Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Travel Time Estimation by means of Google API Data ⋆

Jan M. S. Wagner * **Manuel Eschbach** * **Kari Vosseberg** *
Marc Gennat **

* *Graduate Program in Product Development in Mechanical Engineering, Hochschule Niederrhein, University of Applied Sciences, 47805 Krefeld Germany, (e-mail: jan.wagner1@stud.hn.de, manuel.eschbach@stud.hn.de, kari.vosseberg@stud.hn.de).*
** *SWK E^2 Institute for Energy Technology and Energy Management, Hochschule Niederrhein, University of Applied Sciences, 47805 Krefeld Germany (e-mail: marc.gennat@hs-niederrhein.de)*

Abstract: Modern urban planning not only has to coordinate the needs of many different inhabitants and traffic participants, but also faces other challenges such as modal shift towards sustainable transportation. A comprehensive database of historical traffic, which would facilitate a decision based on data, is lacking in many cities. This became clear in consultation with our partners from the city of Krefeld. A novel method is the use of public available traffic information such as traffic colors or travel time data from navigation providers.
In this contribution a method is applied to estimate the travel duration from the traffic colors on city level. Here we show for the first time how it can be applied cost-effectively and how the accuracy can be estimated in combination with a validation of the database by test drives for urban streets. It will also be investigated, whether speed variation has an influence on the estimation of driving time. We found out that this influence is negligible for the investigated example and the mean deviation of estimation accuracy from the measured values is less than six percent. Based on these promising results, it is possible to build up a database for improved urban and traffic planning at low cost. This can lead to better information for all traffic participants, thus, an improved traffic flow control could result in a reduction of car traffic emissions in the end.

Keywords: Parameter Estimation, Convex optimization, Floating Cars, Travel Time, Google Maps

1. INTRODUCTION

Emission reduction in traffic, for example through development towards Smart City is an important social duty. In consultation with city officials from Krefeld, which represents a German regional center (see Landesplanungsbehörde NRW (2017)), several challenges in the implementation present themselves in the field of mobility management. Past approaches on emission reduction, including driving restrictions and speed limits proved to be of limited use as shown in Mishra et al. (2019); Davis (2008). Additional speed limits for instance lead to an increased amount of stops at red traffic lights and thus to decreased efficiency of traffic flow. The city administration itself has regularly an insufficient knowledge of traffic. A detailed database as basis for planning would be very useful.

Nowadays many traffic information for German medium sized cities as Krefeld comes from induction loops (Gramaglia et al. (2013)). Additionally, a small amount of cameras is in use at bigger intersections. Sporadically the different kinds of vehicles crossing a certain point at peak hours are counted manually to determine if there is a need to alter the traffic situation at that specific point. All of these methods have in common, that they are only able to capture traffic in a local area, but there is no information about the traffic in between those areas, see Lenhart et al. (2008). Moreover, induction loops are often not able to send live data, as it is in Krefeld. Instead each loop has to be assessed individually after they have recorded a certain amount of data. Therefore, the method of Wang and Xu (2011) based on live data of induction loops is not possible.

A first step towards understanding the overall traffic situation is done by gathering live traffic information, which is provided by Bing, TomTom, Google and Here according to Bauer et al. (2019). Car floating is also portrayed by traffic colors in Google Maps and can be queried as images using the Google Maps application programming interface (API). To identify and possibly resolve shortages and other problematic set ups, data from the past would be needed. However, Google does not allow access to their database. By using Google's API it is possible to build up an own database. The cost of the different APIs is a maximum of 10 USD per 1000 queries, see Google Maps (2019b); Google LLC (2019). However, a single image can cover a whole city, while one single travel time query would be required

⋆ This research was supported by Stadtwerke Krefeld AG and SWK E^2 Institute for Energy Technology and Energy Management.

