SmartCamera

Руководство пользователя

Даниил Грабарь

Содержание

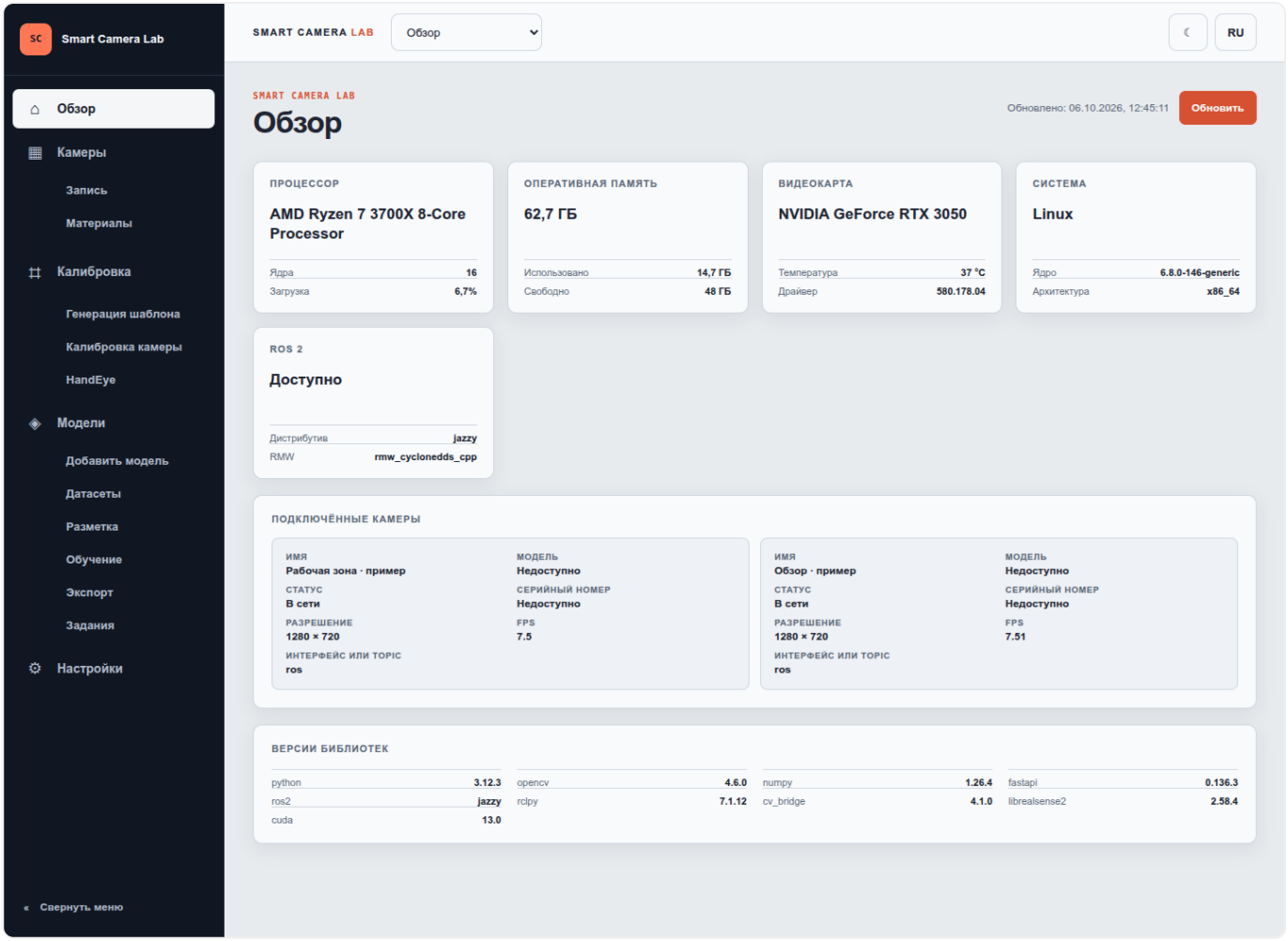
1. Начало работы 3
1.1 SmartCamera 3
1.2 Команда smartcamera 6
1.3 Установка из исходников 13
1.4 Запуск в Docker 16
1.5 Первый запуск 20
1.6 Интерфейс Smart Camera Lab 24
2. Камеры и запись 30
2.1 Камеры и источники изображения 30
2.2 Фото, видео и ROS bag 34
3. Калибровка 40
3.1 Шаблон и внутренняя калибровка камеры 40
3.2 HandEye: привязка камеры к роботу 46
4. Модели 51
4.1 Модели: библиотека и ML-окружение 51
4.2 Запуск модели на камере 57
4.3 Датасеты и разметка 61
4.4 Обучение, валидация и экспорт 67
5. ROS 2 и API 73
5.1 Топики и сервисы ROS 2 73
5.2 Python-клиент 79
5.3 HTTP API 82
6. Справочник 86
6.1 Конфигурация 86
6.2 Диагностика проблем 91

1. Начало работы

1.1 SmartCamera

SmartCamera подключает камеры к ROS 2 и даёт веб-интерфейс для просмотра, записи, калибровки и распознавания. Через один сервер можно работать с обычными USB-камерами, Intel RealSense и изображениями, которые уже публикует другой ROS-узел.

В этом руководстве установка разобрана с первого запуска. Для действий в браузере указаны названия разделов и кнопок. Команды терминала можно скопировать кнопкой в правом верхнем углу блока.



Страница «Обзор»: состояние компьютера, ROS 2, подключённых камер и версии библиотек.

1.1.1 С чего начать

Что нужно сделать	Где описано
Установить проект на новый компьютер	Установка через CLI
Увидеть первый кадр	Первый запуск
Подключить USB-камеру, RealSense или ROS-топик	Камеры
Записать снимки, видео или ROS bag	Запись и материалы
Распечатать мишень и определить параметры объектива	Калибровка камеры
Определить положение камеры относительно робота	HandEye
Запустить YOLO на видеопотоке	Модели, затем запуск на камере
Подготовить свои изображения и обучить модель	Датасеты, обучение
Подключить программу на Python или другой ROS-узел	ROS 2, Python-клиент

1.1.2 Как связаны части программы

Драйвер получает изображения с устройства и публикует их в ROS. Веб-сервер подписывается на эти изображения. Браузер получает предпросмотр от веб-сервера, поэтому на компьютере с браузером устанавливать ROS не нужно.

Распознавание выполняется в отдельном процессе Python. У модели есть запись в библиотеке, а у её подключения к камере — собственный запуск с FPS, устройством вычисления и порогами. Одни веса можно использовать на нескольких камерах; каждый такой запуск имеет собственное состояние.

ROS-команды и веб-панель обращаются к одним настройкам. Если модель включена через сервис, она также будет включена в карточке камеры. Закрытие вкладки браузера не выключает сервер, запись или запущенное распознавание.

1.1.3 Что понадобится

Рабочая среда проекта — Linux с ROS 2 Jazzy и Python 3.12. Для обычной камеры достаточно CPU; GPU нужен, когда выбранной модели или частоте обработки не хватает производительности процессора. Для просмотра интерфейса используйте современный браузер. Точные зависимости и порядок подключения окружений приведены в установке через CLI.

В примерах адрес сервера — `http://127.0.0.1:8000`. При подключении с другого компьютера вместо `127.0.0.1` укажите адрес компьютера, на котором работает SmartCamera. Пути к устройствам и каталогам относятся к серверу. Например, `/dev/video0` — камера Linux-сервера, а не веб-камера ноутбука, с которого открыт браузер.

1.1.4 Об иллюстрациях

Скриншоты сделаны с интерфейса этой версии SmartCamera. Камеры с подписью «пример» показывают синтетическое изображение, а записи модели и датасета подготовлены для демонстрации расположения элементов. Эти изображения не показывают качество работы YOLO или результат реальной калибровки. Числа загрузки компьютера, FPS и идентификаторы в вашей установке будут другими.

В руководстве сохранены названия кнопок из русской версии интерфейса.

1.2 Команда smartcamera

`smartcamera` устанавливает зависимости, сохраняет параметры запуска и управляет приложением из терминала. Самой команде не нужен ROS: его установка выполняется отдельно, при подготовке локального режима. Для Docker ROS устанавливается внутри образа. Установщик рассчитан на Linux; автоматическая локальная установка — на Ubuntu 24.04 с ROS 2 Jazzy. Для Linux Mint с базой Noble установщик использует зависимости Ubuntu Noble; другие базы не поддерживаются.

1.2.1 Быстрая установка

Для загрузки проекта нужны `curl`, Git и Python 3.10 или новее с модулем `venv`. На Ubuntu 24.04 и Linux Mint на базе Noble установите их:

```
sudo apt update

sudo apt install -y curl git python3 python3-venv
```

Скачайте и запустите установщик из основного репозитория:

```
curl -fsSL https://gitverse.ru/api/repos/daniel-robotics/SmartCamera/raw/branch/master/install.sh \
  | bash
```

Установщик скачивает ветку `master` из SmartCamera на GitVerse, сохраняет исходники в `~/.local/share/smartcamera/source` и создаёт команду `~/.local/bin/smartcamera`. Для CLI используется отдельное окружение Python в `~/.local/share/smartcamera/cli-env`.

Добавьте команду в PATH текущего терминала и запустите мастер:

```
export PATH="$HOME/.local/bin:$PATH"

smartcamera setup

smartcamera run
```

Для первой проверки выберите в мастере `local` и `cpu`; для контейнера — `docker` и `cpu`. Остальные значения можно оставить по умолчанию. После установки зависимостей и сборки команда `run` запустит приложение. Первая подготовка требует интернета, нескольких гигабайт свободного места и времени на загрузку библиотек и сборку.

Откройте http://127.0.0.1:8000, перейдите в **Камеры → Добавить камеру** и настройте первый источник по инструкции подключения. Для остановки запуска в терминале нажмите `^ Ctrl` + `C`.

Сам `install.sh` устанавливает только команду CLI. ROS, библиотеки приложения или Docker подготавливаются при `setup`, `local-setup` либо `docker-setup`. Запускайте установщик обычным пользователем; мастер вызовет `sudo` для необходимых системных действий.

Доступ к команде в новых терминалах

Если `~/.local/bin` ещё не входит в PATH, добавьте в `~/.bashrc` строку:

```
export PATH="$HOME/.local/bin:$PATH"
```

Затем откройте новый терминал и проверьте:

```
smartcamera --help
```

Если исходники уже скачаны

Перейдите в корень вашего проекта и установите CLI из этого каталога:

```
cd ~/dev/SmartCamera
bash install.sh --source "$PWD"
export PATH="$HOME/.local/bin:$PATH"

smartcamera setup

smartcamera run
```

Замените `~/dev/SmartCamera` своим путём. В режиме `--source` установщик не выполняет Git-команды: выбранная ветка и локальные изменения сохраняются.

Для ручного клонирования:

```
mkdir -p ~/dev
git clone --branch master https://gitverse.ru/daniel-robotics/SmartCamera.git ~/dev/SmartCamera
```

После клонирования выполните команды установки из каталога выше. Для другой опубликованной ветки замените `master` в команде `git clone`.

Другая ветка и каталоги установки

Чтобы передать параметры установщику по прямой ссылке, используйте `bash -s --`. Например, установка ветки `dev` в отдельный каталог:

```
curl -fsSL https://gitverse.ru/api/repos/daniel-robotics/SmartCamera/raw/branch/master/install.sh \
  | bash -s -- --branch dev --install-dir "$HOME/tools/smartcamera-dev" --bin-dir "$HOME/bin"
```

Ветка должна существовать на GitVerse и содержать CLI. Здесь сам установщик загружается из `master`, а исходники приложения — из указанной ветки. Команда появится в `~/bin/smartcamera`; добавьте `~/bin` в PATH, если его там нет.

Для имеющихся исходников те же параметры задаются так:

```
bash install.sh --source "$PWD" \
  --install-dir "$HOME/tools/smartcamera" \
  --bin-dir "$HOME/bin"
```

`--install-dir` задаёт каталог CLI-окружения и скачиваемых исходников, а `--bin-dir` — место запускаемой команды. Каталог пользовательских данных выбирается отдельно при настройке приложения.

Повторная установка и перенос

Если при установке по ссылке каталог `~/.local/share/smartcamera/source` уже существует, установщик остановится. Для обычного обновления используйте команды обновления. Для восстановления команды из уже скачанного проекта:

```
bash "$HOME/.local/share/smartcamera/source/install.sh" \
  --source "$HOME/.local/share/smartcamera/source"
```

Повторная установка из исходников обновляет запускной файл CLI. Существующую чужую команду с тем же именем установщик не перезаписывает.

Путь к исходникам записывается в запускной файл команды. Перед переносом каталога остановите приложение через `smartcamera stop`. После переноса переустановите CLI и обновите путь в сохранённых настройках:

```
bash /новый/путь/SmartCamera/install.sh --source /новый/путь/SmartCamera
smartcamera configure --yes --source /новый/путь/SmartCamera
```

Установщик обновляет путь загрузки CLI, а `configure` — путь приложения. Для локального режима после переноса повторите `smartcamera local-setup --yes`: окружения Python и сборка ROS могут содержать старые абсолютные пути.

1.2.2 Первая настройка

```
smartcamera setup
```

Мастер последовательно спрашивает:

Параметр	Что выбрать
Режим	`local` для запуска на хосте или `docker` для контейнера
Профиль моделей	`cpu` по умолчанию; `cuda` для совместимой NVIDIA GPU
Каталог данных	Постоянное место для камер, записей, калибровок и моделей
IP веб-сервера	`127.0.0.1` для этого компьютера; `0.0.0.0` для доступа по сети
Порт	По умолчанию `8000`
ROS_DOMAIN_ID	То же число, что у других ROS-узлов, от 0 до 232

После ввода мастер устанавливает зависимости и собирает выбранный режим. Локальная установка использует `sudo` для APT и правил udev; терминал может попросить пароль. Остальная сборка выполняется от вашего пользователя. Не запускайте весь установщик через `sudo`: иначе настройки окажутся у root.

На первом запуске создаётся **пустой список камер**. Камеры из рабочего YAML репозитория не копируются. Добавьте нужные источники через веб-интерфейс по инструкции. Повторная настройка не перезаписывает этот YAML.

Те же действия без вопросов:

```
smartcamera local-setup --yes --profile cpu \
  --data-dir "$HOME/.local/share/smartcamera/data" \
  --host 127.0.0.1 --port 8000 --domain-id 0
```

Перед фактической установкой можно посмотреть команды:

```
smartcamera local-setup --yes --dry-run
smartcamera docker-setup --yes --dry-run
```

`--dry-run` не устанавливает пакеты, не создаёт конфигурацию и не запускает приложение. Вывод показывает команды и, для Docker, конфигурацию Compose.

1.2.3 Что устанавливается локально

Локальная подготовка устанавливает системные зависимости, ROS Jazzy, rosdep, FFmpeg и библиотеки камер. Затем создаёт внутри исходников:

- `.venv/` — Python приложения с доступом к системному ROS/OpenCV;
- `.venv-inference/` — отдельный Python для PyTorch и Ultralytics;
- `.local/node/` — Node.js 22.16.0 для сборки интерфейса;
- `.local/librealsense/` — SDK 2.50.0, используемый проектом для L515;
- `build/`, `install/`, `log/` — результат сборки ROS.

Зависимости Node устанавливаются через `npm ci`. ML-окружение использует те же версии и скрипты, что описаны в разделе моделей. Установка `cuda` проверяет доступность GPU после установки PyTorch. Драйвер NVIDIA автоматически не заменяется. Если проверка не проходит, исправьте драйвер и повторите `local-setup`, либо выберите `cpu`.

Для V4L2 пользователь должен иметь доступ к `/dev/videoN`; обычно это группа `video`. Проверяйте через `id` и `ls -l /dev/video*`. После добавления в группу нужно выйти из сеанса и войти снова. Подготовка устанавливает правила udev RealSense; переподключите камеру после их обновления.

1.2.4 Запуск, остановка и журнал

```
smartcamera run
```

Откройте `http://127.0.0.1:8000`. Локальный запуск показывает вывод ROS в терминале. `Ctrl+C` завершает его процессы.

Для работы в фоне:

```
smartcamera run --detach
smartcamera status
smartcamera logs --follow
smartcamera stop
```

При локальном фоновом запуске журнал находится в `logs/application.log` внутри каталога данных. Выход из просмотра журнала через `Ctrl+C` не останавливает приложение; для этого предназначена команда `stop`. Автозапуск после перезагрузки компьютера в CLI не настраивается.

CLI управляет только зарегистрированным им запуском. Приложение, запущенное отдельной командой `ros2 launch`, нужно останавливать в его терминале. Для локальных процессов проверяется не только PID, но и время создания процесса, чтобы старый файл состояния не указывал на постороннюю программу.

1.2.5 Где хранятся настройки и данные

Путь	Содержимое
`~/.config/smartcamera/settings.json`	Режим, исходники, каталог данных, адрес, порт, ROS_DOMAIN_ID, устройства
Рядом: `settings.run.json`	Состояние запуска CLI
Рядом: `settings.run-<id>.compose.json`	Сохранённая конфигурация Docker-запуска
`<data>/config/cameras.yaml`	Камеры и их параметры
`<data>/config/web.json`	Настройки веб-интерфейса
`<data>/recordings/`	Материалы и связанные данные приложения
`<data>/logs/`	Журнал локального фонового запуска

Посмотреть и изменить настройки:

```
smartcamera configure --show
smartcamera stop
smartcamera configure --yes --port 8080 --domain-id 10
smartcamera run
```

Изменение каталога данных не переносит старые файлы. Остановите приложение, сохраните копию данных и учитывайте абсолютные пути в калибровках, настройках веб-сервера и реестре моделей. Подробности — конфигурация. Для отдельного набора настроек используйте глобальный аргумент перед командой:

```
smartcamera --config "$HOME/.config/smartcamera/test.json" configure --yes \
  --data-dir "$HOME/smartcamera-test" --port 8002
```

1.2.6 Обновление и диагностика

```
smartcamera stop

smartcamera update

smartcamera rebuild

smartcamera run
```

`update` обновляет текущую ветку из `origin` только через fast-forward. Он не переключается на master. Незакоммиченные и неотслеживаемые файлы, расхождение истории или неожиданный адрес

origin останавливают обновление с объяснением. Команда не делает reset и не удаляет локальные изменения. Сохранённые данные находятся вне исходников и не входят в обновление.

`rebuild` пересобирает приложение; для обновления системных зависимостей или смены CPU/CUDA повторите соответствующий `local-setup` / `docker-setup`.

```
smartcamera doctor
smartcamera docs
```

`doctor` проверяет необходимые файлы и инструменты; он не подтверждает качество изображения, доступность всех моделей или работу физического устройства. `docs` подготавливает отдельное окружение в `docs/.venv`, собирает руководство и показывает его на `http://127.0.0.1:8001`. Для сборки без сервера:

```
smartcamera docs --build-only --pdf
```

1.3 Установка из исходников

Этот способ подходит, если вы хотите сами выбрать каталог проекта или ветку. Для установки одной командой используйте быструю установку CLI. Оба способа устанавливают одну и ту же программу и используют мастер `smartcamera`.

1.3.1 Подготовьте компьютер

Для локального запуска поддерживаются Ubuntu 24.04 и Linux Mint на базе Noble. ROS 2 Jazzy, инструменты сборки, библиотеки камер и окружение моделей установит мастер. До его запуска нужны Git и Python с модулем `venv`:

```
sudo apt update

sudo apt install -y git curl python3 python3-venv
```

Для контейнерного режима нужен Linux с работающими Docker Engine и Compose. Подготовка Docker и устройств описана отдельно.

1.3.2 Скачайте проект

```
mkdir -p ~/dev

git clone --branch master \
  https://gitverse.ru/daniel-robotics/SmartCamera.git \
  ~/dev/SmartCamera

cd ~/dev/SmartCamera
```

Для другой опубликованной ветки замените `master` в команде клонирования. Если проект уже скачан, перейдите в его каталог и продолжите со следующего шага.

1.3.3 Установите команду

```
bash install.sh --source "$PWD"

export PATH="$HOME/.local/bin:$PATH"

smartcamera --help
```

CLI будет использовать именно этот каталог. Локальные изменения исходников попадут в сборку, выбранная ветка сохранится. Чтобы команда находилась в новых терминалах, добавьте строку `export PATH="$HOME/.local/bin:$PATH"` в `~/.bashrc`, если этого пути ещё нет в PATH.

1.3.4 Подготовьте приложение

Для выбора параметров в терминале запустите:

```
smartcamera setup
```

Выберите `local` или `docker`, профиль `cpu` или `cuda`, каталог данных, адрес веб-сервера и ROS-домен. Для первого запуска достаточно CPU. При установке системных пакетов потребуется пароль `sudo`.

Вместо мастера можно указать режим явно. Для локального запуска:

```
smartcamera local-setup --yes --profile cpu
```

Для Docker:

```
smartcamera docker-setup --yes --profile cpu
```

Это альтернативные варианты: выполняйте команду для выбранного режима. Для CUDA заранее подготовьте драйвер NVIDIA, а для Docker — также NVIDIA Container Toolkit. Затем замените `cpu` на `cuda`.

Локальная подготовка создаёт в проекте `.venv/`, `.venv-inference/`, `.local/node/` и `.local/librealsense/`, устанавливает зависимости интерфейса и собирает ROS-пакеты. SDK RealSense 2.50.0 устанавливается отдельно от системных библиотек. Подготовка Docker собирает зависимости внутри образа.

1.3.5 Запустите и подключите камеру

```
smartcamera run
```

Откройте http://127.0.0.1:8000. Команда сама подключает нужные окружения. Список камер изначально пустой; добавьте свой источник по инструкции первого запуска.

Настройки и записи хранятся отдельно от исходников. Посмотреть выбранные пути:

```
smartcamera configure --show
```

1.3.6 Пересоберите после изменения кода

```
smartcamera stop

smartcamera rebuild

smartcamera run
```

Для изменения системных зависимостей или профиля моделей повторите `local-setup` либо `docker-setup`. После переноса каталога проекта обновите пути по инструкции CLI. Готовые окружения Python могут содержать абсолютные пути и требуют повторной подготовки на новом месте.

1.4 Запуск в Docker

Docker-режим запускает драйверы, ROS-мост, веб-сервер и модели в одном контейнере. На Linux-хосте нужны Docker Engine и Compose. Docker Engine должен работать на том же компьютере, где находятся исходники, камеры и каталог данных. ROS, Node.js и Python приложения устанавливаются в образ. Управление выполняется той же командой `smartcamera`.

1.4.1 Подготовка

Сначала установите CLI, затем проверьте Docker:

```
docker version
docker compose version
docker info
```

Если Docker отсутствует, `smartcamera docker-setup` запускает установочный скрипт через `sudo` для Ubuntu 24.04. Скрипт использует официальный APT-репозиторий Docker. Конфликтующие пакеты не удаляются автоматически. Уже установленный Docker не переустанавливается. Если после установки пользователь не имеет доступа к сокету, настройте доступ по инструкции Docker, войдите в сеанс заново и повторите команду. CLI не добавляет пользователя в группы самостоятельно.

Для других дистрибутивов установите Docker по официальной инструкции. Далее CLI может использовать работающий Docker Engine. Docker Desktop на macOS и Windows не проверен для проброса камер и ROS DDS.

1.4.2 CPU: сборка и первый запуск

```
smartcamera docker-setup --yes --profile cpu

smartcamera run --detach

smartcamera status
```

Сборка выполняется из каталога исходников, выбранного при установке CLI или переданного через `--source`. Используются также локальные незакоммиченные файлы. При сборке контейнера используются эти исходники; повторного скачивания проекта нет.

Первая сборка скачивает базовые образы, системные пакеты и PyTorch. Это требует интернета, нескольких гигабайт свободного места и заметного времени. Готовый образ получает локальное имя `smartcamera:cpu`; он никуда не публикуется.

Откройте `http://127.0.0.1:8000`. Первый запуск без добавленных камер нужен для проверки веб-сервера. `run --detach` ожидает успешного HTTP health-check; если контейнер не готов за 120 секунд, команда завершится ошибкой. Посмотрите журнал:

```
smartcamera logs --follow
smartcamera stop
```

После неудачного запуска `stop` удаляет контейнер и служебное состояние CLI. Пользовательский каталог данных сохраняется. Команда `run` без `--detach` после проверки готовности показывает журнал; `Ctrl+C` останавливает контейнер.

1.4.3 USB-камера V4L2

По умолчанию контейнеру не передаётся ни одна камера. Разрешите нужное устройство:

```
smartcamera stop
smartcamera configure --yes --device /dev/video0
smartcamera run --detach
```

`--device` можно повторить для нескольких устройств. Переданный список заменяет предыдущий. `--clear-devices` очищает список. Контейнер получает UID/GID пользователя хоста и группы выбранных устройств. На хосте у пользователя тоже должен быть доступ к камере.

В веб-интерфейсе добавьте V4L2-камеру с тем же путём `/dev/video0`. Если используете `/dev/v4l/by-id/...`, передайте именно этот путь через `--device` и укажите его в конфигурации камеры. После переподключения V4L2-камеры остановите и снова запустите контейнер: номер устройства и привязка Docker могли измениться.

1.4.4 RealSense

Для доступа к USB-шине:

```
smartcamera stop
smartcamera configure --yes --usb
smartcamera run --detach
```

В этом режиме монтируется `/dev/bus/usb` и разрешаются USB character devices (major 189). Режим `privileged` не используется. `--usb` даёт доступ к USB-шине, а не только к одному серийному номеру. Конкретную камеру затем выбирают в SmartCamera. Для отключения доступа используйте `configure --yes --no-usb`.

Правила udev устанавливаются **на хосте**, не внутри контейнера. На Docker-хосте без локальной установки SmartCamera настройте правила RealSense отдельно:

```
curl --fail --location \
  https://raw.githubusercontent.com/IntelRealSense/librealsense/v2.50.0/config/99-realsense-
libusb.rules \
  --output /tmp/99-realsense-libusb.rules
sudo install -m 644 /tmp/99-realsense-libusb.rules /etc/udev/rules.d/99-realsense-libusb.rules
sudo udevadm control --reload-rules
```

Затем переподключите камеру. Образ собирается с librealsense 2.50.0, как профиль L515 в проекте. Для камер, использующих V4L2-узлы, также передайте соответствующие `/dev/videoN` через `--device`. Возможность горячего переподключения зависит от SDK, ядра и устройства; после потери доступа пересоздайте контейнер.

