Allied Vision

Application Note

Considerations for Setting Up USB Vision Systems

Document V1.0.0
May 2022

Scope of this document

This document provides information on how to set up a computer vision system with optimal performance using USB3 Vision devices. This includes some special considerations for the use of multiple cameras with one host machine.

Introduction to U3V

USB3 Vision (U3V) is an interface standard on top of the USB standard for industrial cameras. As USB has become the de-facto standard for peripheral devices, this makes it easy to connect cameras to most systems.

USB3.0 introduced the "SuperSpeed" (SS) transfer rate. With a theoretical transfer speed of up to 625 MByte/s, SuperSpeed transfers can deliver large amounts of data in a short time, making it suitable for many vision applications. This upper limit for the bandwidth given by the specification is an idealized theoretical value. For real world applications, the actual achievable transfer rate of the host controller should either be verified in the datasheet of the host controller or be determined by measurements. For the sake of simplicity, this document will consider SuperSpeed connections to be capable of delivering 625 MByte/s.

As the USB protocol is not specifically designed for the needs of computer vision devices, some limitations can lead to performance and stability issues: Not the entire theoretical maximum bandwidth of 625 MByte/s can be used for the transfer of image data. In addition to some overhead that needs to be transferred for each frame, the communication procedure of USB is host-initiated. Data is transferred from a USB device to the host system in the form of transfers with a size set by the host. This means the host system actively requests the transfer of data from a device in blocks of specified sizes. Without this initiation by the host, the device cannot submit data. If the host is overloaded and does not allow USB devices to transmit their data, they are unable to send it. Additionally, as USB is a serial protocol, only a single device at a time may use the bus to transmit data. These limitations may lead to issues further described in section Special considerations for Multi Camera Setups.

USB Vision systems

For the purposes of this document, a USB Vision system consists of one host system with one or more U3V devices connected via USB cables. The connections to the host controllers are all assumed to support at least the SuperSpeed transfer rate. For the host to support SuperSpeed transfers, a suitable host controller must be installed in the system. This can typically be either an integrated host controller on the motherboard or an additional PCIe card. When setting up a USB vision system, it is very important to verify that the host controller can support the required bandwidth for the number of connected devices. For certain cameras (e.g. large resolution or high-fps devices), the required bandwidth may be as high as the full 625 MByte/s SuperSpeed transfer rate. We recommend to always leave some headroom in the bandwidth of a connection to U3V devices to allow for required overhead in the communication that does not directly transfer image data. Where possible, it is recommended to set up the system in such a way that connections utilize the SuperSpeed+ transfer rate of 1250 MByte/s, which was introduced with the USB standard 3.1.

When setting up a host system with PCIe host controllers, the bandwidth limitations of the PCIe connection on the motherboard needs to be considered. Table 1 shows an overview of the supported bandwidth for

different PCIe versions and the number of lanes. Not all version/lane configurations meet the bandwidth recommendation for SuperSpeed connections.

Table 1: Overview of PCIe bandwidths

Lanes ↓	Approximated Bandwidth		
PCIe versions →	1.0/1.1	2.0/2.1	3.0/3.1
x1	250 MByte/s	500 MByte/s	1000 MByte/s
x2	500 MByte/s	1000 MByte/s	2000 MByte/s
x4	1000 MByte/s	2000 MByte/s	4000 MByte/s

Besides this limitation of the bandwidth available via the PCIe port, some USB host controller cards might also not be capable of supporting the required USB bandwidth when all ports of the card are used. Keep this in mind when connecting multiple U3V devices to the same host controller.

Single camera setup

In the simplest case, a single U3V device is connected to the host machine:

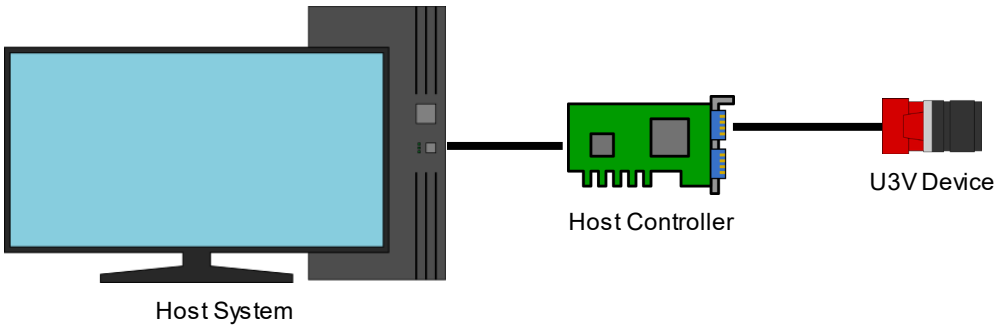

Figure 1: Connecting a single U3V device to a host controller

In this simple case, a single USB connection is set up to one U3V device. This allows the device to utilize the entire bandwidth of the connection.