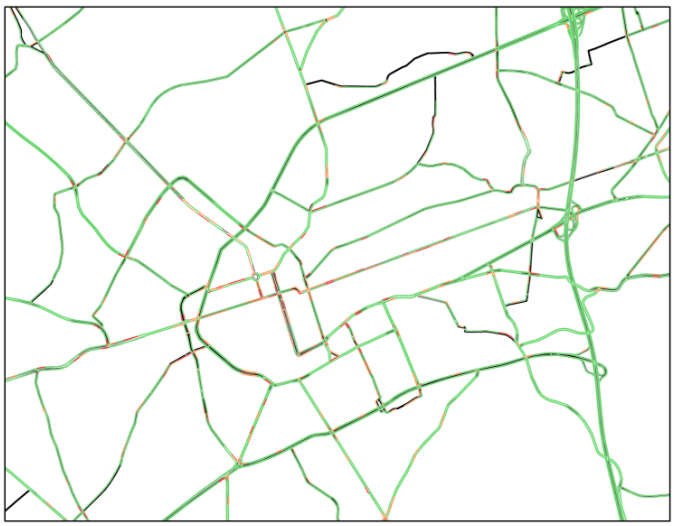

Fig. 1. Traffic information of the main streets in the reduced Google Maps view of Krefeld

for every single street in a city. Wagner et al. (2020) were able to show how travel time can be estimated based on the traffic colors of a section of autobahn A57.

In this paper it is shown for the first time how this analysis can be used for inner-city roads and hopefully help to improve future traffic planning.

2. DATABASE FOR TRAFFIC PLANNING: LOCAL ROADS

Applying the methods of Wagner et al. (2020) to small inner-city roads requires a higher precision of detecting individual road components such as directional lanes or intersections. The robust implementation of this detection is very complex. The method is shown on an example street, which is given as a 3.4 km section of the Cracauer Street and Friedrich-Ebert Street in Krefeld (Fig. 2).

2.1 Data Collection

For the analysis of traffic colors only basic elements of Google Maps are relevant. Other elements of map visualization such as buildings or water are not relevant. Thus, the later map-computation will be more robust by switching off theses layers. The modified map-representation

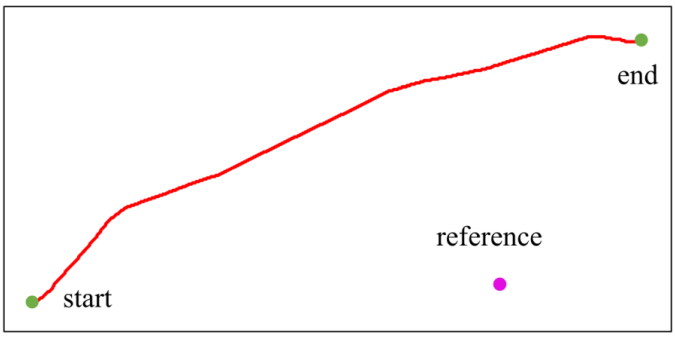

Fig. 2. Street path of the example street: given start and end coordinates in green, street path from Google API and reference pixel as coordinate origin

```
8     "distance" : {
9        "text" : "3,4 km",
10       "value" : 3434},
11    "duration" : {
12       "text" : "8 minutes",
13       "value" : 462},
14    "duration_in_traffic" : {
15       "text" : "8 minutes",
16       "value" : 459}
```

Listing 1. Excerpt from a JSON file for the example street with length of the street, estimated average travel time, travel time with live traffic information

is implemented in a custom website using JavaScript, whereby black roads remain with traffic overlay on white background, as shown in Fig. 1 for the Krefeld map section.

For each investigated street the pixel path of this street has to be determined. Necessary start and end coordinates are assumed to be given. The path is represented by a customized web page as a red line with green start and end markers. In addition, a reference pixel is colored violet, which together with the selected resolution (1 Pixel $\approx$ 7 m) and section defines the position of each pixel of all images in the coordinate system (Fig. 2). This is necessary, because a slight fluctuation of the section position can occur during the creation of the images.

The coordinates are also used to query the travel time on this specific street using Google Maps Distance Matrix API. The JSON file provided by Distance Matrix API returns two important values, which are used in further computation: `duration in traffic` and `duration` (listing 1). The first value provides the actual travel time in seconds using live information, the latter does not use live information and delivers an estimated average travel time for the specific route.

2.2 Data Processing

From image (Fig. 1) traffic information in form of the colors green, orange, red and dark red are extracted for each investigated street. The street path (magenta in Fig. 3) on smaller streets does not always lie exactly in the middle of the road. Since both directions have to be

