Приложение 1

**Общество с ограниченной ответственностью «СК-Роботикс»**

**(ООО «СК-Роботикс»)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | УТВЕРЖДАЮ |
|  |  |  |  | Генеральный директор |
|  |  |  |  | Шумов М.Б. |
|  |  |  |  | 16.01 2023 г.  М.П. |

**ПРОГРАММА ОБНАРУЖЕНИЯ И КЛАССИФИКАЦИИ ДЕФЕКТОВ**

**«DEFECTOSCOPE»**

**ОПИСАНИЕ ПРОГРАММЫ**

**ЛИСТ УТВЕРЖДЕНИЯ**

**РОФ.** **93241908.01-01 13 01-ЛУ**

**Листов 2**

| Наименование организации | Должность исполнителя | Ф.И.О. | Подпись | Дата |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

УТВЕРЖДЕН

РОФ. 93241908.01-01 13 01

**ПРОГРАММА ОБНАРУЖЕНИЯ И КЛАССИФИКАЦИИ ДЕФЕКТОВ**

**«DEFECTOSCOPE»**

**ТЕКСТ ПРОГРАММЫ**

**РОФ.** **93241908.01-01 13 01**

**Листов 24**

общие сведения

Наименование и обозначение программы: «DefectoScope» РОФ.93241908.01-01.

Программные зависимости:

* build-essential, cmake – для сборки части программы на языке C++
* threads, cuda, opencv 4.6.0, gstreamer-1.0, python3, pytorch, torchvision, numpy.

Программа написана с использованием языков программирования Python, C++, Javascript.

функциональное назначение

Назначение программы «DefectoScope» - обнаружение и классификация дефектов обшивки по изображению.

Программа «DefectoScope» предназначена для использования в составе программно-аппаратного комплекса для автоматической диагностики состояния поверхности воздушного судна.

Запуск программы на другом аппаратном обеспечении возможен, но может потребовать доработки программы и перекомпиляции части, написанной на C++.

Для корректной работы программы необходимо наличие GPU с поддержкой NVIDIA CUDA.

Описание логической структуры

Алгоритмы программы

Блок формирования отчетов Reports.

Структура блока.

**Reports**

**---app**

**-------static**

**------------css**

**-----------------bootstrap** (библиотека BOOTSTRAP), **fonts** (шрифты), **chosen-sprite.jpg, chosen.css** (библиотечный выпадающий список), **font-awesome.min.css** (стили шрифтов), **main.css**

**------------js**

**-----------------scripts.js**

**------------lib** (библиотека JQUERY)

**-------templates**

**------------main.html**

**-------\_\_init\_\_\_.py**

**-------views.py**

**---database**

**-------GOSTtypeB.ttf** (файл шрифта по ГОСТ 2.304-81 тип Б, АСКОН)

**-------defects\_base.db** (файл базы данных)

**-------defects\_base.py**

**-------defects\_list.py**

**---run.py**

**Описание блока Reports.**

Блок Reports предназначен для формирования отчетов с результатами диагностики обшивки осмотренных ранее ВС. Данные для отчета хранятся в базе данных (заполнение базы данных происходит при осмотре обшивки ВС в программе «DefectoScope»).

Блок Reports представляет собой HTTP-сервер, обрабатывающий запросы пользователя. Взаимодействие с сервером осуществляется путем запросов с помощью браузера. Исходный код сервера написан на языке программирования Python, исходный код страницы, показываемой пользователю – на языке HTML/CSS. Интерактивность страницы обеспечивается кодом на языке JavaScript.

При обращении к серверу по адресу **/main** (переход по URL 127.0.0.1/main в браузере), сервер отправляет в ответ код страницы с элементами интерфейса (список ВС, кнопка для запроса отчета) и кодом на языке JavaScript для обеспечения интерактивности интерфейса страницы. Страница отображается средствами браузера.

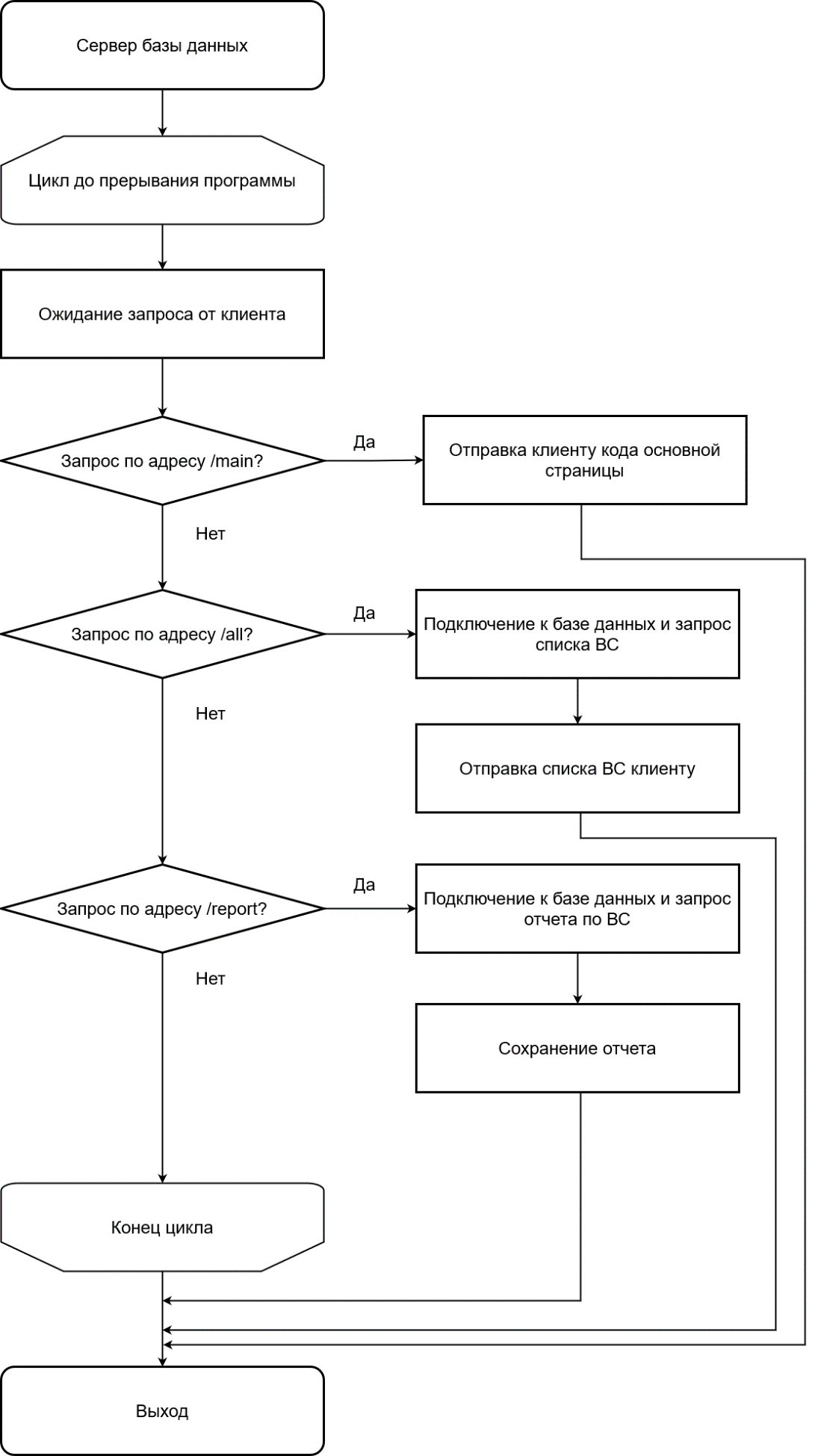
При отображении страницы на стороне клиента JavaScript код страницы запрашивает на сервере список ВС (отправка запроса по адресу 127.0.0.1/all). Сервер при получении этого запроса подключается к базе данных, запрашивает список ВС и передает его клиенту. При получении ответа JavaScript код обновляет содержимое отображаемого списка ВС с использованием полученных от сервера данных.

