Changelog

Contents

- 1.8.1 — 2026-10-09
- 1.8.0 — 2026-10-09
- 1.7.0 — 2026-10-08
- 1.6.0 — 2026-10-08
- 1.5.1 — 2026-10-08
- 1.5.0 — 2026-10-08
- 1.4.0 — 2026-10-08
- 1.3.0 — 2026-10-03
- 1.2.0 — 2026-10-03
- 1.1.0 — 2026-10-02
- 1.0.0 — 2026-10-02
- 0.11.2 — 2026-10-02
- 0.11.1 — 2026-10-02
- 0.11.0 — 2026-10-02
- 0.10.0 — 2026-10-02
- 0.9.0 — 2026-10-02
- 0.8.0 — 2026-10-02
- 0.7.0 — 2026-10-02
- 0.6.0 — 2026-10-02
- 0.5.0 — 2026-10-02
- 0.4.0 — 2026-10-02
- 0.3.0 — 2026-10-02
- 0.2.0 — 2026-10-02
- 0.1.0 — 2026-10-01

1.8.1 — 2026-10-09

Patch: an old NVIDIA card no longer gets an installation that cannot count.

- The PyTorch with CUDA the program installs carries code for NVIDIA cards from the GTX 10 series up. An older card (the GTX 900 series and before) was given that build all the same - there is an NVIDIA card, its driver is new enough - and the first count stopped with "no kernel image is available". Now such a card is treated as a machine without an NVIDIA card: PyTorch for the processor and DirectML, the count through DirectML. The machine check reads the generation of the card (`nvidia-smi`, compute capability) and says so; SETUP shows it as a row; the engine itself passes such a card over even when a folder installed for another machine is carried to it.
- The menu of `builder_main.cmd`: every line fits a console line; the fine points are in the README and the guides.
- Not checked on a card of that age: the rule was run with the generation of the card assumed (`Check-Machine.ps1 -AssumeCapability 5.2`) and by tests of the rule itself.
- RESULTS: the line of a run cut the name of the model of kinds short ("NVIDIA card, TensorRT · cla"); its column is wide enough for both now. CAMERA, English: the hint of the field of the oncoming movement did not fit the field ("do not coun"); it reads "ignored".
- Quieter outlines: the outline of a button, a switch and an input field - the plain grey one and the coloured ones (pressed, under the pointer, the green, amber and orange buttons) - stands halfway between its former colour and the surface, in both themes. Cards, text and icons keep their colours.
- The screenshot of the window is made on free material: a frame of a video under CC BY 3.0, with its credit on the picture and in the README. The window takes it itself: with `AUDION_SHOT_ONLY` the one pass is in the language and the theme named by `AUDION_SHOT_LANG` and `AUDION_SHOT_THEME`, and the language and the theme the user had chosen are put back.

1.8.0 — 2026-10-09

Minor: the installation asks first, and brings the one thing Windows may lack.

- **Nothing is downloaded without a question.** INSTALL WHAT IS NEEDED and the buttons of the parts ask before they download: what comes for this machine, how many gigabytes (about 15 GB of disk on an RTX card, about 5 GB without an NVIDIA card), what goes into Windows, and that without the engine the program does not count. A copy without its engine opens SETUP by itself at the first start and asks it there; before, it only showed a notice. The scenario `offer` of `--shots` presses the button and answers no.

- **The Microsoft Visual C++ runtime, first.** PyTorch cannot start without the Microsoft Visual C++ Redistributable (x64), and a freshly installed Windows may not have it. The installation looks for it before anything else and installs it with winget when it is missing (`install\Ensure-VCRedist.ps1`); Windows asks for the administrator's consent. It is the one thing the program installs into Windows. SETUP says so when the runtime is missing; when winget is not there either, it names the Microsoft download before anything starts. `verify` checks it.
- **The window is built without Python.** The translation table is checked by `Engine\Check-Texts.ps1` (PowerShell) instead of a Python script: `build_app` needs the .NET SDK and nothing else. The tests of the engine run where a Python is and are passed over with a word where none is - before, a machine with neither the engine's Python nor one of its own could not build the window.
- Not checked: the installation by winget on a machine that lacks the runtime (the machines of the workshop have it; the step was run with the runtime taken for missing and the command printed, and winget knows the package).

1.7.0 — 2026-10-08

Minor: a field for every kind of data and for the reports.

