Audion Traffic Counter — user guide

Contents

- What you need
- Starting and installing
- How the window is laid out
- RECORDINGS
- CAMERA: lines, gates, zones
- COUNT
- RESULTS
- TRAINING
- Working with Claude and Codex
- SETTINGS
- Keys
- What accuracy to expect
- Where things are
- What the program does on the network and on the machine
- If something goes wrong

The program counts vehicles in recordings of ordinary cameras: how many cars, buses, trucks and motorcycles made each movement. Counting lines are drawn on the frame, the recording goes into a queue, and the result is a table by movement and class plus a file of events, one per vehicle.

The count runs on your machine. Recordings are not sent anywhere.

What you need

- Windows 10 or 11, 64-bit.
- An NVIDIA video card of the RTX series, driver 570 or newer, is the main way to count: through TensorRT a half-hour recording of 640x480 takes about five minutes on an RTX 5070. An NVIDIA

GTX card counts through PyTorch, several times slower. Without NVIDIA the program counts through DirectML on any DirectX 12 video card (Intel, AMD): on integrated Intel graphics a half-hour recording takes about an hour and a half.

- The internet during installation, and disk space: about 5 GB on a machine without NVIDIA; about 15 GB on a machine with an RTX card (PyTorch and TensorRT are 3 GB downloads each).

Starting and installing

Unpack the program folder anywhere and start `Start.exe`. The program is portable: it keeps everything of its own in this folder. One thing goes into Windows: the Microsoft Visual C++ Redistributable PyTorch needs - and only when it is not there. The list of recordings, the lines and the results can be kept apart from the program, in a work folder - see "SETTINGS".

At the first start the counting engine is not there yet: the window opens the **SETUP** section by itself and asks whether to install it. The question says that without the engine the program does not count, how much comes for this machine (about 15 GB of disk on an RTX card, about 5 GB without an NVIDIA card), and what will be installed into Windows. Nothing is downloaded until you answer **DOWNLOAD AND INSTALL**; after a refusal the button **INSTALL WHAT IS NEEDED** asks the same:

1. The **THIS MACHINE** card shows the video cards and what follows from them.
2. Press **INSTALL WHAT IS NEEDED** and confirm. Green marks exactly what this machine lacks: its own Python, PyTorch for your video card, TensorRT for an RTX card, the detection library, FFmpeg. Before them the program looks whether Windows has the Microsoft Visual C++ Redistributable and, when it has not, installs it with winget - Windows asks for the administrator's consent.
3. The installation is shown in the journal (the **JOURNAL** button, bottom right). It can be stopped with STOP and started again.
4. When everything is installed, the program removes the downloaded installation files by itself: on a machine with an RTX card the program folder then takes about 11.5 GB.

The installed folder can be carried whole to another machine with an NVIDIA card: nothing has to be installed again. On another video card the first count starts half a minute later - the program builds its fast engine for that card. On a machine without an NVIDIA card open SETUP: it shows in green what that machine lacks.

The other buttons of the section repair one part. TENSORRT is the fast count on an RTX card: on a machine with one it is installed together with the environment. DIRECTML and OPENVINO are spare ways to count; install OPENVINO only as the last resort: it is the least reliable for a queue of many recordings.

SETUP holds only what is installed for the counting engine. How the program itself works is chosen in the **SETTINGS** section.

If the NVIDIA driver is too old, the section says so and asks you to update it.

The project file

The program keeps the lines of every recording in its own folder and, as a second copy, in the project file `audion-traffic-counter.json`. By default it lies in the folder of the videos themselves. A recording is known there by the fingerprint of its file, not by its name, so the placement comes back by itself: move the folder of recordings to another machine, install the program anew, add a recording again - the lines are there.

The **PROJECT FILE** card of SETTINGS chooses the place: **BESIDE THE RECORDINGS** or **IN THE USER'S FOLDER** - one file for all recordings in the chosen folder. The second is for recordings that lie on a read-only disc or in somebody else's folder.

