Project acronym: LADIO Project number: 731970 Work package: Deliverable number and name: D1.1: RT Acquisition - Implementation of CamBox and MiniBox	Title: Real-time device data and metadata acquisition Work Package: WP1 Version: 1 Date: May 31, 2017 Author: Stian Z. Vrba
Type: [ ] Report [ ] Demonstrator, pilot, prototype [ ] Website, patent filings, videos, etc. [X] Other	Co-Author(s): To: Albert Gauthier, Project Officer
Status: [ ] Draft [ ] To be reviewed [ ] Proposal [X] Final / Released to EC	Confidentiality: [X] PU – Public [ ] CO – Confidential [ ] CL - Classified
Revision: Final	
Contents:Deliverable 1.1. CamBox/MiniBox source code and description of HW/SW system configuration.	

Table of Contents

Executive summary

This document describes the results of WP 1, Task 1. There are two main deliveries: CamBox and MiniBox.

A prototype CamBox has been built completely, together with a handmade casing. "HAL" software from the POPART project has been extended with new features. CamBox has been successfully tested in a controlled environment to acquire a sample data set. However, there remain critical problems, stemming from unforeseeable hardware issues, that must be resolved before CamBox is ready for use on the set by non-expert users.

MiniBox has been successfully tested as a concept in lab environment, but no casing has been built due to prohibitive costs.

Deliverable: CamBox

CamBox is designed to collect data in real-time from multiple acquisition devices and synchronize it to the master timecode reference provided by the main camera, expressed as timecode (TC). CamBox currently supports the following features:

- Recording of up to three video streams (one color 10-bit HD-SDI stream with audio; two 8-bit grayscale witness camera streams).
- Witness cameras shutter-synchronized with the main camera.

- Extraction of metadata embedded in ancillary packets of the SDI stream.
- Reading of lens encoder data (Arri's LDE and Cooke's i-data[1] formats).

Each camera stream is saved to a separate MOV file, while metadata is timestamped with timecode from the main camera and stored in a separate file in JSON format. Users can choose any video codec supported by FFmpeg, but the functionality has been verified only with 10-bit PRORES, 8- or 10-bit DNXHD and 8-bit H264.

To speed up post-production workflows, CamBox can be configured to automatically upload finished recordings and metadata to one or more FTP folders via any available network connection (Ethernet, WiFi or LTE).

Currently, only RED and Arri Alexa Studio cameras are supported, and shutter synchronization is performed in software. Due to limitations of USB2, recording rates of up to 25 fps are supported with the resolution of witness cameras set to 1280x720 (see section "Known issues").

Experience from POPART project has shown that standard (consumer-grade) plugs are not robust enough for on-set use, so the box provides connectivity mainly through LEMO connectors as follows.