Provided the host controller can support the required bandwidth, this situation generally provides a stable streaming experience. It can however be impacted if the host system cannot process the received data stream fast enough. Depending on the operations performed on each acquired image, it is possible that the host system is too busy analyzing the acquired data to initiate new transmissions from the U3V device. For USB devices this issue is especially critical because the communication must always be initiated by the host. Keeping the system load in consideration is therefore always required and it should be ensured that the system has some headroom instead of running at full capacity.

Special considerations for Multi Camera Setups

Connecting multiple U3V devices to the same host requires some consideration. While in the simplest case a direct connection from each device to the host controller is the obvious choice (see Figure 2), many multi camera setups require additional USB ports. We recommend adding them as PCIe cards (see Figure 3).

Multiple cameras connected to the same host controllers

Some host controllers can support multiple SuperSpeed buses simultaneously. In this case, directly connecting multiple U3V devices to a single host controller is the best approach to set up a vision system with multiple devices. This way, each U3V device may utilize the full SuperSpeed transfer rate.

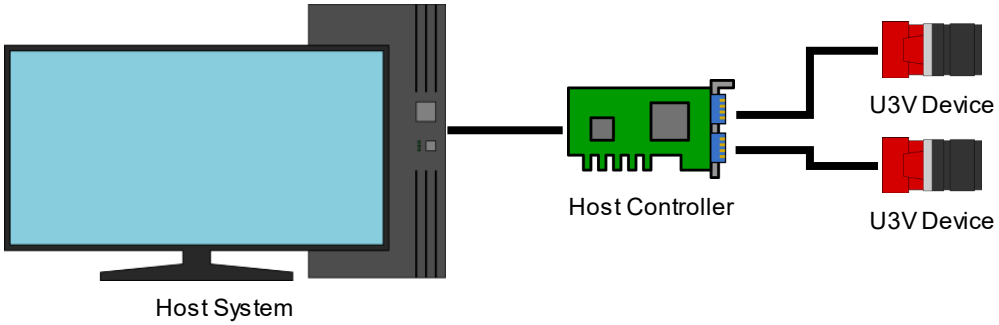

Figure 2: Connecting multiple U3V devices to the same host controller

The resulting data stream from the devices still shares a connection (e.g. PCIe bus) from the host controller to the system's CPU for processing, but as these connections can provide higher bandwidth than a SuperSpeed connection (see e.g. Table 1), this is preferred over e.g. using a USB hub. Check the datasheet of the used host controller to verify if it supports multiple independent SuperSpeed buses or root hubs at their full bandwidth.

Tip

Ensure that the used host controller can support SuperSpeed connections to all connected devices at the same time.

Adding more host controllers to the system

In some cases, it is possible that a host controller is not capable of supplying the required SuperSpeed bandwidth for all ports at the same time. If so, it is generally advisable to add more host controllers to the system. This can for example be done via PCIe expansion cards. It is important to keep in mind though, that the host system itself must be capable of processing the incoming data quick enough without overloading the system (ensure CPU load is within reasonable limits).

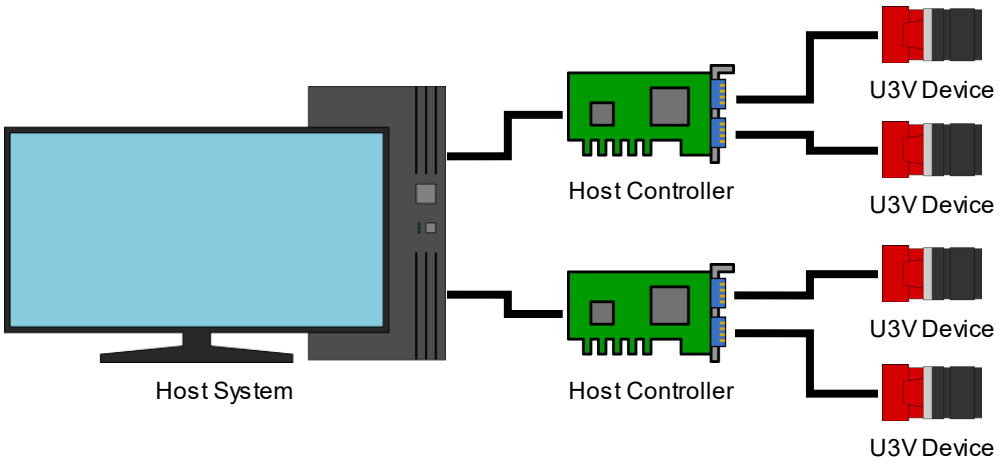

Figure 3: Adding host controllers to support many U3V devices

Using USB Hubs (not recommended)

