MMR+ANPR REST Server 4.1.1

User Guide

Contact:

Address:
Eyedea Recognition, s.r.o.
Vyšehradská 320/49
128 00, Prague 2
Czech Republic

web: http://www.eyedea.ai

email: info@eyedea.ai

Table of Contents

1 Introduction

Eyedea Recognition's MMR+ANPR REST Server is a Java server application with REST interface running on Tomcat Docker container which allows to detect vehicles (or generally "road users") in input images and recognize the type and text of detected plates, as well as the view, category, make, model, generation, variation, color and tags (various traits) of the vehicle.

MMR+ANPR REST Server uses our state-of-the-art libraries, LPM and MMR SDK, with a possibility to easily switch to the latest models or modules for a different region. Both server REST interface and the simple web application built on it provide the detection and recognition used in various use cases, as well as server monitoring.

2 Hardware requirements

2.1 Minimal requirements

- Processor: Intel® Core™ i5, 2 cores (4 logical processors)
- RAM: 4 GB
- Hard disk: 256 GB (optional SSD)
- GPU (optional): NVIDIA Driver version >= 410.48 compatible
- Operating system: Ubuntu 18.04 and higher – x86_64 platform

2.2 Recommended requirements

- Processor: Intel® Core™ i7, 4 cores (8 logical processors)
- RAM: 16 GB
- Hard disk: 512 GB, SSD
- GPU (optional): NVIDIA® GeForce® GTX 1050 Ti, 4GB GDDR5
- Operating system: Ubuntu 18.04 and higher – x86_64 platform

3 Version history

Version 4.1.1

Released: 2024/09/04

- Updated LPM modules
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd_108842-10.11.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 4.1.0

Released: 2024/08/01

- Added possibility to start server with several MMR Box/Plate engines running
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-9.12.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 4.0.1

Released: 2024/06/21

- Updated LPM and MMR modules
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-9.12.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 4.0.0

Released: 2024/04/24

- Unified interface for road users detection and recognition
- Reorganized recognition request and response
- Used combined detector (detecting license plates, boxes, windshields)
- Added possibility to start server with several detection and OCR engines running
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.22.0-Ubuntu-18.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-9.12.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 3.0.0

Released: 2023/06/29

- Supported vehicle detection + recognition
- Changed POST endpoint for license plate recognition: recognition → lpRecognition
- Renamed optional license plate recognition data parameters: boundingBox → roi, lpDetection → lpDetections
- Changed default license plate LPM module to 801 (general)
- Moved LPM module and MMR models settings from Dockerfile to env-*.list files
- Used LPM module: LPM-v7.4.1-2023-01-12-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.21.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd_108842-9.12.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.3.2

Released: 2023/05/23

- Added option to log errors, warnings and recognition statistics to files
- Added server start time to system info response
- Used LPM module: LPM-v7.4.1-2023-01-12-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.20.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd_108842-8.53.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.3.1

Released: 2023/04/14

- Supported new LPM features: added occlusion, truncated and cluster detection attributes
- Modified layout of Recognition page
- Used LPM module: LPM-v7.4.1-2023-01-12-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.20.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd-8.51.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.3.0

Released: 2023/01/05

- Supported new MMR features: added tags to response, all-in-one MMR engine
- Computing all MMR attributes (view, category, make, model, generation, variation, color and tags) by default
- Renamed MMR related Dockerfile and env.list variables
- Used LPM module: LPM-v7.4.0-2022-08-30-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.20.0-Ubuntu-18.04-x86_64-HASP
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.2.0

Released: 2022/09/27

- New product branding as "MMR+ANPR REST Server"
- Changed entry point to *SERVER_IP*:8080/RESTServer
- Used LPM module: LPM-v7.4.0-2022-08-30-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.12.0-Ubuntu-18.04-x86_64-HASP
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.1.1

Released: 2022/08/18

- Used LPM module: LPM-v7.3.1-2022-08-16-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.12.0-Ubuntu-18.04-x86_64-HASP
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.1.0

Released: 2022/07/28

- Fixed returning error message on SDK (usually license) error.
- Renamed Dockerfile and env.list variables.
- Limited number of LPM Detector CPU threads to 1 (optimized internally).
- Extended documentation.
- Updated to Java 12. Built with OpenJDK 12.0.1.
- Used LPM module: LPM-v7.3.0-2022-06-22-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.12.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd-8.41.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.0.4

Released: 2022/04/11

- Fixed LPM memory leak.
- Used LPM module: LPM-v7.2-2021-10-08-Ubuntu-16.04-hasp
- Used MMR module: Eyedea-MMR-2.11.0-Ubuntu-18.04-x86_64-HASP
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.0.3

Released: 2022/02/24

- Added support for GPU detection (currently available for LPM module 800).
- Reduced Sentinel license requirements.
- Fixed initialization of engines running on the CPU.
- LPM module selection moved to Dockerfile.
- Used LPM module: LPM-v7.2-2021-10-08-Ubuntu-16.04-hasp

- Used MMR module: Eyedea-MMR-2.11.0-Ubuntu-18.04-x86_64-HASP
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.0.2

Released: 2021/12/20

- Split MMR-VCMMGV and MMR-Color in settings, renamed Dockerfile and env.list variables.
- Used LPM module: LPM-v7.2-2021-10-08-Ubuntu-16.04-hasp
- Used MMR module: Eyedea-MMR-2.11.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd-8.31.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.0.1

Released: 2021/07/16

- Added generation and variation attributes to MMR result.
- Used LPM module: LPM-v7.1-2020-04-16-Ubuntu-16.04-hasp
- Used MMR module: Eyedea-MMR-2.10.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd-8.21.1
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.0.0

Released: 2021/02/02

- Added more detailed SDK information to serverSystemInfo response.
- Removed null / NaN fields from response.
- Used LPM module: LPM-v7.1-2020-04-16-Ubuntu-16.04-hasp
- Used MMR module: Eyedentify-VCL-2.9.0-Ubuntu-16.04-x86_64-HASP
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 2.0.0-BETA

Released: 2020/12/11

- Used LPM SDK for detection and OCR.
- Added option to specify a bounding box to reduce the input image area scanned by the detector.
- Simplified response for OCR / MMR module disabled (returning null).
- Removed countryID from anprResult response element.
- Used LPM module: LPM-v7.1-2020-04-16-Ubuntu-16.04-hasp
- Used MMR module: Eyedentify-VCL-2.9.0-Ubuntu-16.04-x86_64-HASP
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 1.2.1

Released: 2020/04/14

- Used EyeScan module: EyeScanSDK-v3.8.3-DummyPack-Ubuntu-16.04-x86_64-HASP
- Used ANPR module: Eyedentify-ANPR-2.7.0-Ubuntu-16.04-x86_64-hasp
- Used MMR module: Eyedentify-VCL-2.7.2-Ubuntu-16.04-x86_64-hasp
- Embedded modules with GPU support are using CUDA 10.0.
  (CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

Version 1.2.0

Released: 2019/10/08

- Added support for bypassing the internal detector by specifying license plates positions in JSON format.
- HTML documentation updated.
- Used EyeScan module: EyeScanSDK-v3.7.0-DummyPack-Ubuntu-16.04-x86_64-HASP
- Used ANPR module: Eyedentify-ANPR-2.5.1-Ubuntu-16.04-x86_64-hasp
- Used MMR module: Eyedentify-VCL-2.6.0-Ubuntu-16.04-x86_64-hasp
- Embedded modules with GPU support are using CUDA 9.1.85.
  (CUDA 9.1.85 requires the Linux x86_64 Driver version >= 390.46)

Version 1.1.1

Released: 2019/07/01

- Added option to run the server with ANPR / MMR modules disabled.
- HTML documentation updated.
- Embedded modules with GPU support are using CUDA 9.1.85.
  (CUDA 9.1.85 requires the Linux x86_64 Driver version >= 390.46)

Version 1.1.0

Released: 2019/04/17

- Web interface added: System Info page and Recognition page.
- Updated to the Java 11. Built with OpenJDK 11.0.2.
- HTML documentation added.
- Used EyeScan module: EyeScanSDK-v3.4.1-DummyPack-Ubuntu-16.04-x86_64-HASP
- Used ANPR module: Eyedentify-ANPR-2.5.1-Ubuntu-16.04-x86_64-hasp
- Used MMR module: Eyedentify-VCL-2.5.1-Ubuntu-16.04-x86_64-hasp
- Embedded modules with GPU support are using CUDA 9.1.85.
  (CUDA 9.1.85 requires the Linux x86_64 Driver version >= 390.46)

Version 1.1.0-BETA

Released: 2019/01/11

- Added support for descriptor batch computation.
- Added support for GPU computation (ANPR, MMR).
- Embedded modules with GPU support are using CUDA 9.1.85.
  (CUDA 9.1.85 requires the Linux x86_64 Driver version >= 390.46)

Version 1.0.0

Released: 2018/06/13

- First release version.
- MMR and ANPR recognition REST API server application with basic functionality for still photos.
- Docker image creation scripts included.

