AN14718

USB Camera Object Detection Pipeline on i.MX RT700

Rev. 2.0 — 10 November 2025 Application note

Document information

Information	Content
Keywords	AN14718, USB camera, object detection, i.MX RT700, single-core, dual-core
Abstract	This application note describes the process to build an object detection pipeline on the NXP i.MX RT700.

1 Introduction

The i.MX RT700 crossover microcontroller offers a powerful and flexible platform for implementing edge AI applications, including real-time object detection. It is well suited for low-power and high-performance embedded vision tasks with its multi-core architecture, integrated neural processing unit (NPU), advanced graphics, and multimedia subsystems. This application note describes the process to build an object detection pipeline on the NXP i.MX RT700. It begins with a single-core implementation using only CPU0 in the compute domain. It introduces a dual-core optimization that uses CPU1 in the sense domain to offload image capture and preprocessing tasks. This evolution demonstrates how the heterogeneous architecture of i.MX RT700 can be harnessed for scalable performance in embedded vision applications.

2 Hardware and software requirements

To run the applications mentioned in this application note, the following hardware and software are required:

- MIMXRT700-EVK
- RK055HDMIPI4MA0 Display
- USB camera - Logitech C920
- USB-A to micro-USB adapter
- 5 V external power supply for the MIMXRT700-EVK
- MCUXpresso IDE v24.12+

The complete setup is shown in Figure 1.

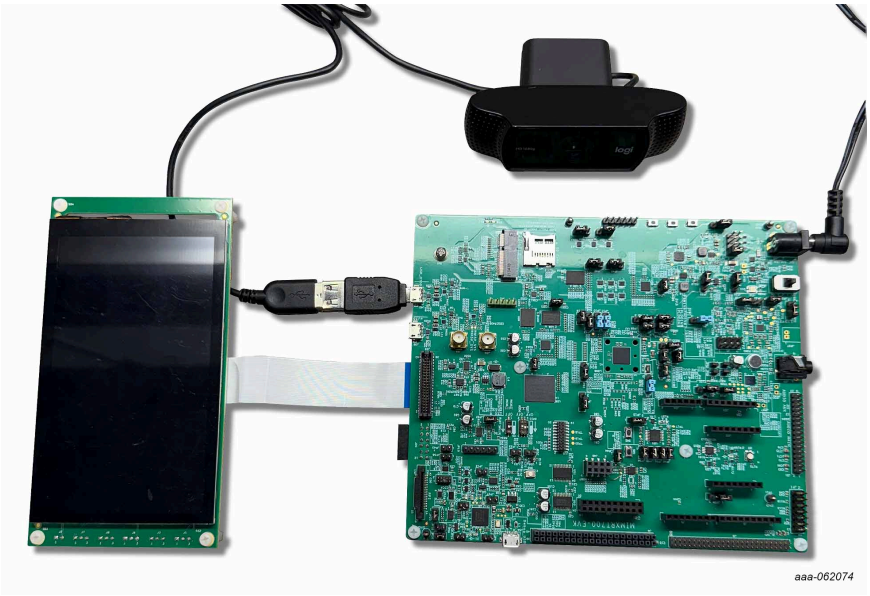

Figure 1. Hardware setup

3 Application context and motivation

The object detection pipeline is designed for a real-time interaction monitoring system that detects and classifies the items that the user is handling. Specifically, the system determines whether an object is being placed into or removed from a designated area such as a shelf, bin, or workstation. This functionality is critical in various use cases, including:

- **Retail automation**: for example, smart checkout or inventory tracking

- **Industrial safety and compliance**: for example, tool usage monitoring
- **Smart storage systems**: for example, automated supply cabinets

To monitor the user and the surrounding environment continuously, the system uses a USB camera. The captured video frames are processed in real time to detect objects and track their movement relative to a predefined region of interest. By analyzing the sequence and direction of motion, the system can infer whether an item is being added or removed.

4 i.MX RT700 Overview

The i.MX RT700 features a heterogeneous architecture with the following:

- Dual Arm Cortex-M33 cores
- Hi-Fi digital signal processor (DSP)
- A suite of hardware accelerators, which include the NPU, vector graphics processing unit (VGPU), and JPEG decoder (JPEGDEC)

These components are distributed across multiple power and functional domains, enabling fine-grained control over performance and energy efficiency.

