Методичка: Система стабилизации масок видеосегментации

📋 Оглавление

Обзор системы

Что делает система?

Система **Mask Stabilization** решает проблему "мерцания" (flickering) при покадровой сегментации видео. Когда нейросеть обрабатывает каждый кадр независимо, маски объектов могут немного отличаться между соседними кадрами, создавая эффект мерцания.

Как это работает?

1. **Загрузка видео** → Видео разбивается на отдельные кадры
2. **Сегментация** → Нейросеть DeepLabv3 находит объекты на каждом кадре
3. **Стабилизация** → Временное сглаживание убирает мерцание между кадрами
4. **Метрики** → Система измеряет улучшение стабильности (IoU)

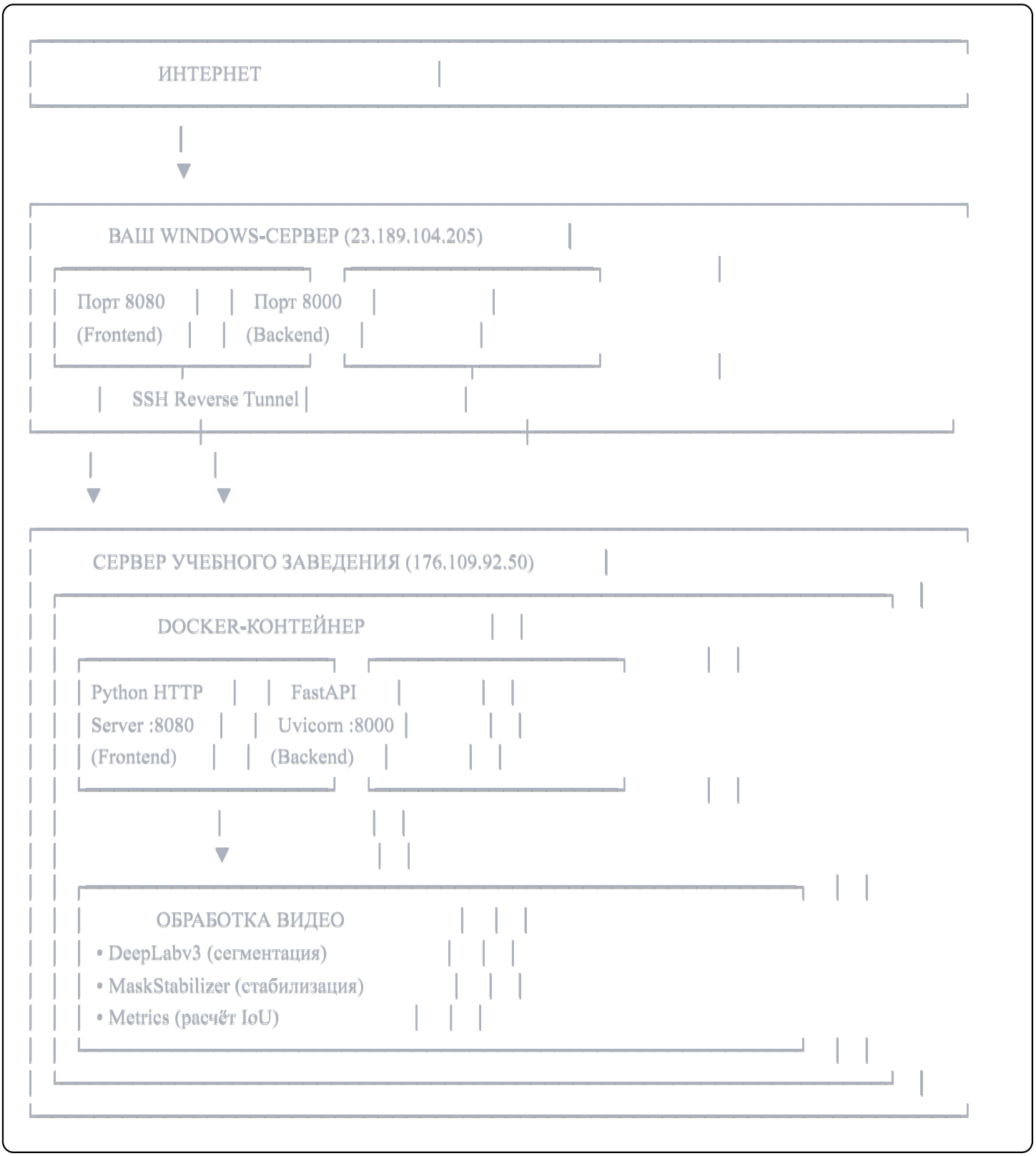
Ключевые метрики

- **IoU (Intersection over Union) —** мера перекрытия масок между соседними кадрами
- **Instability Score —** 1 - IoU (чем выше, тем больше мерцание)

- **Improvement** — процент улучшения после стабилизации

Архитектура

Общая схема



Почему такая схема?

Проблема: Docker-контейнер на сервере учебного заведения не имеет прямого доступа к открытию портов (файрвол управляется снаружи контейнера).

