# Predicting bus arrivals when the feed reports positions but not delays: a reproducible three-layer method with an open reference implementation

Yeray Garrido[1]

**Abstract**

Many public transport networks publish real-time vehicle positions but not delays, and many timetables give intermediate stop times interpolated at constant speed rather than observed. Bizkaibus, the interurban bus network of Biscay (Spain), has both properties: after the first hour of a trip, 85% of buses pass a stop more than two minutes before the published time. This paper specifies the arrival prediction method of the Bizkaibus+ app in enough detail to reimplement it in any language. The method uses timetable differences instead of absolute times; learns the actual travel time between consecutive stops by time band, as a median shrunk towards the all-day median of the segment; and predicts from three layers with explicit abstention rules: the GPS position projected onto the route, the last stop registered by the feed, and the timetable corrected trip by trip. It recalibrates itself daily, adopting only changes that reduce the error on held-out data. In 9.1 million predictions over 23 days, checked against the actual passage, the arrival shown is within ±2 minutes in 92–96% of cases when the bus is less than three minutes away and in 67% at ten to twenty minutes, against 23–25% for the published timetable. Without any live data, the corrected timetable alone doubles the accuracy of the published one. A reference implementation in Python, test vectors with 407 checks produced by the production engine, a public evaluator that reproduces the production histograms, and the learned tables and evaluation data are openly available.

*Keywords:* bus arrival prediction, real-time passenger information, GTFS, SIRI, robust estimation, shrinkage, temporal validation, reproducibility

[1]Independent researcher, Bilbao, Biscay, Spain. ORCID: 0009-0008-0459-0294. E-mail: garridoparrayeraytx@gmail.com. Code and data: `https://github.com/Garridoparrayeray/bide-eta`.

*Preprint v1.1, November 2026. doi:10.5281/zenodo.24286302. Not peer reviewed.* *Licence CC BY 4.0.*

## 1. Introduction

Real-time arrival information reduces both the time passengers wait at stops and the uncertainty of that wait, and makes public transport more attractive (Watkins et al. 2011; Brakewood and Watkins 2019). Producing it requires two inputs that are not always available in the form a prediction method expects: a reliable reference schedule and a live estimate of how far each vehicle is from it.

Bizkaibus, the interurban bus network of the province of Biscay in northern Spain, publishes both as open data: a GTFS timetable (MobilityData 2026) and a SIRI Vehicle Monitoring feed (CEN 2022) with the GPS position and the next stop of every vehicle in service. The feed, however, does not report vehicle delays, and the intermediate times of the timetable are not operator commitments: they are interpolated linearly between the terminal times, at constant speed along the route. Real buses are slower in towns and faster on highways, so the published intermediate times drift further from the actual passage times as the trip progresses. Section 3.3 quantifies this drift: over 169,308 observed stop passages, after the first hour of a trip 85% of buses pass more than two minutes before the published time, and the median passage is almost fourteen minutes early. A passenger who trusted the published time and reached the stop two minutes beforehand would find that the bus had already gone in 52% of all observed passages.

This setting is not specific to Bizkaibus. Position-only feeds are common in automatic vehicle location systems, and interpolated intermediate times are allowed by GTFS (`timepoint=0`) and widespread in regional networks. Under these conditions, the absolute published time is a poor reference, and a method that predicts arrivals by adding a delay to it inherits its error.

This paper describes the method that the Bizkaibus+ app[2] uses to predict arrivals under these conditions. The aim is to make the method reusable: every rule is given with its formula and constants, the evaluation protocol is specified so that it can be repeated on other data, and the reference implementation, test vectors, evaluator and data are public. A longer technical report in Spanish (Garrido 2026) gives the measurements behind each constant, the history of each rule and the evaluation of the first period used here. The contributions are:

1. A representation of the vehicle state in *timetable time*: the GPS position

[2] `https://www.bideplus.eus`. The app is independent and is not affiliated with Bizkaibus, the Diputación Foral de Bizkaia or the operators of the service.

**The method in plain language.** The timetable says, for every stop, when the bus should pass; its intermediate times are wrong in absolute terms but consistent with each other. The method first asks *where in the timetable* the bus is: it projects the GPS position onto the route and reads the scheduled time at that point. A bus that is physically where the timetable puts it at 10:20 is, in timetable terms, at 10:20, whatever the clock says. To predict the arrival at a later stop, the method does not add a delay to the published time of that stop; it adds, to the time at which the GPS position was taken, how long the remaining segments *actually* take, as learned from weeks of observed passages. When the GPS position is missing or unreliable, it uses the last stop that the feed reports the bus has passed; when there is no live data at all, it uses the timetable corrected by what each trip usually does. Each prediction states which of these three sources it comes from.

is projected onto the route and converted into the scheduled time at that point, so that predictions are built from timetable differences, which are reliable even when absolute times are not (Sections 4.2 and 4.3).

2. A segment travel-time model learned from the feed itself, by time band and day type, with medians shrunk towards the all-day median of each segment, and a daily recalibration that adopts only changes that reduce the error on data not used to fit them: other vehicles, or later days for the per-trip correction (Sections 4.4 and 4.6).
3. A three-layer prediction with explicit abstention rules, which decides for each stop whether the vehicle position, the last registered stop or the corrected timetable gives the best available estimate, and states which one it used (Section 4.5).
4. An open, verifiable package: a reference implementation written only from the specification, test vectors with 407 checks produced by the production engine, an evaluator that reproduces the production histograms bin by bin on a 40-minute test capture, and the learned tables and evaluation data (Section 6).

The rest of the paper reviews related work (Section 2), describes the data and the problem (Section 3), presents the method (Section 4) and its evaluation (Section 5), explains how to reproduce and reuse the results (Section 6), discusses limitations and open problems (Section 7) and concludes (Section 8).

## 2. Related work

Bus arrival and travel-time prediction has an extensive literature, reviewed recently by Kumar et al. (2025) and, for models based on artificial intelligence,

by Singh and Kumar (2022). Moreira-Matias et al. (2015) survey the wider use of automatic vehicle location (AVL) data to evaluate and correct timetables. Approaches fall into four families: historical averages, regression and time-series models, Kalman filters, and machine learning; Section 5.3 compares the method with representatives of three of them.

*Historical and segment-based methods.* The simplest predictors average historical travel times by route, time of day or segment. Lin and Zeng (1999) compared GPS-based algorithms that use the schedule and time-check stops on a rural service with long headways, a setting close to that of interurban buses. Splitting a trip into segments between consecutive stops and predicting each segment is common: Gal et al. (2017) predict segment times with a queueing-based snapshot predictor and with machine learning, noting that outliers are a problem, and Mazloumi et al. (2010) show that travel-time variability depends strongly on the time window, which motivates learning by time band. The method presented here belongs to this family, with three distinctive choices: it learns per segment and time band rather than per line, it uses medians shrunk towards the all-day value of the segment, and the live position resets the prediction at each update.

*Kalman filters.* Kalman filtering (Kalman 1960) updates a state, typically the vehicle's position and speed or its delay, with each new observation. Cathey and Dailey (2003) proposed a tracker–filter–predictor framework for AVL data, Shalaby and Farhan (2004) combined AVL data with passenger counts, Vanajakshi et al. (2009) applied a Kalman filter to buses in heterogeneous traffic, and Achar et al. (2020) model the spatial correlation between successive sections. Section 5.3 compares the method with two Kalman filters under the same protocol.

*Machine learning.* Neural networks were applied to bus arrival prediction early on (Chien et al. 2002) and outperformed historical and regression models on a route in Houston (Jeong and Rilett 2005); support vector regression has been used with segment-level inputs similar to ours (Yu et al. 2006). Recent work uses recurrent networks (Pang et al. 2019), convolutional LSTMs that predict all the links of a route at once (Petersen et al. 2019), graph attention networks for sparse data (Ma et al. 2022) and transformers trained on open data (Jeong et al. 2024). In road-traffic forecasting, deep learning is not always the best technique and strong simple baselines matter (Manibardo et al. 2022); our comparison agrees in part: the simple method keeps most of the gain, although a neural network still improves on it by 6–7%.

*Open feeds, published timetables and published predictions.* Several studies use the open GTFS and real-time feeds that agencies publish. Wessel et al. (2017) reconstruct the timetable actually operated from a public vehicle-location feed, much as our ground truth is reconstructed from the live feed, and Wessel and Farber (2019) show that published timetables overstate accessibility compared with what buses actually do. The accuracy of predictions already shown to passengers has also been measured: Cats and Loutos (2016) evaluated a deployed prediction scheme in Stockholm and found that it underestimated the remaining wait, and García-Mauriño et al. (2024) correct the systematic errors of agency predictions in Madrid and Paris using only data from public APIs, a third-party setting close to ours. Real-time feeds also have quality problems of their own, such as latency (Zhou et al. 2025).

*Shrinkage and robust estimation.* Shrinking noisy estimates towards a common value reduces their mean squared error (James and Stein 1961; Efron and Morris 1975), and empirical Bayes estimation is standard in transport safety (Hauer et al. 2002). Woodard et al. (2017) model travel-time distributions with a trip-level effect and fall back to pooled parameters for links with few observations, a close analogue of our per-trip correction and shrinkage. Median-based statistics are robust to outliers and incidents (Huber 1964; Clark et al. 2002).

*Passenger value and evaluation.* Real-time information improves satisfaction and perceived safety, whereas inaccurate predictions reduce ridership (Gooze et al. 2013; Brakewood and Watkins 2019); the value of the information depends on the quality of the predictions (Liu and Miller 2020) and is valued especially on low-frequency services (Rahman et al. 2013), such as interurban buses. Evaluating forecasts requires keeping future data out of what the model learns from, through rolling-origin, out-of-sample evaluation (Tashman 2000), order-preserving validation for non-stationary series (Cerqueira et al. 2020) and strict separation of the data used for learning and for prediction (Kaufman et al. 2012). The day-by-day evaluation of new rules follows these principles; the daily recalibration validates on vehicles not used for fitting and, for the per-trip correction, on later days.

## 3. Data and problem setting

### *3.1. Timetable*

The timetable is the Bizkaibus GTFS feed published through the Spanish National Access Point for transport (Ministerio de Transportes y Movilidad

Sostenible 2026). It contains about 100 lines. Every line has an identifier (`route_id`, e.g. `651`) and a public name (`route_short_name`, e.g. `A0651`). Intermediate stops carry `timepoint=0`: their times are interpolated linearly with distance between the terminal times.

### *3.2. Live feed*

The SIRI Vehicle Monitoring feed is a public XML document regenerated about once a minute, with about 250 vehicles at peak times. For every vehicle it gives the trip reference (equal to a GTFS `trip_id`), the vehicle identifier, the GPS position and its time, and the order of the next stop in the trip. It does not report the delay. From 23:58 to 05:00 the feed repeats the vehicles seen before midnight with their old times; these entries carry no time of the new day and are ignored (Section 4.7).

