Google Street View Images Support the Development of Vision-Based Driver Assistance Systems

Jan Salmen[*], Sebastian Houben[*], and Marc Schlipsing[*]

***Abstract*— For the development of vision-based driver assistance systems, large amounts of data are needed, e. g., for training machine learning approaches, tuning parameters, and comparing different methods. There are basically three possible ways to obtain the required data: using freely available benchmark sets, doing own recordings, or falling back to synthesized sequences. In this paper, we show that Google Street View can be incorporated as a valuable source for image data. Street View is the largest publicly available collection of images recorded from a drivers' perspective, covering many different countries and scenarios. We describe how to efficiently access the data and present a framework that allows for virtual driving through a network of images. We assess its performance and show its applicability in practice considering traffic sign recognition as an example. The introduced approach supports an efficient collection of image data relevant to training and evaluating machine vision modules. It is easily adaptable and extendible, whereby Street View becomes a valuable tool for developers of vision-based assistance systems.**

I. INTRODUCTION

Several vision-based assistance systems are already available in mass production vehicles today, e. g., lane detection, traffic sign recognition, and obstacle avoidance. Many more will follow within the next years, contributing significantly to the evolution of intelligent vehicles towards fully autonomous vehicles.

The development of vision-based modules typically requires large amounts of data: Machine learning approaches for object detection and recognition have to be trained. Parameter tuning for different scenarios can either be done manually or automatically. Finally, algorithms' performance have to be assessed on additional test data, that was not involved in the process of training or parameter optimization.

Collecting appropriate data is often more time-consuming and more expensive than the algorithm development itself. This holds even more if data from other countries has to be considered, e. g., to test the transferability of an existing module or to evaluate approaches for new scenarios.

Instead of recording and annotating hours of driving by oneself, sometimes publicly available data sets can be useful. However, there are only a few, e. g., for pedestrian detection [1][2][3], traffic lights detection [4], traffic sign recognition [5], and stereo vision [6]. Thinking of traffic sign recognition, for instance, only Germany is covered so far. Furthermore, these datasets were usually recorded with a specific goal and may not be applied on other, even similar, problems.

[*]J. Salmen, S. Houben, and M. Schlipsing are with the Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany `firstName.lastName@ini.rub.de`

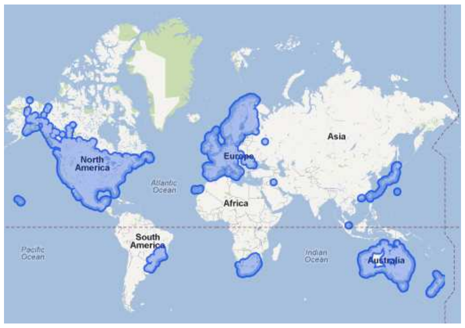

Fig. 1. Availability of Google Street View images, January 2012 (`http://support.google.com`).

Synthesized sequences are also used in practice as they allow to simulate driving long distances at low cost. They are particularly helpful for the evaluation of optical flow and stereo approaches, where *ground truth* data is very hard to obtain for real-world videos [7][8]. Nevertheless, for the development of security relevant systems one cannot rely exclusively on simulated data.

We argue that the retrieval of images from Google Street View is a powerful alternative, especially in earlier stages of development: Huge amounts of image data are available for many different countries all over the world (see Fig. 1) and they can be accessed easily and efficiently – as we showcase in this paper.

Thinking of the different types of vision-based algorithms needed for intelligent vehicles, Google Street View recordings do not satisfy all of them equally well. As the available images feature a relatively coarse temporal resolution, approaches relying on high frame rates (e. g., estimation of optical flow) cannot be applied reasonably. Nevertheless, the data can be very useful for the development of object detection and recognition algorithms.

Of course, relevant data is not annotated in Street View images – i. e., identification (labeling) of relevant objects. This would be done manually as it is common for recorded videos. Nevertheless, as the new approach presented here allows accessing a huge amount of video data, it is inappropriate to examine all images. Rather, this demands *iterative*, *semi-supervised*, and therefore *semi-automatic* approaches to object detection.