4.1 NPU

The NPU is a dedicated hardware accelerator designed to execute deep learning inference tasks efficiently. It is located in the compute domain alongside CPU0 and is tightly integrated with the system memory and direct memory access (DMA) infrastructure. The NPU supports 8-bit and 16-bit integer operations. It is optimized for convolutional neural networks (CNN), which are commonly used in object detection, classification, and segmentation tasks. The NPU operates independently of the CPU, allows CPU0 to offload compute-intensive inference tasks and focus on control and postprocessing logic. This separation of concerns significantly improves system responsiveness and throughput. The CPU0 loads the model weights and input data to the SRAM or PSRAM, sets up the NPU registers, and triggers execution. Upon completion, the NPU writes the output to a designated memory region, which CPU0 then reads for further processing.

4.2 VGPU

The VGPU is a 2.5D graphics accelerator designed to handle vector graphics rendering and image manipulation tasks. It supports the OpenVG 1.1 API and the Vivante VGLite API. The VGPU supports tasks, such as image scaling, rotation, and blending, which makes it ideal for preprocessing images before feeding them into a neural network. The VGPU architecture is composed of the following functional blocks:

- **Host interface**: manages communication with the CPU and external memory, including clock domain crossing.
- **Memory controller**: handles memory requests and manages internal buffers.
- **Graphics pipeline front end**: accepts high-level drawing commands and primitives.
- **Tessellation engine**: converts vector paths into geometric primitives.
- **Rasterizer**: transforms primitives into pixel data.
- **Imaging engine**: applies color fills, gradients, and textures.
- **Pixel engine**: performs final pixel-level operations such as filtering and compositing.

In the object detection pipeline, the VGPU resizes high-resolution images from the USB camera to a fixed 320x320 resolution, which the object detection model requires. The offloading of this task to the VGPU, accelerates the pipeline, reduces the CPU load, and reduces the CPU power consumption.

4.3 JPEGDEC

The JPEGDEC is a high-performance hardware that supports both still image and motion JPEG decompression. It is composed of two submodules: the JPEGDEC core and the JPEG decoder wrapper (JPGDECWRP). The JPEGDEC handles entropy decoding, inverse discrete cosine transform (IDCT), and color space conversion. The JPGDECWRP manages the DMA transfers and context switching. The JPEGDEC supports a wide range of image formats, including YUV444, YUV420, YUV422, RGB, ARGB, and grayscale, with 8-bit or 12-bit pixel precision. It is compliant with the ISO/IEC 10918-1 JPEG standard and can decode multiple bitstreams using a context-switching mechanism. This makes it suitable for applications that require decoding multiple image sources or video streams. The JPEGDEC operates in the following modes:

- **Single bit stream and single frame**: decodes one frame at a time.
- **Single bit stream repeat**: continuously decodes frames from a single stream.
- **Context switch**: switches between multiple streams or frame buffers.

The JPEGDEC module is configured to decode images received from the USB camera. The decoded output is stored in PSRAM and later the VGPU resizes the images. The use of JPEGDEC offloads the computationally intensive decoding process from the CPU and enables the real-time performance even with high-resolution input.

5 System overview

This section outlines the software architecture and runtime configuration of the object detection pipeline implemented on the i.MX RT700. It covers system-level considerations such as clock configuration, memory layout, peripheral integration, and model deployment.

5.1 Configuration

The following frequencies are selected to optimize for performance:

- CPU0 - 325 MHz
- CPU1 - 250 MHz
- VGPU - 325 MHz

The memory layout is planned carefully due to the limited access of the NPU to 5.5 MB of SRAM (0x2000 0000 - 0x2057 FFFF). To optimize performance:

- The full-size RGB frame from the USB camera is always stored in an external PSRAM, which offers more space but slower access.
- The resized 320x320 RGB frame is used as an input to the NPU. It is placed in an on-chip SRAM for faster access and minimal latency during inference.

This layout ensures that the NPU can operate efficiently without memory access bottlenecks, while still accommodating high-resolution input frames.

5.2 USB camera capture

The USB camera interface in this project is based on the FreeRTOS USB host video camera example. It demonstrates how to capture the JPEG images from a USB camera and store them on a microSD card. This example provides a solid foundation for the USB host stack initialization, USB video class (UVC) device enumeration, and basic JPEG frame capture.