How the window is laid out

- **At the top** - the five sections of the work, in order: RECORDINGS, CAMERA, COUNT, RESULTS, TRAINING. Under the name of each is its state: how many recordings, how many of them without lines, what is queued, what names the kinds.
- **Bottom left** - SETUP, SETTINGS and ABOUT; **bottom right** - the **COUNT** button, which becomes STOP while a count runs. SETUP is what is installed for the counting engine (Python, PyTorch, the model, FFmpeg); SETTINGS is how the program itself works.
- Top right - the language (RU | EN), the dark and the light theme, the font size.
- Every button and field has a detailed tooltip: rest the mouse on it.

RECORDINGS

ADD FILES... and **ADD A FOLDER...** put videos on the list. The files stay where they are: the program remembers the path and copies nothing.

The list takes `.avi`, `.mp4`, `.mov`, `.mkv`, `.mts`, `.m2ts`, `.mpg`, `.wmv`, `.ts` and `.webm` files; ADD A FOLDER passes by files with other extensions. The codecs are the usual ones: H.264, H.265, AV1, MJPEG, MPEG-2, MPEG-4. If a recording of a rare format shows in CAMERA and its count "failed", the counting engine could not read it: re-encode it to MP4 (H.264). Make an interlaced recording (DV 720x576) progressive before it is counted.

A row shows the frame size and the length, whether the lines are placed, and how many times the recording was counted. The mark on the left takes the recording into the queue. A click on a row chooses the recording for CAMERA and RESULTS.

A new recording comes without lines.

CAMERA: lines, gates, zones

On the left is the video of the chosen recording, on the right its placement. Every recording has a placement of its own: an edit on one recording does not change the others.

A line

Take the **LINE** tool, press on the frame and drag the line across the road.

- A line counts the vehicles that cross it **along its arrow**. The arrow stands in the middle of the line. The move of the mouse sets it: drawn bottom to top - those going right are counted, top to bottom - left, right to left - away from the camera, left to right - towards the camera.
- The two arrow buttons in the row of the line, on the right, turn the arrow round.
- Type into **Movement** the name these vehicles will bear in the results.
- A line is a **LINE** (one way, along the arrow) or **TWO-WAY** - buttons in the row of the line. A two-way line has two arrows on the frame, and its row has the field **Against the arrow**: under that name the line counts the oncoming flow as a number of its own. It is for a road where the opposite flows are not divided and the camera stands between them: a vehicle crosses the line in any lane, and its direction decides where it is written. Where a barrier divides them, two one-way lines, each across its own carriageway, are surer.
- The **CHECK LINE** button is for a line that counts a flow other lines count further on: the whole flow before a fork, for example, with a line on each ramp behind it. The numbers of such a line are shown in the results and left out of the TOTAL row - the same vehicles would be counted twice otherwise.

Put a line where vehicles are large and do not hide one another: not on a stop line, not behind a pole and not at the very edge of the frame. A vehicle is counted once per pass.

A gate

Gates are for an intersection and a roundabout, when it matters **from where and to where** a vehicle went. With the **GATE** tool draw a line across every branch - over its entry and exit lanes together - and

name the branch. From the gates the program pairs the entry and the exit of one vehicle into a manoeuvre "A → B": left, straight on, right, a U-turn.

A straight road needs no gates - lines are enough.

A zone

Far vehicles take a few pixels, and the program misses them. With the **ZONE** tool draw a rectangle around the far part of the frame and choose the enlargement, ×2 or ×3: the zone is counted on its own, enlarged. A line that lies in the zone is given to it (the "Zone" switch in the row of the line). Enlargement slows the count down.

Editing and going through the recordings

- The **POINTER** tool: an end of a line is dragged, the line itself is moved by its middle, a zone by a corner or an edge. Delete removes what is chosen.
- Under the video: the previous recording, a frame back, play, a frame forward, the next recording. Going to a neighbouring recording saves the edits by itself; so do closing the program and starting a count.
- **AS THE PREVIOUS ONE** copies the lines, gates and zones of the recording above in the list. Handy for recordings of one shoot from one spot: copy and, if the frame has shifted, nudge the ends.
- **VIDEO IN FFPLAY** opens the recording in a large window of its own with the lines drawn.

This is how all recordings are gone through in a row: draw the lines, go to the next one.

COUNT

At the top left is **THE COUNT AS IT GOES**: how many vehicles are counted so far, the speed and the time left. When several recordings are counted at once, each has a row of numbers of its own. On the right are **THIS MACHINE** and **HOW TO COUNT**, below them the **QUEUE**; a long queue scrolls by itself, the rest stays where it is.