4 Docker setup

4.1 GPU Support Prerequisites

If you do not intend to use the GPU, you can skip this chapter.

1. **Install GPU**

   Install GPU supporting NVIDIA CUDA® platform into your Docker host system. The Docker host is the system running the Docker daemon.

2. **Linux x86_64**

   Using the GPU for computation in the Docker environment requires the Docker to be installed and running on the Linux x86_64 system. Use the distribution which is supported by both the Docker and the NVIDIA Driver.

3. **Install NVIDIA Driver**

   Install NVIDIA Driver into the Docker host system. See the Version history for the minimal NVIDIA Driver version required by the CUDA® used in the current build. Write down the version of the NVIDIA Driver you are installing, the same driver version must be installed in the Docker image (see the chapter Building the Docker image).

4. **Install nvidia-container-runtime library**

   Install nvidia-container-runtime library into the Docker host system. As a root, run the following script:

   ```
   ./installNvidiaContainerRuntime
   ```

4.2 Sentinel license protection system

Installation

The SDK engines used by MMR+ANPR REST Server are protected with a standard third-party software licensing solution, *Sentinel LDK by Gemalto*.

1. **Uncompress the package containing the Run-time Environment installer**

   Uncompress the aksusbd*.tar.gz file.

2. **Uninstall prior Sentinel LDK Run-time Environment version**

   If you have installed a prior version of the Sentinel LDK Run-time Environment, as a root, run the following script from the uncompressed directory to uninstall it:

   ```
   ./dunst
   ```

3. **Install the Sentinel LDK Run-time Environment**

   As a root, run the following script from the uncompressed directory to install the Sentinel LDK Run-time Environment:

   ```
   ./dinst
   ```

Note:

If you encounter the "No such file or directory" error, you may need to install the 32-bit compatibility binaries. As a root, execute the following command before installing the Sentinel LDK Run-time Environment:

```
apt-get install libc6:i386
```

4. Verify the Sentinel LDK Run-time Environment installation

To verify the Sentinel LDK Run-time Environment installation, open the address http://localhost:1947/_int_/devices.html in a web browser or check the output of the following command:

```
service aksusbd status
```

4.3 Building the Docker image

0. Prerequisites

Docker (https://www.docker.com) must be installed and Docker daemon must be running before creating the Docker image.

1. Set the Docker image variables (Dockerfile)

Instructions for building the Docker image are listed in Dockerfile which is located in the [PACKAGE]/ directory. Set the NVIDIA_DRIVER_VERSION variable (line 14) to install the proper version of the NVIDIA Driver (needed for GPU computation). The same version of the Driver must be installed on the Docker host system.

The NVIDIA Driver version may be found out by running the following command:

```
nvidia-smi
```

Set the NVIDIA_DRIVER_VERSION variable to empty to disable the NVIDIA Driver installation. In that case only CPU computation will be supported.

Example of line 14 in Dockerfile which enables GPU computation:

```
ENV NVIDIA_DRIVER_VERSION=410.104
```

Example of line 14 in Dockerfile for CPU computation only:

```
ENV NVIDIA_DRIVER_VERSION=
```

To update LPM or MMR SDK, change the LPM_VERSION (line 17) or MMR_VERSION (line 20) variables, respectively. The appropriate *.tar.gz archive must be in the [PACKAGE]/ directory.

Example of line 17 in Dockerfile defining the latest LPM SDK version:

```
ENV LPM_VERSION=v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
```

Example of line 20 in Dockerfile defining the latest Eyedea MMR SDK version:

```
ENV MMR_VERSION=2.23.0-Ubuntu-20.04-x86_64-HASP
```

2. Run the build script (buildDocker)

Use the buildDocker script located in the [PACKAGE]/ directory to build the Docker image:

```
./buildDocker
```

The Docker image is built using the following command:

```
docker build -t mmr-anpr-rest-server .
```

The `-t` option specifies the name of the new Docker image.

4.4 Running the Docker image

1. Set the application variables (env-GPU.list or env-CPU.list)

The application settings (license server address, number of threads, ...) are loaded from the system environment variables by the application. The supported variables are defined in the env-GPU.list and env-CPU.list files (the first one is intended to use the GPU for computation, the other uses the CPU). One of these files is passed to the Docker during the image initialization and its variables are available in the Docker container as system environment variables.

In the following overview, <ENGINE> stands for: LPM_DETECTOR, LPM_PLATE_OCR, MMR_PLATE and MMR_BOX.

HASP_REMOTE_SERVERADDR	The address of the Sentinel License Manager server with the valid license. If the licenses are on the Docker host, you can usually set the IP address to 10.0.75.1 or 172.17.0.1.
LPM_DETECTOR_MODULE_ID	The three-digit identifiers of LPM modules used for the detection and the plate OCR, respectively. The appropriate *.tar.gz archives must be in the [PACKAGE]/LPM-modules directory. Multiple comma-separated module identifiers can be specified; the first one is used as the default. For the detection, the usage of all-in-one detection module 802 or 803 is recommended.
LPM_PLATE_OCR_MODULE_ID	
MMR_PLATE_MODEL	Specify the binary module names used for license plate based (MMR_*) and box based (MMRBOX_*) MMR, respectively. The default MMR modules are *_VCMMGVCT_* which recognize view, category, make, model, generation, variation, color and tags. If you do not have licenses for all these features or prefer fast versions (instead of default precise ones), select the appropriate data files from the [PACKAGE]/Eyedea-MMR-${MMR_VERSION}.tar.gz/${MMR_PATH}/model directory. See the MMR documentation for details. Multiple comma-separated module names can be specified; the first one is used as the default.
MMR_BOX_MODEL	
<ENGINE>_NUMTHREADS	The number of engine's threads to initialize. Set the variable to 0 to disable the given engine. Set -1 to select the number of threads automatically (1 for CPU, or 1 for each <ENGINE>_GPU_ID in case of GPU computation mode). **Note:** When running on CPU, the number of LPM Detector threads is limited to 1 because it internally uses the optimal number of CPU threads.
<ENGINE>_COMPUTATION_MODE	Specifies whether the given engine runs on CPU, or GPU. (Certain older LPM Detectors may support only CPU.)
<ENGINE>_GPU_ID	Relevant only for GPU computation. Specifies the 0-based index or indexes of GPU devices to compute on. If more GPUs are available, you can assign a comma separated list of indexes to a single engine

	(e.g.: MMR_BOX_GPU_ID=0,1), or distribute them among multiple engines (e.g.: LPM_PLATE_OCR_GPU_ID=0 MMR_PLATE_GPU_ID=1).
LOG_STATS_PERIOD_REQUEST	Specifies whether and, if necessary, with what period recognition statistics should be logged to a file. Set a positive integer *N* to log recognition statistics to a new file every *N*-th request, or 0 to disable recognition statistics logging. See Log files chapter for details.
LOG_ERRORS	Specifies whether to log errors to a file. Set 1 to enable, or 0 to disable errors and warnings logging. See Log files chapter for details.