- The principle behind it: recordings and reports are not the main thing - they can be copied again and the counts reproduced; the main things are the program, what it has learned (the model of vehicle kinds, which comes with the program since 1.5.0) and its stability. So the program does not depend on where the data lie: SETTINGS names each kind.
- RECORDINGS, a new field: the folder the recordings lie in. The choice of files in the RECORDINGS section opens there, and - the part that matters - a recording of the list that is not where the list says is looked for under that folder and its subfolders, by its file name and its fingerprint, when the folder is chosen and at every start of the window. Recordings moved to another disk or copied anew are found again and keep their lines, their counts and their manual counts. A name borne by several files is taken only with a matching fingerprint, never on a guess.
- REPORTS, a new field: the folder the reports are written into. The report buttons of RESULTS open the choice of a folder there, and the last export sets the field as before.
- With WORK FOLDER, PROJECT FILE and TRAINING DATA these are the five fields of the data. The cards of SETTINGS stand in two columns - the fields of the data on the left - so the page does not grow taller than the window; a path stands on a line of its own under its buttons.
- Checked by the window's own scenario: a list brought from another machine named its recording on a disk that is not there; with the folder of the recordings named, it was found two folders down

and kept its lines; two files of one name without a fingerprint were not taken. Not checked: the folder dialogs themselves.

1.6.0 — 2026-10-08

Minor: the work folder - the program can be kept free of everything about the user's recordings.

- The list of recordings, the lines of every recording, the counts, the manual counts to compare with, the journal and the throwaway files are the user's own material. They lay in the folder of the program, and of them the cleanup before a publication removed the counts and the list but left the lines and the manual counts - with the names of streets and the numbers of sheets in them. SETTINGS has a card WORK FOLDER now: IN THE PROGRAM'S FOLDER (the default - a fresh copy works at once, nothing changes for anyone) or IN THE USER'S FOLDER. With a folder chosen, all of it is read and written there, in the same layout (`config\recordings.json`, `config\cameras\`, `output\`, `reference\`, `logs\`, `._runtime\`), and the folder of the program holds the program and its settings alone: it can be copied to another machine, cleaned or published without carrying the lists, the lines and the results along.
- One place decides where the work folder is, on each side: `work_root` in the engine (`engine\paths.py`) and `Paths.DataRoot` in the window; both read `"WorkFolder"` of `config\settings.json`, which always stays with the program. Every command that reads the list, the lines, the counts or the manual counts follows it (`recordings`, `frames`, `report`, `count` without `--out`).
- The folder is changed without a restart: lines drawn and not saved are written first, into the folder they were drawn in; then the list, the lines and the results are read from the new place. Refused while a count runs. Choosing a folder moves nothing - what lay in the former place stays there, and the tooltip of the button says which four things to move by hand to take the former work along. A work folder that cannot be reached (a disk that is not connected) is said at the start of the window and in the card, and the list is empty; nothing is written into the folder of the program instead.
- ABOUT says where the work folder is when it lies apart; its buttons LINES, RESULTS and JOURNALS open the folders of the work folder.
- Checked on the workshop's own material moved out of the program: the list of nine recordings read the same but for the folder of the lines; the report of the four counted recordings the same line for line; the window changed the folder to an empty one, to one with a list and back, and found everything as at the start. Not checked: the folder dialog itself, and a disk that is not connected.

1.5.1 — 2026-10-08

Patch: two things that went wrong silently when recordings and training data were moved to other folders.

- A report of a run counted before its recording was moved had no picture of the recording: the run names the file by the path it was counted from. The report now finds the recording through the list of recordings (it is there under the same id) and takes the frame from where the file lies today. A recording that is in neither place is reported as before, without the picture.
- `teach --boost PREFIX N` draws the pictures whose names start with PREFIX more often, and a picture of a set is named after the folder it was cut from. With the folder renamed, the old PREFIX named nothing, and the teaching ran to its end and gave another model than the one meant, without a word. A PREFIX that names no folder of `--train` is now refused before the set is built (a set is never written over, so its name would have been spent), and once more on the set itself; the `start` line of a teaching says how many pictures are boosted (`"boosted"`).

1.5.0 — 2026-10-08

Minor: a car of the detector is no longer renamed a truck, and the default model of vehicle kinds comes with the program.

- The rule "both" let the model of kinds rename a car of the detector when it was 90 % sure of another kind. Looked at by eye on a camera that sees the road from a roof: where the model was sure a detector's car was a bus it was right 21 times of 24 (route vans the detector cannot tell from cars at that height); where it was sure it was a truck - 10 of 42, the rest dark and red cars seen from above, and as sure of the wrong ones as of the right ones (0.90-0.97), so no threshold parts them. Cars are 96 % of such a flow: one wrong rename in three hundred cars is enough to spoil the count of trucks. Since this version a detector's car becomes a bus when the model is sure, and never a truck. What the detector itself called a truck or a bus is the model's to name as before, a truck included.
- Measured for the default model, former rule against the new one. The roof camera, counted exactly: 62 cars called trucks against none, trucks about +40 % against about +9 %; buses the same. City cameras of two days: trucks +8.7 % and -1.7 % against -5.5 % and -6.5 %. Roadside cameras at ground level: no change. The camera of the reference recording, the two near lines: trucks 50 and 31 against 49 and 31, where the pictures show 49 and 32. The reference recording itself was counted again: every event is the same, the rule touches only the names of the kinds.