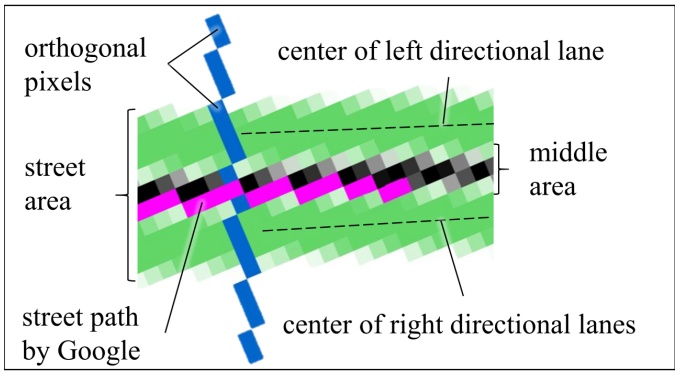

Fig. 3. Street structure: Google street path in magenta, orthogonal pixels o_p in blue and the most important street areas for the analysis

Algorithm 1 color classification

Input matrix $S \in [RGB]^{n_p \times o_p}$
Output matrix C: classified colors $\in [1, \cdots, 6]^{n_p \times o_p}$

Convert color RGB $\rightarrow$ Lab (lightness, a, b, color axes)
for each triplet $(L, a, b) \in [Lab]^{n_p \times o_p}$ **do**
  if $a < 10 \wedge b < 10$ **then** ▷ background colors
    if $L >= 90$ **then**
      $C \leftarrow 5$ ▷ white
    else
      $C \leftarrow 6$ ▷ black
    end if
  else ▷ google colors
    if $a < 0 \wedge b > 0$ **then**
      $C \leftarrow 1$ ▷ green
    else if $L > 50 \wedge a > 60 \wedge b > 20$ **then**
      $C \leftarrow 2$ ▷ orange
    else if $40 < L < 81 \wedge a < 70 \wedge b < 55$ **then**
      $C \leftarrow 3$ ▷ red
    else if $L < 40 \wedge a < 60 \wedge b < 40$ **then**
      $C \leftarrow 4$ ▷ dark red
    else
      classification error ▷ detect color changes
    end if
  end if
end for

analyzed surrounding pixels of the street path is examined to find both directional lanes.

To do that RGB values of the pixels orthogonally to the left and right of the street path are extracted. Empirical investigation on different street paths results in a ten pixels' area to either side of the street center. Thus, a total of 21 blue orthogonal pixels o_p are shown in Fig. 3 for an example position.

The matrix $S \in [RGB]^{n_p \times o_p}$ results from color information acquisition of Fig. 3, whereby o_p represents the number of orthogonal pixels and n_p returns the investigated street length as number of pixels. For these pixels the RGB values are extracted from a traffic image like Fig. 1. These color values form a whole range of color gradations, which are mainly caused by aliasing (see Crow (1977)). The light green pixels at the roadside in Fig. 3 represent good example for aliasing.

For further computation the bandwidth of color values is classified in algorithm 1 as numerical values from 1 to 6 (green, orange, red, dark red, white, black). RGB is not a

Algorithm 2 center directional lanes

Input matrix $C \in [1, \cdots, 6]^{n_p \times o_p}$ (see algorithm 1)
Output vectors $[cl, cr]$: center of direct. lanes $\in \mathbb{Z}^{n_p}$
vectors $[bl, br]$: street boundaries $\in \mathbb{Z}^{n_p}$

for each $c_{row} \in [1, \cdots, 6]^{o_p}$ row in C **do**
  $b \leftarrow c_{row} \geq 5$ ▷ find background colors
  $[ml, mr] \leftarrow$ edges indices middle area in b
    ▷ find neighboring colored areas to define
  $l \leftarrow c_{row}(1 : ml) < 5$ ▷ left directional lane area
  $r \leftarrow c_{row}(mr : end) < 5$ ▷ right direct. lane area
  $[bl, br] \leftarrow$ indices of the outer edges of the street
  $[cl, cr] \leftarrow$ center of area l and r
end for

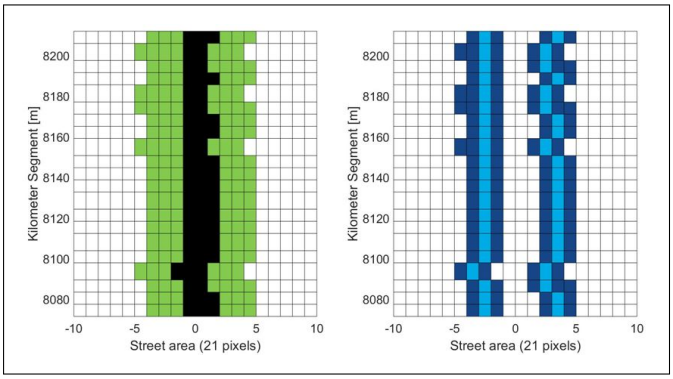

Fig. 4. Visualization of matrix C for a piece of road: classified colors (left) and the light blue centers of directional lane areas cl, cr form algorithm 2 (right)

suitable color space to classify these colors. More robust results have been achieved after converting the image into the LAB color space (L: lightness, a and b represents two color axes, see Kaur and Kranthi (2012); Connolly and Fleiss (1997)).