1.4.5 NVIDIA GPU

Установите на хосте совместимый NVIDIA-драйвер и NVIDIA Container Toolkit. Профиль проекта использует PyTorch с CUDA 13.0; драйвер должен поддерживать эту версию. Наличие установленного CUDA Toolkit на хосте само по себе не означает, что контейнеру доступен GPU.

```
nvidia-smi
smartcamera stop
smartcamera docker-setup --yes --profile cuda
smartcamera run --detach
```

Образ получает имя `smartcamera:cuda`. Compose запрашивает NVIDIA GPU, а entrypoint проверяет `torch.cuda.is_available()` перед запуском ROS. Если GPU недоступен, контейнер останавливается с объяснением в журнале. Сборка CUDA-образа не требует доступа к GPU; проверка выполняется именно при запуске.

Профиль `cuda` подготавливает окружение, но не меняет устройство у уже сохранённых запусков моделей. В карточке модели на камере выберите **Устройство → cuda:0** и примените настройки, как описано в запуске модели.

Механизм передачи GPU описан в документации Compose. CPU-образ не запрашивает NVIDIA runtime.

1.4.6 ROS 2 и сеть

Используется сеть хоста (`network_mode: host`), чтобы контейнер и ROS-узлы хоста могли участвовать в DDS discovery. Сопоставление портов `-p` не требуется. Порт 8000 должен быть свободен. Настройте одинаковый ROS_DOMAIN_ID:

```
smartcamera stop
smartcamera configure --yes --domain-id 10
smartcamera run --detach
export ROS_DOMAIN_ID=10
```

```
source /opt/ros/jazzy/setup.bash
ros2 topic list
```

Для своих программ на хосте установите и соберите `smart_camera_interfaces`. Примеры команд и QoS приведены в разделе ROS 2. Для доступа к веб-интерфейсу с другого компьютера задайте `configure --yes --host 0.0.0.0`; приложение будет слушать сетевые интерфейсы хоста.

1.4.7 Постоянные файлы

Каталог данных монтируется по **тому же абсолютному пути** внутри контейнера, чтобы пути к снимкам и моделям не менялись только из-за перехода в Docker. Исходники не монтируются поверх собранного приложения. После изменения кода выполните `stop`, `rebuild`, затем `run`.

Настройки веб-интерфейса могут содержать другой каталог записи. В контейнере доступны только смонтированные пути: выбирайте папки внутри каталога данных. Перенос существующей установки с моделями и калибровками требует проверки абсолютных путей; CLI не переписывает их автоматически.

Для просмотра Compose-конфигурации без создания файлов:

```
smartcamera run --mode docker --dry-run
```

Сгенерированная конфигурация рабочего запуска хранится рядом с настройками CLI. Команды `stop` и `logs` используют её, поэтому завершают именно зарегистрированный экземпляр. Перед изменением настроек или сборкой выполните `stop`.

1.5 Первый запуск

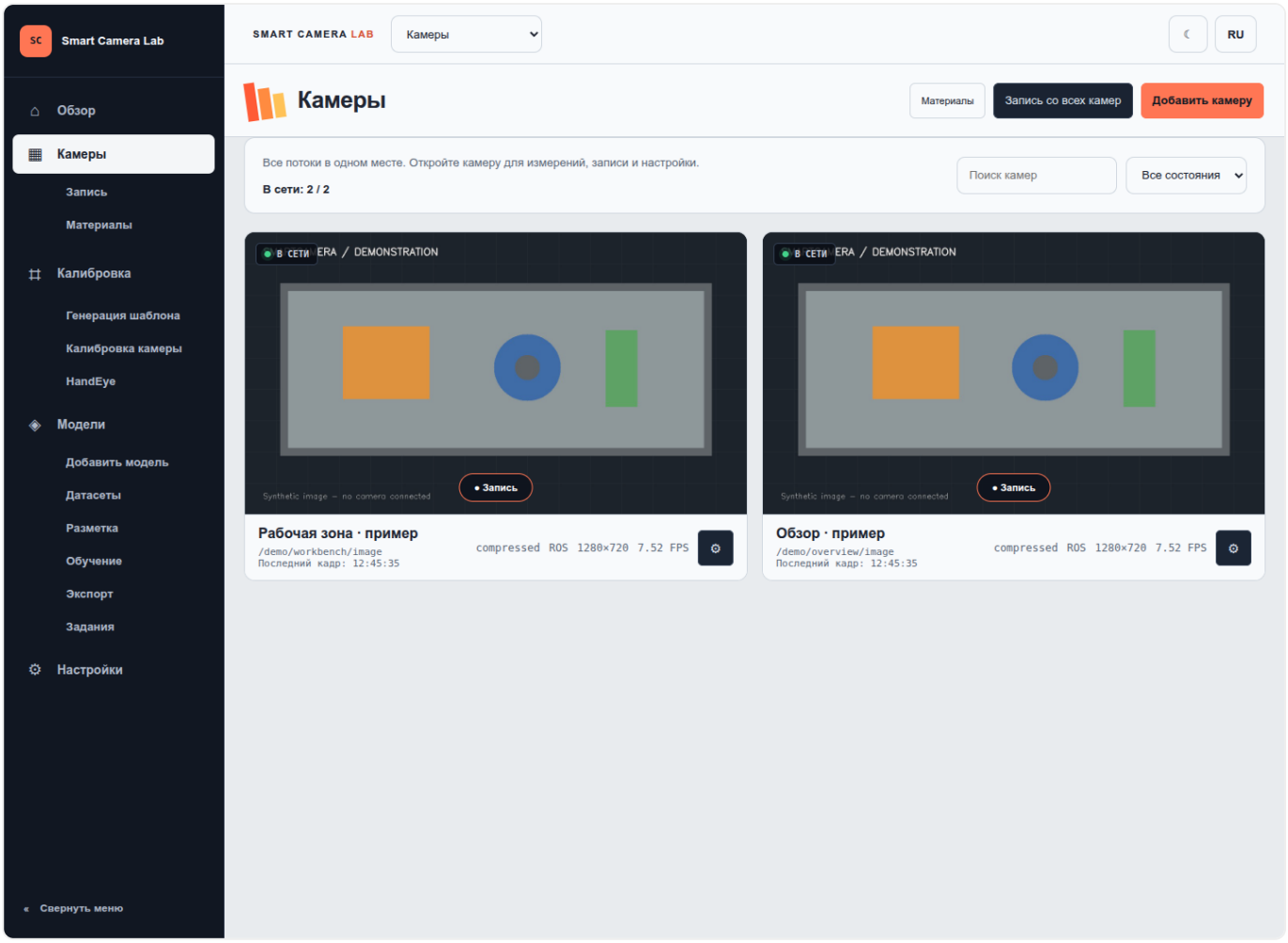
Сначала выполните установку через CLI или установку из исходников. Дальше можно проверить интерфейс без оборудования, подключить одну камеру и сохранить первый снимок. Модель нейросети, GPU и калибровочный шаблон для этого не нужны.

1.5.1 Запустите приложение

```
smartcamera run
```

CLI сам подключает окружения и использует сохранённые настройки. Создавать YAML вручную не нужно: первичная настройка уже подготовила пустой список камер. Повторная настройка не удаляет добавленные источники.

Откройте http://127.0.0.1:8000/cameras. Если вы изменили порт при установке, подставьте своё значение. Оставьте терминал открытым. Для фонового запуска вместо обычного используйте `smartcamera run --detach`, а журнал смотрите через `smartcamera logs --follow`.



Скриншот показывает страницу с уже добавленными камерами; названия и доступные потоки зависят от вашей конфигурации. Обзор остальных разделов находится в руководстве по интерфейсу.

Во втором терминале проверьте, что веб-сервер видит пустую конфигурацию:

```
curl --fail --silent http://127.0.0.1:8000/api/v1/cameras \
  | python3 -m json.tool
```

В JSON должны быть `revision` и пустой список `cameras`. Swagger с актуальной схемой HTTP-запросов доступен по адресу http://127.0.0.1:8000/docs.

1.5.2 Добавьте одну USB-камеру

Подключите камеру к компьютеру с сервером SmartCamera. При локальном запуске её устройства можно проверить командами:

```
v4l2-ctl --list-devices
ls -l /dev/v4l/by-id/
```

При запуске в Docker сначала разрешите доступ к устройству:

```
smartcamera stop

smartcamera configure --yes --device /dev/video0

smartcamera run --detach
```

Замените `/dev/video0` своим устройством. Для RealSense используйте инструкцию Docker, поскольку доступ к USB настраивается иначе.

На странице **Камеры** откройте добавление камеры и выберите V4L2-устройство. Предпочитайте стабильный путь `/dev/v4l/by-id/...`: номер `/dev/videoN` может измениться после переподключения. У одного устройства бывает несколько video-узлов; выбирайте узел, который действительно предоставляет нужный видеопоток.

Для первой камеры задайте ID `front`, понятное название и топик `/smart_camera/front/image_raw`. Выберите разрешение и FPS из списка поддерживаемых режимов. Не задавайте произвольный режим только потому, что он помещается в форму: драйвер должен подтвердить получение первого кадра. Для начала достаточно `640×480` и 30 FPS, **если этот режим есть у устройства**.

Для нескольких USB-камер MJPG часто уменьшает нагрузку на шину. Выбирайте его только при наличии в режимах устройства. Параметр `transport: compressed` управляет передачей изображения в ROS и не заменяет выбор формата захвата `pixel_format`.

Сохраните камеру и дождитесь изображения. При неудачном применении приложение пытается вернуть предыдущую конфигурацию; прочитайте сообщение формы и журнал launch, прежде чем повторять сохранение. Подробности RealSense, внешних ROS-потоков, depth и калибровки — в руководстве по камерам.

1.5.3 Проверьте поток и сохранение

Для камеры `front` с compressed transport:

Эти команды выполняются в терминале с окружением ROS 2.

```
ros2 topic info /smart_camera/front/image_raw/compressed -v
ros2 topic hz /smart_camera/front/image_raw/compressed
ros2 topic echo /diagnostics --once
```

Для raw transport используйте `/smart_camera/front/image_raw`. Камера может работать, даже если конкретный ROS-подписчик не получает изображения: сравните QoS. Изображения передаются с best-effort, volatile и глубиной очереди 1; для подписки используйте совместимую политику.

Сделайте одиночный снимок через интерфейс и найдите материал в разделе **Архив**. Это проверяет не только просмотр, но и запись в выбранный каталог. Если запущена запись видео, серия фото или ROS bag, сначала остановите её: операции записи взаимоисключающие. Для первого теста

достаточно одной камеры — групповая фотография дополнительно требует свежих кадров с допустимым расхождением timestamps.

Остановите приложение через `smartcamera stop` (или `Ctrl+C` в терминале запуска), затем снова выполните `smartcamera run`. Камера должна остаться в конфигурации, а снимок — в архиве. Закрытие вкладки браузера сервер не останавливает. При штатном завершении лучше сначала закончить запись, чтобы материал успел получить окончательный статус.

1.5.4 Подключите нейросеть и внешнее управление

Когда обычный поток работает, откройте библиотеку моделей, скачайте стандартную модель в библиотеку и добавьте запуск в **Камеры → Нейросети**. Выбор модели относится к конкретному запуску: одной камере можно назначить несколько моделей. Успешное включение ещё не означает, что веса уже загружены — дождитесь состояния и обработанного кадра. После перезапуска сервера запуски остаются выключенными.

Для команд ROS ниже сначала подключите окружение ROS 2. Для ROS-управления браузер не нужен, но `web_server` должен продолжать работать. Он предоставляет общую конфигурацию для веб-панели и ROS-команд:

```
ros2 service call /smart_camera/control smart_camera_interfaces/srv/Control \
  '{command: {operation: cameras.list}}'
```

Команды выбора модели, включения камеры и подписки на результаты приведены в ROS 2 API. Для HTTP-клиента используйте справочник HTTP API.

1.5.5 Доступ с другого компьютера

Остановите приложение и измените адрес веб-сервера:

```
smartcamera stop

smartcamera configure --yes --host 0.0.0.0 --port 8000

smartcamera run --detach
```

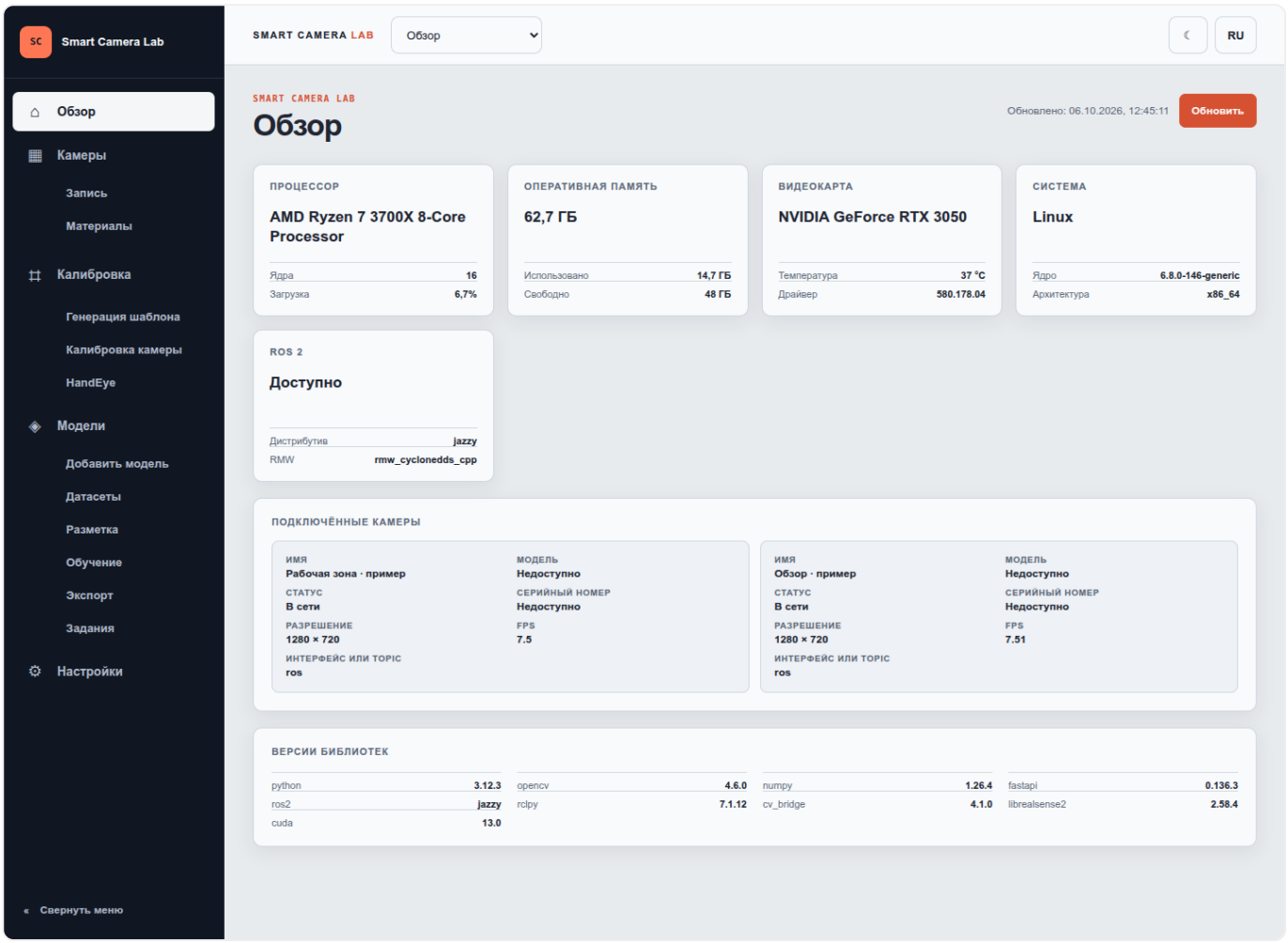
Откройте `http://IP_СЕРВЕРА:8000/cameras` на другом компьютере. В приложении нет встроенной аутентификации и TLS; такой запуск предназначен для доверенной сети. Если локальная страница открывается, а удалённая нет, проверьте адрес сервера и сетевой фильтр. Ошибки установки и потоков разобраны в диагностике.

1.6 Интерфейс Smart Camera Lab

После запуска откройте `http://127.0.0.1:8000/`. Это адрес веб-сервера на том же компьютере. При работе с другого компьютера используйте адрес машины с камерами и разрешённый для локальной сети запуск сервера.

В руководстве приведены снимки **реального интерфейса SmartCamera с синтетическими демонстрационными данными камер**. Они показывают расположение элементов и порядок работы. Изображения потоков, названия устройств и результаты в демонстрации не являются измерениями оборудования.

1.6.1 Обзор системы



1. Нажмите **Обзор** в левом меню. В верхней части страницы расположены сведения о процессоре, памяти, GPU, операционной системе и ROS 2.
2. Найдите список камер ниже карточек системы. Сверьте название, состояние, разрешение и FPS нужной камеры. Интерфейс показывает измеренный FPS, если он доступен; иначе — заданный в конфигурации.
3. Для повторного запроса данных нажмите **Обновить** справа вверху. Страница также обновляется автоматически с интервалом из настроек; резервный интервал — пять секунд.
4. Внизу страницы проверьте версии библиотек. Эти сведения полезны при разборе ошибки: одинаковый YAML с другим SDK или OpenCV может работать иначе.

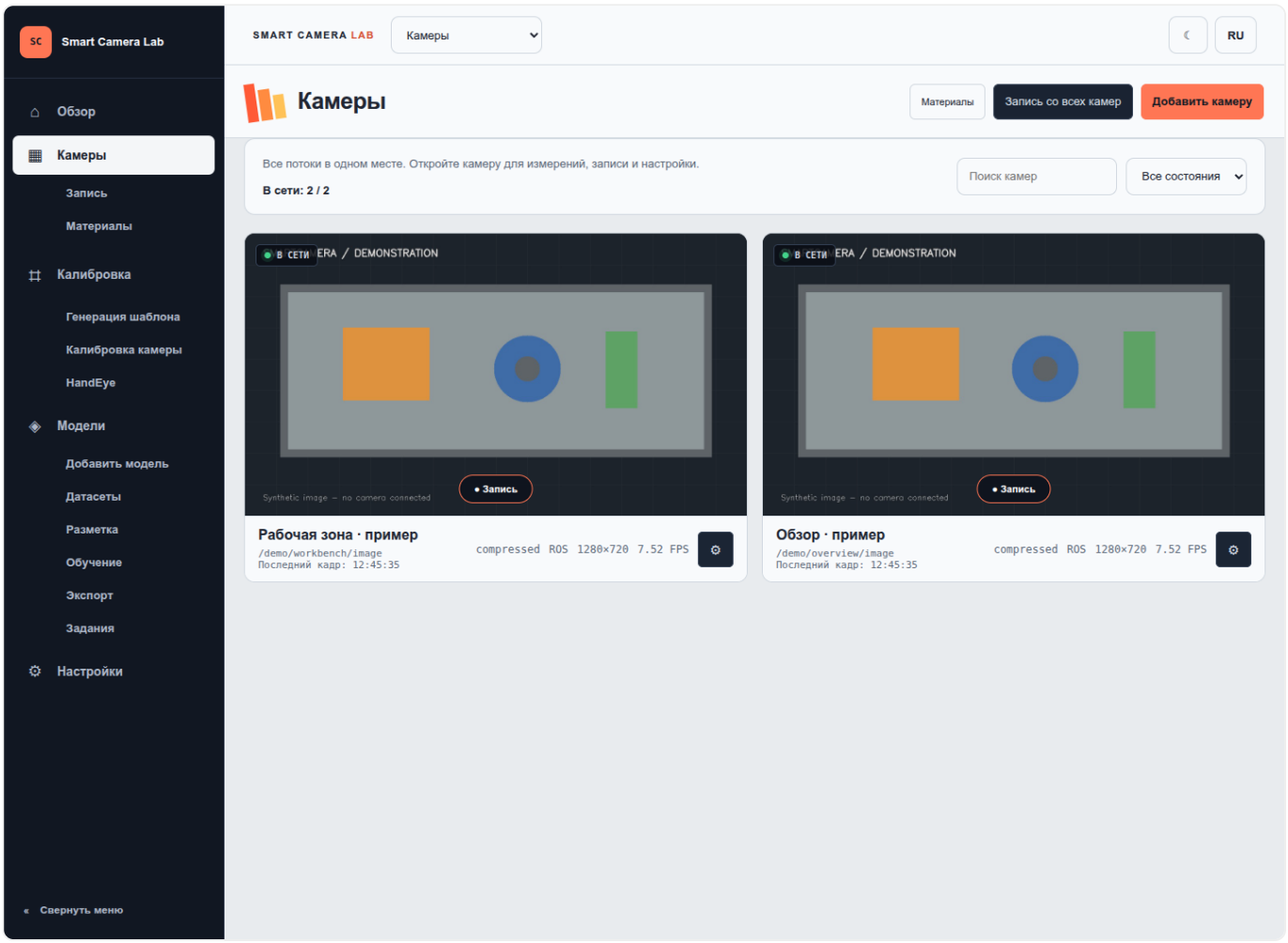
Отсутствие значения обозначается как недоступные данные. Например, пустые сведения о GPU сами по себе не означают неисправность камеры. Если после успешного обновления сервер перестал отвечать, обзор оставляет последние данные и сообщает об их устаревании. Проверяйте время последнего обновления перед тем, как делать вывод о текущем состоянии.

1.6.2 Куда переходить

Пункт меню	Что в нём делают
Камеры	Смотрят потоки, добавляют источники, открывают отдельную камеру и её настройки
Камеры → Запись	Открывают диалог фото, видео или ROS bag с выбранными включёнными камерами
Камеры → Материалы	Просматривают, скачивают и удаляют сохранённые материалы
Калибровка → Генерация шаблона	Задают физические размеры мишени и скачивают файл для печати
Калибровка → Калибровка камеры	Рассчитывают внутренние параметры объектива по изображениям мишени
Калибровка → HandEye	Определяют преобразование между оптическим фреймом камеры и роботом
Модели	Работают с моделями, датасетами, разметкой, обучением и заданиями
Настройки	Меняют общие параметры веб-приложения

Кнопка **Свернуть меню** в нижней части боковой панели оставляет только значки. Для вложенных пунктов наведите указатель на соответствующий значок или перейдите к нему клавиатурой. **Развернуть меню** возвращает подписи. На узком экране меню открывается кнопкой ☰; закрыть его можно крестиком или клавишей `Esc`.

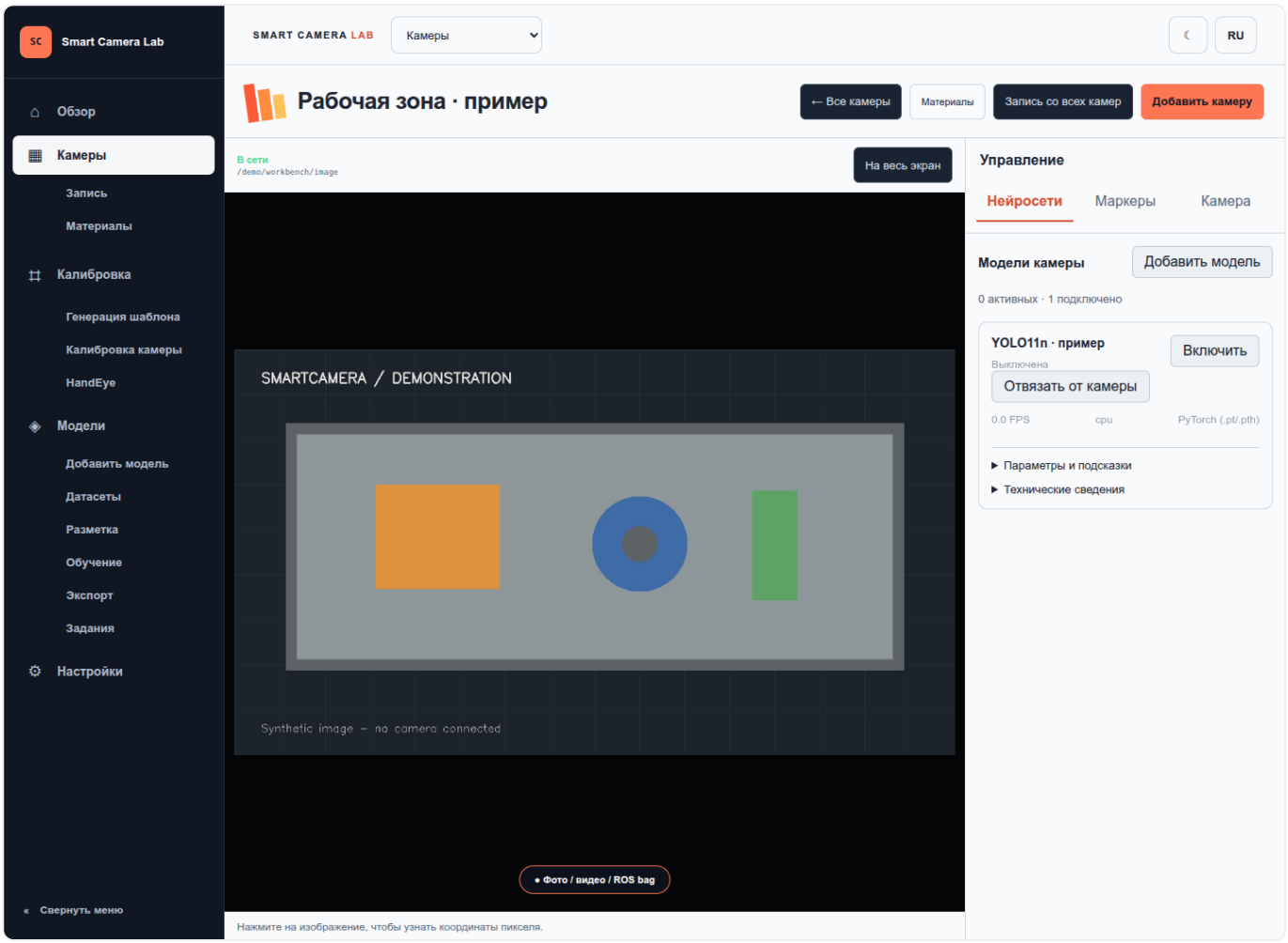
1.6.3 Сетка камер и отдельный поток



1. Откройте **Камеры**. Каждая карточка соответствует одному источнику из конфигурации.
2. Введите часть названия в **Поиск камер** или выберите состояние в списке **Все состояния**. При пустом результате используйте **Сбросить фильтры**.
3. Нажмите на изображение нужной камеры, чтобы открыть её рабочую область. Кнопка **Все камеры** возвращает сетку.
4. Для настройки источника нажмите шестерёнку на карточке. Для записи только этого источника используйте **Запись** на его карточке.
5. Для общей записи нажмите **Запись со всех камер** в верхней панели. Окончательный набор можно изменить в диалоге записи.