The USB protocol also supports connecting multiple devices to a single port on the host controller via USB hubs. For devices with a high bandwidth requirement like U3V devices this is not recommended as the bandwidth that is available for each device is strongly limited and the traffic needs to be processed by the hub instead of being directly transferred to the host controller.

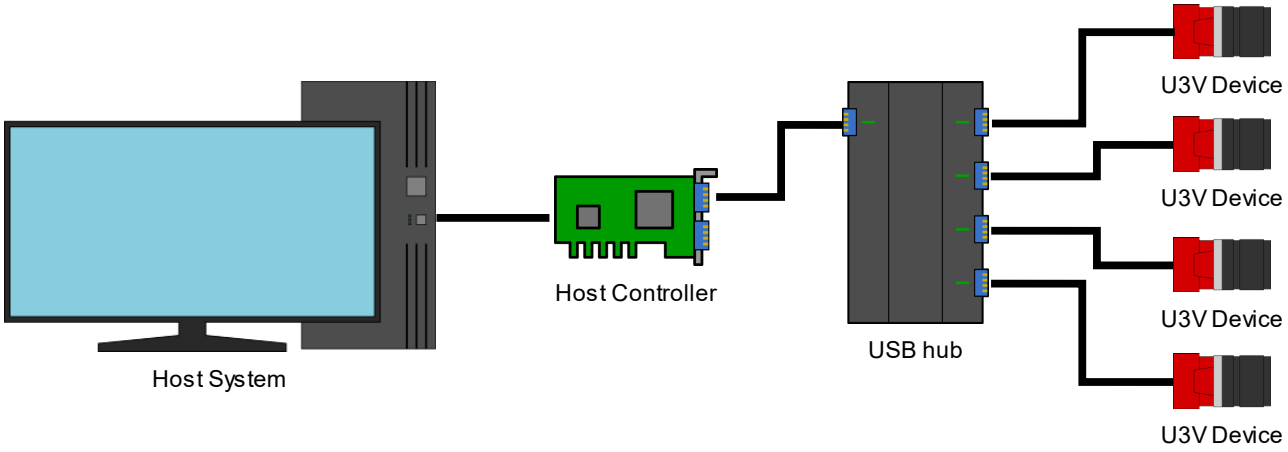

*Figure 4: Using a USB hub to connect many U3V devices (**not recommended – see Figure 3 instead**)*

In Figure 4 all U3V devices are connected to a single USB hub, which is connected to the host controller. The connection from the host controller to the USB hub must carry the data from all connected U3V devices. Since the connection is limited to a USB transfer rate (for simplicity, here a 625 MByte/s SuperSpeed connection), each U3V device cannot rely on receiving a full SuperSpeed transfer rate. Instead, the devices need to share the bandwidth of that single connection from the hub to the host controller leading to a bottleneck. In such a configuration, the data transfer rate of the U3V devices must be drastically limited to not overload the USB connection. While the devices generally provide several settings to adjust the required bandwidth (see section How to adjust required bandwidth), this configuration may still run into issues even if appropriate bandwidth limitations are set for each device.

Average vs. burst bandwidth use

As a first approximation on the required bandwidth of a U3V device, a constant number seems reasonable. A number of frames of a given size needs to be transferred in a given time. Transferring, for example, 20 frames per second with a resolution of 2464 × 2064 pixels and pixel format Mono8 (8 bits per pixel), requires a bandwidth of:

$$1\frac{\text{byte}}{\text{pixel}} \cdot (2464 \cdot 2064)\frac{\text{pixel}}{\text{frame}} \cdot 20\frac{\text{frames}}{\text{second}} \approx 102\frac{\text{MByte}}{\text{second}}$$

This calculation only includes the payload size. To consider the overhead of USB communication as well as leave some headroom for other control channel communication (e.g. reading and writing camera features), the value is increased by 20% for the sake of this example. This results in an estimated required bandwidth of 120 MByte/s to use the device at the given configuration. This estimate makes it seem like it should be possible for up to four devices to share a single SuperSpeed connection via a USB hub. If multiple devices were able to send data at this reduced bandwidth simultaneously, transmission would work as expected.