- The former rule is kept to repeat old numbers: `--car-to-truck` on `count`, `classify` and `check-classes`. A run remembers the rule it was named by (`"car_to_truck"` in the `classes` block of `summary.json`), and a mark of a person laid on it later in TRAINING changes that vehicle only: the rest of the run is not named anew by another rule. A run named before this version is taken as named by the former rule.
- The file of the default model is part of the program: `system_core\kinds\classes-06.pt`, 6 MB, weights only - no picture of any camera is in it. `config\classes_default.json` named a model that was not in the published folder, so a folder installed from the skeleton named buses and trucks by the detector alone; now it counts with the model at once. The owner's word on publishing it: "of course - that is the point". A model of the same name in `models\classes` is the user's own and is taken first; TRAINING lists the models of both folders; `doctor` says which model is the default (`"kinds"`), and `builder_main.cmd verify` has a line for it. A cleanup leaves `system_core\kinds` in place.

1.4.0 — 2026-10-08

Minor: the report of a count for the people who use its numbers, and the model of vehicle kinds on by default.

- The owner's word: "we still have to work on the reports: making them, exporting them. And a BUTTON by which everything is exported (copied) somewhere. The staff understand only DOCX and XLSX... The main thing is to make it handsome, clear, with accents." RESULTS has a card REPORT: REPORT ON ALL RECORDINGS (the newest run of every counted recording) and REPORT ON THIS RUN ask for a folder and write two files there, REPORTS FOLDER opens the last one. The document (.docx): the figures first (recordings, vehicles, cars, buses, trucks, how many recordings are to be checked), the overview, then every recording on a page of its own - a frame with the counting lines and the names of the directions, the table by direction, the total, and the comparison with a manual count when `reference\` holds one. The sheet (.xlsx): the overview, the split by 15 minutes, the list of recordings (what was counted, when, on what, what to check), the comparison.
- The report speaks to a transport engineer, not to a programmer (the owner: "bring everything to an ergonomic form for transport engineers, not geeks"). It is titled "Интенсивность и состав транспортного потока"; beside every count stands the volume an hour (a recording of 15 minutes multiplied by four), under every total the composition of the flow in percent; a line is a "section", a recording is "processed"; the sheet by 15 minutes gives the volume an hour of every interval. What its reader has no use for - the counting model, the model of kinds, what it was computed on, the

frames, the folder of the run - is the last sheet of the workbook, "Служебное". No passenger-car equivalents: the norm their coefficients come from is the reader's to name.

- The report is the engine's: `main.py report --out DIR [--id ...] [--run ...] [--formats xlsx docx md] [--lang ru|en] [--interval 15]`, so an agent or a night script writes the same files the button does. Nothing is counted in it: the numbers are those of `summary.json` and `events.csv`. Markdown is a third format for whoever converts it himself.
- No library was added for .docx: `engine\docx.py` writes the few XML parts of the file directly (headings, tables with fills, pictures, a footer with the page number), so an installed folder writes reports without being installed again. `openpyxl` was pinned already. Both files were opened by Word and Excel and read page by page.
- A doubtful number is said to be one, in words and in colour, not delivered silently: a recording read not to its end; the kinds named by the detector alone; and - new - a model of kinds on a camera it does not know. The measure of the last: such a model renames many of the detector's cars (a model taught on views from above called 20 % of the cars of a roadside camera buses; a model that knows the camera renames 1-3 %). A run in which the model renamed at least 20 and at least 5 % of the detector's cars has `"doubt": true` in the `classes` block of `summary.json`; RESULTS says so in the line of the run, the report marks the recording "check" and explains that the number of vehicles stands and their division into kinds does not.
- The owner's word on the model of kinds: "of course, we torment the video card precisely so that it is on by default". `config\classes_default.json` names the default model; a count started from the window takes it when nothing else is chosen (TRAINING: AUTO, the first choice; DETECTOR is the former behaviour; a model by its name as before), and `count --classes auto` takes it from the command line. The name is written down, not guessed from the folder: `teach` puts a new model into `models\classes` before anyone has checked it. When the named model is not installed, the kinds are the detector's and the tooltip of AUTO and the report say so. The file of the model is not in the skeleton: it was taught on a customer's recordings and is published only by the owner's word.
- The window checks the two buttons by itself: `AUDION_SHOT_SCENARIO=report`.

1.3.0 — 2026-10-03

Minor: teaching a model of vehicle kinds for one's own cameras, from the program itself and together with an AI agent.