Состояние **Отключена** означает выключенный источник в конфигурации. **Нет сигнала** означает, что ожидаемый поток сейчас недоступен. **Перенастраивается** может кратковременно появиться после сохранения параметров. Если состояние не меняется, переходите к диагностике, а не добавляйте вторую запись для того же устройства.

1.6.4 Управление отдельной камерой

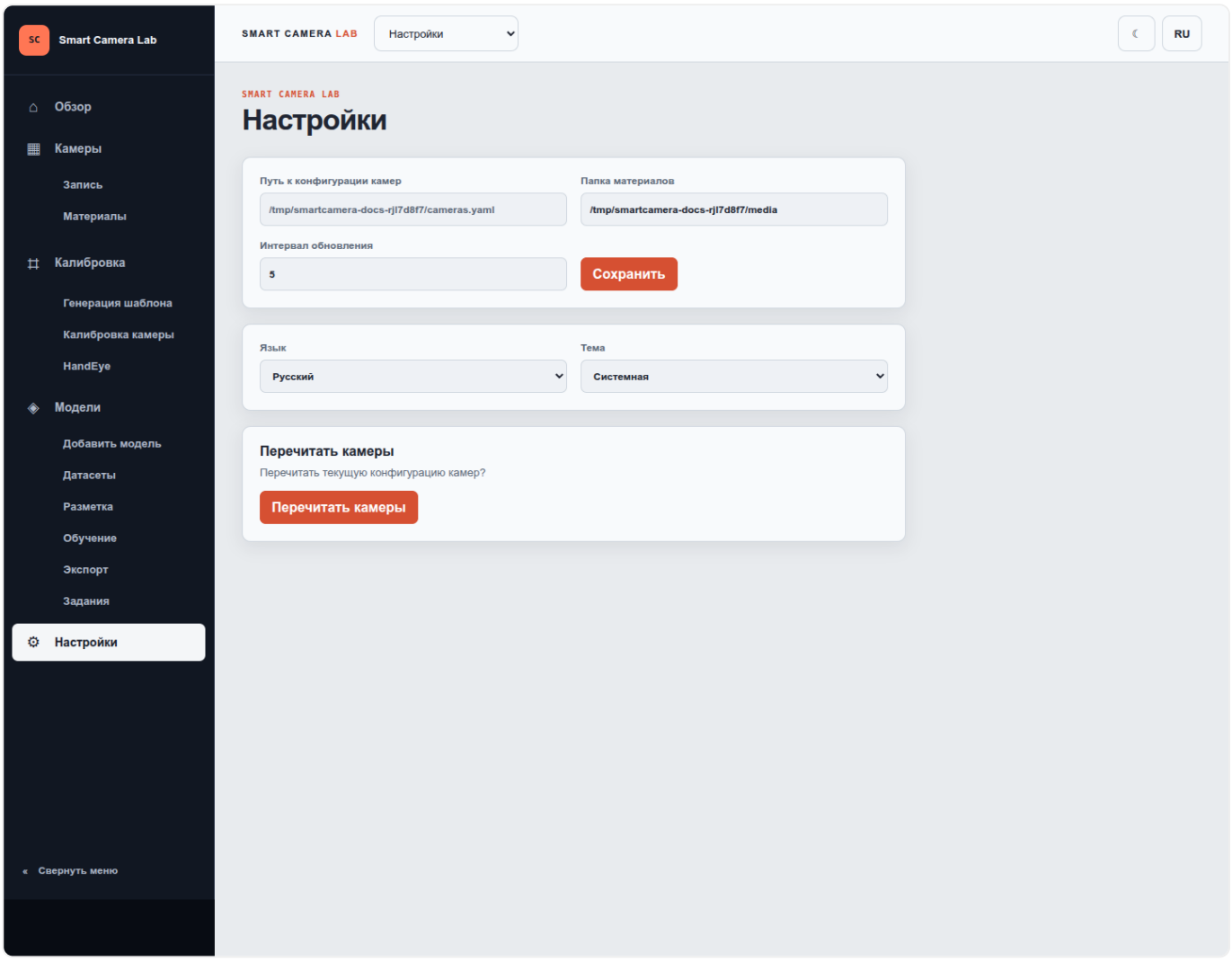


1. В рабочей области откройте **Управление камерой**, если панель скрыта размером окна.
2. На вкладке **Камера** проверьте источник, фактическое разрешение и частоту. Кнопка **Настройки** открывает редактирование. ID и ROS topic находятся в блоке **Технические сведения**.
3. На вкладке **Нейросети** управляйте запусками моделей. Подробный порядок приведён в руководстве по обработке потока.
4. На вкладке **Маркеры** настраивайте распознавание маркеров для выбранного источника.
5. Кнопка **На весь экран** расширяет просмотр средствами браузера. Это режим отображения, он не меняет разрешение публикации или записи.

Нажатие на исходное изображение показывает координаты пикселя. У RealSense дополнительно доступна глубина. Если выбран **Результат модели**, сначала переключитесь на **Исходный поток**: измерение относится к исходным данным. Подробности добавления источников и глубины находятся на странице Камеры.

Изменение настроек и запуск записи — разные действия. Параметры источника сохраняются кнопкой **Сохранить и применить**, а запись запускается в отдельном диалоге. После закрытия браузера сервер продолжает работать; для завершения активной записи используйте **Остановить**.

1.6.5 Общие настройки



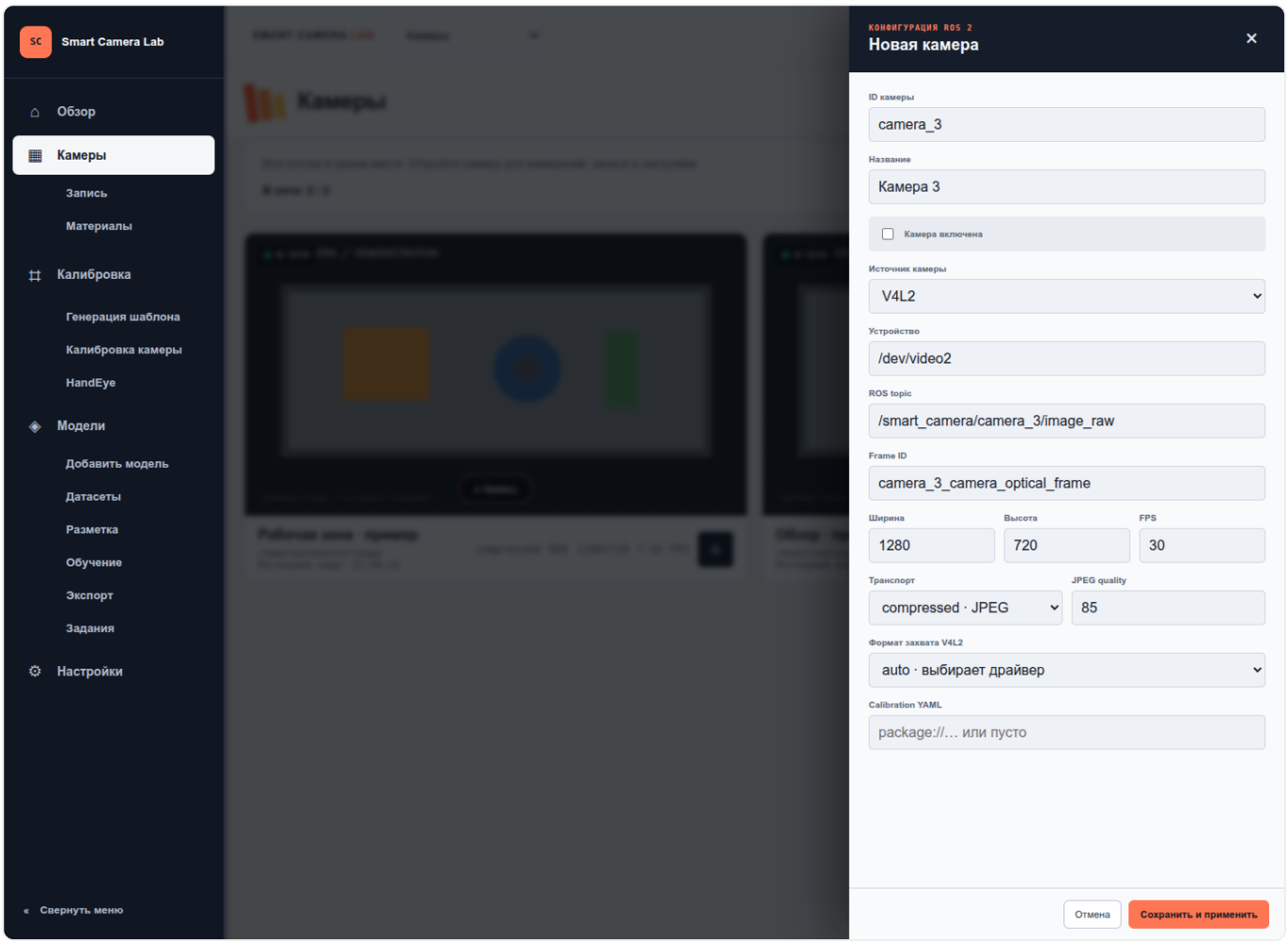
Откройте **Настройки** внизу левого меню. Здесь задаются оформление и каталог хранения материалов на сервере. Перед изменением папки завершите запись. Порядок переноса и поведение уже сохранённых файлов описаны в разделе о конфигурации.

2. Камеры и запись

2.1 Камеры и источники изображения

SmartCamera принимает изображения от V4L2-камер, Intel RealSense и внешних ROS-издателей. Для V4L2 и RealSense проект управляет устройством; для **ROS-топика** подписывается на уже работающий внешний драйвер. Сначала запустите приложение, затем откройте **Камеры**.

2.1.1 Добавление V4L2-камеры



1. Нажмите **Добавить камеру** справа вверху. Откроется форма **Новая камера**.
2. Задайте **ID камеры**, например `entrance`, и **Название**, например `Вход`. ID используется в конфигурации и адресах API; название удобно сделать понятным оператору.
3. В поле **Источник камеры** выберите **V4L2**, а в **Устройство** — нужную камеру. Если список устройств пуст, форма позволяет ввести путь вручную.
4. Укажите **ROS topic**, например `/smart_camera/entrance/image_raw`, и **Frame ID**, например `entrance_camera_optical_frame`.
5. Заполните **Ширина**, **Высота** и **FPS** согласно режиму, который поддерживает камера. Для первого подключения используйте проверенный режим, например `640 × 480`, 30 FPS, если он присутствует у устройства.
6. Выберите **Формат захвата V4L2** и **Транспорт**, проверьте флаг **Камера включена**, затем нажмите **Сохранить и применить**.
7. Дождитесь изображения в сетке. Откройте камеру и на вкладке **Камера** сравните фактические разрешение и FPS с заданными.

Предпочтителен стабильный путь `/dev/v4l/by-id/...`: номер `/dev/videoN` может измениться после переподключения. Некоторые устройства создают несколько узлов; нужен узел захвата

изображения, а не служебный интерфейс. Список форматов можно проверить без запуска SmartCamera:

```
v4l2-ctl --list-devices
v4l2-ctl --device=/dev/video0 --list-formats-ext
```

Формат захвата V4L2 задаёт передачу от устройства к компьютеру: `auto`, `MJPG` или `YUYV`. `MJPG` часто снижает нагрузку на USB при нескольких камерах, но его можно выбирать только при поддержке конкретным устройством. **Транспорт** задаёт публикацию в ROS: `raw · cv_bridge` или `compressed · JPEG`. Для compressed используется `<ROS topic>/compressed`, для raw — имя без добавленного суффикса. **JPEG quality** имеет диапазон 1–100 и доступно для compressed. Это отдельная настройка от USB-формата.

В **Параметры UVC** отображаются только доступные параметры устройства. Для ручной экспозиции выберите **Автоэкспозиция → Вручную** и задайте поддерживаемое значение. Шкала `exposure`, `gain` и `brightness` определяется устройством; её нельзя без проверки трактовать как миллисекунды. Пустые значения и **Не изменять** сохраняют настройку драйвера.

2.1.2 Внешний ROS-поток

1. Запустите внешний драйвер камеры в том же ROS-домене, что и SmartCamera.
2. Нажмите **Добавить камеру**, задайте ID и название, выберите **Источник камеры → ROS-топик**.
3. В **Доступные топики** выберите поток. При необходимости нажмите **Обновить топики** или введите полное имя в **ROS topic**.
4. Проверьте **Тип сообщения**: `sensor_msgs/msg/Image` или `sensor_msgs/msg/CompressedImage`.
5. Нажмите **Сохранить и применить** и дождитесь кадра.

Для такого источника имя задаётся **целиком**, включая `/compressed`, если оно есть у издателя. SmartCamera не дописывает его автоматически. Например, для `/external_camera/image_raw/compressed` выбирают тип `CompressedImage`. Разрешение и частота определяются входящими кадрами; менять экспозицию и режим нужно во внешнем драйвере. Поле **Частота сохраняемого видео, FPS** относится к MP4, а не к перенастройке издателя.

Проверить фактический тип и наличие публикации можно командами:

```
ros2 topic info /external_camera/image_raw -v
ros2 topic hz /external_camera/image_raw
```

Если поток не появился, проверьте окружение, `ROS_DOMAIN_ID`, тип сообщения и совместимость QoS. Подробные примеры приведены в ROS 2 управлении.

2.1.3 Intel RealSense

1. В форме добавления выберите **Intel RealSense**.

2. В списке **Камера RealSense** выберите подключённое устройство по модели и serial.
3. Выберите **Готовый RGB-D профиль** из обнаруженных профилей. Это согласованный набор режимов RGB и глубины.
4. При необходимости включите **Публиковать облако точек** и проверьте **Topic облака точек**.
5. Задайте **Минимальная глубина для карты, м**, **Максимальная глубина для карты, м** и **Палитра карты глубины**. Эти поля управляют визуализацией карты.
6. Нажмите **Сохранить и применить**.

> **Расширенные параметры устройства**
>
> В блоке **Расширенные настройки устройства** числовое значение отправляется устройству при выходе из поля. Для этих параметров не нужно ждать кнопки сохранения всей формы. Поля **только чтение** изменить нельзя; диапазон и шаг поступают от SDK.

Для L515 проект проверен с firmware `1.5.8.1`, SDK `2.50.0`, USB 3.2 и профилем RGB `640 × 480 @ 30` + depth `640 × 480 @ 30`. Это проверенная комбинация проекта, а не обещание совместимости любых SDK. Установку отдельного SDK и параметр `realsense_sdk_prefix` смотрите в инструкции установки. L500 использует LiDAR; параметры Advanced Mode и межкамерной синхронизации D400 к ней не применяются.

В отдельном просмотре RealSense можно переключать RGB и карту глубины. Нажмите на исходное изображение: появятся координаты X/Y и **Depth**. В **Единицы глубины** доступны мм, см, м и дюймы. RGB-измерение использует глубину, совмещённую по заводской калибровке, и последний кадр. **Нет данных** означает отсутствие измерения в точке, а не нулевую дистанцию. **Сбросить измерение** убирает отметку.

2.1.4 Изменение и удаление

Для изменения нажмите шестерёнку в сетке либо **Камера → Настройки** в рабочей области. После сохранения изменённый V4L2-worker перезапускается отдельно; если новая конфигурация не дала первого кадра, менеджер возвращает последнюю рабочую конфигурацию. Не считайте сохранение успешным, пока не проверены сообщение сервера и поток.

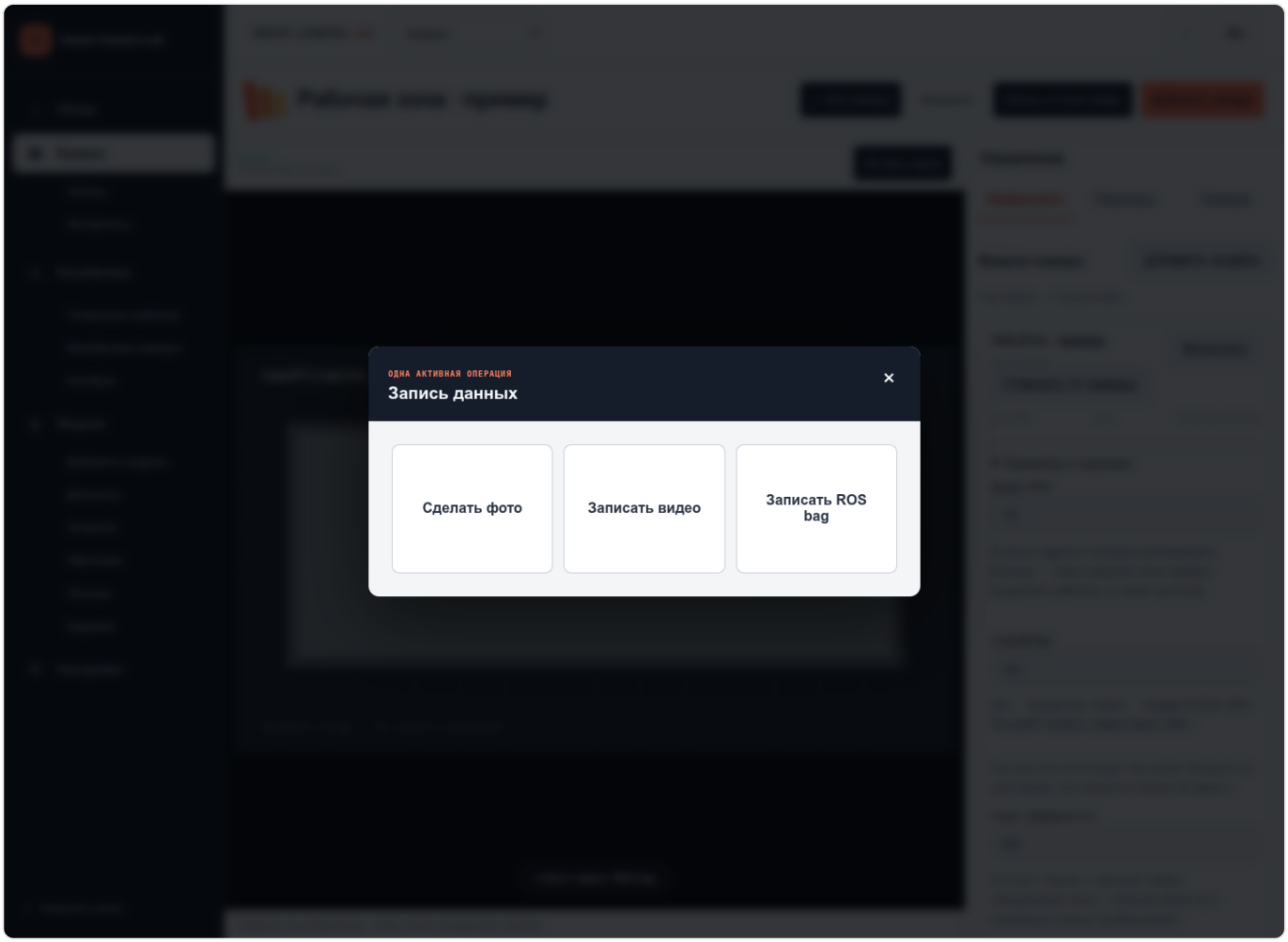
Если показано **YAML доступен только для чтения**, редактирование заблокировано. Проверьте права на фактический файл конфигурации. При параллельном редактировании из двух окон сервер может отклонить устаревшую версию: перечитайте настройки и повторите правку.

Удалить в форме убирает камеру после подтверждения. Для временной остановки достаточно снять **Камера включена** и сохранить. Поле **Calibration YAML** задаёт внутреннюю калибровку; порядок её получения и применения описан в следующей главе.

2.2 Фото, видео и ROS bag

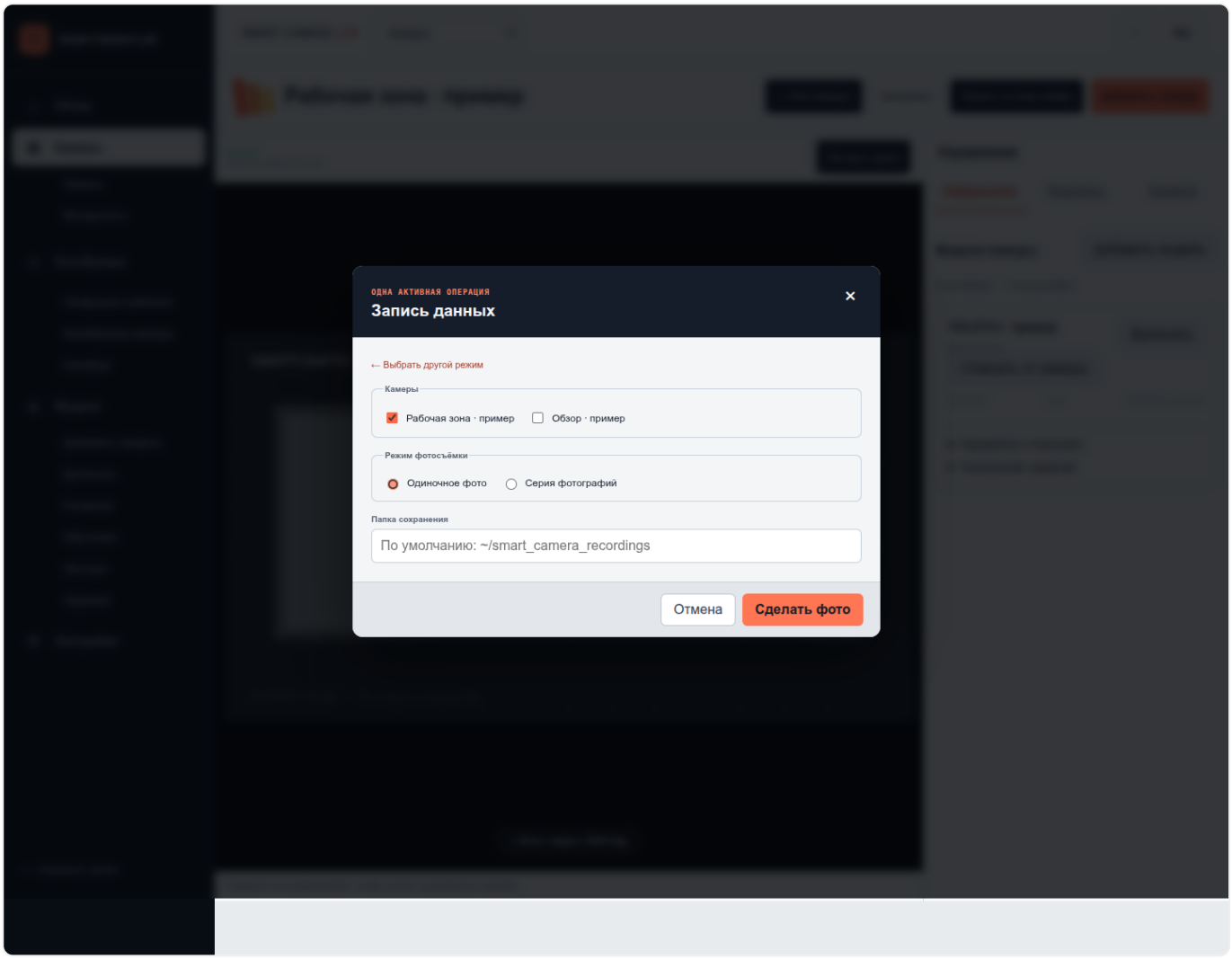
Запись выполняется на компьютере, где работает веб-сервер. Браузер управляет операцией и позволяет скачать результат; закрытие вкладки не останавливает серверную запись. Одновременно разрешена одна операция записи: фото, серия, MP4 или ROS bag.

2.2.1 Открытие диалога



1. В левом меню выберите **Камеры → Запись** либо нажмите **Запись со всех камер** над сеткой.
2. Для одной камеры используйте **Запись** на её карточке или **Фото / видео / ROS bag** в отдельном просмотре.
3. В окне **Запись данных** выберите **Сделать фото**, **Записать видео** или **Записать ROS bag**.
4. В режимах фото и MP4 отметьте нужные камеры. В списке доступны включённые источники; наличие флага ещё не гарантирует свежий кадр.
5. Проверьте **Папка сохранения** и параметры режима, затем нажмите кнопку запуска внизу окна.

Пустая папка означает серверный каталог по умолчанию, обычно `~/smart_camera_recordings`. Допускаются абсолютный путь и путь с `~/`; относительный путь сервер отклонит. Тильда относится к домашнему каталогу пользователя **веб-сервера**, а не пользователя браузера. Например, `/home/daniel/captures/calibration_front` должен быть доступен на запись серверу. Внутри него для операции создаётся отдельный каталог с UUID и `manifest.json`.



2.2.2 Одиночный снимок

Выберите **Сделать фото → Одиночное фото**, оставьте нужные камеры и снова нажмите **Сделать фото** внизу. Для каждой выбранной камеры сервер должен получить свежий кадр. При нескольких источниках проверяется допустимый разброс временных меток; группа не сохраняется частично, если синхронизированный набор не собран.

Это программная проверка времени, а не аппаратная синхронизация камер. Обычная USB-webcam получает ROS timestamp после успешного чтения кадра; он не обязательно совпадает с моментом экспозиции сенсора. Для неподвижной мишени такой режим удобен, но для измерения быстрого движения сначала оцените временные характеристики источников.