To enable the real-time processing, the project is extended to include the hardware-accelerated JPEG decoding using the JPEGDEC. The JPEG frames from the USB camera are decoded directly into YUV format. The

JPEGDEC operates in the Streaming mode and writes the decoded frames to PSRAM or SRAM, which depends on the memory layout.

After decoding, convert the YUV422 image data to RGB format for the following two purposes:

1. Feeding to the neural network, which expects the RGB input.
2. Displaying the frame on a connected screen.

Also, added support for accelerated blitting and color space conversion using the VGPU core. This allows YUV422-to-RGB conversion and the hardware copies the framebuffer.

Once the frame is converted to RGB, the application uses the VG-Lite API to configure the VGPU for resizing. The original image in PSRAM is read, resized to 320x320, and the result is stored in an on-chip SRAM. This hardware-accelerated resizing is faster than software-based methods.

5.3 FastestDet model

The FastestDet model is selected for this application due to its excellent balance of accuracy, speed, and compact size. In comparison to the larger models like YOLOv5 or YOLOv7, the FastestDet offers competitive detection performance while it is smaller and faster.

After training on a custom dataset, the model is converted to TFLite format. It is fully quantized to INT8. The eIQ Toolkit optimizes it for the NPU. The final model size is 600 kB. The small size allows it to fit comfortably within the memory constraints of the i.MX RT700 while still delivering fast and accurate object detection.

6 Baseline: single-core object detection pipeline

In the initial implementation, the entire object detection pipeline is executed on CPU0, the Cortex-M33 core located in the compute domain. This pipeline includes the following:

- USB camera interfacing
- JPEG decoding
- Image resizing using the VGPU
- Model inference using the NPU
- Postprocessing of the inference results

Figure 2 shows the general flow of the pipeline.

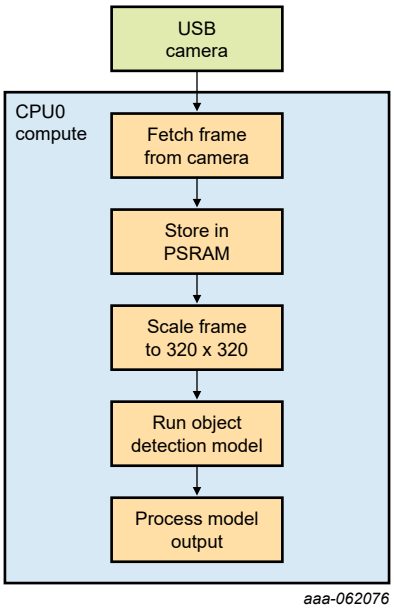

Figure 2. Single-core pipeline

6.1 Running the single-core application

To run the single-core application, perform the following steps:

1. Connect the camera, display, and power supply as shown in Figure 1.
2. Position the camera as shown in Figure 3.

Figure 3. Camera positioning

3. Connect the “Debug USB” (J45) on the MIMXRT700-EVK to your computer.
4. In the MCUXpresso, select **Open Projects from File System…** from the **File** menu.