- The owner's word: the program goes public, and those who find it useful should be able to "run it at home with a video card with the help of Claude, as we did, and teach the kinds for their own

angles". The tools of the experiment that taught model 03 are now commands of the engine: `harvest` (pictures of every vehicle of a recording, no lines, a screen of several cameras by `--grid`), `sheets` (sheets of 48 numbered pictures), `labels` (the letters written beside the sheets), `teach` (the set and the teaching, `models\classes\<name>.pt`, never written over), `check-classes` (the detector, the model and the rule "both" against labels the model was not taught on). The marks of TRAINING (`review.csv` of a run) are labels as well.

- `Docs\AI_AGENT_GUIDE_EN.md` and `Docs\AI_AGENT_GUIDE_RU.md`, a guide for any AI agent (Claude Code, Codex, others) that works in the program's folder with its user: the rules, the folder, the commands, placing lines by frames, counting and comparing, and teaching the kinds step by step - the letters, what decides the result (pictures of the same kind of camera, the boundary of the kinds of the user's sheets), and that an agent asks before it looks at the pictures, because what it looks at goes to the service that runs it. It lies in `Docs\` because the public build carries no `SKILL.md` (the owner's word). The user guides have the chapter "Working with Claude and Codex", the readmes a paragraph, the tooltip of MODEL OF KINDS names the guide.
- Where the training data lie is a choice of SETTINGS, the card TRAINING DATA (the owner's word): BESIDE THE RECORDINGS - a folder `classes` beside the videos, the default, so the data travel with the recordings as the project file does - or IN THE USER'S FOLDER. `harvest` and `teach` take it from there; `teach` keeps the set and a copy of the model in `<training data>\taught\<name>\`, because `models\` goes with a cleanup of the program and the training data must not. They are never in the program's folder unless chosen so (`training\`, which the cleanup removes like `output\`).
- CAMERA: the round ends of a line or a gate are shown only on the chosen one and on the one under the pointer - on a placement that is only looked at they hid the vehicles under them; the direction is a small triangle standing a few pixels off the middle of the line instead of a stick with a head across the road (the owner's words). RESULTS say which model of kinds named the kinds of a run, in the list of runs and over its column of the comparison.
- Checked: 89 engine tests (seven new, of the letters, labels, sets and the place of the training data); the window's scenarios `camera` (with the pointer brought over a line) and `training`; `check-classes` with model 03 on the labelled check part of the experiment gives its table exactly; a minute of the reference recording cut, laid on sheets, labelled, taught for two epochs and checked by the commands on the laptop (DirectML for the detector, the processor for the teaching).

1.2.0 — 2026-10-03

Minor: the kinds of vehicles by a model of kinds, and the TRAINING section.

- The owner's task: "only three kinds are needed: cars, trucks, buses". The detector counts cars well and confuses buses with trucks; on the reference recording, by the line where the vehicles drive away from the camera, buses -53 %, trucks +39 % against a count made by looking at the pictures. A small model of kinds (MobileNetV3, taught on pictures of counted vehicles) names the kind of every counted vehicle on its pictures; the rule "both" keeps a car of the detector a car unless the model is 90 % sure of another kind. On the second half of that recording (the first half was in the teaching, the second was not) buses +4 %, trucks +6 %, cars 0 %; by the line where vehicles come towards the camera trucks -3 %, buses 5 of 6. How the model was taught: `MEMORY.md`, "Классы по вырезкам".
- `count --classes NAME` (a file of `models\classes`) and `classify --run DIR`: the kinds of a run named anew without counting the video again. `events.csv` keeps the detector's kind, the model's kind and share and a person's mark beside the final one; `summary.json` gets a block `classes`. A count without a model is as before, byte for byte.
- TRAINING, a section of the window (the owner named it): the model of kinds for the count (DETECTOR or a file of `models\classes`), and the pictures of the vehicles of a run that the detector and the model disagree on or the model is not sure of - 192 of 3756 on the reference recording. A button under a vehicle sets its kind; the run is named anew at once and RESULTS show it. The marks are the examples of the next teaching, which is done outside the program; a taught model comes as a new file of `models\classes`.
- The route van with windows along its side is a bus: the owner's word, and the way the people who counted by hand took it (with it the count by the pictures agreed with theirs within one or two vehicles).
- `models\classes` goes with the cleanup like the rest of `models\`: a model taught on the customer's recordings is not published.
- Checked: 82 engine tests (five new, of the kinds); the reference run named anew by the engine gives the numbers of the experiment exactly; naming anew twice gives the same `events.csv` byte for byte; one minute counted with `--classes` end to end; the window's scenario `training` (a mark by the path of a button, the run named anew, the mark taken back).

1.1.0 — 2026-10-02

Minor: a count can keep pictures of the vehicles it counted; and a count on DirectML with more than one zone no longer breaks.