### *3.3. Why the timetable is not an absolute reference*

Figure 1 shows the deviation of the actual passage from the published time over 169,308 observed stop passages, as a function of the scheduled time elapsed since departure from the terminal. Near the terminal, buses are slightly late (median +1.3 min in the first ten minutes). From there on they are increasingly early: half an hour after departure, half of the passages occur more than two minutes before the published time, and after one hour 85% do, the median passage being almost fourteen minutes early. Only a quarter of all passages fall within ±2 minutes of the published time. These figures are not a judgement on the punctuality of the service, since the intermediate times are not operator commitments; they are, however, the times that a passenger sees in any application that does not correct them.

## 4. Method

### *4.1. Notation and conventions*

A trip $J$ of line $\ell$ visits stops $p_1, \ldots, p_N$. Stop $p_i$ has order $i$, coordinates, published arrival and departure times $a_i \le d_i$ and, when the trip has a shape, a distance $s_i$ along it (`shape_dist_traveled`). Times are seconds since midnight (Europe/Madrid) of the trip's service day $D$ and, as in GTFS, may exceed 86,400 (26:10 is 94,200 s). Segment $i$ of $J$ is the pair of consecutive stops $\kappa_i = (p_i, p_{i+1})$; a segment $\kappa$ is identified by its two stop identifiers, so that all lines running it share what is learned about it. Its scheduled duration is $h_i = a_{i+1} - a_i$ and, when both distances are known, its length is $L_i = s_{i+1} - s_i$.

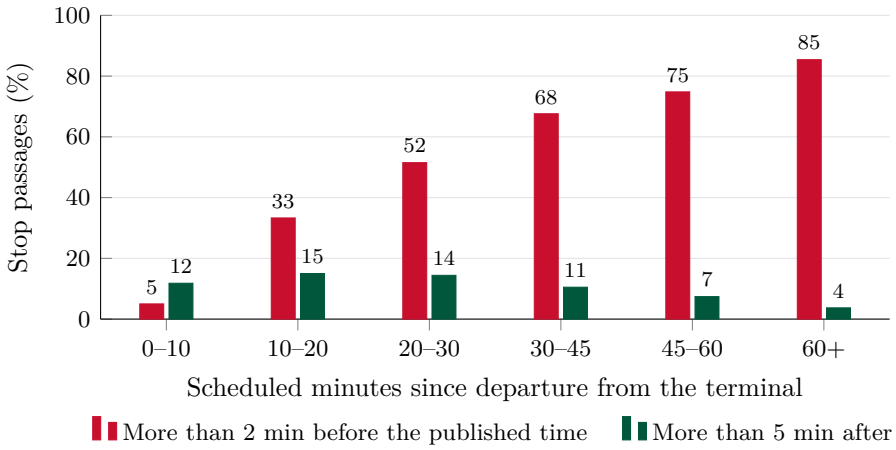


Figure 1: Deviation of the actual passage from the published time, by scheduled time since departure (169,308 stop passages). Because the timetable assumes constant speed and buses run faster on open-road sections, they pass increasingly early as the trip progresses.

Each entry $e$ of the live feed carries a trip reference, a GTFS `trip_id` that also encodes the line, the trip number and the departure $d^{\text{feed}}$ of the trip in seconds (`trp_A0651_737_ABC1_28500` gives line 651, trip number 737 and $d^{\text{feed}} = 28{,}500$; entries whose reference does not have this form are ignored); a vehicle identifier; and, when present, the order $n$ of the next stop (`Order`, or `VisitNumber`), a GPS position with its time $t_{\text{gps}}$ (`LocationRecordedAtTime`, or `RecordedAtTime` if absent) and the time $t_{\text{rec}}$ of the record (`RecordedAtTime`). Feed times are converted to local time; a time whose date is not the current day is discarded, and so is a position whose time is discarded. $t_{\text{now}}$ is the current time.

A *time band* $\text{band}(D, t)$ is one of twelve: the day type of $D$ (weekend on Saturday and Sunday, weekday otherwise) combined with one of six periods of the hour $\lfloor t/3600 \rfloor \bmod 24$, starting at 0, 6, 9, 13, 16 and 20 h; 26:10 on a Friday thus falls in the weekday band that starts at 0 h (the band follows the service day, not the clock date). Four conventions are needed to reproduce every number. rnd rounds halves away from zero, and $\text{rnd}_3(x) = \text{rnd}(1000x)/1000$. $Q_\alpha(x) = x_{(\lfloor \alpha(|x|-1) \rfloor)}$ is the percentile $\alpha$ of the sorted values $x_{(0)} \le \cdots \le x_{(|x|-1)}$, without interpolation, and $\text{med} = Q_{0.5}$ is the low median. $\text{clamp}(x, x_{\min}, x_{\max}) = \max(x_{\min}, \min(x_{\max}, x))$. Line numbers are written as in the timetable, without leading zeros, also in the keys of the learned tables: the feed writes `A0651` for line `651`. Appendix A lists every constant.

*4.2. Matching and localisation*

*Matching.* Calendar variants of the same trip are merged when the timetable database is built, and a table of aliases maps every original `trip_id` to its merged trip. A trip $J$ is matched to the first feed entry whose reference is an alias of $J$; on the live feed every vehicle was matched this way. Only if there is none is the fallback used: among the entries with the line and trip number of $J$ whose reference is not an alias of another trip (an entry matched exactly is never reused), the one whose $d^{\text{feed}}$ is closest to $d_1$ is taken, provided $|d^{\text{feed}} - d_1| \leq 180$ s (the first one on a tie). Otherwise the trip has no feed entry.

*Projection.* All localisation rests on projecting the GPS position onto a straight segment $AB$ in a local equirectangular plane centred on the vehicle. With $(\varphi, \lambda)$ the latitude and longitude of the vehicle in degrees, a point of latitude $\varphi'$ and longitude $\lambda'$ has planar coordinates $\big((\lambda' - \lambda)\cos\varphi \cdot 111{,}320,\ (\varphi' - \varphi) \cdot 111{,}320\big)$ in metres. With $\mathbf{a}$ and $\mathbf{b}$ the planar coordinates of $A$ and $B$ and $\mathbf{u} = \mathbf{b} - \mathbf{a}$,

$$f = \operatorname{clamp}\Big(-\frac{\mathbf{a}\cdot\mathbf{u}}{\|\mathbf{u}\|^2}, 0, 1\Big), \qquad d_\perp = \|\mathbf{a} + f\mathbf{u}\|, \tag{1}$$

with $f = 0$ if $A = B$: $f$ is the fraction of the segment from $A$ to the point closest to the vehicle, $d_\perp$ the distance to that point and $\|\mathbf{u}\|$ the length of the segment. The distance from the vehicle to a stop is computed in the same way, with the cosine of the stop's latitude.

*Search window.* The declared next stop bounds where the vehicle can be. The search covers segments $i_{\min}$ to $i_{\max}$ with

$$i_{\min} = \max(1, n-3), \qquad i_{\max} = \min(N-1, n+2), \tag{2}$$

that is, stops $p_{n-3}$ to $p_{n+3}$ where the trip has them. Looking back covers a feed that changes the next stop too early, while the bus is still before $p_{n-1}$; looking ahead covers one that changes it late, when the bus is already past $p_n$.

*With a shape.* This method is used when the trip has a shape and every stop of the window has a distance. Every pair of consecutive shape points $A, B$, with distances $s_A$ and $s_B$ along the shape, such that $s_B \geq s_{i_{\min}}$ and $s_A \leq s_{i_{\max}+1}$ gives a candidate position along the route, $s^* = s_A + f(s_B - s_A)$. Candidates with $s^* \leq s_n + 100$ m are *expected*, the rest *beyond*; each group keeps its candidate with the smallest $d_\perp$ (the first one on a tie). The expected

candidate wins unless there is none or the beyond one is more than 20 m closer, which resolves routes that pass twice along the same street in favour of the declared next stop. The position is valid if the chosen candidate has $d_\perp \le 50$ m. Then $s^*$ is clamped to $[s_{i_{\min}}, s_{i_{\max}+1}]$, $i$ is the first segment of the window with $s^* \le s_{i+1}$, and the scheduled time at the vehicle's position is

$$T_{\text{pos}} = \text{rnd}\left(d_i + f_i(a_{i+1} - d_i)\right), \qquad f_i = \text{clamp}\left(\frac{s^* - s_i}{s_{i+1} - s_i}, 0, 1\right), \tag{3}$$

with $f_i = 0$ if $s_{i+1} = s_i$. The time is interpolated from the departure at $p_i$, not the arrival, since within a segment the vehicle is moving.

*Without a shape.* Otherwise the vehicle is projected onto the straight segments $p_i p_{i+1}$ of the window and the closest one is kept (the first one on a tie). The position is valid if $d_\perp \le \max(150\text{ m}, 0.15\,\|\mathbf{u}\|)$, a corridor that widens with the straight-line length $\|\mathbf{u}\|$ of that segment, and then $T_{\text{pos}} = \text{rnd}\left(d_i + f(a_{i+1} - d_i)\right)$, with $f$ from Eq. (1). Because the timetable database measures stop distances on the shape when a network does not publish them, this case only remains for trips with no shape at all.

*Conditions.* A position requires the entry to have coordinates, $t_{\text{gps}}$ and $n$, the data to be recent ($t_{\text{now}} - t_{\text{gps}} \le 600$ s) and the trip to have at least two stops. If the trip has started or is about to ($t_{\text{gps}} \ge d_1 - 60$ s), it also requires $n$ to be the order of a stop of the trip, the projection to be valid and the GPS not to contradict the registered stop (the cross-check of Section 4.5). If $t_{\text{gps}} < d_1 - 60$ s, the only possible position is at the first stop: a vehicle within 150 m of $p_1$ and with $d_1 - t_{\text{gps}} \le 3{,}600$ s is placed there, with $T_{\text{pos}} = \min(a_1, d_1)$ and $t_{\text{gps}}$ replaced by $d_1$, because it has not travelled at all and will leave at its time at the earliest. A position at the first stop never gives a passage and is not learned from.

*Shapes that do not match their stops.* The method depends on the shape and the stop distances agreeing. In the Bizkaibus timetable of 3 November 2026, 55 of the 440 shapes used by trips were drawn from the last stop to the first (670 of 6,130 merged trips, on 25 lines), and in 1,417 trips some stop was more than 200 m from the point its distance indicated. The timetable database therefore reverses a shape (it reverses the order of the points and replaces each distance $s$ by $s_{\max} - s$) when, in most of its trips, the distance from the first stop to the last shape point plus that from the last stop to the first point is smaller than the distance from the first stop to the first point plus that from the last stop to the last point. If a shape was reversed, or

some stop of a trip is more than 100 m from the point its distance indicates, all the stop distances of the trip are measured again by projecting each stop onto the shape from the previous one, searching at most three times the straight-line distance between the two stops plus 500 m; where the shape passes twice near a stop, the first pass wins unless the second is 30 m closer. After the correction, 26 trips kept a stop more than 200 m away.

*4.3. Why timetable differences*

Let $T(s)$ be the time at which the timetable places the vehicle at the point $s$ of the route, $R(s)$ the real time and $\Delta(s) = R(s) - T(s)$ the offset. The vehicle was seen at $s_p$ at $t_{\text{gps}} = R(s_p)$, and the stop of interest is at $s_d > s_p$.