THIS MACHINE is what the count runs on: the processor, the memory, the video card, its driver, Windows. The **ENGINE** tile shows what the program counts with: while nothing is counted - the choice of "Count on" (AUTO); during a count - what the recording is really counted on, TensorRT for example.

The **HOW TO COUNT** card:

- **Model** - from YOLO26N (the fastest) to YOLO26X (the most accurate and the slowest). A larger model sees far and partly hidden vehicles better. The weights are downloaded by themselves at the

first count.

- **Count on** - AUTO chooses by itself: an NVIDIA card (TensorRT when it is installed, else CUDA), otherwise DirectML. What the machine has not got is dimmed.

How many recordings to count at once, the threads and the priority of the count are in the SETTINGS section.

TRIAL RUN counts the first minute of the chosen recording and saves frames with boxes and lines. Look at them before a count that takes hours: the boxes must stand on vehicles, the lines across the flows.

COUNT counts all marked recordings in order - several at once on an NVIDIA card, and then every recording has a row of numbers of its own in **THE COUNT AS IT GOES**. The queue can be left for the night:

- a recording without lines is skipped;
- a recording the count failed on is skipped too - the rest are counted;
- a recording that breaks off (a damaged or cut file) is not taken for a counted one: its count "failed", and the journal says why. When no more than a second (or one per cent of the recording) is missing at the end, the run is kept and marked in RESULTS as not read to the end;
- the marks in RECORDINGS can be changed while a count runs: a recording the queue has not reached yet leaves it or joins it;
- while the count runs the computer will not fall asleep by itself (a closed laptop lid can still put it to sleep);
- at the end the bottom line says how many were counted, skipped and failed.

RESULTS

Every count of a recording is kept as a **run** of its own and does not replace the earlier ones.

- **RUNS** - the list: when, with which model, on what, how long, how many vehicles.
- **THE RUN** - the table of the chosen run: movements in rows, classes in columns; below it, the manoeuvres through gates. The last row, **TOTAL**, is the number of vehicles of the whole recording: the sum over all movements for each class and for all of them, like the total column of a survey sheet. Check lines are left out of it; of a gate only the entry is counted.
- **RUNS SIDE BY SIDE** - the runs as columns: what a change of the model or of the lines has moved. The last row is the TOTAL of every run.

While a count runs, the number of vehicles so far stands in the **THE COUNT AS IT GOES** card of COUNT - the **TOTAL** tile before the tiles of the movements.

A run counted by an earlier version of the program has no TOTAL row: count the recording again.

- **TABLE OF EVENTS** opens the file where every row is one crossing: frame, time, movement, class, line. **RUN FOLDER** opens the folder with the files.

The report: Word and Excel files

The **REPORT** card at the top of the section writes the result into a folder you name:

- **REPORT ON ALL RECORDINGS** - on every counted recording, by the newest run of each;
- **REPORT ON THIS RUN** - on one recording, by the run chosen in the list;
- **REPORTS FOLDER** - opens the folder of the last report.

The row **Vehicle sheets** has the switches **TRUCKS**, **BUSES**, **MOTORCYCLES**, **CARS**. A kind that is on goes out as pictures as well: beside the report files there is a folder "Vehicle sheets", a folder per recording in it, and in that sheets by section and kind, 48 vehicles to a sheet. Every picture has its number (from 1 to the number in the table) and the time it passed from the start of the recording. By the sheets a specialist splits the trucks into light and heavy ones, the buses into small and big ones, and finds the special vehicles - everything the program does not tell apart. Trucks and buses are on by default, cars are off: there are many of them. The pictures of the vehicles are kept by a count with a model of kinds; a recording counted without one gets no sheets, and the window says so.