- `count --crops N` keeps up to N pictures of every counted vehicle beside the run (`crops\`, `crops.csv`): the whole vehicle at its largest, cut by the box of the detector. The reason is the owner's task of 02.10.2026 - only three kinds are needed, cars, trucks and buses, and the detector confuses the last two (24 buses against 52 by hand on the near line). The pictures are what the kind is named by, by the eye and then by a small classifier taught on them; the detector itself is not touched. The count does not read the pictures: one minute of the reference recording gave the same `events.csv` byte for byte with and without the key. There is no switch in the window yet.
- DirectML, the fallback of a machine without an NVIDIA card: a count with two zones failed within seconds ("DmlCommandRecorder ... 80004005", shown as "'utf-8' codec can't decode byte 0xcd" - the text of the error comes in the language of Windows). Each zone has its own model, each model had its own lock, and two threads called the two models at once; the provider does not take that. One lock is now shared by all the models of a process. Found on the laptop with the Intel card on the folder installed anew from the skeleton: four counts of four failed in two threads, two of two passed in one; after the change three of three in two threads, the events the same as in one thread. TensorRT, CUDA and OpenVINO are not touched by it.
- Checked on the laptop, on DirectML: 77 engine tests (four new, of the choice of pictures), `verify`, `check_queue` (counted 3, skipped 1, failed 2, every count on `directml`, nine guard checks). Not checked: TensorRT and CUDA - the change does not touch their code, and the reference recording was not counted on this version.

1.0.0 — 2026-10-02

Major: the first version to be published. The number is the owner's word ("bump to v1.0.0"); against 0.11.2 the program removes its downloads after an installation.

- What is published is the skeleton `cleanup_project.cmd` leaves: the built window (`App\`, `Start.exe`), the code, the configs and the documents, 108 MB. Everything else - Python, PyTorch, TensorRT on an RTX card, the detector weights, FFmpeg - is installed by INSTALL WHAT IS NEEDED in the SETUP section; the .NET SDK is needed only to build the window.
- After a successful installation the pip cache and the downloaded archives (`install\download`, 3.6 GB) are removed - by the buttons of SETUP and by `builder_main.cmd install_all` alike; after a failure they stay, so that the next try does not download everything again. Nothing is cut

out of what is installed: the owner's rule ("nothing is cut out of what is installed, except the cache and the installation files").

- Checked on this version: 73 engine tests, the translation table (456 texts), `verify`, `check_queue` (counted 3, skipped 1, failed 2, every count on TensorRT, nine guard checks). On a clean copy of the skeleton the window pressed INSTALL WHAT IS NEEDED by itself: 6.1 minutes, "everything is in place", the downloads gone, 11.5 GB installed; `verify` and `check_queue` passed there, and `install_all` run over it removed the downloads the same way.

0.11.2 — 2026-10-02

Patch: a TensorRT engine file that cannot be loaded is built again.

- The installed folder is carried to another machine as it is: the libraries are the same for every card, and the engine file in `models\` names the card and the TensorRT it was built with - on another card it is built anew by the first count (checked with a note that named another card: 30 seconds, counted on `tensorrt`).
- The gap in that: an engine whose note names this card and which still cannot be loaded (another driver, a damaged file) sent every count to torch, for good - the file stayed. Now it is built again, once; torch is the step down only when that fails too. Checked with half an engine file: one count, and four at once - all on `tensorrt`.

0.11.1 — 2026-10-02

Patch: the first queue on a fresh installation was counted on torch instead of TensorRT.

- Found on the copy that was cleaned and built anew. The queue starts four counts at once; on a folder without a TensorRT engine all four exported the same weights into the same file at the same moment, the file came out broken, TensorRT refused it in every one of them and each stepped down to `cuda` (five times slower, and not the backend the numbers are checked on). Now one process at a time makes the files of a model - the downloaded weights, the ONNX export, the engine (`build_turn` in `engine\detector.py`, a lock on `models\<weights>.lock` that Windows lets go of when its process ends); the others wait and take what was made. The engine file appears whole or not at all.
- Checked by four counts started at once on a folder with no engine, and on one without the weights: all four on `tensorrt`. Three tests of the turn itself (73 engine tests).
- `check_queue` asks `doctor` what this machine counts on and fails when a recording of the queue was counted on something else: the defect above passed the check unnoticed.

- `--shots` has the `setup` scenario: INSTALL WHAT IS NEEDED pressed in the window, for a cleaned copy of the folder. Run on the skeleton left by `cleanup_project.cmd`: the window installed the environment, the model and FFmpeg in six minutes and said "everything is in place"; `verify` and `check_queue` passed on what it installed.
- SETUP: the number under INSTALL WHAT IS NEEDED counts the model that goes along with the environment (it said two parts and installed three).

0.11.0 — 2026-10-02

Minor: the four findings of the audit of 0.10.0 closed; a recording that breaks off is no longer taken for a counted one.