Check the SDK Engine Overview chapter to select the appropriate SDK engines for your use case.

Example:

Let's have the env-GPU.list file with the following configuration (only the relevant variables listed for brevity):

```
HASP_REMOTE_SERVERADDR=172.17.0.1
LPM_DETECTOR_MODULE_ID=802
LPM_DETECTOR_NUMTHREADS=1
LPM_DETECTOR_COMPUTATION_MODE=GPU
LPM_DETECTOR_GPU_ID=0
LPM_PLATE_OCR_MODULE_ID=801,800
LPM_PLATE_OCR_NUMTHREADS=1
LPM_PLATE_OCR_COMPUTATION_MODE=GPU
LPM_PLATE_OCR_GPU_ID=0
MMR_PLATE_MODEL=MMR_VCCT_FAST_2024Q2.dat
MMR_PLATE_NUMTHREADS=1
MMR_PLATE_COMPUTATION_MODE=GPU
MMR_PLATE_GPU_ID=0
MMR_BOX_NUMTHREADS=0
LOG_STATS_PERIOD_REQUEST=1000
LOG_ERRORS=1
```

The license key with the valid licenses must be plugged into the current machine.

There will be four running SDK engines: LPM Detector, two different modules of LPM Plate OCR, and License Plate based MMR, each of them running in one thread on the GPU. The Box based MMR engine will be disabled.

LPM Detector will use the all-in-one detection module 802; LPM Plate OCR will use the global module 801 (as the default one) and the European module 800 (used if requested); the MMR will use the fast model to recognize only view, category, color and tags.

Recognition statistics will be saved to a file after every 1000th recognition request is processed. Any error and warning occurrences will be saved to files, too.

2. Run the image (runDockerGPU or runDockerCPU)

Use the runDockerGPU or runDockerCPU script located in the [PACKAGE]/ directory to run the built Docker image with / without the support for GPU computation. Edit these scripts if needed.

runDockerGPU – support for GPU computation:

```
docker run --gpus device=0 --env-file env-GPU.list -p 8080:8080 mmr-anpr-rest-server
```

runDockerCPU – CPU computation only:

```
docker run --env-file env-CPU.list -p 8080:8080 mmr-anpr-rest-server
```

Notes:

The --env-file option specifies the file with the environment variables.

The -p option publishes port 8080 in the container and maps it to the host's port 8080.

The --gpus option specifies GPU device(s) used by the container. Possibilities:

- Use a GPU device using its index: --gpus device=0
- Use multiple GPU devices using their indexes: --gpus '"device=2,3"'
- Use all available GPU devices: --gpus all

The server tries to initialize all engine instances specified in the env-*.list file. Check the error output for details if there are problems starting any engine. After launching the Docker container, the application is accessible at:

```
http://[MACHINE_IP]:8080/RESTServer/
```

You can check the status of the engines on the application home page or by requesting the serverSystemInfo. "DISABLED" means that no thread was requested for the given engine, "FAILED TO START" means that all requested threads of the given engine could not be started; in both cases, the server handles requests and returns data obtained by other running engines (unless the user request explicitly includes a task processed by an engine that is not running).

The engine status never changes when the container is started. If an engine encounters a license problem after the successful initialization, its status will still be "RUNNING", but the response to a request for that engine will have a status code of 500 with a message indicating the problematic engine (e.g.: "Error during OCR processing. Computation error. Please check your licenses.").

4.5 Log files

There is a possibility to enable logging of recognition statistics and errors in order to regularly monitor the server usage and report problems. Logging can be enabled by setting the corresponding variable in the env-*.list file.

If enabled, each statistics record or problem produces a new file <Type>_<year>-<month>-<day>_<hour>-<minute>-<second>-<millisecond>.log in the Docker container's /usr/local/tomcat/webapps/RESTServer_data/logs/ folder.

Log file name

The following table lists possible values of <Type> in the log file name:

Type	Description
Stats	Recognition statistics.
BadRequest	400 Bad Request response status code is returned.
ErrorResponse	500 Internal Server Error response status code is returned.

RuntimeError	More serious problem during server startup or recognition processing.
RuntimeWarning	Less serious problem during server startup or recognition processing.

Recognition statistics

Recognition statistics log files contain a JSON object which contains the following items:

Response item	Description	Data type
requestTimestamp	Date and time of the user request with a millisecond precision.	String
responseTimestamp	Date and time of the server response with a millisecond precision.	String
serverStartedTimestamp	Date and time when the server started with a millisecond precision.	String
totalTaskCount	Number of user recognition requests (including erroneous).	Integer
totalDetectionCount	Number of processed detection computation tasks.	Integer
totalOcrCount	Number of processed OCR computation tasks.	Integer
totalMmrCount	Number of processed MMR computation tasks.	Integer

Errors

Error log files contain a text message describing the problem.

Notes

To run the Docker container with a volume to store log files on the host, add `-v $PWD/logs:/usr/local/tomcat/webapps/RESTServer_data/logs` to the `runDocker*` script.

The time zone used for date and time associated with log files depends on the Docker container locale, which is UTC by default. To use the host time zone instead, add `-v /etc/timezone:/etc/timezone:ro` to the `runDocker*` script.

5 REST API Documentation

5.1 SDK Engine Overview

Which SDK engines do you need? Use the combination of the SDK engines from the following table.

Use Case	SDK Engine
Plate detection	LPM Detector
Vehicle detection	LPM Detector *(module 802 or 803)*
Windshield detection	LPM Detector *(module 802 or 803)*
Plate OCR	LPM Plate OCR
Vehicle recognition	LPM License Plate based *or* MMR Box based

LPM Detector detects plates on vehicles (license plates, ADR and trash plates, etc.); modules 802 and 803 also detect boxes (whole vehicles and some other "road users" like pedestrians, kickbikes, etc.) and windshields.

LPM Plate OCR recognizes the text and type of plates detected by LPM License Plate Detector or provided in the request.

MMR License Plate based can recognize the view, category, make, model, generation, variation, color and tags of vehicles specified by their license plate. Either LPM Detector detecting plates is needed, or the license plate detections must be provided in the request.

MMR Box based recognizes the view, category, make, model, generation, variation, color and tags of vehicles (or "road users") specified by their bounding box. Either LPM Detector detecting boxes (module 802 or 803) is needed, or the box detections must be provided in the request.

LPM modules

The following table lists the LPM modules shipped with the current version of MMR+ANPR REST Server.

LPM module ID	Detected objects	OCR region
553	Plates	North America
555	Plates	Middle East and Africa (selected countries)
556	Plates	Asia (selected countries)
557	Plates	Australia
800	Plates	Europe
801	Plates	Global (selected countries)
802	Plates, boxes and windshields (fast)	-
803	Plates, boxes and windshields (precise)	-

Whether an LPM module is available depends on the MMR+ANPR REST Server settings. See the chapter Running the Docker image for details.

MMR modules

The following table explains the meaning of each part of the MMR binary module names shipped with the current version of MMR+ANPR REST Server.

MMR module	Description
MMR_*	License plate based MMR module
MMRBOX_*	Box based MMR module
*_VCCT_*	Recognizes view, category, color and tags
*_VCMCT_*	Recognizes view, category, make, color and tags
*_VCMMCT_*	Recognizes view, category, make, model, color and tags
*_VCMMGVCT_*	Recognizes view, category, make, model, generation, variation, color and tags
*_FAST_*	Fast MMR module
*_PREC_*	Precise MMR module

For a more detailed description, check the Eyedea MMR SDK documentation.

Whether a MMR module is available depends on the MMR+ANPR REST Server settings. See the chapter Running the Docker image for details.

