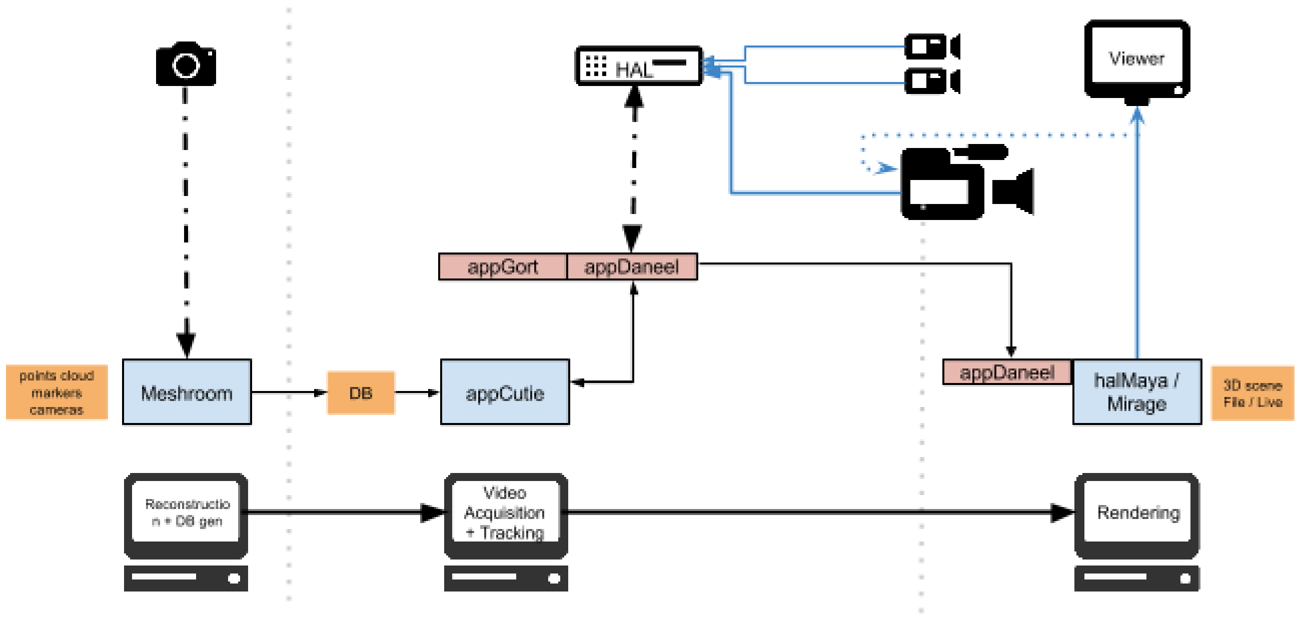
Project acronym: POPART Project number: 644874 Work package: Integration Deliverable number and name: D4.5: Product release and integration report	Title: Product release and integration report Work Package: WP4 Version: 1 Date: 15/04/2016 Author: Håvard Espeland
Type: [X] Report [ ] Demonstrator, pilot, prototype [ ] Website, patent filings, videos, etc. [ ] 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:	
Contents: Deliverable 4.5. This report documents the standalone applications required to use the system.	

Introduction

As of April 2016, the POPART software is in a state where it can be used on-set without a programmer present to setup and tweak the system. In this report, we will document how the different components fit together, the standalone applications and how they are used.

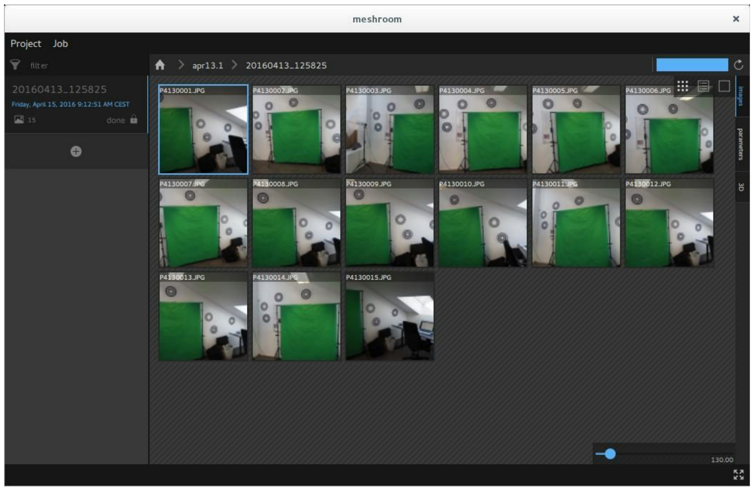


Overview of the main POPART components

Many of the components have been covered in different reports and we will refer to those for the details. In particular we will cover Meshroom and appCutie in this report as these serve as the user interfaces for the whole system. The rendering components (halMaya/Mirage) are covered in the dedicated report D3.3.
In addition to this, we have also developed the tool GAMP to reliably transfer files from HAL and structure the data in a meaningful way in addition to conforming with editing. The GAMP tool is described in detail in the report D5.1.