**Proposition 1.** *The published time* $\hat{R}_{\text{tt}} = T(s_d)$ *has error* $R(s_d) - \hat{R}_{\text{tt}} = \Delta(s_d)$. *The* difference *estimate* $\hat{R}_{\text{dif}} = t_{\text{gps}} + T(s_d) - T(s_p)$ *has error* $\Delta(s_d) - \Delta(s_p)$. *More generally, for any estimate* $\hat{E}$ *of the real travel time from* $s_p$ *to* $s_d$, $\hat{R} = t_{\text{gps}} + \hat{E}$ *has error* $\big(R(s_d) - R(s_p)\big) - \hat{E}$, *which does not depend on* $\Delta(s_p)$.

*Proof.* Since $t_{\text{gps}} = T(s_p) + \Delta(s_p)$, $R(s_d) - t_{\text{gps}} - T(s_d) + T(s_p) = \Delta(s_d) - \Delta(s_p)$; the last statement is $R(s_d) - t_{\text{gps}} - \hat{E} = \big(R(s_d) - R(s_p)\big) - \hat{E}$. □

Three consequences guide the method. The delay accumulated before $s_p$ does not matter: a bus twenty minutes late makes the published time wrong by twenty minutes and the difference estimate by almost nothing. The *profile* of the timetable does matter: with interpolated intermediate times, $T(s_d) - T(s_p)$ can be twice the real time on a highway, so it is replaced by a learned estimate of the real travel time. And the error grows with the number of segments between the vehicle and the stop, as each adds its own variation.

*4.4. Segment travel times*

*Samples.* Let $P_i$ be the real passage of a vehicle at stop $p_i$, reconstructed from its GPS positions as described in Section 4.6. Two consecutive passages $P_i < P_{i+1}$ of the same *track* (one vehicle on one trip on one date) give a segment sample: the real travel time $P_{i+1} - P_i$, rounded to 0.1 s, in the band $\text{band}(D, \lfloor P_i \rfloor)$ of the moment the vehicle entered the segment. For a segment $\kappa$ and a group $g$ (a band, or the group `all` of every band), each track contributes a single value (its last sample in file order), so that a bus seen many times does not weigh more than one seen once; $Y_{\kappa,g}$ is the list of these values and $n_{\kappa,g}$ the number of tracks.

*Shrunk medians.* A segment is considered only if $n_{\kappa,\mathtt{all}} \geq 2$. With $m_\kappa = \mathrm{med}(Y_{\kappa,\mathtt{all}})$ and, for every band $b$ with $n_{\kappa,b} \geq 2$, $m_{\kappa,b} = \mathrm{med}(Y_{\kappa,b})$, the learned time in band $b$ is

$$\tau_{\kappa,b} = \mathrm{rnd}\left(\frac{n_{\kappa,b}\, m_{\kappa,b} + n_0\, m_\kappa}{n_{\kappa,b} + n_0}\right), \qquad n_0 = 5. \tag{4}$$

The any-time value $\tau_{\kappa,\mathtt{all}} = \mathrm{rnd}(m_\kappa)$ is stored only if the segment has samples in at least two bands; otherwise it would be one band's value applied to bands with no measurement. The learned time of $\kappa$ in band $b$ is $\tau(\kappa, b) = \tau_{\kappa,b}$ if it is stored, otherwise $\tau_{\kappa,\mathtt{all}}$ if it is stored; $\kappa$ is *learned* in $b$ if $\tau(\kappa, b)$ is defined. Equation (4) has the form of the posterior mean of a hierarchical normal model (Gelman et al. 2013; Efron and Morris 1975), with $n_0$ in the role of the ratio of within-band to between-band variance; the method uses medians for robustness (Huber 1964) and fixes $n_0$ by validation.

**Proposition 2.** *Let $\hat{\eta}_b$ be an unbiased estimator of $\eta_b$ with variance $\sigma^2/n_b$, and $\mu$ a fixed value with $\eta_b - \mu = \beta$. The shrunk estimator $\tilde{\eta}_b = \omega\hat{\eta}_b + (1-\omega)\mu$, with $\omega = n_b/(n_b + n_0)$, has mean squared error $\omega^2\sigma^2/n_b + (1-\omega)^2\beta^2$, which is smaller than that of $\hat{\eta}_b$ whenever $\beta^2 < \sigma^2(2n_b + n_0)/(n_b\, n_0)$. The bound grows as $n_b$ decreases.*

*Proof.* $\tilde{\eta}_b - \eta_b = \omega(\hat{\eta}_b - \eta_b) - (1-\omega)\beta$, a zero-mean term with variance $\omega^2\sigma^2/n_b$ plus a constant. The inequality is $(1-\omega)^2\beta^2 < (1-\omega^2)\sigma^2/n_b$, i.e. $\beta^2 < \frac{1+\omega}{1-\omega}\frac{\sigma^2}{n_b}$, and $\frac{1+\omega}{1-\omega} = \frac{2n_b+n_0}{n_0}$. □

Shrinkage therefore helps bands with few tracks and barely changes bands with many, while too large an $n_0$ erases real differences between bands; Section 5 shows both effects. The proposition assumes an unbiased estimator and a fixed $\mu$, whereas the method uses low medians and estimates $\mu$ ($m_\kappa$) from data that include the band's own tracks: it is a heuristic justification, and the value of $n_0$ is decided by measurement.

*Length fallback.* A segment without a learned time is estimated from its length, because the error of an interpolated timetable depends mostly on it: short urban segments take about the scheduled time and long interurban ones much less. A length $L$ falls in bucket $c(L)$, the number of the cuts 300, 600, 1,000, 1,500 and 3,000 m that do not exceed $L$ (six buckets). Every segment sample with scheduled duration $h_i > 0$ and known length (at calibration, the low median of $s_{i+1} - s_i$ over the timetable trips that run the segment) gives a ratio of real to scheduled time. For each bucket $c$ and group $g$, $r_{c,g}$ is $\mathrm{rnd}_3$ of the low median of all these ratios (not one per track), and

is stored if there are at least 20. For an unlearned segment $i$ of the trip, $\tau^{\mathrm{L}}(h_i, L_i, b) = \mathrm{clamp}(r, 0.3, 1.5) \cdot h_i$, with $r = r_{c(L_i),b}$ if it is stored, otherwise $r_{c(L_i),\mathtt{all}}$; $\tau^{\mathrm{L}}$ is undefined if $L_i$ is unknown or not positive, or if neither ratio exists. On 31 October 2026 the ratio of the group `all` fell monotonically from 0.987 below 300 m to 0.534 at 3 km or more.

*Pace factor.* The last resort, and the only model when a network starts with no learned segments, is a pace factor by band and, with enough data, by line and band: $k(\ell, D, t) = \mathrm{clamp}(\hat{k}, 0.6, 1.2)$, where $\hat{k} = k_{\ell,b}$ if it exists, otherwise $k_b$ if it exists, otherwise 1, and $b = \mathrm{band}(D, t)$. A *pace sample* (Section 4.6) records, for a vehicle with a position and a later stop of its trip, the scheduled remaining time $H$ and the real remaining time $Z$. Each track contributes the ratio $\rho = \sum Z / \sum H$ over its pace samples and, for a set of $n_\rho$ tracks (those of a band, or of a line in a band),

$$k = \mathrm{rnd}_3\Big(\mathrm{clamp}\left(1 + (\tilde{\rho} - 1)\tfrac{n_\rho}{n_\rho + 10},\ 0.6,\ 1.2\right)\Big), \tag{5}$$

with $\tilde{\rho}$ the low median of their ratios: a shrinkage towards the timetable as it is. $k_b$ needs eight tracks, and $k_{\ell,b}$ 25 tracks and five different vehicles in the band.

*Expected time.* For segment $i$ of $J$, with band $b$ and pace factor $k$, the segment time $\tau_i$ is the first defined value of the cascade $\tau(\kappa_i, b)$, $\tau^{\mathrm{L}}(h_i, L_i, b)$, $k\, h_i$. The expected real time between two scheduled instants $t_0 \le t_1$ of the trip is

$$E(J, t_0, t_1, b, k) = \sum_{i \in I(t_0, t_1)} \max(0, w_i)\, \tau_i, \tag{6}$$

$$w_i = \frac{\min(a_{i+1}, t_1) - \max(a_i, t_0)}{a_{i+1} - a_i},$$

with $w_i = 1$ when $a_{i+1} = a_i$. $I(t_0, t_1)$ walks the segments in order, skipping those with $a_{i+1} \le t_0$ and stopping at the first with $a_i \ge t_1$. Within a segment the vehicle advances at the timetable's pace; between segments, at the learned one. If no segment table exists, $E = k\,(t_1 - t_0)$. The whole path uses a single band $b$ and pace factor $k$: in layers 1 and 2, those of the observation time; in layer 3, those of the midpoint between $d_1$ and the published time of the target stop (Section 4.5).

### *4.5. Three-layer prediction*

The prediction for a *row*, one departure of trip $J$ from a target stop of order $q$ and published time $a_{\mathrm{dest}}$, uses the most reliable source available and

Table 1: States of a prediction.

| State | Meaning |
|---|---|
| `live` | Layer 1: the GPS places the vehicle on the route, or at its first stop before departure; a delay can be asserted. |
| `estimated` | Layer 2 or 3: an estimate without live tracking; no delay is asserted. |
| `departed` | The vehicle has already passed the stop. |
| `likely_departed` | No usable vehicle data, and the corrected timetable places the passage more than ten minutes ago and at least ten minutes before the published time; or, with a position, the computed arrival passed more than three minutes ago without the feed or the GPS confirming the passage. |
| `scheduled` | No layer can support an estimate: the published time, labelled as such. |

says which one it used. Each layer abstains under explicit conditions, and when none can support an estimate the published time is shown as such. A row has a state (Table 1), an estimated arrival ETA and a delay $\delta$ in seconds. Only rows computed from a position (layer 1) assert a delay; the others carry $\delta = 0$. A delay of *none* means that no delay is asserted.

*Layer 1: GPS position.* With a position, $b = \text{band}(D, t_{\text{gps}})$, $k = k(\ell, D, t_{\text{gps}})$ and, if $a_{\text{dest}} \geq T_{\text{pos}}$,

$$\text{ETA} = \text{rnd}\left(t_{\text{gps}} + E(J, T_{\text{pos}}, a_{\text{dest}}, b, k)\right), \qquad \delta = \text{ETA} - a_{\text{dest}}. \tag{7}$$

If $a_{\text{dest}} < T_{\text{pos}}$, the timetable places the stop behind the vehicle: unless the row is `departed` (see *Passed stops* below), $\text{ETA} = t_{\text{gps}} - (T_{\text{pos}} - a_{\text{dest}})$ and $\delta = t_{\text{gps}} - T_{\text{pos}}$. The prediction starts from $t_{\text{gps}}$, not from $t_{\text{now}}$: the feed arrives about a minute after the position was taken, and adding the remaining time to $t_{\text{now}}$ would shift every arrival by that lag. A vehicle waiting at its first stop is treated after layer 3.

