Free-viewpoint Immersive Networked Experience

D4.7 Optimized smooth view interpolation algorithms

Project ref. no.	ICT-FP7-248020
Project acronym	FINE
Start date of project (dur.)	1 April, 2010 (36 months)
Document due Date :	1 April, 2013
Actual date of delivery	29 March, 2013
Leader of this deliverable	Hasselt University (UH)
Reply to	sammy,rogmans@uhasselt.be patrik.goorts@uhasselt.be
Document status	Reviewed

Version	**Date**	**Description**
1.0	1 feb 2013	Initial version, ready for review
1.1	29 mar 2013	Edited according to reviews

SEVENTH FRAMEWORK PROGRAMME

Deliverable Identification Sheet

Project ref. no.	ICT-FP7-248020
Project acronym	FINE
Project full title	Free-viewpoint Immersive Networked Experience
Document name	FINE D4.7 Optimized smooth view interpolation algorithms 20130329
Security (distribution level)	Restricted
Contractual date of delivery	1 April, 2013
Actual date of delivery	29 March, 2013
Deliverable number	D4.7
Deliverable name	Optimized smooth view interpolation algorithms
Type	Report
Status & version	1.1
Number of pages	15
WP / Task responsible	WP4: UH - Sammy Rogmans
Other contributors	
Author(s)	Patrik Goorts Sammy Rogmans
EC Project Officer	Thomas Küpper
Abstract	cfr. Page 4
Keywords	View interpolation, recording, plane sweep
Sent to peer reviewer	26 mar 2013
Peer review completed	28 mar 2013
Circulated to partners	Via plone
Mgt. Board approval	To be approved at the next SB meeting

Table of contents

1. Public executive summary

This document describes the work done for deliverable D4.7 for the FINE project by UHasselt. We improved the algorithmic quality by modifying the algorithm as described in deliverable 4.5. Firstly, we improved the depth filtering by incorporating multiple possible depths per player, instead of one, by using the histograms of the depth values instead of the median values. Secondly, we reduced the amount of ghost players by incorporating the silhouettes of the cameras not used for the colour rendering. Lastly, we improved the background rendering by performing more advanced blending of the different background images and by updating the backgrounds to cope with changes in lighting conditions.

To provide more test data, recordings were done in Barcelona, in cooperation with Barcelona Media and MediaPro. Here, 16 cameras placed in a circle were used and covered one half of the field. Calibration was performed by Tracab.

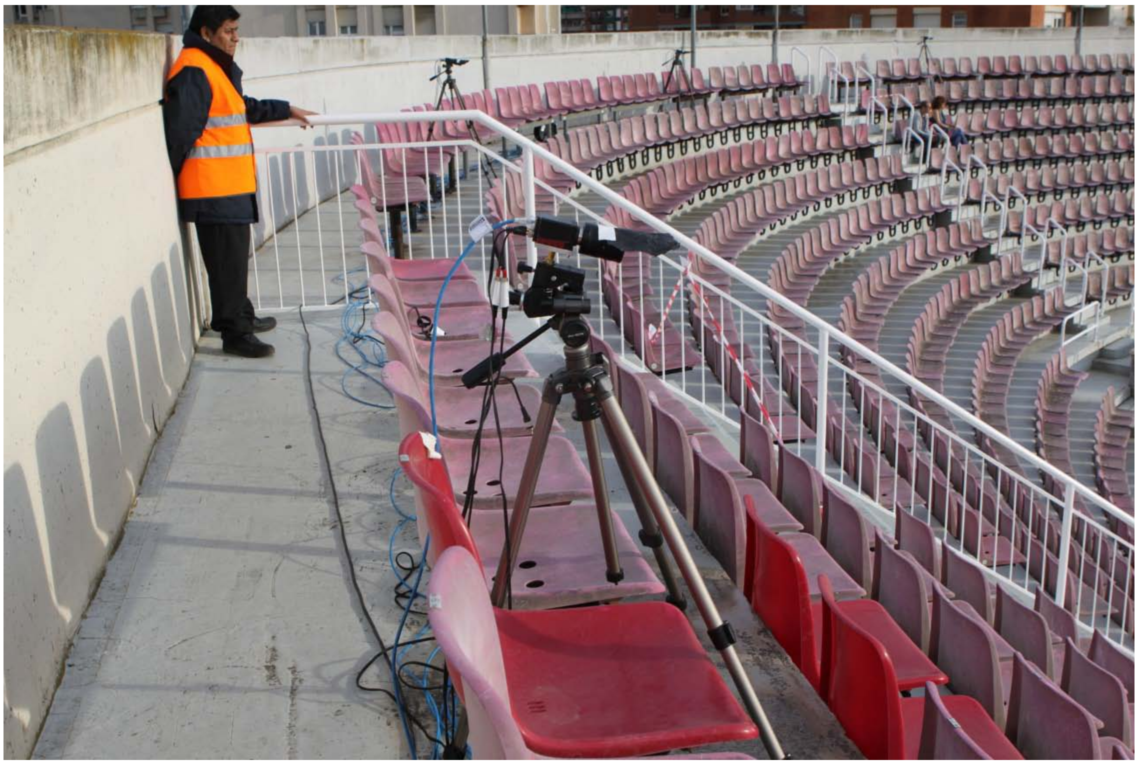
Furthermore, the rendering system is integrated with the storage infrastructure and the viewpath API, provided by EVS.