2.2.3 Серия фотографий

Серия подходит для съёмки калибровочной мишени или наблюдения за медленным процессом.

1. Выберите **Сделать фото → Серия фотографий**.
2. В **Количество фотографий** задайте число шагов. Например, `20` означает до двадцати наборов выбранных камер.
3. Укажите **Интервал между фото, сек.**, например `5`.
4. В **Задержка перед началом, сек.** задайте `10`, чтобы успеть занять первое положение. Пустое значение означает задержку, равную интервалу; `0` — немедленный первый шаг.
5. Нажмите **Начать серию**. Следите за счётчиком выполненных шагов, **Пропущено** и временем до следующего снимка.
6. При необходимости нажмите **Остановить** в полосе активной записи.

Поле	Допустимые значения
Количество фотографий	Целое от 2 до 100000
Интервал	От 1 до 86400 с
Начальная задержка	От 0 до 86400 с либо пустое значение

Шаги выполняются по фиксированному расписанию. Если сервер не получил согласованный набор камер, он пропускает шаг целиком; сохранённых наборов может оказаться меньше заданного числа. Уже полученные фотографии остаются в архиве после штатной остановки. Для калибровки перед каждым шагом удерживайте мишень неподвижно, затем меняйте наклон и положение.

2.2.4 MP4

Выберите **Записать видео**, отметьте камеры и при необходимости заполните **Длительность, сек. (необязательно)**. Например, `30` ограничивает запись тридцатью секундами. Пустое поле оставляет запись до команды **Остановить**. Нажмите **Начать запись**.

Каждая камера сохраняется в отдельный `<camera_id>.mp4` с настроенным FPS. При двух и более камерах дополнительно создаётся `mosaic.mp4`: сетка Full HD `1920 × 1080`, 30 FPS, с названиями камер и ROS timestamp. Пропорции изображения сохраняются. Мозаика удобна для общего просмотра; отдельные файлы оставляют каждый ракурс доступным независимо.

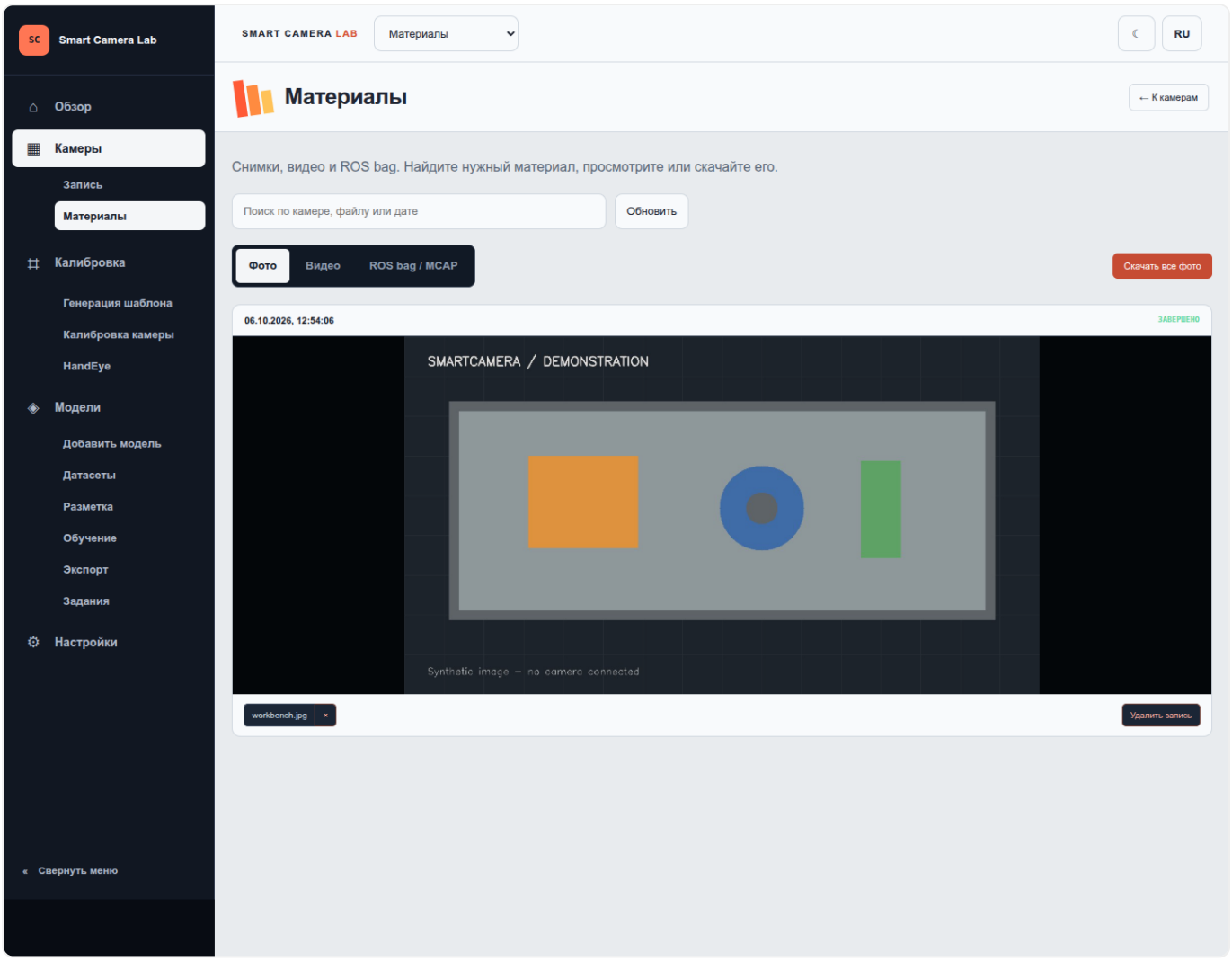
Видео кодируется FFmpeg в H.264. При перегрузке ограниченные очереди могут терять кадры; число отражается в `dropped_frames` manifest. Высокая заданная частота не доказывает, что источник реально выдаёт столько кадров. После **Остановить** дождитесь завершения обработки: временные `.part.mp4` не считаются готовыми материалами. Обычные фото и видео сохраняются без разметки детектора маркеров.

2.2.5 ROS bag

Выберите **Записать ROS bag**, затем **Формат ROS bag → MCAP** или **SQLite3**. В списке **ROS topics** проверьте отмеченные топики и добавьте необходимые. Начальный выбор связан с камерами, но перед запуском нужно проверить его: список показывает активные ROS-публикации, а не все возможные имена из конфигурации.

Для последующей обработки изображения обычно нужны также соответствующий `camera_info`, а для робототехнического опыта — положения суставов или другие данные вашей задачи. Наличие камеры в сетке не добавляет автоматически все эти вспомогательные сообщения. У RealSense при наличии публикаций в начальный набор могут войти глубина и включённое облако точек. Нажмите **Начать запись**, а по окончании — **Остановить**.

2.2.6 Поиск и скачивание материалов



1. Нажмите **Материалы** над сеткой или **Камеры → Материалы** в меню.
2. Выберите тип: **Фото**, **Видео** или **ROS bag / MCAP**.
3. Используйте **Поиск по камере, файлу или дате**, чтобы найти нужную операцию.
4. Просмотрите фотографии или запустите встроенное видео. Для нескольких ракурсов найдите **Объединённое видео**.
5. Скачайте отдельный файл либо **Скачать ZIP** для записи. В разделе фото доступно **Скачать все фото**.
6. Удаляйте ненужное через **Удалить запись** или кнопку удаления отдельного файла. Подтверждение удаления означает удаление серверных данных без возможности восстановления средствами приложения.

Архив восстанавливается после перезапуска. Если процесс завершился во время записи, операция может иметь состояние **Прервано**; это не то же самое, что штатное **Завершено**. Проверьте доступные файлы и manifest перед использованием.

Для bag кнопка **Воспроизвести в /replay** публикует сообщения в `/replay<исходный_топик>`. Доступны **Пауза**, **Продолжить**, позиция в секундах и скорости `0.25`, `0.5`, `1`, `2`. Для завершения используйте **Стоп**. Это реальная публикация ROS-сообщений. Подключайте просмотрщик к `/replay/...`, если хотите анализировать запись отдельно от работающей камеры.

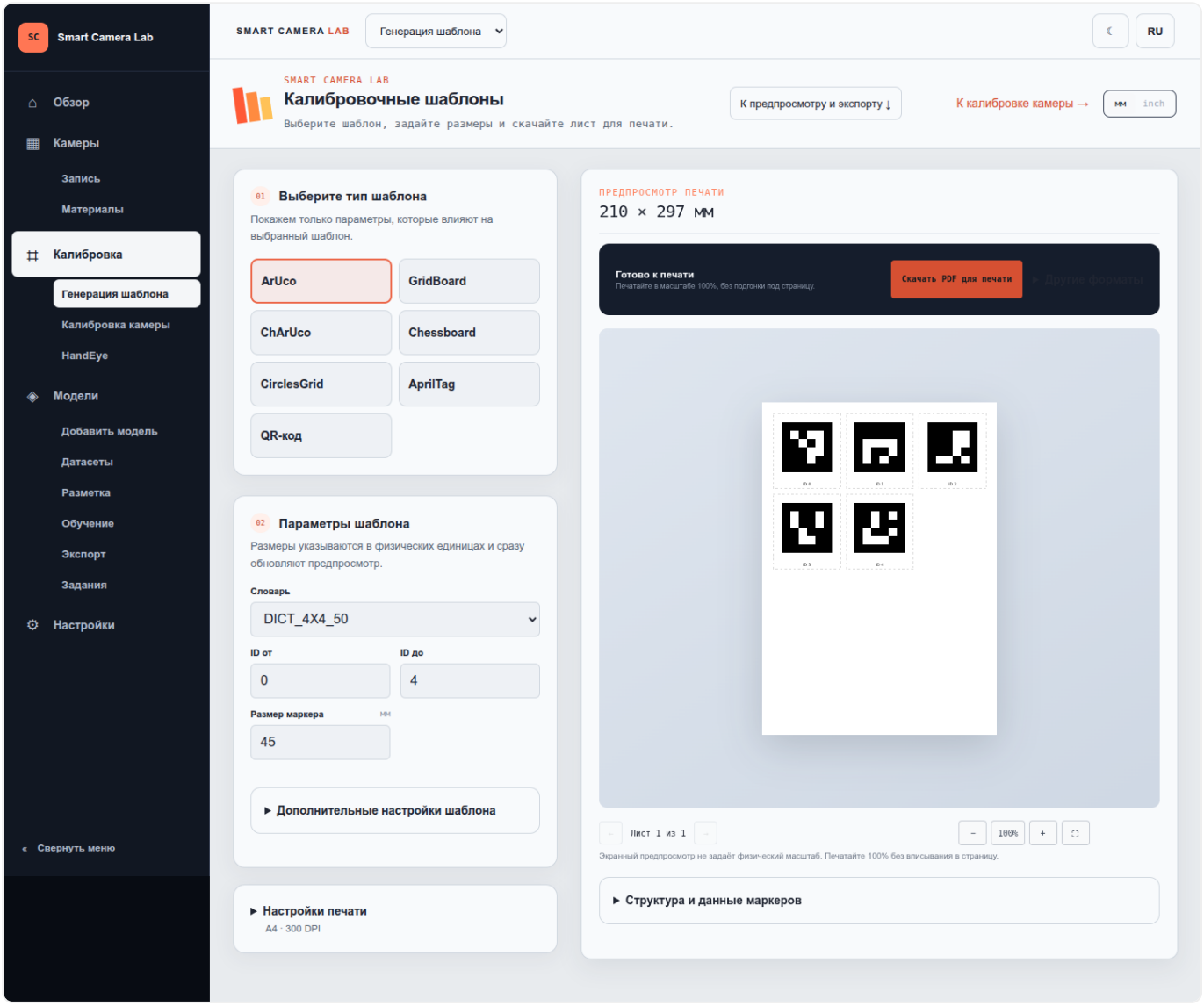
В просмотре bag выберите **ROS topic для просмотра** и нажмите **Просмотреть сообщения**. Изображения можно смотреть как поток; остальные сообщения отображаются JSON. Выбор **Числовое поле** строит временной график. При ошибке записи или воспроизведения сначала проверьте зависимости и диагностику.

3. Калибровка

3.1 Шаблон и внутренняя калибровка камеры

Внутренняя калибровка определяет фокусные параметры, главную точку и дисторсию объектива. Она нужна для геометрических измерений и HandEye. Используйте снимки одной камеры в одном разрешении и с неизменной оптикой. Изменение фокуса или масштаба после съёмки может сделать результат непригодным.

3.1.1 Подготовка печатной мишени



1. Откройте **Калибровка → Генерация шаблона**.
2. Выберите тип мишени. Для мастера внутренней калибровки подходят **Chessboard**, **ChArUco** и **CirclesGrid**. Генератор также создаёт маркеры ArUco, AprilTag и QR, но наличие формата в генераторе не означает поддержку его монокалибровкой.
3. Задайте структуру и физические размеры. Например, для Chessboard укажите 7 внутренних углов по X, 5 по Y и клетку 25 мм.
4. В **Настройки печати** выберите формат и ориентацию листа, проверьте поля и предпросмотр. Если мишень не помещается, увеличьте лист или уменьшите размеры осознанно.
5. Нажмите **Скачать PDF для печати**. Через **Другие форматы → Экспорт JSON** сохраните параметры рядом с PDF; JSON затем можно импортировать в мастер.
6. Распечатайте с масштабом **100%**, отключив подгонку под страницу. Измерьте клетку или шаг линейкой и закрепите лист на плоской жёсткой основе.

У Chessboard считаются **внутренние углы**, а не клетки: 7 × 5 внутренних углов соответствуют 8 × 6 клеткам. У ChArUco задаётся число **клеток** и отдельно размер маркера; маркер должен помещаться внутри клетки. Для CirclesGrid важны тип сетки, число центров, шаг и диаметр круга. В калибровочном мастере физические размеры вводятся в миллиметрах.

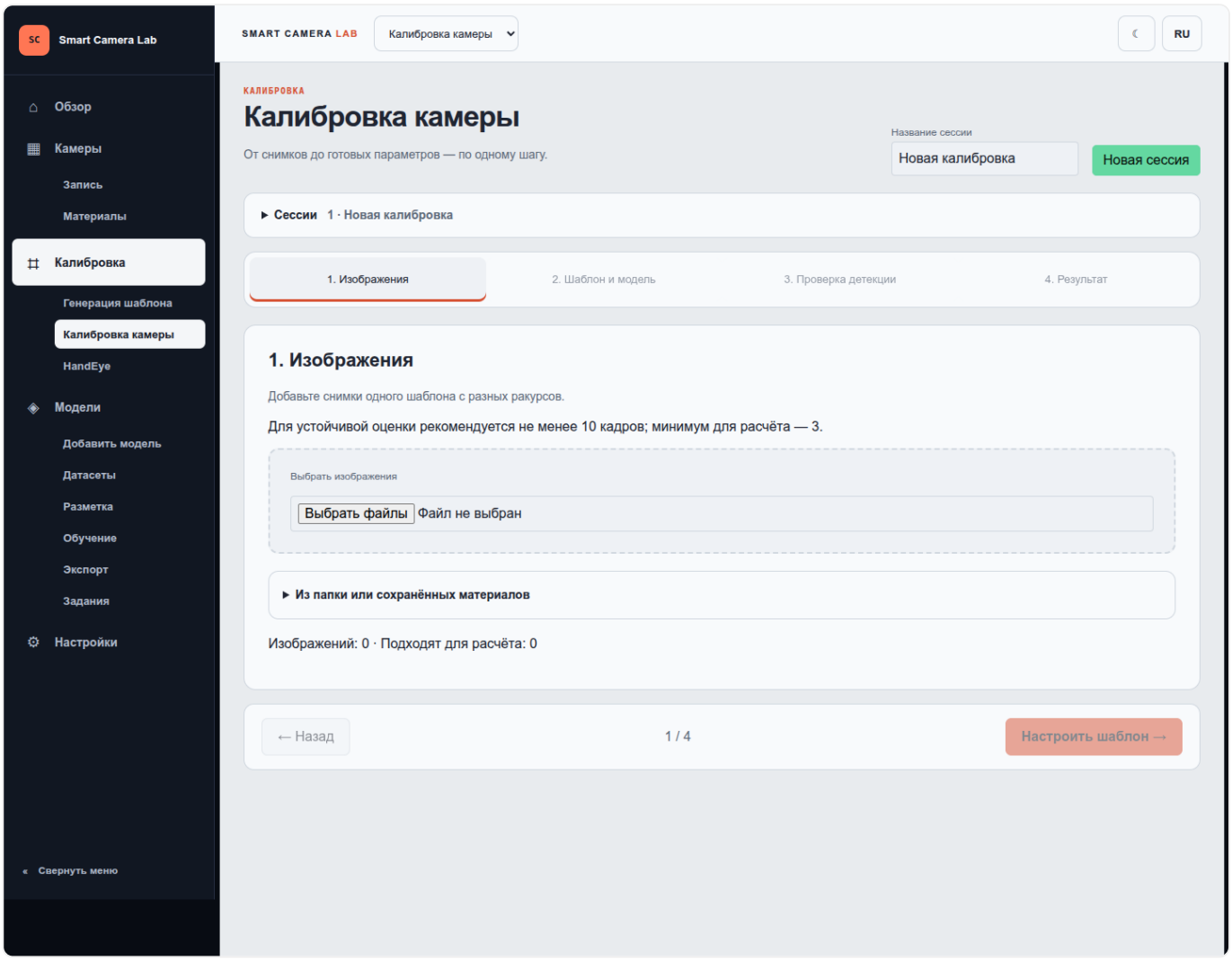
Масштаб экранного предпросмотра не задаёт масштаб печати. Если кнопка **Уменьшить шаблон до допустимого размера** изменила геометрию, скачайте новый JSON и используйте новые размеры. Не смешивайте старое описание с новой распечаткой.

3.1.2 Съёмка набора

Соберите не менее десяти качественных кадров. Минимум для расчёта — три, но это лишь техническое ограничение, а не критерий хорошей калибровки. Снимите мишень в центре, у краёв и углов изображения, с разными наклонами и на нескольких расстояниях. Она должна занимать заметную часть кадра, а точки — оставаться резкими.

Для сбора используйте серию фотографий одной камеры. Меняйте положение между снимками и останавливайте мишень на время экспозиции. Большое число одинаковых фронтальных кадров не заменяет разнообразия ракурсов. Исключите смаз, блики, заметный изгиб листа и кадры из другого разрешения.

3.1.3 Четыре шага мастера



1. Перейдите в **Калибровка → Калибровка камеры**, заполните **Название сессии**, например `front_1280x720_2026-10-06`, и нажмите **Новая сессия**.
2. На шаге **1. Изображения** нажмите **Выбрать изображения** и добавьте файлы из браузера. Для серверной папки или готовой записи раскройте **Из папки или сохранённых материалов**.
3. Нажмите **Настроить шаблон**, укажите геометрию и модель объектива, затем **Сохранить и проверить снимки**.
4. На шаге **3. Проверка детекции** проверьте наложенные точки, исключите плохие кадры и нажмите **Рассчитать калибровку**.
5. На шаге **4. Результат** оцените ошибки, скачайте отчёт и при необходимости примените параметры к камере.

1. Изображения

Для импорта с сервера введите **Путь к серверной папке** и нажмите **Загрузить папку**. Это путь на машине веб-сервера, а не на компьютере с браузером. Сервер разрешает импорт только из

предусмотренных каталогов. Если нужная папка недоступна, используйте загрузку файлов через браузер.

Для сохранённой фотосессии выберите запись в **Материалы…**, затем нужную камеру в **Камера…**, после чего нажмите **Импортировать из «Материалов»**. При многокамерной записи каждая камера калибруется отдельной сессией.

2. Шаблон и модель

В **Тип шаблона** выберите Chessboard, ChArUco или CircleGrid. Заполните параметры именно распечатанной мишени либо нажмите **Импортировать JSON шаблона**. Для ChArUco проверьте словарь, число клеток, размеры клетки и маркера, `Border bits`.

Автоматическое определение помогает подставить структуру шахматной доски или круговой сетки. Оно не определяет физический масштаб и модель объектива. На обрезанном изображении алгоритм может увидеть лишь часть сетки. После подстановки проверьте полное число точек и подтвердите физические размеры. ChArUco задаётся вручную или через JSON.

Модель камеры → Standard предназначена для стандартной модели дисторсии, **Fisheye** — для соответствующей широкоугольной модели. Само большое поле зрения не выбирает математическую модель автоматически. В **Расширенные параметры модели** доступны ограничения коэффициентов; для первого расчёта оставьте исходные параметры, если у вас нет оснований фиксировать отдельные величины.

3. Проверка детекции

Просмотрите кадры с найденными точками. Они должны попадать на реальные углы или центры, а не на фон. Флаг **Включён** определяет участие кадра в расчёте. Снятие флага сохраняет изображение; **Удалить кадр** удаляет его из сессии после подтверждения.

После добавления снимков используйте **Найти точки на новых кадрах**. Если изменились параметры шаблона или требуется повторная проверка всего набора — **Найти точки заново на всех кадрах**. Для расчёта нужны как минимум три включённых кадра с успешной детекцией.

4. Результат и оценка качества

Общий RMS показывает ошибку репроекции **в пикселях**: насколько найденные точки расходятся с проекцией рассчитанной модели. Это не погрешность расстояния в миллиметрах. Малый RMS полезен, но не доказывает правильный масштаб печати или точность измерений на всей сцене.

Раскройте **Технические детали и 3D-диагностика**, сравните ошибки отдельных кадров и покрытие ракурсами. Если несколько кадров резко хуже остальных, вернитесь к ним: проверьте смаз, геометрию мишени и найденные точки. Не удаляйте все сложные наклонные кадры только ради уменьшения общей цифры — набор должен описывать разные положения. Добавьте недостающие ракурсы и нажмите **Пересчитать калибровку**. При изменении набора старый результат отмечается как устаревший.

3.1.4 Применение и сохранение

Скачайте **ROS YAML** для ROS-потребителей, **JSON** для полного результата и **HTML-отчёт** для просмотра вне приложения. Сессии сохраняются в `<recordings_directory>/calibrations` и доступны после перезапуска.

В **Камера для применения** выберите камеру того же разрешения и нажмите **Применить к камере**. Для RealSense меняется `CameraInfo` цветного потока; заводская калибровка глубины сохраняется. Применение внутренних параметров само по себе не означает исправление всех исходных изображений.

При первом применении исходная настройка сохраняется как `calibration_baseline`. Кнопка **Вернуть исходную калибровку** восстанавливает прежний YAML либо заводскую калибровку, если собственного YAML не было. Сохраняйте исходный файл на диске: baseline хранит его путь, а не заменяет резервное копирование. Для V4L2 без calibration-файла `CameraInfo` не публикуется; фиктивная матрица не создаётся.

3.2 HandEye: привязка камеры к роботу

HandEye определяет положение оптического фрейма камеры относительно TCP/фланца или базы робота. Для этого SmartCamera сохраняет пары «изображение мишени — положение робота», а затем рассчитывает преобразование отдельно для каждой камеры. Приложение читает позы, но не отправляет роботу команды движения.

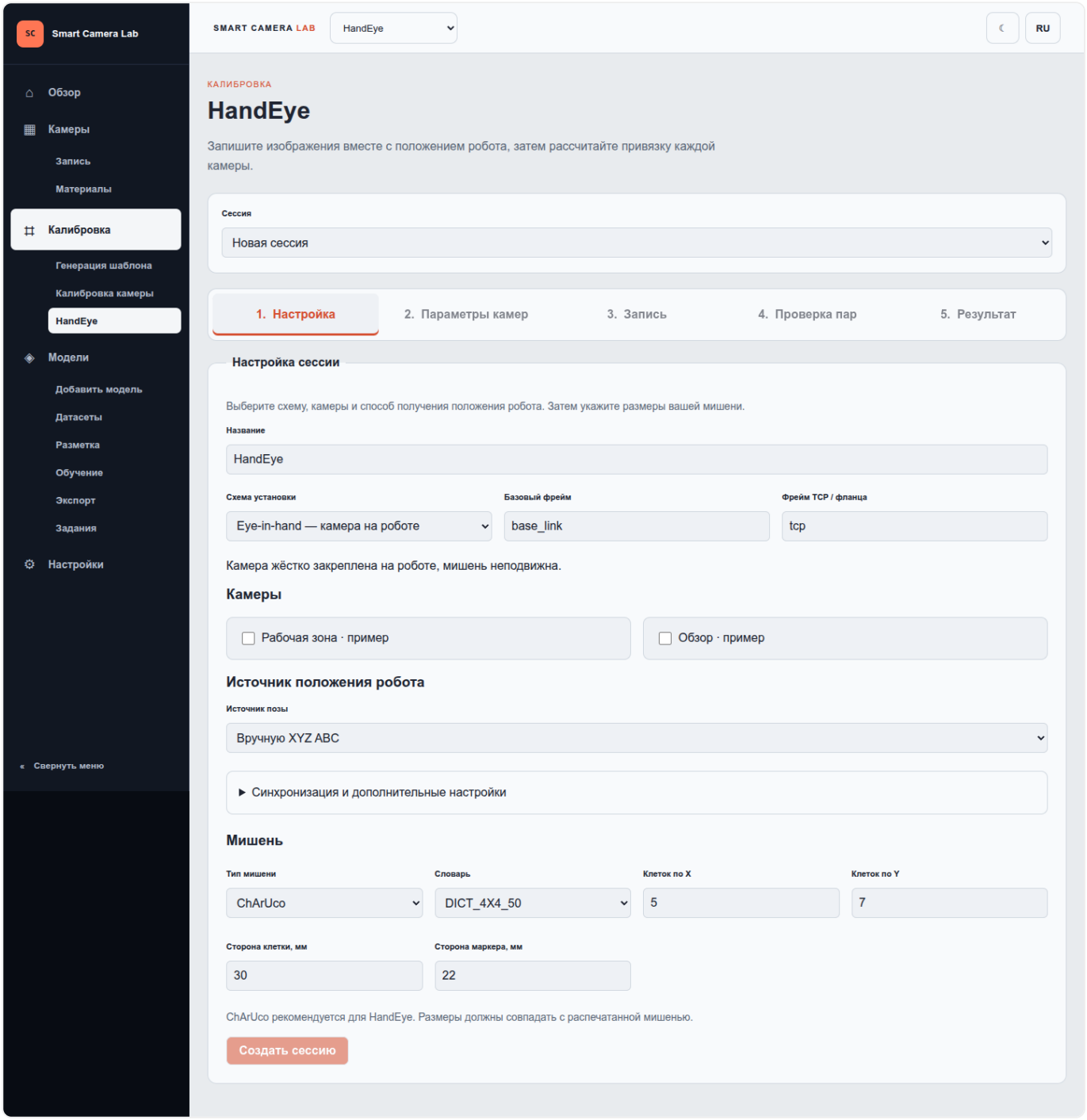
До начала работы добавьте и включите камеры, подготовьте жёсткую мишень с известными размерами и внутреннюю калибровку для текущего разрешения. Собрать изображения можно и до загрузки внутренних параметров; перед расчётом параметры понадобятся каждой камере.

