events out of 119 shown images with the lowest threshold of 0.28. The most important events for image recognition are potholes and cracks which occurred 14 and 10 times. The average working time per true event was 51 sec. The algorithm detected a usable event every 1.6 km.

If the threshold is set higher, the number of images shown is lower. Therefore, the evaluation time is reduced but the number of road defect events in the data set drops as well.

The manual labeling process shows significant differences to the semi-automatic approach. The operator was able to evaluate 8.34 street km in 1,117 sec of video data, which took him 35 min. To compare these values to Maeda we treat this like showing 1,117 individual images to the operator. He was able to detect 31 events, of which were one

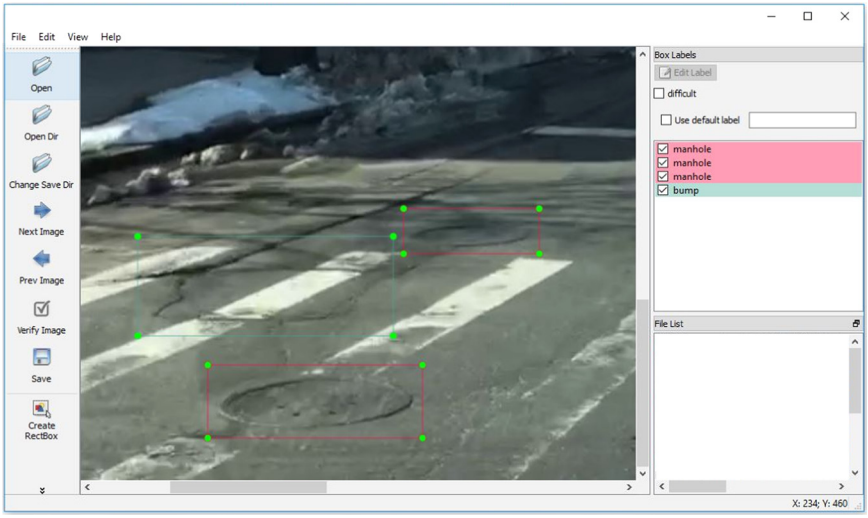

Figure 7: Image annotation using 'labellmg'.

Table 1. Comparison of evaluations with different thresholds.

Threshold	0.28	0.3	0.35	Manual
Total distance (km)	73.95	73.95	73.95	8.34
Total video time (min)	162.88	162.88	162.88	18.62
Evaluation time (min)	40	30	*17*	35
Images shown	119	89	50	1,117
Cracks	10	5	3	20
Patches	6	3	2	5
Potholes	14	9	6	1
Railroad tracks	1	1	0	0
Speedbump	4	4	0	1
Sunken manholes	12	9	5	4
Sum of events	47	31	16	31
Images containing events (%)	*39.5*	34.8	32	2.8
Time per true event (s)	51	58	64	68
Event every × m	1,573	2,385	4,622	*269*

pothole and 20 cracks. The average time per event was slightly higher than the semi-automatic approach. The operator was able to detect an event every 269 m of street data.

We compare our method to the data acquisition of Maeda and Angulo. Both papers used cameras which took one image per second. In 163,664 images the team of Maeda found 15,435 road defect events, which is a rate of 9.4%. Out of these events, 2.65% were severe damages like potholes. We can see very similar numbers in Angulo. They gathered 180,345 road images and labeled 45,435 events at a rate of 25.2%. The pothole percentage is almost the same with 2.64%. We compare these numbers to our approach (with a threshold value of 0.28) in Table 2 and Figure 8. Out of the 119 presented images, 47 contain a unique event of interest. This rate of 39.5% yields to a significantly lower evaluation time if we calculate with

Table 2. Comparison between our approach, Maeda, Angulo, and manual labeling.

	Proposed method	Maeda	Angulu	Manual (video)
Presented images	119	163,664	180,345	1,117 sec
Containing event (%)	*39.5*	9.4	25.2	2.8
Severe damages (%)	*29.8*	2.65	2.64	3.22
Light damages (%)	59.6	67.9	Unknown	93.5
Other (%)	10.6	29.45	Unknown	3.28
Event every × m	1,573	97	36	269

Note: The proposed method has a noticeable higher number of events per shown image and a significantly higher percentage of severe damages in the labeled set.

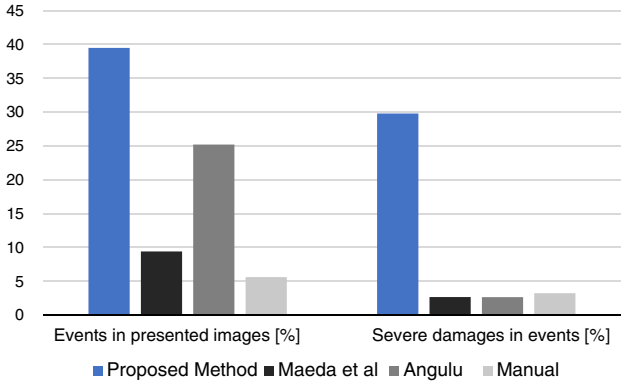

Figure 8: Comparison between our approach, Maeda, Angulo, and manual labeling. The higher number of events per shown image leads to a lower evaluation time and a less tiring workflow. The significantly higher percentage of severe damages in the labeled set is important for a ground truth with a balanced number of events per class.

a constant editing time per shown image. The most significant advantage of our method is the number of severe damages in the labeled data set. The value of 29.8% is more than 10 times higher than those of Maeda and Angulo. Naturally, the number of light damages detected is lower with our approach. A short manual evaluation of 1,117 sec of video data caught 31 events, which equals to a rate of 2.8% events per second. As only one of these events was a pothole, the severe damage percentage is also low with 3.22%. The biggest disadvantage of the proposed method is the large amount of data needed in comparison to manual labeling. In our case an event occurred every 1,573 m as in Maeda it occurs every 97 m and in Angulo every 36 m. This number may also depend on the overall road condition varying from region to region.