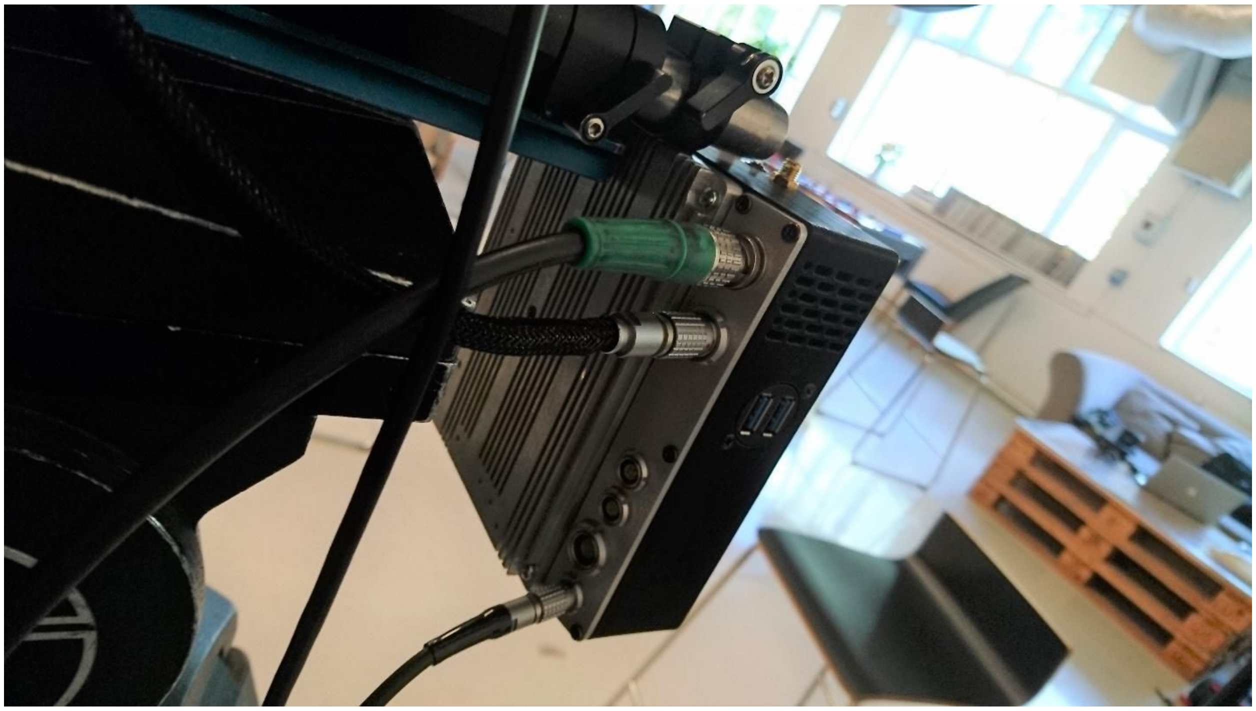
- One standard BNC connector for the main camera.
- One LEMO connector for power supply; 9-36 V. Power can be supplied by the main camera.
- One LEMO↔USB connector supporting up to two USB-2 witness cameras with a hardware trigger signal. Trigger wires are separate from USB wires, and operate at 3.3 V.
- One LEMO connector for reading the shutter sync signal sent by the main camera; operates at 3.3 V.
- One LEMO connector internally connected to two standard serial ports.[2]
- One LEMO to gigabit ethernet connector.[3]
- Two standard USB-3 ports for connecting other peripherals.
- Wireless network connectivity (4G modem, WiFi card - configurable as access point, Bluetooth by using an USB adapter).
- Red and green LED for *very* basic status indication (all ok – ready to record; error; recording on/off).

The external interfaces and casing are shown in the following image:

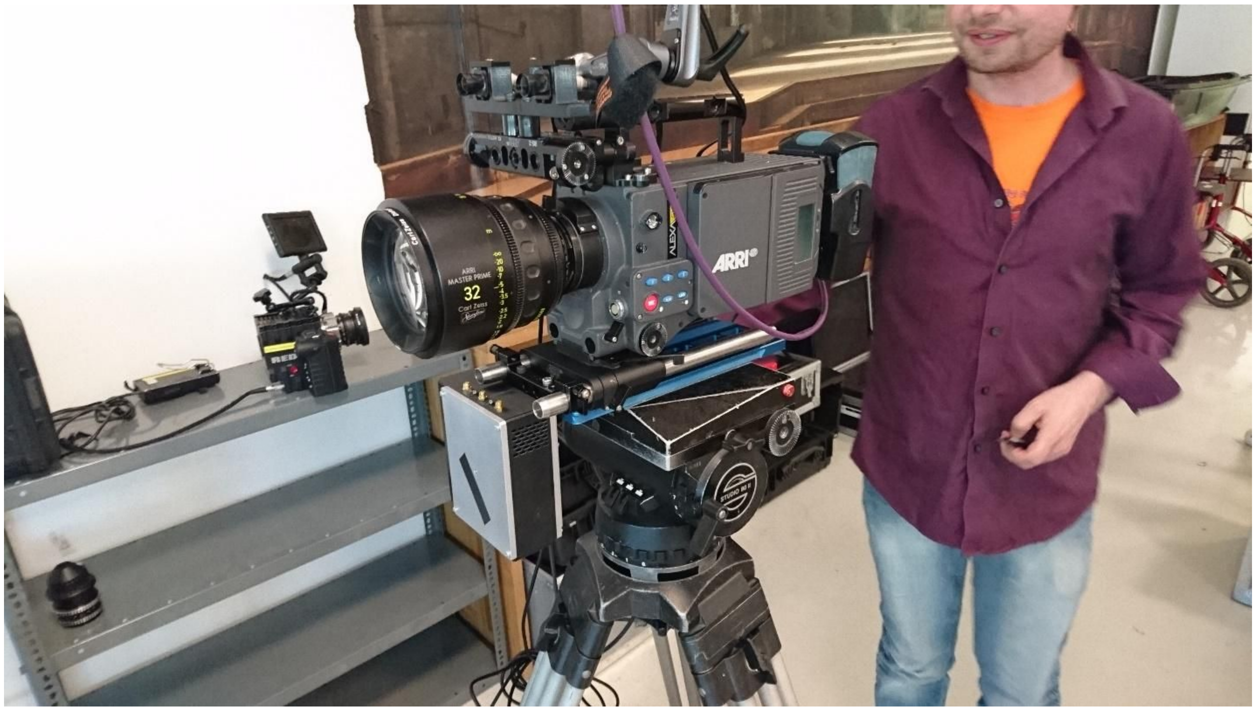
[1] https://cookeoptics.com/i/itech.html