2. Introduction

This document describes the work done for deliverable D4.7 for the FINE project by UHasselt. More specifically, we describe the improvements to the quality of the interpolation algorithm described in D4.5. We also describe the experimental setup for the recordings we did in order to generate datasets to test and develop the interpolation algorithms. Furthermore, we describe the integration with the storage infrastructure and the viewpoint chooser API. Lastly, we give an overview of the publications performed for this project and discuss some future work.

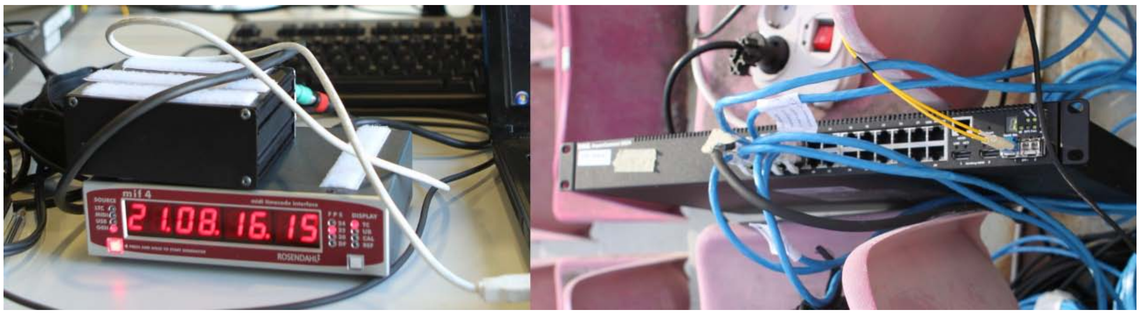
3. Recording

To provide extra scenarios to test the various algorithms of the project, extra recording are performed of a real soccer match in Barcelona, together with Media Pro and Barcelona Media. Here, 16 cameras are used, 8 of UHasselt and 8 of Barcelona Media. The cameras were placed around one half of the field, focused on the penalty mark. This way, all-round interpolation is possible, instead of following action across the field (such as in the last recordings). The cameras were placed 10 meters apart from each other to acquire a broad angular coverage.

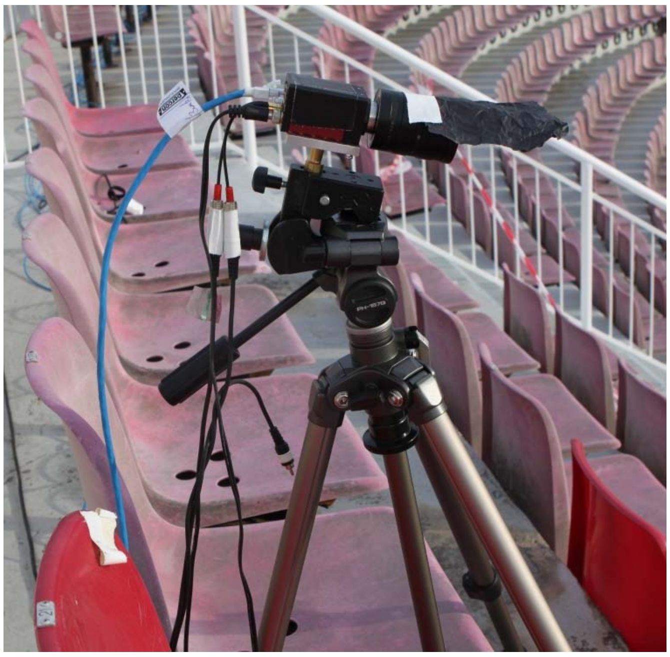


Example of the setup. 4 cameras can be seen, placed 10 meters apart from each other.