*Layer 2: registered stop.* Many entries carry the next stop and the time of the record but no usable position. The feed then says that the vehicle has left $p_{n-1}$ and is heading to $p_n$, at the reference time $t_{\text{ref}}$ ($t_{\text{gps}}$ if it exists, otherwise $t_{\text{rec}}$). The entry has a *registered stop* if $n \geq 2$, the trip has stops of orders $n-1$ and $n$, $t_{\text{now}} - t_{\text{ref}} \leq 600$ s, $t_{\text{ref}} \geq d_1 - 60$ s, and the segment is short ($a_n - a_{n-1} \leq 1{,}200$ s). The vehicle is then assumed to be half-way along it: with $b = \text{band}(D, t_{\text{ref}})$ and $k = k(\ell, D, t_{\text{ref}})$, for a target stop with

$q \geq n$,

$$\theta = \mathrm{rnd}\left(\tfrac{1}{2}E(J, a_{n-1}, a_n, b, k)\right), \qquad \mathrm{ETA} = \mathrm{rnd}\left(t_{\mathrm{ref}}+\theta+E(J, a_n, a_{\mathrm{dest}}, b, k)\right), \tag{8}$$

with the last term zero if $a_{\mathrm{dest}} \leq a_n$. Half is the expected remaining fraction if the observation instant is independent of the position; shares from 0.35 to 0.65 were tested and 0.5 was best. On longer segments the uncertainty of "half-way" is too large and the layer abstains. Layer-2 predictions are shown as `estimated`, without a delay; an arrival more than 120 s in the past is no longer credible, and the row goes to layer 3.

*Cross-check.* When an entry has both a position and a registered stop, their estimates of the offset should agree. With $\delta_{\mathrm{pos}} = t_{\mathrm{gps}} - T_{\mathrm{pos}}$ and $\delta_{\mathrm{reg}} = t_{\mathrm{ref}} + \theta - a_n$, if $a_n - a_{n-1} \leq 360$ s and $|\delta_{\mathrm{pos}} - \delta_{\mathrm{reg}}| > 360$ s, one of them is wrong and there is no way to know which: the entry loses both, and the rows of its trip go to layer 3. The position, with this check, is evaluated before any use of the registered stop.

*Layer 3: corrected timetable.* With no reliable vehicle data, the best estimate is a vehicle that leaves the first stop a little after its time and runs every segment in its learned time. Let $D'$ be the service day $D$, except on a weekday holiday (a weekday on which the timetable runs, by exception, a Sunday-only calendar), when $D'$ is the Sunday of that week; a holiday thus uses weekend bands, pace factors and per-trip corrections. If a segment table exists and $a_{\mathrm{dest}} > d_1$, with $b = \mathrm{band}(D', t_{\mathrm{mid}})$ and $k = k(\ell, D', t_{\mathrm{mid}})$ taken at the midpoint $t_{\mathrm{mid}} = \lfloor (d_1 + a_{\mathrm{dest}})/2 \rfloor$, the corrected timetable is

$$C = d_1 + 60 + \mathrm{rnd}\left(E(J, d_1 - 1, a_{\mathrm{dest}}, b, k)\right) + \mathrm{rnd}(\gamma_v), \tag{9}$$

where $\gamma_v$ is the per-trip correction defined below (0 without a correction table). The 60 s reflect that buses leave the first stop slightly late, and starting at $d_1 - 1$ includes segments of zero scheduled duration at the start of the trip. If $C \geq t_{\mathrm{now}}$, the row is `estimated` with ETA $= C$. If $C < t_{\mathrm{now}} - 600$ s, $a_{\mathrm{dest}} - C \geq 600$ s and the segments of $I(d_1 - 1, a_{\mathrm{dest}})$ learned in band $b$ make up at least 80% of the total scheduled duration of that path, the corrected timetable says with confidence that the bus passed more than ten minutes ago, and at least ten minutes before its published time: the row is `likely_departed`, shown with ETA $= a_{\mathrm{dest}}$ at the end of the list. Otherwise, and whenever there is no segment table or $a_{\mathrm{dest}} \leq d_1$, the row is `scheduled`, with ETA $= a_{\mathrm{dest}}$.

*Per-trip correction.* Without $\gamma_v$, the corrected timetable arrived on average 131 s before the bus: the sum of segment medians underestimates the run, because segment times are right-skewed, and some trips leave late or lose time at the same place almost every day. A trip is recognised by the key $v = (\ell, p_1, p_N, d_1, N)$ (line, first and last stop, departure and number of stops), which does not change when the operator republishes the same trips with new identifiers. Let $\chi$ be the day type of $D'$, $z = \min(2, \lfloor 3(q-1)/N \rfloor)$ the third of the route in which the target stop lies, and $b_1 = \text{band}(D', d_1)$ the band of the departure. The correction is looked up from the finest to the most general level: $\gamma_v = \gamma_{v,\chi,z}$ if the table has values for trip $v$ and day type $\chi$; otherwise $\gamma_{\ell,b_1}$; otherwise $\gamma_{b_1}$; otherwise 0. The tables are fitted on the deviations $\varepsilon$ of the real passages from the corrected timetable without correction ($C$ with $\gamma_v = 0$, on the $D'$ of each day), each clipped to $\pm 900$ s; the real passages are those of the segment samples, omitting the first stop, stops repeated within a trip and passages more than 45 minutes from their published time. Then

$$\gamma_b = Q_\alpha(\varepsilon_b), \qquad \gamma_{\ell,b} = \frac{n_{\ell,b}\, Q_\alpha(\varepsilon_{\ell,b}) + 20\,\gamma_b}{n_{\ell,b} + 20},$$
$$\gamma_{v,\chi,z} = \frac{\sum_{\text{days}} Q_\alpha(\varepsilon_{v,\chi,z,\text{day}}) + \gamma_{\ell,b}}{n_{\text{days}} + 1}, \tag{10}$$

where $\varepsilon_b$ are the deviations of trips departing in band $b$, $\varepsilon_{\ell,b}$ those of line $\ell$ in band $b$ and $n_{\ell,b}$ their number, and $\varepsilon_{v,\chi,z,\text{day}}$ those of trip $v$ in third $z$ on one of the $n_{\text{days}}$ days with data. In the last formula $\gamma_{\ell,b}$ is that of the trip's line and departure band, and a third with no deviations takes $\gamma_{\ell,b}$. Stored values are rounded to whole seconds; the values combined inside the formulas are not. A percentile is used instead of the mean because deviations have a long tail of delays, which pulls the mean above the typical value, and a shown time slightly later than the passage makes passengers miss the bus. The percentile starts at $\alpha = 0.40$, the value in the reference implementation, and is chosen every night among 0.30, 0.35, 0.40, 0.45 and 0.50 (Section 4.6). The correction enters only through $C$: layers 1 and 2 start from where the vehicle is and already reflect what happens to it that day.

*Waiting at the first stop.* A vehicle placed at its first stop before departure (Section 4.2) does not leave exactly on time, so its arrival is that of layer 3: if a segment table exists and $a_{\text{dest}} > d_1$, ETA $= \max(C, t_{\text{now}})$ and $\delta = \text{ETA} - a_{\text{dest}}$; otherwise the layer-1 formulas apply with $t_{\text{gps}} = d_1$. Its state is `live`, because the vehicle is known to be there.

*Passed stops.* A row is `departed`, with ETA $= \min(t_{\text{pass}}, t_{\text{now}})$ and $\delta = 0$, when the vehicle has already passed the stop at $t_{\text{pass}}$. With a position, if $a_{\text{dest}} < T_{\text{pos}}$ and either the feed has left the stop behind ($q < n$) or the GPS is more than 150 m from the stop (a stop without coordinates counts as farther), $t_{\text{pass}} = t_{\text{gps}} - (T_{\text{pos}} - a_{\text{dest}})$; otherwise, if $q < n$, $t_{\text{pass}} = t_{\text{gps}}$. Without a position, if the registered stop has $q < n$, $t_{\text{pass}} = t_{\text{ref}} - \max(0, a_{n-1} - a_{\text{dest}})$. A position at the first stop never gives a passage, even when the feed already declares the second stop as next ($n = 2$) while the vehicle waits.

*Arriving without a confirmed passage.* With a position, a computed arrival already in the past is shown as ETA $= t_{\text{now}}$ ("now"), with $\delta$ raised to at least $t_{\text{now}} - a_{\text{dest}}$, until the feed or the GPS confirms the passage. Once the computed arrival is more than 180 s in the past, the row becomes `likely_departed`, still with ETA $= t_{\text{now}}$, instead of the vehicle disappearing from the list before arriving.

*Implausible delays.* An offset of more than 45 minutes either way ($|\delta| >$ 2,700 s) almost always reveals a wrong match rather than a real delay (real delays reach about 37 minutes): $\delta$ becomes none, so no delay is asserted, but the arrival is kept, because it still comes from the vehicle's own data.

*Passage log.* A trip that has ended, or has dropped out of the feed, has no entry, and its rows would go to layer 3 as if the bus had not left. Every minute, therefore, an external process stores for each trip reference in the feed the last next-stop order $n_{\log}$ and its time $t_{\log}$ ($t_{\text{rec}}$, or $t_{\text{gps}}$ if there is none), under the service day of the trip. The log is consulted only for a trip with no exactly matched feed entry: of the records of all its references (a merged trip may have several), the one with the highest $n_{\log}$ is kept, and a row with $q < n_{\log}$ is `departed`, with ETA $= \min(t_{\log}, t_{\text{now}})$. This check precedes layer 3.

*Delay shown.* In the list of a stop, only a row computed from a position (`live`, or `likely_departed` while arriving) asserts a delay: $\text{rnd}(\delta/60)$ minutes, or none when $\delta$ is none. The card of a vehicle shows the delay at its next stop: with a position, the $\delta$ of Eq. (7) (or of the rule for a vehicle waiting at its first stop) for $p_n$ with published time $a_n$, as $\text{rnd}(\delta/60)$ minutes, or nothing if $|\delta| >$ 2,700 s. Without a position, the engine still computes $\delta_{\text{reg}}$ (0 without a registered stop) but does not assert it: the card, like the list, is shown as estimated and without a delay.

Algorithm 1 gives the complete decision for a row in the order in which it is applied.