Conclusion

Evaluation conclusion

The results in the previous chapter show that the labeling process for image recognition in road defect prediction can be sped up by a substantial margin. Applying the presented methods results in a more balanced training data set. Angulo showed that this can increase the accuracy of machine learning algorithms.

An event that was found with the presented approach is ensured to have correlated visual and acceleration data. Both Maeda and Angulo chose to only save one image per second and search those for damage events. This approach is not suitable for training advanced continuous algorithms like object tracking, visual severity measurement, or sensor fusion applications. Our method is optimal for generating ground truth data for these tasks.

The semi-automatic labeling is advantageous to the manual approach for several reasons. The presented method is much quicker because you skip all video sequences where no road defect is measurable. We were able to go through big data sets and search for high impact events like potholes, speedbumps, and railroad tracks very efficiently.

On the one hand, our approach results in a lot of severe defects detected which is very important for road event image classification. On the other hand, the manual operators were able to detect smaller and low impact events like cracks and patches more frequently. Those events tend to occur very often but have a lower score than events like potholes and fall under the threshold. It can be advantageous to label a few street kilometers manually in order to generate a high number of low impact events.

The operator has the choice to set the detection threshold T according to their data. It is recommended to pick a low threshold if you have limited amount of high-quality data in order to cover as many events as possible. If the data set is big and not professionally recorded, it is better to pick a slightly higher threshold.

The operators reported that the semi-automatic approach was less tiring and more rewarding. The use of a computer with a touchscreen and smart pen is recommended in both cases due to the more natural workflows.

Commercialization outlook

Based on the described method the road inspection of a 5,000 km road network can be monitored by having several IMU and cameras traveling on many vehicles covering the road network regularly. With the proposed optimized, semi-automated analysis process road defects can be quickly mapped and assessed. Once they are prioritized, a road maintenance crew can be automatically dispatched to fix the identified road defects. This process can save up to an order of magnitude in data collection and analysis and cover hundreds of kilometers per unit vs. the current lower distance as described above.

With the proposed method we will be getting closer to preventative road maintenance and can ensure safer and smoother travels. Given the current inefficient spent on road asset monitoring and reactive

maintenance, this will be a substantial commercial opportunity.

N3T has started to incorporate the findings of this paper in its improved road defect analysis process. We have built a Smartphone App which uses an *z*-acceleration threshold to capture an image after winding back the frames of the camera to where the road defect can be seen. The image is automatically sent to a cloud-based web database for near real-time analysis. Further automation of the current human labeling process will be investigated using transfer learning and other deep learning methods.

Acknowledgments

The authors like to thank the Team at University of Berkeley to supply an IMU and image data set to develop and test our analysis model.

Literature Cited

Angulo, A., Vega-Fernández, J. A., Aguilar-Lobo, L. M., Natraj, S. and Ochoa-Ruiz, G. 2019. Road damage detection acquisition system based on deep neural networks for physical asset management. Mexican International Conference on Artificial Intelligence, Springer, Cham, pp. 3–14, available at: https://doi.org/10.1007/978-3-030-33749-0_1

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S. and Balakrishnan, H. 2008. The pothole patrol: using a mobile sensor network for road surface monitoring. Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, Association for Computing Machinery, New York City, NY, 29–39.

Li, Q., Chen, X. and Xu, W. 2013. Noise reduction of accelerometer signal with singular value decomposition and Savitzky-Golay filter. *Journal of Information & Computational Science* 10(15): 4783–4793.

Lin, L. and Liu, Y. 2010. Pothole detection based on SVM in the pavement distress image, 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Hong Kong, pp. 544–547.

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T. and Omata, H. 2018. Road damage detection using deep neural networks with smartphone images, *Computer-Aided Civil and Infrastructure Engineering* 33(12): 1127–1141, available at: https://doi.org/10.1111/mice.12387

Masino, J., Thumm, J., Frey, M. and Gauterin, F. 2017. Learning from the crowd: road infrastructure monitoring system. *Journal of Traffic and Transportation Engineering* 4(5): 451–463, available at: https://doi.org/10.1016/j.jtte.2017.06.003

Radopoulou, S. C. and Brilakis, I. 2016. Automated detection of multiple pavement defects. *Journal of Computing in Civil Engineering* 31(22): 04016057, available at: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000623

Ward, C. and Iagnemma, K. 2009. Speed-independent vibration-based terrain classification for passenger vehicles. *Vehicle System Dynamics* 47(9): 1095–1113, available at: 10.1080/00423110802450193

Wu, L., Mokhtari, S., Nazef, A., Nam, B. and Yun, H.-B. 2014. Improvement of crack-detection accuracy using a novel crack defragmentation technique in image-based road assessment. *Journal of Computing in Civil Engineering* 30(1): 04014118.

Xu, G., Ma, J., Liu, F. and Niu, X. 2008. Automatic recognition of pavement surface crack based on BP neural network. International Conference on Computer and Electrical Engineering, IEEE, pp. 19–22.

Zhou, J., Huang, P. S. and Chiang, F.-P. 2006. Wavelet-based pavement distress detection and evaluation. *Optical Engineering* 45(2): 027007, available at: https://doi.org/10.1117/1.2172917

Software:

Thumm, J. and Masino, J. 2019. Semi-Automatic Video Analysis, v1.0, Python, https://zenodo.org/record/3384989

Lin, T. 2020. LabelImg, v1.8.3, Python, https://github.com/tzutalin/labelImg