The recorded data was stored using a distributed setup, where 3 computers were used to process the data streams. Synchronisation was performed using a daisy-chaining approach using one 25 Hz clock as source. The synchronization signal was transmitted from camera to camera using standard audio cables. The image data was transmitted in raw format using standard 1 Gib Ethernet connections attached to a pair of switches and from there further transmitted to the capture machines using 10 Gib fiber.



The master clock, sending pulses at 25Hz, and the main switch, transmitting the data of 8 cameras.



An example of the used cameras.

To provide consistent calibration for all partners, Tracab performed the calibration after the recordings. From there on, the images could be used by every partner in a consistent manner.

Some example images are provided below.

Examples of the recorded data.

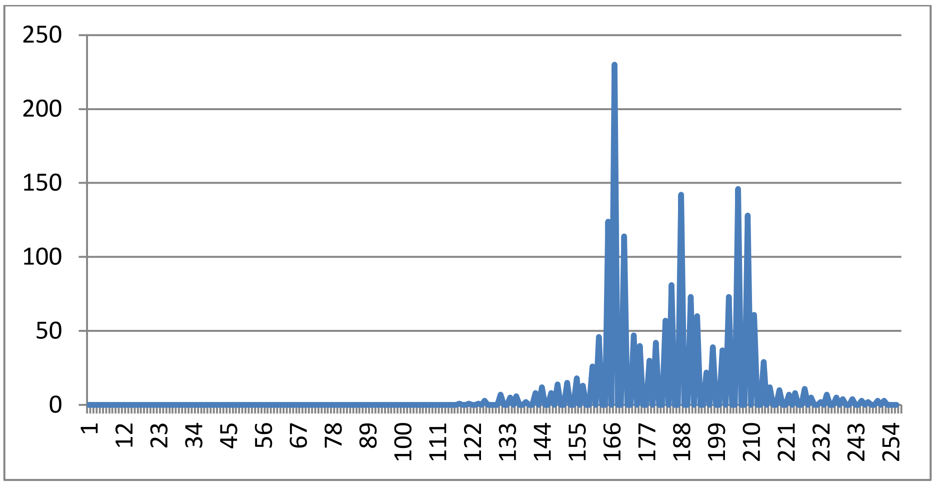
4. Interpolation

The previously described algorithm (see deliverable 4.5) proved to work reasonably well with the new dataset, but proved to perform less optimal in specific cases. We will discuss these cases and propose remedies in the subsequent sections.

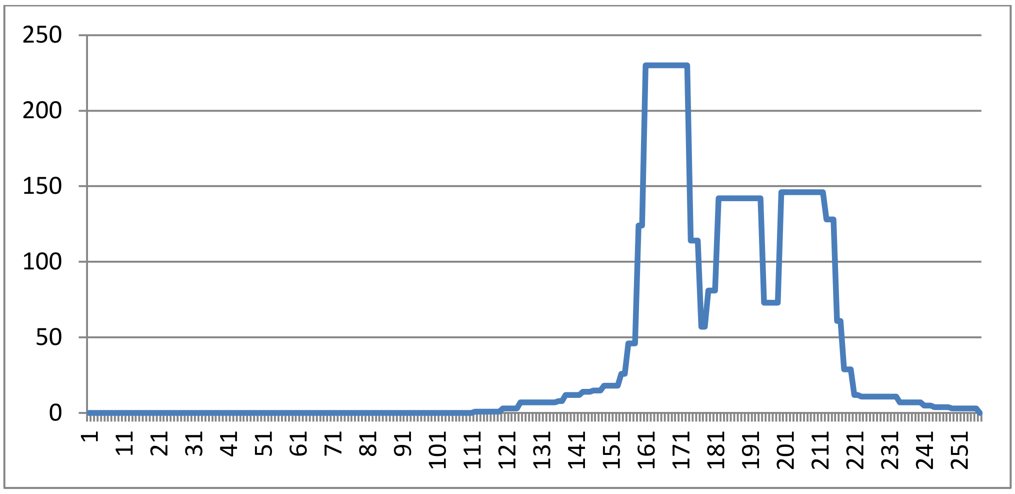
4.1. Large depth differences due to multiple players

In the previously discussed algorithm (see deliverable 4.5), depths were filtered, such that only a range of depths was allowed in the interpolation. This depth is different for every group of connected pixels and is determined by the median depth value as calculated by a naïve depth estimation algorithm. This proved to work well in the provided data, even if different foreground objects overlap in the virtual viewpoint, and thus form one group of pixels. However, as the players are in reality a large distance apart from each other, but still overlap, the players with the least pixels visible in the virtual viewpoint can disappear. Indeed, the difference between the median of the group of pixels and the median depth of the player is larger than the threshold. This will effectively filter out the player and thus remove it completely. The effect is mainly seen in image sets where the camera distance is large, where there are a lot of players simultaneously on the field and where the field of view is large.