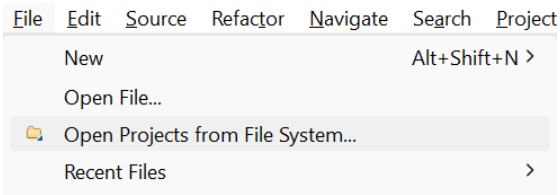

Figure 4. Open project from file system

5. Select the root project directory and import the single core project as shown in Figure 5.

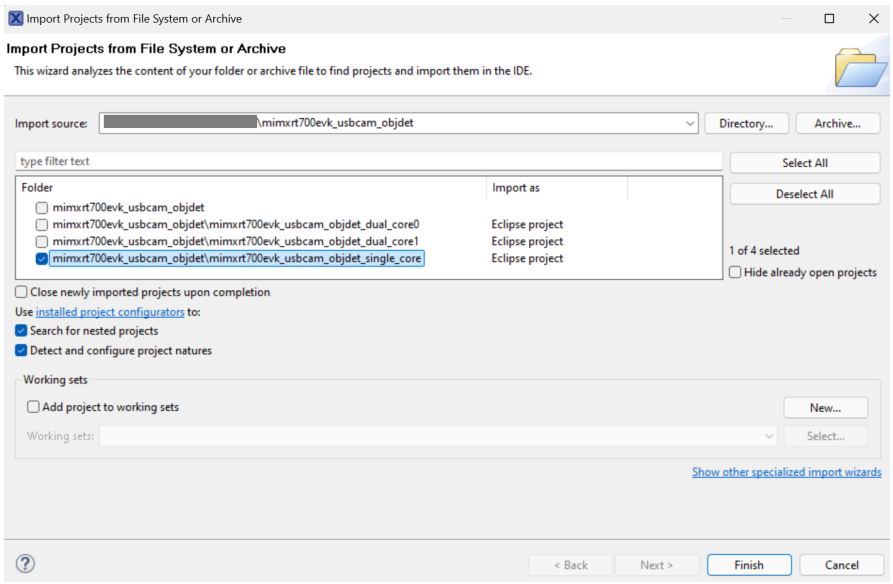

Figure 5. Select a single core project and import

6. Build and flash the application on the board as shown in Figure 6.

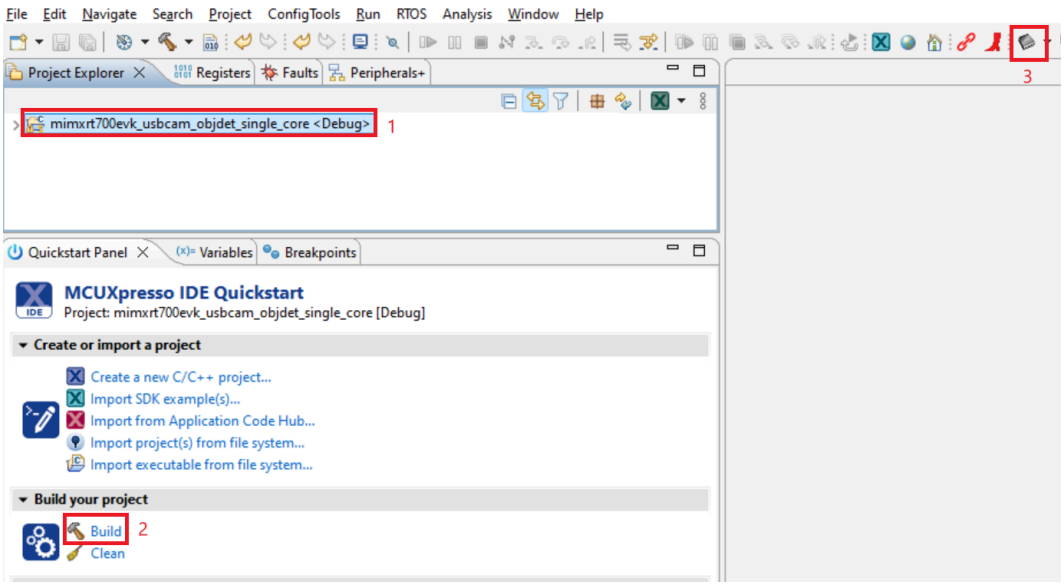

Figure 6. Build and flash the single core project

7. Figure 7 shows the application example output when it is running.

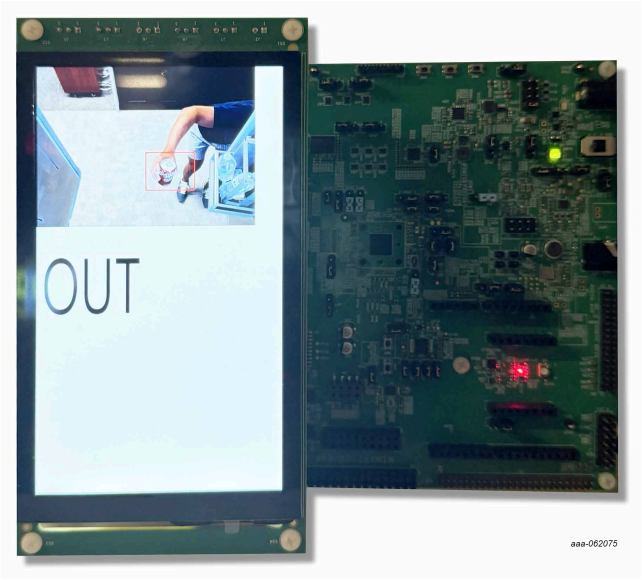

Figure 7. Single-core application example output

8. To profile the pipeline execution time, you can attach an oscilloscope to J21[1] (PIO0_21). This pin goes high when we receive a frame from the USB camera and goes low after processing the model output.