При выборе пользователем необходимого ВС из списка и нажатии кнопки **Report** JavaScript код страницы отправляет запрос с данными выбранного ВС (название и идентификатор) на сервер по адресу 127.0.0.1/report. Сервер запрашивает в базе данных отчет по данному ВС и сохраняет сформированный отчет.

Алгоритм блока Reports (клиент).

****

Алгоритм работы сервера.



Блок видеосервера

Содержимое блока: base64.cpp, base64.h, Camera.h, ClienWorker.h, FIFO.h, Recorder.h, utils.h, utils.cpp, main.cpp, CMakeLists.txt.

Файлы **base64.cpp** и **base64.h** – из открытых источников [<https://renenyffenegger.ch/notes/development/Base64/Encoding-and-decoding-base-64-with-cpp>].

**Описание блока видеосервера.**

Блок видеосервера предназначена для получения в реальном масштабе времени кадров изображения с видеокамеры, масштабирования кадров и передачи их в графический интерфейс (клиент).

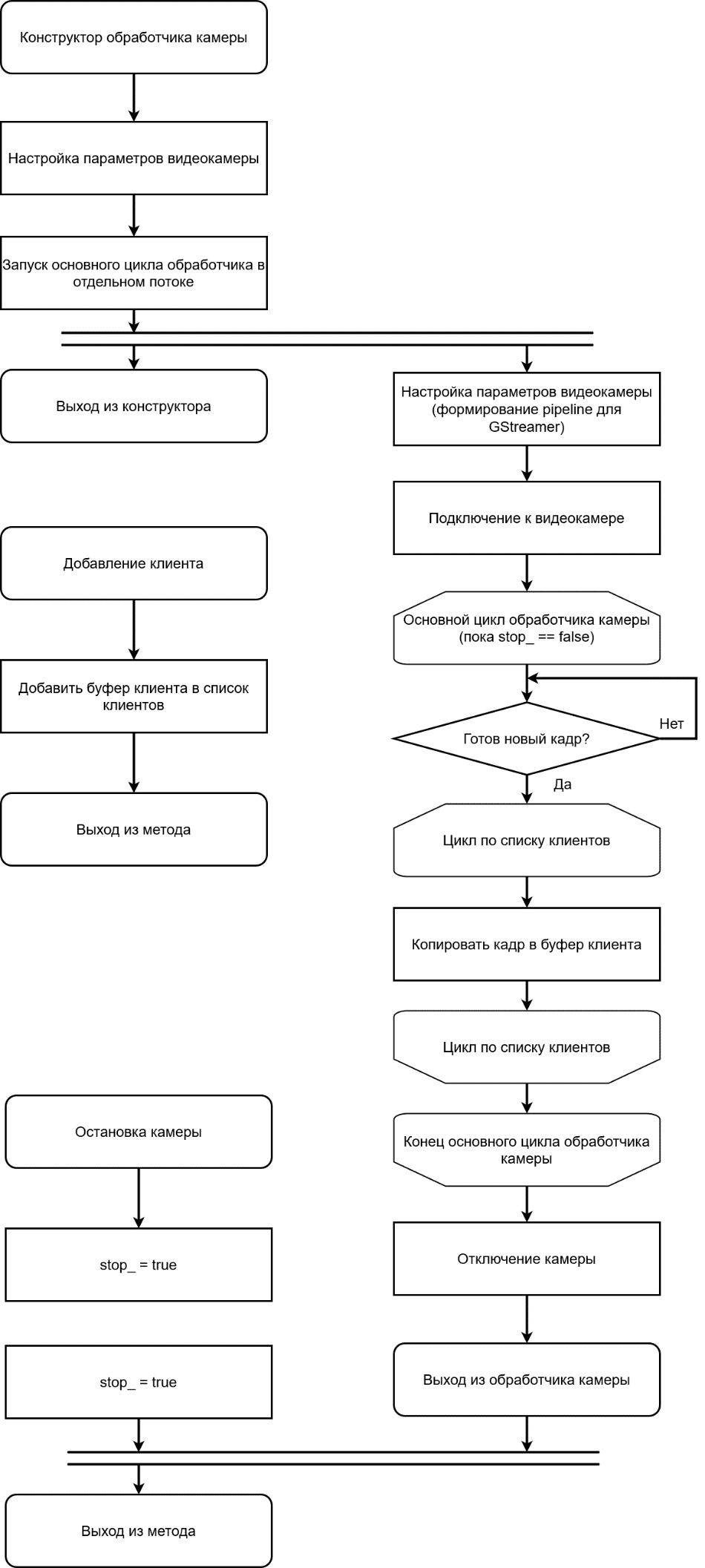
Блок написан на языке программирования C++.

Блок представляет собой сервер, взаимодействие с которым происходит с использованием механизма сокетов. При запуске блока подключается USB-видеокамера (с использованием библиотек OpenCV и GStreamer). Опрос камеры запускается в отдельном потоке. Это позволяет получать кадры с видеокамеры без пропуска данных. После этого настраивается и запускается обработчик запросов от клиента. Блок допускает подключение нескольких клиентов. В этом случае они будут получать одинаковые кадры с видеокамеры.

