Audion Traffic Counter — руководство для ИИ-агента

Содержание

- Правила, которые действуют всегда
- Папка
- Как привести папку в рабочее состояние
- Команды
- Расстановка линий по кадрам
- Счёт и чтение прогона
- Обучение классов под камеры пользователя
- Обучение детектора для вида, которого он не видит
- Если что-то не так

Этот файл — для ИИ-агента, который работает в папке программы вместе с её пользователем: Claude Code, Codex или любого другого, кто умеет запускать команды и смотреть картинки. Пользователь запускает агента в папке программы и говорит: «Прочитай `Docs\AI_AGENT_GUIDE_RU.md`». Всё, что делает окно программы, идёт через команды движка, описанные здесь, поэтому агент может сделать ту же работу из командной строки.

Что делает программа: читает запись дорожной камеры, находит машины в каждом кадре, ведёт каждую от кадра к кадру и считает её один раз, когда она пересекает линию, проведённую по кадру, по стрелке линии. Итог — сколько легковых, грузовых, автобусов и мотоциклов сделали каждый манёвр. Линии записи хранятся в файле камеры; у каждой записи свой.

Люди работают с программой через окно, `Start.exe`, и руководство пользователя (`Docs\USER_GUIDE_RU.md`). Пути ниже — от папки программы.

Правила, которые действуют всегда

- **Записи — пользователя.** Они лежат там, где их держит пользователь: не копировать их в папку программы и никуда не отправлять.
- **Спросить, прежде чем смотреть.** Всё, что рассматривает агент — кадр для расстановки линий, лист вырезок для разметки, — уходит в сервис, на котором агент работает. До первой картинки спросите пользователя, можно ли это для его записей. Если нельзя — картинки размечает и проверяет пользователь, а агент запускает команды.
- **Движок — только `runtime\python.exe`.** В системном Python нужных пакетов нет, и ставить их туда нельзя.
- **Не запускать счёт очереди записей.** Это кнопка СЧИТАТЬ пользователя. Одну запись или её часть из командной строки — можно; прежде чем запускать то, что нагружает видеокарту, скажите, зачем и на сколько минут.
- **Ничего не писать поверх.** Модель, набор, пачка листов сохраняют свои имена; новое получает новое имя. Прогоны в `output\` и данные обучения — пользователя.
- **Закрыть окно** перед `recordings add`: окно тоже держит список.
- **Число — вместе с тем, на чём оно стоит**: «автобусов −6 %» без «из 51 автобуса» ничего не значит. Одна машина из двадцати — пять процентов.

Папка

Путь	Что в нём
`Start.exe`, `App\`	окно
`runtime\`	свой Python программы с библиотеками (ставит окно)
`models\`	веса детектора и то, что из них сделано; `models\classes\` — модели классов машин, обученные здесь
`system_core\kinds\`	модель классов машин, которая идёт с программой (модель по умолчанию)
`Tools\ffmpeg\bin`	FFmpeg
`system_core\main.py`	движок: все команды ниже
`config\cameras\<id>.json`	линии каждой записи (файлы камер)

Путь	Что в нём
`config\recordings.json`	список записей окна
`output\<запись>\<дата-время>_<модель>\`	прогон: `events.csv`, `summary.json`, `camera.json`; с вырезками — `crops\`, `crops.csv`; отметки — `review.csv`
`classes\` рядом с записями	данные обучения: вырезки, листы, метки, наборы (НАСТРОЙКИ, ДАННЫЕ ОБУЧЕНИЯ могут перенести их в выбранную папку)
`reference\<камера>\<запись>.json`	ручные счёты для сверки
`logs\`	журнал окна

Рабочая папка. `config\cameras`, `config\recordings.json`, `output`, `reference` и `logs` — собственный материал пользователя. По умолчанию они лежат в папке программы, а если в НАСТРОЙКАХ, РАБОЧАЯ ПАПКА выбрано В ПАПКЕ ПОЛЬЗОВАТЕЛЯ — в папке, которую он назвал (`"WorkFolder"` в `config\settings.json`; сами настройки всегда остаются у программы). Раскладка в обоих местах одна, и движок следует выбору сам: `recordings list` печатает полный путь файла линий каждой записи, `count` без `--out` пишет в `output` рабочей папки, `report` читает прогоны оттуда. Поэтому, прежде чем писать файл линий или искать прогон, прочитайте, где они лежат, из `recordings list` — не считайте, что в папке программы. Рабочую папку пользователя не переносить и эту настройку не менять, пока он не попросил.