Meshroom

Meshroom is the user interface to generate 3D reconstructions from a set of still images. It has been developed by MIK. It generates point cloud with cameras poses (based on openMVG library) and optionally it can generate an automatic textured mesh (based on *MVE* library). The output of meshroom can then be used in *MayaMVG (D3.1)* to enable a graphic artist to do manual photo-modeling right inside *Autodesk Maya*. The output of Meshroom can also be used in *Cutie*, to perform the real-time camera tracking.

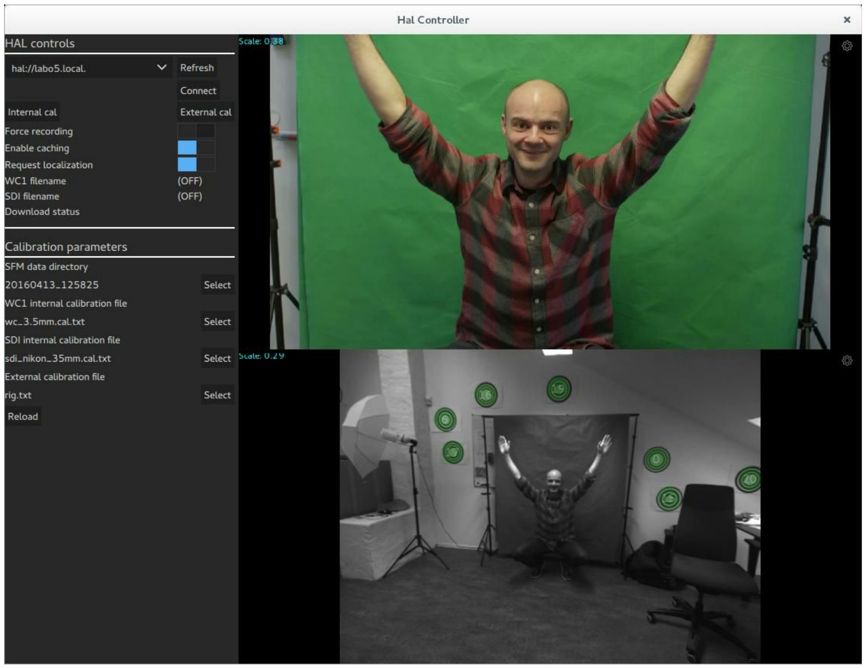


Meshroom tool for 3d reconstruction of scenes, in this case CCtags for camera tracking

Meshroom allows to compute locally on the machine or to perform the process on renderfarm. Currently, we provide an integration with *Tractor, the* renderfarm engine from Pixar (https://renderman.pixar.com/view/pixars-tractor). In both cases, meshroom provide an high level feedback of the progression.
Meshroom allows to visualize and select the images to perform the reconstruction. It provides shortcut to compute multiple variants of the same dataset with different parameters. It includes a 3D viewer to display the results of the reconstruction.

Cutie

Cutie is is the main application to use the POPART previz system. Besides showing streams from both cameras in real-time, it also controls the parameters of HAL and witness cameras, tracking parameters, calibration and other systems. It has two subcomponents, Gort and Daneel, which perform the actual processing and plumbing of the data flows.



HAL controls

hal://labo5.local.

Force recording	
Enable caching	🟦
Request localization	🟦
WC1 filename	(OFF)
SDI filename	(OFF)
Download status	

Calibration parameters

SFM data directory
20160413_125825
WC1 internal calibration file
wc_3.5mm.cal.txt
SDI internal calibration file
sdi_nikon_35mm.cal.txt
External calibration file
rig.txt

The Cutie application allows the operator to inspect how the CCtag detection performs in real-time. The witness cameras provide a wider angle picture that is more suitable for tracking than the main camera.

Cutie is responsible for spawning the sub-components of the system, i.e., Gort and Daneel. The actual camera localization is performed in Gort, while the data flows are handled in Daneel. Details about these components are described in a later section.