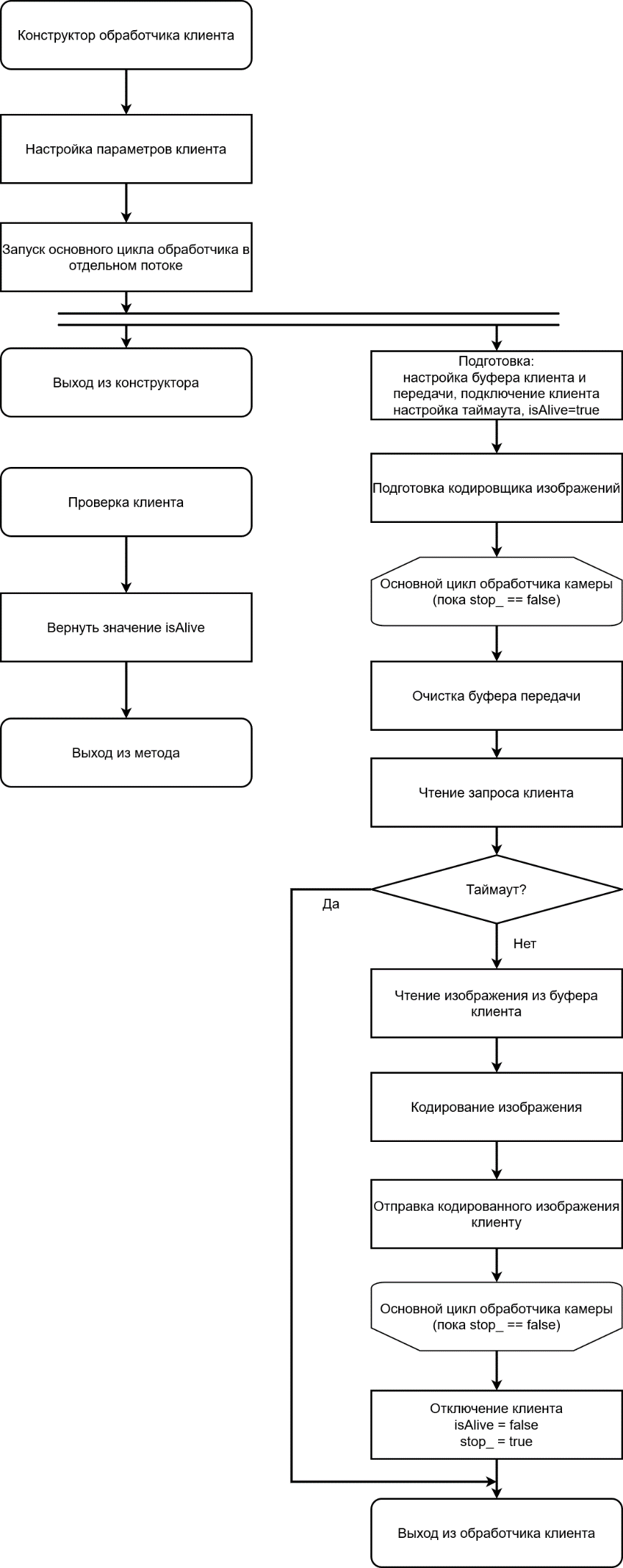
При получении нового запроса на подключение от клиента сервер создает и запускает в отдельном потоке обработчик клиента (воркер), регистрирует обработчик в обработчике камеры (для передачи кадров в буфер клиента). Таким образом, при подключении клиента, он получает индивидуальную копию данных с видеокамеры. Обработчик клиента ожидает запрос от клиента и отвечает на него, отправляя кадр изображения их буфера. Обмен данными между обработчиками клиента и обработчиком камеры реализован с помощью потоко-безопасной версии FIFO. Передача изображений клиенты осуществляется в формате JSON (кодирование бинарных данных в формате BASE64). При длительном отсутствии запросов от клиента обработчик удаляет себя из списка обработчика камеры и завершается. Основной цикл сервера периодически удаляет завершенные обработчики клиента и освобождает занимаемые ими ресурсы.

Такой подход позволяет распараллелить выход видеокамеры на несколько клиентов.

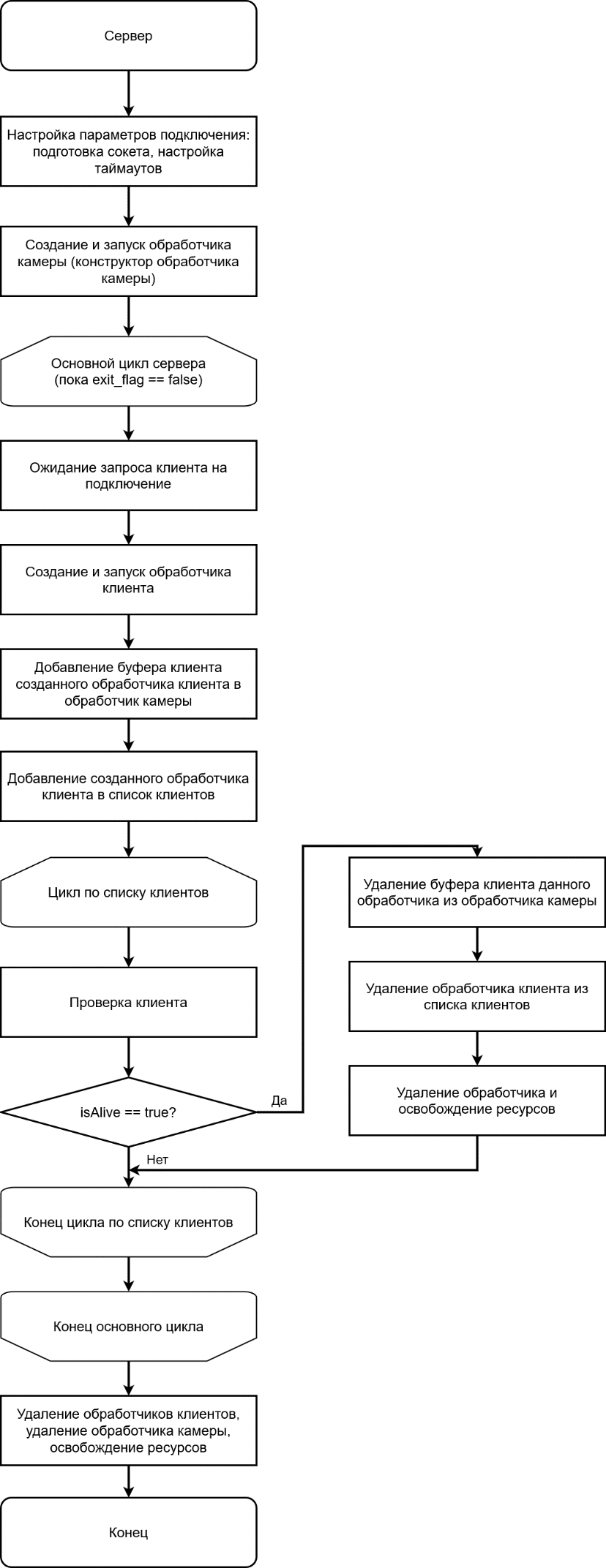
Алгоритм обработчика камеры.



Алгоритм обработчика клиента.



Алгоритм сервера.



Блок графического интерфейса

Структура каталога.

**DefectoScope**

**---database**

**------defects\_base.py**

**------defects\_list.py**

**------GOSTtypeB.ttf**

**---dataset**

**------classification**

**---------validation**

**------------Class\_1**

**---------------0.jpg**

**…**

**------------Class\_5**

**------detection\_verification**

**---------test**

**------------yet\_some\_001.jpg**

**------------yet\_some\_001.json**

**---------train**

**------------yet\_other\_001.jpg**

**------------yet\_other\_001.json**

**---detector**

**------classifier\_net**

**---------CNNModel.py**

**---------config.py**

**---------dataset\_utils.py**

**---------utils.py**

**------dectector\_net**

**---------coco\_eval.py**

**---------dataset.py**

**---------engine.py**

**---------transforms.py**

**---------utils.py**

**------models**

**---------rcnn**

**------------300.torch**

**---------classification\_model.pth**

**------analyzer.py**

**------classifier.py**

**------detector\_t.py**

**------test\_rcnn.jpg**

**---img**

**------img\_capture.png**

**------button\_new.png**

**------button\_record\_off.png**

**------button\_record\_on.png**

**------button\_stop.png**

**------button\_test.png**

**---DefectoScopeGUI.py**

**---defects\_base.db**

**---net\_test.py**

**---video\_cam.py**

**Описание блока интерфейса.**

Блок интерфейса предназначен для обнаружения и классификации дефектов обшивки ВС по изображению, получаемому с видеокамеры. Изображение с видеокамеры отображается на дисплее. Обнаруженные дефекты обводятся на изображении рамкой, рядом с рамкой выводится информация о дефекте (тип, вероятность).

Интерфейс предусматривает несколько режимов работы.