To cope with this aspect of the algorithm, we no longer use one depth, but consider a number of depths. Instead of calculating the median per group of pixels, we consider the histogram per group of pixels and select the peaks herein that are considerable relative to the amount of pixels. See for example the next image; this is the depth histogram for three players, merged in one group of pixels. If we select only one depth, some peaks will disappear, including the corresponding players.



More concrete, we firstly calculate the histogram per group of pixels using CUDA and the thrust library. This is in essence a reduce operation, which is well-known for GPU implementation. Next, we apply an envelope operation on the histograms to reduce noise and to merge close-by peaks together. We do this by considering a range (e.g. 9 values) around a value in the histogram and saving the maximum of all the values in the range. This can be seen in the next figure.

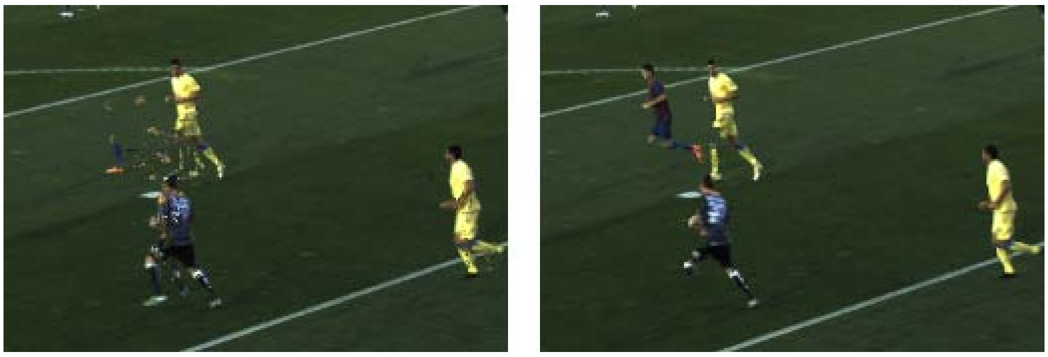


We now have a set of filtered histograms, one for each group of pixels. Then, we generate a new binary image for every processed depth, encoding if this depth is a valid value for this depth (encoded as 1) or not (encoded as 0). To reduce memory usage, the image is generated on-the-fly when executing the depth-aware plane sweep, thus only one image is available, and used, at one time instance. To generate the binary image for a given depth, we consider all the foreground pixels of the previous depth map, and compare the current processed depth with the histogram value for the corresponding group of pixels. If the histogram value is larger than the stated threshold, the depth will be processed (encoded 1), otherwise not (encoded 0). Because we applied an envelope operation on the histogram, there is already a range around the peak of the histogram.