To identify the center of the directional lanes (Fig. 3) it is only important whether a pixel has a background color ($C > 4$) or a traffic color ($C \leq 4$). Algorithm 2 describes how the areas of the street presented in Fig. 3 and the resulting lane centers are determined. Fig. 4 shows the results of both algorithms. On the left hand side traffic image details from Fig. 3 are classified ($C \in [1, \cdots, 6]$) without aliasing (algorithm 1). On the right hand side all background colors ($C \in [5, 6]$) are shown as white, the directional lane areas in dark blue and their centers as light blue (algorithm 2).

The indices of the lane centers are converted back into coordinates and stored for further analysis of traffic images, which are recorded at different points of time and at different dates.

Inner-city streets are characterized by many intersections, which challenge the directional lane identification. An example intersection is given in Fig. 5, whereby the orange pixels of the horizontal road overlap the right lane of the vertical example street. It is necessary to take intersections into account. Without considering intersections, the center directional lane algorithm would be corrupted.

Due to the straightened representation of the road (Fig. 4) a deviation of values in the boundary vectors (see bl and br in algorithm 2), which does not equalize after two steps, is a good classifier for intersection detection (algorithm 3). After empirical evaluation of different intersections, limit

Algorithm 3 intersection detection

Input vectors $[bl, br]$: street boundaries $\in \mathbb{Z}^{n_p}$
Output vector $[intersect]$: inters. detection $\in \mathbb{Z}^{n_p}$

for each $bl, br \in \mathbb{Z}^{n_p}$ **do**
  $\Delta \leftarrow$ difference to the entry after next
  $index_\Delta \leftarrow$ find $\Delta > 2$ ▷ limit differential value
  if $index_\Delta \neq \emptyset$ **then**
    find $start$ and end of each intersection
    $intersect\,(start, \cdots, end) \leftarrow \mathbf{1}$
  end if
end for

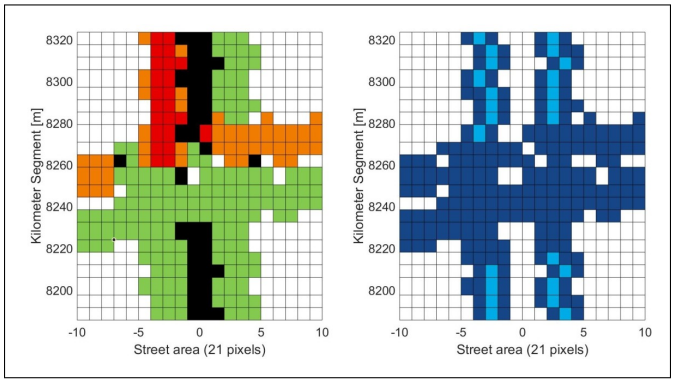

Fig. 5. Visualization of the matrix C for an intersection: vector *intersect* (alg. 3) hides the light blue pixels

differentiation value parameter in algorithm 3 is set to 2. The vector *intersect* with the length of the street n_p is set to one in intersections. Thus, in the intersection the center of directional lanes is not displayed (Fig. 5, light blue line). With the help of this vector for the left and right directional lane, each intersection is filled with the traffic color of the pixel before when evaluating the colors, in order to prevent errors caused by intersecting streets.

Using the center coordinates of the directional lanes the traffic colors are extracted from each image of the day and depicted as a heatmap (Fig. 6). The black areas indicate missing traffic information.

2.3 Estimation of travel time using traffic colors

Heatmap colors represent a traffic delay relative to `duration` (listing 1). A parameter estimation is used to compute travel time for each investigated street. `Duration` (listing 1) is a constant value – this means Google API returns the same value for the same street – while `duration in traffic` varies due to live traffic data.

