Large-scale Continual Road Inspection: Visual Infrastructure Assessment in the Wild

Supplementary Material

BMVC 2017 Submission # 664

1 Overview

In this supplementary material, we provide:

1. More implementation details about fetching images from Google Street View including all the parameter settings.

2. More experimental results that are not presented in the paper due to space limit.

3. More sample images from the proposed dataset.

2 Image Acquisition Details

A street view image request to the Google Street View API is an HTML URL of the form: `https://maps.googleapis.com/maps/api/streetview?parameters`. The `parameters` we used in our paper include:

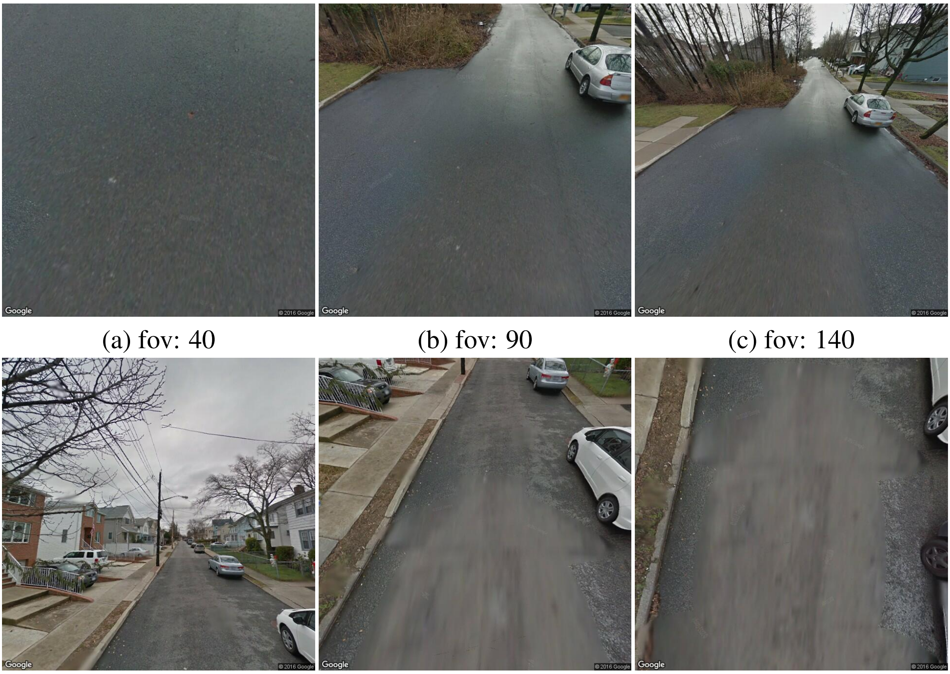
- `size`: the size of the output image.

- `location`: the longitude and latitude of a street segment.

- `fov`: the horizontal field of view. We fix it to 90°, which is chosen empirically. A large fov results in unnecessary distracting information included, which reduces the portion of the pavement in the image. Meanwhile, a small fov only focuses on a small part on the pavement which hardly gives an overall condition rating, as shown in Figure 1.

- `heading`: the facing direction of the camera. It varies from street to street. We set it to the direction of the street computed from start and end coordinates of the street.

- `pitch`: the up or down angle of the camera. We fix it to $-50°$, which is chosen empirically. We prefer the pitch angle that faces towards the pavement while avoids the artifacts caused by post-processing to remove the vehicle where the camera is mounted. The influence of different pitch value can be found in Figure 1.



(a) fov: 40 (b) fov: 90 (c) fov: 140

(d) pitch: 10 (e) pitch: -50 (f) pitch: -80

Figure 1: **The influence of different parameter settings.** (a) (b) and (c) show the fetched image from the Google Street View API with different fov (field of view) values. (d) (e) and (f) show the fetched image with different pitch angles.

Internally, Google Street View is a collection of discrete panoramas, each with an unique ID. The parameters `fov, heading, pitch` enable us to obtain an image that corresponds to a small part of the panorama. To collect data for a street segment, we traverse from the start coordinate to the end coordinate at a constant step. At each step, we check if the panorama ID is identical to that in the previous step. If it is, we skip this image to avoid duplication.

Panorama ID cannot be obtained automatically via the Google Street View API. It is provided by the following HTML URL: `http://maps.google.com/cbk?output=json&hl=en&ll=XX&radius=20&cb_client=maps_sv&v=4`, where `XX` is the longitude/latitude pair as we used before. By parsing the returned JSON data, we can acquire the panorama ID, as well as whether this location has Google Street View images, because some remote parts of the city might be inspected by the pavement condition raters but not covered by Google Street View.