[2] In first instance used for connecting Cooke i-data lens encoders. (https://cookeoptics.com/i/itech.html), but also other devices (e.g., RED cameras, audio recorders, IMU units) deliver data over serial interface.

[3] Can be used both for connecting to wired networks, as well as for receiving data from devices such as witness cameras, 360 cameras and LiDARs.



Here, the single LEMO connector, next to the green one, splits to two USB-2 cables and two trigger cables connected to the two witness cameras. Witness cameras are mounted on a rigid bar, which can be mounted on the camera rig in various ways. The image below shows one of the tested setups:



Currently, CamBox can only be monitored and configured by logging into it through SSH. In the scope of WP1 this is not a problem since task 1.3 will deliver SetBox which will set up WiFi connectivity and provide CamBox management.

Synchronization requirements for data acquisition

For data to be useful in post-production workflows, each data item (lens data and images from witness cameras) must be associated with a particular frame (identified by the timecode) shot by the main camera. To obtain such dataset, data acquisition must obey the timeline depicted in Figure 1:

1. When the main camera starts shooting the next frame, it sets the electrical shutter sync signal to high level. Shutter sync signal remains high for the duration of sensor exposure.
2. CamBox identifies the electrical transition from low to high on its shutter sync input and sends the trigger signal to other acquisition devices.

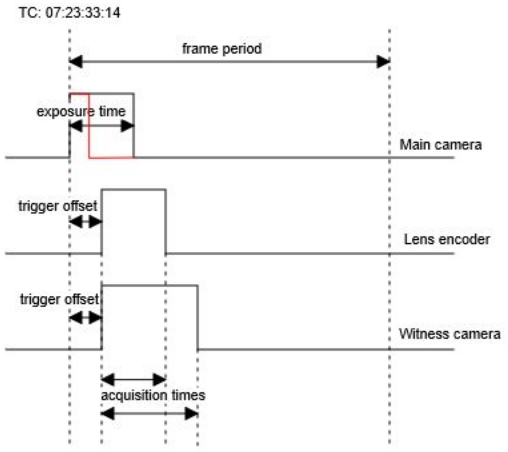


Figure 1: Timing diagram during shooting of a single frame

Because the shutter sync signal is routed through CamBox, the triggering of other devices will necessarily be delayed with respect to start of exposure on the main camera; this delay is denoted by “trigger offset” in Figure 1. Ideally, the offset would always be shorter than exposure time, so that data acquisition starts during the exposure time on the main camera. In practice, this is not always achievable with the current solution for two reasons:

1. Cameras can shoot at extremely short exposure times (under 1 ms), which is illustrated by the red line in Figure 2.
2. Acquisition devices have intrinsic internal delays which we cannot control. This time is included in trigger offset in the figure.

Therefore, we only aim to minimize and stabilize the trigger offset, i.e., that it is as close as possible to a small constant value at the start of each new frame. Users have also indicated that this solution is satisfactory.