5.2 Response Status Codes

The following table lists all response status codes MMR+ANPR REST Server returns.

Status Code	Meaning
200	Success.
400	Bad Request. The request was invalid. The error message provides the details.
500	Internal Server Error. Most of these errors are caused by a licensing issue. The error message provides the details.

5.3 Entry point

SERVER_IP:8080/RESTServer

5.4 System Info

System info contains information about system resources and SDK engines.

Main web page with real time server monitoring

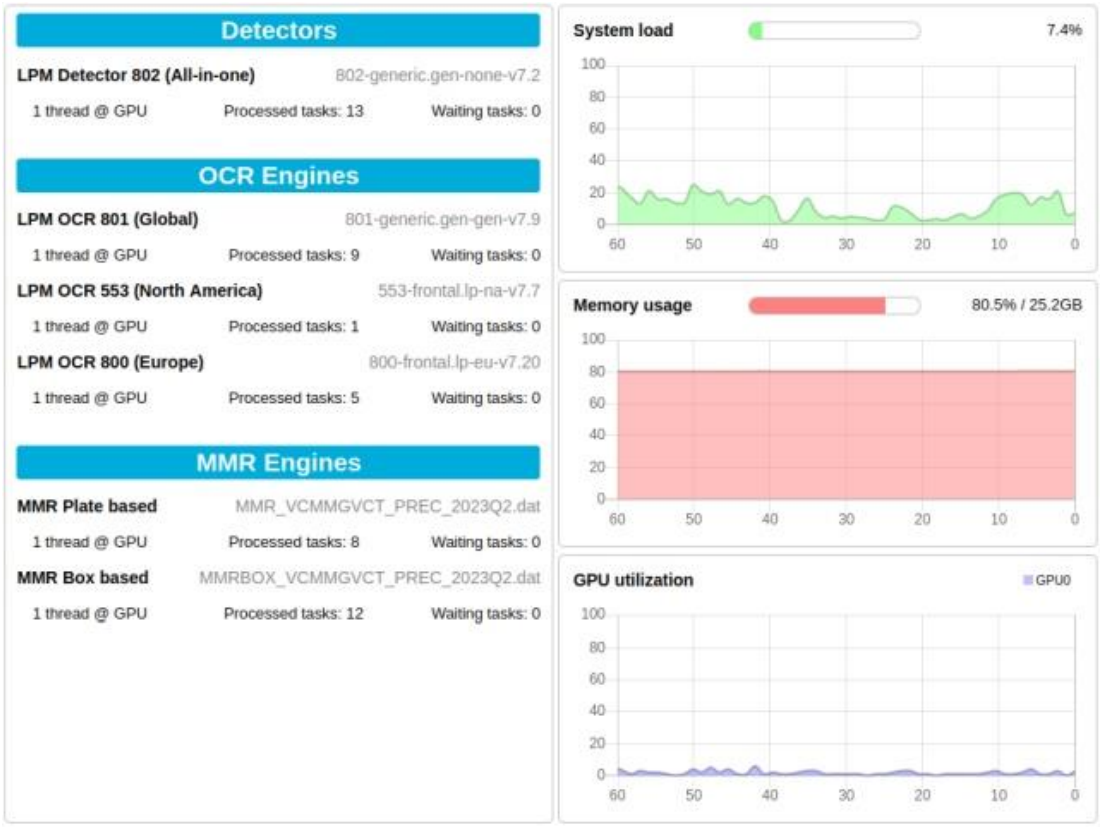
Endpoint:	/

HTTP Method:	GET
Consumes Media Type:	-
Produces Media Type:	text/html
URL Parameters:	-
Data Parameters:	-

CURL Command Example:

```
curl -X GET -H "Accept: text/html" http://localhost:8080/RESTServer/
```

Response Example:



Text output with server monitoring information

Endpoint:	/
HTTP Method:	GET
Consumes Media Type:	-
Produces Media Type:	text/plain
URL Parameters:	-
Data Parameters:	-

CURL Command Example:

```
curl -X GET -H "Accept: text/plain" http://localhost:8080/RESTServer/
```

Response Example:

```
Current time: 2023-10-23 15:02:00
Started on: 2023-10-23 14:59:06

System:
 - Average load: 8.21%
 - Used memory: 34.45% (10.79 GB)

GPU 0 (Graphics Device):
 - Utilization: 1%
 - Free memory: 3172 / 5905 MB

LPM Detector 802:
 - Engine version: v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
 - Module ID: 802
 - Module version: 802-generic.gen-none-v7.2
 - Computation mode: GPU
 - Status: RUNNING
 - Number of running threads: 1
 - Number of waiting tasks: 0
 - Number of processed tasks: 12

LPM OCR 801:
 - Engine version: v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
 - Module ID: 801
 - Module version: 801-generic.gen-gen-v7.9
 - Computation mode: GPU
 - Status: RUNNING
 - Number of running threads: 1
 - Number of waiting tasks: 0
 - Number of processed tasks: 3

LPM OCR 800:
 - Engine: LPM OCR 800
 - Version: v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
 - Module ID: 800
 - Module version: 800-frontal.lp-eu-v7.20
 - Computation mode: GPU
 - Status: RUNNING
 - Number of running threads: 1
 - Number of waiting tasks: 0
 - Number of processed tasks: 5

LPM OCR 553:
 - Engine version: v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
 - Module ID: 553
 - Module version: 553-frontal.lp-na-v7.7
 - Computation mode: GPU
 - Status: RUNNING
 - Number of running threads: 1
 - Number of waiting tasks: 0
 - Number of processed tasks: 4

MMR License Plate based:
 - Engine version: 2.21.0-Ubuntu-18.04-x86_64-HASP
 - Module: MMR_VCMMGVCT_PREC_2023Q2.dat
 - Module version: 20230512
 - Computation mode: GPU
 - Status: RUNNING
 - Number of running threads: 1
 - Number of waiting tasks: 0
 - Number of processed tasks: 12
```

```
MMR Box based:
 - Engine version: 2.21.0-Ubuntu-18.04-x86_64-HASP
 - Module: MMRBOX_VCMMGVCT_PREC_2023Q2.dat
 - Module version: 20230513
 - Computation mode: GPU
 - Status: RUNNING
 - Number of running threads: 1
 - Number of waiting tasks: 0
 - Number of processed tasks: 7
```

Get detailed server system info

Endpoint:	/info
HTTP Method:	GET
Consumes Media Type:	-
Produces Media Type:	application/json
URL Parameters:	-
Data Parameters:	-