3 Additional experiments

This section provides additional experimental results that could not be included in the main submission. Some results are obtained by the same model used in the main paper, but under different parameter settings. Other results are obtained using models that are not directly related to our claims, but they are included here for reference for follow-up studies. The results of these experiments are in Table 1 and Figure 2. We also present the confusion matrix of our best result (FV-CNN L1 Patch + random forest) in the paper in Figure 3.

SIFT-FV. We try different clustering settings. When we directly cluster all the data points into 96 centers, we achieve an average accuracy of 52.7%, 0.8% lower than using 256 centers. Since the dataset is unbalanced, we also try clustering GMM centers per class, known as aggregate clustering. We use 32 centers for each class and 96 centers in total. This configuration achieves results of 30.5%, 13.3%, 89.0% in three classes and average accuracy is 44.3%, which is 8.4% lower compared to the model without aggregate clustering. We hypothesize that the features in each class may share some commonalities. For example, in each class, vehicles are clustered into several centers. This kind of centers are duplicated in all three classes, which implicitly reduces the number of centers used to describe pavement conditions.

The best result using SIFT is achieved with the number of centers set to 384, which is the maximal number of centers we can test due to hardware limitations. The result slightly increases to 53.9% compared to 53.5% using 256 centers.

Fine-tuned CNN. The fine-tuning is done on image level, not street segment level. We assume all images within a street segment have the same label as the street. We start from VGG-D, which is used in our paper, and replace the last fully-connected layer (1000 ways) with a 3-way fully-connected layer. The ground truth labels are converted into one-hot vectors. The loss function is categorical cross entropy. The learning rate is set to 10^{-5} with a decay rate of 10^{-5} and momentum of 0.9. To tackle data imbalance, we use a batch size of 21, which contains 7 images per category. We calculate the validation loss and choose the model that has the lowest validation loss. This chosen model achieves an average accuracy of 49.7%, which is 11.4% higher than SIFT with SVM on image level.

Another way to handle data imbalance is data augmentation. We can augment the minority class by applying controlled transformation to the original images. We can randomly shift the RGB pixel value within a small range, shift the position of images, slightly change the scale of images, or flip the images left to right. Using these operations, for each "poor" image, we create 32 images with different random transformation. Then the network is fine-tuned in the same manner. However, the average accuracy drops by 4.2% to 45.5%. We think that augmentation generates images that are still too similar to the original image, and the network overfits to these images.

Regression Forest. If we turn the labels "poor", "fair" and "good" into numeric labels "0", "1" and "2" respectively and assume the degradation level can be described by a continuous value, we turn the texture classification problem into a regression problem. Based on our method which achieves the best result in classification, we conduct another experiment by replacing the classification tree with a regression tree in the forest. The mean squared error (MSE) for three classes are 0.62, 0.05 and 0.80 respectively. It seems the regression model leans to predict "poor" and "good" conditions into "fair".

4 Dataset samples

We show more sample images from our dataset. Pages 5-6 are images of the street segments in poor condition. Pages 7-8 show the street segments in fair condition, and pages 9-10 show images of good condition. For each street segment, we present 4 images. And we present 8 street segments in each degradation category.

Model	POOR	FAIR	GOOD	AVG
SIFT-FV (AC 96) + SVM	30.5	13.3	89	44.3
SIFT-FV (96) + SVM	74.1	37.3	46.6	52.7
SIFT-FV (384) + SVM	79.5	35.3	46.9	53.9
CNN-FT w/ aug	13.9	60.3	62.2	45.5
CNN-FT w/o aug	51.5	42.3	55.2	49.7
FV-CNN L1 Patch + RF	72.2	50.7	51.7	58.2

Table 1: **Additional experimental results.** Numbers in parentheses are the number of GMM centers. “AC” is short for aggregate clustering. “CNN-FT” is fine-tuned CNN and “aug” indicates whether the network is fine-tuned with minority class data augmentation. The best result in the main paper is also shown in the last row.

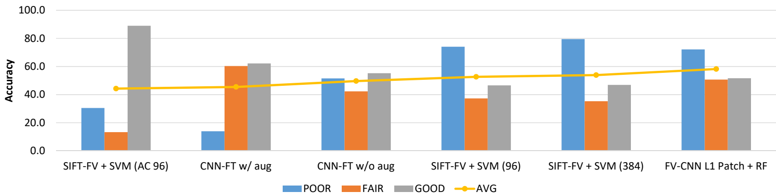


Figure 2: **Ranked extra experiment results.** The experiments are ranked by their average accuracy. The best result we achieved in the main paper (FV-CNN L1 Patch + RF) is also presented here as reference.