6.2 Performance analysis: bottlenecks and inefficiencies

Even with hardware operating at maximum frequencies and using the NPU, VGPU, and JPEGDEC accelerators, the system achieved an effective frame rate of 7.5 FPS only. For more details, see Table 1.

This throughput underscores a fundamental limitation of the single-core architecture: the inherently sequential processing pipeline. As CPU0 acquires and processes the image frames, it must wait for the NPU to complete the inference on the current frame before it captures and preprocesses the next one. This wait introduces the idle time between frames and reduces the effective frame rate of the system. Moreover, the USB camera interface is bandwidth-intensive and requires frequent CPU intervention to manage USB transactions, buffer handling, and DMA coordination. These tasks consume a significant portion of the processing bandwidth of the CPU. As a result, the USB-related interrupts and processing can preempt or delay the NPU configuration and inference routines, leading to jitter and inconsistent inference timing. This contention between real-time data acquisition and compute-heavy inference creates a bottleneck that limits the scalability and responsiveness of the system.

Table 1. Single core performance measurements

Name	Description	Time
Image conversion/ processing	Gather an image from the camera, convert to RGB, and scale resolution to model input size 320x320	~10 ms
ML preprocessing	Normalize scaled image	~12 ms
ML inference	Execute model	~80 ms
ML postprocessing	Process model output to compute bounding box location	~20 ms
Image display	Display camera image with bounding boxes on the LCD	~12 ms
Total		~134 ms (~7.5 FPS)

7 Optimization: dual-core pipeline

To address the limitations of single-core and optimize the pipeline, the CPU1 and the Cortex-M33 core in the sense domain are introduced. The USB camera interface, JPEG decoding, and image resizing tasks are offloaded to CPU1. The system can prefetch and preprocess the next frame while CPU0 and the NPU are still working on the current one. This enables the true pipelining of the object detection process, where image acquisition and inference occur in parallel. This architectural shift not only improves throughput and reduces latency but also isolates time-sensitive inference tasks from interrupt-heavy I/O operations. It results in more deterministic and reliable performance. The updated pipeline is shown in Figure 8.

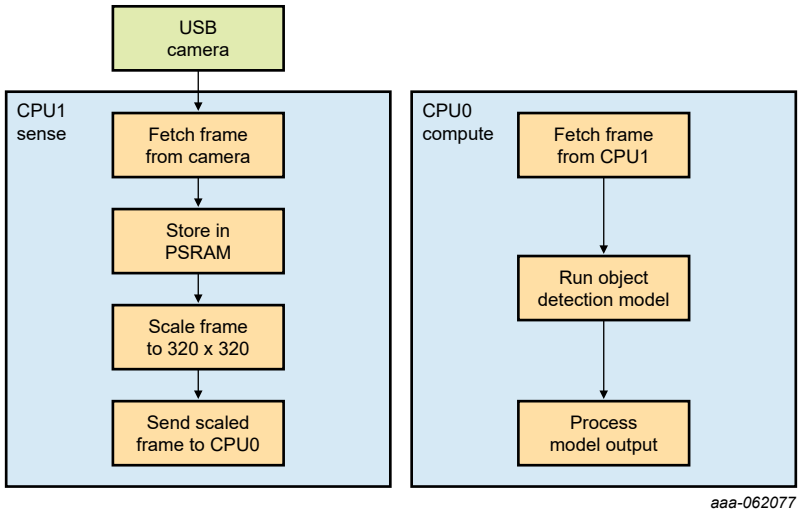

Figure 8. Dual-core pipeline

7.1 CPU1 responsibilities

The CPU1 handles the following responsibilities:

- Manages the USB camera interface
- Decodes the JPEG images using the JPEGDEC
- Resizes the images using the VGPU
- Stores the resized image in SRAM
- Sends the metadata and pointers to the CPU0 via RPMsg-Lite

7.2 CPU0 responsibilities

The CPU0 handles the following responsibilities:

- Configures and manages the NPU
- Runs the frames received from the CPU1 through the model
- Retrieves and processes the model output to extract the bounding boxes

7.3 Running the dual-core application

To run the dual-core application, perform the following steps:

1. Perform the steps 1 to 4 of Section 6.1.
2. Select the root project directory and import both dual-core projects as shown in Figure 9.