3.2.1 Выбор схемы

Схема	Что закреплено на роботе	Что неподвижно	Результат
Eye-in-hand — камера на роботе	Камера	Мишень	Оптический фрейм камеры относительно TCP/фланца
Eye-to-hand — камера неподвижна	Мишень	Камера	Оптический фрейм камеры относительно базы

Крепление должно оставаться жёстким на протяжении всей сессии. Если камера или мишень смещается на креплении, собранные пары перестают описывать одно преобразование.

3.2.2 Шаг 1. Настройка



1. Откройте **Калибровка → HandEye**. В поле **Сессия** оставьте **Новая сессия**.
2. Введите **Название**, выберите **Схема установки** и отметьте камеры.
3. Укажите **Базовый фрейм** и **Фрейм TCP / фланца**, соответствующие источнику положения. Например, `base_link` и `tcp` подходят только если ваш издатель или FK использует именно эту пару.
4. Для каждой камеры проверьте поле **оптический фрейм**. Если `Image.header.frame_id` заполнен, он должен совпадать с этим значением.
5. Выберите **Источник позы**, заполните его параметры и размеры **Мишени**.
6. Нажмите **Создать сессию**. После последующих правок используйте **Сохранить настройки** либо **Отменить изменения** перед переходом на другой шаг.

После первого снимка схема, состав камер и фреймы фиксируются. Для их изменения создайте новую сессию. Это предотвращает смешивание данных с разным смыслом координат.

Для HandEye рекомендуется **ChArUco**: задайте словарь и фактическую геометрию распечатки. У симметричной шахматной доски или круговой сетки порядок точек может измениться при повороте; сохраняйте одну ориентацию координат мишени во всех кадрах. Все размеры мишени вводятся в миллиметрах.

Ручной ввод

Выберите **Источник позы → Вручную XYZ ABC**. На шаге записи вводятся координаты и углы текущего TCP/фланца относительно выбранной базы. По умолчанию используются миллиметры и градусы; доступны метры и радианы.

Конвенция ABC соответствует KUKA: `A` — вращение вокруг Z, `B` — вокруг Y, `C` — вокруг X, порядок `R = Rz(A) · Ry(B) · Rx(C)`. Это не три произвольных угла из любого интерфейса робота. Сверьте порядок и единицы источника, прежде чем переносить числа. Например, `X=500` в мм и `X=0.5` в м обозначают одно расстояние.

ROS 2 топик

Выберите **ROS 2 топик**, задайте **ROS-топик** и **Профиль сообщения**: `PoseStamped`, `TransformStamped` или **Настраиваемые поля**. Для первых двух профиль задаёт стандартный тип. `PoseStamped` не содержит дочерний фрейм: соответствие выбранному TCP должно следовать из документации издателя.

Для пользовательского типа укажите **Тип сообщения**, ориентацию, единицы и пути полей. Примеры путей: `position.x`, `data[0]`, `orientation.euler_deg.a`. Для ABC нужны `x, y, z, a, b, c`; для quaternion — `x, y, z, qx, qy, qz, qw`. Путь `stamp_path`, обычно `header.stamp`, должен вести к структуре `sec/nanosec`. Это пути данных, не Python-выражения. Тип сообщения должен быть установлен в окружении сервера.

lightweight-cobot / FK

Выберите **lightweight-cobot / FK**, проверьте **Топик суставов** `/joint_states` и **Сервис FK** `/compute_fk`. Используются `sensor_msgs/msg/JointState` и `moveit_msgs/srv/GetPositionFK`. При необходимости замените имена и фреймы на принятые в вашей системе.

FK рассчитывается по состоянию суставов, ближайшему ко времени кадра. Время позы берётся из исходного `JointState`, а имена и положения суставов сохраняются вместе с наблюдением. Доступность сервиса и корректность выбранного TCP проверяйте до массовой съёмки.

3.2.3 Шаг 2. Параметры камер

Для каждой камеры загрузите **ROS YAML** либо выберите **Готовая калибровка** из сессий монокалибровки. В HandEye сохраняется копия внутренних параметров. Разрешение должно совпадать с записываемыми кадрами. Поддерживаются `plumb_bob`, `rational_polynomial` и `equidistant`.

Если параметры ещё не готовы, кнопка **Записать пары, параметры позже** позволяет продолжить съёмку. Вернитесь к этому шагу до расчёта; наличие изображения само по себе не заменяет внутреннюю калибровку.

3.2.4 Шаг 3. Запись

1. Переместите робота средствами его системы управления так, чтобы мишень была видна выбранным камерам.
2. Остановите робота и дождитесь неподвижного резкого изображения.
3. Для ручного источника заполните XYZ ABC и отметьте **Робот остановлен, XYZ ABC соответствуют текущему положению**. Подтверждение требуется перед каждым снимком.
4. Для ROS или FK дождитесь **Данные поступают**. Проверьте показанные XYZ ABC, возраст и отсутствие ошибки; в **Разобранные поля** можно сверить исходные данные.
5. Нажмите **Снять с позой** и дождитесь подтверждения сохранения.
6. Соберите 15–25 положений, изменяя не только XYZ, но и наклоны вокруг нескольких осей. Сохраняйте видимость и резкость мишени.

Набор публикуется в manifest после сохранения изображений и поз всех выбранных камер. Если запись одного элемента не удалась, неполный набор удаляется. Не увеличивайте число повторов одного положения вместо разнообразных наклонов: расчёт проверяет пригодность движений.

В **Синхронизация и дополнительные настройки** по умолчанию допуск кадр/поза равен 100 мс, максимальный возраст — 500 мс, таймаут записи — 2 с. Межкамерный допуск по умолчанию составляет 30 мс. Источники должны использовать согласованную шкалу ROS clock; нулевые, старые или несогласованные timestamps препятствуют записи.

Сопоставление времени → Время получения — приблизительно предназначено для сообщений без timestamp. В этом режиме учитывается время получения и кадров, и поз; признак `approximate` сохраняется. Он не устраняет сетевую задержку, поэтому робот должен оставаться неподвижным.

QoS по умолчанию — `BEST_EFFORT`, `VOLATILE`, `KEEP_LAST(100)`; при необходимости reliability переключается на `RELIABLE`.

3.2.5 Шаг 4. Проверка пар

Нажмите **Проверить мишень**, выберите пару в списке и проверьте точки на изображениях. Фильтр помогает найти отклонённые или исключённые пары. Снимите **Использовать в расчёте** у плохого наблюдения: исходные изображения сохранятся.

Для ручных поз кнопка **Исправить XYZ ABC** открывает выбранную пару. Внесите исправление, подтвердите его и нажмите **Сохранить позу**. Исправляйте только по достоверным данным положения в момент этого снимка. **Удалить пару → Удалить навсегда** удаляет изображения; для временного исключения достаточно флага участия.

Скачать набор ZIP сохраняет полный набор для архива. Сессии находятся в `<recordings_directory>/handeye/<session_id>`. Правки данных делают прежний результат устаревшим.

3.2.6 Шаг 5. Результат

Проверьте наличие внутренних параметров всех камер и минимум трёх включённых пар, затем нажмите **Рассчитать HandEye**. Используется метод Park. Технический минимум не заменяет рекомендованные разнообразные 15–25 положений.

В результате показано направление `parent_frame ← child_frame`. Матрица `T_parent_child` преобразует координаты **из дочернего оптического фрейма камеры в родительский**:

```
p_parent = T_parent_child × p_child
Eye-in-hand: T_tcp_camera
Eye-to-hand: T_base_camera
```

В матрице перенос выражен в метрах; отдельное представление **XYZ ABC (mm, deg)** — в миллиметрах и градусах. Quaternion имеет порядок `xyzw`. Для преобразования в обратную сторону нужна обратная матрица, а не та же матрица с переименованными фреймами.

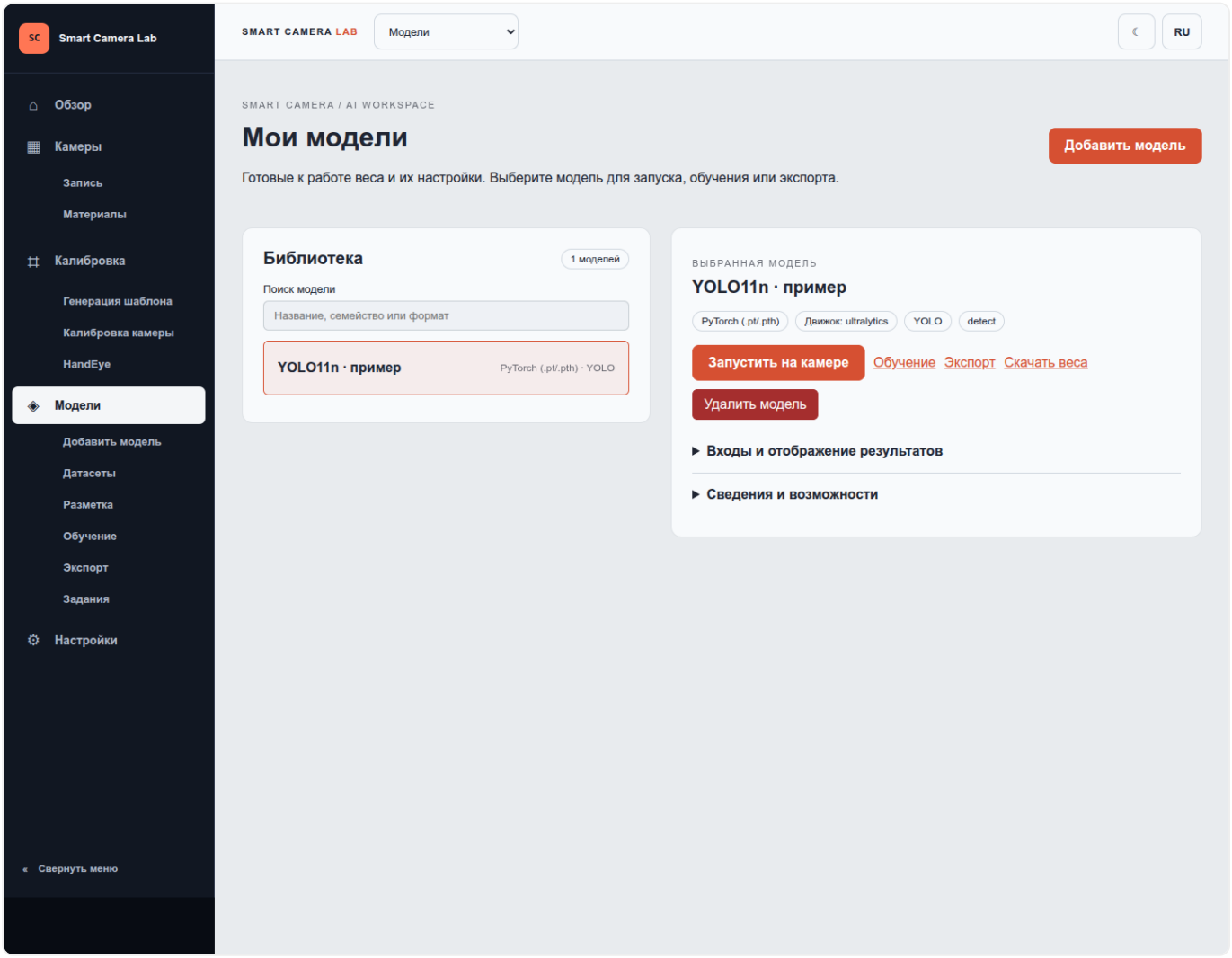
Согласованность мишени, RMS показывает разброс положения мишени в мм и градусах. В **Ошибки по наблюдениям** также приведена репроекция в пикселях. Это показатели согласованности набора, не гарантия абсолютной точности робота. При больших выбросах проверьте позу, единицы, фреймы, временное соответствие и внутреннюю калибровку соответствующей камеры.

Скачайте **JSON** или **YAML** рассчитанного результата. SmartCamera не публикует его в TF автоматически: дальнейшее подключение выполняется в вашей ROS-системе с проверкой направления преобразования и имён фреймов.

4. Модели

4.1 Модели: библиотека и ML-окружение

Раздел **Модели** объединяет веса Ultralytics, их настройки и результаты обучения. Откройте **Мои модели** (`/models`), чтобы выбрать сохранённую модель, скачать веса, перейти к обучению или экспорту. Само добавление весов в библиотеку не включает обработку камеры.



На снимке слева расположены поиск и список библиотеки, справа — карточка выбранной модели. Для повторения сценария сначала выберите запись в списке: действия относятся именно к ней.

4.1.1 Модель, запуск и задание

Понятие	Что хранится и выполняется	Где управлять
Модель	Веса, семейство, задача, формат и описание выходов	**Мои модели**
Запуск	Привязка модели к конкретной камере, устройство, частота обработки и подсказки	**Камеры → Нейросети**
Задание	Загрузка весов, обучение, валидация или экспорт с собственными параметрами и журналом	**Задания**
Датасет	Изображения, классы, геометрия разметки и подтверждение проверки	**Датасеты**, **Разметка**

Одну модель можно подключать к разным камерам. У каждого запуска свои настройки. Отключение запуска сохраняет модель и привязку; отвязка удаляет связь с камерой. Удаление модели из библиотеки — отдельное действие.

Данные ML сохраняются в подкаталоге `inference` каталога `recordings_directory`. После перезапуска сервера привязки остаются, но обработку нужно включить заново. Прерванные задания автоматически не возобновляются.

4.1.2 Подготовка Python для моделей

При установке через CLI окружение моделей уже готовится вместе с приложением. После `local-setup` или `docker-setup` отдельно запускать `examples/inference/setup_cpu.sh` не нужно.

В локальном режиме ML Python находится в `.venv-inference/` внутри исходников, в Docker — в `/opt/inference/` внутри образа. CLI передаёт путь процессам моделей. Веб-сервер и нейросети используют разные Python-окружения.

Для смены профиля сначала остановите приложение. Например, локально на CPU:

```
smartcamera stop

smartcamera local-setup --yes --profile cpu

smartcamera run
```

Для Docker:

```
smartcamera stop

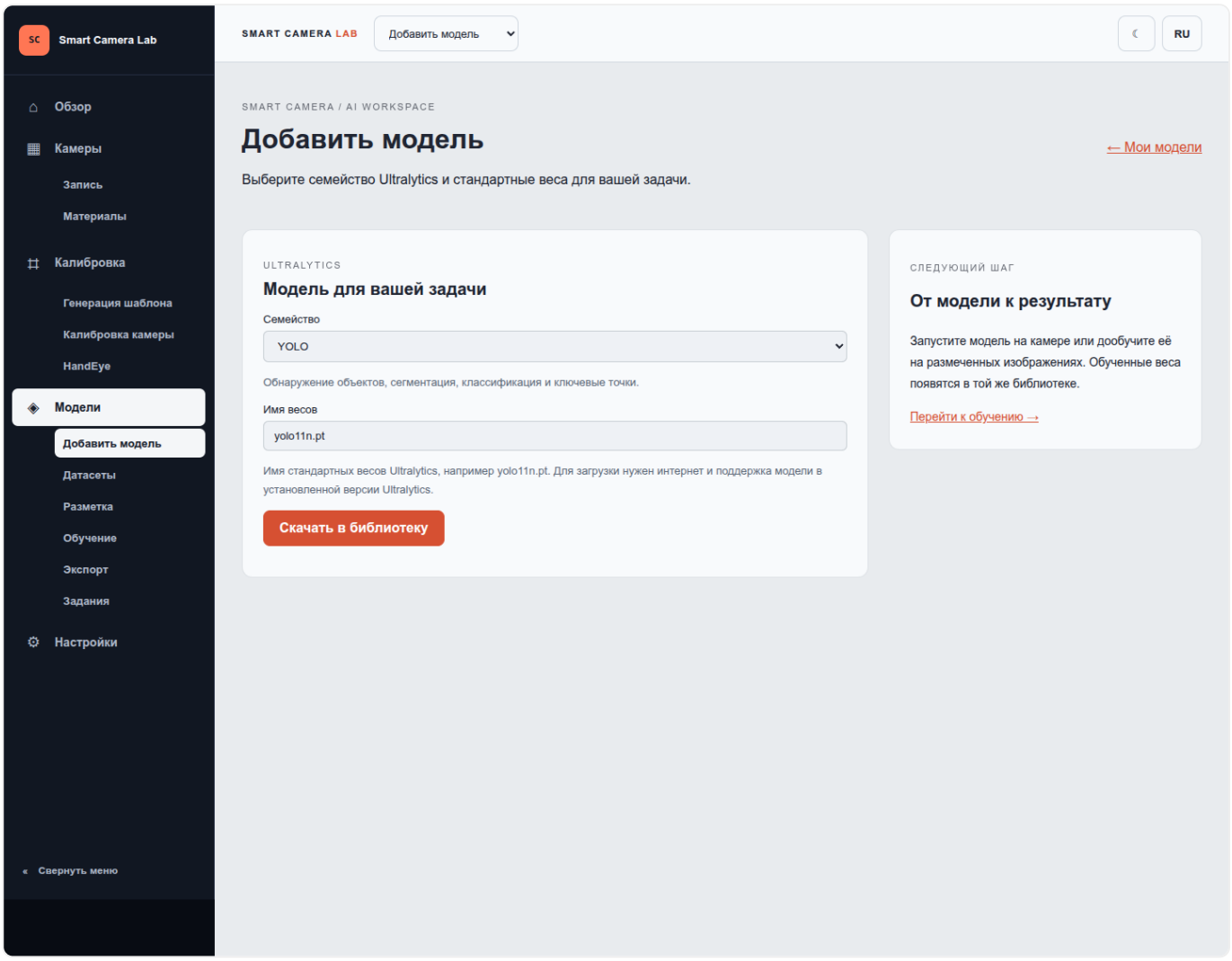
smartcamera docker-setup --yes --profile cpu

smartcamera run --detach
```

Для NVIDIA замените `cpu` на `cuda`. Драйвер GPU должен быть установлен заранее; для контейнера также нужен NVIDIA Container Toolkit. Подробности — GPU в Docker.

Если модель сообщает `ModuleNotFoundError`, проверьте журнал задания и выбранный профиль. Не устанавливайте PyTorch в Python веб-сервера: повторите подготовку нужного режима через CLI. После замены сборки PyTorch перезапустите приложение. Проверка CUDA при установке должна завершиться успешно, иначе обработка на GPU не будет работать. Разбор ошибок — в диагностике.

4.1.3 Добавление стандартных весов



1. На странице **Мои модели** нажмите **Добавить модель**.
2. В поле **Семейство** выберите подходящий вариант. Поле **Имя весов** заполнится примером для этого семейства.
3. Проверьте имя. Для первого опыта с обнаружением объектов можно оставить `YOLO` и `yolo11n.pt`.
4. Нажмите **Скачать в библиотеку**. Только это действие создаёт загрузку; изменение полей её не начинает.
5. После сообщения **Загрузка поставлена в очередь** нажмите **Открыть задание**.
6. Дождитесь успешного завершения и вернитесь в библиотеку. При ошибке откройте журнал задания.

Поле принимает имя стандартных весов с расширением `.pt`, а не путь к произвольному файлу или URL. Для загрузки нужны сеть, поддержка семейства установленной версией Ultralytics и

необходимые доступы к весам. Наличие семейства в списке не гарантирует доступность любых его весов в конкретном окружении.

Семейство	Пример в форме	Назначение
YOLO	`yolo11n.pt`	Детекция; другие веса семейства предназначены для сегментации, классификации, ключевых точек или OBB
YOLOWorld	`yolov8s-worldv2.pt`	Поиск объектов по текстовым классам
YOLOE	`yoloe-11s-seg.pt`	Детекция и сегментация с текстовыми и пространственными подсказками
RTDETR	`rtdetr-l.pt`	Детекция объектов
SAM, SAM2	`sam_b.pt`, `sam2_b.pt`	Интерактивная сегментация; SAM2 сопровождает выбранный объект
SAM3	`sam3.pt`	Сегментация с текстом или пространственными подсказками
MobileSAM, FastSAM	`mobile_sam.pt`, `FastSAM-s.pt`	Варианты сегментации
NAS	`yolo_nas_s.pt`	Детекция и валидация

Задачу определяют веса, а не подпись семейства. Например, детектор нельзя обучать на датасете классификации. Возможности обучения, экспорта и подсказок смотрите в **Сведения и возможности**. SAM/SAM2/SAM3/MobileSAM не обучаются и не экспортируются через этот интерфейс; NAS и FastSAM также не поддерживают обучение здесь.

4.1.4 Работа с карточкой

Поиск модели ищет по названию, семейству и формату. В карточке доступны **Запустить на камере**, **Обучение**, **Экспорт**, **Скачать веса**. Переходы к обучению и экспорту сохраняют выбор модели; запуск камеры настраивается отдельно по инструкции.

Блок **Входы и отображение результатов** описывает вывод результата. Для Ultralytics подготовкой изображения занимается адаптер; пороги и частоту обработки меняйте в параметрах запуска камеры. После редактирования карточки нажмите **Сохранить настройки**.

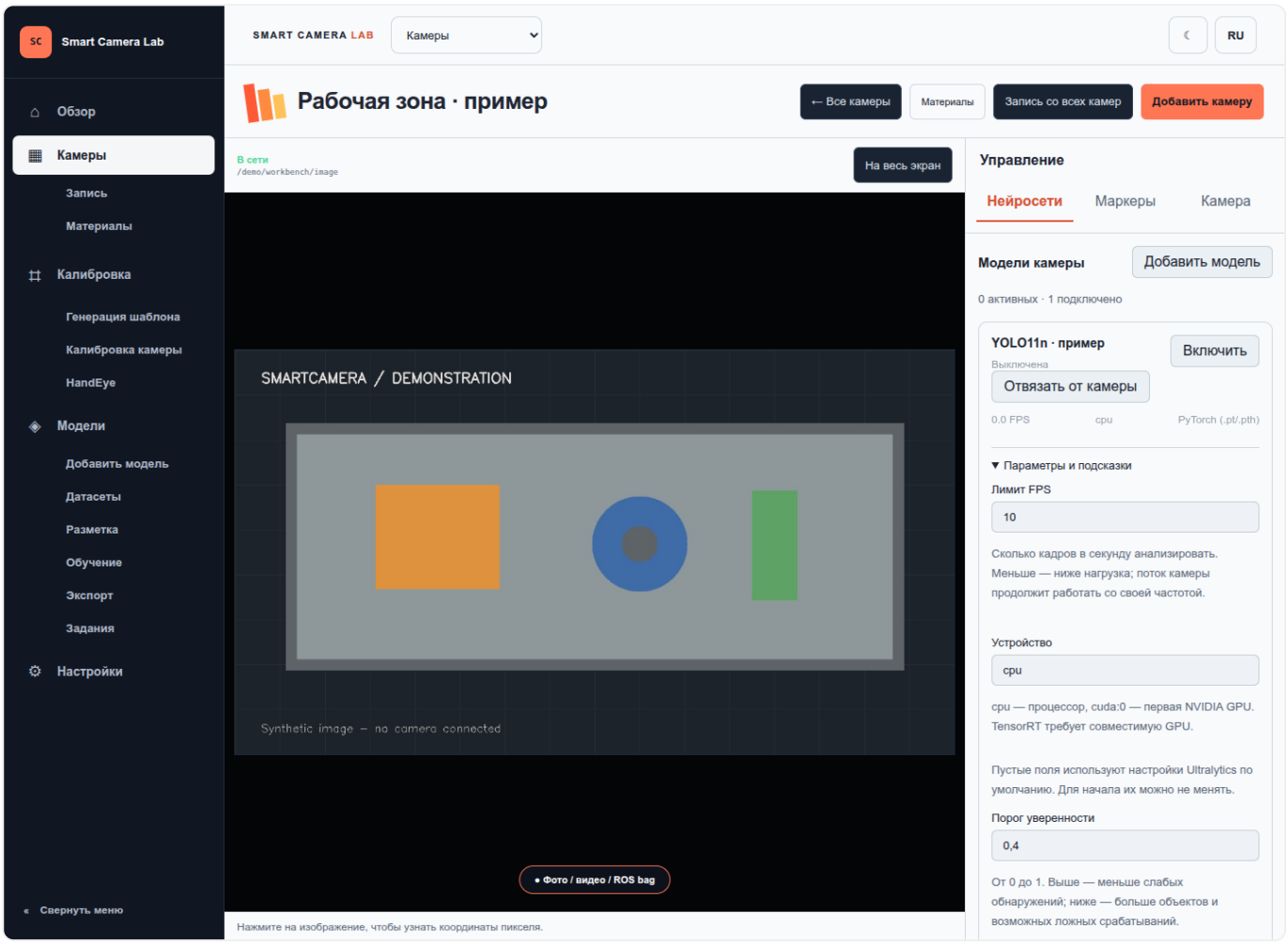
Библиотека принимает стандартные веса и результаты заданий приложения. Импорт внешних файлов моделей, регистрация серверных путей и Python-адаптеров через публичный интерфейс отключены. Не следует использовать старые примеры универсальных адаптеров как инструкцию добавления модели в текущую библиотеку.

Удалить модель открывает подтверждение **Удалить безвозвратно**. Удаляются запись, локальная копия весов и отключённые привязки; исходный файл и история заданий сохраняются. Перед удалением остановите использующие модель запуски.