In theory, it would be possible to eliminate the trigger offset (but not intrinsic delays) by electrically wiring the shutter sync signal from the main camera to all other acquisition devices. Electrical engineers have advised against this approach because the electrical interfaces (voltage levels) differ across cameras and devices, which would lead to building a prohibitive number of adapter cables: given N cameras and M acquisition devices, we would need to make N*M custom cables, compared to N+M when the trigger signal is routed through CamBox.

Hardware components

While choosing CamBox hardware, we have taken into account previous experience:

- Software triggering of units connected to the USB bus is unreliable because of unpredictable latency of signal delivery. Trigger signals must be electrically delivered from a single source
- Standard (consumer-grade) USB and serial connectors are not suitable for (rough) use on the set because they are not secure enough, i.e., they fall out or lose connection to CamBox.

- Custom circuits with associated software are a burden for both maintenance and production.

To reduce costs and development time, we have chosen to build CamBox and MiniBox on off-the-shelf hardware components and Linux operating system, which the code delivered by POPART was based on. The following components have been chosen for the prototype rig, based on their data sheets:

- COM Express Type-6 motherboard with 6th Gen Intel i5 Core CPU featuring:
    - Intel QuickSync technology for hardware-accelerated H.264/H.265 encoding.
    - 8-bit GPIO (4 inputs + 4 outputs) used for controlling status LEDs, triggering acquisition devices and reading of shutter sync signal from the main camera.
    - 2 RS-232 serial ports.
- COM Express Type 6 carrier board.
- WiFi card with external antenna connections and configurable as access point.
- Gigabit ethernet controller.
- USB LTE modem.
- Mini-PCI SDI capture card.
- Two USB2 grayscale witness cameras with lenses.
- Standard 2-, 3-, 4-, and 2x10-pin LEMO connectors.
- One standard BNC connector.
- Voltage regulator; input: 9-36V; output: 12V.

The casing for CamBox and bearer bar for witness cameras have been prototyped manually. The aluminum parts have been manually filed, while the plastic parts are 3-D printed.

Software components

The software is an updated and modernized version of “HAL” software delivered by POPART project. The software is delivered in a ZIP archive accompanying this document. The source code for CamBox developed by Simula Research Laboratory is released under Mozilla Public License version 2. The supporting source files in the ZIP archive are released under their original licenses. “LICENSE” file in the archive provides more detailed information.

CamBox software adds the following features to “HAL” delivered by POPART:

- Support for Windows (8 or higher) operating systems.
- Parsing of per-frame metadata (including lens data) captured from Arri Alexa studio cameras and saving to JSON format.
- Software-based mechanism for triggering other acquisition devices.
- Improved configurability of capture devices, output formats and processing steps.
- Support for USB3 SDI capture device.
- Support for parsing Cooke’s i-data format.

CamBox software is currently deployed on a 64-bit CentOS 7.3 system with custom-built Linux 4.4.0 kernel,[4] but Windows 10 is used as the main development platform. We are considering to use Windows also as a deployment platform since experience has shown that Windows supports a broader range of hardware, that drivers are of higher quality and that vendors provide better support.

CamBox software is written in C++11 with the following dependencies:

- A compiler supporting the C++ language standard (verified with gcc 4.X and Visual Studio 2015)
- Boost 1.59 or later.
- Intel Media SDK 2017.

[4] Support for hardware H.264/H.265 encoding requires building a custom kernel with provided patches provided by Intel.

- Intel threading building blocks (TBB) 2017 or later.
- FFmpeg version 3.1 with a patch which improves PRORES encoding performance.[5]
- SEMA EAPI, used for controlling the GPIO pins; available from ADLINK.
- FreeType, OpenSSL, DBus-C++ and ZLib.
- Drivers and SDKs for capture devices (SDI, witness cameras, etc); available from the manufacturers.

"HAL" software has been modularized and transformed into a node-based system using the "flow graph" feature of TBB. Within a flow graph system, information (e.g., video frames) is processed by nodes (e.g., video encoder, burn-in of timecode) and sent along edges to other nodes. Since one node executes one well-defined function, the system is easier to maintain and extend with new functionality such as support for new acquisition devices.

Known issues

Most of the issues described stem from hardware limitations not evident from product specifications or bugs in drivers. We are in contact with equipment vendors and working on resolving the issues.