**Algorithm 1** State and arrival of a row.

**Require:** row $(J, q, a_{\text{dest}})$, feed entry $e$ matched to $J$ (or none), $t_{\text{now}}$
**Ensure:** (state, ETA, $\delta$)

*pos* $\leftarrow$ position of $e$ on $J$, after the cross-check ▷ Sections 4.2, 4.5
*reg* $\leftarrow$ registered stop of $e$ on $J$
**if** no $e$ **or** (no *pos* **and** no *reg*) **then return** NoData$(J, q, a_{\text{dest}})$
**end if**
**if** passed the stop at $t_{\text{pass}}$ **then return** $(\texttt{departed}, \min(t_{\text{pass}}, t_{\text{now}}), 0)$
**end if**
**if** no *pos* **and** no layer-2 arrival for $q$ **then return** NoData$(J, q, a_{\text{dest}})$
**end if**
**if** *pos* exists **then** (ETA, $\delta$) $\leftarrow$ layer 1 (or waiting at the first stop); $\pi \leftarrow$ true
**else** ETA $\leftarrow$ layer 2, Eq. (8); $\delta \leftarrow$ ETA $- a_{\text{dest}}$; $\pi \leftarrow$ false
**end if**
**if** $|\delta| > 2{,}700$ s **then** $\delta \leftarrow$ none ▷ implausible delay: keep the arrival
**end if**
**if not** $\pi$ **then**
    **if** ETA $< t_{\text{now}} - 120$ **then return** NoData$(J, q, a_{\text{dest}})$
    **end if**
    **return** $(\texttt{estimated}, \text{ETA}, 0)$
**end if**
**if** ETA $< t_{\text{now}}$ **then** ▷ arriving, passage not confirmed
    **if** $\delta$ is not none **then** $\delta \leftarrow \max(\delta, t_{\text{now}} - a_{\text{dest}})$
    **end if**
    **if** $t_{\text{now}} -$ ETA $> 180$ **then return** $(\texttt{likely_departed}, t_{\text{now}}, \delta)$
    **end if**
    **return** $(\texttt{live}, t_{\text{now}}, \delta)$
**end if**
**return** $(\texttt{live}, \text{ETA}, \delta)$
**function** NoData$(J, q, a_{\text{dest}})$
    **if** $J$ has no exactly matched entry **and** the passage log gives $q < n_{\log}$ **then**
        **return** $(\texttt{departed}, \min(t_{\log}, t_{\text{now}}), 0)$
    **end if**
    **return** layer 3, Eq. (9): `estimated`, `likely_departed` or `scheduled`
**end function**

*4.6. Learning and daily recalibration*

*Capture.* The feed is downloaded every 20 s and every distinct version (by its `RecordedAtTime`) is stored, in one-hour blocks. Each snapshot is processed with $t_{\text{now}}$ equal to its time and its date as the service day, with the current day's matcher only, so the part of a trip after midnight is not learned. For each snapshot and vehicle with a position, the point $(t_{\text{gps}}, T_{\text{pos}})$ is appended to the vehicle's track on its trip. Only valid GPS positions are learned from: layer-2 and layer-3 estimates never are, nor are positions at the first stop, which describe no movement.

*Real passage.* The feed has no passage events. The passage at a stop with published time $a$ is reconstructed from the track, using the timetable as a distance scale. Let $t_{\text{gps},j}$ and $T_{\text{pos},j}$ be the $t_{\text{gps}}$ and $T_{\text{pos}}$ of the $j$-th point of the track. For the first pair of consecutive points with $T_{\text{pos},j-1} \le a \le T_{\text{pos},j}$, $T_{\text{pos},j} > T_{\text{pos},j-1}$ and $t_{\text{gps},j} - t_{\text{gps},j-1} \le 300$ s,

$$P(a) = t_{\text{gps},j-1} + \frac{a - T_{\text{pos},j-1}}{T_{\text{pos},j} - T_{\text{pos},j-1}} \left(t_{\text{gps},j} - t_{\text{gps},j-1}\right); \tag{11}$$

if there is no such pair, the passage is unknown. The real passage at stop $p_i$ is $P_i = P(a_i)$.

*Pace samples.* In every snapshot, each vehicle with a position predicts every stop of its trip with $a_{\text{dest}} > T_{\text{pos}}$. Once the tracks are complete, the prediction becomes a pace sample, in the band of $t_{\text{gps}}$, with $H = a_{\text{dest}} - T_{\text{pos}}$ and $Z = P(a_{\text{dest}}) - t_{\text{gps}}$, if $P(a_{\text{dest}})$ is known, $H \ge 120$ s, $Z > 0$, and the observation window (the run of snapshots with no gap of more than 300 s) extends at least ten minutes after $a_{\text{dest}}$, so that slow buses are not censored. A track with $M > 30$ samples keeps those at positions $\lfloor iM/30 \rfloor$, $i = 0, \ldots, 29$, so that a bus seen for a long time does not weigh more than others.

*Retention.* Samples older than 120 days are discarded, each segment keeps its 300 most recent samples of each day type, and at most the 500,000 most recent pace samples are kept. Keeping the most recent data lets a diversion or road works enter what is learned within one or two days.

*Recalibration.* Every night the pace factors, then the segment times and length ratios, then the per-trip corrections are re-fitted from all accumulated samples, in this order, because each validation uses the tables just written. Each new table is fitted on all the data, but it is written only if versions fitted without the test data predict those data better than the method without the

table, and only if it differs from the current one. For pace factors and segment times, validation is a two-fold split by track, with fold crc32(track) mod 2 (IEEE 802.3 CRC of the UTF-8 track key, taken as unsigned); each fold is predicted with the tables fitted on the other. The pace factors are written if, on the test samples with $k \neq 1$, the mean of $|kH - Z|$ is below that of $|H - Z|$. The segment times are written if, on the test samples of segments learned in their band, the sum of $|\tau(\kappa_i, b) - (P_{i+1} - P_i)|$ is below that of $|k\,h_i - (P_{i+1} - P_i)|$; otherwise nothing is written. The length ratios are included only if, on the test samples where $\tau^{\mathrm{L}}$ is defined, they lower the same sum. The per-trip correction, which is specific to each trip, is validated by days: for each candidate $\alpha$, the tables are fitted on all days but the last seven and tested on those seven. The $\alpha$ with the highest share of test deviations (unclipped) within $\pm 2$ minutes after correction wins, ties going to the lower mean absolute error and then to the first candidate in the list. The table fitted on all days with that $\alpha$ is written only if, on the test days, the winning $\alpha$ gives a lower mean absolute error than no correction. A day with anomalous data therefore changes the model only if the change improves the prediction of buses not used to learn it. To decide whether a new *rule* enters the method, a stricter day-by-day protocol is used (Section 5).

### *4.7. Practical issues*

*Trips past midnight.* A trip listed at 26:10 passes at 02:10 the next day, but the feed gives clock times of the current day, and the reference of a daily trip is the same every day. An entry therefore belongs to yesterday's trip if its departure $d^{\mathrm{feed}}$ lies more than 43,200 s (twelve hours) ahead of $t_{\mathrm{now}}$. Those entries go to a separate matcher in which $t_{\mathrm{gps}}$, $t_{\mathrm{rec}}$ and $t_{\mathrm{now}}$ are increased by 86,400 s, the service day $D$ is yesterday and the passage log is yesterday's; the current day's matcher receives only the other entries. The computation is otherwise unchanged.

*Frozen feed.* From 23:58 to 05:00 the Bizkaibus feed repeats the vehicles seen before midnight with their old times. Entries with neither $t_{\mathrm{gps}}$ nor $t_{\mathrm{rec}}$ of the current day are ignored, so night buses are shown with layer 3 or the timetable instead of as ghost vehicles.

*Missing readings.* The feed sometimes omits a vehicle for one or a few readings while it is still on its trip (about 700 times a day for a single reading). Each new reading is therefore completed with the last entry of every vehicle of the previous reading that is missing from it, if that entry has a vehicle identifier and its time ($t_{\mathrm{gps}}$, or $t_{\mathrm{rec}}$ without a position) lies between

0 and 180 s before $t_{\text{now}}$, the vehicle does not appear on another trip, and no other vehicle carries its trip (same line and trip number) in the new reading. The entry is used as it is, and layer 1 starts from its $t_{\text{gps}}$. On 2–4 November 2026, for those buses and arrivals less than five minutes away, this raised the share within ±2 minutes from 54.5% to 77.5%.

## 5. Evaluation

### *5.1. Protocol*

The evaluation replays captures of the live feed, which is polled every 20 s; each new version of the feed (about one per minute) is kept as a snapshot, and snapshots are stored in one-hour blocks. In each snapshot, for every vehicle and every stop from its declared next stop $n$ to $n + 8$, the method predicts the arrival $t_{\text{shown}}$ that a user would see, and the published time $a$ is kept for comparison. The error is

$$e = \frac{t_{\text{shown}} - t_{\text{actual}}}{60} \text{ minutes}, \tag{12}$$

positive when the bus arrives before the time shown.

*Ground truth.* The feed has no passage events, so the actual passage is reconstructed from changes in the declared next stop: if, in snapshot $m$, a vehicle still on the same trip declares for the first time a next stop beyond $q$, it passed stop $q$ between snapshots $m - 1$ and $m$, and

$$t_{\text{actual}} = \frac{t_{m-1} + t_m}{2}, \tag{13}$$

where $t_m$ is the most recent position time in snapshot $m$. The resolution is half the interval between snapshots, about ±30 s, and the measured error includes this uncertainty. The reconstruction does not use GPS positions and is therefore independent of the projection used by layer 1; it is not independent of layer 2, which uses the same changes of next stop. Only vehicles present in the feed whose passage is later observed are evaluated, so trips without a vehicle, shown with layer 3, are not.

*Metrics.* Predictions are grouped by horizon, i.e. the time actually remaining until the arrival (0–3, 3–6, 6–10 and 10–20 minutes), because accuracy depends mainly on it. The main metric is the share of predictions within ±1, ±2 and ±3 minutes, which answers the passenger's question ("will I make it if I leave when it says?") and, unlike the mean squared error, is not dominated by a few large errors. Errors are stored as histograms of 15-second bins, so that any figure can be recomputed by summing the histograms of the captures involved.

*Modes.* In addition to the full method, two ablations replay the same captures: *without GPS*, where all positions are deleted and only the stop registered by the feed remains, and *without feed*, where the method sees no live data and can only use the corrected timetable.

*Data.* Two periods are evaluated, which overlap on 31 October (23 distinct days). From 15 to 31 October 2026, the evaluation stored a summary of each capture block (17 days with predictions, 2.21 million predictions within 20 minutes). From 31 October to 6 November, it stored the full histograms (7 days, 6.87 million predictions within 20 minutes in each mode), and the raw captures were archived, so that this period can be re-evaluated with any version of the method. The method was updated during both periods; the figures describe it as it ran in production. The technical report (Garrido 2026) covers the first period; the second period, the ablations and the breakdowns by layer and day type are new in this paper and are recomputed from the public data by `articulo-metodo/resultados.py`.

### *5.2. Results*

Table 2 gives the accuracy by horizon. In both periods, the arrival shown is within ±2 minutes in 92–96% of cases below three minutes and in 67% at ten to twenty minutes, while the published timetable stays at 23–25% at every horizon. The median absolute error grows from 30 s below three minutes to 90 s at ten to twenty minutes (second period, at the 15-second resolution of the histograms).

*What each source of information contributes.* Table 3 compares the full method with the two ablations over the second period. Deleting the GPS positions costs little: the stop registered by the feed is updated often enough that accuracy below three minutes is unchanged, and at ten to twenty minutes it drops from 67% to 64%. Without any live data, the corrected timetable reaches 50–54% within ±2 minutes at every horizon, twice the share of the published timetable: between 38% and two thirds of the gain over the timetable, depending on the horizon, comes from learning how long the segments really take, before any live data is used.