Как привести папку в рабочее состояние

Программа может быть голым скелетом: окно, код, настройки — и ничего не установлено. `Start.exe`, раздел УСТАНОВКА, **УСТАНОВИТЬ НУЖНОЕ** ставит то, чего не хватает машине (Python, PyTorch под видеокарту, TensorRT на видеокарте NVIDIA RTX, веса детектора, FFmpeg), около шести минут и 6 ГБ скачивания. Кнопка сначала спрашивает и говорит, что придёт под эту машину. Первый шаг проверяет, есть ли в Windows Microsoft Visual C++ Redistributable, который нужен PyTorch, и ставит его через winget, если нет (`install\Ensure-VCRedist.ps1`; единственное, что ставится в Windows, с разрешения администратора). Из командной строки: `builder_main.cmd install_all`. Потом:

```
builder_main.cmd verify
runtime\python.exe system_core\main.py doctor
```

`verify` должен закончиться строкой «Everything is in place»; `backend` в ответе `doctor` — то, на чём машина считает: `tensorrt` (NVIDIA RTX, быстрый путь), `cuda` (другие видеокарты NVIDIA), `directml` (любая видеокарта с DirectX 12 — Intel, AMD — в несколько раз медленнее), `cpu`.

Команды

Все — `runtime\python.exe system_core\main.py <команда>`; каждая печатаемая строка — один объект JSON.

```
doctor                                  что установлено и на чём идёт счёт
recordings add ПАПКА_ИЛИ_ФАЙЛ [...]     добавить видео в список окна (окно закрыто)
recordings list                         все записи: id, размер, файл линий,
расставлены ли, что не так
frames [--id ID ...] [--at 0.5]         кадр из середины каждой записи в
._runtime\frames\<id>.jpg с сеткой
                                        в пикселях кадра; <id>.lines.jpg — с
нарисованными линиями
count --video ФАЙЛ --camera config\cameras\ID.json --out ПАПКА
      [--start C] [--end C] [--model yolo26s.pt] [--device auto]
      [--crops 3] [--classes ИМЯ] [--snapshot-every 10]
                                        счёт одной записи (или части); --crops
сохраняет вырезки каждой
                                        посчитанной машины, --classes называет классы
моделью
compare --summary ПРОГОН\summary.json [...] --reference reference\КАМЕРА\ИМЯ.json
                                        прогон (или несколько) против ручного счёта
classify --run ПАПКА [--classes ИМЯ]    переназвать классы посчитанного прогона без
пересчёта видео
harvest --video ФАЙЛ [--out ПАПКА] [--start C] [--end C] [--grid 2x2]
                                        вырезки всех машин, которые двигались; линии
не нужны
sheets --dir ПАПКА --name ИМЯ [--classes ...] [--sample N] [--lines ...]
                                        листы по 48 пронумерованных вырезок для
разметки глазами
labels --dir ПАПКА [...]                буквы, записанные рядом с листами ->
labels.csv
teach --name ИМЯ --train ПАПКА ... --val ПАПКА ... [--epochs 25] [--boost ПРЕФИКС N]
      [--minibus bus|car] [--light truck|car]
                                        модель классов -> models\classes\ИМЯ.pt
check-classes --classes ИМЯ --val ПАПКА ... [--car-to-truck]
                                        детектор, модель и правило «оба вместе»
против меток
report --out ПАПКА [--id ID ...] [--run ПРОГОН ...] [--formats xlsx docx md] [--lang
ru|en]
       [--sheets truck bus motorcycle car]
                                        посчитанные записи файлами для людей:
документ Word и ведомость
                                        Excel; без --id и --run — последний прогон
каждой записи списка;
                                        --sheets — пронумерованные картинки машин
этих видов по сечениям
```

```
detector harvest | marks | frames | tiles | pack | check | eye | join | teach | exam
| ship
                                        обучение детектора для вида, которого общий
не видит:
                                        «Обучение детектора для вида, которого он не
видит»
```