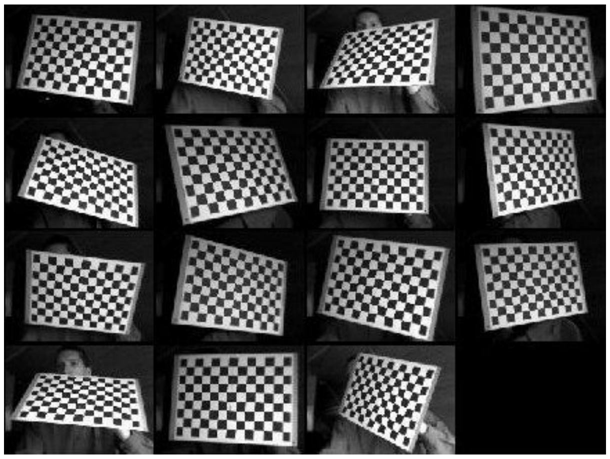
Calibration of cameras

To do tracking with the POPART system we need to perform two types of calibration: lens (intrinsics) and rig calibration. The lens calibration must be done for both the main and witness cameras. For each camera, we retrieve the focal length, optical center and distortion parameters of the lens. The rig calibration calculates the relative position and orientation of the various cameras. Both of these calibrations can now be performed in the "*Cutie"* user interface instead of using the command line tools.

Lens calibration

A video is recorded on HAL as normal holding a checkerboard in front of each of the cameras.

- Be sure that the checkerboard is well fixed to a flat surface, there should not be "bumps" on the surface.
- the checkerboard must fill as much as possible the whole image, as here for example



or in this video https://www.youtube.com/watch?v=NiLBhQKZeLY

- the more poses we have of the checkerboard wrt the camera the better it is
- do slow movements (either the camera or the checkerboard) to avoid motion blur

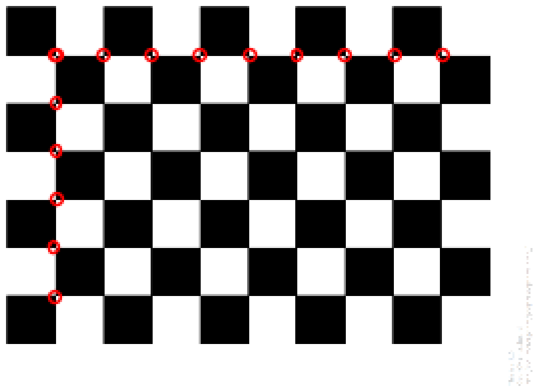
You can get the checkerboard to print by searching images on google. Otherwise OpenCV has a python script that allows to generate a svg file with a custom checkerboard. The script `gen_pattern.py` is in `opencv-3.0.0/doc/pattern_tools`. For example

```
./gen_pattern.py -o out13x10.svg -r 13 -c 10 -T checkerboard -n 30 -s 10.0  -u mm -w 216 -h
279
```

generates a checkerboard fitting an A4, with 13x10 squares of size 10mm.

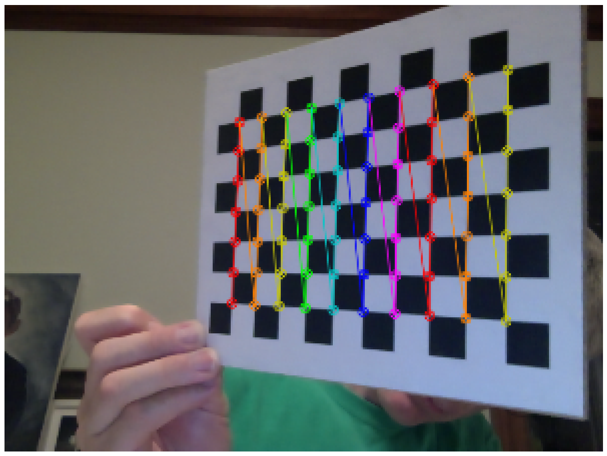
The most important parameters are:

- w and h are the number of **corners** of the checkerboard. The corners are counted like this:



In this case w=9 and h=6

Note that for `gen_pattern.py` this is a 10x7 checkerboard (it counts white and black squares, so in general the w and h to use for the calibration script must be decrease by 1 wrt the one give in `gen_pattern.py`)



When the calibration starts, the video starts to play. You see that the application is able to detect the checkerboard when it overdraw the detected corners.

Rig Calibration

The rig calibration find the relative poses between all cameras used. It takes a point cloud as input and can use both CCTag and SIFT features for localization. The implication is that all cameras must see features (either SIFT or CCTag) that are part of the point cloud, but they do not have to observe overlapping regions. To perform a rig calibration, the operator will start recording on HAL of all cameras. The algorithm will perform localization of all cameras and compute a relative pose. Details of the rig calibration process can be found in deliverable D2.3

Gort sub-component

"*Gort*" is the localization server component, initialized by Cutie with SIFT features, lens calibration data (for main and witness cameras), and rig calibration data. Once running, it continuously receives video frames from "*Daneel"*, computes camera localization, and sends it to Daneel which further delivers it to consumers of localization data ("*Cutie*", "*Mirage"*, "*halMaya*"…)

Gort currently processes video streams at ~6 fps, which means that localization data will always be late with respect to the current video frame. For previz purposes, localization data must be synchronized in time with its corresponding video-frame, while frames without localization data must be skipped. This synchronization is performed by a component which is a part of "*Marvin"* library.