To determine the correlation between travel time and colors, the latter are assigned to parameters p_1 to p_5. They are related to the mean travel time per pixel t_p, which is calculated by the travel time t (`duration` in listing 1) divided by the amount of pixels in the street path n_p with $t_p = {}^{t}/_{n_p}$. Travel times t_{F_j} in seconds are extracted from each `duration in traffic` capturing, whereby j represents the j^{th} capturing of the day and n_c the amount

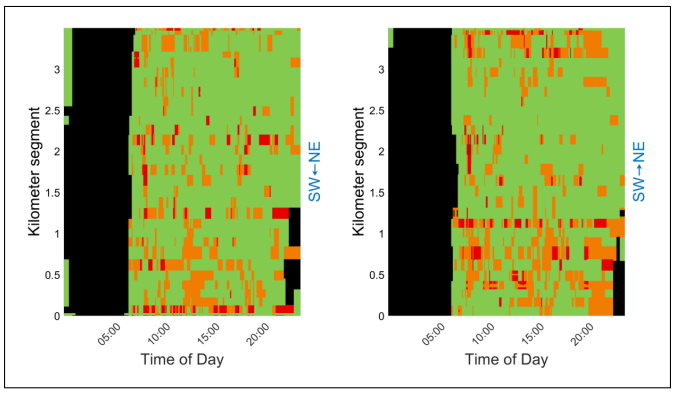

Fig. 6. Heatmap of traffic colors for the example street on June, 5^{th} 2019 for both directions

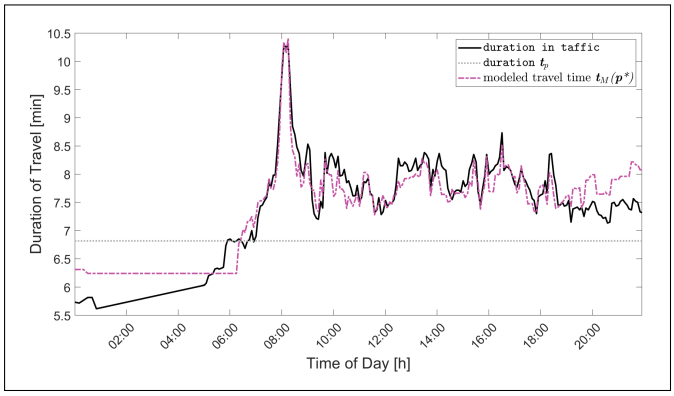

Fig. 7. Visualization of `duration in traffic`, `duration` and the modeled travel time $\mathbf{t}_M(\mathbf{p}^*)$

of image capturing with

$$\mathbf{t}_F = (t_{F_1}, \cdots, t_{F_{n_c}})^T. \tag{1}$$

Matrix $\mathbf{A}$ is defined with

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & \cdots & A_{1,n_c} \\ \vdots & \ddots & \vdots \\ A_{n_p,1} & \cdots & A_{n_p,n_c} \end{pmatrix}, \mathbf{A} \in [1,2,3,4,5]^{n_p \times n_c}, \tag{2}$$

and is created from color values of the heatmap (Fig. 6). Values of the modeled travel time t_{M_j} at the j^{th} capturing are computed with

$$t_{M_j}(\mathbf{p}) = \sum_{i=1}^{n_p} p_{A_{ij}}\, t_p, \qquad j \in [1, \cdots, n_c], \tag{3}$$

by which $p_{A_{ij}}$ returns the parameter value of the color at i, j^{th} position in $\mathbf{A}$. The objective function G used for parameter estimation is given by

$$G(\mathbf{p}) = \sum_{j=1}^{n_c} \left(t_{Fj} - t_{M_j}\right)^2 = \sum_{j=1}^{n_c} \left(t_{Fj} - \mathbf{c}_j^T\, \mathbf{p}\, t_p\right)^2, \tag{4}$$

whereby $\mathbf{c}_j^T = (c_{1j}, c_{2j}, c_{3j}, c_{4j}, c_{5j})$ represents the count of the k^{th} parameter with $k = (1, \cdots, 5)$ during the j^{th} travel time capturing. With this the Hessian matrix of the objective function is computed to

$$\mathbf{H}(\mathbf{p}) = \frac{\partial^2 G(\mathbf{p})}{\partial \mathbf{p}^2} \succeq 0. \tag{5}$$

The Hessian is positive semi-definite, which can be shown using the given symmetric matrix $\mathbf{H}$ and all principal