Изображение выглядит как текст

Автоматически созданное описание

Кнопка**** – запуск нового осмотра. При этом необходимо ввести название и идентификатор осматриваемого ВС.

Кнопка  - сохранение текущего изображения в файл в формате JPEG. Отображаемая информация о дефектах также сохраняется в файле.

Кнопка  - начало/остановка записи видео в файл в формате MPEG-4. Отображаемая информация о дефектах также сохраняется в файле.

Кнопка  - остановка текущего осмотра обшивки ВС. Сохранение информации об обнаруженных дефектах в базу данных с привязкой к ВС (по названию и идентификатору). В базу сохраняются изображения дефектов, их координаты, тип и вероятность.

Кнопка  - тестирование программного обеспечения в соответствии в Программой и методикой испытаний 93241908.412239.101ПМ. В этом режиме определяются метрики детектора и классификатора дефектов, в том числе точностные и временные.

Блок интерфейса определяет дефекты обшивки ВС с использованием детектора на базе обученной нейронной сети типа Mask R-CNN, тип дефектов определяется с использованием классификатора на базе обученной нейронной сети LeNET-5.

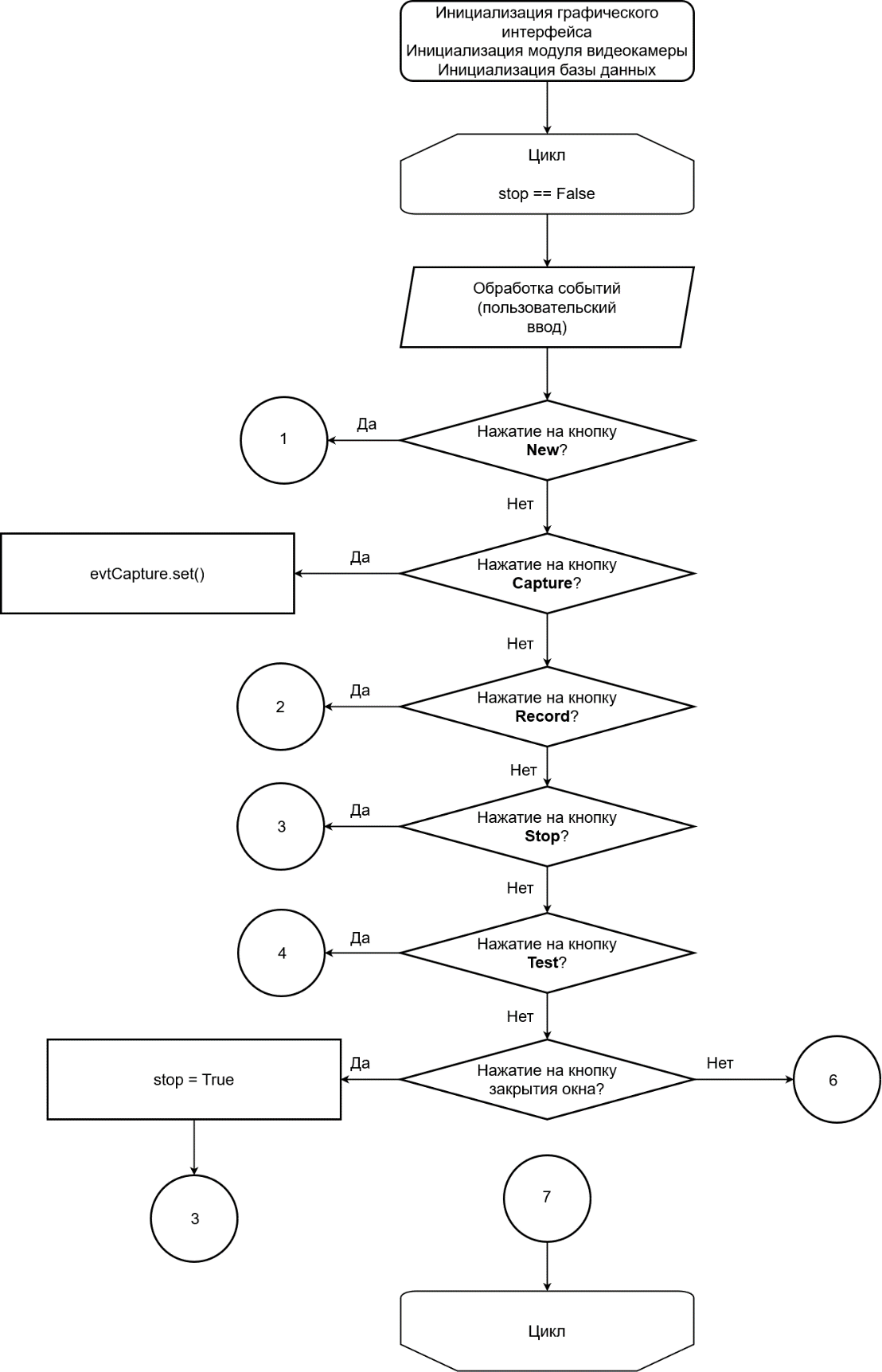
Изображение с видеокамеры отображается на дисплее. Если в данный момент активен режим осмотра обшивки ВС, изображение подается на вход детектора дефектов. Результаты его работы (координаты рамок вокруг обнаруженных дефектов) используются для выделения отдельных дефектов на изображении. Эти изображения дефектов подаются на вход классификатора дефектов для определения их типа. Результаты работы детектора и классификатора (рамки, тип дефекта, вероятность) накладываются на изображение, выводимое на дисплей.