There are two files, both written for a transport engineer. **The Word document** begins with the main figures (how many recordings, vehicles, cars, buses, trucks) and an overview with the volume in vehicles an hour; then every recording has a page of its own: the number of vehicles, the volume and the composition of the flow in percent, a frame with the counting sections and the numbers of the directions, the table by direction, the total. **The Excel sheet** holds the same numbers as tables: the sheet "Overview" (by direction, with the volume and the composition), the sheet "By 15 minutes" (the volume by interval from the start of the recording, with a filter), the sheet "Recordings", and the sheet "Service" - by what and when every recording was processed, for whoever maintains the program. The volume is the count of the recording brought to an hour: a recording of 15 minutes is multiplied by four. There are no passenger-car equivalents in the report. The name of a file carries the date and the time, so a new report does not replace an earlier one.

A recording whose numbers are not to be trusted without a check is marked **"check"**, with the reason beside it:

- the recording was not read to its end - the numbers cover only the part that was read;

- the kinds are doubtful - the model of kinds renamed too many of the vehicles the detector took for cars. It happens on a camera unlike those the model was taught on. The number of vehicles is right then; their division into cars, buses and trucks is not to be trusted on such a recording.

When recordings were counted by hand (see below), the report holds the comparison as well.

Checking against a manual count

If the recording was once counted by hand, the program shows the difference in percent beside every number. Put a file `reference\<recording name>\<recording name>.json`, where the recording name is the name of its file without the extension (for `DSCN0002.AVI` it is `reference\DSCN0002\DSCN0002.json`):

```
{
  "engine_classes": { "car": [0], "bus": [1], "truck": [2, 3] },
  "counts": {
    "N1": [120, 4, 9, 2],
    "N2": [87, 2, 5, 0]
  }
}
```

`counts` holds a row of numbers per movement, in the order of the columns of your sheet. `engine_classes` says which columns add up into the classes of the program: `car`, `bus`, `truck`.

TRAINING

The detector confuses the kinds of vehicles - cars, trucks, buses: it calls a route van now a bus, now a truck, and a long truck a bus. The TRAINING section names them more exactly.

The **MODEL OF KINDS** card - what names the kinds in a count:

- **AUTO** (the choice from the start) - the default model of kinds. It comes with the program (`system_core\kinds`), so nothing has to be installed or taught for it. Which model is the default one is written in `config\classes_default.json`; should its file be missing, the detector names the kinds, and the tooltip of the button says so;
- **DETECTOR** - as before, the kind by the boxes of the detector;
- **a model of kinds** (the one that comes with the program and your own, the files `models\classes\*.pt`) - the count keeps three pictures of every counted vehicle, and the model names the kind by them. What the detector called a car stays a car unless the model is 90 % sure it

is a bus; a detector's car is never renamed a truck. What the detector called a truck or a bus is the model's to name. A motorcycle stays what the detector called it.

The choice takes effect on the next count. The pictures take about 50 MB for a half-hour recording and lie in the folder of the run.

The **VEHICLES OF THE RUN** card - the pictures of the vehicles of the chosen run:

- **DISPUTED** - the vehicles the detector and the model disagree on, or the model is not sure of; a half-hour recording usually has some tens to a couple of hundred;
- **MARKED** - the ones whose kind has been set by hand;
- **NOT CARS** - every vehicle the detector or the result of the run calls other than a car.

The buttons under a vehicle - CAR, TRUCK, BUS, NOT A VEHICLE - set its kind. A mark corrects the results of the run in RESULTS at once; the kind of the detector and the kind of the model stay beside it in the table of events (`events.csv`: `detector_class`, `classifier_class`, `classifier_share`, `mark`), the marks lie in `review.csv` of the run's folder. NOT A VEHICLE takes the vehicle out of the results: two vehicles in one box, not a vehicle of the three kinds.

How to mark: a route van - a minibus with windows along the whole side - is a **bus**, as in the sheets of the people who count by hand; a Gazelle without windows along its side, a van, a flatbed is a **truck**; a minivan, a UAZ van, an ambulance, a pickup is a **car**.

The marks are also the examples a model of kinds is taught on for your cameras. The model is taught by commands of the engine (most easily by an agent, see "Working with Claude and Codex") and comes as a new file in `models\classes`, in the MODEL OF KINDS card beside the earlier ones. There is no teaching button in the window.

Working with Claude and Codex

The program is made to be worked on together with an AI agent - Claude Code or Codex - the way its authors worked on it. The guide for the agent is `Docs\AI_AGENT_GUIDE_EN.md` (in Russian: `AI_AGENT_GUIDE_RU.md`): the commands of the engine, the file formats, the order of the work and what has already been measured.