Supported cameras / capture card issues

Cameras deliver important metadata in HANC/VANC[6] packets embedded in the SDI stream. For example, RED cameras deliver recording state (on/off), timecode and filename embedded in a SMPTE timecode packet. Arri cameras deliver extended metadata in a number of VANC packets, while the recording state is delivered in a dedicated HANC packet.

This, combined with the desired form-factor for CamBox and intended use, places following requirements on the SDI capture card:

- mini-PCI form factor,
- delivery of 10-bit image data,
- capture of HANC/VANC data,
- support for Linux operating system.

After extensive search, we have not been able to find such product on the market. We have therefore settled on a capture card that does not support capture of HANC data. The manufacturer has also indicated that it has no intention of supporting HANC capture in the future.

The inability to capture HANC packets allows CamBox to support a limited selection of cameras. Currently, it has been successfully tested with RED and Arri Alexa Studio cameras. While *all* Arri camera models deliver the same *documented* set of metadata in VANC, *only* Arri Alexa Studio also reports the state of the recording flag in VANC metadata. This is an undocumented and not an officially supported feature, but it has been shown to work consistently on several tested Alexa Studio cameras.

Since the electrical interfaces of main cameras differ between manufacturers, CamBox can receive the shutter sync signal only from RED and Arri cameras by using a specially built cable.

In Q1 of 2017, an SDI capture device has appeared that can deliver both HANC and VANC data. However, it is a rather large USB3 device with cumbersome connectors, which does not fit in CamBox casing. We have successfully tested the capture device with the MiniBox proof-of-concept.

GPIO latency

While implementing shutter synchronization, we found out that GPIO on the motherboard is not usable for this purpose. We have measured that sending a single trigger pulse to witness cameras (setting the GPIO output to

[5] This patch has been developed by an intern at Simula and is available at http://ffmpeg.org/pipermail/ffmpeg-devel/2016-May/194582.html
[6] Horizontal/vertical ancillary data.

high, then to low) takes ~160 milliseconds, i.e., four frames while recording at 25 fps. This flaw was not apparent from motherboard specifications, and we are in a dialog with the manufacturer to see whether the issue can be resolved.

Additionally, GPIO does not support interrupts, but the documentation indicates that this is being worked on. This could be worked around in software by continuous polling at a high frequency,[7] but such approach is not feasible either because of the high latency of GPIO operations.

The high latency of GPIO operations has forced us to implement a software-based triggering mechanism, where the arrival of the frame on SDI port is used to trigger witness cameras. While implementing triggering in software, we found that delivery of SDI frames is sporadically significantly delayed. For example, at 25 fps, >99% of frames arrive on time, i.e., in regular intervals of 40ms, but we have observed that a frame can be delayed by up to 70ms. The frame after the delayed one is again delivered on time, i.e., after 10ms.

If sporadic delays are not compensated for, the witness cameras would not be triggered for the delayed frames, and these frames would be missing in the recording. We have therefore introduced a timer which is set to fire 41ms after the last SDI frame that arrived on time. When an SDI frame is delayed, the timer activates triggering of the witness cameras, ensuring that no frames are missing.

Witness camera bandwidth and latency

Initially we wanted to use USB3 witness cameras because USB3 has much higher available bandwidth (5 Gbit/s or more, depending on the version) and lower latency. However, USB3 places much more stringent requirements on contacts and cables. Both manufacturers of robust connectors (Fischer and LEMO) have indicated that they did not expect their connectors to work with the USB3 protocol. We have received the same response from the electrical engineers we cooperated with.

The motherboard has just one USB2 bus (maximum bandwidth of a 480 Mbit/s (60 MB/s)), which is shared between all devices connected to it. *Practically achievable* bandwidth is in practice much lower. First, overheads inherent in the USB protocol (packets headers and checksums) lower the maximum theoretically achievable bandwidth to ~53.2 MB/s. Second, limitations in current implementations of host controllers lower this bandwidth further.[8] Cypress semiconductor has measured that the maximum practically achievable bulk data transfer rate of their host controller is ~43.8 MB/s.[9]

Recording two witness cameras at 25 fps in full resolution would require bandwidth of $1280 * 960 * 25 * 2 = 61.4\, MB/s$, which exceeds even the theoretical maximum bandwidth of a USB2 bus. We have therefore configured witness cameras to deliver frames at 1280x720 resolution, which is a standard SMPTE resolution, and requires a total bandwidth of 46 MB/s. The motherboard has Intel's USB controller which mostly managed to sustain the required bandwidth, but with two major problems.