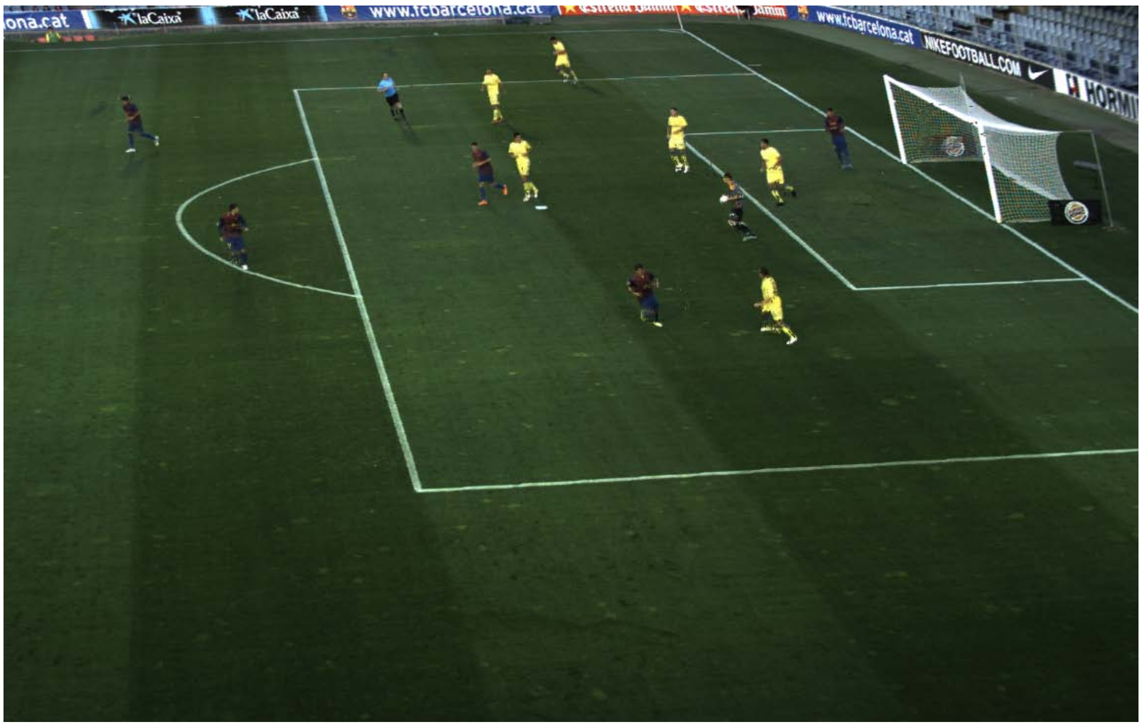
If there is only one player in the group of pixels, there will be only one peak in the histogram that is of considerable height, and thus only depth values around this value will be considered.

Because of the extra processing to increase the quality of the final result, extra processing power is required. To allow stable real-time processing, multiple rendering computers are now supported to acquire an overall real-time processing pipeline. See section 6 for more information.

The following images compare the old and the new system.



The following image is a full interpolated view. The camera is placed in the middle of two real camera viewpoints.



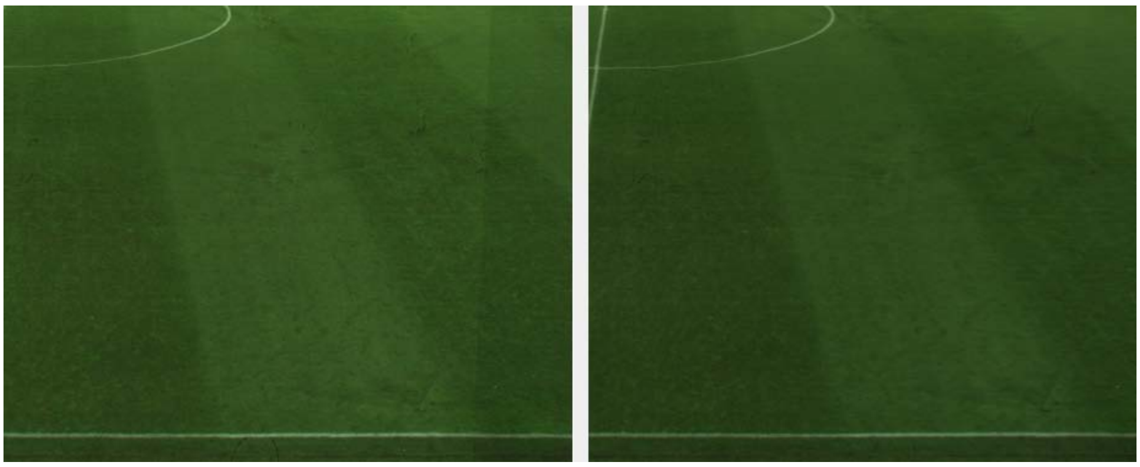
4.2. Color consistency for very wide baseline setups

Cameras with a large distance between them provide very different images. Therefore, the color consistency check using SSD to determine the best depth can provide wrong results when using more than 2 cameras. Therefore, we only use 2 cameras for the color and only use the foreground/background information of the other cameras. This way, the problem of matching pixels for very different viewpoints is reduced, while still reducing artifacts from ghost players by using the silhouettes of the players.

4.3. Color differences between cameras

When color differences in the camera images can be perceived, the background rendering can introduce artifacts. For example, if one camera is darker than the other, the background will show a sudden transition from the background of one camera to the other, resulting in a perceivable lighting error. In the previous data, this effect could easily be perceived in the large field-of-view interpolation. Because this effect is very distinct and color calibration can be hard, we incorporated a method to reduce the perceivable error.

