# Core Algorithm

## 1. Purpose and Context

We need a function that, given a set of anonymized cruise-control samples for a particular road segment plus the official speed-limit rules, produces a single, safe, “best” cruise speed. This chosen speed will drive our simulation and, eventually, real vehicles in adaptive cruise mode.

## 2. Inputs and Outputs

| Input | Description |
| --- | --- |
| **Samples** `<CruiseSample[]>` | A list of records `{ speedKmh, timestamp }`, each one anonymized and stripped of PII |
| **Rules** `<SegmentRules>` | Contains `mapSpeedLimitKmh` and an optional list of `timeDependentLimits` (e.g. school-zone hours) |
| **Now** `<Date>` (optional) | The current time, used to evaluate time-dependent limits and filter out stale data |

| Output | Description |
| --- | --- |
| **Profile** `<CruiseProfile>` | `{ segmentId, chosenSpeedKmh, reason }` — the final speed recommendation and a human-readable note |

## 3. High-Level Steps

1. **Filter Out Bad Data**
    - **Age filter:** Discard any sample older than a configurable window (e.g. 30 days).
    - **Speed-limit outliers:** Drop any sample exceeding 120 % of the map’s speed limit (GPS glitches or driver error).
2. **Determine the Active Speed Limit**
    - Start with the official `mapSpeedLimitKmh`.

- For each defined `timeDependentLimits` window, if the current time falls inside it, take the minimum of that window's limit and the map limit.

3. **Select the Best Cruise Speed**
   - **No valid samples:**
     - Fallback immediately to the active speed limit.
     - Reason: "no valid samples."
   - **Few samples (< 8):**
     - Take the **most recent** sample's speed (drivers implicitly set what felt best last).
     - Cap it at the active limit.
     - Reason: "only N samples, using most recent."
   - **Many samples (≥ 8):**
     - Compute the **median** of all sample speeds (robust to remaining outliers).
     - Cap the median at the active limit.
     - Reason: "median of N samples = X km/h."
4. **Emit the Cruise Profile**
   - Package `{ segmentId, chosenSpeedKmh, reason }` and return it.

---

# 4. Edge Case Considerations

- **All samples too old or too fast:**

  Triggers the "no valid samples" branch.

- **Time-window wrap-around** (e.g. night-time restrictions from 22:00 to 06:00):

  Ensure your logic correctly handles `startHour > endHour`.

- **Sparse data regions:**

  Use a low threshold (e.g. 1 sample) so we can still offer a useful speed, even if it's just the last known driver's choice.

- **Rapidly changing speed limits:**

  If a road's posted limit changes (construction, dynamic signage), freshly ingested samples might conflict. You may want to tag samples with the map version they came from and invalidate old-limit samples when rules update.

- **Uniform driver behavior:**

  If all drivers always set exactly the limit, median and recent-sample branches converge. That's fine—the algorithm gracefully handles "everyone drives at the limit."

---

## 5. GDPR & Privacy Compliance

- **Anonymization:** Samples contain **no user identifiers**. Data is aggregated solely by segment.
- **Aggregation:** We never store or surface individual driver histories—only the chosen profile per segment.
- **Data retention:** Enforce a rolling window (e.g. 30 days) so old data is purged automatically.
- **Transparency:** The `reason` field lets us explain how we arrived at each speed, aiding audit and compliance.

---

## 6. Extensibility & Future Enhancements

- **Weighted sampling:** Add recency weights so newer samples slightly influence the median more.
- **Clustering by context:** Group samples by weather or traffic conditions (if that metadata becomes available) and choose different profiles accordingly.
- **Confidence scoring:** Alongside `chosenSpeedKmh`, compute a confidence metric (e.g. based on sample count or variance) to flag segments that need more data.
- **Dynamic fallback:** For segments with zero samples but low confidence (e.g. rare roads), you could use neighboring segment profiles to estimate a safe speed.