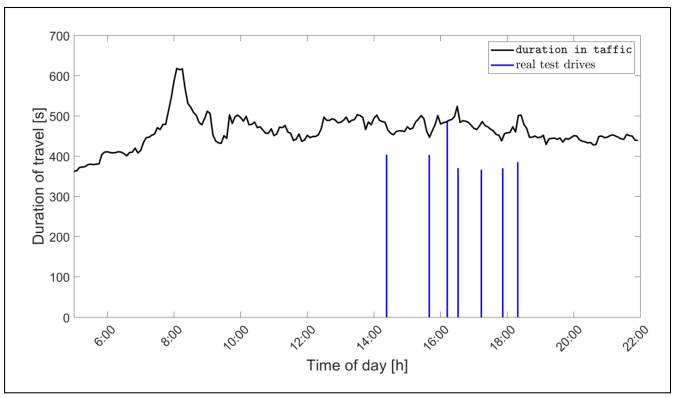

Fig. 8. Travel times with traffic influence for example street: `duration in traffic` in black, the real test drives in blue

minors equal to 0 with exception of the first one, which is computed to $\sum_{j=1}^{n_c} 2{c_{1_j}}^2 {t_p}^2 \geq 0$. Thus, $G(\mathbf{p})$ is a convex function over a convex set $\mathbf{p} \in {\mathbb{R}_{\geq 0}}^5$ of non-negative real numbers, and

$$\mathbf{p}^* = \operatorname{argmin}\left(G(\mathbf{p})\right) \tag{6}$$

represents a convex optimization problem computable with Matlab Optimization Toolbox from The MathWorks Inc. (2018). The travel time is computed from (6) with $\mathbf{t}_M(\mathbf{p}^*)$, which is shown in Fig. 7, where a larger deviation can be seen between 7 pm and 5 am. The relative error

$$e_{rel_j} = \frac{\left|t_{F_j} - t_{M_j}(\mathbf{p}^*)\right|}{t_{F_j}}, \quad j \in [1, \cdots, n_c] \tag{7}$$

is computed for each j. Mean value and standard deviation is calculated from it (see Tab. 1).

3. VALIDATION OF GOOGLE MAPS DATA

Real test drives were carried out to check for deviations or lags from `duration in traffic` received from Google API in comparison to the actual travel time. The before introduced example street was driven off repeatedly in the afternoon, hereby GPS data was recorded every second. This was done with two Android smartphones using Phyphox (see Staacks et al. (2018)). Simultaneous recording of different sensor data such as time, location, and speed for later evaluation was carried out (see Monteiro et al. (2019)).

In the following only one direction is examined in order to show the procedure. Travel times measured during the real test drives versus `duration in traffic` from Google Distance Matrix API are shown in Fig. 8. All GPS measurements which are affected by traffic show a deviation from values received from Google Maps (2019a). All measured times are shorter and have a relative error of $(16.5 \pm 10)\,\%$, whereby the notation $(x \pm s)$ is given in this contribution with the assumption x is normally distributed and s represents the standard deviation.

So far, the value travel time per pixel t_p is assumed to be constant. Regardless of the traffic situation, speed in urban areas varies significantly due to many traffic lights and local speed limits. Therefore t_p is adjusted for each pixel using the collected data taking into account speed differences along the street.

The length of test drives was measured with $(3.34 \pm 0.05)\,\mathrm{km}$, which means a deviation of -94 meter compared to the length given by Google Maps (listing 1). For each test drive the calculated length is divided into n_p equidistant segments, which represents the number of pixels along the examined street. Deviation of length is eliminated by this normalization. Driving time is interpolated linearly as a function of the distance using roughly 350 individual measuring points. Travel time per segment with the total number of segments $n_p = 500$ is determined using these interpolated values. The mean value of the real test drive travel times of each segment (length of one pixel) is used for the vector $\mathbf{t}_{np}$, which is shown in Fig. 9. Modeling the Travel time is done by adapting (3), which returns the model travel time vector $\mathbf{t}_{Mp}$ with

$$t_{Mp_j}(\mathbf{p}) = \sum_{i=1}^{n_p} p_{A_{ij}}\, t_{np_i}, \qquad j \in [1, \cdots, n_c]\,. \tag{8}$$

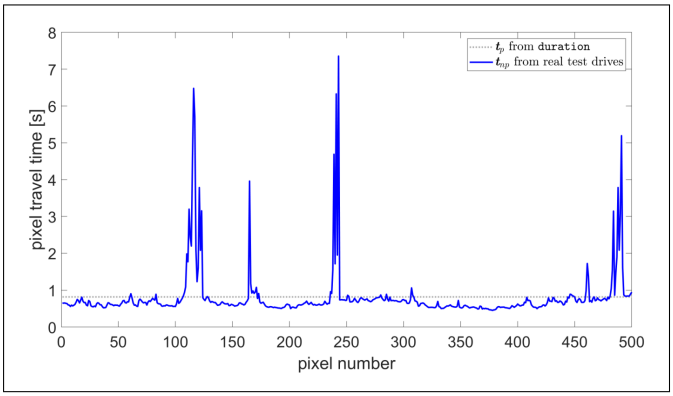

Fig. 9. Duration of stay per pixel for the example: constant `duration` in grey, blue from real test drives