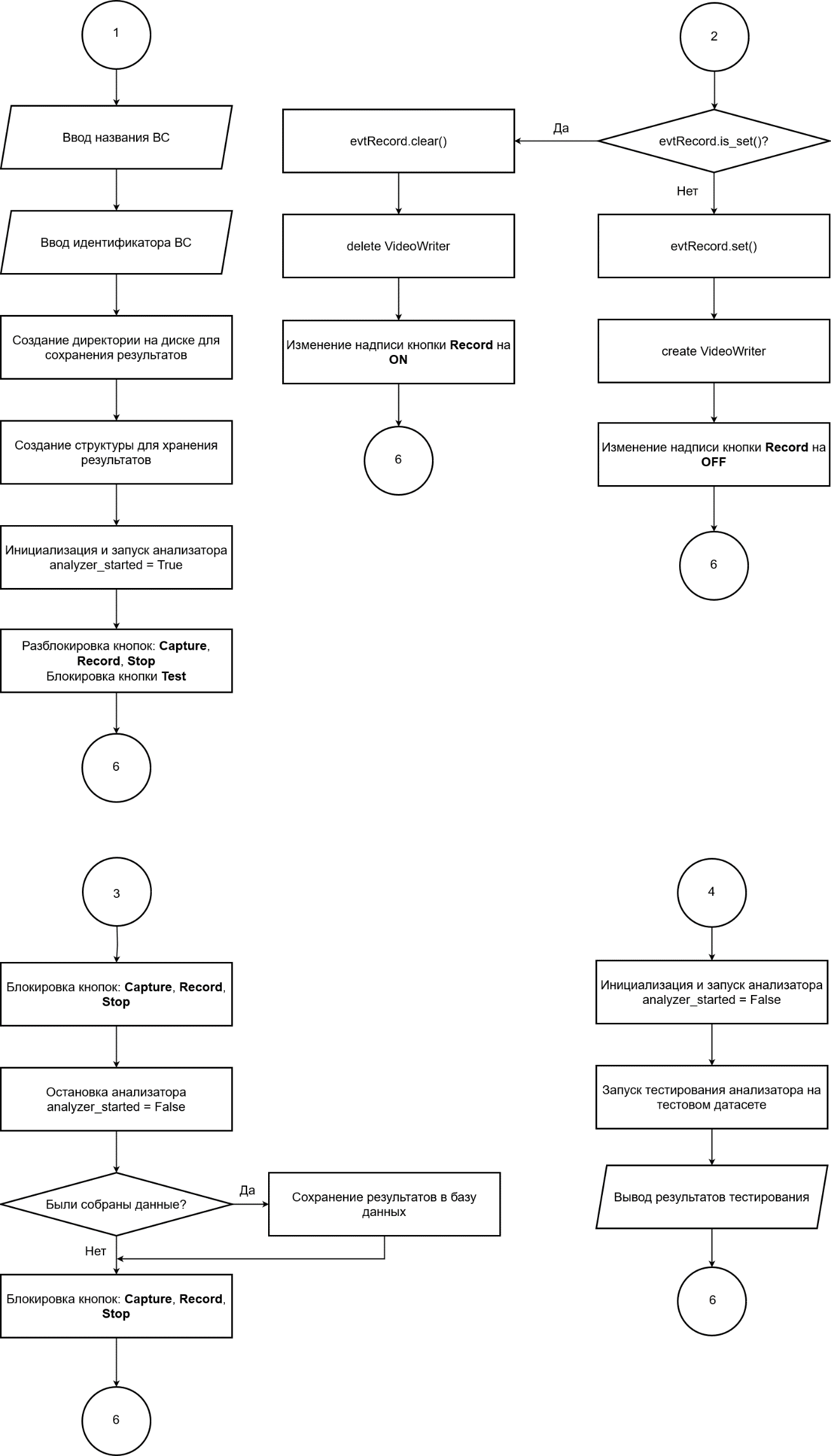
При захвате изображения текущее изображение сохраняется в файл. Если активен режим записи видео, изображение, выводимое на дисплей, также записывается в видеофайл.

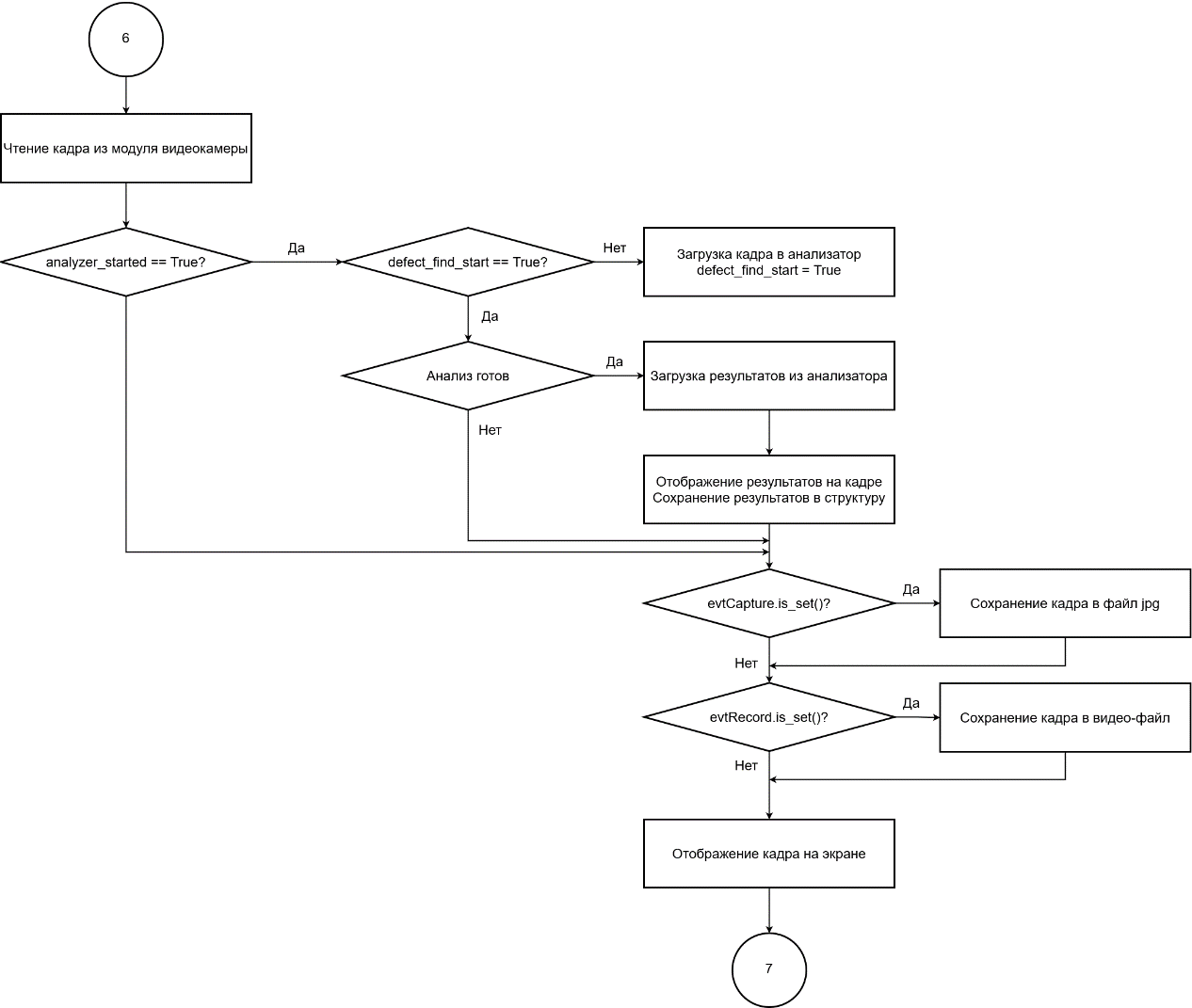
При деактивации режима осмотра обшивки ВС производится подключение к базе данных, в которой создается новая запись, содержащая название ВС, его идентификатор, дату осмотра. Также в базу данных записывается информация о найденных дефектах обшивки – изображение, координаты рамок, ограничивающих дефекты на изображении, типы дефектов, вероятности. Также автоматически формируется отчет с результатами осмотра.