Дальнейший порядок работы: проверить модель на камере, подготовить и разметить данные, затем дообучить и проверить результат. Для автоматизации камер отдельно описано управление через ROS 2.

4.2 Запуск модели на камере

Обработка настраивается для выбранной камеры в **Камеры → Нейросети**. Модель должна уже присутствовать в библиотеке, а камера — передавать кадры. Проверку подключения и выбор исходного потока выполните по руководству по камерам.



На снимке показано место управления привязками: здесь выбирают веса из библиотеки, включают обработку и открывают **Параметры и подсказки**. Параметры одной карточки относятся только к её запуску.

4.2.1 Подключить и включить

1. Откройте **Камеры**, выберите нужную камеру и убедитесь, что исходное изображение обновляется.
2. Перейдите в **Нейросети** и нажмите **Добавить модель**.
3. Найдите запись через **Поиск модели**. Поиск учитывает название, семейство и формат весов.
4. Выберите запись и нажмите **Подключить модель**.

5. В появившейся карточке откройте **Параметры и подсказки**, проверьте **Устройство** и **Лимит FPS**, затем нажмите **Применить настройки**.
6. Нажмите **Включить** и дождитесь состояния **Работает** и первого результата.

Подключение создаёт выключенную привязку. Оно не запускает загрузку новых весов и не включает inference автоматически. Если библиотека пуста, перейдите по **Библиотека моделей →** и добавьте веса.

Одна камера может иметь несколько независимых запусков. При этом каждый процесс расходует память и вычислительные ресурсы. Для первой проверки удобнее включить одну модель на CPU с небольшим лимитом, например 5 FPS, и затем менять по одному параметру.

4.2.2 Исходный поток и результат

Активная модель передаёт обработанный кадр в основное окно камеры. При нескольких запусках нажмите **Показать в основном окне** в карточке нужного: подпись станет **В основном окне**. Это переключает отображаемый результат, а не выключает остальные модели.

Переключатель **Исходный поток / Результат модели** позволяет вернуться к обычному изображению. Координаты и глубину измеряют в исходном потоке. Не используйте нарисованную рамку нейросети как замену измерению глубины: геометрия обнаружения и показания камеры имеют разное назначение.

Карточка показывает фактический FPS, устройство, номер кадра, длительность обработки в миллисекундах и временную метку `stamp`. Фактический FPS может быть ниже заданного ограничения. Ограничение задаёт максимум анализируемых кадров, а не гарантированную производительность и не частоту камеры.

В **Технические сведения** находятся ID запуска, число пропущенных кадров и версия настроек. Обработка не накапливает старые кадры в длинной очереди: при недостаточной производительности часть кадров пропускается. После отключения, изменения параметров или потери камеры старый результат убирается из просмотра.

4.2.3 Параметры обработки

После изменений нажимайте **Применить настройки** и дождитесь сообщения **Настройки запуска сохранены**. Пустые необязательные поля означают стандартные значения Ultralytics. Набор полей зависит от семейства и задачи.

Поле	Значение и практическое применение
Лимит FPS	От 1 до 120. Уменьшите при высокой нагрузке; поток камеры продолжает работать со своей частотой
Устройство	`cpu`, `cuda:0`, `cuda:1` и далее. Для GPU требуется подготовленное ML-окружение
Порог уверенности	От 0 до 1. Например, `0.5` отсекает менее уверенные обнаружения; снижение порога помогает увидеть слабые ответы, но увеличивает ложные срабатывания
Размер входа	Одна сторона, например `640`, или высота × ширина, например `384 × 640`. Больший размер повышает нагрузку
Порог перекрытия рамок (IoU)	Порог удаления перекрывающихся рамок при NMS. Применим только к моделям с этим этапом; у RTDETR и NAS поле не показывается
Максимум объектов в кадре	Ограничение количества результатов от 1 до 10 000
Классы объектов (ID)	Целые ID через запятую, например `0, 2`. Пустое поле оставляет все классы
Трекинг объектов	Сопоставление объектов между кадрами с сохранением идентификаторов, если модель объявляет такую возможность

Дополнительные параметры открываются одноимённым раскрывающимся блоком. ID класса относится к набору классов конкретных весов: нельзя переносить номера из другого датасета без проверки. В частности, ID `0` не обязан означать один и тот же объект у стандартной и дообученной модели.

Порог уверенности фильтрует отдельные обнаружения, а IoU регулирует обработку перекрытий. Если модель пропускает объект, сначала проверьте её классы и изображение, затем попробуйте снизить порог уверенности. Если появляются несколько рамок на одном объекте, изучайте перекрытия и применимость NMS. Для классификации и интерактивной сегментации часть детекторных параметров отсутствует.

Для экспортированных ONNX/TensorRT размер входа должен соответствовать параметрам экспорта. Простое увеличение этого поля не делает статический engine динамическим.

4.2.4 Текст, точки и рамка

Если семейство поддерживает текст, появится **Текст / классы через запятую**. Для YOLOWorld/YOLOE можно ввести `person, vehicle` и нажать **Применить запрос**. Используйте понятные названия искомых объектов; обычный детектор YOLO не получает новые классы от ввода текста.

Для пространственной подсказки:

1. Откройте **Параметры и подсказки** и нажмите **Выбрать точки / рамку на кадре**.
2. На зафиксированном изображении выберите режим **Объект (+)**, **Фон (−)** или **Рамка: два угла**. У семейства с поддержкой только рамок доступен соответствующий режим.
3. Отметьте объект и, при необходимости, фон. Для рамки щёлкните по двум противоположным углам.
4. Исправьте выбор через **Очистить выделение** или нажмите **Применить выделение**.

SAM2 запоминает выбранный объект и сегментирует его на последующих кадрах. Для других пространственных адаптеров подсказка относится к зафиксированному кадру. У SAM3 в одном запуске выбирают текст либо пространственную подсказку. Перенастройка, отключение и потеря потока уничтожают память модели. **Сбросить подсказки** очищает текущий запрос; для нового объекта зафиксируйте новый кадр.

4.2.5 Смена модели и остановка

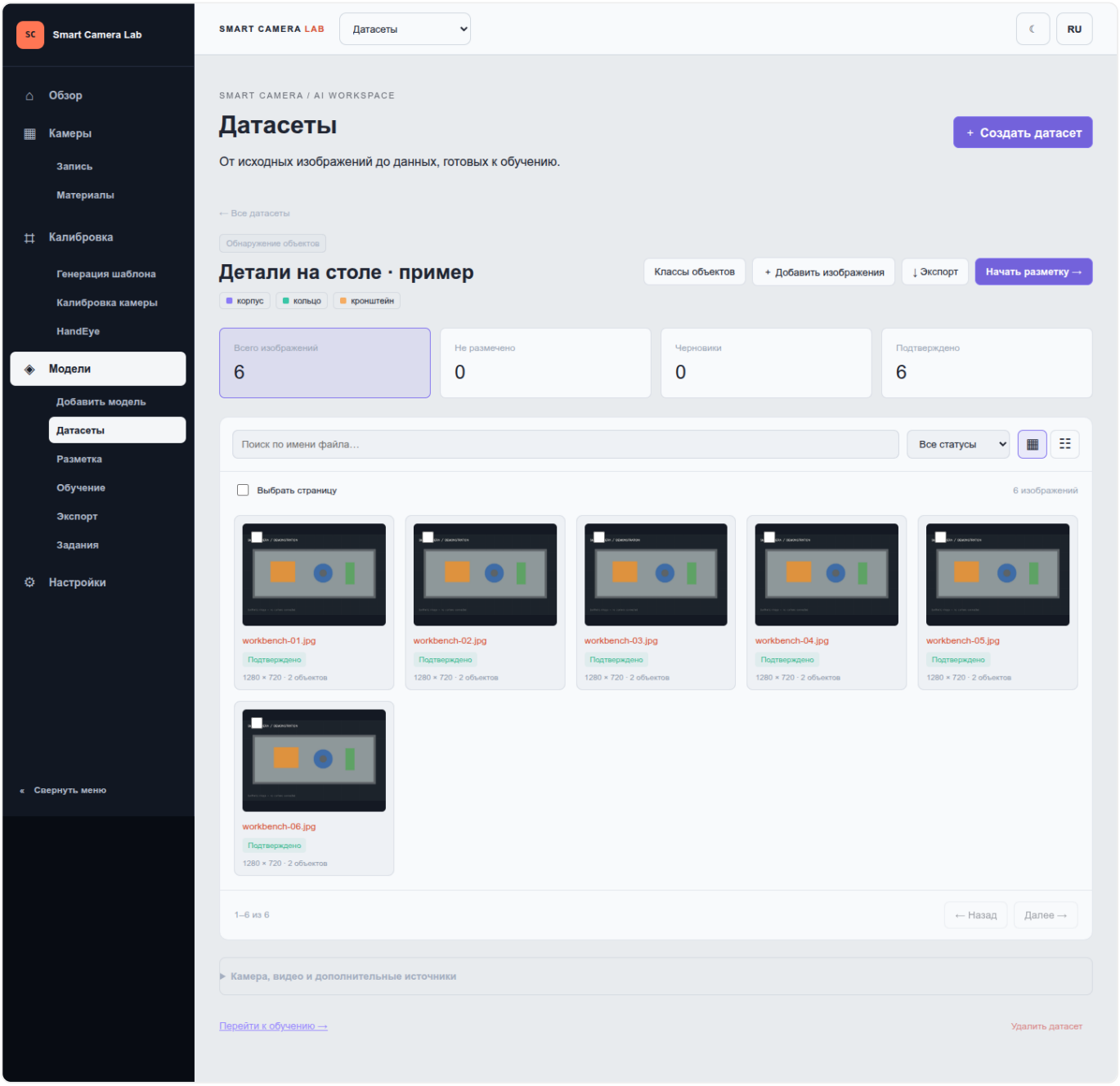
Для сравнения весов нажмите **Отключить** у прежнего запуска, подключите другую модель, настройте и включите её. Если нужны обе, оставьте обе привязки и выбирайте результат кнопкой **Показать в основном окне**. Прямой замены весов в параметрах существующего запуска нет.

Отключить сохраняет настройки для следующего включения. **Отвязать от камеры** открывает подтверждение **Остановить и отвязать**: обработка остановится, привязка исчезнет, веса останутся в библиотеке, связи с другими камерами сохранятся.

Состояние **Ожидание кадра** означает, что запуск ждёт изображение; **Камера недоступна** требует проверки источника. При **Ошибка** прочитайте сообщение карточки и проверьте интерпретатор, устройство и совместимость весов. Если обучение остаётся в очереди, отключите live-запуски на том же устройстве. Дополнительные проверки — в диагностике; управление источниками через ROS описано отдельно.

4.3 Датасеты и разметка

Датасет объединяет изображения, список классов и разметку для одной задачи. Рабочий порядок: создать набор, импортировать изображения, разметить и подтвердить их, затем передать в обучение или скачать архив. Страница **Датасеты** (`/datasets`) отвечает за состав данных; **Разметка** (`/annotations`) — за отдельное изображение.



В библиотеке выберите карточку набора. Внутри доступны счётчики **Всего изображений**, **Не размечено**, **Черновики**, **Подтверждено**, поиск по имени файла и переключение **Галерея /**

Таблица. Нажатие счётчика фильтрует список. **Начать разметку →** формирует очередь из текущего поиска и фильтра; щелчок по изображению открывает его непосредственно.

4.3.1 Создание и классы

1. Нажмите + **Создать датасет** и задайте **Название**, например `Детали на столе`.
2. Выберите **Тип разметки** по задаче будущей модели.
3. Введите **Классы через запятую**, например `bolt, nut`.
4. Для ключевых точек заполните **Точки скелета через запятую** и **Связи точек, индексы с нуля**. Например, точкам `start, end` соответствует связь `[[0, 1]]`.
5. Нажмите **Создать и загрузить данные →** и добавьте изображения в открывшемся окне импорта.

Задача	Что размечать	Пример
`detect`	Прямоугольник вокруг каждого объекта	Найти все гайки
`segment`	Полигон по границе объекта	Выделить поверхность детали
`classify`	Один класс целого изображения	Определить тип детали на снимке
`pose`	Рамка объекта и упорядоченные ключевые точки	Отметить начало и конец детали
`obb`	Четыре последовательных угла повёрнутой рамки	Найти ориентированные детали

Класс — категория объекта, а аннотация — её конкретный экземпляр и геометрия на изображении. Порядок классов задаёт числовые ID, начиная с нуля. Не меняйте смысл одного класса от изображения к изображению.

Классы объектов открывает редактирование: **Добавить класс**, переименование, **Удалить**, затем **Сохранить классы**. Удаление класса удаляет его аннотации, но оставляет изображения. Затронутую разметку нужно проверить и подтвердить повторно.

4.3.2 Импорт изображений и готовой разметки

Нажмите + **Добавить изображения**. В окне **Импорт в датасет** доступны три источника:

- **С компьютера** — файлы, папка изображений, ZIP изображений или YOLO-датасета.
- **По ссылке** — прямая общедоступная HTTP(S)-ссылка на файл. Ссылка на страницу просмотра облачного диска не равнозначна ссылке на файл.
- **S3 хранилище** — S3-совместимое хранилище, включая MinIO. Укажите endpoint, region, bucket и prefix, проверьте подключение и выберите объекты.

Затем нажмите **Импортировать**. Импорт копирует данные в локальное хранилище и выполняется в фоне. Окно можно закрыть и открыть снова для просмотра результата. Это разовая загрузка: дальнейшие изменения в S3 автоматически не синхронизируются.

Для S3 пустые ключи используют серверную цепочку credentials. Введённые ключи не сохраняются; для повтора их вводят заново. S3 использует Python веб-сервера и `boto3`, а не отдельный ML Python. Для локального MinIO используйте вкладку S3, поскольку импорт по ссылке предназначен для общедоступных файлов.

Ограничения импорта: 32 MiB и 40 мегапикселей на изображение; 1 GiB на ZIP; до 2 GiB и 10 000 элементов после распаковки. Ошибки отдельных элементов смотрите в **Подробности ошибок**. **Отменить импорт** сохраняет уже добавленные изображения. **Повторить без дубликатов** пропускает успешно обработанные элементы исходного задания; для локального источника нужно снова выбрать те же файлы.

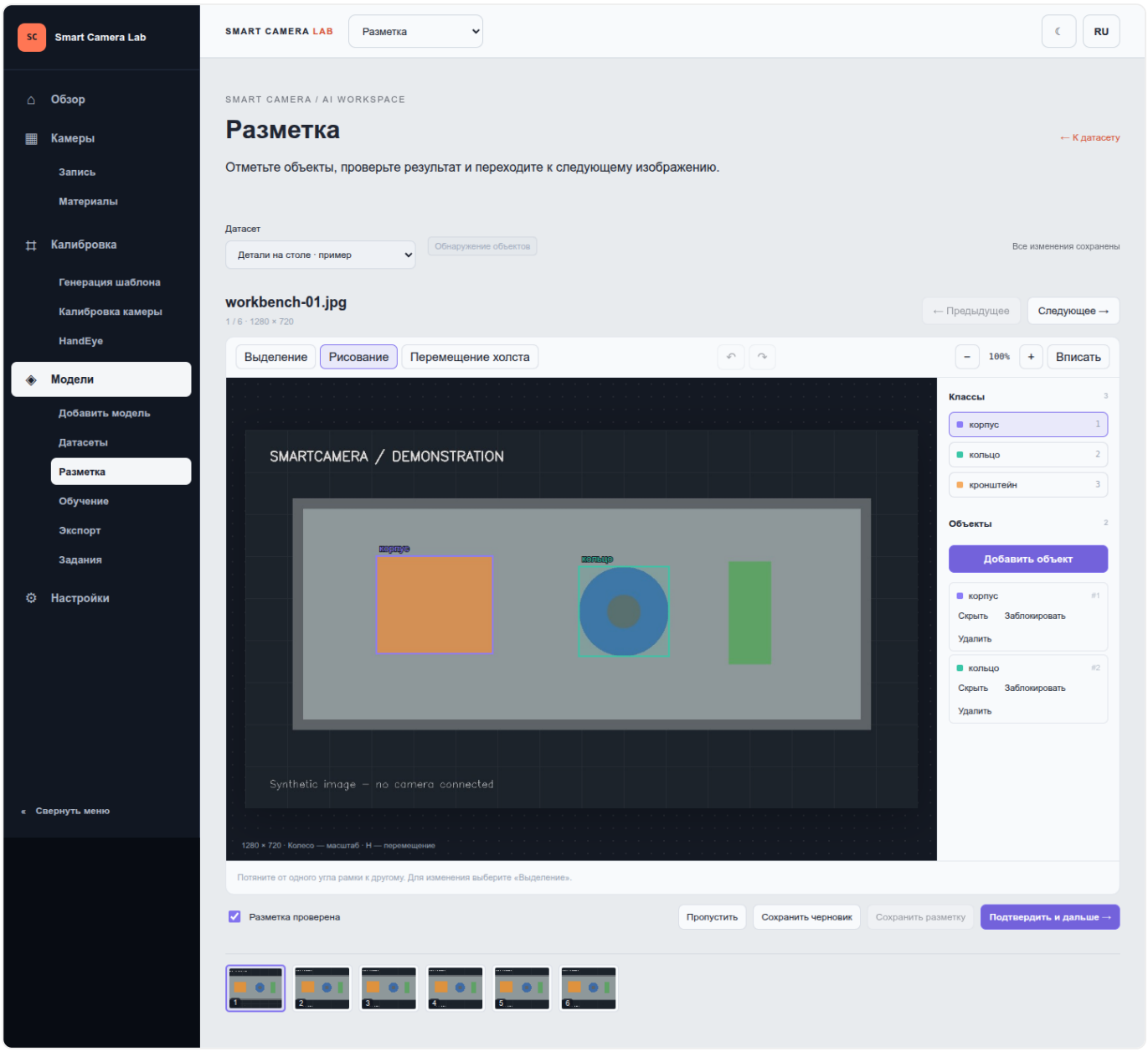
Готовый YOLO-датасет описывается через `data.yaml`, изображения и соответствующие метки. Для классификации YAML указывает каталоги train/val/test с папками классов. Проверьте совпадение задачи и классов с создаваемым набором. Импорт COCO, Pascal VOC и SmartCamera JSON пока не реализован, хотя эти форматы доступны для экспорта.

В **Камера, видео и дополнительные источники** находятся ещё четыре способа:

Источник	Действие
Кадр камеры	Введите **ID камеры** и нажмите **Снять кадр**; камера должна передавать изображение
Кадры из видео	Укажите **Путь к записанному видео** на сервере и **Интервал, секунд**, затем **Извлечь до 100 кадров**
YOLO на сервере	Введите **Путь к data.yaml на сервере** и нажмите **Импортировать разметку**
Предложение модели	Введите ID запуска из технических сведений карточки модели и нажмите **Добавить результат модели**

Путь на сервере относится к машине SmartCamera, а не компьютеру с браузером. Предложения нейросети проверяйте вручную: ошибки модели не должны автоматически становиться эталонной разметкой.

4.3.3 Разметка изображения



На снимке найдите инструменты слева, классы и объекты справа, действия сохранения под изображением. Для детекции выберите класс, инструмент **Рисование (R)** и протяните рамку от одного угла к другому. Для следующего объекта нажмите **Добавить объект**, выберите класс и нарисуйте новую фигуру.

Для сегментации последовательно отметьте вершины и нажмите **Замкнуть полигон**; нужны хотя бы три точки. Для OBB отметьте четыре угла по порядку. Для pose нарисуйте рамку и расставьте точки в порядке подсказки; видимость каждой точки задаётся как **Видима**, **Скрыта** или **Не размечена**. Для classify выберите класс изображения справа.

Инструмент **Выделение (V)** позволяет перемещать объект и менять углы или вершины. **Перемещение холста (H)** сдвигает видимую область. Используйте увеличение для мелких деталей.

Скрытие объекта убирает его только с холста; блокировка защищает от изменения и удаления. Для удаления выберите объект и нажмите `Delete` либо его кнопку **Удалить**.

Клавиши	Действие
`V`, `R`, `H`	Выделение, рисование, перемещение холста
`1` – `9`	Выбор класса
`Ctrl/Cmd+Z`	Отменить действие
`Ctrl/Cmd+Shift+Z`	Повторить действие
`Delete`	Удалить выбранный незаблокированный объект

Незавершённую фигуру закончите либо нажмите **Отменить фигуру**: пока она строится, сохранение недоступно. Undo/redo работает с редактированием текущего изображения и не заменяет сохранение на сервере.

4.3.4 Черновик и подтверждение

Автосохранения нет. **Сохранить черновик** сохраняет работу, но оставляет изображение неподтверждённым. **Сохранить разметку** сохраняет текущее состояние флажка **Разметка проверена**. **Подтвердить и дальше →** сохраняет подтверждённую разметку и переходит к следующему неподтверждённому изображению очереди.

Пропустить не подтверждает изображение. При выходе с правками редактор предлагает **Остаться**, **Отбросить изменения** или **Сохранить и перейти**; последний вариант сохраняет черновик. **Отменить изменения / перечитать** возвращает сохранённое состояние после подтверждения. При ошибке сохранения правки остаются в редакторе.

В обучение и экспорт входят только подтверждённые изображения. Пустой подтверждённый кадр может быть отрицательным примером для детекции. Для классификации нужен назначенный класс. Например, кадр пустого стола полезен детектору гаек, но кадр с пропущенной гайкой создаёт неверный отрицательный пример.

4.3.5 Экспорт и удаление

Нажмите **↓ Экспорт**, выберите формат и проценты **Обучение**, **Валидация**, **Тестирование**; сумма должна быть 100%. Нажмите **Скачать архив**. Разделение применяется к ZIP, исходный набор не меняется. Одинаковый набор изображений и **Ключ перемешивания** воспроизводят выборки.

YOLO поддерживает все задачи; COCO — все перечисленные, кроме классификации; Pascal VOC — детекцию; SmartCamera JSON — полную разметку всех задач. В COCO повёрнутая рамка сохраняется как полигон с охватывающим bbox. JSON включает `manifest.json`, изображения, классы, нормализованную геометрию, имена точек и скелет. Перед внешним обучением распакуйте ZIP.

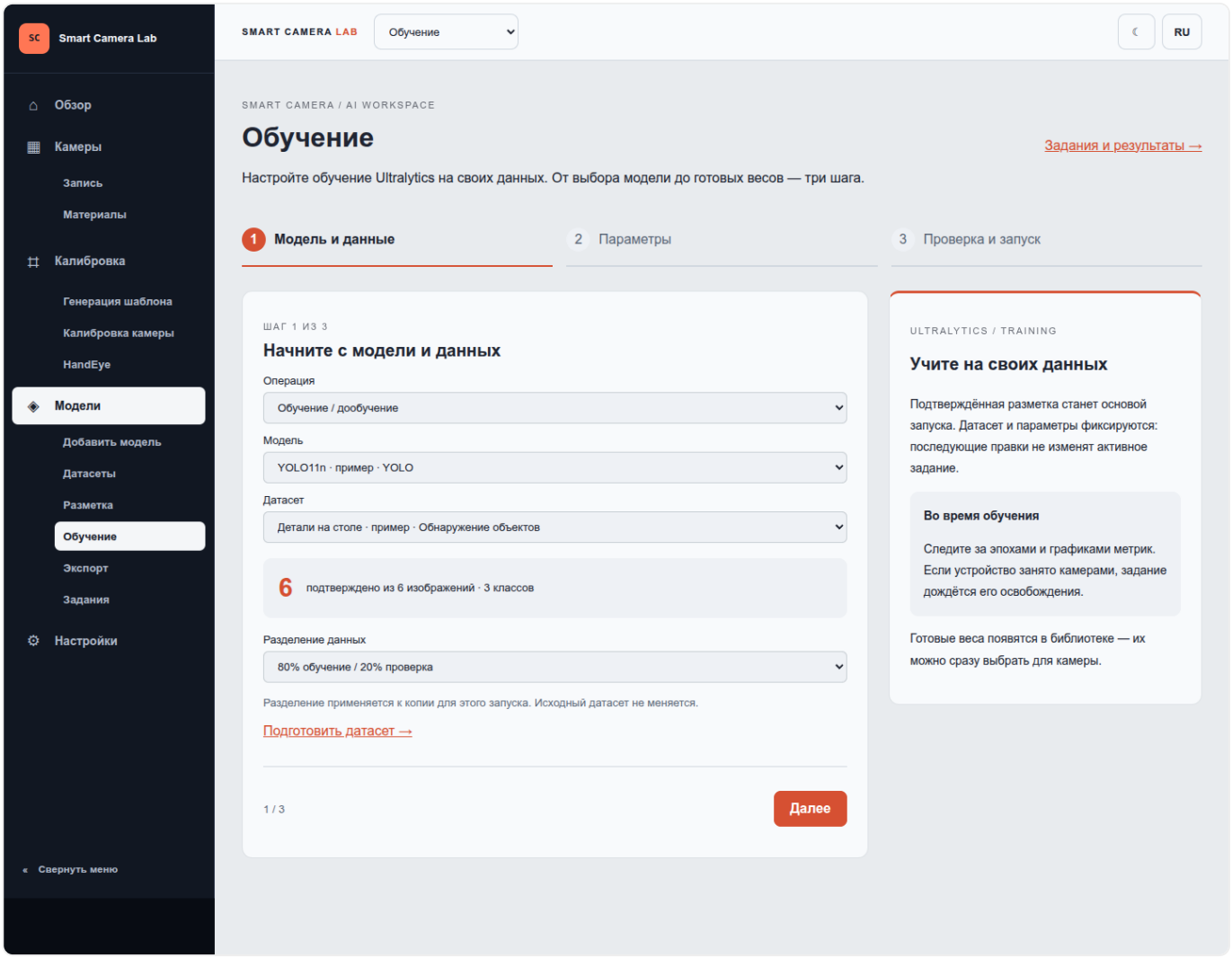
Удалить изображения удаляет выбранные файлы и их разметку после подтверждения. **Удалить датасет** удаляет локальные данные набора; исходные импортированные файлы, модели и снимки

датасета в заданиях сохраняются. Для продолжения нажмите **Перейти к обучению →**. Проблемы импорта и сохранения разбираются в диагностике.