Решение: Обратный SSH-туннель через ваш Windows-сервер, у которого есть публичный IP и открытые порты.

Компоненты системы

Backend (FastAPI + Uvicorn)

Расположение: `/tf/darachubarova/defliker/src/main.py`

Эндпоинты API:

Метод	Путь	Описание
GET	`/`	Информация об API
POST	`/api/upload`	Загрузка видео
POST	`/api/segment`	Запуск сегментации
POST	`/api/stabilize`	Запуск стабилизации
GET	`/api/status/{job_id}`	Статус задания
GET	`/api/results/{job_id}`	Результаты
GET	`/api/metrics/{job_id}`	Метрики
GET	`/api/frames/{job_id}/{type}/{num}`	Получить кадр
GET	`/api/classes`	Доступные классы
DELETE	`/api/job/{job_id}`	Удалить задание

Frontend (HTML + JavaScript)

Расположение: `/tf/darachubarova/defliker/frontend/index.html`

Функции:

- Drag & Drop загрузка видео
- Выбор класса объекта (Person, Car, и т.д.)
- Выбор метода стабилизации
- Отображение прогресса
- Просмотр результатов покадрово
- Отображение метрик

Модули обработки

Сегментация (`src/segmentation.py`):

- Модель: DeepLabv3 ResNet-101
- Классы: person, car, bus, truck, dog, cat, horse, и др.

Стабилизация (`src/stabilization.py`):

- Moving Average — скользящее среднее
- Median Filter — медианный фильтр
- Exponential Smoothing — экспоненциальное сглаживание

Метрики (`src/metrics.py`):

- IoU (Intersection over Union)
- Dice coefficient
- Temporal consistency

Сетевая инфраструктура

Что такое обратный SSH-туннель?

Обычно SSH работает так: вы подключаетесь К серверу.

Обратный туннель работает наоборот: сервер "пробрасывает" свой порт на вашу машину.

```bash
```

```
ssh -R 0.0.0.0:8080:localhost:8080 user@your-server
```

Эта команда означает:

- `-R` — создать обратный туннель
- `0.0.0.0:8080` — слушать на всех интерфейсах вашего сервера на порту 8080
- `localhost:8080` — перенаправлять на локальный порт 8080 (в контейнере)
- `user@your-server` — подключиться к вашему серверу

SSH-ключи для автоматизации

Чтобы не вводить пароль каждый раз, используются SSH-ключи.

Принцип:

1. Генерируется пара ключей: приватный (секретный) и публичный
2. Публичный ключ добавляется на сервер в `~/.ssh/authorized_keys`
3. При подключении система автоматически проверяет ключи

Установка и настройка

Шаг 1: Настройка Windows-сервера

1.1 Включите SSH-сервер

```powershell
# PowerShell от администратора
Start-Service sshd
Set-Service -Name sshd -StartupType Automatic
```

1.2 Настройте файрвол

```powershell
```

```
New-NetFirewallRule -Name "SSH" -DisplayName "SSH Server" -Protocol TCP -LocalPort 22 -Action Allow -Direction Inb
New-NetFirewallRule -Name "WebApp8080" -DisplayName "Web Frontend" -Protocol TCP -LocalPort 8080 -Action Allow
New-NetFirewallRule -Name "WebApp8000" -DisplayName "Web Backend" -Protocol TCP -LocalPort 8000 -Action Allow
```

1.3 Включите GatewayPorts

Отредактируйте `C:\ProgramData\ssh\sshd_config`:

```
GatewayPorts yes
```

Перезапустите SSH:

```powershell
Restart-Service sshd
```

Шаг 2: Настройка SSH-ключей (для автоматизации)

2.1 Создайте ключи в Docker-контейнере

```bash
ssh-keygen -t ed25519 -f ~/.ssh/id_tunnel -N ""
cat ~/.ssh/id_tunnel.pub
```

2.2 Добавьте публичный ключ на Windows

```powershell
# Создайте файл (PowerShell)
New-Item -Path "$env:USERPROFILE\.ssh" -ItemType Directory -Force
Add-Content -Path "$env:USERPROFILE\.ssh\authorized_keys" -Value "ВСТАВЬТЕ_ПУБЛИЧНЫЙ_КЛЮЧ"
```

2.3 Для администраторов Windows (важно!)

```powershell
```

```
# Ключи администраторов хранятся в другом месте
Add-Content -Path "C:\ProgramData\ssh\administrators_authorized_keys" -Value "ВСТАВЬТЕ_ПУБЛИЧНЫЙ_КЛЮЧ"

# Установите правильные права
icacls "C:\ProgramData\ssh\administrators_authorized_keys" /inheritance:r /grant "SYSTEM:(F)" /grant "Administrators:(F)"
```