- AUD-002. A recording that cannot be read to its end was reported as counted, with the length its header promised (a test clip cut at 55 %: exit code 0, 113 frames of 200, the full length in the summary). Now the count compares the frames it read with the frames the file promised: short by up to a second or one per cent - the run is kept and marked (`complete` false in `summary.json`, a mark in RESULTS, a warning in the journal); short by more - nothing is written, the exit code is 1 and the reason comes as an `error` line. `--partial` counts what can be read. `seconds` is the length really read; `frames_expected` and `frames_read` stand beside it. The window takes a recording for counted only when the `done` line came and `summary.json` is there, and says the engine's reason when it did not.
- AUD-003. The rules of a placement were asked three different ways: the window, `recordings list` and the count each had their own. They are one table now (`placement_problem` in `engine\camera.py`), the count refuses whatever the table refuses (a line outside its zone was counted before), and the window holds the same table in the same order (`Camera.Broken`). `tests\placement_cases.json` is put to both.
- AUD-001. CAMERA: when another recording was chosen with unsaved lines, "save" was answered and the save failed, the window went on to the other recording and the edits were lost. It stays on the recording now, with the edits and the reason.
- AUD-004. `recordings add` looked for the lines of a recording beside the video first even when the project file is kept in a folder of the user; the window looked there first. Both look in the order of `KeepsakeMode` now.
- The checks that make each finding happen on purpose: 70 engine tests (27 before), the guard checks `05` and `rules`, a cut recording in the queue of `check_queue` - it must end with counted 3, skipped 1, failed 2.

- The header: a section button no longer has a least width. Four of them were wider than the piece they stand in, and the chosen one lay over the separator at its right.
- CAMERA: the round handle of the playback slider is a fifth smaller (16 against 20).
- The lists of formats in the documents name AV1 (read whole on a short test clip).

0.10.0 — 2026-10-02

Minor: one vehicle, one box; the COUNT section laid out anew.

- One vehicle, one box is the rule of the count now: overlapping boxes of different classes are not both kept. On the reference recording the near lines came from +6.8 % and +3.3 % of the manual count to +0.8 % (N10) and -0.1 % (P107); trucks on N10 from 101 to 68 against 64 by hand. The counts of every recording change with it: a run made before is marked in RESULTS ("the earlier rule of boxes"), and `--boxes-per-class` counts the old way. The owner's choice after the comparison.
- COUNT: THE COUNT AS IT GOES stands first, with THIS MACHINE and HOW TO COUNT beside it, and the queue below; the queue is the only part of the page that scrolls. THIS MACHINE is a badge: the processor, the memory, the video card with its memory, the driver, Windows, and the engine - the choice of "Count on" while nothing is counted, the backend a recording is really counted on during a count. The note "now this is..." beside the chips is gone.
- The numbers of a count are tiles of one line, the name and the value with a bar between them; "VEHICLES IN ALL" is "TOTAL". The blocks are filled in, not made anew at every line of the engine.
- CAMERA: the lines and the zones scroll inside one card, and the two columns end on one line.
- Fixed: a queue started before the engine had answered what it counts on took the recordings one at a time; the count now waits for that answer.
- Fixed: under load the picture in CAMERA could stay the same when the recording was changed (a late frame of the old recording took the frame of the new one). The `browse` scenario of `--shots` checks it.
- Fixed: `install\Check-Machine.ps1` is ASCII again (a Cyrillic word broke its parsing under Windows PowerShell); it reports the processor, the memory and Windows as well.
- The engine reports the first frame at once and measures the speed from it.

0.9.0 — 2026-10-02

Minor: the TensorRT backend, and the speed settings moved into SETTINGS.

- Backend `tensorrt` for NVIDIA RTX cards: the model built into one TensorRT engine per zone size (full precision), the frame turned into the blob and the head sieved on the card, the suppression of boxes in numpy. On an RTX 5070 with `yolo26s` the reference recording of 29 minutes is counted in 5.2 minutes against 24.4 on torch (140 frames a second against 30); the counts differ by one vehicle on a movement at most.
- `--device auto` on an NVIDIA card takes `tensorrt` when it is installed, else `cuda`. When the engine cannot be built, `auto` steps down to `cuda`, says so in the journal, and the summary of the run names the backend it was really counted on.
- Install: `install\Install-TensorRT.cmd` (the wheels of NVIDIA's package index and onnx, about 3 GB), the builder step `tensorrt` (24). `Check-Machine.ps1` decides by the card: an RTX card takes TensorRT, a GTX card counts on PyTorch with CUDA alone; `build_env` installs it where the check says so. `Verify.ps1` and `doctor` report it. SETUP has the TENSORRT button and marks it for the machine like the others.
- COUNT: the TENSORRT chip in "Count on"; the heads of runs name TensorRT.
- SETTINGS: the SPEED OF THE COUNT card - "Recordings at once", "Threads" and "Priority" moved there from COUNT (they are the work of the program, not a choice of a count).
- On TensorRT detector threads help: four by default (one - 102 frames a second, three - 146, four - 153, six - 149, eight - 136), the counts the same at any number.
- The engine option `--one-box`: boxes of different classes that overlap are not both kept (one vehicle, one box). Off by default until it is checked against the reference.
- The `start` line of a count now comes after the models are made, so it names the backend the count really runs on.