In this paper, we describe the basic principles needed for accessing Google Street View data from own applications (Sec. II). Section III presents our framework that allows virtual driving through arbitrary regions in order to collect data and evaluate algorithms. We conducted experiments

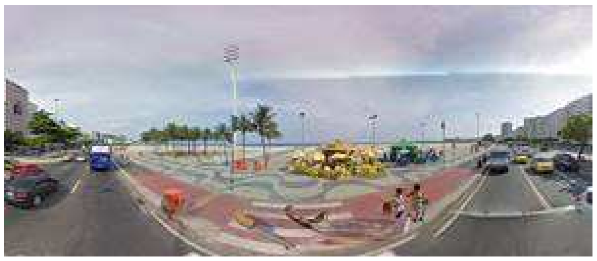

Fig. 2. Example of a thumbnail image obtained using the static API.

concerning performance and characteristics of the framework (Sec. IV). As an example of use, we considered the collection of traffic sign images for detection and recognition (Sec. V). Finally, results are discussed in Sec. VI.

II. THE GOOGLE STREET VIEW STATIC API

The image database of Google Street View is a network of adjacent 360° high resolution panorama images, which are divided into quadratic tiles. Those images are intended to be accessed via the Google Maps static application programming interface (API). In order to make them usable for algorithm development, some undocumented features have to be considered. Many of the technical details have been described recently by Jamie Thompson in his weblog[1]. Images from Google Street View have also been considered for research projects in other domains [9][10].

It is not completely clear that the presented way of usage is compliant with Google's terms of service[2] (for additional information, also see the related FAQ[3]). However, accessing the provided data conforms to the conditions, as long as no copies or modified versions of the electronic content are published. In particular, Google grants unlimited and free access to the Google Maps API for "non-profits and applications deemed in the public interest", e. g., universities. For the reasons discussed above, we are not going to publish data sets extracted from Google Street View for now.

The undocumented API features mentioned above are some simple HTTP requests to `cbk0.google.com/cbk` along with some parameters. For the sake of completeness, we introduce all four known commands below, which are simply attached to the prefix `http://cbk0.google.com/cbk?...` (i. e., HTTP GET). For now, two of them are used by the current framework.

A. Thumbnail from GPS coordinates

`...output=thumbnail&w=[SX]&h=[SY]&ll=[LAT,LNG]`

This request requires GPS coordinates (latitude `LAT` and longitude `LNG`). If a Street View panorama image close to the defined position is available, a thumbnail image of size `SX` × `SY` in *JPEG* format is returned. The maximum possible size is 300×128 pixels. The thumbnail shows the full 360° panorama, see Fig. 2 for an example.

[1]http://jamiethompson.co.uk/web/2010/05/15/google-streetview-static-api
[2]http://www.google.com/intl/en_ALL/help/terms_maps.html
[3]http://code.google.com/intl/en/apis/maps/faq.html

B. XML

`...output=xml&ll=[LAT,LNG]`

Using this request, one can receive additional information for a panorama image in XML format, e. g., place and recording time. For our purposes, the most important entries are the so called `pano_ids`. This is a unique identifier for the current panorama, but additional IDs are given for the adjacent panoramas, i. e., on the same road and / or for crossing roads. Those entries are called `links`.

It is noteworthy that also the street type (e. g. highway, rural road, inner city, etc.) can be derived from the XML data (entry `road_argb`), allowing for specialized processing.

TABLE I
ZOOM LEVELS AVAILABLE IN THE STREET VIEW STATIC API, NUMBER OF TILES, AND RESULTING PANORAMA IMAGE SIZES.