CURL Command Example:

```
curl -X GET http://localhost:8080/RESTServer/info
```

Response Example:

```
{
  "timestamp": 1697697742603,
  "serverStartedTimestamp": 1697697574258,
  "system": {
    "cpuUsage": 4.120443,
    "memoryUsage": 34.440250,
    "memoryUsageBytes": 11584397312,
    "gpus": [
      {
        "id": 0,
        "name": "Graphics Device",
        "totalMemoryMB": 5905,
        "freeMemoryMB": 3172,
        "gpuUtilizationPerc": 1
      }
    ]
  },
  "engines": [
    {
      "computationMode": "GPU",
      "engineName": "LPM",
      "engineType": "Detector",
      "engineVersion": "v7.6-2023-11-10-Ubuntu-18.04-hasp9.0",
      "moduleId": 802,
      "moduleVersion": "802-generic.gen-none-v7.2",
      "numComputingThreads": 1,
      "numProcessedTasks": 12,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "DETECTION"
    },
    {
```

```
      "computationMode": "GPU",
      "engineName": "LPM",
      "engineType": "OCR",
      "engineVersion": "v7.6-2023-11-10-Ubuntu-18.04-hasp9.0",
      "moduleId": 801,
      "moduleVersion": "801-generic.gen-gen-v7.9",
      "numComputingThreads": 1,
      "numProcessedTasks": 3,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "OCR"
    },
    {
      "computationMode": "GPU",
      "engineName": "LPM",
      "engineType": "OCR",
      "engineVersion": "v7.6-2023-11-10-Ubuntu-18.04-hasp9.0",
      "moduleId": 800,
      "moduleVersion": "800-frontal.lp-eu-v7.20",
      "numComputingThreads": 1,
      "numProcessedTasks": 5,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "OCR"
    },
    {
      "computationMode": "GPU",
      "engineName": "LPM",
      "engineType": "OCR",
      "engineVersion": "v7.6-2023-11-10-Ubuntu-18.04-hasp9.0",
      "moduleId": 553,
      "moduleVersion": "553-frontal.lp-na-v7.7",
      "numComputingThreads": 1,
      "numProcessedTasks": 4,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "OCR"
    },
    {
      "computationMode": "GPU",
      "engineName": "MMR",
      "engineType": "License Plate based",
      "engineVersion": "2.21.0-Ubuntu-18.04-x86_64-HASP",
      "moduleName": "MMR_VCMMGVCT_PREC_2023Q2.dat",
      "moduleVersion": "20230512",
      "numComputingThreads": 1,
      "numProcessedTasks": 12,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "MMR"
    },
    {
      "computationMode": "GPU",
      "engineName": "MMR",
      "engineType": "Box based",
      "engineVersion": "2.21.0-Ubuntu-18.04-x86_64-HASP",
      "moduleName": "MMRBOX_VCMMGVCT_PREC_2023Q2.dat",
      "moduleVersion": "20230513",
      "numComputingThreads": 1,
      "numProcessedTasks": 7,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "MMR"
    }
  ]
}
```

Response Definitions:

The following table describes each item in the response.

Response item	Description	Data type
timestamp	Current system time in milliseconds.	Integer
serverStartedTimestamp	System time in milliseconds when the server started.	Integer
system	System information	Object
system/cpuUsage	Recent CPU usage for the whole system in percent.	Decimal number
system/memoryUsage	Amount of physical memory used by any process in percent.	Decimal number
system/memoryUsageBytes	Amount of physical memory used by any process in bytes.	Integer
system/gpus	Information about available GPU devices.	Array of objects
system/gpus/id	GPU device ID.	Integer
system/gpus/name	GPU device name.	String
system/gpus/totalMemoryMB	Total amount of physical GPU device memory in megabytes.	Integer
system/gpus/freeMemoryMB	Amount of free physical GPU device memory in megabytes.	Integer
system/gpus/gpuUtilizationPerc	Utilization of the GPU device in percent.	Integer
engines	Information about the SDK engines.	Array of objects
engines/computationMode	Type of processing unit used for computation. Either "CPU", or "GPU".	String
engines/engineName	SDK engine name.	String
engines/engineType	SDK engine type. Either "Detector", "OCR", or "MMR".	String
engines/engineVersion	SDK engine version.	String
engines/moduleId	DK engine module ID.	Integer
engines/moduleName	SDK engine module name.	String
engines/moduleVersion	SDK engine module version.	String
engines/numComputingThreads	Number of initialized engine's threads.	Integer
engines/numProcessedTasks	Number of processed computation tasks.	Integer
engines/numWaitingTasks	Number of queued computation tasks to be processed.	Integer
engines/status	SDK engine status. Either "RUNNING" (at least 1 thread successfully initialized), "DISABLED" (no thread requested), or "FAILED TO START" (all requested threads could not be started).	String

	engines/task	The computation task associated with the SDK engine. Either "DETECTION", "OCR", or "MMR".	String

5.5 Recognition

Recognition provides information about recognized road users and their components in the input image.

Supported image formats: JPG, JPEG, PNG, BMP, TIFF, WEBP.

Web page with image upload and recognition functionality using the license plate, or box detector

Endpoint:	/recognition
HTTP Method:	GET
Consumes Media Type:	-
Produces Media Type:	text/html
URL Parameters:	-
Data Parameters:	-

CURL Command Example:

```
curl -X GET http://localhost:8080/RESTServer/recognition
```

Response Examples:

Plates, boxes and windshields:

Boxes only:

Notes:

LPM detector, LPM Plate OCR and MMR (compatible with the detector) engines must be running for both ANPR and MMR results to be displayed.

The ANPR section displays the OCR result of *license* plates. If multiple license plates are detected on one vehicle, the OCR of the most confident detection is selected.

Basic information about the detected objects in the image is displayed as a tooltip when you hover the mouse cursor over them.

Occluded attribute is based on the value of **occlusion**: if occlusion is greater or equal to 0.25, the plate is considered to be occluded (marked with a green ✓); otherwise it is considered to be not occluded (marked with a red ✗).

The score of an MMR tag (in the MMR tags section) is displayed as a tooltip when you hover the mouse cursor over it.

Detect and/or recognize license plates or vehicles in the input image

Endpoint:	/recognition
HTTP Method:	POST
Consumes Media Type:	multipart/form-data
Produces Media Type:	application/json
URL Parameters:	-
Data Parameters:	file – form data parameter containing input image

request – (optional) JSON object specifying the recognition request:

Request item	Description	Data type
tasks	An array of requested computation tasks: "DETECTION", "OCR", "MMR" (case insensitive). Optional; if absent, all available tasks are applied depending on other request items.	Array of strings; *optional*
combinations	An array containing combinations of detected road user objects (box and/or plates). Road users in a combination belong to each other and are usually physically connected. If absent, the detections must be obtained by the detector using the DETECTION task. Allowed only in combination with the [OCR, MMR], [OCR] and [MMR] tasks (without DETECTION task).	Array of objects; *optional*
combinations/roadUsers	An array containing road users belonging to the combination. Road users sharing the same combination belong to each other and are usually physically connected.	Array of objects
combinations/roadUsers/box	Information about the detected vehicle.	Object; *optional*
combinations/roadUsers/box/position	Information about the detected vehicle bounding box location in the input image.	Object
combinations/roadUsers/box/position/topLeftCol	The horizontal coordinate of the top-left corner of the box within the image.	Decimal number
combinations/roadUsers/box/position/topLeftRow	The vertical coordinate of the top-left corner of the box within the image.	Decimal number
combinations/roadUsers/box/position/bottomRightCol	The horizontal coordinate of the bottom-right corner of the box within the image.	Decimal number
combinations/roadUsers/box/position/bottomRightRow	The vertical coordinate of the bottom-right corner of the box within the image.	Decimal number
combinations/roadUsers/plates	Information about the detected plates (license plate, ADR or trash plate, etc.).	Array of objects; *optional*
combinations/roadUsers/plates/position	Information about the detected plate location in the input image. Mandatory for OCR. Optional for MMR; if absent, the center coordinates, rotation and either pixelsPerMeter, dimension or isSingleLine must be provided.	Object; *optional*
combinations/roadUsers/plates/position/topLeftCol	The horizontal coordinate of the top-left corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/topLeftRow	The vertical coordinate of the top-left corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/topRightCol	The horizontal coordinate of the top-right corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/topRightCol	The vertical coordinate of the top-right corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/bottomRightCol	The horizontal coordinate of the bottom-right corner of the plate within the image.	Decimal number