0.8.0 — 2026-10-02

Minor: the SETTINGS section; a control of the window moved.

- A new permanent button at the bottom, SETTINGS, between SETUP and ABOUT: how the program itself works. The PROJECT FILE card (where the project file with the lines is kept) moved there from SETUP.
- SETUP keeps only what belongs to the backend: the machine, the installers, what is installed. The owner's division: "in SETUP only what belongs to the backend; what belongs to the work of the program - SETTINGS".

- The sections are now seven: Ctrl+1 ... Ctrl+7; Esc leaves SETTINGS as it leaves SETUP and ABOUT.

0.7.0 — 2026-10-02

Minor: the queue counts several recordings at once on an NVIDIA card.

- COUNT: the "Recordings at once" spinner (1 to 8, four by default - the owner's number). On `cuda` the queue runs that many engine processes side by side; when one recording is through, the next of the queue takes its place. Other backends count one at a time, and the window says so beside the spinner.
- The count as it goes shows a block per recording being counted; the status line shows how many are through and the share of the whole queue.
- Measured on an RTX 5070 with `yolo26s` on one-minute pieces: one process 27 frames a second, two 51 over all, four 71, eight 80; the event tables of every piece were the same byte for byte at every level.
- The rules of the queue are unchanged: read again before every recording, unsaved lines written first, a failed or silent engine reported and passed, only STOP ends it.

0.6.0 — 2026-10-02

Minor: the total number of vehicles of a recording.

- The total of a recording: `summary.json` carries `total` (class -> vehicles, the sum over the movements, as the total column of a survey sheet) and `check`; the `progress` and `done` lines of a count carry it too. A run counted before this version has no total.
- A check line: a plain line marked `"check": true` (the CHECK LINE button in its row in CAMERA) counts a flow other lines count further on; it is shown in the results and left out of the total. Of a gate only the entry is in the total.
- RESULTS: the TOTAL row under the table of a run and under the runs side by side (with the difference from the manual total when the recording has a reference); a check line is named as one in its row; the list of runs shows the vehicles of a run.
- COUNT: the VEHICLES IN ALL tile in the count as it goes.
- `main.py compare` prints the total against the manual count and gives it as `total` in its JSON line.
- The first verification on an NVIDIA card (RTX 5070, driver 591.86): installation, the queue and the guard checks on `cuda`, the reference recording against the manual count; the `cuda` baseline is

saved in `reference\<camera>\`.

- The reference lines file: the ends of three lines are swapped so that the arrow of each shows its main flow (`P107` away from the camera, `N13` and `N9` to the right), and the near line `P107` is marked as a check line. The counts did not move.

0.5.0 — 2026-10-02

Minor: the project file beside the recordings, and the four findings of the audit of 0.4.0.

- The project file: the lines, gates and zones of every recording are also kept in `audion-traffic-counter.json` in the folder of the videos, by the fingerprint of the file. A recording added again, from another folder or on another machine, gets its lines back by itself. SETUP has the choice of the place: beside the recordings (the default) or one file in a folder of the user.
- Audit ATC-01: lines that cannot be saved keep their recording out of the count; it was counted by its old file while the editor showed other lines.
- Audit ATC-02: the run's `camera.json` is made before the count and the engine counts by that copy; it was copied at the end and could hold lines saved during the count.
- Audit ATC-03: closing the window writes unsaved lines, and asks before closing without lines that cannot be saved; they were lost in silence.
- Audit ATC-04: the queue is read again before every recording, so a mark changed during a count takes a recording out of the queue or into it; the window showed one queue and counted another.
- `builder_main.cmd check_queue` makes each of those four happen on purpose and checks the project file (the "guard" scenario).

0.4.0 — 2026-10-02

Minor: a fallback backend, so the program counts on a machine without CUDA.

- Backend `directml`: the model exported to ONNX and run by ONNX Runtime with the DirectML provider, on any DirectX 12 video card (Intel, AMD, NVIDIA).
- `--device auto` now picks by reliability: `cuda`, `directml`, `openvino-gpu`, `openvino-cpu`, `cpu`. OpenVINO stays as the last resort: the least reliable for a queue of many recordings.
- COUNT: a row of buttons "Count on" (AUTO, CUDA, DIRECTML, OPENVINO, PROCESSOR); what the machine has not got is dimmed; beside it, what AUTO means on this machine.
- Install: `install\Install-DirectML.cmd`, the builder step `directml` (23), the DIRECTML button in SETUP. The environment of a machine without an NVIDIA card gets DirectML in place of

OpenVINO; OpenVINO is installed only by its own step or button.