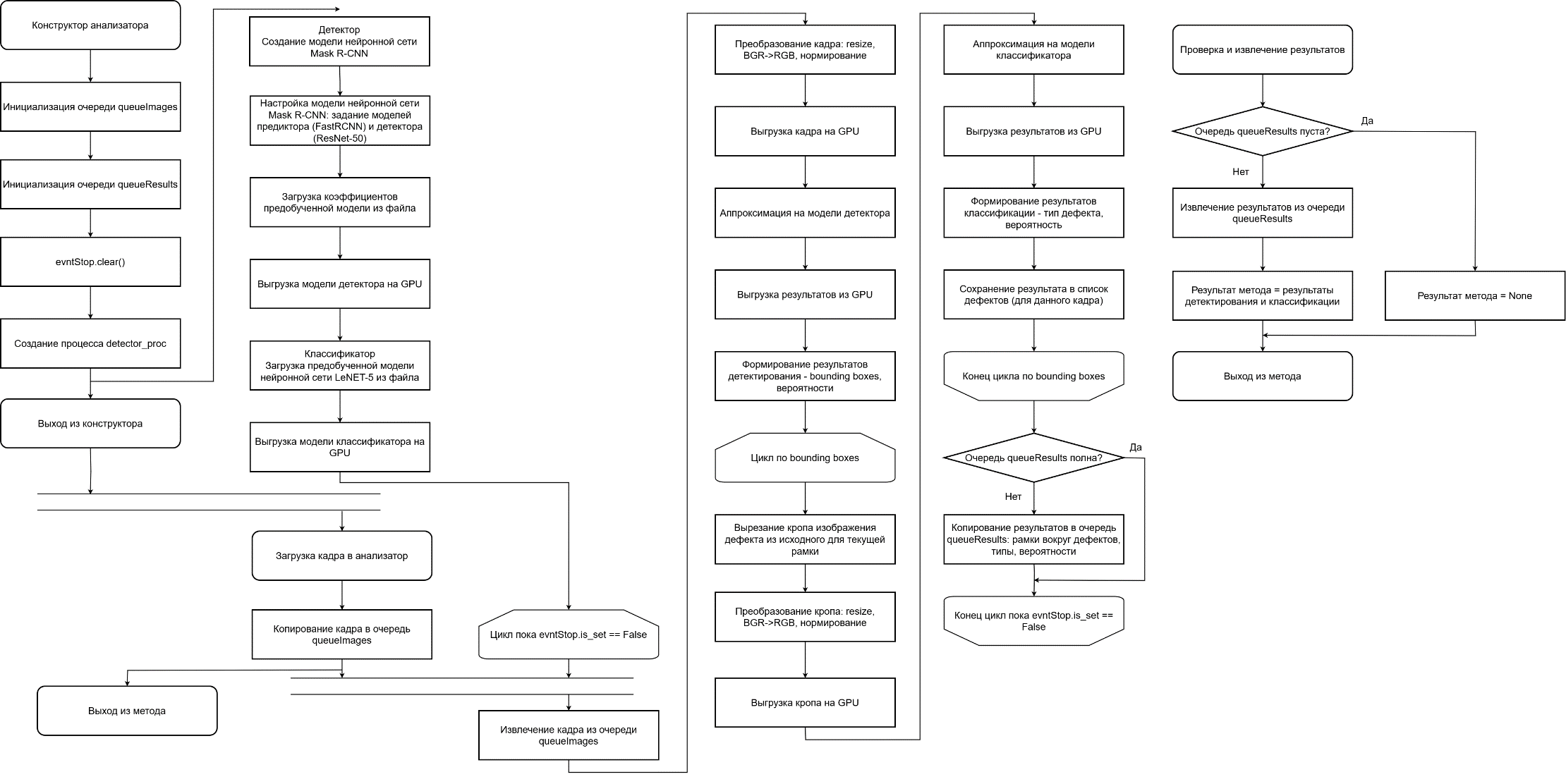
В режиме тестирования на вход детектора (и классификатора) подаются изображения из тестового набора. При этом определяется точность обнаружения дефектов (классификации) и время обнаружения (классификации). Результаты (точностные и временные характеристики детектора и классификатора) выводятся на дисплей.

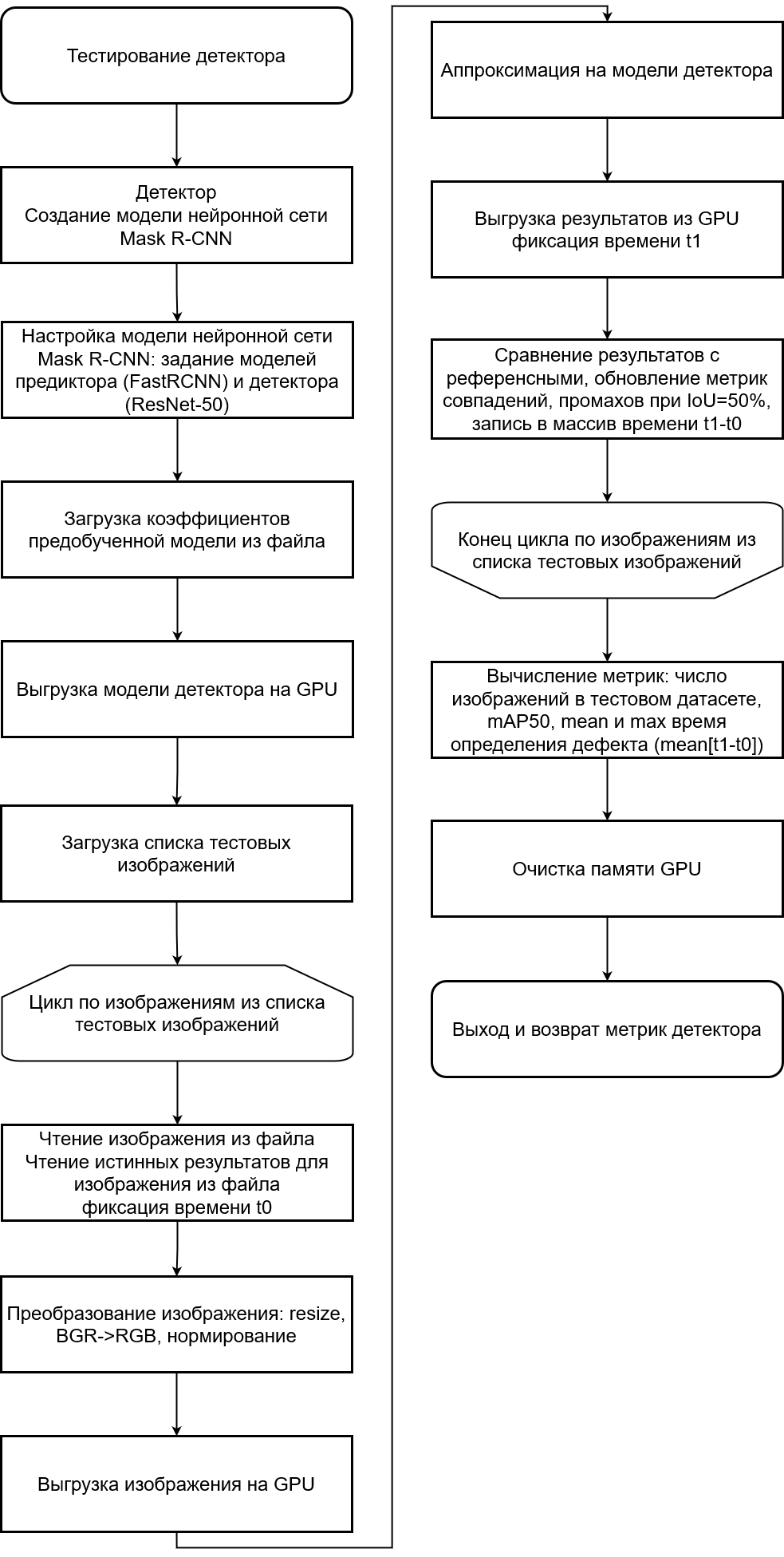
Алгоритм программы «DefectoScope».