*By layer.* In the second period, layer 1 (GPS position) produced 85–94% of the predictions, depending on the horizon, with 97%, 92%, 84% and 68% within ±2 minutes from the shortest to the longest horizon. Layer 2 (registered stop) produced about 3% of the predictions, and layer 3 (corrected timetable) between 3% and 13%. Layer-3 predictions below three minutes are rarely accurate (6%): they correspond to buses that should be arriving

Table 2: Share (%) of predicted arrivals within ±1, ±2 and ±3 minutes of the actual passage, by horizon.

| | | Method | | | Published timetable | | |
|---|---|---|---|---|---|---|---|
| Horizon (min) | Predictions | ±1 | ±2 | ±3 | ±1 | ±2 | ±3 |
| *15–31 October 2026 (17 days)* | | | | | | | |
| 0–3 | 534,046 | 84 | 96 | 98 | 13 | 24 | 34 |
| 3–6 | 485,601 | 67 | 90 | 96 | 13 | 24 | 34 |
| 6–10 | 548,058 | 53 | 82 | 92 | 12 | 23 | 33 |
| 10–20 | 640,763 | 39 | 67 | 82 | 13 | 25 | 35 |
| *31 October–6 November 2026 (7 days)* | | | | | | | |
| 0–3 | 1,485,997 | 81 | 92 | 94 | 12 | 23 | 33 |
| 3–6 | 1,371,035 | 67 | 89 | 95 | 13 | 24 | 34 |
| 6–10 | 1,673,835 | 54 | 82 | 92 | 12 | 23 | 33 |
| 10–20 | 2,339,535 | 39 | 67 | 82 | 13 | 25 | 35 |

Table 3: Share (%) of predicted arrivals within ±2 minutes with all the information, without GPS positions and without any live data, compared with the published timetable (31 October–6 November 2026).

| Horizon (min) | Full method | Without GPS | Without feed | Published timetable |
|---|---|---|---|---|
| 0–3 | 92 | 93 | 50 | 23 |
| 3–6 | 89 | 87 | 54 | 24 |
| 6–10 | 82 | 79 | 53 | 23 |
| 10–20 | 67 | 64 | 53 | 25 |

but send no usable data. This probably contributes to the lower accuracy below three minutes in the second period than in the first, for which no breakdown by layer and horizon is available.

*By day type.* Accuracy is similar on weekdays and at weekends (92% for both below three minutes; 67% and 65%, respectively, at ten to twenty minutes).

### *5.3. Day-by-day evaluation of the components*

Rules enter the method only after a day-by-day evaluation that reproduces how they will be used: for each of seven test days, everything is learned from the previous days only (exceptions are noted below), and the prediction is evaluated end to end on runs of consecutive segments (the arrival shown to the user is built from such sums, and a change can improve each segment without improving the sum). The measure concerns sums of segment times,

Table 4: Day-by-day evaluation of the shrinkage towards the all-day median of the segment: mean absolute error of the sum of segments and share within ±120 s.

| | | Mean abs. error (s) | | | Within ±120 s (%) | | |
|---|---|---|---|---|---|---|---|
| Horizon | Runs | $n_0 = 0$ | $n_0 = 5$ | $n_0 = 10$ | $n_0 = 0$ | $n_0 = 5$ | $n_0 = 10$ |
| 4–8 min | 107,052 | 42.8 | **42.0** | 42.3 | 96.0 | **96.2** | 96.0 |
| 8–12 min | 83,336 | 62.7 | **61.7** | 62.4 | 87.4 | **87.9** | 87.5 |
| 12–20 min | 113,581 | 89.3 | **88.2** | 89.4 | 73.6 | **74.1** | 73.4 |

not the full arrival shown. A rule is adopted if it improves accuracy at every horizon and on almost every day.

*Shrinkage.* Table 4 compares the band median alone ($n_0 = 0$) with the shrunk value of Eq. (4) on 195,786 segment samples over 26 days (test days 25–31 October 2026). With $n_0 = 5$, every measure improves at every horizon. With $n_0 = 10$, the gain holds at short horizons, but at 12–20 minutes the result is worse than without shrinkage: too much shrinkage erases real differences between bands, and these accumulate over many segments, consistent with Proposition 2. The day-by-day tool falls back to the all-day median when a band has fewer than two tracks, whereas production keeps a band only if the segment has samples in two bands; the difference concerns rarely observed segments.

*Per-trip correction.* On 87,333 test passages (26 October–1 November 2026), the corrected timetable without the per-trip correction, using no live data, had a mean absolute error of 210 s, against 438 s for the published timetable, and its times were on average 131 s earlier than the actual passage. With a correction by trip and third of the route, the mean absolute error fell to 158 s and the share within ±2 minutes rose from 47% to 54%, whereas a single correction per band lowered the mean error but worsened the hit rates at tight tolerances, because the bias is not the same for every line and trip. Splitting the route into fifths, or down to individual stops, did not help. The segment times of layer 3 were learned from all 27 days, test days included, so these absolute errors are in-sample with respect to segment times; this probably favours the uncorrected row. Replacing the mean of the deviations with their 40th percentile, evaluated by reconstructing what the app would have shown on two days not used for learning, raised the share within ±2 minutes by 4.1 and 2.5 percentage points and cut by a third the cases in which the bus passed more than two minutes before the time shown. The 35th percentile performs almost the same (0.4 points better on one day

and 0.2 worse on the other); 0.40 is the starting value, and the recalibration chooses each night, among 0.30 to 0.50, the percentile with the best share within ±2 minutes on the test days (ties go to the lower mean error).

*Comparison with other methods.* Table 5 compares the method, under the same protocol and on the same runs, with alternatives, most of which also use same-day data: a single scale factor fitted on earlier days; the recent ratio of the last buses on each segment; a Kalman filter per segment in the spirit of Shalaby and Farhan (2004); a Kalman filter on the vehicle's own pace; and a neural network with 28 inputs and two hidden layers of 32 and 16 units, in the spirit of Chien et al. (2002) and Jeong and Rilett (2005), trained for each test day on the previous days, also shown without its same-day inputs. The competitors' hyper-parameters were tuned on 18–24 October; the method's $n_0$ was chosen when the rule was adopted, on data overlapping the test days, which if anything favours the method. The method already captures almost all of the gain over the published timetable: at 4–8 minutes, the mean absolute error falls from 154 s to 42 s. The alternatives that use same-day data improve on it slightly, and on almost every day: the buses that have just run the same segments, used directly or through a Kalman filter, by up to 2%; the vehicle's own pace by about 4%; and the neural network, which combines all these inputs, by 6–7%. The network is the only alternative that is better on all seven days at all three horizons (the 95% bootstrap intervals of the difference exclude zero). Its gain is stable across random seeds (81.7 and 81.4 s at 12–20 minutes), whereas that of the network without same-day inputs is not (82.5 and 85.3 s); even so, with both seeds this network, which uses the same information as the method, improves on it by 3–7% at 12–20 minutes (88.2 s). The method is kept because it can be audited by hand and runs without a trained model, not because it is the most accurate. These comparisons concern sums of segment times, not the full arrival shown to the user.

## 6. Reproducing and reusing the method

All the material needed to verify the results and to reuse the method is in the public repository `https://github.com/Garridoparrayeray/bide-eta`, under the MIT licence for code and CC BY 4.0 for text and data.

- `referencia/referencia.py` implements the prediction method and part of the calibration (segment times, pace shrinkage, per-trip correction and the choice of the percentile) in Python using only the standard

Table 5: Comparison with other methods under the day-by-day protocol (test days 25–31 October 2026): mean absolute error of the sum of segments, share within ±120 s at 12–20 minutes, and number of test days (out of seven) on which each alternative beats the method, at each of the three horizons in turn.

| | Mean abs. error (s) | | | Within | Days |
|---|---|---|---|---|---|
| Method | 4–8 min | 8–12 min | 12–20 min | ±120 s (%) | better |
| Published timetable | 153.9 | 233.5 | 329.7 | 24.0 | 0 |
| **This method** | **42.0** | **61.7** | **88.2** | **74.1** | — |
| Scaled by one factor | 42.6 | 61.9 | 86.9 | 74.9 | 2 · 4 · 5 |
| Previous buses | 41.6 | 60.7 | 86.5 | 75.0 | 5 · 7 · 7 |
| Kalman filter per segment | 41.7 | 60.7 | 86.4 | 75.0 | 5 · 7 · 7 |
| Kalman filter on own pace | 40.4 | 59.3 | 84.6 | 76.0 | 6 · 7 · 7 |
| Neural network, no same-day data | 40.7 | 58.8 | 82.5 | 77.2 | 6 · 6 · 6 |
| Neural network | 39.5 | 57.5 | 81.7 | 77.4 | 7 · 7 · 7 |

library. It was written from the specification, not from the production code.

- `referencia/vectores.json` contains test vectors with 407 checks: concrete inputs with the exact outputs of the production engine, covering feed parsing, the matching of each vehicle to its trip, vehicles missing from a single download, trips past midnight, the projection, the three layers, the segment model, the pace factor, the per-trip correction, the passage log, the delay shown, the calibration and the nightly choice of the percentile. `referencia/comprobar.py` runs the reference implementation on all of them and reproduces all 407 outputs.

- `referencia/evaluar.py` replays captures of the live feed and writes the same 15-second histograms as the production evaluation. On 40 minutes of the live feed from 7 November 2026 (92,686 predictions), its histograms are identical, bin by bin, to those of the production evaluator in all three modes.

- `memoria/` holds the technical report (Garrido 2026), which describes the same method in more detail.

- `datos/` holds the learned tables (segment times, pace factors and per-trip corrections) and the evaluation data behind Tables 2 and 3 and the breakdowns by layer and day type, which

`articulo-metodo/resultados.py` recomputes; the day-by-day evaluation of Section 5.3 is documented in the technical report.

Three conventions must be preserved when porting the method to another language: times are seconds since midnight of the service day and may exceed 86,400; halves are rounded away from zero; and medians and other percentiles are taken without interpolation, as the element $\lfloor \alpha(|x| - 1) \rfloor$ of the $|x|$ sorted values (the low median for $\alpha = 0.5$).

## 7. Discussion and limitations

### *7.1. Where the accuracy comes from*

Three observations help to separate the contributions. First, knowing which segment the vehicle is in carries most of the information: deleting every GPS position, so that only the stop registered by the feed remains, costs nothing below three minutes and three percentage points at ten to twenty minutes (Table 3). What matters, as Proposition 1 anticipates, is to start from where the vehicle is rather than from the published time. Second, correcting the shape of the timetable already improves on it substantially: without any live data, the corrected timetable doubles the hit rate of the published one. Third, the coverage of learned segments is what changes accuracy the most: in the sample of the technical report (Garrido 2026), arrivals less than ten minutes away were within ±2 minutes in 93% of cases when the whole path was learned, in 76% with one unlearned segment and in 59% with two or more. The remaining error is limited by the data themselves (one position per minute and a ground truth known to within ±30 s) and by what no model can foresee from them: traffic lights, traffic conditions and dwell times.