Zoom level	Number of tiles	Resulting image size
0	1×1	512×256
1	2×1	$1{,}024 \times 512$
2	4×2	$2{,}048 \times 1{,}024$
3	8×4	$4{,}096 \times 2{,}048$
4	16×7	$8{,}192 \times 3{,}584$
5	28×13	$14{,}336 \times 6{,}656$

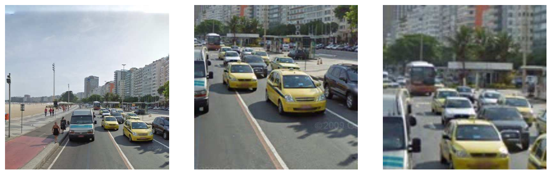

Fig. 3. Tiles in the panorama image (Fig. 2) for zoom levels 3, 4, and 5.

C. Tile

`...output=tile&panoid=[ID]&zoom=[Z]&x=[X]&y=[Y]`

Given the panorama ID, we are able to access the panorama image using this `ID` in the request above instead of GPS coordinates. In contrast to the thumbnails mentioned in Sec. II-A, larger images are divided into tiles of size 512×512 pixels which have to be requested independently specifying their relative positions `X` and `Y` within the complete image. Additionally, six different zoom levels (`Z` $\in [0, \ldots, 5]$) are made available. Table I gives an overview of the number of tiles and the resulting image size. Figure 3 illustrates some example tiles from different zoom levels. In the largest level, a panorama image with an amazing number of 90 mega pixels can be accessed.

D. Thumbnail from ID

`...output=thumbnail&w=[SX]&h=[SY]&panoid=[ID]`

This does basically the same as II-A, returning a thumbnail image, but taking a panorama `ID` instead of GPS coordinates.

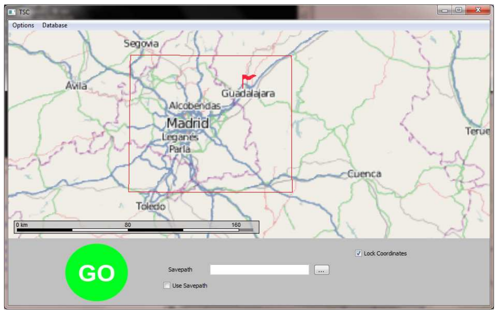

Fig. 4. User interface of our framework.

III. FRAMEWORK FOR VIRTUAL DRIVING

Based on the interface described in the last section, a framework that allows for virtual driving in Google Street View was developed. The system is operated through a graphical user interface shown in Fig. 4. The user is able to scroll and zoom a map realized with the freely available *Marble*[4] toolkit. The interface features the choice of a starting point, the definition of a rectangular search area and a visualization of the recent path. The map material is based on the publicly available OpenStreetMap data which might in principle deviate from the GoogleMaps coordinates. However, we did not experience any significant inaccuracies.

After initialization, the virtual drive can be started. It follows the logic documented in pseudo-code in algorithm 1. We make use of a *depth-first-search* algorithms, that is, always following the first possible branch, storing other possibilities for later processing. The algorithm has to keep track of all panorama IDs that already have been visited (list $visitedIds$) to avoid circular routes. If a node is visited for the first time and is contained in the search area (line 11), the corresponding XML data is requested. It is always followed the first link and the rest is stored in a stack ($storedIds$).

The algorithm stops if all reachable nodes inside the search area have been visited (line 5). Note that this does not necessarily mean that all roads have been considered – this requires a path to the starting point that lies completely inside the search area.

If the result data is to be stored locally (cf. Sec. V), a save path for images and / or a database can be specified. We are planning to extend the framework by the support of *plug-ins*, allowing to customize and extend the system dynamically at runtime.