4.4 Обучение, валидация и экспорт

Мастер **Обучение** (`/training`) создаёт серверное задание Ultralytics. При обучении изменяются веса модели; при валидации оцениваются существующие веса на размеченных данных. Экспорт меняет формат представления модели и сам по себе не обучает её. Обработка текущих кадров камеры настраивается отдельно в Нейросетях.

До начала подготовьте ML-окружение и модель, а затем датасет с подтверждённой разметкой. Для первого запуска достаточно небольшой проверки всего процесса; несколько эпох на нескольких изображениях не являются проверкой качества для реального применения.



На снимке показаны этапы **Модель и данные**, **Параметры**, **Проверка и запуск**. Если открыть **Обучение** из карточки модели или **Перейти к обучению →** из датасета, соответствующий выбор уже будет заполнен.

4.4.1 Шаг 1. Модель и данные

1. В поле **Операция** выберите **Обучение / дообучение** либо **Валидация**.
2. Выберите **Модель**. Для обучения нужны обучаемые веса Ultralytics `.pt`; ONNX/TensorRT используются для выполнения, а не как исходные веса обучения в этом мастере.
3. Выберите **Датасет** с той же задачей: `detect`, `segment`, `classify`, `pose` или `obb`.
4. Проверьте число подтверждённых изображений. Для classify учитываются только изображения с назначенным классом.
5. В **Разделение данных** выберите **80% обучение / 20% проверка** либо **Сохранить выборки датасета**.
6. Нажмите **Далее**.

Автоматическое разделение требует минимум три пригодных подтверждённых изображения. При сохранении исходных выборок нужны подтверждённые данные в `train` и `val`. Это условие проверяется и при выборе валидации. Если исходный набор импортирован с продуманным разделением, сохраните его; случайное перераспределение соседних кадров одной сцены может дать слишком оптимистичную оценку качества.

Разделение выполняется только для копии задания. При создании сохраняются снимок изображений, разметки и параметры. Правки исходного датасета после запуска не изменяют обучение. Чтобы обучиться на исправленной разметке, создайте новое задание.

Не все семейства поддерживают обучение: смотрите **Сведения и возможности** карточки. Интерфейс сообщает **Эти веса не поддерживают обучение** или **Задачи модели и датасета не совпадают** до отправки задания. Python-тренеры и внешние пути checkpoint через публичный интерфейс не принимаются.

4.4.2 Шаг 2. Параметры

Поле	Как выбрать
Устройство	`cpu` для процессора или `cuda:0` для первой NVIDIA GPU с подходящим окружением
Эпохи	Число проходов по обучающим данным; поле показывается только для обучения
Размер изображения	Положительное целое число пикселей для входа модели
Batch	Число изображений в одном шаге; при нехватке памяти уменьшайте
Параметры задания (JSON)	Дополнительные параметры Ultralytics в блоке **Дополнительные параметры**

Пример проверки на CPU: **Эпохи** `3`, **Размер изображения** `320`, **Batch** `1`, дополнительные параметры `{}`. Это пример небольшого технического запуска, а не рекомендуемый режим

окончательного обучения. После проверки данных подберите размер и длительность по задаче и ресурсам.

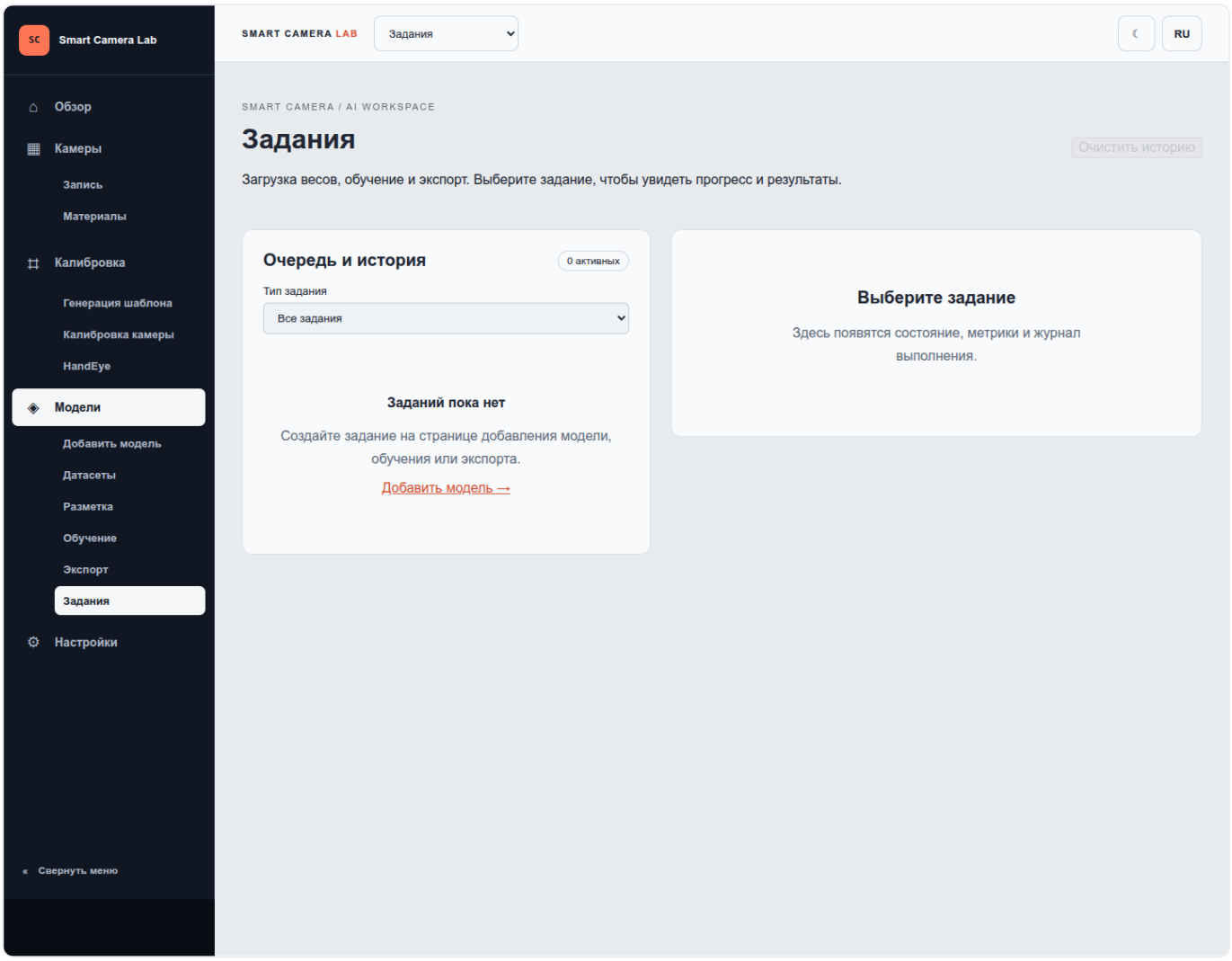
Числа в форме должны быть положительными целыми. Поля формы имеют приоритет над одноимёнными ключами JSON: запись `{"batch": 8}` не заменит значение **Batch**, установленное в форме. Дополнительные параметры должны соответствовать установленной версии Ultralytics и выбранной операции. CPU обычно существенно медленнее подходящей GPU; выбор `cuda:0` не устанавливает драйвер или CUDA-сборку PyTorch автоматически.

4.4.3 Шаг 3. Проверка и запуск

Проверьте операцию, модель, датасет, число подтверждённых изображений, выборки, устройство и размер/batch. В **Все параметры запуска** показан отправляемый набор параметров. Вернитесь кнопкой **Назад**, если что-либо выбрано неверно.

Нажмите **Начать обучение** либо **Запустить валидацию**, затем **Следить за обучением →**. Задание выполняется на сервере: закрытие вкладки его не останавливает. Выключение сервера, напротив, прерывает работу; после перезапуска автоматического продолжения нет.

4.4.4 Очередь, метрики и журнал



В **Заданиях** (`/jobs`) выберите запись в **Очередь и история**. Фильтр **Тип задания** отделяет загрузку весов, обучение, валидацию и экспорт. Справа показаны устройство, состояние, сообщение, параметры и результат.

Состояние	Значение
Подготовка	Создание снимка и подготовка задания
В очереди	Ожидание возможности занять устройство
Выполняется	Работает отдельный процесс задания
Завершено	Результат сохранён
Отменено	Задание остановлено пользователем
Прервано	Выполнение не завершилось, например после перезапуска сервера
Ошибка	Требуется прочитать сообщение и журнал

Фоновые задания ждут остановки live-запусков на том же устройстве. Если обучение на `cuda:0` остаётся в очереди, отключите модели камер на `cuda:0`. Переключение основного окна обратно на исходный поток не освобождает устройство: нужно нажать **Отключить** в карточках запусков.

Прогресс обучения содержит номер эпохи, процент и графики числовых метрик. До первой валидации надпись **Метрик пока нет** ожидаема. Данные обновляются после завершения эпохи и валидации, поэтому неподвижный график между эпохами сам по себе не означает зависание. Завершённое задание показывает 100%, даже если обучение закончилось раньше запрошенного числа эпох.

Названия метрик приходят от Ultralytics и зависят от задачи. Сравнивайте модели на одной проверочной выборке и с одинаковым смыслом метрик. Уменьшение обучающей ошибки ещё не доказывает улучшение на новых сценах; проверяйте результат валидации и отдельные реальные изображения.

Загрузить журнал получает журнал по нажатию. **События и диагностика**, **Параметры запуска** и **Результат задания** позволяют проверить подробности. При ошибке окружения сопоставьте путь ML Python с установленными пакетами, исправьте среду, перезапустите сервер и создайте задание заново. См. диагностику.

4.4.5 Готовые веса и checkpoint

После успешного обучения нажмите **Открыть готовую модель**. Результат регистрируется в библиотеке; используется лучший checkpoint, если он есть, иначе последний. Скачайте веса через **Скачать веса**, затем подключите новую запись к камере. Уже существующие привязки к исходной модели не становятся новой моделью автоматически.

Остановить доступно для активного задания. Для остановленного обучения предлагается **Возобновить из checkpoint**. Продолжение создаёт новое задание и использует исходный снимок датасета, а не его текущую отредактированную версию.

Возобновление требует сохранённого checkpoint; Ultralytics использует `last.pt` с состоянием обучения. Файл после окончательного завершения может быть очищен библиотекой от нужного состояния, поэтому сама видимость кнопки не гарантирует успешное resume. Если checkpoint отсутствует или не подходит, выберите зарегистрированную модель и создайте обычное дообучение.

Убрать из истории скрывает одну завершённую запись. **Очистить историю** после **Подтвердить очистку** убирает завершённые, ошибочные и полностью остановленные записи независимо от фильтра; активные остаются. Модели, журналы и файлы результатов сохраняются. Это действие не освобождает место на диске.

4.4.6 Экспорт в ONNX и TensorRT

Из карточки модели нажмите **Экспорт**. Выберите ONNX или TensorRT engine, точность FP32/FP16/INT8, устройство и batch. По умолчанию размер входа сохраняется; для переопределения отметьте

Изменить размер входа и укажите сторону `640` или `480, 640` — высоту и ширину. Возможность экспорта зависит от семейства и весов.

TensorRT требует совместимой NVIDIA GPU и установленного окружения TensorRT. INT8 требует калибровочного датасета и поддержки калибровки моделью; для задания сохраняется его снимок. Выбор точности в форме не гарантирует поддержку сочетания модель/устройство/формат.

Нажмите **Начать экспорт**: создаётся отдельное задание. Приложение проверяет загрузку результата и пробный запуск, а при сопоставимых выходах — численное соответствие. После успеха экспортированная модель появляется отдельной записью библиотеки. При ошибке совместимости engine читайте журнал задания. Экспорт не запускает обработку камеры: новую запись нужно подключить и включить, согласовав размер входа с экспортом.

5. ROS 2 и API

5.1 Топики и сервисы ROS 2

ROS-интерфейс использует те же камеры, модели и запуски, что и веб-панель. Нужен запущенный `web_server` (входит в обычный launch); браузер открывать не надо. Целевая версия — ROS 2 Jazzy. После обновления пересоберите пакеты и перезапустите приложение; на другом компьютере соберите и подключите `smart_camera_interfaces`. На обоих компьютерах используйте одинаковый `ROS_DOMAIN_ID` и настройки DDS, разрешающие обнаружение участников в вашей сети.

Запустите приложение через `smartcamera run` или `smartcamera run --detach`. Команды из этого раздела выполняются в отдельном терминале с ROS-окружением. Для локальной установки по прямой ссылке:

```
cd "$HOME/.local/share/smartcamera/source"

source /opt/ros/jazzy/setup.bash
source .venv/bin/activate
source install/setup.bash

export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
export ROS_DOMAIN_ID=0
```

Если устанавливали из своего каталога, замените путь в `cd`. В `ROS_DOMAIN_ID` укажите значение из `smartcamera configure --show`. Повторно запускать `ros2 launch` в этом терминале не нужно.

В Docker библиотеки ROS находятся внутри контейнера. Для работы с хоста установите ROS 2 Jazzy и соберите пакет `smart_camera_interfaces` на хосте, либо выполняйте команды внутри запущенного контейнера с подключённым окружением. Настройка сети описана в руководстве Docker.

Запуски после перезапуска сервера остаются выключенными, как и в веб-интерфейсе. Модель выбирается для **запуска** (`run_id`), а не глобально для камеры. Одна камера может иметь несколько запусков с разными моделями. Модели предварительно добавьте или обучите через существующий интерфейс; ROS выбирает их по `model_id` из реестра.

5.1.1 Топики и сервис

Узел: `/smart_camera_web/web_bridge`. Имена ниже абсолютные, при необходимости используйте ROS remapping при создании узла.

Имя	Тип	Назначение / QoS
`/smart_camera/control`	`smart_camera_interfaces/srv/Control`	Команда и ответ, штатный service QoS
`/smart_camera/commands`	`smart_camera_interfaces/msg/ControlCommand`	Те же команды, reliable / volatile / depth 10
`/smart_camera/command_results`	`smart_camera_interfaces/msg/ControlReply`	Ответы с `request_id`, reliable / volatile / depth 10
`/smart_camera/status`	`std_msgs/msg/String`	JSON: камеры, модели, запуски, ошибки публикации; 1 Гц, reliable / transient-local / depth 1
`/smart_camera/inference/results`	`smart_camera_interfaces/msg/InferenceResult`	Результаты всех запусков, best-effort / volatile / depth 1
`/smart_camera/inference/r_<run_id>/preview`	`sensor_msgs/msg/CompressedImage`	JPEG с разметкой, при наличии подписчика; best-effort / volatile / depth 1

Изображения камер, `/diagnostics` и сервисы маркеров продолжают работать по описанию камер.

5.1.2 Команды

`Control.srv` принимает поле `command`. Через топик отправляется тот же `ControlCommand` без внешнего поля `command`:

- `request_id`: уникальная строка запроса, обязательна для топика. Для сервисов рекомендуется тоже указывать: повтор с тем же ID возвращает тот же ответ.
- `operation`: одно из значений таблицы ниже.
- `target_id`: ID камеры или запуска; пусто для списков и создания.
- `expected_revision`: текущая ревизия строкой для изменения/удаления. У камеры это хеш конфигурации, у запуска — целое число строкой.
- `payload_json`: строка с JSON-объектом, по умолчанию `"{}"`.

operation	**target_id**	**payload_json**	**Ответ `data_json`**
`cameras.list`	пусто	`{}`	конфигурация с `revision` и `cameras`, включая runtime
`cameras.set`	ID камеры	изменяемые поля CameraConfig, например `{"enabled":false}`	новая конфигурация
`inference.models`	пусто	`{}`	`{"models":[...]}`
`inference.list`	пусто	`{}`	`{"runs":[...]}` с параметрами, состоянием, FPS, ошибками
`inference.get`	ID запуска	`{}`	состояние выбранного запуска
`inference.create`	пусто	`camera_id`, `model_id`, необязательные настройки	созданный запуск с `id`, `revision`
`inference.set`	ID запуска	изменяемые настройки	обновлённый запуск
`inference.delete`	ID запуска	`{}`	`{"id":"...","deleted":true}`

Настройки запуска: `model_id`, `enabled`, `fps` (0 < FPS ≤ 120), `device` (`cpu` / `cuda:0` и т. д.), `parameters`, `prompts`. Начальные значения: выключен, 10 FPS, CPU. Confidence/IoU/трекинг передаются как `parameters: {"conf":0.4,"iou":0.6,"track":true}`; диапазон порогов — [0, 1]. `parameters` и `prompts` **заменяются целиком**, как в веб-API: сначала прочитайте запуск и сохраните необходимые текущие значения. Подсказки и ограничения семейств моделей проверяются существующим InferenceService.

`cameras.set` позволяет включать/выключать камеру и менять её существующие настройки (разрешение, FPS, топики и т. д.). `id` менять нельзя. Изменения оборудования проходят через существующую перезагрузку драйверов с откатом при ошибке. Для внешнего ROS-источника отключение камеры прекращает подписку SmartCamera; сам внешний издатель продолжает работать.

```
# Информация и реестр моделей
ros2 service call /smart_camera/control smart_camera_interfaces/srv/Control \
  '{command: {operation: cameras.list}}'
ros2 service call /smart_camera/control smart_camera_interfaces/srv/Control \
  '{command: {operation: inference.models}}'

# Подставьте camera_id и model_id из ответов выше; сохраните id нового запуска.
ros2 service call /smart_camera/control smart_camera_interfaces/srv/Control \
  '{command: {request_id: create-front-1, operation: inference.create, payload_json:
"{\"camera_id\":\"front\",\"model_id\":\"MODEL_ID\",\"enabled\":true,\"fps\":10,\"parameters\":
{\"conf\":0.4,\"iou\":0.6}}"}}'

# Смена модели. Подставьте текущую ревизию из inference.get/list.
ros2 service call /smart_camera/control smart_camera_interfaces/srv/Control \
  '{command: {request_id: select-model-1, operation: inference.set, target_id: RUN_ID,
expected_revision: "1", payload_json: "{\"model_id\":\"OTHER_MODEL_ID\"}"}}'

# Подписка на результаты и состояние
ros2 topic echo /smart_camera/inference/results --qos-reliability best_effort
ros2 topic echo /smart_camera/status --qos-durability transient_local --once

# До отправки команды из другого терминала подпишитесь на подтверждения.
ros2 topic echo /smart_camera/command_results
ros2 topic pub --once /smart_camera/commands smart_camera_interfaces/msg/ControlCommand \
  '{request_id: stop-front-1, operation: inference.set, target_id: RUN_ID, expected_revision: "2",
payload_json: "{\"enabled\":false}"}'
```

Ответ содержит `success`, `code`, `message` и `data_json`. `revision_conflict` / `configuration_conflict` означает, что нужно перечитать состояние; `revision_required` — не передана ревизия. Ошибки входных данных возвращаются как `invalid_request`; ошибки модели/камеры сохраняют код существующего сервиса. Успешное включение подтверждает применение настроек: загрузка модели происходит при следующем кадре, а её результат отражается в `state` / `error` запуска.

До 32 запросов могут выполняться одновременно; остальные получают `busy`. JSON команды ограничен 64 КиБ, ответа — 4 МиБ. Повторы дедуплицируются в памяти процесса: до 256 завершённых запросов и до 4 МиБ суммарного кеша. После вытеснения или перезапуска ID может быть исполнен снова. Один ID с другими аргументами даёт `request_id_conflict`. Для нового действия, включая повторное чтение, используйте новый ID; повторять прежний ID следует только для получения ответа на тот же запрос.

Сервис ждёт до 30 секунд. `timeout` означает, что операция **может ещё завершиться**; окончательный ответ придёт в `command_results`. Проверьте состояние или повторите запрос с тем же ID. Команды не сохраняются в DDS для будущих подписчиков.

5.1.3 Данные YOLO

`InferenceResult` содержит `run_id`, `model_id`, `camera_id`, `instance_id`, `revision`, `sequence`, размеры кадра и время обработки в миллисекундах. `header.stamp` и `header.frame_id` берутся из входного

изображения; нулевой timestamp сохраняется. Координаты — **пиксели исходного кадра**, начало слева сверху, u вправо, v вниз. Это 2D-данные, не координаты объекта в системе робота.

`objects[]` содержит класс, confidence, `has_box`, `xyxy`, строковый `track_id`, контур маски `polygon`, `keypoints` с confidence и углы `oriented_box` для OBB. Отсутствующий track ID — пустая строка; неизвестный класс/confidence — -1. Маски SAM могут приходить без класса и рамки. Пустой массив объектов — корректный результат без обнаружений. ID треков действуют только внутри запуска/его поколения.

`data_json` содержит `raw` и `outputs` с полными результатами модели, в том числе вероятностями классификации, растровыми масками и нестандартными выходами. Внешние `.npy`-ресурсы преобразуются в массивы: удалённому клиенту не нужны пути или HTTP-запросы к серверу. Лимит JSON — 16 МиБ; при загрузке массивов также применяется консервативный лимит памяти. Слишком большой/истёкший результат пропускается с объяснением в `result_errors` топика состояния и журнале.

Издатель опрашивает последние готовые результаты каждые 20 мс. Очереди хранят свежие данные: при высокой частоте модели, медленной передаче или обработке клиентом часть кадров может быть пропущена. Повторы последнего результата не публикуются. После смены модели/настроек результаты старой ревизии отбрасываются. Preview появляется после первого результата и передаётся только подписчикам, начиная со следующего кадра.

5.1.4 Python

```
python3 examples/ros/control.py models
python3 examples/ros/control.py cameras
python3 examples/ros/control.py create --json
'{"camera_id":"front","model_id":"MODEL_ID","enabled":true}'
python3 examples/ros/control.py get --target RUN_ID
python3 examples/ros/control.py set --target RUN_ID --revision 1 --json '{"enabled":false}'
python3 examples/ros/control.py watch --target RUN_ID
```

Пример клиента Python использует сервис для команд и подписку для результатов. Совместимость QoS проверяется через `ros2 topic info <topic> -v`. Описание политики QoS: ROS 2 Jazzy.

5.1.5 Проверки без оборудования

```
source install/setup.bash
ROS_LOCALHOST_ONLY=1 ROS_DOMAIN_ID=185 colcon test \
  --packages-select smart_camera_web smart_camera_inference smart_camera_detection
smart_camera_utils smart_camera_driver \
  --python-testing pytest --return-code-on-test-failure
colcon test-result --verbose
```

Новые DDS-тесты используют отдельные домены 183, 184 и 187. Сквозной тест публикует синтетическое изображение и запускает тестовый адаптер в отдельном процессе; реальные камеры и веса YOLO не нужны.

5.2 Python-клиент

В репозитории есть готовый клиент `examples/ros/control.py`. Он вызывает сервис управления и умеет подписываться на результаты. Его можно запускать на компьютере SmartCamera или на другом компьютере с ROS 2 Jazzy и собранным пакетом `smart_camera_interfaces`.

5.2.1 Подготовить терминал

Откройте второй терминал, оставив сервер работающим:

```
cd ~/SmartCamera
source /opt/ros/jazzy/setup.bash
source install/setup.bash
python3 examples/ros/control.py --help
```

Если проект установлен в другом каталоге, замените `~/SmartCamera`. На удалённой машине согласуйте `ROS_DOMAIN_ID` с сервером. При разных доменах сервис не будет обнаружен, даже если веб-панель открывается по HTTP.

5.2.2 От списка камер до первого результата

Получите камеры и модели:

```
python3 examples/ros/control.py cameras
python3 examples/ros/control.py models
```

В первом ответе найдите `cameras[].id`, во втором — `models[].id`. Название модели помогает выбрать запись, но в запрос передаётся её ID. Подставьте свои значения:

```
CAMERA_ID=workbench
MODEL_ID=вставьте_id_модели
python3 examples/ros/control.py create --json \
  "{\"camera_id\":\"$CAMERA_ID\",\"model_id\":\"$MODEL_ID\",\"enabled\":true,\"fps\":5,\"device\":
\"cpu\",\"parameters\":{\"conf\":0.4,\"iou\":0.6}}"
```

Сохраните `id` созданного запуска. Ниже `RUN_ID` — этот ID, а не ID модели.

```
RUN_ID=вставьте_id_запуска
python3 examples/ros/control.py get --target "$RUN_ID"
python3 examples/ros/control.py watch --target "$RUN_ID"
```

`watch` печатает кадры по мере обработки: номер, систему координат, обнаруженные классы, confidence, рамки и ID треков. Для завершения просмотра нажмите `^ Ctrl` + `C`. Подписка остановится, а модель продолжит работать.

Для остановки самой модели сначала перечитайте её состояние и возьмите текущую ревизию. Например, если `revision` равна 1:

```
python3 examples/ros/control.py set --target "$RUN_ID" --revision 1 \
  --json '{"enabled":false}'
```

Если настройки уже менялись, число будет другим. Ошибка `revision_conflict` означает, что нужно повторить `get` и использовать новую ревизию. Не подставляйте постоянно одно и то же число в управляющую программу.

5.2.3 Подписчик для своей программы

Этот пример не управляет моделями. Он слушает уже работающие запуски и выводит рамки объектов. Сохраните его как `listen_detections.py` и запустите после `source install/setup.bash`:

```
import rclpy
from rclpy.node import Node
from rclpy.qos import qos_profile_sensor_data
from smart_camera_interfaces.msg import InferenceResult


class Detections(Node):
    def __init__(self):
        super().__init__("my_detections")
        self.subscription = self.create_subscription(
            InferenceResult,
            "/smart_camera/inference/results",
            self.on_frame,
            qos_profile_sensor_data,
        )

    def on_frame(self, message):
        # При нескольких запусках здесь можно отфильтровать message.run_id.
        stamp = message.header.stamp
        print(message.camera_id, message.sequence,
              f"{stamp.sec}.{stamp.nanosec:09d}", message.header.frame_id)
        for obj in message.objects:
            if obj.has_box:
                left, top, right, bottom = obj.xyxy
                print(obj.class_name, round(obj.confidence, 3),
                      (left, top, right, bottom), obj.track_id)


rclpy.init()
node = Detections()
try:
```

```
    rclpy.spin(node)
except KeyboardInterrupt:
    pass
finally:
    node.destroy_node()
    rclpy.shutdown()
```

```
python3 listen_detections.py
```

`qos_profile_sensor_data` совместим с издателем результатов: best-effort, volatile. Если задать подписчику reliable, сообщения от best-effort издателя не придут. Наличие топика в `ros2 topic list` само по себе не подтверждает совместимость QoS.