`--classes auto` берёт модель классов по умолчанию — ту, что названа в `config\classes_default.json`. Окно делает то же, когда в разделе ОБУЧЕНИЕ не выбрано иное. Её файл идёт вместе с программой (`system_core\kinds\`), поэтому только что установленная папка считает с ней; модель с тем же именем в `models\classes` — своя, пользователя, и берётся первой. Какая модель стоит по умолчанию, говорит `doctor` (`"kinds"`). Ключ `--car-to-truck` у `count`, `classify` и `check-classes` — правило классов, каким оно было до 1.5.0: чтобы повторить старые числа.

Отчёт. Когда пользователь просит результат, запишите его командой `report` в папку, которую он назовёт, и скажите, что в нём: сколько записей, сколько помечено «проверить» и почему. Запись помечается, когда она прочитана не до конца или когда её классы сомнительны — модель классов переназвала не меньше 20 машин и не меньше 5 % из тех, что детектор назвал легковыми (`cars`, `renamed`, `doubt` в блоке `classes` файла `summary.json`). Это признак камеры, на которой модель не училась: число машин у такого прогона верно, а деление на легковые, автобусы и грузовые — нет. Что с этим делать — раздел «Обучение классов под камеры пользователя»; выдавать такое деление за результат нельзя.

Расстановка линий по кадрам

Удобный способ подготовить много записей: агент расставляет линии по одному кадру каждой, пользователь проходит записи в окне (КАМЕРА, кнопки «предыдущая / следующая»), поправляет мышью и нажимает СЧИТАТЬ.

1. Закройте окно. `main.py recordings add <папка>`. Каждая запись получает `id` (имя файла без расширения, с номером, когда имена повторяются).
2. `main.py frames`. Посмотрите каждый `<id>.jpg`: кадр из середины записи (начало часто тёмное или трясётся), узкий кадр показан вдвое крупнее, на сетке — пиксели кадра: координаты читаются прямо с неё.
3. Для каждой записи напишите `config\cameras\<id>.json`:

```
{
  "name": "название точки съёмки, для людей",
  "frame_size": [640, 480],
  "zones": [
    { "id": "near", "rect": [0, 0, 640, 480], "scale": 1 },
    { "id": "far",  "rect": [0, 196, 640, 292], "scale": 2 }
  ],
  "lines": [
    { "id": "south", "label": "что считает", "zone": "near",
      "p1": [60, 300], "p2": [280, 300], "pos": "N1", "neg": "" },
    { "id": "gate-A", "label": "", "zone": "near", "p1": [360, 430], "p2": [600,
455],
      "pos": "A:in", "neg": "A:out", "kind": "gate", "branch": "A" }
  ]
}
```

- `frame_size` — размер, который напечатал `frames`; координаты — пиксели исходного кадра, x от левого края, y от верхнего.
- По линии на каждый поток, который считает обследование: поперёк проезжей части под прямым углом к движению, там, где машины крупные и не заслоняют друг друга, — не на стоп-линии, не за столбом, не у самого края кадра (след должен устояться до линии).
- Направление: под именем `pos` линия считает машины, которые оказываются слева от `p1 -> p2`. Проведённая снизу вверх — считает едущих вправо; сверху вниз — влево; справа налево — вверх по кадру; слева направо — вниз. `neg` оставьте пустым, если встречный поток на этой линии не считается; если считается — `pos` у большего потока.
- Линия перед развилкой, у ветвей которой есть свои линии, — контрольная: `"check": true`, иначе её машины войдут в итог дважды.
- Имена — названия манёвров из ведомости пользователя, если они известны; иначе `N1`, `N2`...
- Перекрёсток или кольцо: ворота поперёк каждой ветки, через въезд и выезд разом (`"kind": "gate"`, `"branch": "A"`, `pos` / `neg` = `A:in` / `A:out`); движок сводит въезды и выезды в манёвры «A → B».
- Дальняя часть, где машины в несколько пикселей: зона вокруг неё со `scale` 2 и дальние линии в этой зоне. Без нужды зон не заводить.