Our near-term plans for Gort consist of improving the observed detection issues:

- Performance: should be real-time.
- Reliability: CCTag markers are sometimes missed or misidentified.
- Jitter: the computed localization is unsteady even while the camera is standing still.

Daneel sub-component

Daneel is the hub component for video streaming. It receives raw data packets containing H.264-encoded video and metadata and provides decoded data to consumer components (Cutie, Mirage, etc) through shared memory. In addition to this fundamental function, Daneel performs two other tasks:

- It can save raw data packets to a cache file on the user's disk.
- It always re-sends the received data packets over a local *wired* network interface.

In both cases, the network packets have the same contents *as if* they were generated by a real HAL.

The ability to save raw network packets to a file and replay them on demand has proven to be very useful. It enables developers to work without having access to a real HAL and to perform tests with the same dataset. Obviously, during replay from file, some functionality is unavailable, e.g., setting parameters on witness camera.

Re-sending of packets over wired network is useful in a distributed scenario where the previz application runs on a machine separate from the one running *Gort*. If the previz machine also received video data directly from HAL, the wireless network would get congested and *all* receivers would get video of low quality. The problem is avoided by connecting only *one Daneel* directly to HAL, whereas other consumers connect to the "first" *Daneel* over a wired network which has much more capacity and less packet loss than a wireless network.

Daneel is controlled exclusively by *Cutie*, where the user can choose which of the available streams to play (either network devices -- HALs or other *Daneels* -- or locally cached files) and where the caching function can be enabled or disabled. It runs in an isolated process and communicates with consumers over a fast RPC protocol. This isolation has proven advantageous for reliability and has also made development easier because a failed component can be restarted without other components being affected.

Marvin sub-component

This is a library providing common low-level functionalities to other system components. These functionalities include:

- Custom UDP protocol for low-latency delivery of video streams over wireless networks. TCP was unsuitable for this purpose because of its exponential backoff in presence of packet loss, which is quite common in wireless networks. This part of *Marvin* is used also by HAL.
- Shared memory interface for video frame delivery with zero overhead.
- H.264 decoder (uses internally the ffmpeg open-source library).
- A Qt video source component, which communicates with *Daneel* and uses queueing to ensure smooth playback and delivery of video frames synchronized with localization data.
- An implementation of RPC protocol that components use to communicate with *Daneel*.

We consider *Marvin* (and *Daneel*) a mature infrastructure components undergoing very little change as we develop other components.

Testing of the system

LAB has done a lot of internal testing of the components in a lab environment and many issues have already been sorted out. Still, the previz system has not yet been used in a full scale installation, which will happen in the coming weeks. Some failure cases can be observed including better handling of CCtag markers that are unreliably detected, e.g., too far away or partially occluded. We plan to introduce a temporal dependence to avoid interfering with the localization in case a marker comes and goes. Additionally, work needs to be done to improve the encoding quality of the witness cameras since H.264 encoding artifacts interferes with CCTag detection. We plan to investigate pre-filtering the raw sensor data with an edge-preserving noise reduction filter such as boxed bilateral filtering before encoding.

The GAMP tool has been thoroughly tested at NRK during the production "Snowfall" as a result of this they have purchased a license to use the GAMP tool for other productions. The feedback from NRK was very insightful and helped improved the product. They liked how the HAL box is mounted to the camera. However, there were some issues that needs improvements:

- The HAL power consumption was too high.
- WiFi antennas were easily broken due to the rough handling on the set.
- The witness cameras from Point Grey could easily damage their connectors, so we need to add some form of extra protection around the connectors.
- Using LEDs for feedback was something NRK appreciated, however they wanted to be able to see more details about errors if a LED indicated issues. The way they asked us to improve this was to add details in the GAMP software. In total NRK was pleased with the testing, and they have already purchased licenses for GAMP to use in another production.

LAB is currently talking with several potential customers to provide servicing with the full POPART previz system in the coming months.

Conclusion

The POPART standalone applications are in a state now that they can be tested on a film set and we will be performing multiple tests and demonstrations in the weeks going forward. Several improvements are planned including improved reliability and increased performance. It will be vitally important to gain experience with using the system on real film sets over the coming weeks. Only end-users can describe what benefits the system actually gives.