5.2.4 Полные выходы модели

Массив `objects` удобен для обычной детекции. Для вероятностей классификации, растровых масок и нестандартных выходов прочитайте JSON:

```
import json

payload = json.loads(message.data_json)
raw = payload["raw"]
outputs = payload["outputs"]
probabilities = raw.get("probs", {}).get("data")
```

Доступные поля зависят от задачи модели. Проверяйте их наличие; классификатор может вернуть вероятности и пустой `objects`. Сервис уже заменяет ссылки на временные NumPy-файлы значениями массивов, поэтому клиенту не нужен доступ к каталогу записей сервера.

Систему координат и ограничения размеров сообщений смотрите в описании ROS-интерфейса. Перевод рамки в 3D требует калибровки и глубины; сами четыре числа `xyxy` являются пикселями.

5.3 HTTP API

Веб-панель использует HTTP API того же сервера. Он подходит для скриптов, которым не нужен ROS: получения списка камер, изменения настроек, работы с моделями и архивом.

Откройте `http://127.0.0.1:8000/docs` на работающем сервере. Это интерактивный справочник OpenAPI: у каждого запроса указаны параметры, схема тела и ответы. Страница `/docs` сервера — справочник API, а этот сайт MkDocs — руководство пользователя.

5.3.1 Проверить доступность

```
BASE=http://127.0.0.1:8000
curl --fail "$BASE/healthz"
curl --fail "$BASE/api/v1/cameras" | python3 -m json.tool
```

На другом компьютере замените адрес. Сервер по умолчанию принимает соединения только с локальной машины. Порядок запуска для локальной сети описан в первом запуске.

5.3.2 Камеры

Запрос	Назначение
`GET /api/v1/cameras`	Конфигурация, ревизия и состояние камер
`GET /api/v1/cameras/devices`	Обнаруженные V4L2-устройства и режимы
`GET /api/v1/ros/topics`	Список ROS-топиков и типов
`POST /api/v1/cameras`	Добавление камеры
`PUT /api/v1/cameras/{id}`	Полная новая конфигурация выбранной камеры
`DELETE /api/v1/cameras/{id}`	Удаление
`GET /api/v1/cameras/{id}/stream.mjpeg`	Предпросмотр

Изменение конфигурации требует `If-Match` из предыдущего чтения. Иначе два клиента могли бы незаметно перезаписать настройки друг друга. Здесь ревизия — хеш всей конфигурации камер, а не счётчик отдельной камеры.

Следующий пример выключает только одну камеру, сохраняя остальные её поля. Для него нужен пакет `requests` в Python-окружении клиента.

```
import requests

base = "http://127.0.0.1:8000"
camera_id = "workbench"
```

```
response = requests.get(f"{base}/api/v1/cameras", timeout=10)
response.raise_for_status()
camera = next(c for c in response.json()["cameras"] if c["id"] == camera_id)
camera.pop("runtime", None)  # runtime относится к ответу, а не к конфигурации
camera["enabled"] = False
updated = requests.put(
    f"{base}/api/v1/cameras/{camera_id}",
    headers={"If-Match": response.headers["ETag"]},
    json=camera,
    timeout=90,
)
updated.raise_for_status()
print(updated.json())
```

Отключение V4L2/RealSense применяется к драйверу. У внешнего ROS-источника SmartCamera прекращает подписку, а издатель другой программы продолжает работать. При неудачном применении конфигурация восстанавливается; прочитайте ответ и повторно получите список камер, прежде чем отправлять следующую правку.

5.3.3 Модели и запуски

Запрос	Назначение
`GET /api/v1/models`	Библиотека Ultralytics
`GET /api/v1/models/capabilities`	Интерпретатор ML, пакеты, доступность вычислителей
`GET /api/v1/inference`	Запуски моделей
`POST /api/v1/inference`	Подключение модели к камере
`PATCH /api/v1/inference/{id}`	Изменение запуска с `If-Match`
`DELETE /api/v1/inference/{id}`	Удаление запуска с `If-Match`
`GET /api/v1/inference/{id}/result`	Последний доступный результат
`GET /api/v1/jobs`	Фоновые задания
`POST /api/v1/jobs`	Загрузка, обучение, валидация или экспорт

В отличие от камер, ревизия запуска — целое число. Пример создания выключенного запуска, проверки ответа и включения:

```
import requests

base = "http://127.0.0.1:8000/api/v1"
response = requests.post(f"{base}/inference", json={
    "camera_id": "workbench",
    "model_id": "ЗАМЕНИТЕ_ID_МОДЕЛИ",
    "enabled": False,
```

```
    "fps": 5,
    "device": "cpu",
    "parameters": {"conf": 0.4, "iou": 0.6},
}, timeout=15)
response.raise_for_status()
run = response.json()
response = requests.patch(
    f"{base}/inference/{run['id']}",
    headers={"If-Match": str(run["revision"])},
    json={"enabled": True},
    timeout=15,
)
response.raise_for_status()
print(response.json())
```

Получение успешного ответа означает, что настройки приняты. Загрузка весов и обработка идут в отдельном процессе; последующую ошибку модели нужно читать в состоянии запуска. Списки `parameters` и `prompts` заменяются целиком, поэтому для изменения одного вложенного значения сначала сохраните остальные настройки.

Произвольная загрузка файла весов через `POST /models` или `/models/upload` в текущей версии отключена и возвращает 410. Для добавления стандартной модели используется задание загрузки; работа через интерфейс описана в библиотеке моделей.

5.3.4 Запись и материалы

Фото создаётся запросом `POST /api/v1/recordings/photos`, видео — `POST /api/v1/recordings/videos`, bag — `POST /api/v1/recordings/bags`. Параметры отличаются: сначала выберите нужную схему в OpenAPI. Для остановки активной записи используется `POST /api/v1/recordings/active/stop`.

`GET /api/v1/recordings` возвращает материалы; отдельная запись доступна по `GET /api/v1/recordings/{id}`. ZIP скачивается через `GET /api/v1/recordings/{id}/download.zip`. Идентификаторы файлов берите из manifest, не формируйте пути к файлам вручную.

5.3.5 Как обрабатывать ошибки

Код HTTP	Что проверить
404	Существуют ли камера, запуск, материал или ещё не истёк ли временный ресурс
409	Ревизия устарела либо ресурс занят другой операцией
410	Эта операция больше не поддерживается текущим API
422	Типы, диапазоны и обязательные поля запроса
428	Не передан требуемый `If-Match`
503	Доступен ли ROS-узел, кадр или запись конфигурации

Не повторяйте изменяющий запрос бесконечно после сетевого тайм-аута: он мог успеть выполниться. Перечитайте состояние. Для ROS-команд есть отдельный механизм повторов по `request_id`, описанный в контракте команд.

6. Справочник

6.1 Конфигурация

SmartCamera хранит настройки камер в YAML, настройки веб-приложения — в отдельном JSON, а записи, калибровочные сессии и данные моделей — в каталоге материалов. Эти файлы решают разные задачи: изменение списка камер не переносит архив, а смена папки материалов не меняет ROS-топики.

6.1.1 Где находится рабочий файл

При запуске через `smartcamera` пути задаёт CLI. По умолчанию:

Что хранится	Путь
Настройки запуска CLI	`~/.config/smartcamera/settings.json`
Камеры	`~/.local/share/smartcamera/data/config/cameras.yaml`
Настройки веб-интерфейса	`~/.local/share/smartcamera/data/config/web.json`
Записи и данные моделей	`~/.local/share/smartcamera/data/recordings/`

Выбранные пути и режим запуска можно посмотреть командой:

```
smartcamera configure --show
```

Если при настройке задан другой каталог данных, YAML камер и настройки веб-интерфейса находятся в его подкаталоге `config/`. CLI передаёт эти пути приложению и создаёт пустую конфигурацию при первой настройке. В Docker используется тот же каталог данных на хосте.

При самостоятельном запуске через `ros2 launch` передавайте свой YAML через аргумент `config_file`. Без него launch использует файл из пакета `smart_camera_config`, который может содержать конфигурацию разработчика. Не запускайте второй экземпляр через `ros2 launch` одновременно с CLI.

Фактический путь доступен в настройках веб-панели и поле `camera_config_path` ответа `GET /api/v1/settings`. Для сохранения нужен доступ к файлу и его родительскому каталогу: запись выполняется через временный файл и атомарную замену. Скопированный путь к чужой калибровке или серийный номер из примера не становится действительным на другом компьютере.

6.1.2 Минимальный YAML

```yaml
schema_version: 1
cameras: []
```

Пропущенные `manager` и `web` получают значения по умолчанию. Модели строгие: неизвестные поля отклоняются, поэтому опечатка не превращается в незаметно проигнорированную настройку.

Раздел и поле	По умолчанию	Допустимое значение / смысл
`schema_version`	`1`	Только версия 1
`manager.open_timeout_ms`	`5000`	100–60000 мс; ожидание открытия камеры
`manager.reconnect_delay_ms`	`2000`	100–60000 мс; задержка переподключения
`manager.failure_threshold`	`5`	1–1000; порог последовательных ошибок
`web.grid_preview_fps`	`12`	1–30; частота предпросмотра сетки
`web.raw_preview_jpeg_quality`	`80`	1–100; JPEG для просмотра raw-потока

Частота браузерного предпросмотра отличается от частоты захвата камеры и частоты обработки нейросетью. Уменьшение `grid_preview_fps` не меняет аппаратный профиль камеры.

6.1.3 Пример USB-камеры

Замените `device` реальным путём из `/dev/v4l/by-id/`, а разрешение, FPS и `pixel_format` — режимом, который показывает `v4l2-ctl` для этой камеры:

```yaml
schema_version: 1
manager:
  open_timeout_ms: 5000
  reconnect_delay_ms: 2000
  failure_threshold: 5
web:
  grid_preview_fps: 12
  raw_preview_jpeg_quality: 80
cameras:
  - id: front
    display_name: Передняя камера
    enabled: true
    interface: v4l2
    device: /dev/v4l/by-id/usb-YOUR_CAMERA-video-index0
    topic: /smart_camera/front/image_raw
    frame_id: front_camera_optical_frame
    resolution: {width: 640, height: 480}
    fps: 30
    pixel_format: MJPG
    transport: compressed
    jpeg_quality: 85
    output_encoding: bgr8
    controls:
      auto_exposure: null
      exposure: null
      gain: null
```

```
      brightness: null
    calibration_file: null
```

`null` в `controls` означает «не менять значение драйвера». Это отличается от выключения автоматической экспозиции: для него задаётся `auto_exposure: false`. Набор доступных controls зависит от устройства. Значения экспозиции и усиления выбирайте по его диапазонам, а не переносите с другой модели камеры.

6.1.4 Общие поля камеры

Поле	Назначение
`id`	Уникальный идентификатор длиной 1–64; начинается с маленькой латинской буквы, далее буквы, цифры, `_` и `-`
`display_name`	Название в интерфейсе, 1–128 символов
`enabled`	Включена ли камера; по умолчанию `true`
`interface`	`v4l2`, `realsense` или `ros`; по умолчанию `v4l2`
`device`	Обязательная непустая строка устройства или обозначения источника
`topic`	Абсолютное ROS-имя без завершающего `/` и `//`
`frame_id`	Непустое имя системы координат изображения
`resolution`	`width` и `height`, каждое от 1 до 16384
`fps`	Частота больше 0 и не выше 240; оборудование должно поддерживать выбранное значение
`pixel_format`	`auto`, `MJPG` или `YUYV`; по умолчанию `auto`
`transport`	`raw` или `compressed`; по умолчанию `raw`
`jpeg_quality`	1–100, по умолчанию 85
`output_encoding`	`bgr8`, `rgb8` или `mono8`; по умолчанию `bgr8`
`calibration_file`	Путь к ROS calibration YAML либо `null`

Для аппаратной камеры с `compressed` фактический топик изображения — `<topic>/compressed`. Для внешнего источника `interface: ros` поле `topic` всегда содержит **точное** имя уже существующего топика, включая суффикс, если он есть. SmartCamera подписывается на него и не управляет внешним издателем. Пример такого источника:

```
schema_version: 1
cameras:
  - id: external
    display_name: Внешний ROS-поток
    interface: ros
    device: ros
```

```
      topic: /external/image/compressed
      frame_id: external_optical_frame
      resolution: {width: 640, height: 480}
      fps: 30
      transport: compressed
```

Конфигурация проверяет уникальность ID, топиков, V4L2-устройств и серийных номеров RealSense. Проверяются и эффективные топики: например, изображение одной камеры не должно пересекаться с `camera_info` другой. Отключённая запись остаётся частью конфигурации и тоже участвует в проверке уникальности.

6.1.5 RealSense и калибровка

Для `interface: realsense` обязателен `serial`. Поля `color` и `depth` описывают ширину, высоту и FPS потоков; у depth также есть `topic` и `frame_id`. Если `color` не задан, он получается из `resolution` и `fps`; пропущенный depth создаётся с тем же профилем. Реальную поддержку сочетания потоков проверяйте через discovery, особенно для L515.

`pointcloud.enabled` по умолчанию `false`. Для облака можно задать `topic`, `frame_id` и `max_points` (по умолчанию 150000, диапазон 1–2000000). `depth_visualization` задаёт только отображение: `min_m` по умолчанию 0.2, `max_m` 4.0, `colormap` — `turbo`, `viridis`, `magma` или `gray`. Максимальная дистанция должна быть больше минимальной. Цветной поток RealSense допускает `bgr8` и `rgb8`.

`calibration_file` принимает абсолютный путь или адрес вида `package://smart_camera_config/config/calibration/front.yaml`. Для обычной камеры без калибровки фиктивный `CameraInfo` не создаётся; RealSense предоставляет заводские параметры. При применении результата монокалибровки приложение сохраняет первоначальную настройку в `calibration_baseline`. Не удаляйте исходный calibration YAML, если планируете вернуться к нему. Работа с профилями описана в руководстве по камерам.

6.1.6 Применение и конфликты изменений

Веб-панель сохраняет конфигурацию с текущей ревизией и применяет изменения через менеджеры. При неудачном открытии выполняется откат. HTTP-клиент передаёт `If-Match`, ROS-клиент — `expected_revision`; устаревшая ревизия требует повторного чтения. Это защищает от перезаписи изменений другого клиента.

Если YAML редактировался вручную, используйте действие перечитывания конфигурации в настройках приложения. Само сохранение файла редактором не равнозначно применению через API. Для интеграции см. HTTP API и ROS 2 управление.

6.1.7 Launch, настройки веб-сервера и материалы

Аргумент launch	По умолчанию
`config_file`	`config/cameras.yaml` установленного пакета
`web_host`	`127.0.0.1`
`web_port`	`8000`
`recordings_dir`	`~/smart_camera_recordings`
`photo_sync_tolerance_ms`	`30` (веб-сервер принимает 0–30)
`realsense_sdk_prefix`	`~/.local/librealsense-l515-2.50.0`

JSON настроек находится в `~/.config/smart_camera_web/settings.json`; путь можно изменить переменной `SMART_CAMERA_WEB_SETTINGS_FILE`. Он содержит `recordings_directory` и `dashboard_refresh_seconds` (1–300, по умолчанию 5). При наличии сохранённого JSON его каталог материалов используется вместо стартового значения `recordings_dir` в обычном запуске сервера.

В каталоге материалов находятся записи и подкаталоги `calibrations`, `handeye`, `inference`. Смена папки в настройках не переносит существующие файлы; службы калибровки и inference выбирают свои корни при старте. После изменения папки перезапустите приложение, чтобы все службы использовали новое расположение. Для резервной копии сохраняйте YAML камер, связанные calibration-файлы, JSON настроек и весь каталог материалов.

CLI автоматически задаёт `SMART_CAMERA_INFERENCE_PYTHON`: `.venv-inference/bin/python` внутри исходников для локального режима и `/opt/inference/bin/python` для Docker. При самостоятельном запуске порядок выбора: `SMART_CAMERA_INFERENCE_PYTHON`, затем существующий `~/.venvs/smart-camera-inference/bin/python`, затем Python веб-сервера. Его окружение не заменяет YAML камер; подробности — в разделе моделей.

6.2 Диагностика проблем

Начинайте с границы, на которой пропадают данные: устройство → ROS-топик → веб-просмотр → запись или модель. Если камера не публикует изображение, настройка нейросети не восстановит поток. Если изображение есть в ROS, но отсутствует в браузере, сначала проверяйте подписку и конфигурацию веб-сервера.

При запуске через CLI начните с `smartcamera doctor` и `smartcamera logs --follow`. При запуске без `--detach` оставляйте терминал открытым. В нём видны ошибки открытия камеры, применения YAML, запуска процессов и обработки запросов. Для обращения с проблемой сохраните сообщение целиком, команду запуска, тип камеры, выбранный режим и версию окружения. Не прикладывайте ключи S3 и другие credentials.

6.2.1 Не найден ROS-пакет или Python-модуль

Для локальной установки по прямой ссылке восстановите окружение в новом терминале:

```
cd "$HOME/.local/share/smartcamera/source"
source /opt/ros/jazzy/setup.bash
source .venv/bin/activate
source install/setup.bash
ros2 pkg prefix smart_camera_web
python - <<'PYTHON'
import sys
import rclpy
import cv2
import pydantic

print(sys.executable)
print(cv2.__version__, pydantic.__version__)
PYTHON
```

Если `ros2 pkg prefix` не находит пакет, workspace не собран либо подключён не тот `install/setup.bash`. Если Python не видит ROS, проверьте, что `.venv` создана через `/usr/bin/python3 -m venv --system-site-packages`, а не отдельным Python без системных библиотек. Ошибки импорта Pydantic обычно означают, что приложение использует версию 1 вместо 2.

Активация окружения не меняет Python в уже созданном ROS-скрипте. Проверьте первую строку установленного веб-сервера:

```
head -n 1 "$(ros2 pkg prefix smart_camera_web)/lib/smart_camera_web/web_server"
```

Если устанавливали из своего каталога, подставьте его путь. Для Docker повторите `smartcamera docker-setup --yes` после остановки контейнера; окружение хоста не изменяет образ.

Для восстановления зависимостей локальной установки остановите приложение и повторите подготовку выбранного профиля:

```
smartcamera stop

smartcamera local-setup --yes --profile cpu

smartcamera run
```

Если используете NVIDIA, укажите `--profile cuda`. Для обычной пересборки исходников достаточно `smartcamera rebuild`. Не устанавливайте `opencv-python` поверх системного OpenCV окружения ROS. Ошибка `externally-managed-environment` при `pip --user` не возникает в штатной установке CLI: он использует отдельные окружения Python.

6.2.2 Не собирается веб-интерфейс

Проверьте Node.js и локальные зависимости:

```
cd "$HOME/.local/share/smartcamera/source"
export PATH="$PWD/.local/node/bin:$PATH"

node --version
npm --version

cd src/smart_camera_web/frontend
npm ci
npm run build
```

Vite 8 требует Node `^20.19.0 || >=22.12.0`; Node 18 не подходит. Ошибка об отсутствии `frontend/node_modules` означает, что bootstrap не выполнен. `colcon` запускает сборку интерфейса, но npm-зависимости не скачивает. После успешной отдельной сборки вернитесь в корень workspace и повторите сборку ROS-пакета.

Если браузер показывает старую версию страницы, убедитесь, что запущен сервер из нужного workspace, пересоберите `smart_camera_web` и перезапустите сервер. Само изменение React-файлов не обновляет установленный production frontend. Сообщения об отсутствующих frontend-файлах проверяйте отдельно от состояния камер.

6.2.3 Веб-страница не открывается

Сначала выполните на компьютере сервера:

```
curl --fail --verbose http://127.0.0.1:8000/api/v1/cameras
ss -ltnp | rg ':8000'
```

`Connection refused` означает, что по этому адресу и порту нет слушающего сервера. Посмотрите, не завершился ли `web_server` при старте из-за YAML, Python-зависимостей или занятого порта. HTTP-ответ с ошибкой — другая ситуация: сервер работает, и нужно читать код и текст ответа.

По умолчанию `web_host` равен `127.0.0.1`: доступ возможен только с этого компьютера. Для доверенной локальной сети добавьте `web_host:=0.0.0.0`; на другом компьютере открывайте IP сервера, а не `127.0.0.1`. Если локальная проверка проходит, проверьте сетевой маршрут и firewall. Аргумент `web_port:=8001` позволяет использовать другой свободный порт.

6.2.4 USB-камера не открывается или пропадает

```
v4l2-ctl --list-devices
ls -l /dev/v4l/by-id/
v4l2-ctl --device=/dev/video0 --list-formats-ext
id
ls -l /dev/video0
```

Замените `/dev/video0` узлом вашей камеры. Сверьте путь с YAML, права пользователя и поддерживаемые режимы. Если устройство доступно группе `video`, а пользователь в неё не входит, добавьте его через `sudo usermod -aG video "$USER"`, затем полностью завершите пользовательский сеанс и войдите снова. Уже запущенные процессы новые группы не получают.

Закройте другую программу, которая захватывает ту же камеру. Проверьте USB-кабель, питание и порт. Для нескольких камер уменьшите разрешение/FPS либо выберите поддерживаемый MJPG; одинаковые высокие raw-потоки могут исчерпать пропускную способность общей USB-шины. Сначала добейтесь устойчивой работы одной камеры.

Если изменение настроек отклоняется, это может быть штатный откат: новый профиль не дал первый кадр. Не увеличивайте таймаут без проверки режима и доступности устройства. При успешном переподключении драйвер продолжает работу; постоянный цикл ошибок виден в `/diagnostics` и журнале.

6.2.5 ROS-топик есть, изображения нет

```
ros2 topic list -t
ros2 topic info /smart_camera/front/image_raw/compressed -v
ros2 topic hz /smart_camera/front/image_raw/compressed
ros2 topic echo /diagnostics --once
```

Подставьте фактический топик. Для аппаратного compressed-потока к `topic` добавляется `/compressed`; у внешнего `interface: ros` имя уже должно быть полным. Проверьте тип: `Image` и `CompressedImage` требуют разных подписок. Для изображений используется best-effort/volatile/depth 1; reliable-подписчик может оказаться несовместимым.

На разных компьютерах сравните `ROS_DOMAIN_ID`, DDS-настройки и сетевую доступность. `ROS_LOCALHOST_ONLY=1` ограничивает обмен локальным компьютером. Сам факт присутствия имени топика ещё не подтверждает поступление кадров: смотрите частоту и время последних сообщений.

6.2.6 RealSense видна в USB, но не в приложении

Для L515 проверьте, с каким SDK собран драйвер и какую библиотеку получает процесс. Проект использует отдельный librealsense 2.50.0; установка современной системной библиотеки не заменяет этот шаг. При самостоятельном запуске вне общего launch нужно явно выбрать библиотеку:

```
L515_PREFIX="$HOME/.local/librealsense-l515-2.50.0"
export LD_LIBRARY_PATH="$L515_PREFIX/lib${LD_LIBRARY_PATH:+:$LD_LIBRARY_PATH}"
```

Для общего launch есть `realsense_sdk_prefix`; переменная `CMAKE_PREFIX_PATH` выбирает SDK при сборке, а не только при запуске. После смены SDK пересоберите с чистым CMake-кэшем. Сверьте serial и RGB-D профиль с discovery. Проверенный профиль L515 — RGB и depth `640×480@30`; это не обещание поддержки любого сочетания размеров. Настройки D400 Advanced Mode и аппаратной синхронизации не следует переносить на L500. См. установку и камеры.

6.2.7 YAML не сохраняется или возникает конфликт

`configuration_conflict` или HTTP 409 означает, что другой клиент уже изменил конфигурацию. Перечитайте её и повторите изменение с новой ревизией. HTTP 428 требует `If-Match`. Не подставляйте случайную ревизию и не повторяйте устаревший запрос бесконечно.

При `config_not_writable` проверьте фактический `camera_config_path` в `/api/v1/settings`, права файла и каталога. При symlink-install запись может идти в исходный YAML. `invalid_camera_config` часто связан с неизвестным полем, повтором ID/устройства/топика или недоступным calibration-файлом. Формат описан в справочнике конфигурации.

6.2.8 Не записывается материал или исчез архив

Проверьте свободное место (`df -h`), доступность каталога материалов и наличие FFmpeg (`ffmpeg -version`). Одновременно выполняется одна операция записи: фото, серия, MP4 или bag. Для группового фото все выбранные камеры должны предоставить свежие кадры с допустимым timestamp skew. Увеличение количества выбранных камер не создаёт аппаратную синхронизацию.

Если после смены каталога архив пуст, проверьте сохранённый `recordings_directory`: старые файлы автоматически не переносятся. Также сохранённый JSON настроек может переопределить стартовый каталог launch. После изменения каталога перезапустите приложение, чтобы калибровки, HandEye и inference использовали согласованные корни. Прерванная запись после восстановления может иметь статус `interrupted`; это отличается от успешного завершения.

6.2.9 Модель включена, результата нет

Проверьте обычный поток камеры, состояние запуска и поле ошибки. Веса загружаются при следующем кадре, поэтому ответ на команду включения подтверждает настройки, а не готовность модели. После рестарта сервера запуск нужно включить снова.

В разделе моделей используйте проверку среды и устройств: она показывает именно ML Python. Установка Ultralytics в `.venv` веб-сервера не помогает, если `SMART_CAMERA_INFERENCE_PYTHON` указывает на другое окружение. PyTorch с суффиксом `+cpu` не использует CUDA. Фоновые задания могут ждать освобождения устройства от live-запусков. Подробности — в руководстве моделей.

Если результат есть в браузере, но отсутствует в ROS, проверьте best-effort QoS подписчика и `result_errors` в `/smart_camera/status`. Слишком большой JSON или истёкший ресурс пропускается с объяснением. `timeout` команды ROS не доказывает отмену операции: проверьте состояние и повторяйте запрос с тем же `request_id`, как описано в ROS 2 API.