First, witness camera frames are occasionally delivered late, i.e., after the next SDI frame has already been recorded. This, combined with the fact that the witness camera cannot be triggered to record the next frame before the previous one has been delivered, leads to frame loss.

Second, we have observed inexplicable delays in delivering the trigger signal to the two cameras over USB. The usual delay between delivery of the trigger signal to the two cameras is ~1.5 ms, but this delay sporadically reaches 8-10ms. This leads to loss of shutter synchronization, but also to dropping of later frames, since late trigger delays delivery of the frame.

[7] To detect the firing of the shutter sync signal on *every* frame, input pin must be polled at a rate that is at least twice as high as the recording frame rate.
[8] http://www.cypress.com/file/88486/download
[9] http://www.cypress.com/file/139866/download

Both problems indicate that we're operating at the maximum practically achievable capacity of USB2. Simultaneously, we could not crop the picture to an even smaller size because witness cameras must capture much larger FOV than the main camera to deliver a reliable track for previz.

Fan control

CamBox generates a lot of heat while running, so we installed two high-RPM small fans. The fans make significant noise, which is unwanted in the audio track, so they must be turned off during recording. Even though the motherboard data sheet indicates that the fans can be controlled by software, we have not yet succeeded in turning them off, not even from BIOS.

This is not a critical fault since CamBox won't overheat without fans, but the aluminum plates will become very hot to touch and uncomfortable to handle.

Deliverable: MiniBox

MiniBox runs the same software as CamBox, but configured for acquisition from a single source via a wireless network or a physical cable. When data is captured from devices other than studio cameras, data must be timestamped with wall-clock time instead of timecode. Accurate time is obtained by synchronizing with a NTP server at startup time.

When wireless data acquisition is possible, MiniBox can be a *pure software* component running on the SetBox. Range limitations and unpredictable latencies of wireless networks present two challenges. First, timely delivery of the trigger signal cannot be ensured. The device must therefore be configured for "free-run" mode, i.e., for data acquisition without shutter synchronization. Second, we must support devices without a time source. For these devices, CamBox must continuously estimate the delay of network packets in order to attach accurate timestamp to acquired data. Lost data packets can be easily detected by attaching a monotonically increasing counter to the captured data.

Estimation of network delays will be implemented in Task 1.3 which will deliver WiFi connectivity to test against.