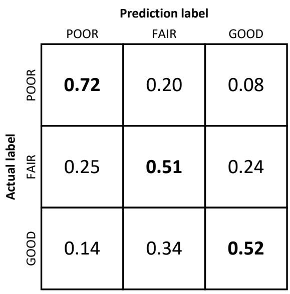
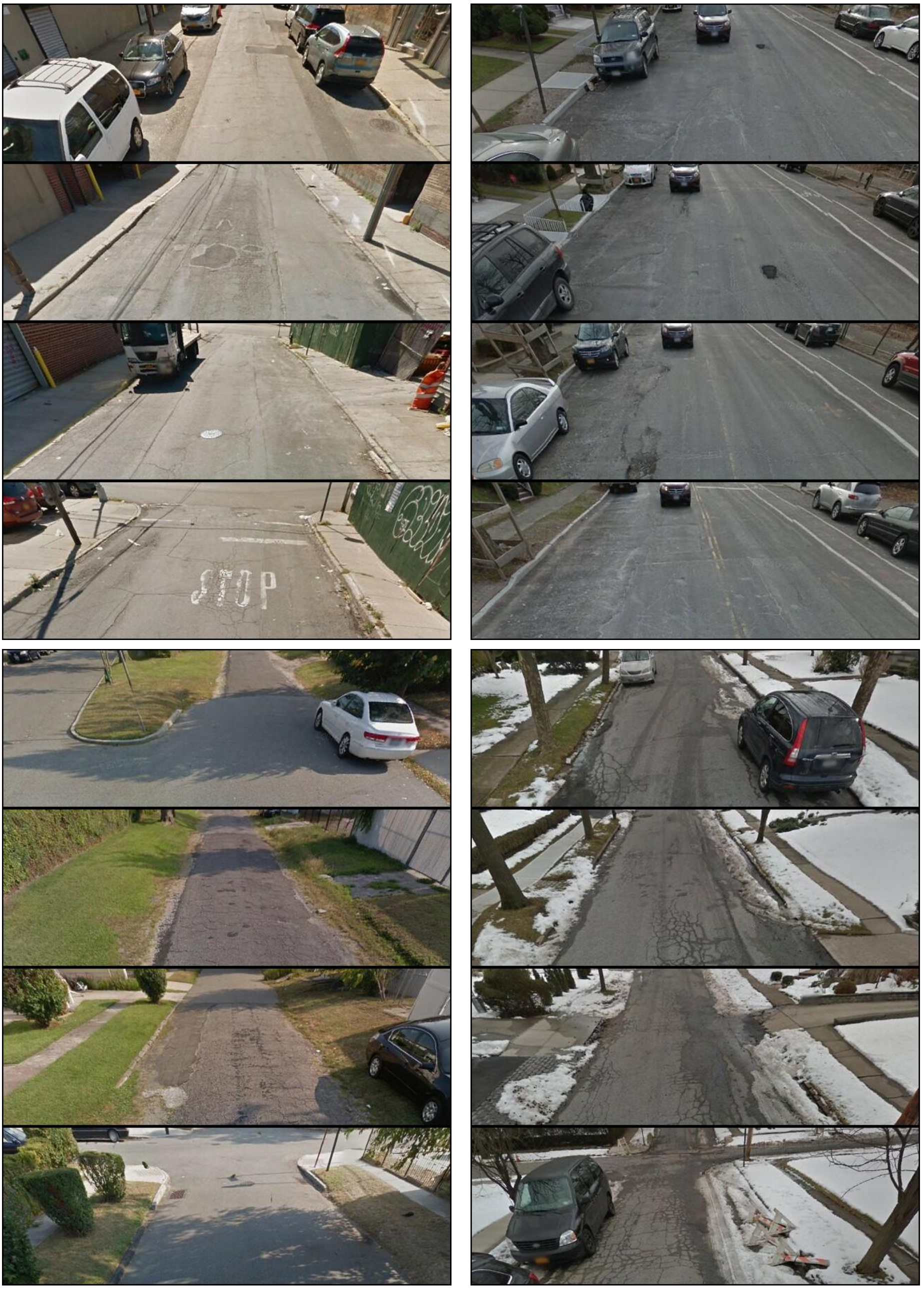


Figure 3: **Confusion matrix of the best result.** This is the confusion matrix of FV-CNN L1 Patch + RF, which achieves the best result in our paper.

POOR



POOR

FAIR

FAIR

GOOD

GOOD