[4]`http://edu.kde.org/marble`

Algorithm 1: Pseudo-code for virtual driving

Input: GPS coordinates (X, Y) of starting point and rectangular search area

```
1  $currentId \leftarrow getPanoIdForGpsCoordinates(X, Y)$;
2  Repeat
3      if $contains(visitedIds, currentId)$ then
4          if $isEmpty(storedIds)$ then
5              $STOP$;
6          else
7              $currentId = dequeue(storedIds)$;
8      else
9          if $isInsideSearchArea(currentId)$ then
10             $append(visitedIds, currentId)$;
11             $xmlData \leftarrow requestXmlData(currentId)$;
12             $links \leftarrow linksFromXml(xmlData)$;
13             $currentId \leftarrow links[0]$;
14             for $i \leftarrow 1$ to $i < numberOf(links)$ do
15                 $enqueue(storedIds, links[i])$;
16             $img \leftarrow requestImage(currentId)$;
17             $process(img)$;
```

IV. EVALUATION

The performance of the proposed framework is measured with respect to different quantities: What distance can be virtually covered per hour? How many panorama images are available meanwhile? How does the proposed search algorithm perform regarding runtime complexity and memory consumption? Moreover, we are interested in details concerning coverage and density of Street View within certain areas or street types.

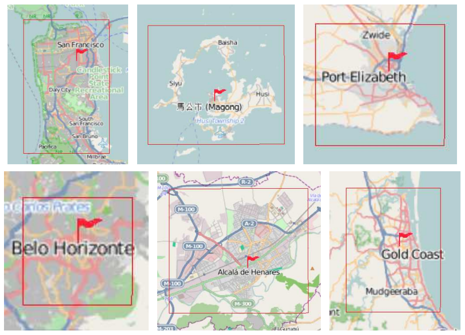

Fig. 5. Regions considered for an initial evaluation of our framework.

TABLE II
RESULTS OF EVALUATION CONSIDERING REGIONS FROM FIG. 5.

City	Distance *[km]*	#Panoramas	Time *[h]*
San Fransisco, USA	3023.8	268,127	10:03
Penghu Islands, Taiwan	514.8	44,736	1:41
Port Elizabeth, South Africa	2105.1	175,369	5:58
Belo Horizonte, Brazil	1426.3	128,459	4:33
Alcalá de Henares, Spain	409.9	32,645	1:01
Gold Coast, Australia	2509.8	219,558	8:34

In order to obtain baseline results, we only access XML data in our first experiment (i. e., we skip lines 16 and 17 of the pseudo-code given in algorithm 1). We selected several cities from different continents, specifying start point and search area (see Fig. 5) and collected relevant statistics over time. All experiments were conducted on a standard desktop PC (Intel Xeon W3520 CPU, 6 GB RAM) with a broadband

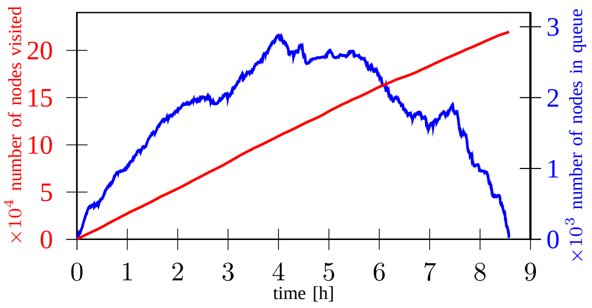

Fig. 6. Detailed performance analysis of virtual driving on Gold Coast, Australia.

connection (download speed approx. $5{,}000\ kb/s$). The results are given in Tab. II.

Compared to real-world recordings the effort and costs of 'virtual driving' is very low and the data acquisition is very fast, e. g. the mean virtual driving speed on our system was at $300\ km/h$. The mean distance between two panorama images amounted to $11.5\ m$, while no significant differences between the probed cities could be determined.

Detailed statistics for the virtual trip at Gold Coast, Australia are given in Fig. 6. The number of visited nodes develops almost linearly over time. The slight flattening that can be observed is very probably due to the increasing number of stored panorama IDs that have to be processed (see algorithm 1, line 3). The number of IDs stored for later processing first increases quite constantly up to $2{,}900$ and is then worked through.

If images are to be accessed, the resulting performance also depends on the processing that is done on the images, as well as on the considered zoom level. Therefore, the hardware used and the internet connection available can limit processing speed. In the following section, we consider a more complex example.