Previously, the background of the closest camera was projected on the plane of the field and the other cameras were used to fill up all the background pixels that are not covered by the closest camera. This will result in distinct transitions especially for virtual camera positions that are in the middle of the real camera positions. To cope with the problem, we store for every projected pixel the squared distance to the border of the projected background. We use these distances to determine how much we will blend the backgrounds of the other cameras; the larger the distance (i.e. further from the center), the less we use the color of the first background. This will provide unperceivable transitions, thus effectively reducing these sorts of artifacts.



4.4. Changing lighting conditions during twilight

When the lighting changes gradually, the foreground and background subtraction can introduce errors, such that too much will be detected as foreground, resulting in low quality foreground interpolation. This subtraction error is caused by a too large difference between the previously determined background and the current, changed background. To reduce the effects of gradual lighting changes, the background used for subtraction is continually updated with the current detected background pixels. This will skip updating the current foreground pixels, but will be updated when the foreground has moved. This methodology can only work for gradually changing backgrounds. This is a valid assumption due to the nature of changing lighting conditions in outdoor scenes, such as twilight.

5. Publications and Dissemination

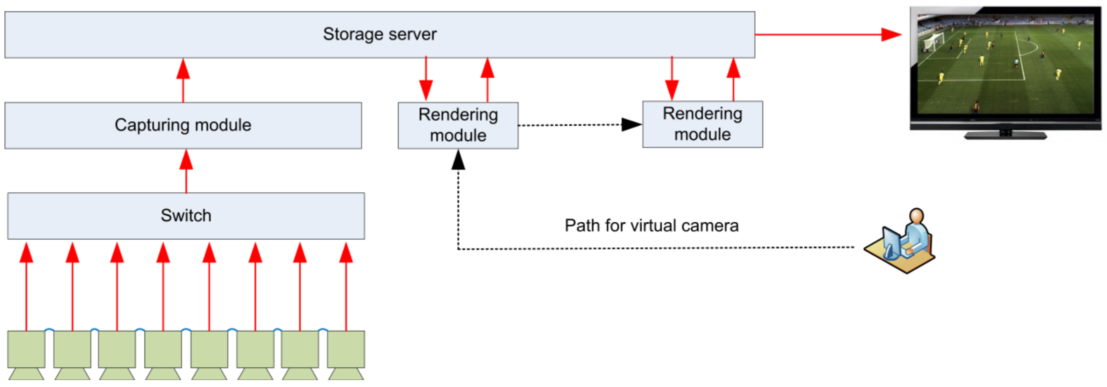
The results of the interpolation algorithm and the system integration have been submitted to several conferences.

1. A system overview paper is accepted and presented on IC3D 2012, in Liege. The conference aims to bring together people and companies working with 3D content. The paper was rewarded the best paper award.
2. A technical paper regarding the interpolation algorithm is accepted and presented on VISAPP 2013, Barcelona. These results will be presented in February 2013. The conference is focused on computer vision and their applications.
3. A technical overview of the GPU optimization and methodologies used is presented at GTC 2013. This conference aims to bring together companies and academia using computational GPU methodologies.

Furthermore, the interpolation algorithm is presented at IBC 2012, Amsterdam, together with the 3D reconstruction setup.

6. Integration in overall setup

To provide integration with the global setup, integration is implemented in out framework. The method of linking the different modules together is shown in the next figure.



Overview of the system, using the different modules.

Firstly, recording can be done using the storage server, developed by EVS. The raw images are captured by a capture module using the API of the cameras. Then, extra information is appended to the image data, such as timestamps and camera numbers, and is subsequently transmitted to the storage server, using the API provided by EVS.

Secondly, stored images can be retrieved by the rendering modules, using the API provided by EVS. The required raw data is transmitted from the storage server, and processed by the rendering module. The results can be transmitted back to the storage server as RGB images.

Lastly, the choice of the path of the virtual camera can be provided by the ViewPath API provided by EVS. The API provides a rendering request consisting of a set of virtual camera positions and a location on the storage server to store the results. To reduce the hardware cost and to guarantee fast processing speed, multiple rendering modules can be used to render a requested virtual camera path. This is invisible for the API. First, a master rendering module accepts a rendering request. Then, this rendering master delegates a part of the rendering request to another rendering module. This slave module will render half of the request and the master will render the other half. Both results will be merged to the storage server.

7. Summary of future work

- Continue the publication of the results, especially optimization and interpolation techniques.
- Perform more testing of the integration to increase the robustness of the overall system.