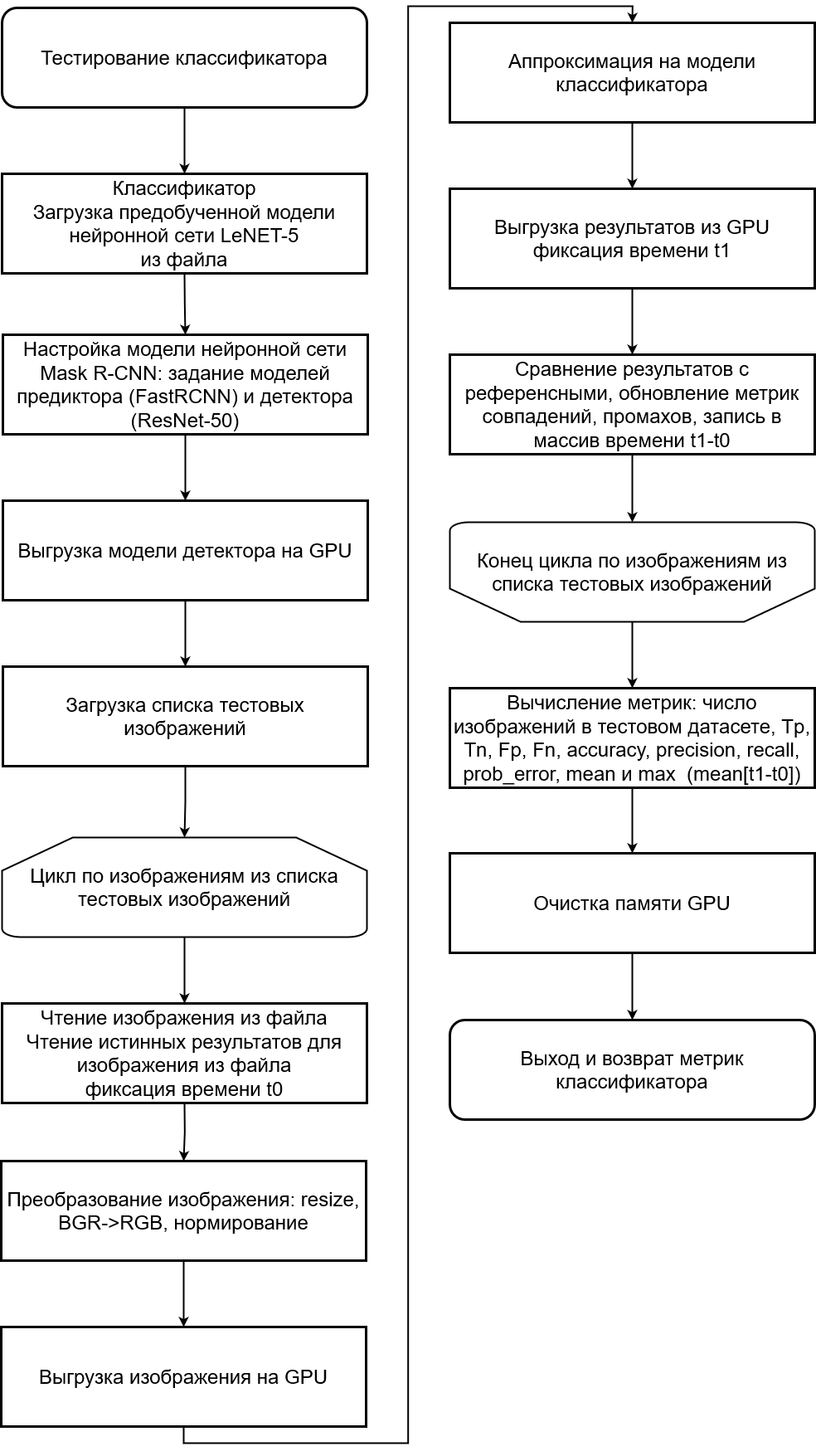












Блок обучения детектора

Структура каталога.

**DetectorTrain**

**---dataset**

**------detection\_train**

**---------test**

**------------some\_001.jpg**

**------------some\_001.json**

**---------train**

**------------other\_001.jpg**

**------------other\_001.json**

**------detection\_verification**

**---------test**

**------------yet\_some\_001.jpg**

**------------yet\_some\_001.json**

**---------train**

**------------yet\_other\_001.jpg**

**------------yet\_other\_001.json**

**---detector**

**------dectector\_net**

**---------coco\_eval.py**

**---------dataset.py**

**---------engine.py**

**---------transforms.py**

**---------utils.py**

**------models**

**---------rcnn**

**------------300.torch**

**------detector\_t.py**

**------test\_rcnn.jpg**

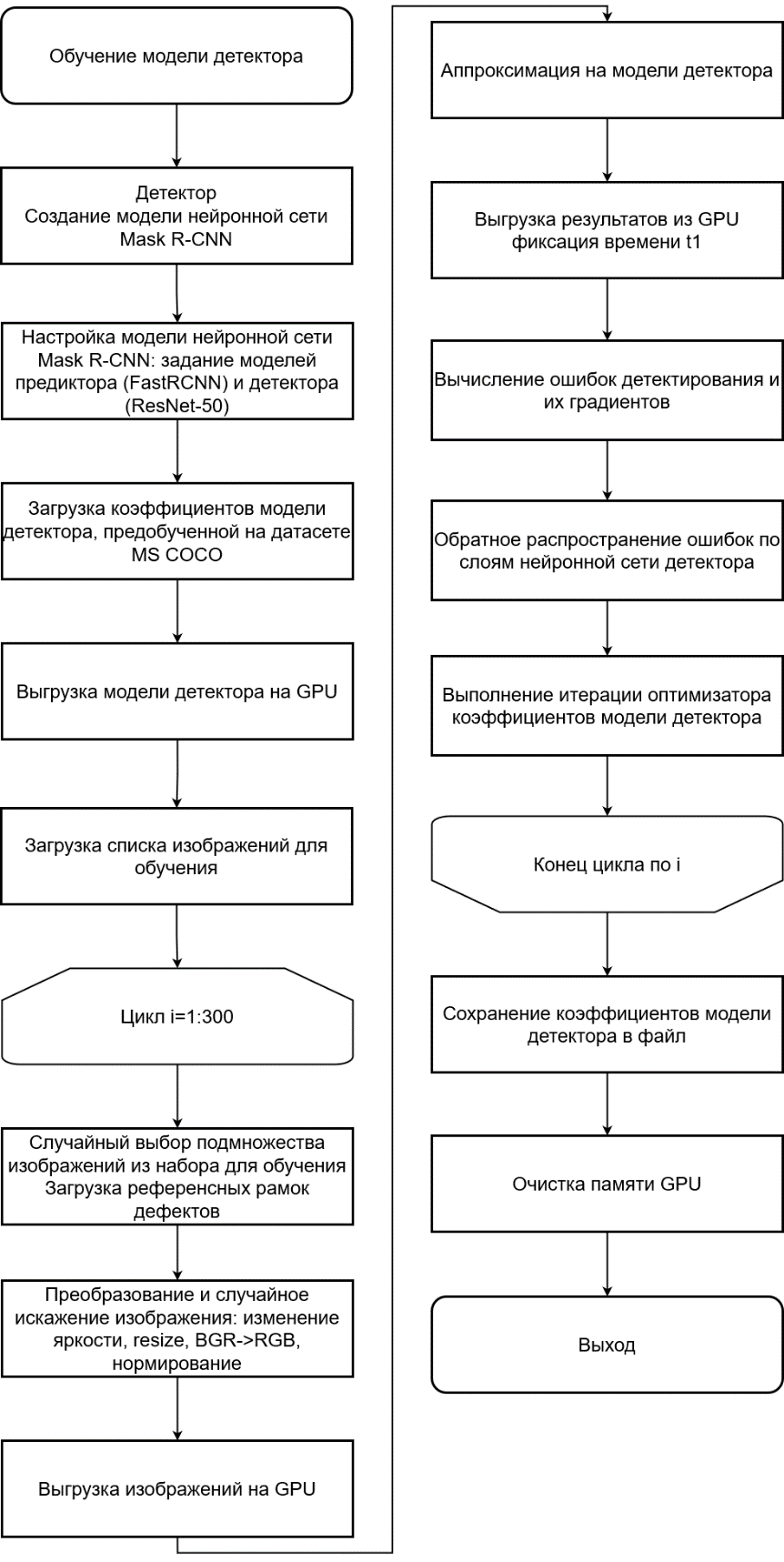
**---train\_detector.py**

**Описание блока обучения детектора.**

Блок предназначен для обучения детектора дефектов на базе нейронной сети типа Mask R-CNN. Для обучения используется библиотека pytorch и набор тестовых изображений дефектов, содержащий изображения фрагмента обшивки ВС и координаты рамок, ограничивающих дефекты на каждом изображении.