- `doctor` reports `onnxruntime`, `directml` and `backends`.
- RESULTS: the head of a run's column names what it was counted on.
- The engine can write where its threads stand (`AUDION_ENGINE_STACKS`).
- A new program icon: the camera frame, the counting line with its arrow, a car.
- Documents for people: `README.md`, `Docs\README_RU.md`, `Docs\README_EN.md` and the user guides `Docs\USER_GUIDE_RU.md`, `Docs\USER_GUIDE_EN.md`; F1 opens the guide.
- The builder: `check_queue` (15) - the window counts a queue of test recordings by itself; the menu headings in capitals. `main.py recordings add --lines NAME`.
- The check against a manual count takes movements of any name (it failed on a name that was not a letter followed by digits).

0.3.0 — 2026-10-02

Minor: the window is rebuilt around the owner's case - lines and gates placed on two hundred recordings one after another, then the queue counts by itself overnight.

- Every recording has a placement of its own (`config\cameras\<recording>.json`). The list of cameras and NEW CAMERA are gone: a recording without lines is drawn on at once. AS THE PREVIOUS ONE copies the lines of the recording above into this one's own file.
- A new recording is no longer given the lines of another one because the frame size matched: it comes without lines and is marked so in RECORDINGS and on the START button.
- CAMERA: previous / next recording under the video (Page Up / Page Down); going on saves the edits; the name of the recording and its place in the list stand above the video.
- A line counts one way, along its arrow. The arrow follows the hand that drew the line and is turned by two arrow buttons in the row of the line (in place of the two name fields and the swap button); a name for the oncoming flow is optional. The arrow on the frame is bolder, and the name of the movement stands beyond its head.
- The queue never stops by itself: a recording without lines is skipped, a recording the engine failed on is reported and the next one starts, an engine silent for 15 minutes is stopped and the next one starts; unsaved lines are written before the count; the last line says counted / skipped / failed. Windows is kept from idle sleep while it runs.
- Engine: the OpenVINO infer requests are created before the first frame (`Detector.prepare`). Created lazily by the worker threads, they now and then left the engine hung at the start on Intel graphics.

- Lines can be placed without the window, by an agent, for the owner to review: `main.py recordings add | list` and `main.py frames` (the middle frame of every recording with a coordinate grid, and with its lines drawn). A file `config\cameras\<id>.json` is the lines of the recording of that id. The window opens a recording on its middle frame.
- COUNT: the card says the count is starting while the model loads (it said nothing was being counted).
- ABOUT: the Audion mark.

0.2.0 — 2026-10-02

Minor: a new capability of the window, and the place of the results changed with it.

- History of runs. Every count of a recording is kept as a run of its own in `output\<recording>\<yyyyMMdd-HHmmss>_<model>\` (`events.csv`, `summary.json`, the camera file the run was counted with) instead of one result per recording that the next count replaced. A cancelled count leaves no folder. `summary.json` carries `finished`.
- RESULTS: the list of runs of the recording (date, model, backend, time, crossings; a run can be deleted); on the left the chosen run by movement and class and its manoeuvres; on the right, as a card of its own, the runs side by side - a column per run, the manual count when the recording has a reference, the difference in percent in colour.
- `main.py compare` takes several summaries at once (`--summary a b c`).
- COUNT: the model chips carry the Ultralytics names (YOLO26N, S, M, L, X) in place of БЫСТРО / ОБЫЧНО / ТОЧНЕЕ; each has a detailed tooltip: parameters, size of the weights, mAP on COCO, when to take it.
- Every tooltip of the window comes after 1200 ms (was 1500).
- `--shots`: `AUDION_SHOT_TIP` takes a picture with a tooltip open; `tips.txt` lists the delay of every tooltip.

0.1.0 — 2026-10-01

The first version: nothing was released before it.

- The counting engine (`system_core\`): detector and tracker, zones with enlargement, lines with two named directions, gates and manoeuvres, backends `cuda` / `openvino` / `cpu`, the commands `doctor`, `count`, `trails`, `compare`.

- The window (`Source\`, Avalonia 12): sections RECORDINGS, CAMERA, COUNT, RESULTS, SETUP, ABOUT; Russian and English; dark and light; the video with lines, gates and zones drawn and stretched with the mouse; FFplay with the lines over the video.
- The builder (`builder_main.cmd`), the launcher (`Start.exe`), the installers of the engine runtime and of portable FFmpeg.
- Installation by machine: `check_machine` reads the video cards and the NVIDIA driver, the FFmpeg release follows the driver, `install_all` installs a clean folder; the SETUP section marks in green only what this machine needs and installs it with one button.
- `cleanup_project.cmd`: the folder down to the published skeleton.