V. EXAMPLE OF USAGE

The "German Traffic Sign Recognition Benchmark" (GTSRB) presented in [5] is a single-image classification benchmark incorporating more than 40 sign classes. It comprises more than $50{,}000$ images originating from over $1{,}700$ real-world traffic signs (see Fig. 7). During the first competition phase, almost 200 results were submitted[5], reflecting the vivid interest in this topic.

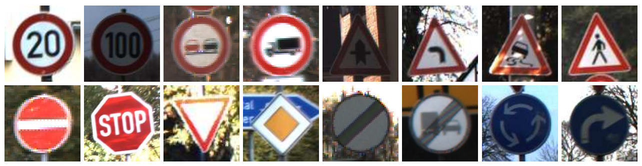

Fig. 7. Example images from the GTSRB benchmark.

While benchmarks like GTSRB can obviously have great impact, it took our group several months to collect this amount of data – in fact, more than $150{,}000$ images of traffic signs had to be labeled (semi-)manually in order to obtain the final dataset (we refer to [5] for more details). Therefore, it cannot be expect to have similar datasets for many other countries in the near future. Nevertheless, it would be very useful to see how far results for detection and classification of German traffic signs transfer to other countries having different signs types (e. g. different shapes, colors, pictograms, etc.).

The following experiment wants to show that the presented framework can be used to collect a large amount of data – similar to that of GTSRB – with significantly less effort.

A. Setup

In order to detect traffic signs in Street View images, we used the approach proposed by Viola and Jones [11], arguably the most popular real-time object detection framework. This detector is typically applied based on Haar-like features (see Fig. 8), simple filters which can be evaluated very cheaply using a pre-calculated *integral image*. It is noteworthy that Haar features can also be employed for other vision tasks in intelligent vehicles, e. g., estimation of optical flow [12] and, therefore, allow for a universal preprocessing.

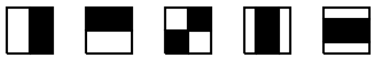

Fig. 8. Basic types of Haar wavelet features used for the Viola-Jones detector.

For the purpose of minimizing manual efforts, we propose a *semi-supervised* training procedure. An initial detector is trained based on an available, small dataset and is then applied to some Street View images. The extracted detections are manually sorted in three categories: *positive* (i. e., images showing traffic signs), *negative*, and *unfeasible*. The latter category applies to images that feature similar characteristics than the positive examples, but that are not explicitly searched. The positive and negative images are added to the initial training set and the detector is trained anew. This procedure can be repeated several times, increasing the detector's performance iteratively.

The only crucial point is how to obtain the data for the very first detector training. For the example considered here, images from the GTSRB dataset were used as positive examples and randomly chosen images from Street View as negative examples. We will discuss this issue in the next section.

For this experiment, we accessed panorama images of zoom level 3, which translates to a resolution of $3{,}072 \times 1{,}536$ pixels. For finding traffic signs it is sufficient to consider a third of the vertical resolution, discarding top and bottom parts.

B. Results

In contrast to the first evaluation presented in Sec. IV, we retrieved and processed panorama images here, resulting in a significant decrease of virtual driving speed to approx. $40\ km/h$ on the test system.

[5]`http://benchmark.ini.rub.de`

We were able to collect more than $10{,}000$ images of traffic signs within a single week. It is noteworthy that using Street View, typically less than five images are obtained from each real-world traffic sign instance. This means in reverse that the $10{,}000$ new images already cover more different sign instances than the original GTSRB dataset.

Beside the raw number of retrieved images, the performance of the final Viola-Jones detector can be measured regarding *precision* and *recall*. Therefore, an amount of $2{,}000$ panorama images were assessed manually, resulting in a precision of $49.3\,\%$ and a recall of $81.6\,\%$. Thus, only half of the detections are *false alarms*, while the major amount of the traffic signs have been found.

C. Discussion

The application of Street View as a data pool for training and testing of image-based machine learning algorithms includes several caveats that one should take care of to avoid statistically biased results.

