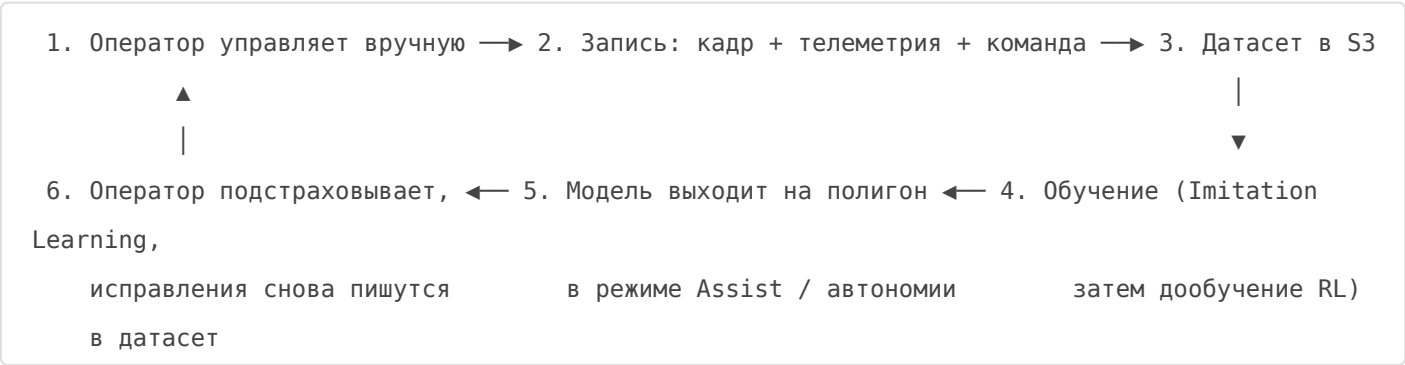


???? ?????? ? ??????????

????????

Каждая сессия ручного управления на платформе — это готовые обучающие данные: платформа синхронно записывает, **что видел робот** (видео), **в каком он был состоянии** (телеметрия) и **что сделал оператор** (команды). На этих данных обучается следующая версия модели.

???? ?????????????? ????????



Особенно ценны **вмешательства оператора** в работу модели: момент, когда человек перехватил управление, показывает, где модель ошибается. Такие фрагменты стоит брать в датасет с повышенным весом (подход DAgger).

???????? ??????????

Поле	Источник	Пример
t	Время записи	1790155251.120
robot	Идентификатор робота	tank-01
frame	Ссылка на кадр в S3	s3://artex-video/tank-01/s42/rgb/1790155251120.jpg
telemetry	Актуальный вектор телеметрии	{"battery": 11.8, "speed": 0.4}
command	Последняя команда	forward
author	Автор команды	operator-7 или model:demo-rules@1.0

?????????? ??????????

1. **Синхронизация.** Каждому кадру сопоставьте последнюю команду, отданную до момента кадра (с учётом задержки — обычно сдвиг на 100–200 мс вперёд даёт лучшие результаты).
2. **Фильтрация.** Удалите участки, где робот стоял без команд, и участки с потерей связи.
3. **Балансировка.** Команды `forward` обычно преобладают — уравновесьте классы (прореживание или веса).
4. **Разбиение.** Делите на обучающую и проверочную выборки **по сессиям**, а не по кадрам, иначе соседние почти одинаковые кадры попадут в обе выборки.

```
import pandas as pd

log = pd.read_parquet("session_42.parquet")          # t, frame, telemetry, command, author
log = log.sort_values("t")
log["command"] = log["command"].ffill()              # команда действует до следующей
log = log[log["command"].notna()]
log = log[~((log["command"] == "stop") & (log["command"].shift() == "stop"))] # длинные
стоянки
print(log["command"].value_counts())
```

Imitation Learning (Behavior Cloning)

Модель учится предсказывать команду оператора по кадру (и телеметрии). Минимальный пример на PyTorch:

```
import torch, torch.nn as nn
from torchvision import models

COMMANDS = ["forward", "back", "left", "right", "stop"]

class Policy(nn.Module):
    def __init__(self):
        super().__init__()
        self.backbone = models.mobilenet_v3_small(weights="DEFAULT")
        self.backbone.classifier[-1] = nn.Linear(1024, len(COMMANDS))

    def forward(self, x):
        # x: [B, 3, 224, 224]
```

```

        return self.backbone(x)

policy = Policy()
opt = torch.optim.AdamW(policy.parameters(), lr=3e-4)
loss_fn = nn.CrossEntropyLoss()

for images, labels in train_loader:      # DataLoader по подготовленному датасету
    opt.zero_grad()
    loss = loss_fn(policy(images), labels)
    loss.backward()
    opt.step()

```

Обученная модель подключается к роботу так же, как демо-модель на странице [Разработка модели ИИ для платформы](#): метод `step()` возвращает `COMMANDS[argmax]`.

???????? ? ?????????????? (RL)

Behavior Cloning ограничен качеством операторов. Чтобы модель превзошла человека, её дообучают методами RL — как правило, на [виртуальном полигоне](#), где можно проводить тысячи эпизодов без риска для техники:

- **Наблюдение** — кадр (или признаки из YOLO) + телеметрия;
- **Действие** — команда из словаря;
- **Награда** — задаётся задачей: продвижение по трассе, выполнение задания, победа в поединке; штрафы за столкновения и выезд за границы;
- **Инициализация** — весами модели, обученной Imitation Learning, что сильно ускоряет обучение.

???????????????? ????????

Храните каждую версию модели в S3 вместе с описанием: на каких сессиях обучена, метрики на проверочной выборке, результаты на полигоне (в том числе ELO в соревнованиях). Автор команд модели должен содержать имя и версию (`model:<name>@<version>`) — так в записях видно, какая версия что делала.

Revision #3

Created 2026-09-23 09:32:31 UTC by Маслов Алексей

Updated 2026-09-23 09:35:12 UTC by Маслов Алексей