combinations/roadUsers/plates/position/**bottomRightCol**	The vertical coordinate of the bottom-right corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomLeftCol**	The horizontal coordinate of the bottom-left corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomLeftRow**	The vertical coordinate of the bottom-left corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/**centerCol**	The horizontal coordinate of the plate center within the image. Optional for MMR; if absent, position (the coordinates of all four corners) must be provided.	Decimal number; *optional*
combinations/roadUsers/plates/**centerRow**	The vertical coordinate of the plate center within the image. Optional for MMR; if absent, position (the coordinates of all four corners) must be provided.	Decimal number; *optional*
combinations/roadUsers/plates/**rotation**	Clockwise correction of the source image in degrees. Optional for MMR; if absent, position (the coordinates of all four corners) must be provided.	Decimal number; *optional*
combinations/roadUsers/plates/**pixelsPerMeter**	The real-world scale in pixels per meter of the image computed on the plate. For example, plate of width = 300 millimeters, photographed to have 150 pixels, has 150 / 0.3 = 500 pixels per meter scale. Optional for MMR; if absent, position (the coordinates of all four corners), dimension or isSingleLine must be provided.	Decimal number; *optional*
combinations/roadUsers/plates/**isSingleLine**	True for single-line license plate, False for multi-line or ADR. Optional for MMR; if absent, position (the coordinates of all four corners), pixelsPerMeter or dimension must be provided.	Boolean; *optional*
combinations/roadUsers/plates/**dimension**	The physical dimensions of the plate. Optional for MMR; if absent, position (the coordinates of all four corners), pixelsPerMeter or isSingleLine must be provided.	Object; *optional*
combinations/roadUsers/plates/dimension/**width**	The physical width of the plate in millimeters.	Integer
combinations/roadUsers/plates/dimension/**height**	The physical height of the plate in millimeters.	Integer
requestedDetectionTypes	An array of object types to be detected by the detector: "BOX", "PLATE", "WINDSHIELD" (case insensitive). Optional; if absent, detections of all types supported by the detector will be returned. Allowed only if the tasks contain DETECTION.	Array of objects; *optional*
roi	The area of the input image to be scanned by the detector. Optional; if absent, the whole image will be scanned. Allowed only if the tasks contain DETECTION.	Object; *optional*
roi/**topLeftCol**	The horizontal coordinate of the top-left corner of the region of interest within the image.	Integer
roi/**topLeftRow**	The vertical coordinate of the top-left corner of the region of interest within the image.	Integer
roi/**bottomRightCol**	The horizontal coordinate of the bottom-right corner of the region of interest within the image.	Integer
roi/**bottomRightRow**	The vertical coordinate of the bottom-right corner of the region of interest within the image.	Integer

detectionModuleId	The three-digit LPM Detector identifier to be used for the detection (see the LPM modules table). If absent, the default one will be used. Allowed only if the tasks contain DETECTION.	Integer; *optional*
ocrModuleId	The three-digit LPM Plate OCR identifier to be used for the OCR computation (see the LPM modules table). Optional; if absent, the default one will be used. Allowed only for the requestedDetectionTypes containing PLATE and tasks containing OCR.	Integer; *optional*
mmrPreference	The preferred type of object for which the MMR is computed. Either "BOX", or "PLATE" (case insensitive). Optional; if absent, BOX is preferred. Allowed only if the tasks contain MMR. Must not be combined with the mmrModuleNames containing a single module name.	String; *optional*
mmrModuleNames	An object containing the names of the binary modules used for the MMR computation. Optional; if absent, the default ones (depending on the mmrPreference) will be used. Allowed only if the tasks contain MMR.	Object; *optional*
mmrModuleNames/box	The name of the binary module used for the MMR Box computation, e.g. "MMRBOX_VCMMGVCT_PREC_2024Q2.dat" (case insensitive). Optional; if absent, the MMR Box engine will not be used.	String; *optional*
mmrModuleNames/plate	The name of the binary module used for the MMR Plate computation, e.g. "MMR_VCMMGVCT_PREC_2024Q2.dat" (case insensitive). Optional; if absent, the MMR Plate engine will not be used.	String; *optional*

Notes:

Request can contain manual detections. In that case, the **tasks** must not contain "DETECTION" (the detector will not be used). Each **roadUser** (vehicle) can contain one **box** and/or one or more **plates**. The encapsulating **combinations** are sets of one or more **roadUsers**, e.g. a tractor with a trailer. The **combinations** structure will be projected into the response.

For the OCR of a manually specified **plate**, its **position** (all four corners) must be specified. In case of a license plate, the OCR needs to know whether it is a single-line or multi-line. The engine takes this information from **isSingleLine**, if available, or from **dimension** otherwise. If none of them is specified, the type will be estimated.

If a **roadUser** contains more than one detected object (and MMR **task** is requested), the MMR is computed using only one of them based on the **mmrPreference**. If there are multiple plates and one of them is to be used for the MMR, the license plate with the highest detection score will be selected.

If the **mmrModuleNames** element is specified with both **box** and **plate**, the **box** module will be used for the MMR Box computation, and the plate module will be used for the MMR Plate computation. If only one of these modules is specified, only the corresponding engine will be used for the MMR computation. The **mmrPreference** is allowed only when **mmrModuleNames** include both **box** and **plate** elements or when **mmrModuleNames** are not specified at all (using the default MMR modules).

For the MMR of a manually specified **plate**, its center coordinates (**centerCol** and **centerRow**), **rotation** and **pixelsPerMeter** needs to be known. If not specified, they can be estimated from the plate corners (**position**); however, for the more accurate estimation of pixelsPerMeter parameter, either **dimension** or **isSingleLine** should be also specified.

CURL Command Examples:

Default request: detection, plate OCR and MMR

```
curl -X POST \
     -H "Content-Type: multipart/form-data" \
     -F "file=@image.jpg" \
     http://localhost:8080/RESTServer/recognition
```

Request in CURL command: plate detection, OCR and MMR

```
curl -X POST \
     -H "Content-Type: multipart/form-data" \
     -F "file=@image.jpg" \
     -F "request={\"requestedDetectionTypes\":[\"PLATE\"]}" \
     http://localhost:8080/RESTServer/recognition
```

Request in JSON string (see JSON string examples below)

```
curl -X POST \
     -H "Content-Type: multipart/form-data" \
     -F "file=@image.jpg" \
     -F "request=@request.json" \
     http://localhost:8080/RESTServer/recognition
```

JSON string example: box and plate detection and (plate) OCR

```
{
  "tasks": [ "DETECTION", "OCR" ],
  "requestedDetectionTypes": [ "BOX", "PLATE" ]
}
```

JSON string example: box detection and MMR in the specified region of interest

```
{
  "requestedDetectionTypes": [ "BOX" ],
  "roi": {
    "topLeftCol": 176, "topLeftRow": 362,
    "bottomRightCol": 1234, "bottomRightRow": 1000
  }
}
```

JSON string example: detection, (plate) OCR and MMR for American region (LPM Plate OCR module 553)

```
{
  "ocrModuleId": 553
}
```

JSON string example: MMR based on the specified box detection

```
{
  "combinations": [{ "roadUsers": [{ "box": {
    "position": {
      "topLeftCol": 5, "topLeftRow": 69,
      "bottomRightCol": 954, "bottomRightRow": 642
    }
  } }] }]
}
```

JSON string example: MMR based on the specified plate detection

```
{
  "combinations": [{ "roadUsers": [{ "plates": [{
    "centerCol": 176.5,
    "centerRow": 362.0,
    "rotation": -5.1,
    "pixelsPerMeter": 214.3
  }] }] }]
}
```

JSON string example: OCR and MMR based on the specified plate detection

```
{
  "combinations": [{ "roadUsers": [{ "plates": [{
    "position": {
      "topLeftCol": 817.2, "topLeftRow": 892.5,
      "topRightCol": 1181.4, "topRightRow": 880.0,
      "bottomRightCol": 1184.0, "bottomRightRow": 956.5,
      "bottomLeftCol": 819.9, "bottomLeftRow": 969.0
    },
    "isSingleLine": true
  }] }] }]
}
```

Notes:

Content type of the request is **multipart/form-data**, where the inputs are stored in the form data fields.

Input image is referenced with filepath (starting with @) in the form field **file**.

Optional parameter **request** can be referenced with filepath (starting with @, referencing a text file containing JSON string) or can contain the whole JSON as a string (special characters must be escaped).