But since USB is a serial communication protocol, this simultaneous data transfer of multiple devices via the same connection is not possible. A USB hub is needed to connect multiple USB devices to the same port on a host controller (see Figure 4). If multiple U3V devices record frames simultaneously (for example, if multiple devices are triggered by the same hardware signal), they will not be able to transfer the acquired frames immediately. Instead, the host must initiate transfers for each device. The corresponding device will then send the requested amount of data using all available bandwidth as quickly as possible in a burst. After this transmission, it must hold all remaining data it was not able to send yet in some buffer. If the device does not have an internal buffer big enough to store the remaining data, it is possible that the transmission fails because the host does not retrieve the frame data quick enough from the device before the next frame is acquired and overwrites the existing data in memory.

If instead the number of devices is split up over multiple host controllers (see Figure 3), the frame transfer becomes much easier. Each host controller needs to support fewer devices and transfers from devices to different host controllers can take place in parallel. This allows for a larger number of devices to work reliably, improving the stability of the entire system.

How to adjust required bandwidth

Several approaches are possible to control how much bandwidth is required to stream image data from a U3V device. Some of these can be adjusted as camera settings, others change the format of transferred data.

DeviceLinkThroughputLimit

U3V devices provide the feature *DeviceLinkThroughputLimit*. It reduces the maximum bandwidth of the data streamed from the device. It directly influences other camera features like the acquisition frame rate. It is the simplest method of directly adjusting the required bandwidth of a U3V device.

By default, the camera may be set to a low value, to allow for very reliable data transmission. For higher performance, increasing the value of this feature allows the camera to use more bandwidth for its data transfer. It is however not recommended to always set this feature to its maximum value. Instead, some headroom should be left in the bandwidth of the USB connection. Typically, around 350000000 Byte/second (350 Mbyte/s) is a good starting point to find a suitable value for the used system setup.

Adjusting the frame rate

Transmitting more frames per second results in a higher bandwidth requirement. Limiting the number of frames acquired by the device is possible by enabling the *AcquisitionFrameRate* camera feature and setting a fixed value for it, or using hardware triggering to record frames. However, keep in mind, that not only the number of frames acquired but their respective timing can also impact stream stability (see section Average vs. burst bandwidth use).

Selecting a pixel format

The pixel format affects the required bandwidth. Some pixel formats provide multiple channels to transmit color data (e.g. RGB8 vs. Mono8) or they provide a higher bit-depth (e.g. Mono16 vs. Mono8). For some pixel formats and camera models, it is possible to transmit the image data in a compact format to the host system. Two examples where this is possible are color images, or images with bit-depths that are not aligned to 8 bit boundaries.

Calculating colors on the host

Devices supporting color formats such as RGB8 generally record the color information by utilizing a bayer pattern on the sensor. If the camera transmits the acquired image as an RGB image to the host, it first debayers the acquired sensor data to determine the intensity for the R, G, and B channels of the image and transmits that data. This takes (for an 8-bit image) 3 channels × 1 byte/channel = 3 byte of data per pixel. Alternatively, it is also possible to transmit the non-debayered sensor data and perform the debayering on the host. This greatly reduces the amount of data that needs to be transferred from device to host, but requires additional system resources on the host for debayering.

Packing high bit-depth data

Another case where bandwidth may be saved is when recording images with higher bit-depths. Some U3V devices support recording pixel intensity with a bit-depth between 9 and 15 bit (i.e. not aligned to 8 bit boundaries). For these bit-depths, cameras may provide multiple pixel formats. In the case of e.g. 10 bit resolution, these would be named Mono10 or Mono10p. The names of these formats follow the Pixel Format Naming Convention (PFNC). In the Mono10 format, the intensity for each pixel is transmitted as a 16 bit field, where the unused bits are zero-padded. By contrast, the packed equivalent Mono10p does not pad the transmitted data. This reduces the amount of data that needs to be transferred, as the 0 bits that are used for padding no longer need to be transmitted for every pixel. As further image processing likely requires pixel data to properly align with 8 bit boundaries, unpacking of the frame data might be necessary on the host system, requiring additional resources.

Our Vimba SDK contains the *Image Transform Library*, which transforms images received via Vimba APIs into common packed and unpacked image formats. You can choose between several debayering algorithms.

General tips

Some settings for our U3V devices are by default set to conservative values. Adjusting them for optimal performance on your specific system can lead to better performance than the default values. Some available settings have already been described in section How to adjust required bandwidth. The following sections describe additional settings that may be adjusted to improve performance on some systems.