A neural network that also uses same-day data reduces the error of segment sums by 6–7%, and a two-parameter Kalman filter on the vehicle's own pace reduces it by 4%. The method nevertheless keeps per-segment medians, shrinkage with a single constant and a four-level fallback, because every parameter has a physical meaning and can be checked by hand, because any figure the app shows can be explained, and because it runs without a server. The own-pace filter is the natural next step, once it has been evaluated on the full arrival shown to the user.

### *7.2. Abstention*

Several rules do not raise the measured hit rate but prevent the errors that most damage trust: labelling estimates as such, hiding an implausible delay

while keeping the arrival, discarding two sources that contradict each other, and stating that a bus may already have passed rather than announcing it as approaching. On 3 November 2026, 97.8% of the trips shown as `likely_departed` were not seen in the feed at all that day (with tables that included the morning measured; 94.8–96.5% on days not used for learning). That these buses had passed cannot be verified.

*7.3. Limitations*

- **A changing method.** The figures describe the app as it ran each day, with calibrations and rules that changed during the evaluation, not a frozen version.
- **Ground truth.** The actual passage is known to within ±30 s, which especially affects the ±1-minute figures, and it is the moment at which the feed changes the next stop, which may lag slightly behind the physical passage (equally for layer 1 and the timetable). It is not independent of layer 2, which uses the same changes of next stop, so the results without GPS may be optimistic.
- **Censoring and dependence.** Only passages observed within the capture are evaluated, so very late buses may be missed; successive predictions for the same vehicle are correlated, so the effective sample size is smaller than the number of predictions.
- **Season and holidays.** The data cover October and early November 2026, with a single weekday holiday; summer and school holidays have not been measured.
- **What the feed does not say.** A trip without a vehicle in the feed is shown with layer 3 even if it does not run (at any given moment, about 4% of the trips that should be running cannot be interpreted), extra services not in the timetable do not appear, and night trips are not measured because the feed is frozen at night.
- **One network, one evaluator.** All results are from Bizkaibus and were obtained by the author with his own tools. The reference implementation, the test vectors and the public evaluator make it possible to check that the method described is the one that was measured, but they are no substitute for independent replication on other data.

*7.4. Applying the method to another network*

The method needs, in order of importance: a GTFS timetable; a real-time feed that identifies the trip of each vehicle and its next stop, with the time of the record (enough for layers 2 and 3); the GPS position and its time (for layer 1 and for learning); shapes (if stop distances along them are missing, they are measured at build time; without shapes, localisation uses straight lines between stops); and a way to capture the feed periodically. Only the reading of the feed and the matching depend on the operator's format. The time bands reflect demand in Biscay, and several constants were measured there and should be re-measured (those marked M in Appendix A), notably the 50 m corridor, the 20-minute limit of layer 2, the 45-minute cap on plausible delays and the 60 s late departure from the terminal. With no samples, the model reduces to timetable differences with $k = 1$, which already removes the accumulated delay; learned segments appear after a few days of capture. The steps for another city are: check what the operator publishes; build the timetable database with merged calendar variants, trip aliases and corrected shapes; adapt the reading of the feed and check that every vehicle is matched to its trip; start the capture and the daily recalibration; evaluate with the protocol of Section 5.1 before showing anything as live; and check that the implementation reproduces the test vectors. If the operator already publishes per-stop predictions (GTFS-Realtime Trip Updates), they can be used directly, after checking that they are based on real data and are not a copy of the timetable.

*7.5. Open problems*

The following questions remain open. The public evaluator, learned tables and evaluation data allow answers to be compared with the method; the raw captures and the frozen timetable database are being released separately, and the day-by-day evaluation of Tables 4–5 cannot yet be recomputed from public data.

1. **Own pace.** Evaluate the Kalman filter on the vehicle's own pace on the full arrival shown to the user.
2. **Prediction intervals.** Estimate the dispersion of each segment time, not only its median, and show an interval ("in 6 to 8 minutes") with measured coverage.
3. **Comparison with the operator's predictions.** Compare the method with the waiting times shown by the operator itself, on the same vehicles and stops; a capture for this purpose is running in the public repository.

4. **Season, special days and events.** With months of data, measure whether summer, school holidays, local festivities or events should be learned separately.
5. **Trips without a vehicle.** Distinguish a trip that should already have departed but has no vehicle in the feed from one that is merely late.
6. **Vehicle sequences.** Follow the vehicle on its previous trip to estimate its actual departure from the terminal, if the assignment of vehicles to trips repeats from day to day.
7. **Other networks.** Apply the method to other cities with position-only feeds and check how much of what was measured here still holds.
8. **Journey planning.** Feed the predicted arrivals into a journey planner with transfers, so that proposed journeys use predicted rather than published times.

## 8. Conclusions

Bus networks that publish vehicle positions but not delays, with intermediate times interpolated at constant speed, can still offer accurate arrival predictions. By working in timetable time, learning the real duration of every segment by time band with shrunk medians, and choosing for each stop the most reliable of three sources with explicit abstention rules, the method described here places the arrival shown within ±2 minutes in 92–96% of cases below three minutes and in 67% at ten to twenty minutes, against 23–25% for the published timetable, and doubles the accuracy of the published timetable even without live data. More complex models that also use same-day data reduce the error of segment sums by at most 7% (up to 2% with the most recent buses, about 4% with the vehicle's own pace, 6–7% with a neural network), and a network without same-day data by 3–7%, which supports keeping a method that can be audited by hand.

Every rule, constant and evaluation step is specified, and the reference implementation, the test vectors, the evaluator, the learned tables and the evaluation data are public, so that the method can be reproduced, applied to other networks and compared with new methods on equal terms. A comparison with the waiting times shown by the operator itself is under way with the same tools.

## Declaration of competing interest

The author designed the method and develops the independent Bizkaibus+ (Bide+) passenger information app that uses it, and therefore has an interest

in its results. The app is free to use; it accepts voluntary donations, which are used to pay for the server that runs it, and the author receives no other income from it. To make the results checkable, the method is fully specified, and its reference implementation, test vectors, evaluator and evaluation data are public. The author has no relationship with Bizkaibus, the Diputación Foral de Bizkaia or the operators of the service.


### Funding

This research did not receive any specific grant from funding agencies in the public, commercial or not-for-profit sectors.


### Data and code availability

The technical report is archived on Zenodo (doi:10.5281/zenodo.24041572). The reference implementation, test vectors, evaluator, learned tables and evaluation data are archived with this preprint on Zenodo and maintained at `https://github.com/Garridoparrayeray/bide-eta` (MIT licence for code, CC BY 4.0 for text and data). The timetable and the live feed are open data from their respective publishers. The production engine of the app is not public.

### References


Achar, A. et al. (2020). "Bus Arrival Time Prediction: A Spatial Kalman Filter Approach". In: *IEEE Transactions on Intelligent Transportation Systems* 21.3, pp. 1298–1307. DOI: 10.1109/TITS.2019.2909314.

Brakewood, C. and K. Watkins (2019). "A Literature Review of the Passenger Benefits of Real-Time Transit Information". In: *Transport Reviews* 39.3, pp. 327–356. DOI: 10.1080/01441647.2018.1472147.

Cathey, F. W. and D. J. Dailey (2003). "A Prescription for Transit Arrival/Departure Prediction Using Automatic Vehicle Location Data". In: *Transportation Research Part C: Emerging Technologies* 11.3–4, pp. 241–264. DOI: 10.1016/S0968-090X(03)00023-8.

Cats, O. and G. Loutos (2016). "Real-Time Bus Arrival Information System: An Empirical Evaluation". In: *Journal of Intelligent Transportation Systems* 20.2, pp. 138–151. DOI: 10.1080/15472450.2015.1011638.

Cerqueira, V., L. Torgo, and I. Mozetič (2020). "Evaluating Time Series Forecasting Models: An Empirical Study on Performance Estimation Methods". In: *Machine Learning* 109.11, pp. 1997–2028. DOI: 10.1007/s10994-020-05910-7.

Chien, S. I.-J., Y. Ding, and C. Wei (2002). "Dynamic Bus Arrival Time Prediction with Artificial Neural Networks". In: *Journal of Transportation Engineering* 128.5, pp. 429–438. DOI: 10.1061/(ASCE)0733-947X(2002)128:5(429).

Clark, S. D., S. Grant-Muller, and H. Chen (2002). "Cleaning of Matched License Plate Data". In: *Transportation Research Record: Journal of the Transportation Research Board* 1804, pp. 1–7. DOI: 10.3141/1804-01.

Efron, B. and C. Morris (1975). "Data Analysis Using Stein's Estimator and its Generalizations". In: *Journal of the American Statistical Association* 70.350, pp. 311–319. DOI: 10.1080/01621459.1975.10479864.

European Committee for Standardization (CEN) (2022). *Public Transport. Service Interface for Real-Time Information Relating to Public Transport Operations. Part 3: Functional Service Interfaces*. European Standard EN 15531-3:2022.

Gal, A. et al. (2017). "Traveling Time Prediction in Scheduled Transportation with Journey Segments". In: *Information Systems* 64, pp. 266–280. DOI: 10.1016/j.is.2015.12.001.

García-Mauriño, C., P. J. Zufiria, and A. Jarabo-Peñas (2024). "Improving Bus Arrival Time Predictors Using Only Public Transport API Data". In: *Transportation Letters* 16.8, pp. 804–813. DOI: 10.1080/19427867.2023.2245994.

Garrido, Y. (2026). *Predicción de llegadas de autobuses con posiciones en tiempo real y horarios interpolados*. Tech. rep. Version 2.1. Zenodo. DOI: 10.5281/zenodo.24041572. In Spanish.

Gelman, A. et al. (2013). *Bayesian Data Analysis*. 3rd ed. CRC Press. DOI: 10.1201/b16018.

Gooze, A., K. E. Watkins, and A. Borning (2013). "Benefits of Real-Time Transit Information and Impacts of Data Accuracy on Rider Experience". In: *Transportation Research Record: Journal of the Transportation Research Board* 2351, pp. 95–103. DOI: 10.3141/2351-11.

Hauer, E. et al. (2002). "Estimating Safety by the Empirical Bayes Method: A Tutorial". In: *Transportation Research Record: Journal of the Transportation Research Board* 1784, pp. 126–131. DOI: 10.3141/1784-16.

Huber, P. J. (1964). "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1, pp. 73–101. DOI: 10.1214/aoms/1177703732.

James, W. and C. Stein (1961). "Estimation with Quadratic Loss". In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*. Ed. by J. Neyman. Vol. 1. University of California Press, pp. 361–379.

Jeong, R. and L. R. Rilett (2005). "Prediction Model of Bus Arrival Time for Real-Time Applications". In: *Transportation Research Record: Journal of the Transportation Research Board* 1927, pp. 195–204. DOI: 10.1177/0361198105192700123.