Starting from the modified vector $\mathbf{t}_{np}$ a new color parameter estimation can be made, which considers different speed during one test drive. Therefore (4) is adapted to

$$G(\mathbf{p}) = \sum_{j=1}^{n_c} \left(t_{Fj} - t_{Mp_j}\right)^2 = \sum_{j=1}^{n_c} \left(t_{Fj} - \sum_{i=1}^{n_p} p_{A_{ij}} t_{np_i}\right)^2 . \tag{9}$$

The comparison of `duration in traffic` and the modeled travel time per segment $\mathbf{t}_{np}$ using the optimal parameter set $\mathbf{p}^* = \operatorname{argmin}(G(\mathbf{p}))$ results in Fig. 10.

4. CONCLUSION AND DISCUSSION

The correlation between traffic colors and travel time can be determined by means of parameter estimation. This parameter estimation has to be conducted for each road. Once factors for the parameters $\mathbf{p}^*$ have been specified, a single image can be used to calculate the delays for all roads shown.

Until now, small roads could not be evaluated without errors due to the significant deviation of the road path from Google to the actual location of the road. Therefore, only large roads such as freeways (autobahns) were analyzed (Wagner et al. (2020)).

With the new algorithms 2 and 3 identifying directional lanes, the traffic colors can be extracted without errors and the estimated driving time can be determined with less than 6 % deviation ($1\,\sigma$) to `duration in traffic` using the parameter procedure (Tab 1). This error estimation

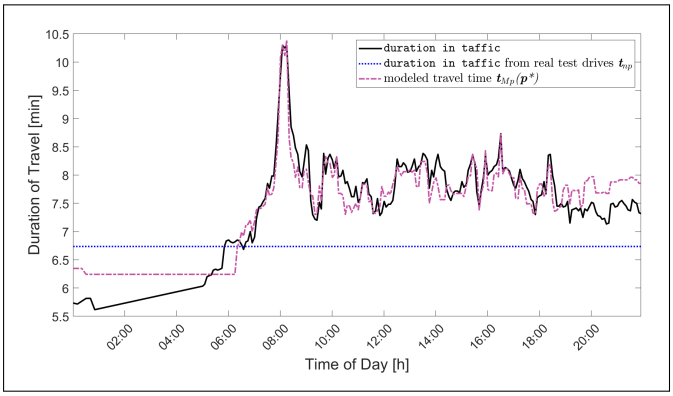

Fig. 10. Visualization of `duration in traffic`, `duration` and the model values $\mathbf{t}_{Mp}(\mathbf{p}^*)$ from colors for the example street with $\mathbf{t}_{np}$ from real test drives

Table 1. Parameters and resulting relative error

evaluation	example street, constant t_p	example st., $\mathbf{t}_{np}$ real test drives	A57 from Wagner et al. (2020)
p_1 (green)	1.0468	1.0566	1.0919
p_2 (orange)	1.2738	1.2649	2.1796
p_3 (red)	2.2937	2.3103	3.1213
p_4 (dark red)	4.4213	3.2832	5.2527
p_5 (black)	0.9154	0.9267	–
rel. error e_{rel}	3.408 %	3.328 %	10.366 %
standard dev. s	2.645 %	2.532 %	6.618 %

applies, as long as input data is randomly distributed, which is not investigated here. In further research more extensive test drives might be able to replace `duration in traffic` as a basis for the model. Comparing the accuracy of Google API through test drives, a shorter real driving time is observed. There are different possible explanations for deviation of the GPS measurements to driving times of Google:

- Real test drive measurements do not take into account the time for parking out and accelerating. Nor the time for decelerating at destination were carried out, because the real test drive were measured with a driving start.
- It seems probable Google schedules a security buffer in case of being unexpectedly slowed down. This slow down could be caused by a missed traffic light or a parked car.
- Google API returns only average values for all divers with all kind of vehicles. The test drives were made by only one driver with a car who may have driven closer to the speed limit than others.