USB Transport Layer MaxTransferSize and MaxTransferCount

In USB communication, the host can specify the size of each individual transfer. By choosing a larger transfer size, a bigger payload can be submitted with each transfer. This reduces the overhead associated with the frame transmission and allows a U3V device to write out larger amounts of data at a time instead of having to split it up into many small transfers. Depending on the operating system and hardware, it is even possible to select a transfer size that is large enough to transmit the payload for an entire frame in a single transfer.

The *MaxTransferSize* setting of the USB Transport Layer is defined in the VimbaUSBTL.xml configuration file. It controls the upper boundary for the size of USB transfers. Increasing its value can improve the stream stability, but the results strongly depend on the used system. On some systems it may be possible to set a *MaxTransferSize* value large enough to submit the entire image data in one USB transfer. The size of the data that is being transmitted can be read from the devices *PayloadSize* feature.

The setting *MaxTransferCount* affects the number of transfer requests handed over to the OS at a time. This queue of requests serves as buffer not only for the currently ongoing data transfer, but also to bridge times when the OS switches to other processes in a multi-tasking environment. In the general case, the default value for this setting is a good choice, but in a scenario with a weak system like a small embedded board and a high number of active threads or tasks, an increased number of queued transfers can contribute to a more stable stream and fewer incomplete frames.

Increasing USBFS buffer size (Linux only)

On most modern Linux systems, support for USB devices is implemented via usbcore and made available to the user via USBFS. By default, the memory space reserved for USB operations is set to a low value (usually 16 MByte). When transferring larger amounts of data, it is first received with the help of memory from that small buffer, and if required moved to some other location in memory to make room for further transfers. As the size of this buffer is so small, it is not suitable for the large amounts of data that are being transmitted by U3V devices. Increasing the USBFS memory buffer size is strongly recommended.

The current size of the USBFS buffer can be read from the USBFS directly with the following command:

```
cat /sys/module/usbcore/parameters/usbfs_memory_mb
```

Temporarily Increasing the USBFS memory buffer size

Writing a different value to that file adjusts the size of the USBFS buffer until the next reboot of the system. The following command will set it to a value of 1024 MByte (`sudo` is used here to indicate that root privileges are required for this):

```
sudo sh -c 'echo 1024 > /sys/module/usbcore/parameters/usbfs_memory_mb'
```

Permanently Increasing the USBFS memory buffer size

Tip

Before changing the bootloader configuration, create a backup of your system. A faulty bootloader may prevent your system from booting. A fix might require an external boot media or reinstallation of the system. Changing the bootloader configuration is done at your own risk.

To increase the buffer size permanently, add the kernel parameter `usbcore.usbfs_memory_mb=1024` to the bootloader configuration. How to do this depends on the bootloader used by your system.

GRUB 2

1. Open `/etc/default/grub`
2. Replace: `GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"` (or other contents within the quotation marks depending on your system) with: `GRUB_CMDLINE_LINUX_DEFAULT="quiet splash usbcore.usbfs_memory_mb=1024"`
3. Update grub: `sudo update-grub`
4. Reboot the system.

Syslinux

1. Open `/boot/extlinux/extlinux.conf`
2. Add `usbcore.usbfs_memory_mb=1024` to the `APPEND` line
3. Reboot the system.

Other bootloaders

To configure additional kernel parameters of other bootloaders, please see the manual of your bootloader.

Cables

The widespread adoption of USB as an interface technology results in great availability of accessories like cables. Unfortunately, not all cables fulfill the high requirements of U3V devices. It must be ensured that the cable supports at least SuperSpeed connections. Screw-lockable USB cables are recommended to ensure a robust and reliable connection. Cable lengths should be kept as short as possible as longer cables can introduce more noise into the signal or may cause a voltage drop that can result in power issues for the U3V device.

Summary

USB3 Vision is a mature standard used in many productive vision systems. For simple setups, the plug and play experience of USB makes it very easy to try out devices and start development. When systems become more complex and require multiple cameras, things become more complicated because of the design of the USB technology itself. It must be considered how devices are connected to the host system to prevent bottlenecks and optimized device configurations might be necessary to allow a reliable data transfer. The large number of involved components, from operating system to drivers and hardware implementations of the host controllers, mean that different systems might show subtle differences in behavior and therefore require different settings to function in a stable manner. U3V provides several options to tweak a system setup for best performance. This makes U3V a great option to setup scalable vision systems from simple single camera solutions to very complex multi-camera applications. If you have further questions, please contact us.