Jeong, S., C. Oh, and J. Jeong (2024). "BAT-Transformer: Prediction of Bus Arrival Time with Transformer Encoder for Smart Public Transportation System". In: *Applied Sciences* 14.20, 9488. DOI: 10.3390/app14209488.

Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1, pp. 35–45. DOI: 10.1115/1.3662552.

Kaufman, S. et al. (2012). "Leakage in Data Mining: Formulation, Detection, and Avoidance". In: *ACM Transactions on Knowledge Discovery from Data* 6.4, 15. DOI: 10.1145/2382577.2382579.

Kumar, B. A. et al. (2025). "Bus Arrival Time Prediction: A Comprehensive Review". In: *IEEE Transactions on Intelligent Transportation Systems* 26.6, pp. 7362–7379. DOI: 10.1109/TITS.2025.3545695.

Lin, W.-H. and J. Zeng (1999). "Experimental Study of Real-Time Bus Arrival Time Prediction with GPS Data". In: *Transportation Research Record: Journal of the Transportation Research Board* 1666, pp. 101–109. DOI: 10.3141/1666-12.

Liu, L. and H. J. Miller (2020). "Does Real-Time Transit Information Reduce Waiting Time? An Empirical Analysis". In: *Transportation Research Part A: Policy and Practice* 141, pp. 167–179. DOI: 10.1016/j.tra.2020.09.014.

Ma, J. et al. (2022). "Multi-Attention Graph Neural Networks for City-Wide Bus Travel Time Estimation Using Limited Data". In: *Expert Systems with Applications* 202, 117057. DOI: 10.1016/j.eswa.2022.117057.

Manibardo, E. L., I. Lana, and J. Del Ser (2022). "Deep Learning for Road Traffic Forecasting: Does it Make a Difference?" In: *IEEE Transactions on Intelligent Transportation Systems* 23.7, pp. 6164–6188. DOI: 10.1109/TITS.2021.3083957.

Mazloumi, E., G. Currie, and G. Rose (2010). "Using GPS Data to Gain Insight into Public Transport Travel Time Variability". In: *Journal of Transportation Engineering* 136.7, pp. 623–631. DOI: 10.1061/(ASCE)TE.1943-5436.0000126.

Ministerio de Transportes y Movilidad Sostenible (2026). *Punto de Acceso Nacional de transporte multimodal*. URL: https://nap.transportes.gob.es (visited on 10/31/2026).

MobilityData (2026). *GTFS Schedule Reference*. URL: https://gtfs.org/documentation/schedule/reference/ (visited on 10/31/2026).

Moreira-Matias, L. et al. (2015). "Improving Mass Transit Operations by Using AVL-Based Systems: A Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 16.4, pp. 1636–1653. DOI: 10.1109/TITS.2014.2376772.

Pang, J. et al. (2019). "Learning to Predict Bus Arrival Time from Heterogeneous Measurements via Recurrent Neural Network". In: *IEEE Transactions on Intelligent Transportation Systems* 20.9, pp. 3283–3293. DOI: 10.1109/TITS.2018.2873747.

Petersen, N. C., F. Rodrigues, and F. C. Pereira (2019). "Multi-Output Bus Travel Time Prediction with Convolutional LSTM Neural Network". In: *Expert Systems with Applications* 120, pp. 426–435. DOI: 10.1016/j.eswa.2018.11.028.

Rahman, M. M., S. C. Wirasinghe, and L. Kattan (2013). "Users' Views on Current and Future Real-Time Bus Information Systems". In: *Journal of Advanced Transportation* 47.3, pp. 336–354. DOI: 10.1002/atr.1206.

Shalaby, A. and A. Farhan (2004). "Prediction Model of Bus Arrival and Departure Times Using AVL and APC Data". In: *Journal of Public Transportation* 7.1, pp. 41–61. DOI: 10.5038/2375-0901.7.1.3.

Singh, N. and K. Kumar (2022). "A Review of Bus Arrival Time Prediction Using Artificial Intelligence". In: *WIREs Data Mining and Knowledge Discovery* 12.4, e1457. DOI: 10.1002/widm.1457.

Tashman, L. J. (2000). "Out-of-Sample Tests of Forecasting Accuracy: An Analysis and Review". In: *International Journal of Forecasting* 16.4, pp. 437–450. DOI: 10.1016/S0169-2070(00)00065-0.

Vanajakshi, L., S. C. Subramanian, and R. Sivanandan (2009). "Travel Time Prediction under Heterogeneous Traffic Conditions Using Global Positioning System Data from Buses". In: *IET Intelligent Transport Systems* 3.1, pp. 1–9. DOI: 10.1049/iet-its:20080013.

Watkins, K. E. et al. (2011). "Where Is My Bus? Impact of Mobile Real-Time Information on the Perceived and Actual Wait Time of Transit Riders". In: *Transportation Research Part A: Policy and Practice* 45.8, pp. 839–848. DOI: 10.1016/j.tra.2011.06.010.

Wessel, N., J. Allen, and S. Farber (2017). "Constructing a Routable Retrospective Transit Timetable from a Real-Time Vehicle Location Feed and GTFS". In: *Journal of Transport Geography* 62, pp. 92–97. DOI: 10.1016/j.jtrangeo.2017.04.012.

Wessel, N. and S. Farber (2019). "On the Accuracy of Schedule-Based GTFS for Measuring Accessibility". In: *Journal of Transport and Land Use* 12.1, pp. 475–500. DOI: 10.5198/jtlu.2019.1502.

Woodard, D. et al. (2017). “Predicting Travel Time Reliability Using Mobile Phone GPS Data”. In: *Transportation Research Part C: Emerging Technologies* 75, pp. 30–44. DOI: 10.1016/j.trc.2016.10.011.

Yu, B., Z. Yang, and B. Yao (2006). “Bus Arrival Time Prediction Using Support Vector Machines”. In: *Journal of Intelligent Transportation Systems* 10.4, pp. 151–158. DOI: 10.1080/15472450600981009.

Zhou, J. Q. et al. (2025). “Addressing Data Latency in GTFS (General Transit Feed Specification) Realtime to Improve Transit Signal Priority”. In: *Transportation Research Record: Journal of the Transportation Research Board* 2679.1, pp. 1329–1341. DOI: 10.1177/03611981241255027.

## Appendix A. Constants

Tables A.1 and A.2 list every constant of the method, with its name in the reference implementation (`referencia/referencia.py`) where one exists. Constants marked M were fixed by measurement on Bizkaibus and should be re-measured on another network.

Table A.1: Constants of the method.

| Meaning | Name in the reference | Value | M |
|---|---|---|---|
| *Matching and localisation* | | | |
| Fallback matching tolerance | `MATCH_TOLERANCE` | 180 s | |
| Metres per degree | `METERS_PER_DEGREE` | 111,320 m | |
| Window: segments before / from the next stop | `SEGMENTS_BEHIND`, `_AHEAD` | 3 / 2 segments | |
| Slack beyond the next stop | `BEYOND_SLACK` | 100 m | |
| Ambiguity margin | `AMBIGUITY` | 20 m | |
| Corridor with a shape | `MAX_FROM_SHAPE` | 50 m | M |
| Corridor without a shape (minimum, ratio) | `MAX_FROM_ROUTE`, `_RATIO` | 150 m, 0.15 | M |
| Maximum age of the data | `MAX_LOCATION_AGE` | 600 s | |
| Trip counts as started before departure | `NOT_STARTED_GRACE` | 60 s | |
| Maximum wait at the first stop | `MAX_WAIT_AT_ORIGIN` | 3,600 s | |
| *Segment times, length fallback and pace factor* | | | |
| Start of the six periods of the day | `BANDS` | 0, 6, 9, 13, 16, 20 h | |
| Shrinkage weight $n_0$ | `SEGMENT_BAND_PRIOR_TRACKS` | 5 tracks | M |
| Minimum tracks per segment and group | `MIN_TRACKS_PER_SEGMENT` | 2 tracks | M |
| Length-bucket cuts | `LENGTH_CUTS` | 300, 600, 1,000, 1,500, 3,000 m | |
| Minimum ratios per bucket and group | — | 20 samples | |
| Bounds of the length ratio | `RATIO_MIN`, `RATIO_MAX` | 0.3, 1.5 | |
| Bounds of the pace factor | `K_MIN`, `K_MAX` | 0.6, 1.2 | M |
| Shrinkage of the pace factor | `SHRINKAGE_TRACKS` | 10 tracks | |
| Minimum tracks: band / line (and vehicles) | — | 8 / 25 (5) tracks | M |
| Minimum scheduled remaining time of a pace sample | — | 120 s | |

Table A.2: Constants of the method (continued).

| Meaning | Name in the reference | Value | M |
|---|---|---|---|
| *Three layers* | | | |
| Remaining share of the segment in layer 2 | `REGISTRATION_SHARE` | 0.5 | M |
| Longest segment for layer 2 | `REGISTRATION_MAX_SEGMENT` | 1,200 s | M |
| Cross-check: longest segment / disagreement | `CROSSCHECK_MAX_SEGMENT`, `MAX_GPS_VS_STOP` | 360 / 360 s | M |
| Late departure from the terminal | `SCHEDULE_ORIGIN_OFFSET` | 60 s | M |
| `likely_departed`: passed / gap / learned share | `WRONG_SCHEDULE_*` | 600 s / 600 s / 0.8 | |
| Parts of the route in the per-trip correction | `TRIP_PARTS` | 3 | M |
| Clip of each deviation in the per-trip correction | `TRIP_OFFSET_CLIP` | 900 s | M |
| Weight of the band in the line correction | `TRIP_OFFSET_LINE_PRIOR` | 20 passages | |
| Weight of the line in the trip correction | `TRIP_OFFSET_TRIP_PRIOR_DAYS` | 1 day | M |
| Percentile of the per-trip correction: start / nightly choice | `TRIP_OFFSET_RANK`, `_RANKS` | 0.4 / 0.3–0.5 | M |
| Test days of the per-trip correction | — | 7 days | |
| Passages left out of the per-trip correction (deviation from the published time) | — | 2,700 s | |
| GPS distance that confirms a passage | `PASSED_DISTANCE` | 150 m | |
| Arriving without a confirmed passage | `ARRIVING_MAX` | 180 s | |
| Implausible delay | `MAX_PLAUSIBLE_DELAY` | 2,700 s | M |
| Grace for a past layer-2 arrival | `STALE_GRACE` | 120 s | |
| Hold of a vehicle missing from the feed | `HOLD_SECONDS` | 180 s | M |
| Entry of yesterday's trip (departure ahead of now by more than) | `DAY_SECONDS/2` | 43,200 s | |
| *Learning* | | | |
| Feed download interval | — | 20 s | |
| Largest gap between track points | `MAX_GAP_BETWEEN_SNAPSHOTS` | 300 s | |
| Maximum pace samples per track | `MAX_SAMPLES_PER_TRACK` | 30 samples | |
| Observation window after the published time (pace samples) | — | 600 s | |
| Lifetime of the passage log, after the end of its day | — | 36 h | |
| Retention / samples per segment and day type / pace samples | — | 120 days / 300 / 500,000 | |