Regardless of the time difference over the entire route, the local speed can be determined via the test drives. Usage of local speed $\mathbf{t}_{np}$ leads to an insignificantly smaller deviation from the measured values `duration in traffic` than on the basis of Google's `duration`. The additional effort of real drives is not necessary considering the minor effect it has for the examined street (Tab. 1). A certain deviation is always caused by the continuous function of `duration in traffic` being approximated by means of discrete parameters for only five colors (green, orange, red, dark red, black).

The method with constant pixel time $\mathbf{t}_p$ (see Wagner et al. (2020)) was used to calculate the color parameters for a section of autobahn A57. The investigated road length was 8.8 km and results yielded an accuracy of $(10.4 \pm 6.6)\,\%$. The inner-city road length examined in this work is only 3.4 km. The calculation resulted in an accuracy of $(3.4 \pm 2.6)\,\%$ with the constant pixel time $\mathbf{t}_p$ (see Fig. 7 and Tab. 1). Further investigations are needed to determine, if shorter streets will show always a smaller relative error.

All in all, Google Maps data and the Google API provide a cost-effective opportunity to build a city-wide traffic database to support local authorities in urban planning. Before deploying this method, the terms and conditions (Google Distance Matrix API) must be reviewed. Further investigations must show whether a larger number of test runs could improve the accuracy even further. If this method could be applied to metropolis areas, further research is needed. The cause of the deviation in the evening (Fig. 8 and 10) should also be investigated.

REFERENCES

Bauer, T.P., Edinger, J., and Becker, C. (2019). A qualitative and quantitative analysis of real time traffic information providers. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops*, 113–118. IEEE.

Connolly, C. and Fleiss, T. (1997). A study of efficiency and accuracy in the transformation from rgb to cielab color space. *IEEE Transactions on Image Processing*, 6(7), 1046–1048.

Crow, F.C. (1977). The aliasing problem in computer-generated shaded images. *Communications of the ACM*, 20(11), 799–805.

Davis, L.W. (2008). The effect of driving restrictions on air quality in mexico city. *Journal of Political Economy*, 116(1), 38–81.

Google LLC (2019). Google maps platform billing. `https://developers.google.com/maps/billing/gmp-billing`. Accessed: 2019-11-03.

Google Maps (2019a). Distance matrix API. URL `https://developers.google.com/maps/documentation/distance-matrix/intro#duration_in_traffic`. Accessed: 2019-04-20.

Google Maps (2019b). Maps javascript API. URL `https://developers.google.com/maps/documentation/javascript/tutorial`. Accessed: 2019-11-03.

Gramaglia, M., Bernardos, C., and Calderon, M. (2013). Virtual induction loops based on cooperative vehicular communications. *Sensors*, 13(2), 1467–1476.

Kaur, A. and Kranthi, B. (2012). Comparison between ycbcr color space and cielab color space for skin color segmentation. *International Journal of Applied Information Systems*, 3(4), 30–33.

Landesplanungsbehörde NRW (2017). Landesentwicklungsplan nrw. Accessed: 2019-11-03.

Lenhart, D., Hinz, S., Leitloff, J., and Stilla, U. (2008). Automatic traffic monitoring based on aerial image sequences. *Pattern Recognition and Image Analysis*, 18(3), 400–405.

Mishra, R.K., Pandey, A., Pandey, G., and Kumar, A. (2019). The effect of odd-even driving scheme on pm2. 5 and pm1. 0 emission. *Transportation Research Part D: Transport and Environment*, 67, 541–552.

Monteiro, M., Stari, C., Cabeza, C., and Marti, A.C. (2019). Physics experiments using simultaneously more than one smartphone sensors. *arXiv preprint arXiv:1901.05068*.

Staacks, S., Hütz, S., Heinke, H., and Stampfer, C. (2018). Advanced tools for smartphone-based experiments: phyphox. *Physics Education*, 53(4), 045009.

The MathWorks Inc. (2018). Matlab and optimization toolbox, release 2018a. Natick, MA, USA.

Wagner, J.M.S., Scholz, S., and Gennat, M. (2020). *Verarbeitung, Visualisierung und Kalibrierung von Verkehrsdaten*, 477–487. Springer Fachmedien, Wiesbaden. doi: 10.1007/978-3-658-29746-6_39.

Wang, F. and Xu, Y. (2011). Estimating o–d travel time matrix by google maps API: implementation, advantages, and implications. *Annals of GIS*, 17(4), 199–209.