Блок формирует структуру нейронной сети, и последовательно подает на вход нейронной сети изображения обшивки. Оцененные нейронной сетью области дефектов (рамки) сравниваются с эталонными и вычисляется невязка – ошибка детектирования. Эта невязка используется для вычисления ошибок на каждом слое нейронной сети (метод обратного распространения ошибки). Затем вычисляется градиент ошибки и проводится итерация оптимизации коэффициентов нейронной сети. После повторения этой последовательности операций в течение 300 эпох ошибка детектирования уменьшается и достигает целевых приемлемых значений. Процесс обучение детектора заканчивается сохранением вычисленных коэффициентов нейронной сети в файл модели детектора.

Алгоритм обучения детектора дефектов.

****

Блок обучения классификатора

Структура каталога.

**DefectoScope**

**---dataset**

**------classification**

**---------train**

**------------Class\_1**

**---------------0.jpg**

**…**

**------------Class\_5**

**---------validation**

**------------Class\_1**

**---------------0.jpg**

**…**

**------------Class\_5**

**---detector**

**------classifier\_net**

**---------CNNModel.py**

**---------config.py**

**---------dataset\_utils.py**

**---------utils.py**

**------models**

**---------classification\_model.pth**

**------classifier.py**

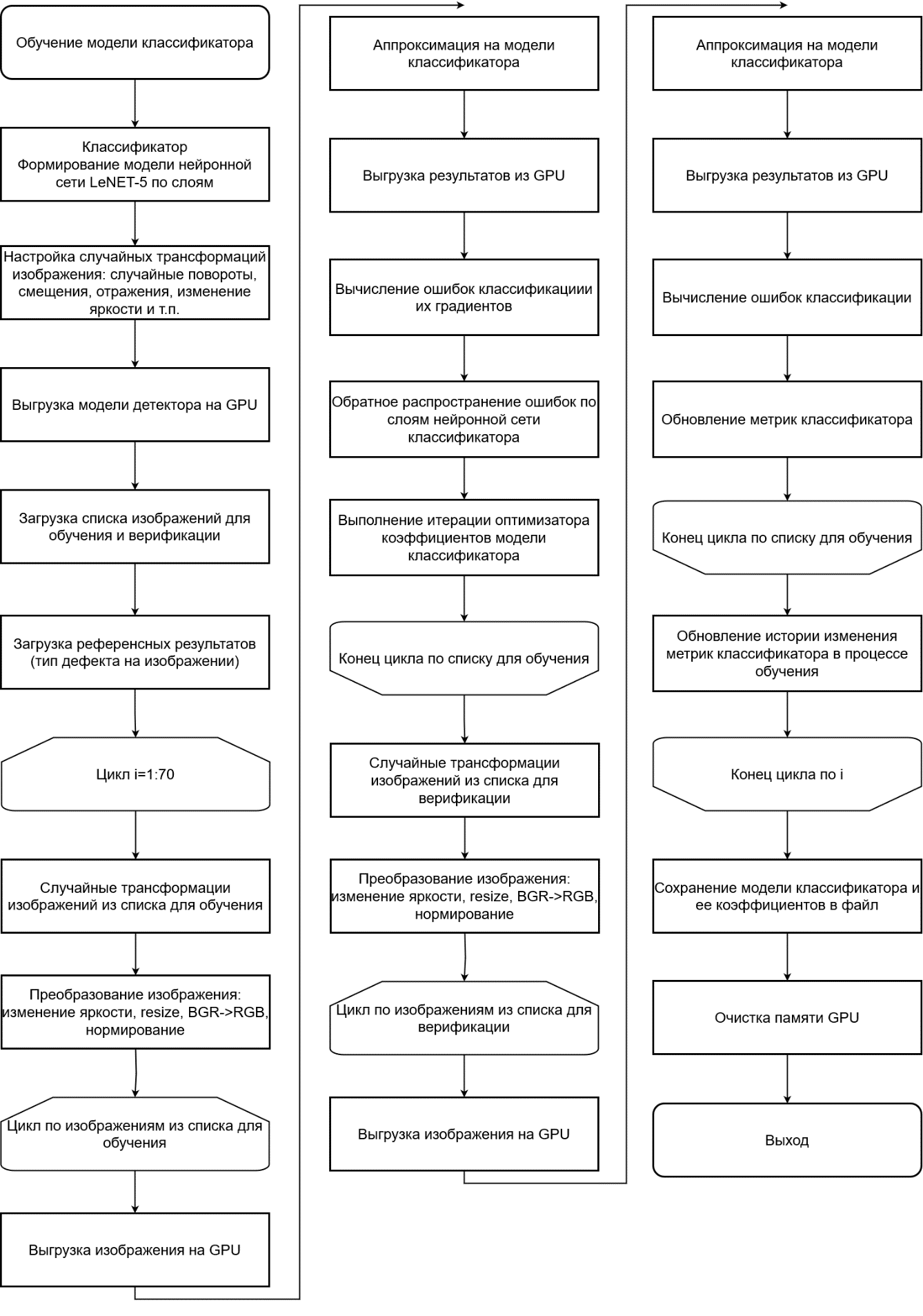
**---train\_classifier.py**

**Описание блока обучения классификатора.**

Блок предназначен для обучения классификатора дефектов на базе нейронной сети типа LeNET-5. Для обучения используется библиотека pytorch и набор тестовых изображений дефектов, сгруппированных по типам.

Блок формирует структуру нейронной сети, и последовательно подает на вход нейронной сети изображения дефектов. Оцененный нейронной сетью тип дефекта сравнивается с эталонным и вычисляется невязка – ошибка классификации. Эта невязка используется для вычисления ошибок на каждом слое нейронной сети (метод обратного распространения ошибки). Затем вычисляется градиент ошибки и проводится итерация оптимизации коэффициентов нейронной сети. После повторения этой последовательности операций в течение 70 эпох ошибка классификации уменьшается и достигает целевых приемлемых значений. Процесс обучение классификатора заканчивается сохранением вычисленных коэффициентов нейронной сети и самой структуры нейронной сети в файл модели классификатора.

Алгоритм обучения классификатора дефектов.



используемые технические средства

Программа предназначена для функционирования в операционной системе Ubuntu 18.04 на процессоре NVIDIA Jetson Xavier AGX (ARM8.2, NVIDIA Carmel, GPU NVIDIA Volta).