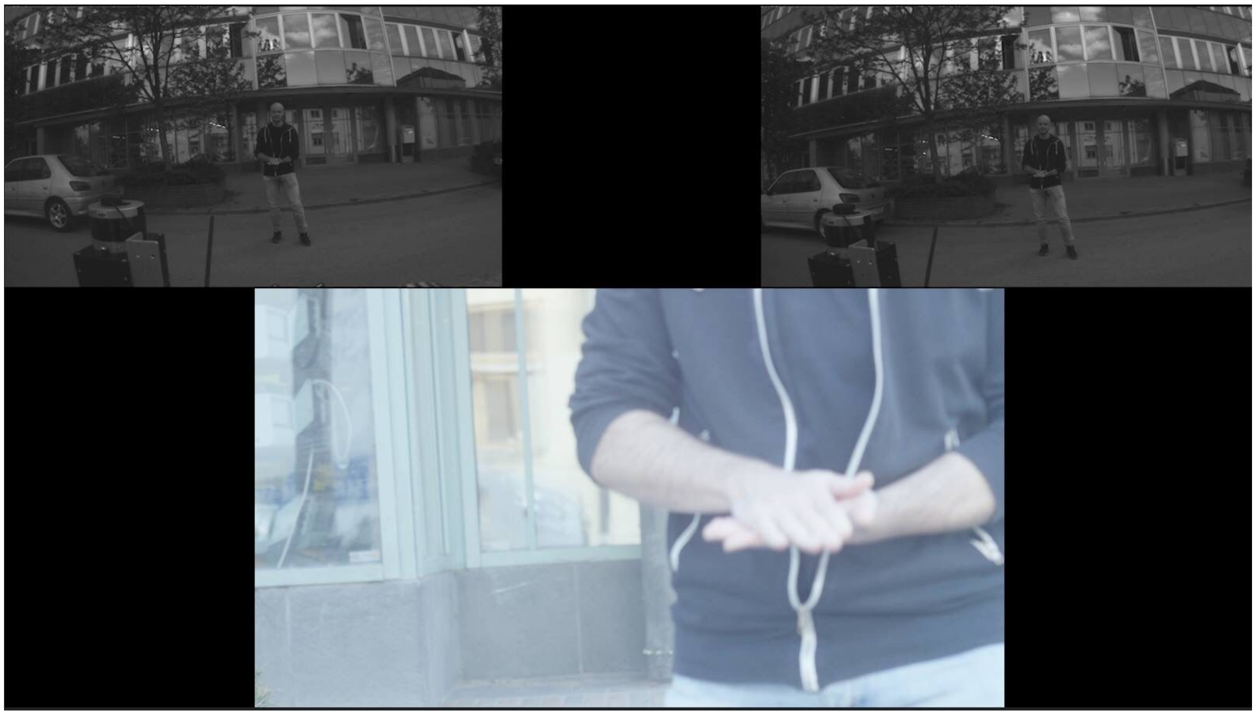
For use as a unit attached to an acquisition device, MiniBox can use far less powerful and cheaper components. We have successfully tested an off-the-shelf Intel Compute stick. It comes with built-in WiFi and Bluetooth connectivity, and other connectivity can be added by external USB adapters.

MiniBox can use the same casing as CamBox, with external LEMO connectors routed to an USB3 hub placed in the same casing, but we believe that it is possible to design a more suitable casing. This has not been pursued due to high costs associated with designing and producing a prototype casing. As of now, no complete boxes have been built, but the concept has been successfully tested by capturing data from one SDI or one witness camera using "loose" components.

Sample data

Sample data has been captured on May 26th, 2017 between 14:00 and 16:00 in front of Visimind's offices in Stockholm, who borrowed us a Velodyne VLP-16 lidar. We have used a RED Dragon camera and 2 grayscale witness cameras with wide lenses connected to CamBox, all mounted on the same handheld rig, as shown in the image below:

The whole dataset is available at http://www.quine.no/datasets/qb-rt-set-01/. We have uploaded a couple of selected takes to youtube with short explanations of what makes them challenging to track in previz. During the shootings we used clapping at the start and the end of each shot (sample image below) to check that the cameras are in sync during the whole take and that no frames have been lost:



Conclusion

We have developed data acquisition software for CamBox and physically built one prototype box which was used to collect the sample data. Due to prohibitive costs, MiniBox casing has not been manufactured, but the concept has been verified to work in the lab using the same software as for CamBox. The software currently supports data acquisition from RED and Arri Alexa Studio cameras and Cooke's i-data lens encoders. The software is modular and easy to extend with support for more devices.

During development, we have encountered critical hardware issues with GPIO (long latencies), which were not apparent from motherboard data sheets. This has forced us to implement shutter-synchronized data acquisition in software: SDI frame arrival and a software timer are used as (mutually exclusive) triggering events, while the trigger signal is sent over USB to the witness cameras.

Software triggering, *with many ad-hoc tweaks*, has worked well enough for shooting the sample dataset, but we do not consider it to be reliable enough for general use on the set: if the recording is longer than approximately 90 seconds, dropped frames from the witness cameras and loss of synchronization are almost guaranteed to occur. Triggering offset also differs between witness cameras; the difference is ~1.5ms, which was tolerable for shooting the sample data.

We are still looking for a workable solution which can ensure shutter-synchronized data acquisition from multiple devices. One possible way is to replace the mini-PCI capture device with a USB3 device, and use the slot for a mini-PCI GPIO expander. This will also give us the possibility of capturing *all* ancillary data and to support a wide range of cameras.

To resolve bandwidth and latency problems with USB2 witness cameras, we are considering replacing them with models from the same vendor, but featuring a gigabit Ethernet port.