How to start. Install the program (SETUP, INSTALL WHAT IS NEEDED), open its folder in Claude Code or Codex, and ask in your first message to read `Docs\AI_AGENT_GUIDE_EN.md`. Then speak plainly:

- "Place the lines on every recording of the list from a frame of each, I will check the pictures" - the agent takes a frame from the middle of each recording, draws the lines and gates and shows them

on the frame;

- "Count the recording DSCN0002 and compare it with the manual count";
- "Why are there fewer buses than in the sheet? Lay the large vehicles of line N10 on sheets" - the agent shows the pictures, and the dispute is settled by the eye;
- "Teach the kinds for my cameras: the recordings are in D:\Video, check on another day".

Teaching the kinds with an agent goes so:

1. the agent cuts pictures of every vehicle of a part of a recording (the command `harvest`) or takes them from a run counted with a model of kinds;
2. lays them on sheets of 48 (`sheets`) - first everything the detector called a bus or a truck, then a sample of the cars;
3. looks at the sheets and gives every vehicle the letter of its kind (`labels`) - tell it how your sheets count route vans and small trucks, and look at a couple of its sheets;
4. teaches the model (`teach`) - half an hour on a laptop processor, minutes on an NVIDIA card;
5. checks it on vehicles it has not seen (`check-classes`) and tells by how many percent the count of each kind is off;
6. you choose the model in the TRAINING section, and the next count goes with it.

The main thing we learned: what decides is not the length of the teaching but pictures from cameras of the same kind and the right boundary between the kinds. Two hundred labelled vehicles of your own camera do more than thousands of others. In detail: the guide for the agent, "Teaching the kinds for the user's cameras".

About the recordings. The agent works on your machine, and the recordings themselves go nowhere. But what the agent looks at - a frame to place the lines by, sheets of pictures to label - goes to the service it runs on (Anthropic for Claude, OpenAI for Codex). If the recordings must not be shown to others, label them yourself: by the buttons of the TRAINING section or by letters in the files beside the sheets - the agent then needs only the commands, not the pictures.

The agent does not start the count of the queue itself: that is the COUNT button.

SETTINGS

How the program itself works (the button at the bottom left, between SETUP and ABOUT).

The **SPEED OF THE COUNT** card - these settings do not change the counts, only the time:

- **Recordings at once** - how many recordings of the queue are counted at the same time on an NVIDIA card, four by default. When one is through, the next takes its place. A machine without an NVIDIA card counts recordings one at a time.
- **Threads** - how many threads run the model inside one recording; 0 - the program chooses.
- **Priority** - BACKGROUND leaves the computer usable while the count runs.

SETTINGS has a field for every kind of data: the recordings, the work folder, the reports, the project file, the training data. Recordings and reports can be had again - the recordings copied anew, a report written once more; what the program has learned (the model of vehicle kinds) lies in the program itself.

The **RECORDINGS** card - the folder the recordings lie in. The program only reads them. The choice of files in the RECORDINGS section opens in this folder. When the recordings were moved to another disk or copied anew, choose the new folder here: the recordings of the list that are not where they were are found in it and in its subfolders by the name and the fingerprint of the file, and their lines and results stay with them. The program does the same at every start.

The **REPORTS** card - the folder the Word and Excel reports are written into; the report buttons of the RESULTS section open the choice of a folder at this place.

The **WORK FOLDER** card - where what the program writes about your recordings lies: the list of recordings, the lines and gates, the results of the counts, the sheets of manual counts to compare with, the journal. **IN THE PROGRAM'S FOLDER** (the default) - all of it in the folder of the program itself. **IN THE USER'S FOLDER** - in the folder you choose, and the folder of the program holds the program and its settings alone: it can be copied to another machine, cleaned and handed over without carrying the lists, the lines and the results along. The layout of a work folder is the one of the program: `config\recordings.json`, `config\cameras\`, `output\`, `reference\`, `logs\`. Choosing a folder moves nothing: to take the former work along, close the program and move these folders and the list file into the new work folder. The folder is changed without a restart; it cannot be changed while a count runs.

The **PROJECT FILE** card - where the second copy of the lines of the recordings is kept (see "The project file").