4. Снова `main.py frames`. У каждой записи `problem` должен быть `null`, `placed` — true. Посмотрите каждый `<id>.lines.jpg`: линия лежит поперёк своей дороги, стрелка смотрит туда, куда едут считаемые машины. Поправьте и повторите.

5. Записи с одной точки съёмки делят расстановку: скопируйте файл под другим id и проверьте копию на её кадре — штатив двигают.

6. Скажите пользователю, что расставлено и что нет, с путями к картинкам `.lines.jpg`. Чего по кадру не понять — куда идёт движение по дороге без единой машины, какие манёвры в его ведомости, — спросите, а не угадывайте: линия со стрелкой не туда за ночь не посчитает ничего.

Счёт и чтение прогона

Одна запись для проверки:

```
main.py count --video <файл> --camera config\cameras\<id>.json --out output\<id>\
<дата-время>_yolo26s
```

Прогон пишет `events.csv` — по строке на каждый посчитанный проезд: `frame`, `time`, `movement`, `class`, `line`, `zone`, `track` (с моделью классов ещё `detector_class`, `classifier_class`, `classifier_share`, `mark`), — и `summary.json`: `summary` (манёвр -> класс -> машин), `total` (вся запись без контрольных линий и выездов из ворот), `manoeuvres`, `backend`, `complete` (false — запись прочитана не до конца). Прогон повторяется байт в байт на том же пути счёта; разные пути различаются на единицы машин.

Ручной счёт для сверки — `reference\<камера>\<запись>.json`:

```
{
  "classes": ["Легковой/Микроавтобус", "Автобус", "Малый грузовой", "Средний
грузовой", "Большой грузовой", "Спец"],
  "engine_classes": { "car": [0, 5], "bus": [1], "truck": [2, 3, 4] },
  "counts": { "N1": [1073, 12, 28, 16, 20, 0] },
  "derived": { "P1": ["N2", "N3"] }
}
```

В `counts` — по числу на столбец ведомости пользователя, `engine_classes` складывает столбцы в классы программы, `derived` задаёт контрольные манёвры как суммы. `main.py compare` печатает по каждому манёвру и классу: вручную, программа, разница, проценты.

Ручной счёт — точка отсчёта, а не истина. Люди точно считают хорошо видные машины, а остальное прикидывают. Где прогон и ручной счёт расходятся по классу, разложите машины этой линии на листы (счёт с `--crops 3`, потом `sheets --dir <прогон> --lines <id линии> -`

`--classes bus truck` и выборка легковых), разметьте и посчитайте по картинкам — получится третий счёт, который может проверить кто угодно. На записи, на которой делалась программа, такой счёт сошёлся с ручным до одной-двух машин на ближних линиях — когда маршрутки посчитали автобусами, как их считали люди.

Обучение классов под камеры пользователя

Детектор (YOLO, обученный на фотографиях со всего мира) хорошо называет легковые и плохо — остальное: маршрутка у него то автобус, то грузовая, длинный грузовик — автобус. Модель классов — MobileNetV3-small, 6 МБ — смотрит на три вырезки каждой посчитанной машины и называет её класс. С детектором её соединяет правило «оба вместе»: что детектор назвал легковой, остаётся легковой, пока модель не уверена на 90 %, что это автобус, — а грузовой легковая детектора не становится никогда; то, что детектор назвал грузовой или автобусом, называет модель; мотоцикл остаётся за детектором. Так модель исправляет автобусы и грузовые и не портит легковые. (Почему грузовой — никогда: с высокой камеры тёмные легковые кажутся модели грузовиками, и уверена она в неверных так же, как в верных; легковых девять машин из десяти, и нескольких неверных переназваний хватает, чтобы испортить счёт грузовых.)