There are many conditions limiting the use of optional **request** items. If the element combination is not allowed, the Bad Request error code 400 is returned with a message describing the problem. Some of the illegal combinations:

- **tasks** containing "DETECTION" with **detections** (if the **detections** element is present, no detector is used).
- **mmrPreference** with **tasks** explicitly not containing "MMR" (the **mmrPreference** element is related to the MMR task).

If the requested **detectionModuleId** or **ocrModuleId** engines are not available, the Bad Request error code 400 is returned with a message describing the problem.

If the requested engines (specified explicitly by **tasks**, **detectionModuleId** or **ocrModuleId** elements) are available but not running, the Internal Server error code 500 is returned with a message describing the problem. This makes the difference between the following cases:

- If the request contains **tasks** containing "MMR" (e.g.: "tasks": ["DETECTION", "OCR", "MMR"]) and none of the MMR engines is running, the Internal Server error code 500 is returned.
- If the request does not contain **tasks** element at all and all MMR engines are disabled, the server runs just the detection and plate OCR.

Response Example:

Automatic detection of all object types, OCR and vehicle recognition:

```
{
  "combinations": [
    {
      "roadUsers": [
        {
          "box": {
            "position": {
              "topLeftCol": 207.98956,
              "topLeftRow": 254.33014,
              "bottomRightCol": 1550.7578,
              "bottomRightRow": 1209.3694,
              "score": 0.9650459
            },
            "occlusion": 0.0010549176,
            "truncated": false
```

```
        },
        "plates": [
          {
            "position": {
              "topLeftCol": 817.24255,
              "topLeftRow": 892.5229,
              "topRightCol": 1181.434,
              "topRightRow": 879.9859,
              "bottomRightCol": 1184.0664,
              "bottomRightRow": 956.45624,
              "bottomLeftCol": 819.875,
              "bottomLeftRow": 968.9932,
              "score": 0.9058306
            },
            "occlusion": 0.0011757425,
            "truncated": false,
            "clusterScore": 1.0,
            "isSingleLine": true,
            "type": {
              "value": "N",
              "score": 0.99995106
            },
            "text": {
              "value": "CV50070",
              "score": 0.9999991
            },
            "dimension": {
              "width": 520,
              "height": 110,
              "score": 1.0
            }
          }
        ],
        "windshield": {
          "position": {
            "topLeftCol": 360.14645,
            "topLeftRow": 341.9826,
            "topRightCol": 1299.7959,
            "topRightRow": 321.45706,
            "bottomRightCol": 1304.929,
            "bottomRightRow": 556.4519,
            "bottomLeftCol": 365.27963,
            "bottomLeftRow": 576.9774,
            "score": 0.9375341
          },
          "occlusion": 0.00004424591,
          "truncated": false,
          "clusterScore": 1.0
        },
        "mmr": {
          "view": {
            "value": "frontal",
            "id": 1,
            "score": 0.9999329
          },
          "category": {
            "value": "CAR",
            "id": 2,
            "score": 0.9997303
          },
          "make": {
            "value": "VW",
            "id": 43,
            "score": 0.9999216
          },
          "model": {
            "value": "Passat",
```

```
            "id": 3569,
            "score": 0.9990373
          },
          "generation": {
            "value": "Mk VI (2005)",
            "id": 3691,
            "score": 0.9979515
          },
          "color": {
            "value": "WHITE",
            "id": 13,
            "score": 0.94161624
          },
          "tags": [
            {
              "name": "ambulance",
              "value": "no",
              "id": 14,
              "score": 0.99998176
            },
            {
              "name": "caravan",
              "value": "no",
              "id": 10,
              "score": 0.99946773
            },
            {
              "name": "fire_brigade",
              "value": "no",
              "id": 16,
              "score": 0.9999783
            },
            {
              "name": "law_enforcement",
              "value": "yes",
              "id": 11,
              "score": 1.0
            },
            {
              "name": "pickup",
              "value": "no",
              "id": 38,
              "score": 0.6701908
            },
            {
              "name": "push_bumper",
              "value": "no",
              "id": 30,
              "score": 0.9948259
            },
            {
              "name": "towed",
              "value": "no",
              "id": 51,
              "score": 0.9955255
            },
            {
              "name": "wood_truck",
              "value": "no",
              "id": 46,
              "score": 0.9893941
            }
          ],
          "input": {
            "box": {
              "topLeftCol": 207.98956,
              "topLeftRow": 254.33014,
```

```
                "bottomRightCol": 1550.7578,
                "bottomRightRow": 1209.3694
              }
            }
          }
        }
      ]
    }
  ],
  "engines": [
    {
      "task": "DETECTION",
      "moduleId": 802,
      "moduleVersion": "802-generic.gen-none-v7.2"
    },
    {
      "task": "OCR",
      "moduleId": 801,
      "moduleVersion": "801-generic.gen-gen-v7.9"
    },
    {
      "task": "MMR",
      "moduleName": "MMRBOX_VCMMGVCT_PREC_2023Q2.dat",
      "moduleVersion": "20230513"
    }
  ]
}
```


Notes:

If the request contains **combinations** (manual detections), the corresponding elements with the input values are projected into the **combinations** response element (in the same order). However, the **position** element is copied from the request to the response only if its encapsulating object contains some information added by the analysis. This means that a manually detected **plate** will be returned as an empty element if it is not processed by the OCR; a manually detected **box** will always be returned as an empty element.

The **engines** element lists only SDK engines, which were used to obtain the results. For example, if the request contains **combinations** (manual detections), there is no "DETECTION" **task** in the **engines** element.

Response Definitions:

The following table describes each item in the response.

Response item	**Description**	**Data type**
combinations	An array containing the detection and recognition results.	Array of objects
combinations/roadUsers	An array containing clusters of detected objects belonging to a single road user and recognition results.	Array of objects
combinations/roadUsers/box	Information about the detected vehicle bounding box.	Object
combinations/roadUsers/box/position	Information about the detected vehicle bounding box location in the input image.	Object
combinations/roadUsers/box/position/topLeftCol	The horizontal coordinate of the top-left bounding box corner within the image.	Decimal number
combinations/roadUsers/box/position/topLeftRow	The vertical coordinate of the top-left bounding box corner within the image.	Decimal number

combinations/roadUsers/box/position/bottomRightCol	The horizontal coordinate of the bottom-right bounding box corner within the image.	Decimal number
combinations/roadUsers/box/position/bottomRightRow	The vertical coordinate of the bottom-right bounding box corner within the image.	Decimal number
combinations/roadUsers/box/position/score	The vehicle detection confidence factor. Range 0 – 1.	Decimal number
combinations/roadUsers/box/occlusion	Specifies how much the detection is occluded. Range 0 (not occluded) – 1 (fully occluded).	Decimal number
combinations/roadUsers/box/truncated	Specifies whether the detection is truncated (the bounding box does not cover the whole object).	Boolean
combinations/roadUsers/plates	Information about the detected plates.	Array of objects
combinations/roadUsers/plates/position	Information about the detected plate location in the input image.	Object
combinations/roadUsers/plates/position/topLeftCol	The horizontal coordinate of the top-left plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/topLeftRow	The vertical coordinate of the top-left plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/topRightCol	The horizontal coordinate of the top-right plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/topRightCol	The vertical coordinate of the top-right plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/bottomRightCol	The horizontal coordinate of the bottom-right plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/bottomRightCol	The vertical coordinate of the bottom-right plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/bottomLeftCol	The horizontal coordinate of the bottom-left plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/bottomLeftRow	The vertical coordinate of the bottom-left plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/score	The plate detection confidence factor. Range 0 – 1.	Decimal number
combinations/roadUsers/plates/occlusion	Specifies how much the detection is occluded. Range 0 (not occluded) – 1 (fully occluded).	Decimal number
combinations/roadUsers/plates/truncated	Specifies whether the detection is truncated (the bounding box does not cover the whole object).	Boolean
combinations/roadUsers/plates/clusterScore	Confidence factor that the plate belongs to the road user. Range 0 – 1.	Decimal number
combinations/roadUsers/plates/isSingleLine	Specifies whether the license plate has a single line: true for single-line, false for multi-line or ADR.	Boolean