The **TRAINING DATA** card - where the pictures of vehicles, the sheets, the labels and the sets for teaching a model of kinds lie: **BESIDE THE RECORDINGS** (a folder `classes` beside the videos - the default, the data travel with the recordings) or **IN THE USER'S FOLDER**. The program never erases them, and they do not go into a publication of the program.

Keys

Keys	What they do
Ctrl+1 … Ctrl+8	the sections in order
Ctrl+J	the journal
F1	this guide
Esc	from SETUP, SETTINGS and ABOUT - back; in CAMERA - back to the pointer
Space	CAMERA: play or stop the video
← →	CAMERA: a frame back and forward
Page Up, Page Down	CAMERA: the previous and the next recording
Delete	CAMERA: remove the chosen line or zone

What accuracy to expect

- Near the camera, where vehicles are large, the count agrees with a manual one within one or two percent (on the check recording: +0.8 % and -0.1 % on the two lines by the camera).
- Far away, where a vehicle is a few pixels, some vehicles are lost. A zone with enlargement and a larger model help; the smallest ramps of a low-resolution recording are beyond what the program sees.
- The classes are car, bus, truck, motorcycle. Cars are counted exactly. Buses and trucks by the detector alone roughly (on the check recording, at one line: buses -53 %, trucks +39 %); with a model of kinds taught on a camera of the same type, within a few percent (there: +6 % and +4 %; see TRAINING). Trucks are not split by the number of axles.
- The program does not read number plates.
- Different ways of counting (TensorRT, CUDA, DirectML) differ by single vehicles. Compare runs made on the same one.

Where things are

Folder	What is in it
`App\`, `Start.exe`	the window of the program
`runtime\`, `models\`, `Tools\`	Python with its libraries, model weights, FFmpeg - installed by the SETUP section
`models\classes\`	your own models of kinds for the TRAINING section
`system_core\kinds\`	the model of kinds that comes with the program
`classes\` beside the videos	the training data: pictures, sheets, labels, sets and copies of the models (or the folder chosen in SETTINGS)
`config\settings.json`	the settings of the program - always in its folder
the work folder	the list of recordings, the lines, the results, the manual counts and the journal - the four rows below. It is the folder of the program until another one is chosen in SETTINGS
`config\recordings.json`, `config\cameras\`	the list of recordings and the placements of lines, a file per recording
`output\<recording>\<date-time>_<model>\`	a run: `events.csv`, `summary.json`, the lines it was counted with; with a model of kinds - `crops\` (pictures of the vehicles) and `review.csv` (the marks)
`reference\`	manual counts to check against
`audion-traffic-counter.json` in the folder of the videos	the project file: a second copy of the lines, beside the recordings
`logs\`	the journal, a file per day

What the program does on the network and on the machine

It collects nothing and sends nothing. It goes to the network only during installation and at the first count with a new model - to download Python (python.org), the libraries (PyPI, pytorch.org), the model weights (GitHub), FFmpeg (gyan.dev, GitHub) and 7-Zip to unpack it (7-zip.org).

It installs nothing into the system: no services, no scheduled tasks, no registry entries. To remove the program, delete its folder.

If something goes wrong

What you see	What to do
"The counting engine is not installed"	the SETUP section, INSTALL WHAT IS NEEDED
"FFmpeg was not found", no video in CAMERA	the SETUP section, the FFMPEG button
SETUP asks to update the NVIDIA driver	update the driver and press CHECK AGAIN
a recording says "no lines", the queue says "will be skipped"	draw the lines in CAMERA
a line counts nothing	its arrow points the wrong way: turn it with the arrow button in the row of the line
a far line counts far fewer than there are	draw a zone with enlargement, take a larger model
"the edits of its lines are not saved … - the recording is not counted"	the recording has a line without a movement name, or not a single line is left: fix it in CAMERA and count it again
"the count failed" on one recording	the file cannot be read as a video, breaks off before its end, or its frame size differs from that of its lines; details are in the journal
"the engine was silent … - stopped"	the engine hung; the queue went on. Count this recording again; if it repeats, choose another way to count in COUNT
the count is slow	it runs on DirectML or on the processor: look at the "Count on" row in COUNT