Модель идёт вместе с программой и включена с самого начала. Она обучена примерно на 4 300 размеченных машинах городских камер — со столба, с моста, с обочины, с крыши, летом и зимой. Свою модель учат, когда модель по умолчанию плохо называет машины на камерах пользователя: сначала проверьте её (шаг 5 ниже) на нескольких сотнях его размеченных машин.

Замер на записи, на которой делалась программа, на одной линии: у детектора автобусы −53 %, грузовые +39 %; с моделью, обученной на первой половине той же камеры, на второй половине — автобусы +4 %, грузовые +6 %, легковые 0 %.

Решают две вещи, а не число эпох и не модель крупнее:

- **вырезки с камер того же вида.** Модель, обученная на 1400 машинах городских камер, снятых сбоку и сверху, называла грузовики камеры, которая видит их спереди, легковыми (−56 %). Около двухсот машин этой камеры это исправили. Учите на камерах пользователя и проверяйте на их машинах, которых модель не видела;
- **граница классов, как её проводит ведомость пользователя.** Люди, которые считают вручную, относят маршрутку (окна по всему борту) к автобусам; размеченная легковой, она дала автобусов −75 %. Прежде чем размечать, спросите пользователя, как его ведомости считают маршрутки и малые грузовые.

Время: дольше всего нарезка (20 минут записи 1080p с четырьмя камерами — 80 минут на графике Intel, минуты на RTX); обучение 25 эпох на 3500 вырезках — полчаса на процессоре ноутбука и несколько минут на видеокарте NVIDIA. Первое обучение скачивает веса ImageNet (10 МБ).

Порядок

1. **Вырезки.** Нарежьте все машины части записи — линии не нужны:

   ```
   main.py harvest --video <файл> --start 0 --end 1200
   ```

   (в `classes\cut\<имя записи>\` рядом с записью; запись экрана с несколькими камерами — `--grid 2x2`, каждая камера в свою папку `11 12 21 22`). Или посчитайте запись с `--crops 3`: в папке прогона будут вырезки посчитанных машин. Возьмите две разнесённые по времени части — две записи, два дня, две половины: на одной учить, на другой проверять.

2. **Листы.** Для каждой папки вырезок — сначала всё, что детектор назвал не легковой, потом выборку легковых:

   ```
   main.py sheets --dir <папка> --name big --classes bus truck motorcycle
   main.py sheets --dir <папка> --name car --classes car --sample 48
   ```

   Лист — `<папка>\sheets\<имя>_<nn>.jpg`: до 48 вырезок рядами по 8, ячейки пронумерованы с

   1. Автобусы и грузовые — примерно каждая десятая машина; почти все они в первой пачке.

3. **Метки.** Посмотрите лист и запишите рядом `<имя>_<nn>.txt`: по букве на ячейку, в порядке ячеек, по восемь букв в строке (пробелы и переводы строк ничего не значат; букв должно быть столько же, сколько ячеек).

Буква	Метка	Что это
`c`	car	легковая, внедорожник, пикап, минивэн, скорая, полиция
`v`	minibus	маршрутка: окна по всей верхней половине борта
`l`	light	малый грузовой: фургон без окон по борту, бортовая, будка
`t`	truck	грузовик крупнее: грузовой, самосвал, тягач с прицепом

Буква	Метка	Что это
`g`	cargo	точно грузовой, малый или большой — не разобрать
`b`	bus	автобус
`o`	trolley	троллейбус
`r`, `m`	tram, moto	трамвай, мотоцикл
`x`	junk	не разобрать, две машины в одной рамке, не машина

Сомневаетесь — `x`: неверная метка учит хуже, чем отсутствующая. Потом `main.py labels --dir <папка>` собирает буквы в `<папка>\labels.csv` и говорит, сколько каких меток и сколько ячеек ещё ждут букв. Прежде чем идти дальше, покажите пользователю лист-другой со своими буквами. Отметки, которые пользователь ставит в разделе ОБУЧЕНИЕ окна (`review.csv` прогона), — тоже метки.