Firstly, we point out that the available data itself is biased. The recordings were predominantly made during daytime with stable weather conditions. We note, however, that this argument is valid for several published benchmark datasets as well.

Secondly, there is no context between recordings. There are only few recordings of the same object and sometimes acquisitions were done months apart even with connected panoramas. Therefore, any algorithms can only work on single images.

The third caveat concerns the systematics. A data set of this size has to be processed automatically. Possible results (e. g., image sections) can be annotated manually afterwards to gradually improve the performance. This systematically excludes false negative examples from the training or test data.

This is the severest argument that we want to address carefully. Possibilities to circumvent biased training are semi-manual labeling, i. e., the manual annotation of randomly chosen panoramas which can be favored in the training phase, and the use of Street View data as a supplement to a more solid data base. In our example of German traffic signs, we do indeed strongly benefit from the GTSRB dataset that had already been collected. Nevertheless, this is only relevant for the very first training stages. If no data had been available, one could have collected a few training examples manually from Street View panorama images or any other image source.

We therefore argue that due to its tremendous size and nearly effortless access Street View can reasonably improve the acquisition of training and test data for video-based driver assistance systems in awareness of the above mentioned pitfalls.

VI. CONCLUSIONS AND OUTLOOK

The development of vision-based modules for intelligent vehicles has to rely on large amounts of image data. When no data is publicly available and simulated data cannot be used, appropriate image sequences have to be recorded. This can be very time consuming and expensive – and one can assume that this hinders the development of new assistance systems.

We propose the use of images from Google Street View in order to support the processes of machine learning from examples and algorithm evaluation. As driving and recording already have been done, it becomes much easier for scientist and developers to train and test their approaches on large data sets covering different countries.

In this paper, we explained in detail how the image data can be accessed through the static API provided by Google. This is permitted (at least tolerated) in particular for non-profit institutions. We presented a framework allowing for virtually driving through the Street View image data network and assessed its performance. Using a desktop PC, several hundreds of kilometers can be processed on a single day.

The gained image data is versatilely usable, e. g., for training and testing object detection and recognition algorithms and, thus, denote a highly valuable support for their further development. In future work, we plan to collect datasets for object detection and classification (e. g., traffic signs, traffic lights, cars) in different countries.

REFERENCES

[1] P. Dollár, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: A benchmark,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 304–311.

[2] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. M. Gavrila, “Multi-cue pedestrian classification with partial occlusion handling,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 990–997.

[3] C. Keller, M. Enzweiler, and D. M. Gavrila, “A new benchmark for stereo-based pedestrian detection,” in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2011, pp. 691–696.

[4] R. de Charette and F. Nashashibi, “Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates,” in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2009, pp. 358–363.

[5] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “The German Traffic Sign Recognition Benchmark: A multi-class classification competition,” in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2011, pp. 1453–1460.

[6] S. Morales and R. Klette, “A third eye for performance evaluation in stereo sequence analysis,” in *Proceedings of the International Conference on Computer Analysis of Images and Patterns*, 2009, pp. 1078–1086.

[7] W. van der Mark and D. M. Gavrila, “Real-time dense stereo for intelligent vehicles,” *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 1, pp. 38–50, 2006.

[8] T. Vaudrey, C. Rabe, R. Klette, and J. Milburn, “Differences between stereo and motion behaviour on synthetic and real-world stereo sequences,” in *Proceedings of the Conference on Image and Vision Computing New Zealand*. IEEE Press, 2008, pp. 1–6.

[9] B. Micusik and J. Kosecka, “Piecewise planar city 3d modeling from street view panoramic sequences,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2906–2912.

[10] A. R. Zamir and M. Shah, “Accurate image localization based on google maps street view,” in *Proceedings of the European Conference on Computer Vision*, 2010, pp. 255–268.

[11] P. Viola and M. Jones, “Robust real-time object detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2001.

[12] J. Salmen, L. Caup, and C. Igel, “Real-time estimation of optical flow based on optimized haar wavelet features,” in *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization*, 2011, pp. 448–461.