Шаг 3: Структура проекта

```
/tf/darachubarova/defliker/
├── src/
│   ├── __init__.py
│   ├── main.py          # FastAPI сервер
│   ├── segmentation.py  # DeepLabv3
│   ├── stabilization.py # Методы стабилизации
│   ├── metrics.py       # Расчёт метрик
│   └── utils.py         # Утилиты
├── frontend/
│   └── index.html       # Веб-интерфейс
├── results/
│   ├── uploads/         # Загруженные видео
│   └── outputs/         # Результаты обработки
├── logs/                # Логи
├── start_system.sh      # Скрипт запуска
└── requirements.txt     # Зависимости Python
```

Запуск системы

Автоматический запуск (рекомендуется)

bash

```
cd /tf/darachubarova/defliker
./start_system.sh
```

Ручной запуск

Терминал 1: Backend

bash

```
cd /tf/darachubarova/defliker
uvicorn src.main:app --host 0.0.0.0 --port 8000
```

Терминал 2: Frontend

```bash
cd /tf/darachubarova/defliker/frontend
python3 -m http.server 8080
```

Терминал 3: SSH-туннель

```bash
ssh -R 0.0.0.0:8000:localhost:8000 -R 0.0.0.0:8080:localhost:8080 Admin@23.189.104.205
```

Проверка работоспособности

```bash
# Backend
curl http://localhost:8000

# Frontend
curl http://localhost:8080

# Через туннель (с другого компьютера)
curl http://23.189.104.205:8000
curl http://23.189.104.205:8080
```

Использование

Обработка видео через веб-интерфейс

1. Откройте в браузере: `http://23.189.104.205:8080`
2. Перетащите видеофайл в зону загрузки
3. Выберите класс объекта (Person, Car, и т.д.)
4. Выберите метод стабилизации

5. Настройте параметры (Window Size или Alpha)
6. Нажмите "STABILIZE VIDEO"
7. Дождитесь завершения обработки
8. Просмотрите результаты и метрики

Параметры стабилизации

Moving Average:

- Window Size: 3-9 (нечётные числа)
- Больше = сильнее сглаживание, но медленнее реакция

Median Filter:

- Window Size: 3-9
- Лучше убирает выбросы, сохраняет края

Exponential Smoothing:

- Alpha: 0.1-0.9
- Меньше = сильнее сглаживание
- Больше = быстрее реакция

Устранение неполадок

Проблема: Туннель отключается

Симптомы: Сайт перестаёт открываться через несколько минут.

Решение: Используйте autossh или параметры keepalive:

```
bash

ssh -o ServerAliveInterval=60 -o ServerAliveCountMax=3 -R ...
```

Проблема: "Address already in use"

Симптомы: Порт занят при запуске сервера.

Решение:

```bash
pkill -f "http.server"
pkill -f "uvicorn"
```

Проблема: "ndarray is not JSON serializable"

Симптомы: Ошибка при сохранении результатов.

Решение: Убедитесь, что в `save_job_state` удаляются все numpy-массивы:

```python
job_data.pop('frames', None)
job_data.pop('masks_before', None)
job_data.pop('masks_after', None)
job_data.pop('binary_masks', None)
```

Проблема: SSH не подключается

Проверьте:

1. Запущен ли SSH на Windows: `Get-Service sshd`
2. Открыт ли порт 22: `Test-NetConnection -ComputerName localhost -Port 22`
3. Настроен ли GatewayPorts в sshd_config
4. Разрешён ли порт в файрволе

Проблема: Обработка очень медленная

Причины и решения:

1. **Большое видео** → Используйте видео короче 30 секунд для тестов
2. **Нет GPU** → Проверьте `nvidia-smi`, настройте CUDA
3. **Мало памяти** → Уменьшите разрешение видео

Оптимизация

Многопоточная обработка

Для ускорения сегментации можно добавить в `segmentation.py`:

```python
from concurrent.futures import ThreadPoolExecutor

def segment_video_parallel(self, frames, target_class=None, max_workers=4):
    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        results = list(executor.map(
            lambda f: self.segment_frame(f, target_class),
            frames
        ))
    return results
```

GPU-ускорение

Убедитесь, что PyTorch использует GPU:

```python
import torch
print(f"CUDA available: {torch.cuda.is_available()}")
print(f"Device: {torch.cuda.get_device_name(0)}")
```

Кэширование моделей

Модель загружается один раз при старте сервера (уже реализовано в `get_segmenter()`).

Полезные команды

```bash
```

```
# Проверить логи backend
tail -f /tf/darachubarova/defliker/logs/backend.log

# Проверить статус задания
curl http://localhost:8000/api/status/JOB_ID

# Список заданий
ls /tf/darachubarova/defliker/results/outputs/

# Очистить результаты
rm -rf /tf/darachubarova/defliker/results/outputs/*
rm -rf /tf/darachubarova/defliker/results/uploads/*

# Перезапустить всё
pkill -f uvicorn; pkill -f "http.server"; pkill -f "ssh -R"
```

Авторы и благодарности

Система разработана для учебных целей.

Технологии:

- FastAPI — веб-фреймворк
- PyTorch + torchvision — нейросети
- DeepLabv3 — семантическая сегментация
- OpenCV — обработка видео
- NumPy/SciPy — численные вычисления

Документ создан: Декабрь 2024