4. **Обучение.**

```
main.py teach --name <камеры>-01 --train <папки для обучения> --val <папки для проверки>
```

Маршрутка идёт к автобусам, малый грузовой — к грузовым; `--minibus car` и `--light car` переносят их — спросите пользователя. Когда своих вырезок пользователя в `--train` мало рядом с другими, берите их чаще: `--boost <начало имени их папки> 4` (вырезки набора называются по папке, из которой они нарезаны; ключ, который не называет ни одной папки из `--train`, отклоняется). Лучшая эпоха выбирается по папкам `--val`; набор и копия модели хранятся в `classes\taught\<имя>\`, модель — `models\classes\<имя>.pt`.

5. **Проверка** на размеченных машинах, на которых модель не училась:

```
main.py check-classes --classes <имя> --val <папки>
```

Три таблицы — детектор, модель одна, правило «оба вместе» — и сколько машин каждого класса дал бы счёт этих папок против того, сколько их на самом деле. Важно последнее, столбец «both». В пределах 10 % по каждому классу — хороший результат; скажите, на скольких машинах стоит каждый процент.

6. **Работа.** В окне: ОБУЧЕНИЕ, МОДЕЛЬ КЛАССОВ, имя модели — следующий счёт называет классы ею. Прогон, посчитанный раньше с `--crops`, переназывается без пересчёта видео: `main.py classify --run <папка прогона> --classes <имя>`. Спорные машины прогона (детектор и модель разошлись или модель не уверена) — в разделе ОБУЧЕНИЕ, пользователь отмечает их кнопкой.

7. **Лучше.** Посмотрите, на чём модель ещё ошибается в проверочных папках, разметьте больше таких машин с камер пользователя, обучите новую модель под новым именем и проверьте на тех же папках. Остановитесь, когда проверка в пределах цели: больше эпох не помогает, больше вырезок нужной камеры — помогает.

Обучение детектора для вида, которого он не видит

Общий детектор (`yolo26s.pt`) пропускает целые потоки на некоторых видах: дальнюю проезжую часть за отбойником, где от легковой видна крыша в отдельных кадрах, и неосвещённую дорогу ночью, где от машины видны фары. С программой идут два обученных детектора, в `system_core\detectors\`: `yolo26s-barrier-01.pt` (машины за отбойником, днём) и `yolo26s-night-01.pt` (ближняя проезжая часть неосвещённой дороги ночью; дальнюю за отбойником он по-прежнему пропускает). Обученный детектор работает только в зоне, которая его называет, — `"model": "yolo26s-barrier-01.pt"` у зоны в файле камеры, — остальные линии сохраняют числа общего детектора. В такой зоне легковая детектора остаётся легковой (модель классов принимала крыши за автобусы); то же у линии с `"keep_car": true`. Зона задаётся в файле; окно показывает её детектор в строке зоны.

Если ни один не подходит к виду пользователя, детектор учат под этот вид. Сначала — слово пользователя: это часы работы видеокарты, а при разметке глазами агент смотрит много кадров его записей (см. «Правила, которые действуют всегда»). Порядок:

1. **Сначала экзамен.** Выберите два-три окна по 90 с, где вид не считается, посчитайте их глазами (две узкие полосы по обе стороны линии, кадр за кадром) и общим детектором (`detector exam --model yolo26s.pt`). На этих окнах не учат никогда: каждый шаг ниже, который берёт кадры, получает их в `--skip`.