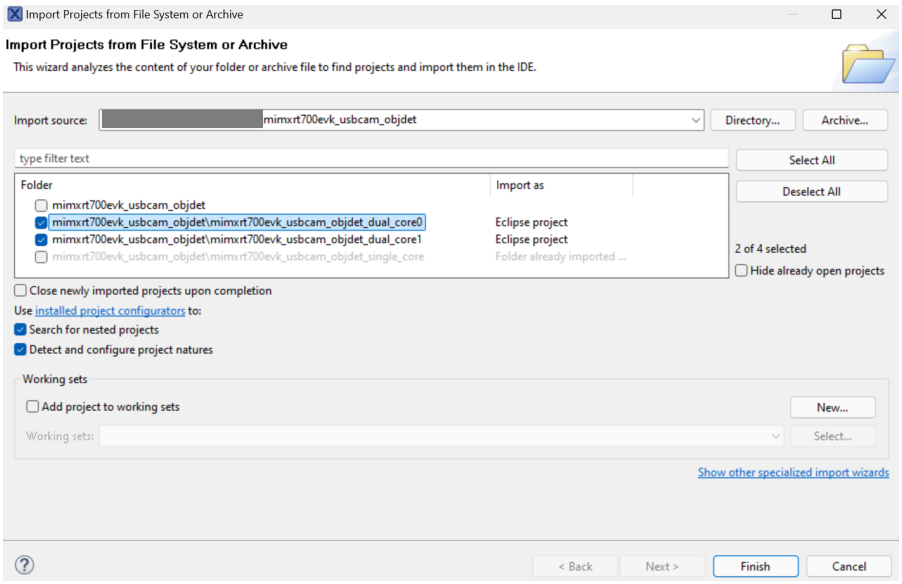

Figure 9. Select both dual-core projects and import

3. Select the **mimxrt700evk_usbcam_objdet_dual_core0** project and build it. This also builds the **mimxrt700evk_usbcam_objdet_dual_core1** project since it is linked in the project settings.

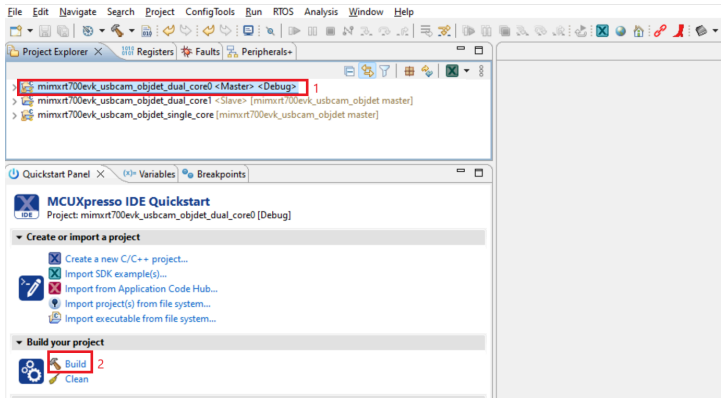

Figure 10. Select a core0 project and build

4. Flash the application on the board.

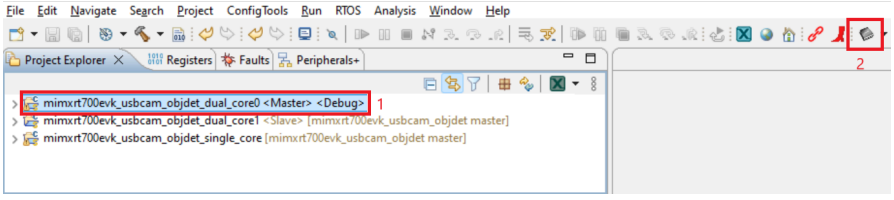

Figure 11. Flash dual-core application

5. This application does not use the display. To verify the proper operation, attach an oscilloscope to J21[1] (PIO0_21). This pin profiles the pipeline execution time. It goes high when CPU0 fetches a frame from CPU1 and goes low after CPU0 processes the model output. It looks like as shown in Figure 12.