combinations/roadUsers/plates/**type**	Information about the recognized plate type.	Object
combinations/roadUsers/plates/type/**value**	The international country code for license plates; "ADR" or "TRASH" for ADR plates; "TRAILER" with the international country code for license plates on trailers. If the value is "UNK", then it was recognized as a false positive detection.	String
combinations/roadUsers/plates/type/**score**	The confidence factor for the type prediction. Range 0 – 1.	Decimal number
combinations/roadUsers/plates/**text**	Information about the recognized plate text.	Object
combinations/roadUsers/plates/text/**value**	The plate text recognized by the OCR. In case of multi-line text, the lines are separated by a semicolon (";").	String
combinations/roadUsers/plates/text/**score**	The confidence factor for the text prediction. Range 0 – 1.	Decimal number
combinations/roadUsers/plates/**dimension**	The predicted physical dimensions of the detected plate.	Object
combinations/roadUsers/plates/dimension/**width**	The predicted physical width of the detected plate in millimeters.	Integer
combinations/roadUsers/plates/dimension/**height**	The predicted physical height of the detected plate in millimeters.	Integer
combinations/roadUsers/plates/dimension/**score**	Confidence factor for the dimensions prediction. Range 0 – 1.	Decimal number
combinations/roadUsers/**windshield**	Information about the detected windshield.	Object
combinations/roadUsers/windshield/**position**	Information about the detected windshield location in the input image.	Object
combinations/roadUsers/windshield/position/**topLeftCol**	The horizontal coordinate of the top-left windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**topLeftRow**	The vertical coordinate of the top-left windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**topRightCol**	The horizontal coordinate of the top-right windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**topRightCol**	The vertical coordinate of the top-right windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**bottomRightCol**	The horizontal coordinate of the bottom-right windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**bottomRightCol**	The vertical coordinate of the bottom-right windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**bottomLeftCol**	The horizontal coordinate of the bottom-left windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**bottomLeftRow**	The vertical coordinate of the bottom-left windshield corner within the image.	Decimal number

combinations/roadUsers/windshield/position/score	The windshield detection confidence factor. Range 0 – 1.	Decimal number
combinations/roadUsers/windshield/occlusion	Specifies how much the detection is occluded. Range 0 (not occluded) – 1 (fully occluded).	Decimal number
combinations/roadUsers/windshield/truncated	Specifies whether the detection is truncated (the bounding box does not cover the whole object).	Boolean
combinations/roadUsers/windshield/clusterScore	Confidence factor that the windshield belongs to the road user. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr	Information about the recognized vehicle attributes.	Object
combinations/roadUsers/mmr/view	The recognized vehicle view.	Object
combinations/roadUsers/mmr/view/value	The value of the vehicle view, either "FRONTAL", or "REAR".	String
combinations/roadUsers/mmr/view/id	ID of the recognized vehicle view.	Integer
combinations/roadUsers/mmr/view/score	The confidence factor for the view result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/category	The recognized vehicle category.	Object
combinations/roadUsers/mmr/category/value	The value of the vehicle category, e.g., "BUS", "CAR", "HVT", ... For the full list of possible categories and their definition, check the Eyedea MMR SDK documentation.	String
combinations/roadUsers/mmr/category/id	ID of the recognized vehicle category.	Integer
combinations/roadUsers/mmr/category/score	The confidence factor for the category result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/make	The recognized vehicle make.	Object
combinations/roadUsers/mmr/make/value	The recognized vehicle manufacturer, e.g., "VW", "Ford", "Fiat", ... For the full list of possible makes, check the Eyedea MMR SDK documentation.	String
combinations/roadUsers/mmr/make/id	ID of the recognized vehicle manufacturer.	Integer
combinations/roadUsers/mmr/make/score	The confidence factor for the make result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/model	The recognized vehicle model.	Object
combinations/roadUsers/mmr/model/value	The recognized vehicle model (vehicle instance defined by a bodywork), e.g., "Golf", "Mondeo", "500", ...	String
combinations/roadUsers/mmr/model/id	ID of the recognized vehicle model.	Integer

combinations/roadUsers/mmr/model/score	The confidence factor for the model result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/generation	The recognized vehicle generation.	Object
combinations/roadUsers/mmr/generation/value	The recognized vehicle generation (vehicle mark and first model year), e.g., "Mk VI (2019)", "Mk I (2020)", ...	String
combinations/roadUsers/mmr/generation/id	ID of the recognized vehicle generation.	Integer
combinations/roadUsers/mmr/generation/score	The confidence factor for the generation result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/variation	The recognized vehicle variation.	Object
combinations/roadUsers/mmr/variation/value	The recognized vehicle variation (vehicle trim level and/or body type), e.g., "AMG", "AMG-Line SUV", "Coupe", ...	String
combinations/roadUsers/mmr/variation/id	ID of the recognized vehicle variation.	Integer
combinations/roadUsers/mmr/variation/score	The confidence factor for the variation result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/color	The recognized vehicle color.	Object
combinations/roadUsers/mmr/color/value	The recognized vehicle color, e.g., "BLUE", "GRAY", "RED", ...	String
combinations/roadUsers/mmr/color/id	ID of the recognized vehicle color.	Integer
combinations/roadUsers/mmr/color/score	The confidence factor for the color result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/tags	An array of recognized vehicle traits.	Array of objects
combinations/roadUsers/mmr/tags/name	The name of the trait, e.g., "caravan", "ambulance", ... For the full list of possible tags and their definition, check the Eyedea MMR SDK documentation.	String
combinations/roadUsers/mmr/tags/value	The value of the trait. Either "yes", or "no".	String
combinations/roadUsers/mmr/tags/id	ID of the trait value.	Integer
combinations/roadUsers/mmr/tags/score	The confidence factor for the tag value. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/input	The source object within the cluster used for the MMR computation. Contains either box, or plate object.	Object
combinations/roadUsers/mmr/input/box	The parameters of the bounding box used for the MMR computation.	Object

combinations/roadUsers/mmr/input/box/**topLeftCol**	The horizontal coordinate of the top-left bounding box corner within the image.	Decimal number
combinations/roadUsers/mmr/input/box/**topLeftRow**	The vertical coordinate of the top-left bounding box corner within the image.	Decimal number
combinations/roadUsers/mmr/input/box/**bottomRightCol**	The horizontal coordinate of the bottom-right bounding box corner within the image.	Decimal number
combinations/roadUsers/mmr/input/box/**bottomRightRow**	The vertical coordinate of the bottom-right bounding box corner within the image.	Decimal number
combinations/roadUsers/mmr/input/**plate**	The parameters of the plate used for the MMR computation.	Object
combinations/roadUsers/mmr/input/plate/**index**	The 0-based index of detected plate used for the MMR computation.	Integer
combinations/roadUsers/mmr/input/plate/**centerCol**	The horizontal coordinate of the plate center within the image.	Decimal number
combinations/roadUsers/mmr/input/plate/**centerRow**	The vertical coordinate of the plate center within the image.	Decimal number
combinations/roadUsers/mmr/input/plate/**rotation**	Clockwise correction of the source image in degrees.	Decimal number
combinations/roadUsers/mmr/input/plate/**pixelsPerMeter**	The real-world scale in pixels per meter of the image computed on the plate. For example, plate of width = 300 millimeters, photographed to have 150 pixels, has 150 / 0.3 = 500 pixels per meter scale.	Decimal number
engines	An array containing the detection and recognition results.	Array of objects
engines/**task**	The computation task processed by the SDK engine. Either "DETECTION", "OCR", or "MMR".	String
engines/**moduleId**	SDK engine module ID.	Integer
engines/**moduleName**	SDK engine module name.	String
engines/**moduleVersion**	SDK engine module version.	String