2. **Самообучение** — когда детектор машины видит, только слишком слабо, чтобы их вести:

```
main.py detector harvest --video ЗАПИСЬ --out weak\r1.csv --start С --end С --skip
2940-3030
main.py detector marks   --weak weak\r1.csv --out marks\r1.json
main.py detector frames  --video ЗАПИСЬ --marks marks\r1.json --out frames --name
r1 --every 5
main.py detector tiles   --frames frames --out sets\self
```

Его рамки при низкой уверенности сшиваются в следы; след, который длится и хоть раз был уверенным, — машина в каждом своём кадре, а кадр с уверенной рамкой вне всех следов отбрасывается. Кадры режутся на плитки 640x640 в своём масштабе. Не помогает там, где детектор машину не видит вовсе (тёмная ночь): тогда — разметка глазами.

3. **Разметка глазами**:

```
main.py detector pack  --video ЗАПИСЬ --out label\r1 --start С --end С --count 20
                       --models yolo26s.pt --skip 2940-3030
main.py detector check --pack label\r1 --frame fNNNNNN
main.py detector eye   --video ЗАПИСЬ --pack label\r1 --out sets\eye --name r1 --
copies 8
```

`pack` пишет на каждый кадр полосу дороги, высветленную, со шкалой в пикселях кадра и пронумерованными догадками детекторов (`fNNNNNN.jpg`, `fNNNNNN.json`), и ту же полосу чистой. Размечающий пишет `fNNNNNN.marks.json`: `{"vehicles": [{"from": 2}, {"from": 4, "box": [x1, y1, x2, y2]}, {"box": [...], "kind": "truck"}], "unsure": [[x, y]]}` — догадка оставлена, догадка с исправленной рамкой, машина, которой нет среди догадок; неверная догадка просто не перечисляется; то, что не различить, — в "unsure", это место в плитке закрашивается. `check` рисует метки на чистой полосе: у каждой машины рамка, каждая рамка на машине. Машина, видная только по фарам: рамка вокруг пары фар плюс шестая часть их ширины с каждой стороны, высотой около 0,7 ширины у легковой. Отражения на дороге и на отбойнике, лучи и фонари — не машины. `eye` режет плитки из кадров как они есть — тёмных, какими их видит счёт — и повторяет их, чтобы несколько сотен кадров не утонули в остальном.

4. **Обучение, экзамен, установка**:

```
main.py detector join  --out sets\eye --sets sets\self --every 2
main.py detector teach --set sets\eye --name view-01 --epochs 8 --runs runs
main.py detector exam  --video ЗАПИСЬ --camera config\cameras\ID.json
                       --model runs\view-01\weights\best.pt --windows 2940-3030 --
out exam\view-01
main.py detector ship  --weights runs\view-01\weights\best.pt --name yolo26s-view-
01
```

`teach --from` продолжает от детектора, обученного раньше. Экзамен — настоящий счёт окон шага 1 обученным детектором в зоне `--zone` (без неё — во всех зонах): он должен оказаться ближе к счёту глазами, чем общий детектор, а линии, которые он не должен двигать (ближняя сторона, светлое время), — остаться прежними. Числа, которые печатает обучение, — не результат. `ship` кладёт веса в `system_core\detectors` под новым именем и никогда поверх файла.

Если что-то не так

Признак	Что делать
`camera lines are drawn for (640, 480), the video is (...)`	у записи другой размер: ей нужен свой файл камеры
`cannot open video`	неверный путь или файл, который не читает OpenCV: проверьте `Tools\ffmpeg\bin\ffprobe.exe`
дальняя линия почти ничего не считает	машины там слишком мелкие: зона со `scale` 2 или модель крупнее (`--model yolo26m.pt`)
на разных машинах числа отличаются на несколько машин	разные пути счёта; сравнивайте прогоны одного пути
`DirectML is not installed`	УСТАНОВКА, кнопка DIRECTML, или `builder_main.cmd directml`
`teach` отказывает: no labelled bus, truck	разметьте их больше: `sheets --classes bus truck` на других вырезках
`labels` отказывает: letters for cells	число букв в `.txt` не равно числу ячеек его листа
модель портит легковые	она учена на чужих камерах: разметьте около двухсот машин этой и обучите заново

Признак	Что делать
компьютер засыпает во время долгой нарезки	`harvest` и `teach` сами не дают машине уснуть; закрытая крышка усыпит всё равно