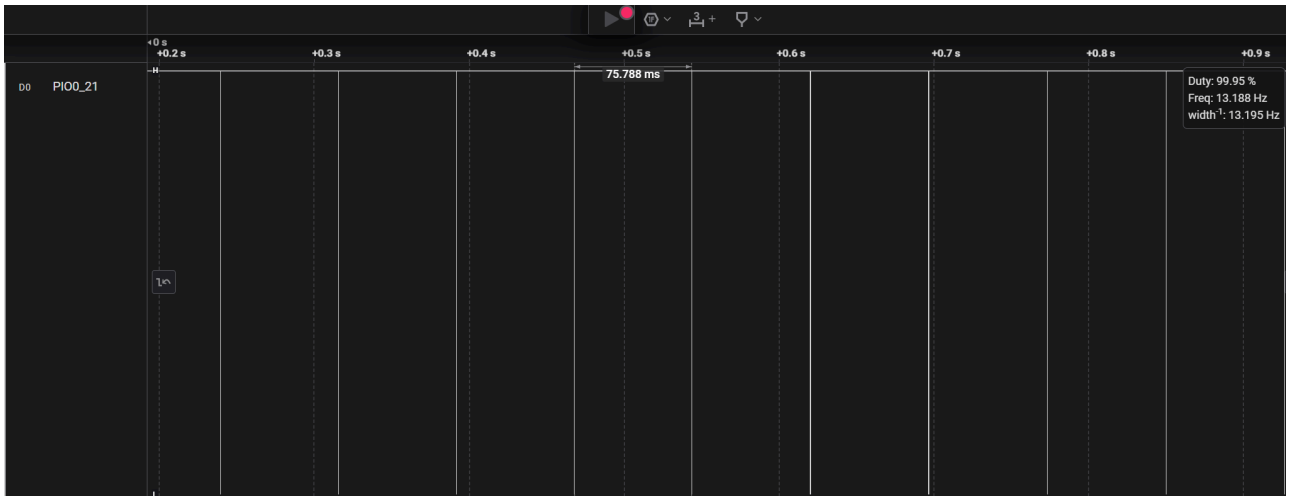

Figure 12. Dual-core application execution time profiling

7.4 Performance analysis

Table 2 shows the transition from a single-core to a dual-core architecture, which yields performance improvements:

- **Image capture and preprocessing**: improved from 22 ms to 2 ms, a 10 x speed up. It offloads the USB handling and image scaling to CPU1.
- **Model inference time**: dropped from 80 ms to 60 ms, as USB-related interrupts no longer interferes with the NPU execution on CPU0.
- Even when excludes the display rendering from the single-core pipeline, total processing time improves from 122 ms to 76 ms. It boosts the throughput from ~8.2 FPS to ~13 FPS.

Table 2. Dual-core performance measurements

Name	Description	Time
Get frame	Wait to receive a frame from core1 and copy to tensor arena	~2 ms
ML inference	Execute model	~60 ms
ML postprocessing	Process model output to extract bounding box locations	~14 ms
Total		~76 ms (~13 FPS)

8 Conclusion

The i.MX RT700 enables the developers to build sophisticated and real-time object detection systems with a high degree of flexibility and performance. The developers can create the scalable solutions with its dual-core architecture and hardware accelerators. The solutions, which meet the demands of modern embedded vision applications from smart cameras and industrial automation to consumer electronics and beyond.

9 Abbreviations

Table 3 lists the acronyms used in this document.

Table 3. Abbreviations

Acronym	Description
API	Application programming interface
CNN	Convolutional neural networks
CPU	Central processing unit
DMA	Direct memory access
DSP	Digital signal processor
IDCT	Inverse discrete cosine transform
JPEGDEC	JPEG decoder
JPGDECWRP	JPEG decoder wrapper
NPU	Neural processing unit
PSRAM	Pseudo static random-access memory
ROM	Read-only memory
SRAM	Static random-access memory
UVC	USB video class
VGPU	Vector graphics processing unit

10 Revision history

Table 4 summarizes the revisions to this document.

Table 4. Revision history

Document ID	Release date	Description
AN14718 v.2.0	10 November 2025	Initial public release
AN14718 v.1.0	20 August 2025	Initial internal release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents