



V4Design

Visual and textual content re-purposing FOR(4) architecture, Design and virtual reality games

H2020-779962

D4.3

First iteration of 3D reconstruction and scientific report

Dissemination level:	Public
Contractual date of delivery:	Month 18, 30 June 2019
Actual date of delivery:	Month 18, 28 June 2019
Workpackage:	WP4: 3D model extraction from 2D visual content
Task:	T4.3
Type:	Report
Approval Status:	Approved
Version:	1.5
Number of pages:	91
Filename:	D4.3_V4Design_v1.5.pdf

Abstract

This deliverable represents the first iteration of 3D reconstruction and scientific report describing and showing the SoA methods for image sequence analysis, and 3D sparse and dense reconstruction.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.



co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	08/04/2019	ToC creation	Maarten Vergauwen (KUL)
0.2	12/04/2019	First text on shot and keyframe detection	Maarten Vergauwen (KUL)
0.3	16/04/2019	First text on photogrammetric pipeline	Maarten Vergauwen (KUL)
0.4	17/04/2019	Restructure and rename paragraphs in TOC based on comments from CERTH. Added examples of shot detection	Maarten Vergauwen (KUL)
0.5	03/05/2019	Text on photogrammetric pipeline and reconstructions + enhanced 3D model extraction	Maarten Bassier & Jens Derdaele (KUL)
0.6	22/05/2019	Elaborated on keyframe extraction + implementation	Maarten Vergauwen & Jens Derdaele (KUL)
0.7	23/05/2019	Moved two parts to D4.2 + adapted text (shot detection and blurriness)	Maarten Vergauwen (KUL)
0.8	24/05/2019	Preliminary text on user and system requirements. Extra data and examples for reconstruction + enhanced 3D models	Maarten Bassier (KUL)
0.9	27/05/2019	Revision of text	Maarten Bassier (KUL)
1.0	27/05/2019	Revision of text and restructuring numbering. First version of the introduction	Maarten Vergauwen (KUL)
1.1	03/06/2019	Methodology system integration + SfM pipeline discussion	Jens Derdaele (KUL)
1.2	03/06/2019	Final revision of deliverable before internal review	Maarten Bassier (KUL)
1.3	17/06/2019	Version, incorporating comments by internal reviewer	Maarten Vergauwen (KUL)
1.4	18/06/2019	Remove content present in D6.3	Jens Derdaele (KUL)
1.5	19/06/2019	Move parts back from D4.2 (undo 0.7)	Maarten Vergauwen (KUL)

Author list

Organization	Name	Contact Information
KUL	Maarten Vergauwen	maarten.vergauwen@kuleuven.be
KUL	Jens Derdaele	jens.derdaele@kuleuven.be

KUL	Maarten Bassier	maarten.bassier@kuleuven.be
CERTH	Kostas Afgerinakis	koafgeri@iti.gr

Executive Summary

This deliverable reports on the development and implementation of the 3D reconstruction pipeline. Next to the relevant scientific description, the integration of the message passing schema of the 3D reconstruction and functions from Open Source libraries are discussed in this document as well. The goal of the 3D reconstruction is the production of 3D mesh models from image sequences originating from video and crawled imagery. This is further extended by integrating building intelligence to extract meshes of existing patrimony.

This document describes in detail the WP4 modules and includes the presented methods, functions and resources that were adopted in order to fulfil the requirements specified in the DoA and user requirements (D7.1, D7.2). The deliverable thoroughly describes the literature, methodology and experiments of the 3D reconstruction that were implemented in the first prototype (M18). The inputs from STBOL, AE&TP and the crawling are discussed with respect to the message passing schema. For each module, an overview of the State-of-the-Art (SoA) and a comparison to other approaches is presented. This includes the literature concerning Structure-from-Motion (SfM), existing software functionalities, video processing and enhanced model extraction. The implemented functions are tested on realistic project data and the performance is explained at the end of each section.

The following modules are presented. We highlight the most important parts for each:

- a) The extraction of keyframes from input video sequences, which includes keyframe estimation and detecting of degenerate inputs and blurry frames.
- b) The photogrammetric SfM process including image preprocessing, sparse reconstruction, dense reconstruction, meshing and model texturing.
- c) The enhanced model extraction which interprets STBOL and AE&TP outputs as described in D3.2 to segment building geometry.

KUL was responsible for the development of the described methodologies and modules for 3D reconstruction and enhanced model extraction. The process is fully automated including parameter estimation, message passing and model extraction. The input raster data is processed on KUL servers and the outputs are stored in the knowledge base developed by CERTH conform the schema set up in WP2. The future work includes increasing system robustness, integration of Linked Data from WP5 and additional reconstruction functions.

Abbreviations and Acronyms

AE&TP	Aesthetics extraction and texture proposals
SfM	Structure-from-motion
STBOL	Spatio-temporal building and object localization
SoA	State-of-the-Art
GRIC	Geometric Robust Information Criterion
MLE	Maximum Likelihood Estimator
SIFT	Scale-Invariant Feature Transformation
DoG	Difference of Gaussians
RPC	Remote Procedure Call
JE	Joint Entropy
w.r.t.	with respect to
SAD	Sum of Absolute Differences

Table of Contents

1	INTRODUCTION.....	9
2	3D RECONSTRUCTION REQUIREMENTS.....	10
2.1	3D reconstruction requirements.....	10
2.2	Enhanced model extraction requirements	11
3	3D RECONSTRUCTION PIPELINE: OVERVIEW AND RELATED WORK	13
3.1	Image and/or video content	13
3.2	Camera calibration via sparse matching and reconstruction	13
3.2.1	Image features.....	14
3.2.2	Matching strategy.....	14
3.2.3	Camera recovery and sparse point cloud reconstruction	14
3.3	Dense matching.....	15
3.4	Dense point cloud generation, modelling and texturing	15
4	PROCESSING VIDEO.....	17
4.1	Issues with processing video w.r.t. images	17
4.2	Shot detection	17
4.2.1	Hard cuts: SAD-score and Histogram	17
4.2.2	Fade cuts: Mutual Information.....	20
4.3	Dealing with blurry frames.....	22
4.4	Image and video sequence analysis and keyframe selection	28
4.4.1	Video and photogrammetric reconstruction.....	28
4.4.2	GRIC	30
4.4.3	Selecting keyframes.....	32
4.4.4	Detecting degenerate sequences	33
4.5	Implementation and examples	33
4.5.1	Implementation	33
4.5.2	Bauhaus - Dessau example	35
4.5.3	Kochuu: example of panorama	39
5	3D RECONSTRUCTION: IMPLEMENTATION AND COMPARISON OF PHOTOGRAMMETRIC RECONSTRUCTION SOLUTIONS	42
5.1	General overview	42

5.1.1	Software	42
5.2	In-depth comparison of software tools and libraries	42
5.2.1	List and details of compared tools	43
5.2.2	List and details of compared methods	43
5.3	Experiments.....	46
5.3.1	Datasets	47
5.3.2	Overall test results.....	52
5.3.3	Comparison of the sparse reconstruction	56
5.3.4	Comparison of the dense reconstruction and meshing	59
5.4	Architecture design 3D reconstruction Pipeline	62
1.	Content preparation	62
2.	Frame extraction	62
3.	Sparse, dense and textured mesh	63
4.	Metadata management and result publishing	63
5.5	Implementation of SfM algorithms.....	63
5.5.1	Keyframe extraction	64
5.5.2	Sparse Reconstruction.....	64
5.5.3	Dense reconstruction	64
5.5.4	Service output.....	64
5.5.5	Pipeline experiments.....	66
5.5.6	Conclusions.....	68
6	ENHANCED 3D MODEL EXTRACTION	69
6.1	Input from STBOL and AE&TP algorithms	69
6.1.1	STBOL.....	69
6.1.2	AE&TP	70
6.2	Segmentation of 3D models.....	72
6.2.1	Masking in SfM	72
6.2.2	Methodology	74
6.2.3	Experiments.....	75
6.2.4	Future work enhanced model extraction.....	78
6.3	Texture enhancement	80
6.3.1	Methodology	81
6.3.2	Texture Experiment	83
6.3.3	Future Work texturing.....	87
7	CONCLUSION & FUTURE WORK	88
7.1	Conclusions	88
7.2	Future Work	88

1 INTRODUCTION

The objective of the current deliverable is to describe the progress and the first implementation of the 3D reconstruction module of the V4Design project, as well as provide background information on the State-of-the-Art in a scientific manner. The 3D reconstruction module is a vital part of the V4Design pipeline. This document describes the challenges we face when dealing with data from the V4Design content provider partners and the solutions that have been developed. As such it relates to deliverable D2.1 *“Initial visual and textual dataset creation and legal and ethical requirements”* that provides an overview of the image content, and to deliverable D4.1 *“Empirical study of visual content”*, which highlights the issues present in this data. This deliverable also relates to deliverable D4.2 *“Basic version of interior and exterior localization algorithms and tool”*, as will be explained in chapter 3 to 6.

The deliverable starts with an overview of the relevant user and system requirements that drove the development of the 3D reconstruction and the enhanced model extraction algorithms. These requirements are listed in chapter 2.

In order to be comprehensive, chapter 3 provides a description of a typical photogrammetric 3D reconstruction pipeline and the State-of-the-Art of algorithms that contribute to the result: the extraction of 3D information from imagery. Camera estimation, sparse and dense reconstruction, 3D model generation and texturing are explained.

In chapter 4 we elaborate on an how we tackle the important issue, encountered when processing V4Design input data: the fact that a significant portion of the data consists of video sequences. Multiple problems arise, such as the shot detection, blurry frames, and keyframe extraction. All three issues are thoroughly explained in the current document. We also show that a positive side effect of our solution is the possible detection of degenerate cases, i.e. videos for which 3D reconstruction is not feasible.

Chapter 5 is an extensive chapter that deals with the implementation of the reconstruction module for the V4Design pipeline. An overview is provided of the available State-of-the-Art packages and experimental results are presented that reveal their pros and cons. We explain which choices were made and provide more information on the implementation.

Chapter 6 deals with enhanced 3D model extraction. We explain how the input from the STBOL and AE&TP modules can improve the 3D models, both for segmentation (via masking) and texturing. We also highlight possible future improvements.

Finally, chapter 7 concludes the document, focusing on the strong points of the developed 3D reconstruction algorithms.

2 3D RECONSTRUCTION REQUIREMENTS

The V4Design user requirements have been identified in D7.1,7.2 through the initial use case scenarios and user requirements. Some of them are associated and directly linked to the 3D reconstruction pipeline, as detailed in Section 2.1 and Section 2.2.

2.1 3D reconstruction requirements

A number of user requirements from D7.2 (Use cases, requirements and evaluation plan) have been associated with the SfM module (Table 1). Regarding UR_002, an architect wants to retrieve a 3D model from a set of input images. Subsequently, high quality textures are extracted as desired by UR_003. UR_8,B,C,D and E concerning various file formats is fulfilled by publishing the outputs conform existing specifications and standards. These formats can easily be exchanged using native i/o functions of existing software such as Rhinoceros. In the UR_11A, an architect wants access to a gallery of 3D models, which refers to the knowledge base that is populated through the proposed SfM pipeline. UR_13,14 pose requirements concerning the accessibility of the metadata of the inputs and outputs of the SfM pipeline. The 3D reconstruction process will accomplish these requirements by exporting metadata at key steps of the process including the sparse reconstruction, the dense reconstruction, the meshing and the model texturing. Similarly, UR_21 is fulfilled by not only populating the knowledge base with meshes but also with the dense point clouds.

Table 1: Relevant user requirements reported in D7.1,7.2 for 3D Reconstruction

User Requirement (UR)	Associated High Level User Requirement (HLUR)	Detailed description of the user requirement	Associate Pilot Use Case (PUC)	Functional or Non Functional. (FR/N-FR)	MoSCoW framework based analysis
UR_002	HLUR_201	As an Architect I want to be able to retrieve 3D-Models	PUC1 PUC2	FR	MH
UR_003	HLUR_202	As an Architect I want to be able to retrieve high and reduced resolution textures	PUC1 PUC2	FR	MH
UR_008	HLUR_204	As a user I want various file formats as outputs:	PUC1 PUC2 PUC3 PUC4	FR	SH
UR_008B	HLUR_204	As a user I want various output file formats such as OBJ and FBX for 3D models	PUC1 PUC2 PUC3 PUC4	FR	SH
UR_008C	HLUR_204	As a user I want various output file formats such as JPG, TIFF, BMP and PNG for textures	PUC1 PUC2 PUC3 PUC4	FR	SH
UR_008D	HLUR_204	As a user I want various output file formats such as vmat and mdl for Materials	PUC1 PUC2 PUC3 PUC4	FR	SH
UR_008E	HLUR_204	As a user I want various output file	PUC1	FR	SH

		formats such as Adobe swatches library for Colour Palette	PUC2 PUC3 PUC4		
UR_011A	HLUR_203 HLUR_207 HLUR_208	As an Architect I want a UIX a detailed view of a Gallery of 3D model (with/without texture) and usage examples from other users	PUC1 PUC2	N-FR	MH
UR_013	HLUR_203 HLUR_207	As an Architect I want UIX: Detailed search by features: - Quality (3D model/ texture), Footage features, augmented data	PUC1 PUC2	N-FR	SH
UR_014	HLUR_203	As an Architect I want UIX: Download settings (saveable profiles): - Mesh quality/format, Texture quality/format/ layers (checkboxes), Material definition file, Colour palette (e.g.: adobe swatches)	PUC1 PUC2	N-FR	MH

2.2 Enhanced model extraction requirements

Seven user requirements from D7.2 have been associated with the enhanced model extraction module of V4Design, namely the UR_002, UR_003, UR_004, UR_011A, UR_013, UR_017 and UR_018 (Table 2). Similar to the 3D reconstruction, UR_002, UR_003, UR_011A and UR_013 are fulfilled by segmented models only depicting building geometry. Additionally, the user can decide texture quality. The texture reuse discussed in UR_004 is met by integrating the texture superimposition of D3.2 (Basic version of aesthetics concept extraction algorithms & tools) onto the calculated models. The system allows the definition of multiple texture files for one mesh model, allowing the user to choose a texture of a model according to their needs. As far as UR_017 is concerned, the model recognition is focussed on building extraction in addition to the more general 3D reconstructions. Finally, in the UR_018, an Architect wants to have the "intelligence" of an architectural composition tool (combination of texture, colours, shapes). The enhanced model extraction and texturing identifies combo's of mesh objects and textures and produces detailed building geometry.

Table 2: Relevant user requirements reported in D7.2 for Enhanced model extraction

User Requirement (UR)	Associated High Level User Requirement (HLUR)	Detailed description of the user requirement	Associate Pilot Use Case (PUC)	Functional or Non Functional. (FR/N-FR)	MoSCoW framework based analysis
UR_002	HLUR_201	As an Architect I want to be able to retrieve 3D-Models	PUC1 PUC2	FR	MH
UR_003	HLUR_202	As an Architect I want to be able to retrieve high and reduced resolution textures	PUC1 PUC2	FR	MH
UR_004	HLUR_202	As an Architect I want to be able to reuse textures (Pattern extraction / seamless texture generation)	PUC1 PUC2	FR	CH
UR_011A	HLUR_203 HLUR_207 HLUR_208	As an Architect I want a UIX a detailed view of a Gallery of 3D model (with/without texture) and usage examples from other users	PUC1 PUC2	N-FR	MH

UR_013	HLUR_203 HLUR_207	As an Architect I want UIX: Detailed search by features: - Quality (3D model/ texture), Footage features, augmented data	PUC1 PUC2	N-FR	SH
UR_017	HLUR_203 HLUR_205 HLUR_206	As an Architect I want texture and material recognition that might appear in images and videos.	PUC1 PUC2	N-FR	CH
UR_018	HLUR_203 HLUR_208	As an Architect I want to have the "intelligence" of an architectural composition tool (combination of texture, colours, shapes)	PUC1 PUC2	N-FR	CH

3 3D RECONSTRUCTION PIPELINE: OVERVIEW AND RELATED WORK

The 3D reconstruction pipeline of V4Design is built according to well-established photogrammetric principles and consists of a set of consecutive steps that are executed automatically. Manual intervention is not necessary but can improve the results in certain situations. This chapter outlines the procedure and briefly describes the most important concepts of every step.

3.1 Image and/or video content

Every photogrammetric 3D reconstruction pipeline depends heavily on the quality of the input data. Deliverable D4.1 dealt with this issue in a thorough manner. In this document the requirements of the input data were described and the available data from content providers and outside sources was assessed. It was concluded that the most important requirements for good input data are:

- The presence of a **baseline** between camera positions, since it is not possible to calculate depth information from images that are captured without changing camera position between recordings.
- The **rigidity** of the recorded scene. Moving objects (people, cars, ...) cannot be reconstructed but if the majority of the scene is rigid, they can be eliminated from the 3D reconstruction.
- The presence of sufficient **distinct** features in the scene, i.e. no untextured, homogeneous surfaces.
- The image **quality** and **resolution** have a direct impact on the 3D result.

Most photogrammetric software packages or libraries focus on processing separate images, because their resolution is typically (much) higher than that of videos and the baseline constraint is also more easily fulfilled when the camera shoots individual pictures. However, in V4Design we deal with many datasets that contain video. The adaptation of the algorithms (such as the detection of shots and the extraction of keyframes) is important in this regard and will be described in chapter 4.

3.2 Camera calibration via sparse matching and reconstruction

Photogrammetric 3D reconstruction is built upon the principle of resection, in which the intersection between the backprojected rays from two or more images is computed. In order to execute this, two pieces of information are needed: the camera calibration (both internal and external) in order to compute the backprojected rays, and the correspondences between images through which these rays are constructed.

The number of pixels in an image is so large that it is unfeasible to employ every pixel from every image to recover the camera calibration. Moreover, not every pixel is equally well suited for this task. That is why most photogrammetric pipelines recover the camera calibration through a so-called ‘sparse matching and reconstruction’ step. This is a set of algorithms, that typically consists of the following elements:

1. Extraction and description of image features
2. Matching of image features between images
3. Recovery of the camera setup and reconstruction of a sparse point cloud

We will briefly describe these steps and highlight the relevant issues for V4Design.

3.2.1 Image features

Sparse feature points in the images are extracted and their appearance is described using a numerical descriptor. Widely used and well-performing extraction and descriptor algorithms are SIFT (Lowe, 2004) and SURF (Bay e.a., 2008), together with their variants (SIFT-GPU, SURF-GPU, ASIFT, DAISY, and so on). Feature extraction has a very high influence on the performance and success of the entire pipeline. For this reason, all photogrammetric reconstruction packages employ several settings to control the number of feature points per image. COLMAP (Schönberger, 2016) for example allows very in-depth adjustable thresholds and parameters for the feature extraction algorithm, while commercial packages typically do not disclose their internal algorithms. Their settings are typically limited to internal image resizing and selecting the maximum number of feature points per image.

When dealing with video frames, it is also possible to employ video tracking instead of matching. Many tracking algorithms exist but literature agrees that the KLT tracker (Simon et al., 2004) and its variants (affine KLT, adaptive KLT, Conv KLT) (Ramakrishnan, 2016) outperform all others. Video trackers make use of the fact that the features to detect and match appear in images that are consecutive and close together.

3.2.2 Matching strategy

The relative camera motion between a set of images will be determined with the use of corresponding features. Standard exhaustive matching approaches attempt to match every image against every other image. Since in this approach the number of matching candidates increases quadratically with the image count, exhaustive matching is only viable with a relatively low number of images. It is beneficial to build a feature space database and apply approximate nearest neighbors search algorithms to speed up the matching of features in this space. This is of particular importance for relative matching strategies, such as proposed by Lowe (Lowe, 2004) in which the ratio of the best and second best match is compared to a threshold. For larger datasets, sequential approaches are appropriate for ordered images sets with consecutively captured images. For unordered datasets a vocabulary tree approach can be used (Schönberger, 2016) as well. Prior knowledge is also employed such as GPS coordinates in the EXIF data. Commercial software packages typically include multiple matching strategies depending on the input data.

3.2.3 Camera recovery and sparse point cloud reconstruction

The extracted image correspondences are used to estimate camera poses, camera internal parameters and 3D coordinates of image points, yielding a sparse point cloud. The computation is based on the inherent structure between two images: the epipolar geometry. Algorithms exist to compute this geometry from correspondences in the form of the fundamental matrix. If the internal parameters of the camera are known (approximately), the essential matrix, rather than the fundamental matrix, can be computed (Nistér, 2004), which requires fewer correspondences, is more stable and also directly yields a metric reconstruction, instead of a projective one that needs to be upgraded using self-calibration.

Two major pipelines can be distinguished to perform this step: incremental and global. An incremental pipeline is the most common approach that adds one image at a time, calculates the unknown parameters and thus grows the reconstruction. Due to a potential buildup of error, better known as drift, this process requires repeated operations of a bundle adjustment (Triggs, 2010) optimizing the camera parameters to minimize the reprojection error. This is a computationally expensive step that severely impacts the performance for large datasets. A global reconstruction pipeline follows an alternative approach and considers an entire view graph at the same time instead of incrementally adding images to the reconstruction (Jiang 2013). These algorithms estimate the relative orientation of all images in a single step by first computing the pairwise orientation of all image pairs and then combining this information into one graph. This way only a single iteration of the bundle adjustment is required at the end of the process. While much more efficient, this approach is more sensitive to outliers. To tackle the issues of efficiency, accuracy and robustness, recent efforts focus on the implementation of a hybrid reconstruction technique (Cui, 2017).

3.3 Dense matching

The result of the previous steps consists of the camera calibration and a sparse point cloud, containing the 3D reconstruction of the matched feature points. This point set is limited by design and is not a detailed or convincing representation of the filmed scene. Once a sparse representation of the scene has been completed, however, denser scene geometry may be recovered by matching as many pixels between images as possible. This process, called dense matching, is the most time-consuming part of the entire photogrammetric pipeline but can be sped-up by employing graphical processing hardware and parallel processing. Typical dense reconstruction pipelines produce depth maps from stereo pairs for all registered images. This relies on accurate exterior and interior camera parameters and epipolar geometry between images to constrain the search for matches (Remondino, 2017). Other methods include the use of region growing (Shin, 2010) or graph-cuts (Kolmogorov, 2002). The reconstructed sparse point cloud can be employed as seeding areas for the region growing or to constrain the search range for the dense matching. It can be beneficial in this respect to slightly clean up the sparse cloud (manually or automatically, for instance using masks) in order to decrease the processing time of the dense matching and improve its results.

3.4 Dense point cloud generation, modelling and texturing

Depth maps are subsequently fused into a dense point cloud. It is important to note that the information contained in the depth maps is often redundant, as SfM-compliant images are usually taken with large overlap. Unnecessary overlap can increase the computational complexity of the mesh generation step. Moreover, low-quality images (e.g. with motion blur) may produce low-quality depth maps that harm the mesh generation process. That is why a selection of the most suited depth maps can be made to generate the dense point cloud (Tingdahl, 2011).

The dense point cloud, a union of the points from the depth maps of the selected views, is then filtered, for instance using the quality maps (if available) of the dense matching or taking into account the angle between the normal and the view angle. A dense surface is then estimated from this fused point cloud, using surface reconstruction techniques, the

most common of which is Poisson reconstruction (Kazhdan, 2006). Finally the reconstructed mesh is colored using texture maps, that combine and blend the input images (Lempitsky, 2007).

4 PROCESSING VIDEO

4.1 Issues with processing video w.r.t. images

In V4Design we focus on the 3D reconstruction from video as well as images. The photogrammetric pipeline of chapter 3 is targeted towards the processing of images of the same scene. These images should be taken from different positions and should adhere to several important constraints that were listed and described in deliverable D4.1. Photogrammetric pipelines typically don't focus on processing video. In such cases, the user is typically instructed to extract separate frames from the video and feed these to the photogrammetric package.

There are three important issues with this strategy. The first issue is the fact that videos (especially those by a professional director) contain multiple and sometimes quickly varying shots. Secondly, due to the rapid recording of consecutive image frames and the motion of the camera, blurry frames are rather common. Finally, the extraction of optimal frames (so called keyframes) for the photogrammetric processing is an important issue.

The two first issues are discussed at length in D4.2. We will only briefly address them here. The third issue (keyframe extraction) will be discussed in detail. The final paragraph of this chapter will explain more details on the actual implementation for the V4Design pipeline.

4.2 Shot detection

In many cases videos and movies, directed by professionals, contain shots of multiple scenes and as such are not directly useful for photogrammetry. It is therefore important to pre-process these videos. In a preliminary step the various shots are first delineated and can then be further processed to extract proper frames for reconstruction (see D4.3).

The V4Design pipeline already offers an important advantage over other systems, in that the STBOL component will evaluate the videos and determine where interesting content, such as buildings, is recorded. This results in a set of frames and even masks in these frames that should be evaluated for 3D reconstruction.

However, it might be the case that the STBOL component misses some frames in which the object of interest is present, or (more common) that the video records other parts of the same scene before or after the highlighted frames. It is advantageous to include these frames in the 3D reconstruction processing because the camera calibration of a video sequence will be more accurate and complete if a larger area with more depth variation is filmed, and/or if the camera motion is larger, leading to larger baselines and therefore better intersection angles. That is why we want to extend the detected frames to the entire shot they are part of. This shot-detection algorithm is explained in this paragraph and makes use of FFMpeg (FFMPEG, 2019), an open source library that is used to handle video streams efficiently.

4.2.1 Hard cuts: SAD-score and Histogram

The first shot-detection algorithm we employ is straightforward but very fast. It makes use of a threshold-based scene detector to segment the shots from each video. To detect the various shots a *scene score* is calculated between consecutive frames, starting from the

frames that were highlighted by the STBOL component and going both forward and backward in time. The scene score is determined by the sum of absolute difference (SAD) between all pixels of consecutive frames. The resulting value varies between 0 and 1 and may be used as a measure for similarity between two consecutive image blocks. A video cut, and thus a new shot, is assumed when the scene score exceeds a certain threshold. Extensive testing was done on videos provided by the content providers and it was determined that a threshold of 0.3 was the optimal value.

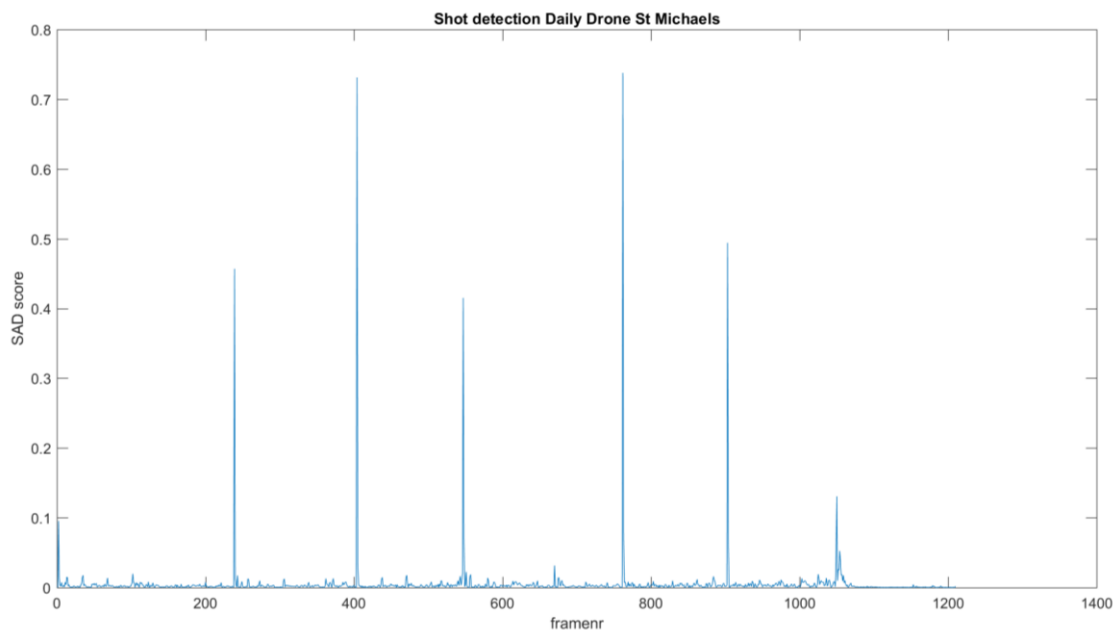


Figure 1: Shot detection using SAD on the Daily Drone video of St Michael's

A first example is a video from the Daily Drone series by Deutsche Welle, depicting video of the St. Michael's church in Hamburg. There are clear shot cuts in the video, at frames 1, 404, 547, 762 and 903. Figure 2 shows images around frame 762, where the shot cut is detected.



Figure 2: Consecutive frames 760, 761, 762 and 763 of the St Michael's church Daily Drone video. The scene cut between 761 and 762 is detected automatically.

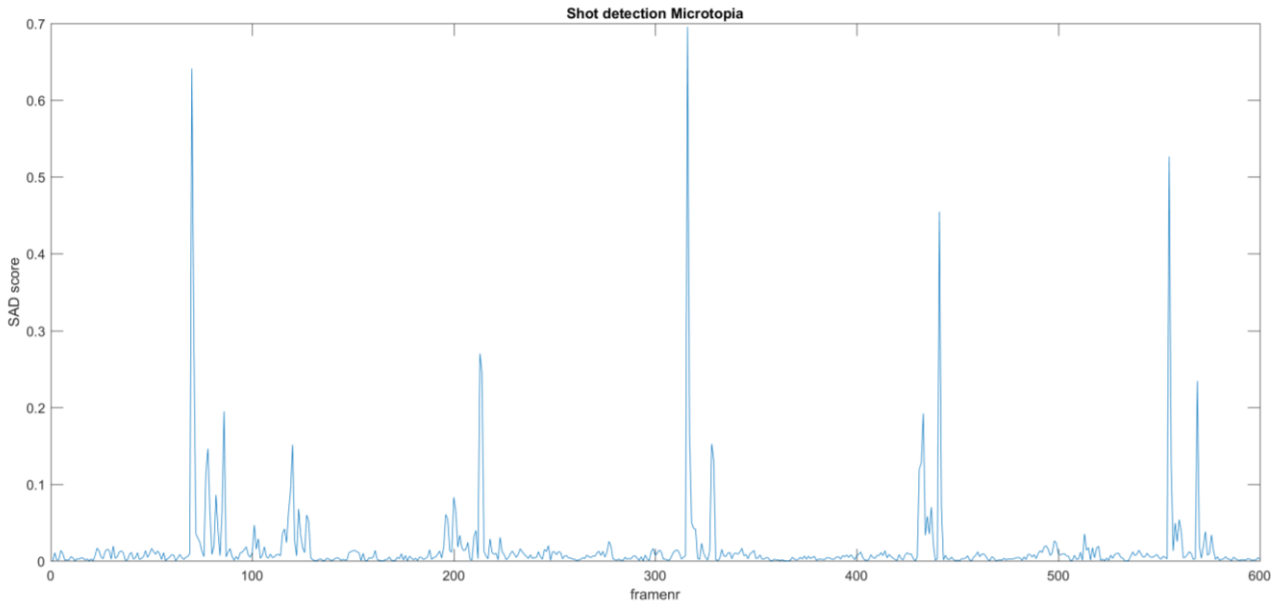


Figure 3: Shot detection using SAD on "Microtopia"

A less straightforward example is that of the train sequence in *Microtopia*, a movie by SLS. It shows footage, filmed on-board a train, filming city buildings. The shot cut detection results are shown in Figure 3 and clearly depict peaks that come in pairs. This is due to the specific content of the movie: the trains pass through underpasses, bridges and short tunnels, and our algorithms therefore detect a cut before and after these short passages. An example around frame 316 is shown in Figure 4. The first cut is detected between the first and second row (frame 316) and the second cut between the third and fourth row (frame 330).

Methods that are based on colour and intensity histograms are described in literature (Patel 1997, Tsekeridou 2001). These algorithms compute the histogram of the entire image and/or columns of rows and compare them to the next frame, using a chi-squared value, computed according to Equation 1 in which we sum the influence of every bin j of histogram H for frames t and $t+1$. However, our experiments clearly show a much better performance for the (slightly more computationally expensive) SAD method.

$$\chi^2 = \sum_j \frac{(H_{t,j} - H_{t+1,j})^2}{(H_{t,j} + H_{t+1,j})^2}$$

Equation 1: Computing the chi-squared value, comparing the histogram of frame t and $t+1$.



Figure 4: Frames of “Microtopia” train sequence with a shot cut between the first and second and between the third and fourth row

4.2.2 Fade cuts: Mutual Information

The SAD method of the previous paragraph works well for hard cuts. However, modern (edited) videos also regularly show smooth cuts with fade-ins and fade-outs. These cuts are not detected by the SAD method because the pixel information gradually changes over a length of several frames. A possible solution to this is to make use of an Information Theory-based method, such as the one proposed by Cernekova e.a. (Cernekova, 2006). In their method the mutual information (MI) and joint entropy (JE) between consecutive video frames is computed. MI is a measure for the correspondence between two sets of data and also takes into account the information that is carried by each frame at their overlap: MI increases when the amount of shared information is large. The JE is related to the MI, in that it is the sum of the entropy of both frames minus the MI.

The MI and JE of two consecutive frames can be computed using so-called *carrying matrices* that store the number of pixels that change from intensity i to intensity j for every possible value of i and j ($0 \leq i, j < 256$). The MI and JE are then computed as in Equation 2.

$$MI_{t,t+1} = - \sum_i \sum_j C_{t,t+1}(i,j) \log \left(\frac{C_{t,t+1}(i,j)}{C_t(i,j)C_{t+1}(i,j)} \right)$$

$$JE_{t,t+1} = - \sum_i \sum_j C_{t,t+1}(i,j) \log (C_{t,t+1}(i,j))$$

Equation 2: Computation of MI and JE values for two consecutive frames

In our experiments we find that the MI value is good at detecting hard cuts, but not as good as the SAD method, as can be seen from comparing Figure 5 with Figure 1. However, the JE value can be used to predict fade cuts. An example is shown in Figure 6 which contains the graph of the JE score for a video showing footage of the Cathedral Notre-Dame in Strasbourg. More than ten fade cuts are present in this video, several of which can be detected from the JE score by looking for sudden large increases in the JE value. Two examples of such fades are shown in Figure 7. Notice how these frames typically contain information from two shots and therefore result in a large JE score.

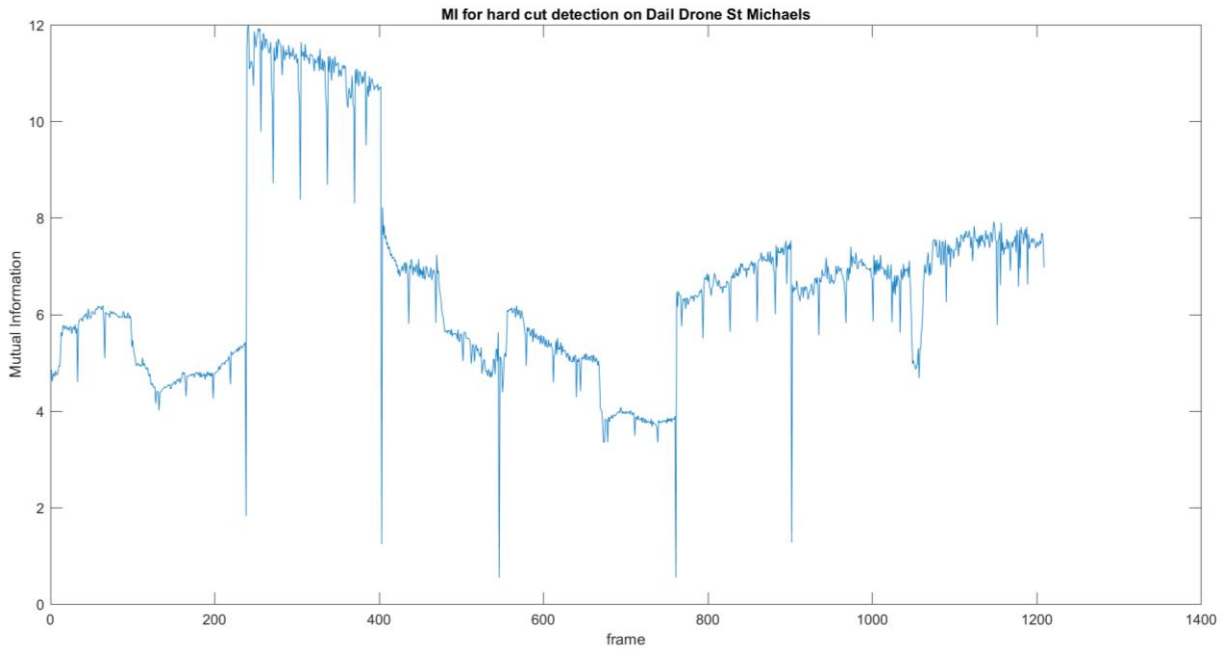


Figure 5: Mutual Information score for Daily Drone video of St. Michael's church

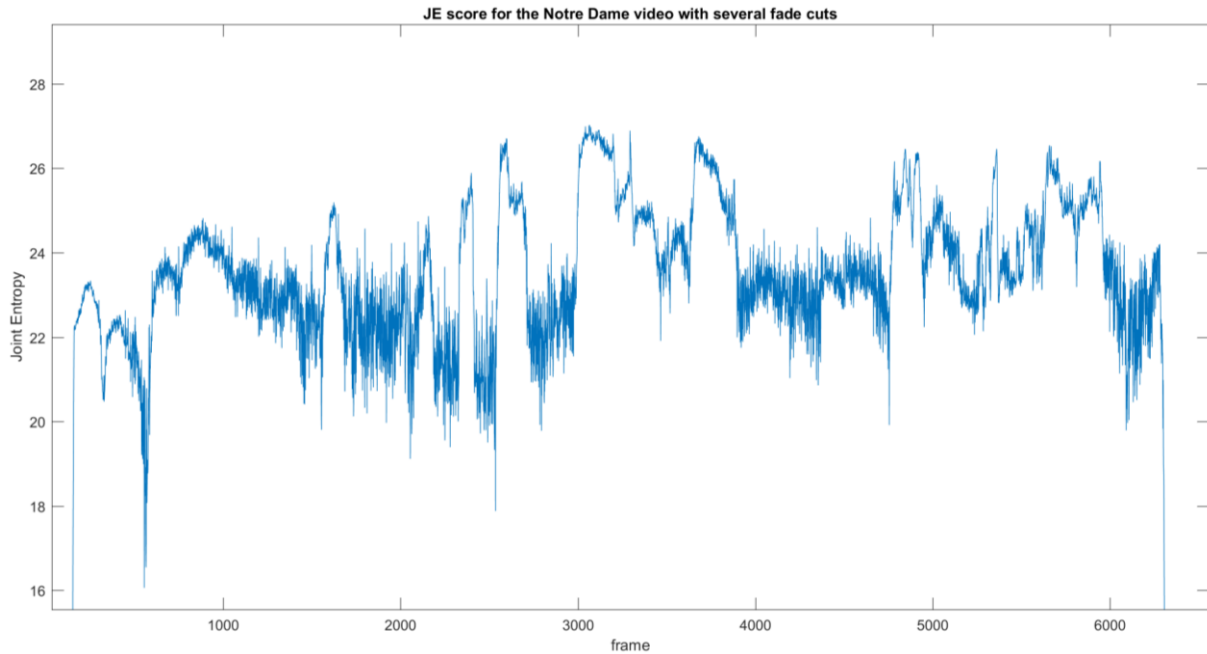


Figure 6: Joint Entropy score for the video “Notre Dame in Strasbourg” with several fade cuts that are visible in the curve



Figure 7: Frames 2994 and 4765 where a fade cut is detected. These frames clearly contain information from two different shots, yielding large increases in the JE score.

4.3 Dealing with blurry frames

When video sequences are recorded in a freehand style, one can never totally exclude the possibility of blurred frames. In most cases this blurriness is actually motion blur, which is the result of sudden motions of the camera. Blurred frames cause problems for photogrammetric reconstruction techniques, mostly because the amount of matched or tracked features drops dramatically when such a frame is encountered. It is therefore best to try to detect these frames beforehand and deal with them.

There are two options to deal with blurry frames. The simplest solution is of course to simply discard them before extracting keyframes. A second option is to incorporate their detection into the keyframe extraction algorithm itself. In both cases, however, it is imperative that we can discern blurry frames from their sharp counterparts.

A possible technique to spot blurred frames automatically is based on comparing the image sharpness of nearby frames. The value for the sharpness S of an image can for instance be

computed as the mean square of horizontal and vertical intensity gradients, evaluated as finite differences:

$$S = \frac{\sum_{i=0}^{W-1} \sum_{j=0}^{H-1} (I(i+1, j) - I(i, j))^2 + (I(i, j+1) - I(i, j))^2}{WH}$$

Equation 3: Sharpness of a frame from global intensity gradients

where W and H are the width and height of the image and I is the intensity value matrix of the image.



Figure 8: Frame 8156 and 8157 of the Akropolis video sequence. The second image clearly suffers from motion blur.

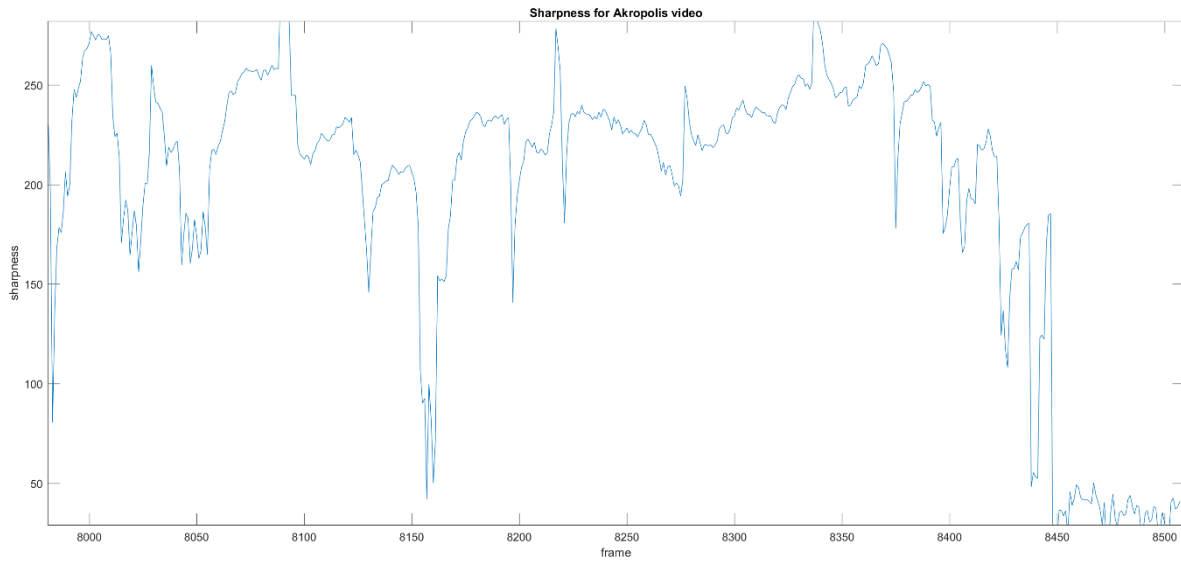


Figure 9: Sharpness and detected blurry frames for the Akropolis video sequence

The value of such a sharpness measure depends on the image content and thus no absolute threshold can be used to detect blurry frames. For example an image of a plastered wall of a modern building has a much lower absolute S-value than that of a wall of an ancient Greek temple. However, if one compares the sharpness of frames in each other's neighbourhood, then conclusions can be drawn. Due to motion blur, the gradient strength in blurred images is significantly lower than in the other images. Figure 8 shows two frames of the Akropolis video sequence. The second image suffers from motion blur, which is especially apparent when looking at the building, which contains a lot of high-frequency information. Figure 9 shows a plot of the sharpness values for the relevant frames of this video sequence. A

significant drop is noticeable around frames 8157 and 8161. The images in Figure 8 depict frames 8156-8157 where the drop in sharpness happens. The blurred frames are detected and can be removed from the sequence before further processing.

Another example deals with video from content providers. DW's DailyDrone video of the Bauhausuniversität in Weimar was recorded during a cloudy day, which leads to longer opening times of the camera in order to capture enough light. However, longer opening times and sudden motions lead to motion blur, causing sudden drops in the sharpness value. This is apparent from the sharpness curve in Figure 10. The sudden drops in this plot all correspond to blurry frames, an example of which is shown in Figure 11 and Figure 12.

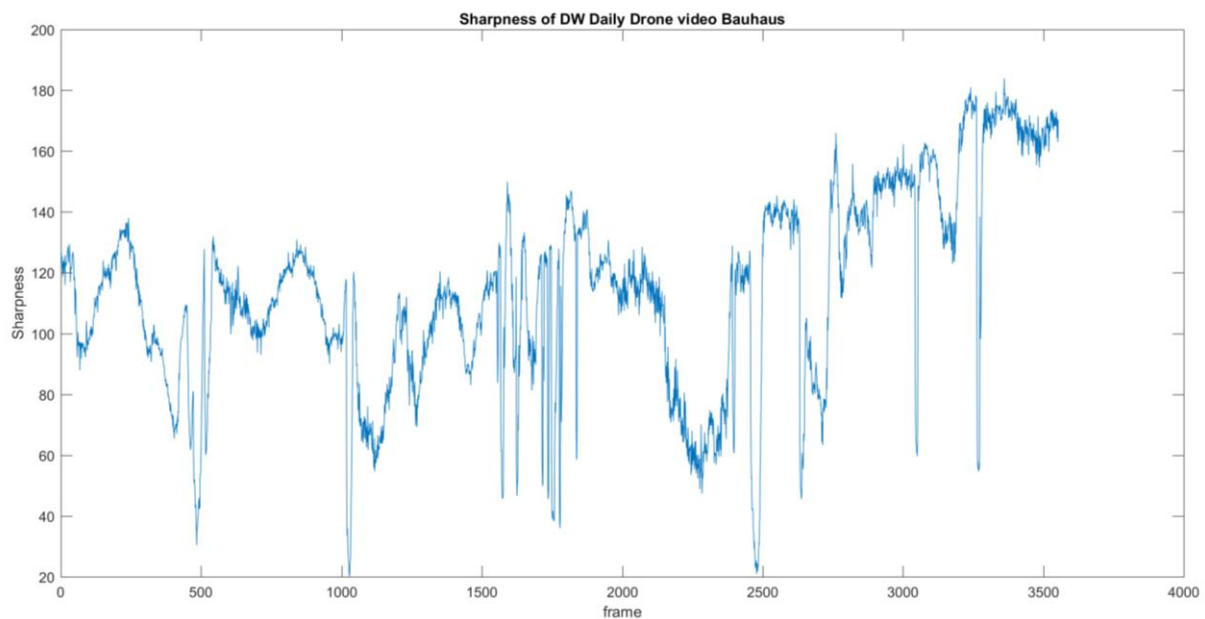


Figure 10: Sharpness of DW's Daily Drone video of the Bauhaus in Weimar. Several sudden drops indicate blurry frames, caused by motion blur.



Figure 11: Consecutive frames 1017 and 1018 from DW's Bauhaus video, where a sudden drop in sharpness was detected. The right image is blurred.



Figure 12: Cut out details of frames 1017 and 1018, showing a sudden motion blur.

An alternative approach also makes use of the image content but, instead of using a global value as in Equation 3, makes use of the results of a feature detection algorithm. The detector we look for must adhere to the following specifications:

- It must be a very efficient detector, because the detection of blurry frames is only a pre-processing step, must be performed for all frames and therefore should be concluded as quickly as possible.
- It should be a detector that focuses on local intensity changes. This rules out detectors such as SIFT or SURF because these typically extract blob-like features, spanning areas with continuous intensity.
- It should be a detector that allows for many features in the same neighbourhood and should therefore not suppress multiple detections in small areas.

Based on the above specifications we chose the FAST (Rosten, 2010) detector, an algorithm proposed by Edward Rosten.

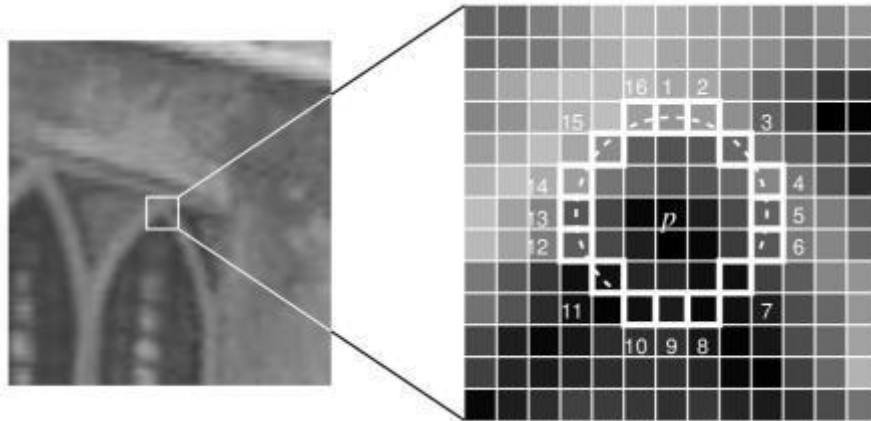


Figure 13: FAST feature detection algorithm (Rosten, 2010)

The algorithm (shown in Figure 13) starts from the premise that a pixel p is a corner feature if there exists a set of n contiguous pixels in the circle (of 16 pixels) which are all brighter than $I(p)+t$, or all darker than $I(p)-t$ (with t a given threshold). The algorithm speeds up the test by examining only the four pixels at 1, 9, 5 and 13. If p is a corner, then at least three of these must all be brighter than $I(p)+t$ or darker than $I(p)-t$. The full segment test criterion can then be applied to the passed candidates by examining all pixels in the circle. This algorithm clearly uses local intensity information and is very efficient: it reaches speeds of 10fps or higher. The third requirement is achieved by omitting the standard non-maximal suppression step that typically follows the algorithm to select the best feature amongst its neighbours in a region. The result is a detector for which the number of detected features is a very good indication of the blurriness of the image, especially compared to neighbouring frames.

An example is shown below. Figure 14 shows the detected features in a frame of the *Bauhausuniversität* video from DW. It is clear they are located on local intensity gradient maxima. Figure 15 shows a graph of the number of detected FAST features for the entire video sequence. We clearly detect significant drops that correspond to the drops in Figure 10 but are sometimes more outspoken. Figure 16 shows frame 484 where such a drop occurs. It clearly suffers from motion blur.

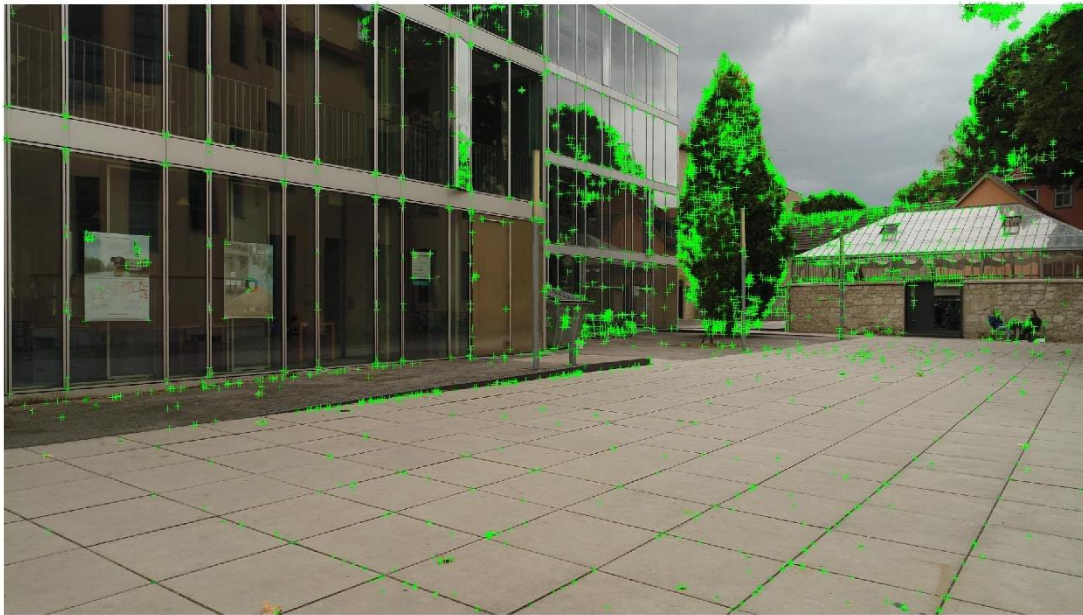


Figure 14: Extracted FAST features in frame 1356 of the Bauhausuniversität video

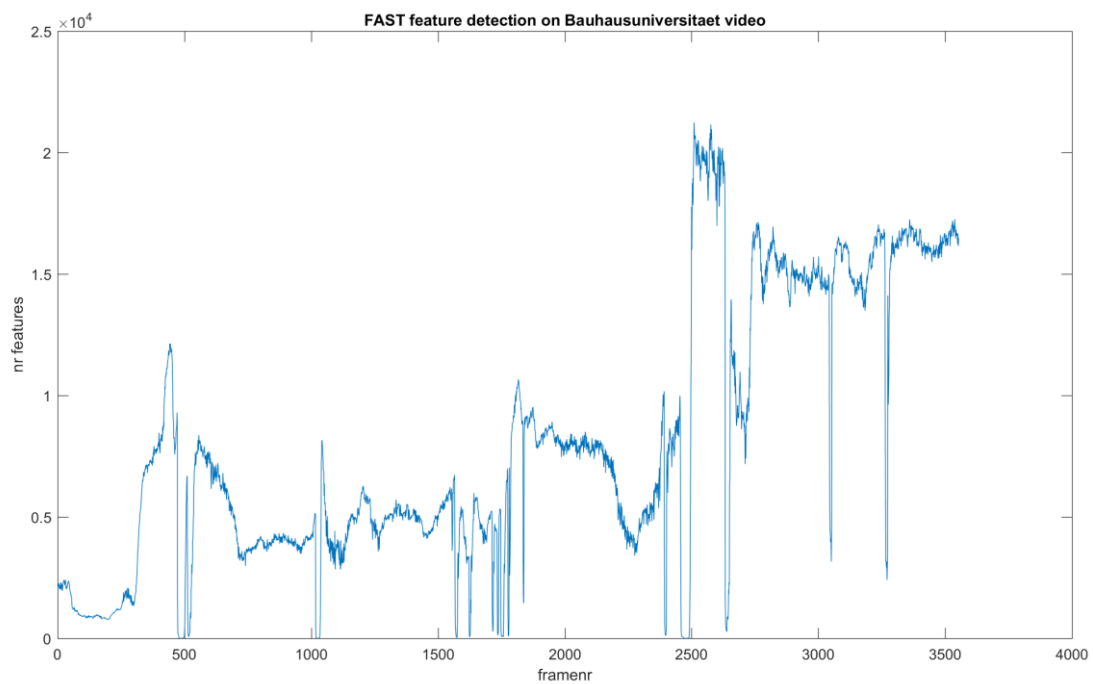


Figure 15: FAST Feature detection yields a measure of sharpness for every frame.



Figure 16: Frame with plenty of motion blur, detected by the blurriness algorithm, based on the FAST feature extractor.

4.4 Image and video sequence analysis and keyframe selection

4.4.1 Video and photogrammetric reconstruction

Video recordings yield image sequences with frame rates of 25 (PAL) or 30 (NTSC) frames per second. If the video camera does not move at very high speed (which would cause blurring effects anyway, more on which was written in paragraph 4.3) this means that consecutive video frames are typically very close together and thus very much alike. This gives us an important advantage for the computation of matching points between images because such points are typically located close to the coordinates in the previous image and very little distortion has to be dealt with. This makes it possible to either limit the search range for matching points or to employ a different strategy in which we track features rather than match them. Both strategies are valid and fast because we can make use of features that can be extracted efficiently, such as FAST-features for matching or KLT-features (Shi, 1994) for tracking.



Figure 17: Two consecutive frames of the video sequence *Bauhaus* from DW

Unfortunately, the fact that consecutive images are so similar also has its drawbacks. Inspect Figure 17. This figure shows two consecutive frames of a video sequence of Deutsche Welle's Daily Drone video, depicting part the *Bauhaus*. If we attempt to compute the epipolar geometry between these images we notice that this process is very sensitive to noise.

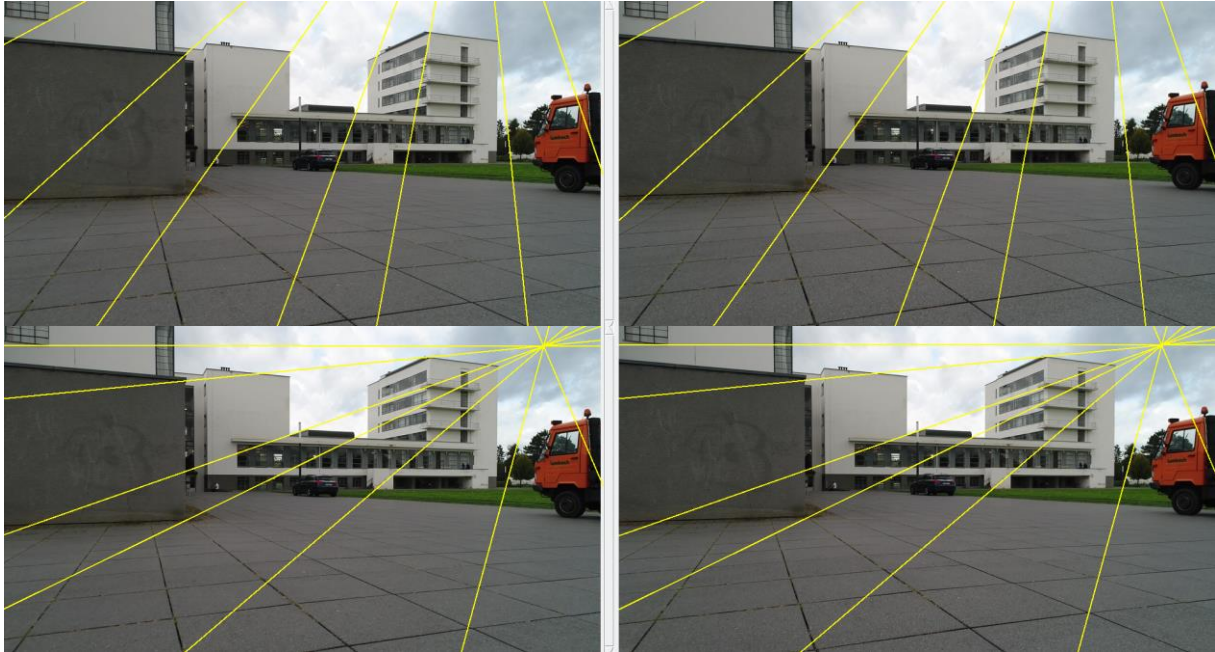


Figure 18: Two possible solutions to the Fundamental Matrix that are equally valid. The yellow lines show the epipolar lines indicating the camera movement.

Two possible solutions of the epipolar geometry are shown in Figure 18 and both are equally valid. To verify this, inspect the easily identifiable points through which the epipolar lines go (for instance: corner of a building, intersection on the pavement, ...) and verify that the corresponding point in the other image lies on the corresponding epipolar line. This is the case for both solutions of the epipolar geometry, even though they differ very much. The images shown in this figure depict a scene with plenty of 3D information. Yet we cannot compute the epipolar geometry stably because of the small baseline: the two camera centres are too close together, so the translation between the cameras is very small. This means that the camera setup resembles that of a purely rotating camera and hence the relation between corresponding pixels can also be explained by a homography as shown in Figure 19.

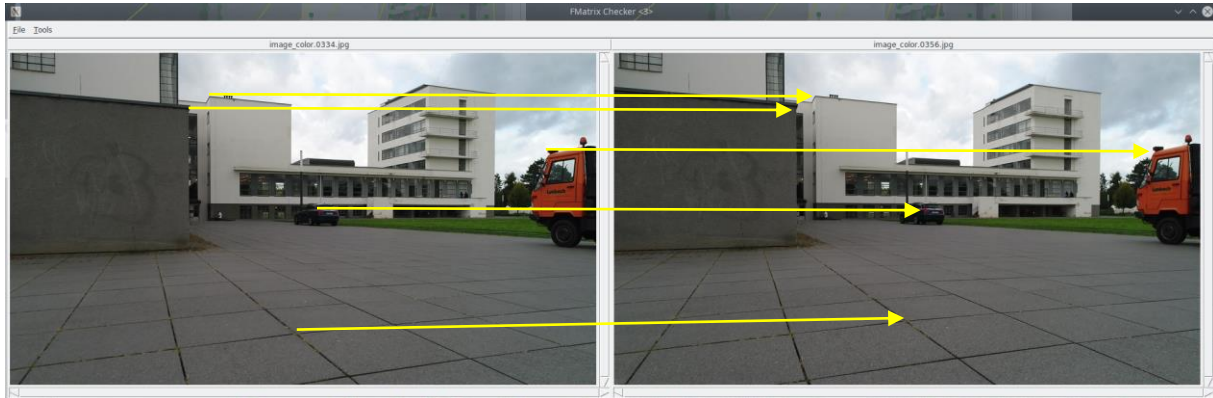


Figure 19: The motion between the consecutive frames can be explained by a homography.

The computation of epipolar geometry breaks down if images are too close together. The same is true for projective camera estimation where newly computed 3D points and cameras suffer from large uncertainties due to the small baseline between cameras. This means we have to find a technique to increase the baseline between images that are used for structure-and-motion recovery. A possible solution is not to deal with consecutive images but to select every n -th frame of the video sequence. This is not a good idea because it is impossible to choose a value of n that will work for all video sequences. It would depend on the velocity of the camera, the frame rate and the distance to the scene. These properties could even change during the recording of the video. Imagine a camera filming a scene, standing still every once in a while and then continuing. The value of n would have to be adapted constantly. A possible solution for this makes use of the concept of the Geometric Robust Information Criterion (GRIC), introduced by Torr (Torr, 1997).

4.4.2 GRIC

The concept of GRIC was introduced to solve a particular problem: how can we find out if a certain scene is dominantly planar and does this have an impact when computing the epipolar geometry of an image pair depicting such a scene? In this case, the transformation between the images can be described by a general projective transformation, modeled via a homography H . Since every point correspondence yields 2 equations (one in x and one in y), only 4 points are needed to compute H . All other correspondences can be predicted from these 4. This would suggest that only 4 independent point matches can be used to compute F but it is known (Hartley, 2004) that we need at least 7 correspondences. This means that there are 3 DOF in F we cannot determine without information of points outside the plane. If no such points are available, these remaining DOF are filled by noise on the extracted features.

A similar situation arises when the baseline between two images (of any scene) is small, because this situation resembles a setup with a purely rotating camera, which can also be described with a homography. The top and bottom image-pairs of Figure 18 depict the same images but with a different fundamental matrix. Both matrices were computed with the same RANSAC algorithm. The only difference is a small perturbation on one of the parameters (i.e. the outlier distance which was changed from 2 to 2.2 pixels). The resulting F -matrices are quite different which is apparent from the epipolar lines. If one compares the epipolar lines in any of the image pairs however, points on corresponding lines are always in correspondence. This means that both solutions are equally valid and no claim can be made

as to the quality of the result. One can intuitively understand this as follows. Take an arbitrary pencil of lines in one image. If a homography describes the relation between the two images, then every line can be transformed to the other image and all points on the line in the first image will lie on the transformed line in the other.

A first step in solving the small baselines problem is the ability to detect this situation. Since such images can be transformed into each other via a homography, one could compute such H matrix and inspect the residual error. If this error is below a threshold, one could consider the baseline to be too small. However, it is unclear which value one should choose for this threshold. A better and more stable approach is to take into account not only the residual error but also the degrees of freedom. This can be done by computing and comparing values of the Geometric Robust Information Criterion for different models.

The problem of determining whether the baseline between images is too small, based on correspondences between these images is a specific example of a more general problem, which can be stated as follows. If one has a data-set and different models that can explain the data, then which model should one choose? Already in the middle ages a Franciscan friar called William of Ockham (Arriew, 1976) shed some light on this. His theory, known as Occam's Razor states:

One should not increase, beyond what is necessary, the number of entities required to explain anything.

In essence this theory means that if two or more theories explain the observations equally well, one should always choose the simplest one. For the case of multiple view geometry, Torr proposed a mathematical description of this principle that is completely general, based on Akaike's information criterion (Akaike, 1974) and Rissanen's minimum description length (Rissanen, 1978). It calculates a score function for each model taking into account the number of inliers, outliers, the residuals, the standard deviation of the error, the dimensionality of the data, the number of the parameters and dimensionality of the model. The general formula is given by Equation 4

$$GRIC = \sum \rho(e_i^2) + (nd \ln(r) + k \ln(rn))$$

$$\rho(e_i^2) = \min\left(\frac{e_i^2}{\sigma^2}, 2(r - d)\right)$$

Equation 4: Geometric Robust Information Criterion

In which

- n is the number of data (inliers + outliers)
- e_i is the residual for every correspondence i . For a homography H for instance this is the distance between point m_2 and the transformed corresponding point Hm_1 :

$$e_i = D(Hm_1, m_2) + D(H^{-1}m_2, m_1)$$

For a fundamental matrix F the error is computed as the sum of the distances to the epipolar lines:

$$e_i = D(l_2 = Fm_1, m_2) + D(l_1 = F^T m_2, m_1)$$

- σ is the standard deviation on the measurement error.
- r is the dimensionality of the data or the amount of observations. In the case of 2 views, r is 4 (two times two coordinates).

- k is the number of model parameters. For a homography k is 8, for a fundamental matrix it is 7. For an essential matrix it is 5.

Inspecting Equation 4 we clearly identify two parts. The first part is the goodness of the fit. How well does a model describe the data? This is given by the residual error of the corresponding matches. Of course, this residual alone does not suffice since a complicated model that contains a simpler model always explains the data better since it has more degrees of freedom. In the case of a small baseline for instance, the 3 remaining degrees of freedom of a computed fundamental matrix are estimated from noise in the data, allowing for a lower residual score.

That is why the GRIC model has a second part $nd \ln(r) + k \ln(rn)$ which takes into account the so called *parsimony* of the model. This means that the more complex a model is, the higher this penalty term will become. If we want to choose a more complex model over a simpler one, it should explain the data much better, to the extent that its residual error should be so much lower than the residual error of the simpler model that this reduction nullifies the extra cost associated with the higher complexity.

Let us apply this algorithm to the problem of a pair of images with a small baseline. We execute the following algorithm.

- Compute the fundamental matrix between the two images, as well as the inliers and outliers using F-RANSAC.
- Compute a non-linear optimization of F using the inliers only, e.g. with a MLE (Maximum Likelihood Estimator) (Aldrich, 1997)
- Compute a planar homography matrix H .
- Compute H-GRIC and F-GRIC with Equation 4 using all the inliers of F . The latter is absolutely necessary because if only inliers for H are used to compute it, the H-GRIC value will always be lower than the F-GRIC because of its lower complexity.
- Compare both GRIC values. If H-GRIC is lower than F-GRIC, then the preferred model to explain the data should be H . If the F-GRIC value is lower, then there was sufficient camera motion to compute the F -matrix correctly.

4.4.3 Selecting keyframes

We can employ the GRIC algorithm, explained above, to solve the problem of selecting the optimal frames in a video sequence for photogrammetric reconstruction. To do so we introduce the concept of *keyframes*. We select the first frame in the sequence as the first keyframe and track features to the second frame. We estimate the fundamental matrix and a homography between these frames and compute the corresponding F-GRIC and H-GRIC values. If the H-GRIC is lower than F-GRIC, the homography describes the data better than F and we track the features to the next frame in the sequence where we perform the same process now between the first and the third frame. As long as the H-GRIC is lower, we continue to advance in the sequence. Once the F-GRIC yields the lowest penalty, the epipolar geometry can be reliably estimated and we could select the frame where this happens as the new keyframe. Note: in case the internal camera calibration parameters are known (or can be reliably estimated), the essential matrix E can be computed instead of the fundamental matrix F , which can avoid problems with certain degenerate cases.

However, in order to reduce the processing time of the photogrammetric camera estimation and the dense matching, we might want to progress a little further in the sequence. For every frame we track the features and estimate epipolar geometry w.r.t. the previous keyframe. While more than 90% of the inliers at the crossing-point are still matched, we continue. When we drop under this 90%, we select this frame as the new keyframe.

4.4.4 Detecting degenerate sequences

Deliverable D4.1 (Empirical study of visual content) described in detail how sequences with degenerate motion types are not suitable for 3D reconstruction. The most common of these are movies with a pure rotation around the optical center, so-called *panorama* sequences. Since there is no baseline between the frames, it is mathematically impossible to extract 3D information from these sequences photogrammetrically. However, it was also shown in D4.1 that these sequences are quite common, especially in older material. The content from the content providers in V4Design also contains several examples of such data. It would be beneficial if we could detect such unsuitable movies early and notify the user. This can be achieved with the same GRIC-based algorithm described above! The relation between images in panorama sequences can mathematically be described by a homography: this 2D projective transformation suffices to capture the change from one panorama image to another. Since the parsimony of the homography model is lower than that of a fundamental matrix (corresponding to 3D motion), the homography will be chosen for all images in a panorama. If no turning point between H- and F-GRIC is detected (i.e. the H-GRIC is always lower), we can deduce that the camera underwent a panorama-like rotation and that the movie is unsuitable for 3D reconstruction.

4.5 Implementation and examples

Commercial as well as open source photogrammetric solutions primarily focus on processing images, rather than videos. The V4Design project also wants to reconstruct 3D models from video sequences from a wide variety of sources. When dealing with video, photogrammetric packages typically instruct the user to extract every n^{th} frame and feed the resulting images to the reconstruction algorithms. We already showed earlier in this chapter that this is not a good solution.

4.5.1 Implementation

The implementation in the V4Design pipeline combines the different building blocks that were discussed in D4.1 and in this deliverable. It follows the pipeline depicted in Figure 20. In a first stage, the video is split into different shots, using the shot-detection algorithms of paragraph 4.2. This ensures that no hard cuts are present in the video to be processed. Secondly, the separate shots are fed to a block that analyses the sharpness of the consecutive video frames, as discussed in paragraph 4.3. Only frames that are clearly blurry, i.e. showing a clear drop in sharpness, are flagged and eliminated from the remainder of the processing pipeline. The next step then deals with the extraction of suitable keyframes that can be fed to the photogrammetric algorithms.

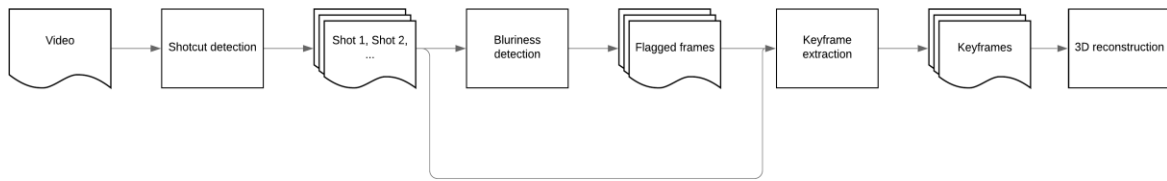


Figure 20: Video handling pipeline

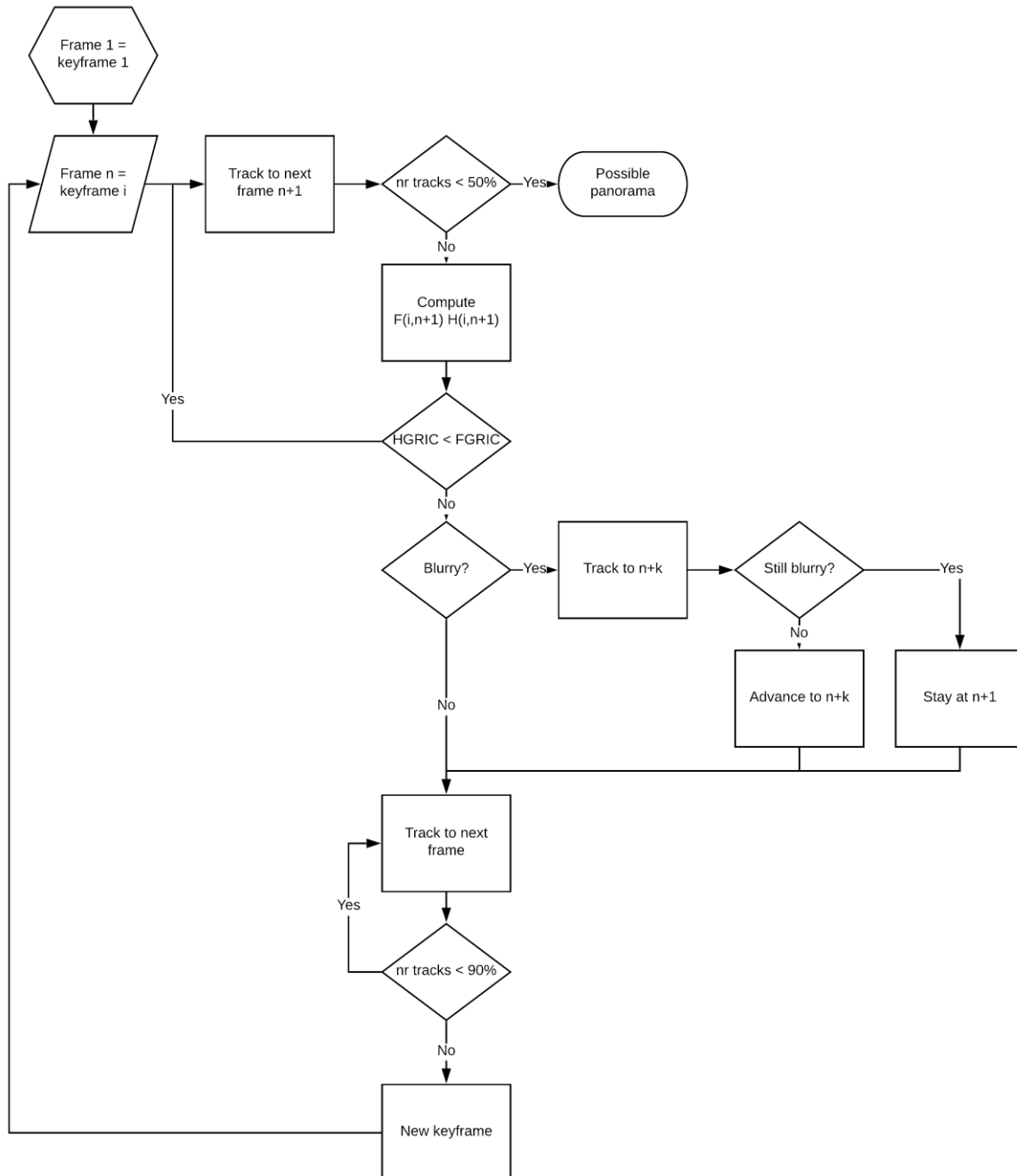


Figure 21: Detailed keyframe extraction

This extraction follows the logic described in paragraph 4.4. The procedure is depicted in Figure 21. It employs a tracking strategy, rather than a matching strategy, in which we exploit the fact that consecutive video frames are typically very close together. The selection procedure is based on the GRIC score, in the sense that a new keyframe is only selected when there has been sufficient camera motion. In order not to limit the number of keyframes, we keep tracking the features and select a new keyframe when the number of tracked features falls below 90%. Two additions to this basic strategy are also implemented. First, if we notice a sudden drop in tracked features from frame n to frame $n+1$, we check the local sharpness score. In many videos such lower sharpness is due to motion blur, which typically lasts for several frames and hence might not be detected in the blurriness detector. If frame $n+1$, or its k successors, show a significantly lower sharpness than frame n and frame $n+k$, we attempt to track the features from frame n to frame $n+k$ in order to overcome the blurry part of the video.

A second addition is the detection of degenerate motion. As described in paragraph 4.4.4, a pure rotation (panorama recording) is not suitable for 3D reconstruction. However, if this panorama-shot is short and is preceded and followed by actual camera motion, photogrammetric pipelines are able to deal with this, by reconstructing the preceding and following parts in 3D and estimating the panorama cameras afterwards. That is why we implemented an algorithm that is shown as part of Figure 21. The tracked features are fed to the keyframe selection algorithm as explained above. If the video only contains a panorama, no frame will be found where the F-GRIC-score is lower than the H-GRIC score. Such videos are automatically flagged. When we do find a crossing point between the GRICs (indicating a 3D motion), we compare the number of tracks to the features extracted in the previous keyframe. If this is above a certain threshold (currently set at 50%), we accept the new keyframe. Even in the case of a short intermediate panorama, the photogrammetric pipeline will still be able to reconstruct the scene. If the number of tracks falls below the threshold, we decide that this panorama part is too large for the 3D reconstruction to succeed.

4.5.2 Bauhaus - Dessau example

The first example we will discuss is from a *Daily Drone* video from content provider Deutsche Welle. It depicts the Bauhaus in Dessau, Germany. The video is about 1 minute long (1548 frames) and contains several pauses as is clearly shown in Figure 22, which depicts every 10th frame of the video. A simple strategy to extract every n^{th} frame of this video is doomed to fail.

The first operation we employ is a shotcut detection. In this case, the video consists of a single shot, which is subsequently fed to the module responsible for computation and analysis of the sharpness of the video. The result is shown in Figure 23. Several frames show a sudden drop in sharpness, indication of blurriness. This for instance the case for frame 32, depicted in Figure 24.

This information is now sent to the actual keyframe extraction algorithm. This procedure, explained above, first computes the tipping point where the H-GRIC score exceeds the F-GRIC score, indicating the point where the 3D motion of the camera is too large to describe the resulting images with a homography. From that point on, we keep tracking the features until less than 90% survives, at which point we select a new keyframe. Figure 25 shows an example around frame 750.

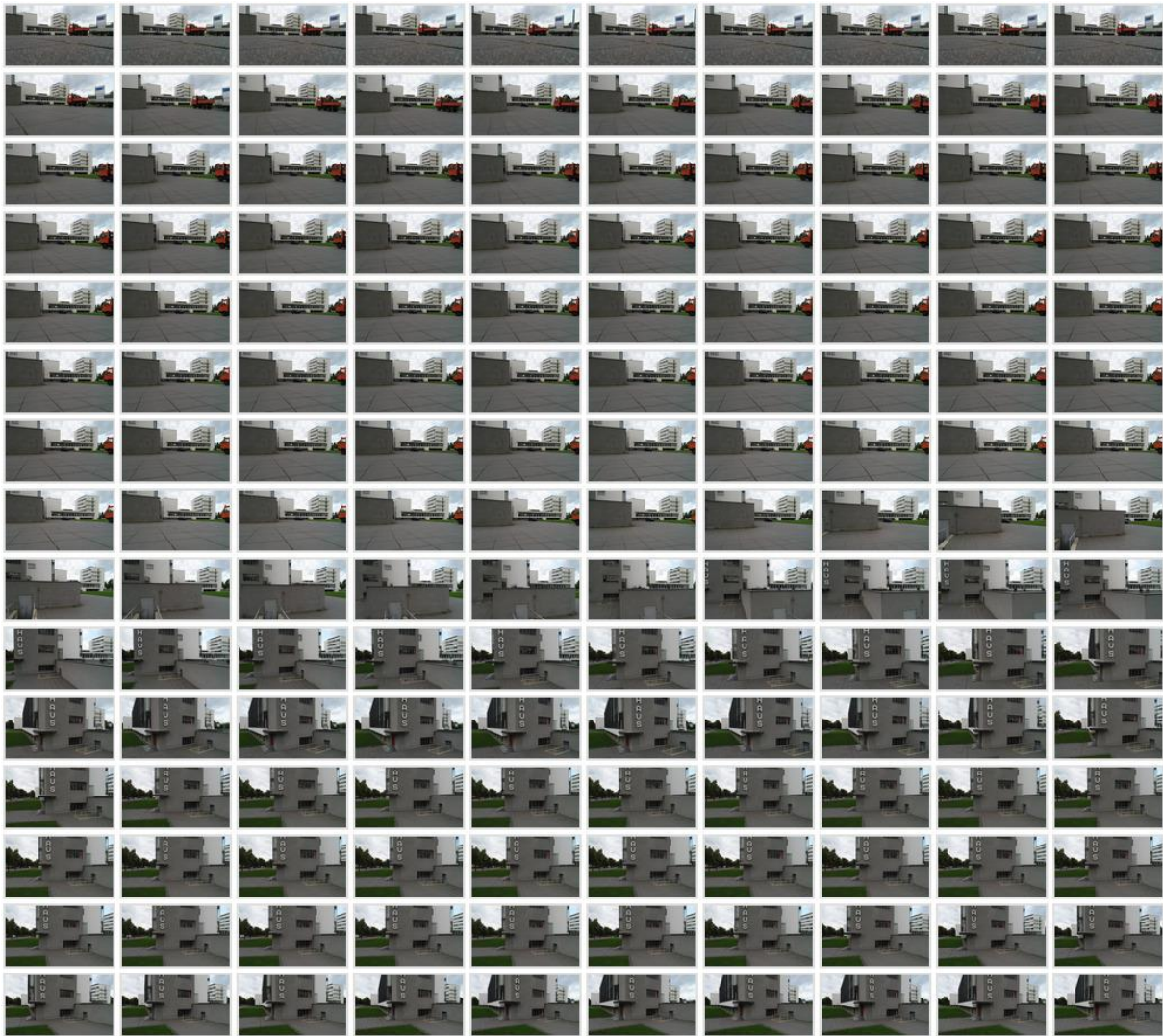


Figure 22: Every 10th frame of the *Bauhaus* video from Deutsche Welle

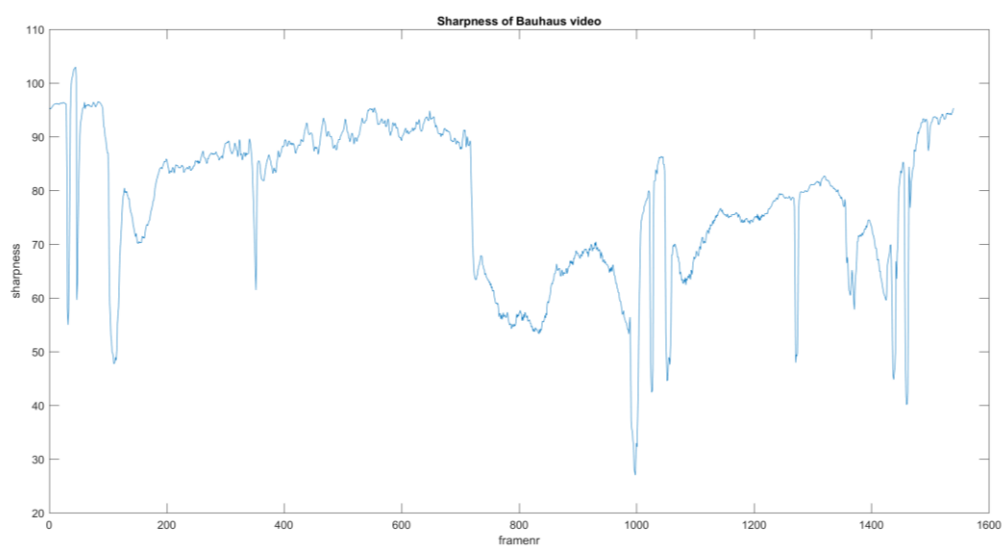


Figure 23: Sharpness of the *Bauhaus* video, showing several blurry frames



Figure 24: Blurry frame, detected at frame 32

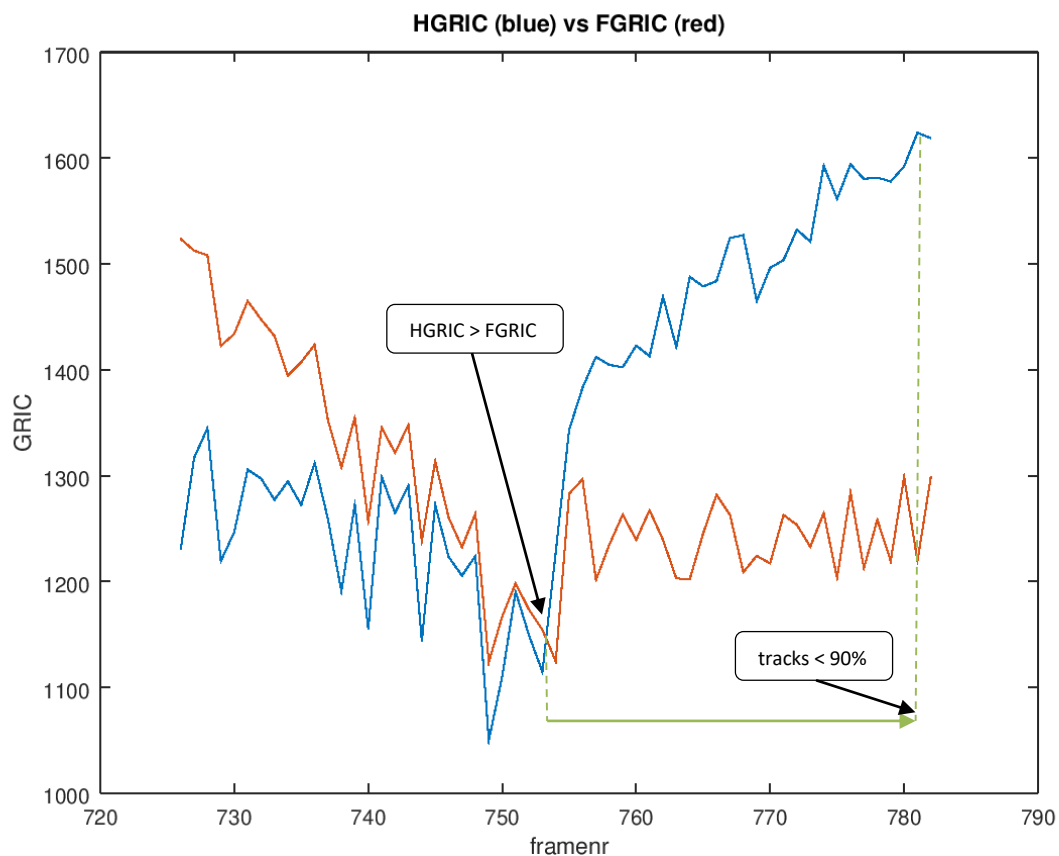


Figure 25: F-GRIC (red) and H-GRIC (blue) score around frame 750

The previously extracted keyframe is frame 725. When features are tracked and GRIC scores are computed, the H-GRIC score is lower than the F-GRIC at first, indicating little motion. At frame 754 we see the tipping point where $H-GRIC > F-GRIC$. These two frames are shown in

Figure 26. Note that especially the foreground (ground) differs and that there is parallax between the front wall and the buildings in the background, which is an indication of 3D information and motion. However, frame 754 is still rather close to 725 and we could probably skip more frames without jeopardizing the 3D reconstruction. Therefore we continue to track until 10% of the tracks of 757 have disappeared. This happens at frame 782, which is selected as the new keyframe. In the end, the algorithm extracted 27 keyframes, shown in Figure 27.



Figure 26: Tipping point: H-GRIC > F-GRIC at frame 754



Figure 27: Selected keyframes for the *Bauhaus* video

These keyframes are fed to the 3D reconstruction algorithm. The result is shown in Figure 28.



Figure 28: Resulting 3D model from the keyframes of the *Bauhaus* video

4.5.3 Kochuu: example of panorama

A second example we will describe shows the result on a video with a degenerate motion sequence. The video *Kochuu* from Solaris Film Productions contains several shots that might be of interest to V4Design users, but that are, unfortunately, not suitable for 3D

reconstruction purposes. An example of such is shown in Figure 30 where we depict every 10th frame of the shot. The movie shows a built structure around a pond. The camera rotates on the spot and hence performs a panorama motion, unsuitable for 3D reconstruction.

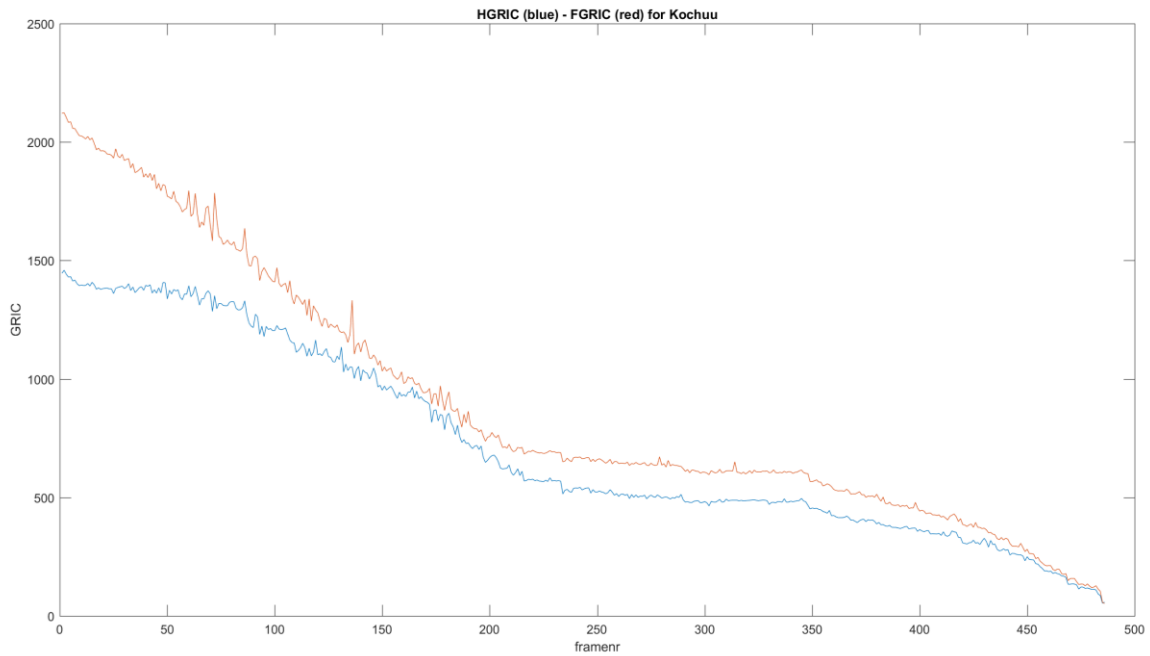


Figure 29: H-GRIC (blue) and F-GRIC (red) of the panorama shot of *Kochuu*.

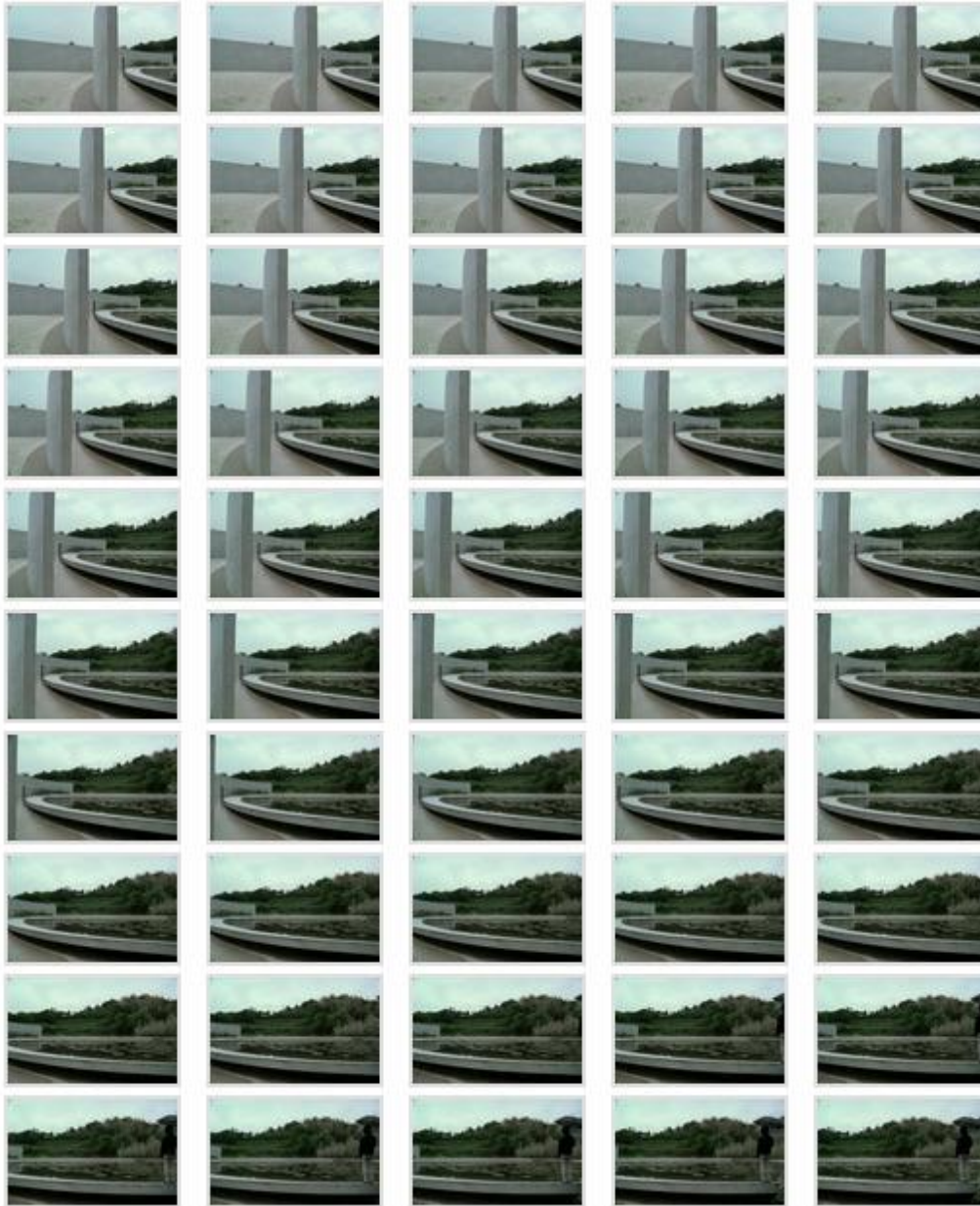


Figure 30: Every 10th frame of a shot in the movie *Kochuu* from SfP

The keyframes module computes and compares the H- and F-GRIC. Results of this are shown in Figure 29. Because the motion is degenerate, the algorithm should always prefer the H-model over the F-model because it is less complex. This expected result is indeed achieved, because the H-GRIC value always lies below the F-GRIC. Note that the absolute value of both GRICs steadily decreases because the number of tracked features decreases as well: indeed, at the end of the shot, no more features are present that were visible in the first frame. This indicates that this shot is unsuitable for 3D reconstruction.

5 3D RECONSTRUCTION: IMPLEMENTATION AND COMPARISON OF PHOTOGRAMMETRIC RECONSTRUCTION SOLUTIONS

The following chapter describes the State-of-the-Art of available photogrammetric software. The goal is to investigate whether existing components can be used to extend the State-of-the-Art. An empirical study is conducted with realistic data from V4Design to evaluate the different steps of the reconstruction pipeline of each software package.

5.1 General overview

Several photogrammetric software packages are available, ranging from semi-automated to fully automated procedures. This study focuses on the evaluation of methods that fully-automatically produce point cloud data, mesh geometry and high quality textures. More specifically, the emphasis of the work is on determining the best-fit methods to process the raw imagery and videos discussed in Deliverable 4.1 to a set of 3D outputs. This includes the reconstruction of objects and buildings from both close-range and oblique imagery.

5.1.1 Software

Both Open source software as well as commercial applications are considered. While numerous photogrammetric libraries and software exist, it suffices to evaluate only the best performing solutions from the industry and the academic community. The selection was limited to general-purpose reconstruction pipelines as the raw imagery and videos can originate from varying sources. For the evaluation of the commercial software Metashape (Agisoft, 2018), RealityCapture (CapturingReality, 2016) and 3DF Zephyr (3Dflow, 2014) are selected based on their performance. Another popular software, Pix4D (Pix4D, 2011), was not evaluated as it does not support command line interfacing which is necessary for the integration of the 3D reconstruction in the V4Design pipeline. For the evaluation of the Open source software, COLMAP (Schönberger et al., 2016) and Meshroom (AliceVision, 2019) are considered.

5.2 In-depth comparison of software tools and libraries

In this section, the advantages and disadvantages of the tools in the available software are discussed. More specifically, the opportunities are discussed to integrate functionalities of the different software into the V4Design reconstruction pipeline.

Table 3: Overview user adjustable tools of the evaluated photogrammetric software.

	Video Import	Adjust Feature Extraction	Adjust Camera Model	Adjust Matching Method	Allow Masking	Adjust Bundle Adjustment	Allow Segmentation	Filtering functions	GCP integration	Batch Processing
MetaShape	+	-	+	+	+	-	+	+	+	+
3D Zephyr	+	-	-	+	+	+	+	+	+	+
RealityCapture	+	-	+	+	-	+	-	-	-	+
COLMAP	-	-	+	+	-	+	-	-	-	+
Meshroom	-	+	-	+	-	+	-	-	-	+

5.2.1 List and details of compared tools

An initial comparison between the software is in their basic functionalities and parameter accessibility. Figure 31 depicts a set of useful functionalities to the reconstruction pipeline. For instance, batch processing is a necessity since the V4Design procedure will be simultaneously used by numerous users and content providers. The reconstruction box, masking and filtering capabilities reduce outliers and mitigate noise. The ability to influence feature extraction, camera models, matching methods, bundle adjustments and integration of Ground Control points is essential to produce proper models from the high variance data inputs from the content providers.

Table 3 depicts the ability of each software package to influence the respective parameters. As expected, do the commercial applications embody significantly more functions than their Open source counterparts. However, the Open source solutions allow more parameters to be influenced by the user. An interesting aspect is that some of the functionalities of Meshroom and COLMAP are complementary. This is especially useful for the V4Design 3D reconstruction pipeline since functions from both libraries can be combined to extend the current State-of-the-Art. The separate functions of the SfM pipeline are further investigated in the following sections to evaluate which code can be applied to the V4Design project.

5.2.2 List and details of compared methods

The functions of each software package are evaluated for the subsequent steps in a general-purpose SfM pipeline (Table 1). This includes the feature extraction, matching, alignment, dense reconstruction, meshing and texturing as discussed in paragraph 3. These functions are dependent on the type of SfM that is being employed. As previously stated, incremental SfM's are very popular. Typically, the iterative process is initialized by aligning as many images as possible in a graph followed by a geometric verification. The resulting scene graph is used to start the reconstruction. Incrementally, a two-view reconstruction is selected, new images are registered, new scene points are triangulated and outliers are filtered. At the end of each iteration, the reconstruction is refined by updating the bundle block adjustment. Alternatively, global reconstructions are also presented. In this strategy, images are matched pairwise and the essential matrix is computed, from which the relative orientation can be derived. These orientations are then matched throughout the entire image set. Both COLMAP, RealityCapture and 3DF Zephyr implement this approach, which allows them to process a wide variety of image sequences.

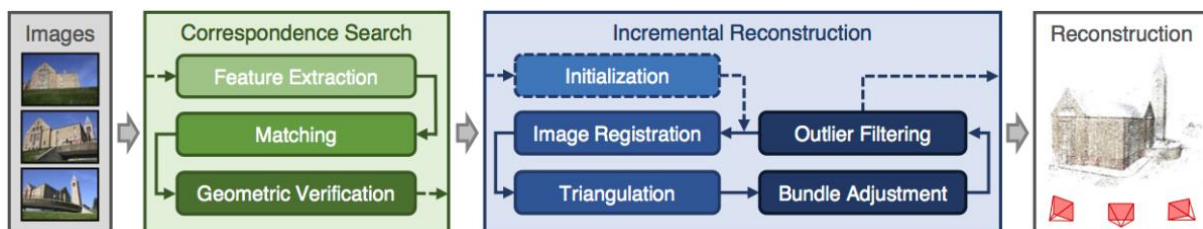


Figure 31: COLMAP's incremental Structure-from-Motion pipeline (COLMAP, 2016)

Table 1: Overview functions of the evaluated Photogrammetric software. Functions of commercial software are often unknown.

	Type pipeline	Feature Extraction	Feature Matching	Image Exterior Orientation	Triangulation	Bundle Adjustment	Robust Estimation
MetaShape	Incremental	-	-	-	-	-	-
3D Zephyr	Incremental/Global	DoG Based	Sequential Circular Unordered Approximate Grid	P3P DLT	Linear LS	-	MSAC
RealityCaptre	-	-	-	-	-	-	-
COLMAP	Incremental	SIFT	Exhaustive Sequential Vocabulary Tree Spatial Transitive	P3P	Sampling –based DLT	Ceres Solver	RANSAC PROSAC LO-RANSAC
Meshroom	Incremental	SIFT AKAZE CCTAG	ANN Cascade Hashing Brute Force Vocabulary Tree	P3P	DLT	Ceres Solver	RANSAC ACRANSAC LO-RANSAC

Feature extraction

The extraction and matching of proper features is crucial in any SfM pipeline. Most software packages currently employ Scale-Invariant Feature Transformation (SIFT) (Lowe, 2004) for their descriptor estimation. This descriptor has certainly proven its worth and is considered very robust and efficient in matching imagery even with limited overlap. For instance, COLMAP implements SIFT as does Alicevision Meshroom. Several alternatives are discussed in the literature in paragraph 3.2.1 such as SURF, SHOT, DAISY and so on. 3DF Zephyr Implements their own descriptor based on Difference of Gaussians (DoG). Meshroom also provides libraries for CCTag and AKAZE descriptors. As expected with commercial software, there is limited information on the exact inner workings of the algorithms involved. Overall, numerous descriptors are capably of robustly encoding a set of input imagery. Therefore, in the V4Design project, we will use one of the existing descriptors and implement it into our own reconstruction pipeline.

Feature Matching

Closely aligned to the feature extraction is the matching of the feature vectors. This involves retrieving the correct correspondence for as many features as possible in an efficient manner. Not only does this include matching the proper descriptors but also selecting suitable image pairs to reduce the number of computations. However, the optimal selection method differs for varying image sequences. For instance, sequential methods implemented

by 3DF Zephyr and RealityCapture specifically target subsequent images such as the frames extracted from videos. This is very efficient since the number of matches to be evaluated is equal to the dimensionality of $O(n)$ compared to brute force methods that only have a computational efficiency of $O(n^2)$. Aside from the obvious choice for video sequences, the V4Design project requires a more versatile image selection. This is especially true for the results from the crawling unit developed by CETH. These images are typically randomly organized, have little overlap and suffer from poor image quality in terms of photogrammetric compatibility as they were often taken with a different purpose at a different time period. Numerous solutions have been presented in the literature to tackle these random datasets. Table 1 shows several implementations of popular methods. For instance, Meshroom and COLMAP also offer vocabulary Trees for efficient pair estimation. Once a set of images is determined, the sets of feature vectors are compared to determine a match. Similar to the image selection, the combination possibilities should be limited as much as possible. Typically, a variant of the RANSAC algorithm is used in order to deal with and eliminate erroneous matches and outliers.

Alignment

The alignment for images in an incremental SfM and a global SfM is different. The former starts with a two-pair reconstruction and iteratively extends the alignment. The latter considers the entire image batch for the bundle block adjustment and computes the exterior orientation of all images at once. Separate algorithms are developed for both approaches. Typically, incremental solutions employ P3P for aligning new images which is the case for Meshroom, COLMAP and 3DF Zephyr. Alternatively, 3DF Zephyr also implements the DLT algorithm for their global SfM pipeline. Both approaches typically perform an alignment optimization. Global SfM only optimizes once with the entire batch while incremental SfM iteratively computes smaller bundle adjustments. Both optimizations have the same mathematical representation and thus algorithms are shared across pipelines. A popular solver based on Levenberg-Marquardt is the Ceres Solver. A prominent factor in incremental SfM is the size of the bundle adjustment that is considered during each iteration. Early versions optimized for the entire aligned batch which leads to computational complexity of $O(n!)$ which is suboptimal. Recent methods only consider a subset, significantly increasing the methods efficiency rendering it a viable option compared to global methods. However, sparse subsets lead to drift which reduces global accuracy. Therefore, incremental methods will often perform an intermediate global bundle adjustment if the cluster has sufficiently expanded as is the case in COLMAP. In V4Design, regular bundle adjustments are mandatory since overlap between the input imagery is low and the alignments often uncertain.

Dense Reconstruction

Once the images are aligned, photogrammetric pipelines produce depth maps and fuse them to compute dense point clouds. This is a memory intensive procedure that accounts for a significant portion of the total computation time. Global methods implemented in RealityCapture and 3DF Zephyr are faster overall but can struggle with larger datasets since the dense point cloud is simultaneously computed for all the depth maps. Therefore, the incremental methods embedded in MetaShape, COLMAP and Meshroom are preferred. Depth maps are computed and fused during each iteration of the process after the bundle adjustment. While the methods for computing depth maps and fusion are similar for the

evaluated software, their parameters and accessibility are not. The efficiency of dense point cloud generation is significantly influenced by the number of input images, the choice of filtering algorithms and whether or not the point cloud is kept in memory during subsequent iterations. It is important in the V4Design pipeline that these parameters can be accessed and dynamically set to suitable values to retrieve proper results from whatever images are crawled from the databases.

Meshing

After the computation of the dense point cloud, the resulting points are used to compute a mesh. Several triangulation and meshing methods are available in literature and the evaluated software. For instance, Meshroom and COLMAP respectively use the DLT and sampling-based DLT methods. Alternatively, 3DF Zephyr employs Linear LS which shows promising results. Similar to the dense reconstruction, the performance of these methods is significantly influenced by the size of the inputs. Also, subsampling drastically improves efficiency but reduces the accuracy. A vital aspect in the implementation of these methods is the availability of parallel processing and GPU integration. All commercial software are heavily optimized, resulting in fast reconstruction algorithms. The Open source software packages do this to a lesser extent but also target the efficiency of the methods themselves. Overall, it is stated that the V4Design pipeline requires both efficient methods and proper CPU and GPU integration which are suited to run on cloud services in order to retrieve the most optimal triangulation.

Texturing

A major advantage of photogrammetric methods is the possibility to compute high quality textures for the 3D models. This is possible because the 3D geometry is actually derived from the images, so the link between the geometry and the texture is readily available. The reprojection of the imagery onto the mesh to retrieve high quality textures is vital to the V4Design outputs. Aside from the high quality textures that are extracted from recent imagery, it should also be possible to replace the aesthetics with user defined textures. Furthermore, it is within the scope of “Reliving the past” to map textures from the past (PUC4). In order to do this, imagery from different time periods should be matched together. Given a set of properly aligned images, the reprojection of the pixel information to UV maps can be computed using the homography between the mesh and the initial image. This is a fairly standard procedure present in all software that is mainly driven by parameters such as the target texel size and the choice of images that are best suited for texturing.

Overall, it is stated that the State-of-the-Art can be best extended by combining the best performing methods from the different software solutions. Each software package has its pros and cons in terms of functionality, efficiency and performance. However, it is challenging to theoretically determine which components are best suited to deal with the wide variety of imagery in the V4Design project. Therefore, practical tests are required to empirically determine the best suited methods.

5.3 Experiments

In this paragraph, the reconstruction pipeline of each software package is tested on its efficiency and quality. Also, the software’s ability to match images from different time

periods is taken into account. The intermediate results of each step of the reconstruction process are observed including the sparse reconstruction, the dense reconstruction and the final model creation along with the high quality textures. The evaluation itself is performed in two steps. First, an overall evaluation is performed of the intermediate outputs with respect to the processing time. Secondly, each step is investigated with more detail in terms of quality and accuracy. Several realistic test cases are presented in order to determine the best suited methods for the V4Design project.

5.3.1 Datasets

The datasets are selected in such a way that they represent the variety of input data in the V4Design project. Also, the image sequences pose significant challenges to photogrammetric process. More specifically, both imagery extracted from videos and unordered images are processed to evaluate the performance of the different algorithms. Four test cases are presented that vary in scale, detailing, image quality and capturing method. In the following paragraph, each data set is discussed in detail along with the difficulties it imposes for the 3D reconstruction pipeline.

Dataset 1: Schiller Monument

The first data set is a video sequence of the Schiller Monument on the Gendarmenmarkt, Berlin (Figure 32). A 2 min video fragment was captured with an unknown device at 25fps on the 23th of May 2018 by a tourist visiting the square. The video shows the statue from different points of view as the tourist slowly walks around the statue. 464 frames were extracted from the video. Overall, this is a fairly good test as there inherently is significant overlap between the imagery. It is therefore expected that the alignment by the different software will yield proper results. However, the video is taken at pedestrian height and other tourists are walking in front of the target object causing occlusions. Also, the textures are suboptimal as the lighting conditions change as the tourist walk around the statue, there is little texture on the statue itself and there is a lot of shading. Furthermore, the degree of lens distortion will certainly affect the reconstruction accuracy.



Figure 32: Schiller Monument, Gendarmenmarkt, Berlin

Dataset 2: Technology campus Ghent

The second data set is a video fragment of the Technology campus Ghent, Belgium (Figure 33). It is a 8min drone flight at 59fps of the campus grounds captured with a DJI Phantom 4Pro operated by a pilot in training on the 24th of December 2017. 475 frames were extracted from the video. Similar to the first test case, the overlap is inherently high. However, the flightpath is more arbitrary due to the pilot's training exercise. Also, the flight was conducted when the campus was covered in snow, causing reflections and poor textures. Overall, it is expected that the reconstruction process will yield proper results of the grounds and the surrounding buildings.



Figure 33: Dataset 2, Technology campus Ghent

Dataset 3: Gendarmenmarkt

The third dataset is a data dump from the crawling unit developed by CERTH. It contains 993 images of the Gendarmenmarkt, Berlin. It is a representative case for the type of imagery the V4Design project is looking to use. However, there are significant challenges for the 3D reconstruction pipeline. First of all, the imagery originates from different sources, time periods and were captured both during the day and night (Figure 34). Numerous images are irrelevant, showing only fragments of the scene or simply pictures that were semantically linked to the Gendarmenmarkt. Most images are also artistic including special lighting effects and selfies. On top of that, the quality of the usable imagery is poor and heavily distorted. It is expected that the 3D reconstruction based on this imagery will be challenging if not impossible. However, it is vital to evaluate which software components are capable of dealing with this sort of data.



Figure 34: Dataset 3, image results of the crawling of the Gendarmenmarkt, Berlin. The bottom left image originates from 1955, the bottom centre image from 1988, and the bottom right image from 2005.

Dataset 4: Brandenburger Gate

The fourth dataset is another data dump from the crawling unit. It contains 991 arbitrary images of the Brandenburger Gate, Berlin (Figure 35). Similar to the Gendarmenmarkt, this monument has been thoroughly documented throughout history, resulting in a wide variety of images. The same challenges occur as most images are unsuited for photogrammetric purposes. However, there is a distinct difference between the image set of the Gendarmenmarkt and the Brandenburger Gate: at the market, the scene of interest lies at the edges of the market, and thus, most images are captured pointing outwards. This causes issues in the 3D reconstruction pipeline since a minimal baseline is required for multi-view reconstruction. With the Brandenburger Gate, the scene of interest lies at the center, and thus most images are captured pointing towards the center of the scene. This is a preferable photogrammetric setup and thus it is expected that despite the shortcomings of the imagery, the Brandenburger Gate will yield adequate results, provided that the images from different time periods can be matched.



Figure 35: Dataset 4, image results of the crawling of the Brandenburgate, Berlin. The bottom left image depicts an image at MiniEuropa, the bottom centre is an artistic image captured in the evening and the bottom right image dates from 1955.

5.3.2 Overall test results

Table 4: Intermediate 3D Reconstruction results of the four test cases with their respective processing time

	Aligned images	Reprojection Error [pix]	Sparse point cloud	Computation time	Dense Point Cloud	Computation Time	Meshing [triangles]	Computation Time	Texturing [%]	Computation Time	Total Computation time
Schiller Monument	464										
MetaShape	463	0.92	158 032	0:19:21	1 170 169	1:00:31	358 617	0:09:30	100%	0:00:48	1:30:10
3DF Zephyr	464	0.42	54 914	1:07:00	3 912 433	0:46:00	678 891	0:15:00	100%	0:02:30	2:10:30
RealityCapture	464	0.88	492 576	0:04:42	300 442			0:08:31	100%	0:02:02	0:15:15
COLMAP	464	0.94	462 547	0:16:55	10 54 503	5:21:00	-	-	-	-	5:37:55
Meshroom	462	0.65	36 760	0:04:00	2 767 089	2:22:00	541 649	0:08:00	100%	0:01:00	2:35:00
Technology Campus	475										
MetaShape	352	1.07	37 191	0:31:47	1 846 373	1:00:36	804 639	0:12:35	73%	0:01:30	1:46:28
3DF Zephyr	463	0.31	82 159	0:42:00	1 181 614	0:14:17	405 876	0:09:07	86%	0:03:58	1:09:22
RealityCapture	474	0.66	351 585	0:06:48	1 642 194			0:15:29	75%	0:00:58	0:23:15
COLMAP	475	0.61	112 502	0:09:25	2 870 517	1:53:01	-	-	-	-	2:02:26
Meshroom	464	0.51	56 406	0:21:00	3 265 817	0:49:40	2 024 618	0:08:40	79%	0:03:00	1:22:20

Table 5: Intermediate 3D Reconstruction results of the four test cases with their respective processing time.

	Aligned images	Reprojection Error [pix]	Sparse point cloud	Computation time	Dense Point Cloud	Computation Time	Meshing [triangles]	Computation Time	Texturing [%]	Computation Time	Total Computation time
Gendarmenmarkt	993										
MetaShape	278	1344.42	23 952	4:48:50	-	-	-	-	-	-	4:48:50
3DF Zephyr	283	1.22	16 816	0:33:11	948 963	0:21:55	1 351 715	0:28:06	81%	0:21:50	1:45:02
RealityCapture	290	1.60	42 060	0:16:58	1 050 248			0:21:49	93%	0:13:14	0:52:01
COLMAP	723	0.74	105 893	0:56:22	6 099 431	7:53:00	3 011 079	0:17:05	75%	0:25:59	9:32:26
Meshroom	-	-	-	-	-	-	-	-	-	-	-
Brandenburgerpoort	991										
MetaShape	42	1.38	939	0:20:45	-	-	-	-	-	-	0:20:45
3DF Zephyr	209	0.27	5 858	0:27:33	960 945	0:03:59	104 774	0:02:07	25%	0:00:21	0:33:39
RealityCapture	131	1.08	19 530	0:01:51	439 060			0:08:49	32%	0:00:52	0:11:32
COLMAP	461	0.14	46 310	1:18:33	1 640 873	4:06:08	-	-	-	-	5:24:41
Meshroom	5	0.05	583	12.03.07	-	-	-	-	-	-	-

As an initial test, the entire 3D reconstruction process is conducted for every test case in every software package. The 5 packages processed the data and at every step, intermediate results were saved. As it is impossible to have a uniform set of parameters for the reconstruction process, each package processed the test cases several times until an optimal set of settings for that software was established. The parameters were chosen in such a way that for each test case the number of matched images, the dense point cloud, the mesh quality and texturing quality were maximized while the processing time was minimized.

Table 4 depicts the results of the processing of the video sequences of the Gendarmenmarkt and the Campus in Ghent. Basic results are reported for each intermediate step such as the number of reconstruction points and the reprojection error along with the computation time. From these values, a superficial evaluation can be performed of each software's efficiency which will then be later discussed in detail in paragraph 5.3.3 and 5.3.4. Overall, all software packages were capable of computing a proper reconstruction from the extracted keyframes. This is expected since the GRIC detector ensures proper baselines between consecutive images while retaining sufficient overlap. This is reflected in the average reprojection errors of the tiepoints between the images, which is subpixel for both videos. However, not all software was capable of matching all images. For instance, MetaShape struggled with the vast number of pixels containing snow in Ghent, which are challenging to match. The processing time also differs between the different projects. RealityCapture outperforms the other software on both occasions in the sparse and dense matching. MetaShape, 3D Zephyr, COLMAP and Meshroom all take second place depending on the procedure. MetaShape has adequate sparse and dense matching but is not capable of matching a portion of the imagery. COLMAP scores average on the matching but gets outperformed in the dense reconstruction. Meshroom shows promising results in the dense matching but uses significantly less tiepoints for the sparse reconstruction, possibly introducing errors in the matching. 3D Zephyr also uses less tiepoints but has significantly higher processing times for the sparse reconstruction.

Table 5 shows the results for the imagery batches crawled by CERTH of the Gendarmenmarkt and the Brandenburger Gate. Similar to Table 4, a superficial comparison can be made between the software in terms of processing speed with respect to the outputs. In contrast to the video processing, it is expected that severe problems will occur with the sparse and dense reconstruction. These test cases are very relevant since it can be observed how robust the different algorithms are to noise, artefacts, lighting variation, minimal overlap, defocus and other effects. Furthermore, the majority of the V4Design imagery will contain similar challenges. Table 5 shows that severe issues indeed surface for both test cases. MetaShape and Meshroom completely misalign the Gendarmenmarkt and only succeed in matching a small portion of the Brandenburger Gate. 3D Zephyr and RealityCapture do match about a third of the images but show increased reprojection errors. COLMAP is the undisputed victor in the sparse reconstruction, matching over double the number of images than other software while retaining proper reprojection errors. The algorithm's ability to match these types of inputs in combination with its Open Source functions are interesting features to incorporate into the V4Design pipeline. However, COLMAP's dense reconstruction is relatively slow in comparison to other software. Referring back to Table 5, it is observed that, despite the increased number of points being reconstructed, COLMAP's dense reconstruction is outperformed by all other software. Since

the V4Design project aims to advance the State-of-the-Art through the use of Open Source algorithms, it is a promising proposal to combine COLMAP's matching with other Open Source algorithms such as Meshroom. Table 5 shows that, given a set of aligned imagery, Meshroom's dense matching and subsequent meshing shows promising results. RealityCapture does still outperform in both steps due to its innovative approach to combine the dense reconstruction and meshing in a single step, thus passing over the dense point cloud. However, we argue that within the scope of the V4Design project, we want to retain access to the dense points cloud as UR_021 (D7.1,7.2) indicates architects want to edit and clean the dense point cloud. Together with the Open Source nature of Meshroom, it is proposed to adopt and extend algorithms from Meshroom for the dense reconstruction and meshing.

5.3.3 Comparison of the sparse reconstruction

While the previous tests give a general overview of the efficiency of the different software packages, a more detailed study of the reconstruction performance is needed to constitute a proper reconstruction pipeline. In this section, the results of sparse reconstruction step are briefly discussed in terms of quality, noise and density.

Table 6: Overview sparse reconstruction

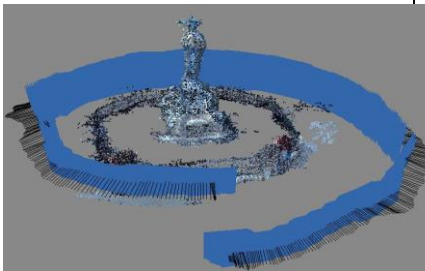
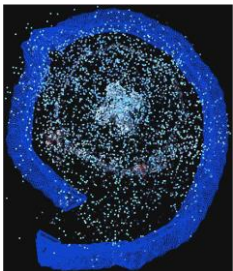

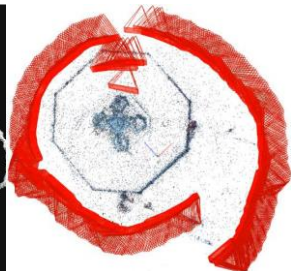
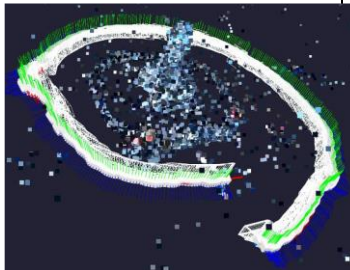
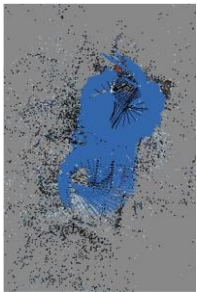
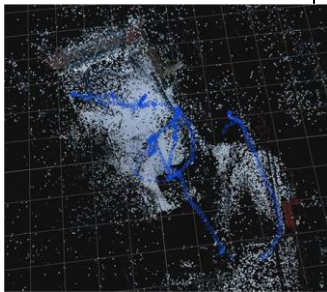



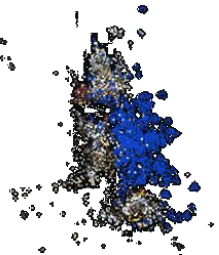

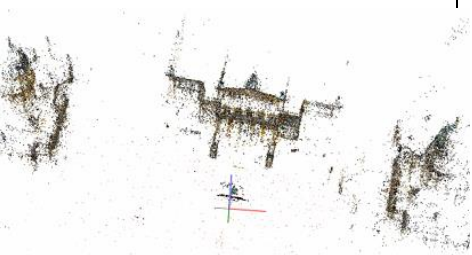

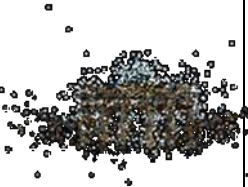



	MetaShape	3DF Zephyr	RealityCapture	COLMAP	Meshroom
Schiller Monument	 <p>Good quality</p>	 <p>Medium quality</p>	 <p>Good quality</p>	 <p>Good quality</p>	 <p>Medium quality</p>
Technology Campus	 <p>Medium quality</p>	 <p>Medium quality</p>	 <p>Good quality</p>	 <p>Good quality</p>	 <p>Good quality</p>

Table 7: Overview sparse reconstruction

	MetaShape	3DF Zephyr	RealityCapture	COLMAP	Meshroom
Gendarmenmarkt	Poor quality	 Poor quality	 Poor quality	 Good quality	failed
Brandenburger Gate	 Poor quality	 Medium quality	 Medium quality	 Good quality	 Poor quality

This paragraph contains the detailed study of the sparse reconstruction of the four test cases by the different software packages. A visual inspection is conducted of the camera alignment, the outliers of the tiepoint reconstruction and the overall quality and density of the models. Table 6 and Table 7 depict the reconstruction of the tiepoints in 3D. Similar to the results presented in the paragraph above, the videos show good results for all packages. For instance, the camera path can be clearly observed with the user circumventing the statue in the middle of the Gendarmenmarkt. COLMAP even registers images taken by the user stepping closer to the statue along the trajectory. However, there are differences in the quality of the reconstructed tie points. For instance, RealityCapture reconstructs tie points on the main body of the statue but not on the pedestal while this is the case with other algorithms. In contrast, there are limited outliers, which is not the case for other software. MetaShape and COLMAP still provide good quality results but Meshroom and 3D Zephyr clearly generate more noise. These outliers will have significant impact on the dense reconstruction and subsequent meshing since they are the basis for the depth map generation. Similar results are observed for the University campus in Ghent. RealityCapture and COLMAP provide low levels of noise while the other software packages do contain some outliers. MetaShape again struggles with the large amounts of snow in the imagery and produces more outliers, which is also reflected in the increased reprojection error (see Table 6, Table 7).

A significant difference is observed between the sparse reconstruction of the Gendarmenmarkt and the Brandenburger Gate. Not only do most packages fail to match the majority of the imagery, there are also alarming levels of noise present in the datasets. For instance, both 3D zephyr and Meshroom match less than a third of the imagery and produce so much noise that the structure is unrecognizable. RealityCapture provides better results but the asset is still covered in high levels of outliers that will obstruct the dense matching. The performance of COLMAP is again confirmed as it does not only match more images, but also produces acceptable levels of noise in challenging conditions.

5.3.4 Comparison of the dense reconstruction and meshing

Similarly to the previous section, a brief overview of the results of the dense reconstruction and subsequent meshing is presented with regard to the quality, noise and density. The main focus is on the dense reconstruction since meshing is predominantly driven by the input point clouds.

Table 8: Overview dense reconstruction








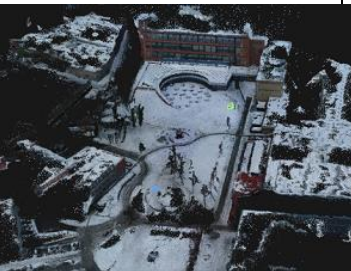





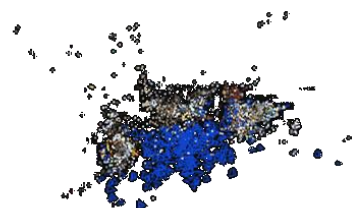


	MetaShape	3DF Zephyr	RealityCapture	COLMAP	Meshroom
Schiller Monument	 <p>Good quality</p>	 <p>Medium quality</p>	 <p>Good quality</p>	 <p>Poor quality</p>	 <p>Good quality</p>
Technology Campus	 <p>Good quality</p>	 <p>Good quality</p>	 <p>Good quality</p>	 <p>Medium quality</p>	 <p>Good quality</p>

Table 9: Overview dense reconstruction

	MetaShape	3DF Zephyr	RealityCapture	COLMAP	Meshroom
Gendarmenmarkt	Failed	 <p>Poor quality</p>	 <p>Poor quality</p>	 <p>Good quality</p>	failed
Brandenburgerpoort	Failed	 <p>Poor quality</p>	 <p>Medium quality</p>	 <p>Good quality</p>	Failed

This paragraph contains the detailed study of the dense reconstruction of the four test cases by the different software packages. A visual inspection is conducted of the point cloud/mesh quality with respect to the sparse reconstruction. Table 8 and Table 9 depict the dense matching and/or subsequent meshing in 3D. As discussed in the superficial evaluation in paragraph 5.3.2, the results from the dense matching differ from the sparse reconstruction presented above. For instance, it is expected that RealityCapture outperforms the other software due to its innovative method that combines dense matching and meshing. Table 8 shows that RealityCapture is indeed not only fast, but also provides excellent results in terms of the dense reconstruction. Especially for first two test cases, it is observed that, given proper sparse point clouds, the reconstruction is very clean. Similar results are observed for Meshroom and Metashape (for the statue) that produce clean 3D reconstruction even though they are less performant. 3D Zephyr and COLMAP produce significantly more noise despite their proper sparse reconstruction.

For the last two test cases, the quality of the models is inadequate. As expected, the sparse point cloud drastically influences the dense reconstruction. The depth maps are prone to noise and so is the meshing. It is clear that no software can deal with overwhelming noise in the sparse point clouds. RealityCapture is even more affected by this since it does not produce a dense cloud which can be manually processed by the user. The result is that all models are heavily distorted except for COLMAP that had a better sparse reconstruction. However, Table 9 shows that COLMAP is significantly slower than the other packages given proper inputs. Furthermore, the dense matching of COLMAP is often prone to failure. It is therefore clear that the State-of-the-Art can be extended by merging COLMAP's sparse reconstruction with Meshroom's dense reconstruction and build our own functionalities on top of the Open Source code to constitute both a robust and performant reconstruction pipeline.

5.4 Architecture design 3D reconstruction Pipeline

A detailed description of the architectural design of the reconstruction is provided in D6.4 paragraph 3.1. Figure 36 shows the different executed steps throughout the pipeline according to this design.

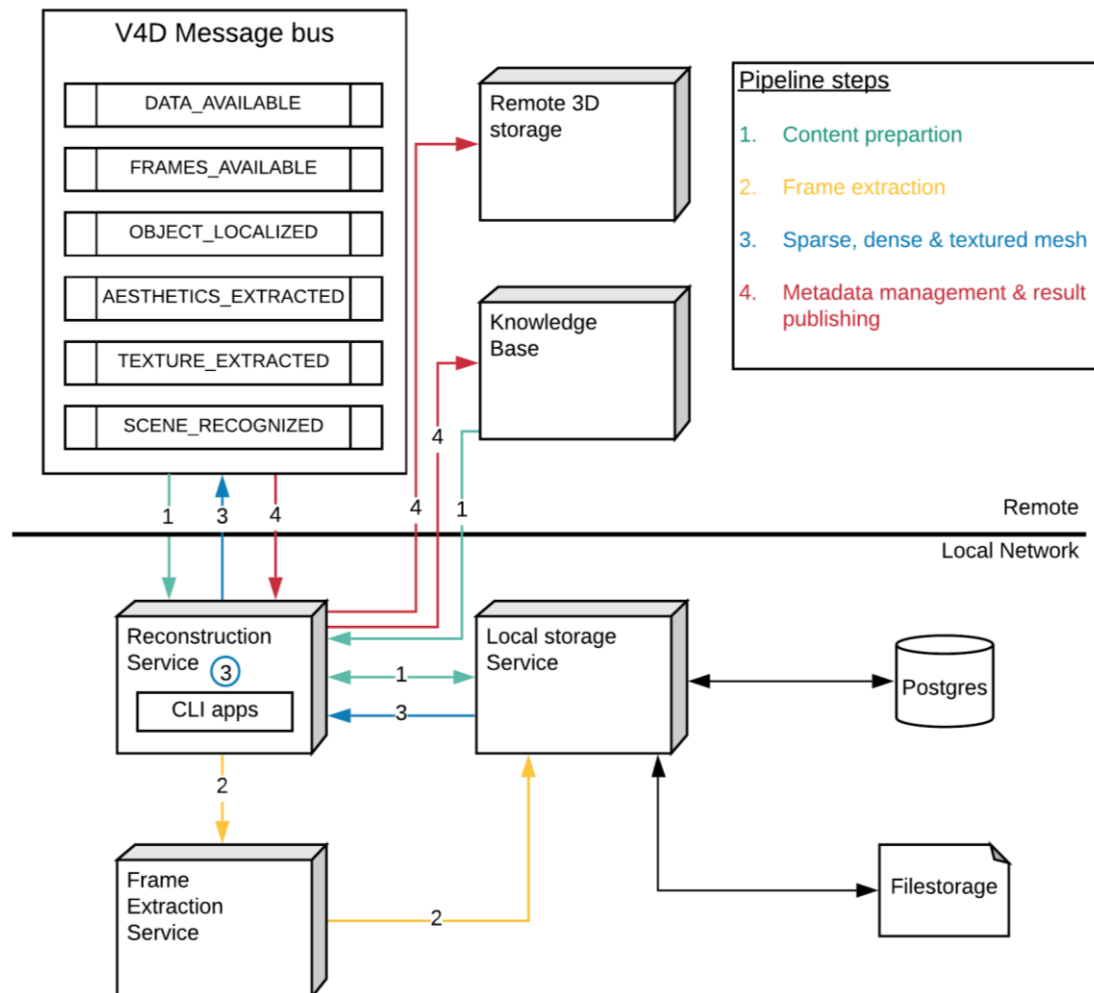


Figure 36: Overview of steps performed in the current iteration of the reconstruction service.

Each of these steps are described below:

1. Content preparation

The service is informed of new data to process. Video data is downloaded from the central knowledge base and the processing environment is prepared on the network storage system. New data is managed in an sql-based database.

2. Frame extraction

A command is issued to the frame extraction service. The storage service provides the appropriate video data and handles the storage of the individual frames. Once all keyframes

are extracted a message is sent to “FRAMES_AVAILABLE” topic on the V4D message bus. This will cause the STBOL and texture proposal service to process the frames.

3. Sparse, dense and textured mesh

The reconstruction service requests the appropriate keyframes from the local network storage for a specific video. Sparse, Dense and texturing algorithms are executed and all results are once again managed by the storage system.

A processing environment is prepared based on the extracted keyframes. Here Sparse, Dense and texturing algorithms are executed resulting in image alignment, pointcloud and mesh output respectively. All output data is managed by the local storage service.

4. Metadata management and result publishing

Once all necessary metadata has been received from other modules (Aesthetics, STBOL, TP) the resulting 3D reconstruction, including metadata, is published to the knowledge base and thus made available for subsequent processing or the user tool environments. Optionally additional texturing routines may be run based on TP results in order to restyle the mesh (Example shown in Figure 41).

5.5 Implementation of SfM algorithms

In the previous chapter the system was explained from an architectural standpoint (use of microservices and communication methods). This chapter clarifies the internal SfM algorithms across all these micro services. Figure 37 provides a clear view of all the necessary steps from images towards a textured mesh. For this we use a combination of Colmap- and AliceVision-based code. Optional steps are displayed in red.

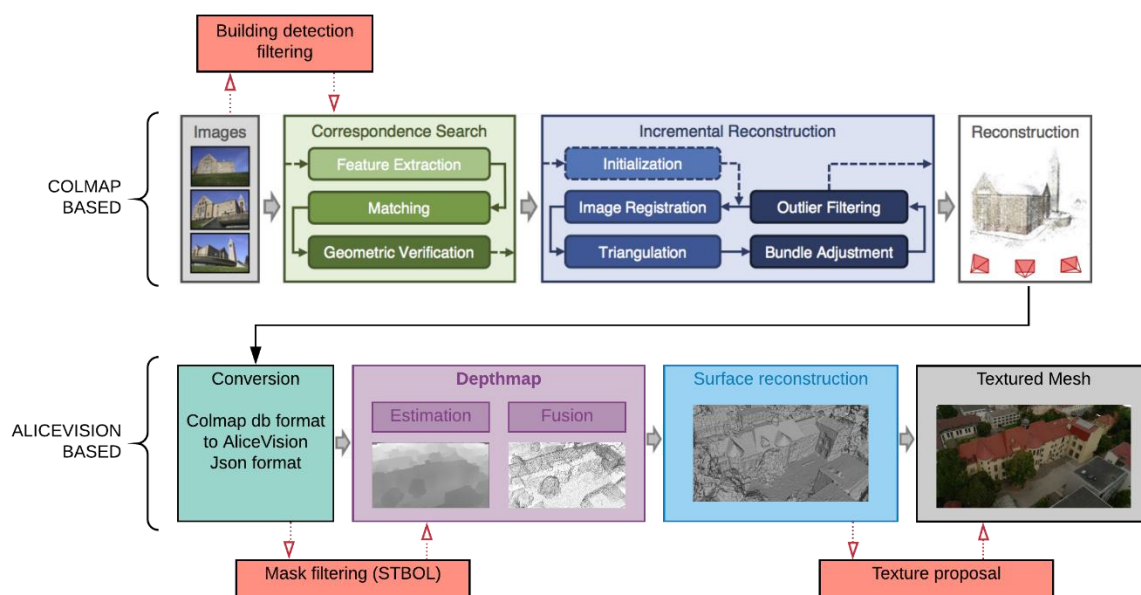


Figure 37: Steps performed in the current pipeline (After initial keyframe extraction).
Optional (red) algorithms with data from other partners.

5.5.1 Keyframe extraction

Paragraph 4 already discussed the developed algorithm in detail. The service has been implemented in C++ with help of the following libraries:

FFmpeg ¹	Video decoding
OpenCV ²	Linear algebra computation, Kanade-Lucas-Tomasi (KLT) feature tracking

5.5.2 Sparse Reconstruction

The calculation of camera alignment and sparse point cloud is based on our custom COLMAP implementation. This functionality is integrated inside the “Reconstruction Service” node as seen in Figure 36. C# was used for managing and initiating of the several subtasks as well as IO. C++ was used for high performance structure-from-motion code, this code is built as a set of command-line interface (CLI) programs. Following libraries were used (Excluding any non-SfM related libraries):

Eigen3 ³	Linear algebra operation
COLMAP	Incremental based sparse reconstruction

5.5.3 Dense reconstruction

The implementation of the dense reconstruction algorithms are performed very similarly to those of the sparse reconstruction. C# was also used in combination with C++ compiled CLI applications. These CLI applications are built using the Open Source AliceVision project, which has received funding from the European Union’s Horizon 2020 research and innovation programme in the past⁴. Here, the Depth-map generation, surface reconstruction and texturing code has been reused. Additional filtering functionalities based on masks (STBOL) may be used as seen in Figure 37. Whereas the texturing would normally be executed on the original undistorted frames, this pipeline also accepts input from the texture proposal service developed in T4.2.

5.5.4 Service output

Once the pipeline has successfully finished, a result is pushed to both the V4D message bus and the knowledge base. The result is Json-formatted which contains the following data (for the 1st prototype): Unique ID, visual analysis results, mesh information, point cloud information, data storage URLs, used raw data (ID’s from video & images) and a thumbnail.

¹ <https://ffmpeg.org/>

² <https://opencv.org/>

³ <http://eigen.tuxfamily.org>

⁴ POPART (project ID 644874) and LADIO (project ID 731970)


```
{
  "Id": "AH7X940UUE",
  "visualAnalysisMaskCollisions": [ "trees", "building/inn" ],
  "meshes": [
    {
      "Id": "KLSH4D",
      "format": 0,
      "faceCount": 147416,
      "textureCount": 4,
      "reconstructionId": "AH7X940UUE",
      "modelurl": "https://160.40.49.184/reconstructions/KLSH4D/texturedMesh.fbx"
    }
  ],
  "pointclouds": [
    {
      "Id": "",
      "reconstructionId": "AH7X940UUE",
      "pointCount": "124863",
      "file": "",
      "potreeLink": ""
    }
  ],
  "Simmos": [ // id of used content (1 video in this case)
    "11988275-7604-4121-b3bf-6e5d848f6261"
  ],
  "modelurl": "https://160.40.49.184/reconstructions/KLSH4D/texturedMesh.fbx",
  "Idmodel": "KLSH4D",
  "thumbnailurl": "https://160.40.49.184/reconstructions/thumbnail.jpg"
}
```

Figure 38: Example Json generated after the pipeline finishes. Pointcloud data is currently not used by the tools and thus not published to the storage service.

5.5.5 Pipeline experiments

A total of 47 videos with a SIMMO ID are currently available in the V4D knowledge base. These videos are well annotated. The 3DR pipeline was initiated using a DATA_AVAILABLE message. Currently 24 sparse point clouds have been automatically extracted from these videos and a total of 14 textured models are automatically created. An initial, currently semi-automatic, pipeline test for texture proposal was executed successfully and can be seen in Figure 41.

A large set of AF video data yielded good results in spite of the poor video quality (640x480 pixel, grainy due to low light conditions) and can be seen in Figure 39. However in a minority of the cases the meshing process, though executing successfully, was unable to fully reconstruct the objects (seen in Figure 39, 3rd and last mesh).

For some videos, containing on-screen text or edited in another fashion, the automatic pipeline did not succeed. Figure 31 shows an example of a video found in the V4D knowledge base with on-screen text. This in particular causes bad image alignment due to a high amount of matched features in the textual areas. Tests showed that by removing these keyframes good reconstructions can be achieved.

For the first prototype only consortium partner data with sufficient textual annotations has been imported into the knowledge base. Additional models were also retrieved from data not yet present in the V4D knowledge base. This includes models extracted from: crawling unit data, YouTube and non-annotated consortium partner data (Figure 40).



Figure 39: Some results from AF data present in KB. Bottom right: failed bench meshing

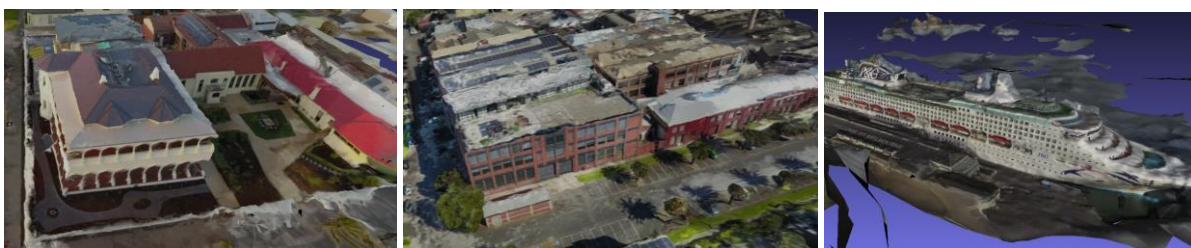


Figure 40: Reconstruction from AF data not yet present in the KB.

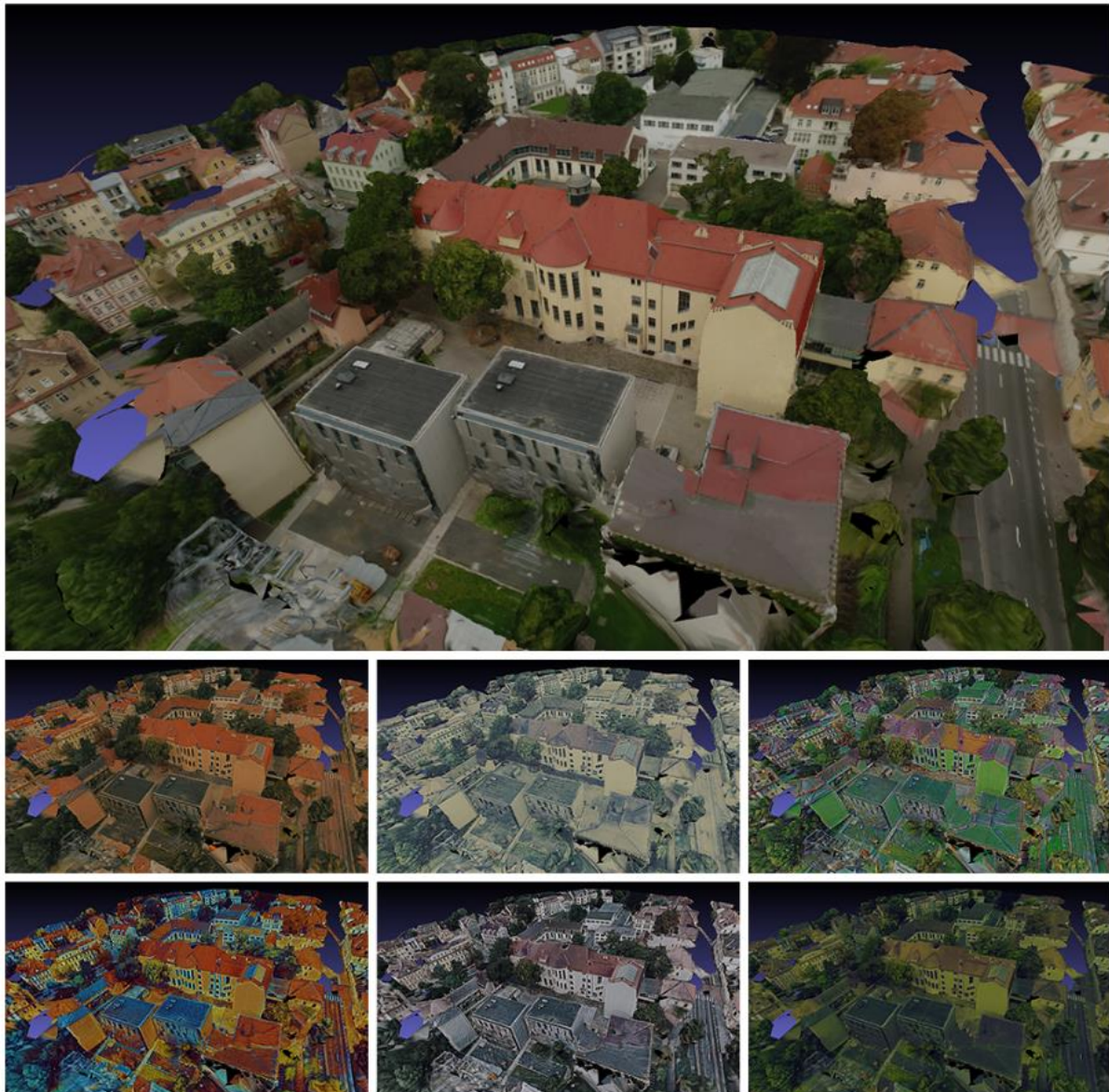


Figure 41: Reconstruction of DW's data. 6 additional texture styles were computed based on texture proposal results.

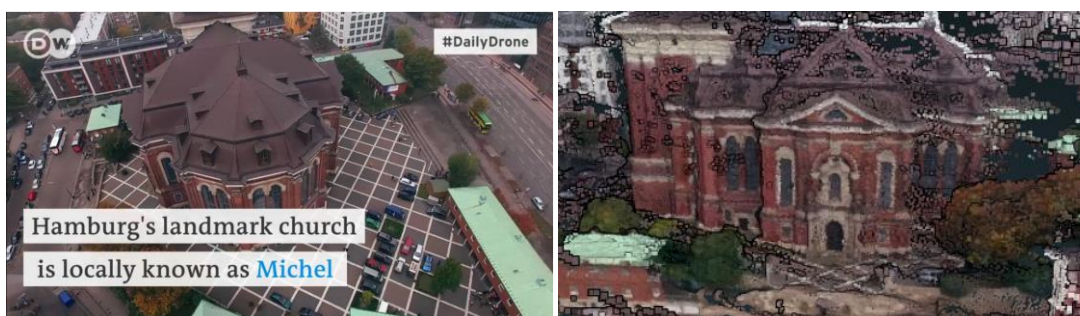


Figure 42: Left: Example 'bad' keyframe containing onscreen text and logos. Right: resulting 3D, obtained by removing the bad keyframes.

5.5.6 Conclusions

The modular architecture design allows for easy further development and addition of new algorithms in the upcoming 2nd prototype, such as improved mesh filtering, model segmentation and IFC object extraction.

In this 1st prototype emphasis was put on the system architectural design. The modularity and backwards-compatible gRPC protocol eases further implementation of new algorithms currently being developed for the 2nd prototype (mask filtering, segmentation, IFC extraction or single view reconstruction). After a thorough study of current state-of-the-art commercial and non-commercial photogrammetry software, our SfM pipeline has combined the strengths of 2 major open source packages: Colmap and AliceVision. Research and development done in V4Design's photogrammetry service builds on top of this software and thus contributes to these packages. A complete and automatic SfM pipeline has been implemented and tested on a variety of data. Tests for restyling a mesh based on texture proposal results were successful and this feature can be automatized and added to the current automatic pipeline.

6 ENHANCED 3D MODEL EXTRACTION

As stated in D3.2, Seven user requirements from D7.2 have been associated with the texture proposals module of V4Design, namely the UR_002, UR_003, UR_004, UR_011A, UR_013, UR_017 and UR_018. These include both high quality textures from reused imagery in the V4Design project and fictional textures that can be used for conceptual design. In this chapter, the implementation of texturing and masking results in the SfM pipeline are discussed along with experimental results.

6.1 Input from STBOL and AE&TP algorithms

The inner workings of the Spatio-Temporal Building and Object Localization and the Aesthetics Extraction algorithm are discussed in D3.2. The former classifies pixel information into object classes. From these classes, image masks are generated that are used to support the SfM pipeline. The latter is an architectural synthetic tool to support model texturing. The outputs of both algorithms are embedded in the reconstruction pipeline to enhance the results. In the following paragraphs, each of the inputs is discussed in detail.

6.1.1 STBOL

The Spatio-Temporal Building and Object Localization algorithm detects various types of objects in the input imagery. It consists of a pre-trained classification model based on the VGG16 model that interprets each pixel of a set of input imagery and defines its class based on a set of local and contextual features (see D4.2). As an output, binary masks are computed for each image that are used to segment the reconstruction results (Figure 43). In this chapter, the emphasis is on the integration of the building masks since the SfM pipeline predominantly focuses on the built assets.

The output masks can serve various purposes. First, they can be used to speed up the reconstruction since only 3D data has to be computed on structures instead of the entire environment. Secondly, they can be used to enhance the sparse and dense reconstruction by filtering outliers in the point clouds. For instance, the removal of faraway points on moving clouds, humans and cars can aid in the production of proper point cloud data. This in turn results in significantly better meshes since less noise is present in the data set. Finally, the prior segmentation in the imagery facilitates the semantic enrichment since a prior classification is performed limiting the number of available classes to those present in building environments.



Figure 43: Masking results of the VGG16 CNN algorithm: (a) initial image, (b) building mask and (c) sky mask.

6.1.2 AE&TP

The aesthetics algorithm also builds upon the pre-trained VGG16 architecture and is fine-tuned by learning the aesthetics parameters from the benchmark Places2 dataset as discussed in D3.2. Both the style, genre and artist characteristics are extracted per painting. These are merged with input imagery of the crawling to generate novel textures from existing data. For instance, Figure 44 depicts the texture style of Van Gogh extracted by the AE&TP algorithm superimposed on a crawled image of the Gendarmenmarkt.



Figure 44: Style transfer of the famous painting Cafe Terrace at Night of Vincent Van Gogh to an image of the Gendarmenmarkt.

As stated in D3.2, the AE&TP results can be combined with the masks of STBOL to transfer the texture proposals to a specific part of an image or video. An innovative application of the proposed algorithm is background to background and foreground to foreground transfer between art and real life inputs. For instance, Figure 45 shows the superimposition of the yellow and blue building texture style of Van Ghogh projected on the wall of the Konzerthaus at the Gendarmenmarkt. This targeted style transfer allows for a wide variety of styles to be combined for the different objects in the scene, offering numerous possibilities for conceptual design and game development.



Figure 45: Targeted texture superimposition of the Van Gogh style (b) on the mask of the Konzerthaus (a) at the Gendarmenmarkt, resulting in an impressionistic digital representation of the asset (c).

A third potential application of the AE&TP algorithm is the enhancement of the texture quality of the results of the Structure-from-Motion pipeline. Typically, photogrammetric texturing is performed with an equal input of all oriented imagery. While this is the ideal setting for videos or image sequences that were captured at the same time, this proves subideal for the crawling outputs. After all, the emphasis of the crawling unit is to retrieve all the different imagery taken from the same asset. This includes pictures taken in various lighting conditions such as during festivals, different time periods and so on (Figure 46). As multiple states are useful to conceptual design, it is interesting to separately texture the final model using only the imagery of a specific style or period. As a result, users can have their assets textured according to their needs without the need to alter lighting conditions or

manually adjust textures, which requires expert knowledge concerning texture mapping in shaded conditions.



Figure 46: Difference between images taken during different time periods, which results in subideal texturing of the SfM reconstruction.

6.2 Segmentation of 3D models

As previously discussed the resulting masks of the STBOL algorithm can be beneficial to the proper reconstruction of building geometry from crawled imagery. In this section, the first implementation of masked imagery is discussed along with the literature study, experimental results and future work.

6.2.1 Masking in SfM

SfM pipelines rely on significant image redundancy in order to create photorealistic models of objects of interest. The State-of-the-Art and the presented experimental results show that scenes can be reliably reconstructed under the assumption of shape constancy and appearance congruency, commonly associated with static structures. However, as shown in Figure 47 the crawled imagery and videos are littered with what is referred to as dynamic objects due to visual motion. 3 types of visual motion are defined in the literature (Nelson and Polana, 1992).

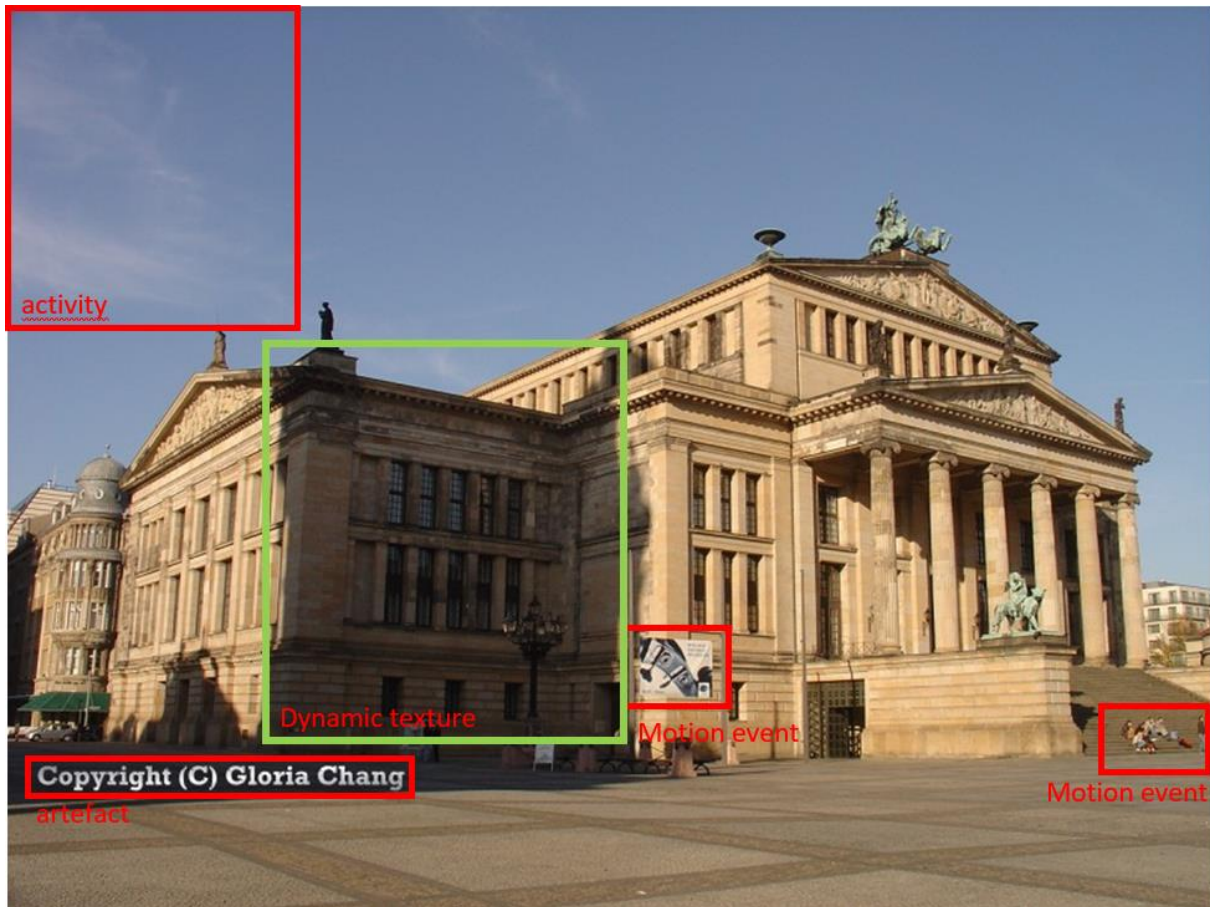


Figure 47: Crawled imagery and videos are littered with dynamic objects.

- **Activities:** These are moving objects such as cars or humans, which have motion patterns that are periodic in time. These mainly cause artefacts in sequential imagery and frames extracted from videos, distorting both the 3D reconstruction and the subsequent texturing. In the crawled imagery, activities are considered motion events since the occurrences typically are observed only in a single image and are not predictable.
- **Motion Events:** non-periodic events like the opening of a door, which lack spatial periodicity. The impact of these events is similar to that of the activities. Video frames will have their 3D reconstruction and texturing distorted. The crawled imagery will show only artefacts in the texture as it is unlikely that the event occurs in multiple crawled images.
- **Dynamic textures:** these occurrences exhibit statistical regularity but have uncertain spatial and temporal extent such as rain, shadow, and smoke. These do not affect the reconstruction but have drastic effects on the texture quality of the model.

It is vital that these occurrences are detected and taken into consideration for the 3D reconstruction and the subsequent texturing. Both aspects are treated separately in the literature. The enhancement of 3D reconstruction by masking is considered an instance of motion detection (Jodoin et al., 2014). The simplest of options is masking non-buildings of the images before starting the SfM pipeline. However, by removing a portion of the scene, the number and distribution of tiepoints is reduced, possibly resulting in a less accurate reconstruction. As discussed in paragraph 4, a reduction in tie-points leads to an increase in

errors. Alternatively, the exterior camera parameter estimation is performed on the full images while the dense point cloud is constructed using the masked images. This is a promising approach which is also proposed in the V4Design pipeline. A potential drawback is the lack of 3D interpretation of motion events, which in this method solely relies on the 2D masks. An alternative approach is to use the masks during the iterations of an incremental reconstruction procedure. One method is to filter the outputs based on a shape-from-silhouette procedure presented by Schmid et al. (Schmid2014). They perform a coarse identification of the dynamic scene elements and use this information to perform a foreground-background segmentation similar to the masks computed by STBOL. An essential part of their work is the iterative process of projecting dynamic occurrences back onto the input imagery to confirm their results. Post-reconstruction filters are also considered such as raytracing the masks to segment the dense reconstruction of the full images (Rashad et al., 2017). However, it is computationally more efficient to incorporate the masks earlier in the workflow.

6.2.2 Methodology

In this section, the exploitation of mask extracted by STBOL is presented. As discussed in Section 4 and D4.2, keyframes are forwarded to STBOL. In parallel, the sparse reconstruction is computed for all inputs. Once the images are aligned, the imagery is replaced with segmented/masked imagery to generate the depth maps. Currently, only building masks are retained as defined by the user requirements. The different step of the masking are discussed in detail in the paragraphs below (Figure 48).

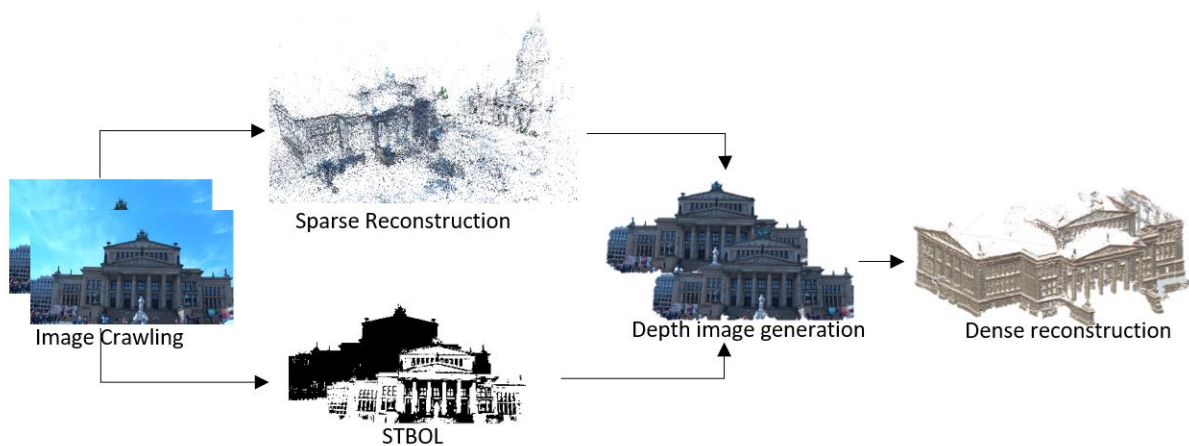


Figure 48: Dense reconstruction procedure, based on presegmented imagery.

```

{
  "fps": "0",
  "quality": "null",
  "resolution": "1280 x 720",
  "frame": "8",
  "simmo": "2866f985-3585-4f66-ad9a-327d7c6e9f6c",
  "mask_url": "http://160.40.51.32:10010/download/test/8852.jpg",
  "timestamp": "2019-05-29 11:38:04.137"
}
  
```



Figure 49 Left: received Json data from STBOL. Right: resulting mask

The automation of the STBOL masks and its integration in the 3D reconstruction pipeline is currently being implemented.

6.2.3 Experiments

The Gendarmenmarkt is used several times as a demonstrator throughout the project. For the prototype of the enhanced model extraction, a video sequence is tested, created by a photographer walking over the square filming the Konzerthaus and the French Cathedral, cluttered with tourists and other artefacts (Figure 50). It is an ideal dataset for 3D reconstruction due to the camera movement and motion detection is paramount since the artefacts are located directly in front of the buildings and thus would cause problems in the reconstruction and subsequent texturing of the site. During the preprocessing, 1057 keyframes are extracted from the video using the GRIC algorithm presented in paragraph 4. The resulting images are segmented by the STBOL algorithm (Figure 51) into different classes. Following, the output classes are used to mask all non-building classes in the imagery (Figure 52). Given the exterior camera orientation of the sparse reconstruction and the masks, a set of depth maps are produced depicting the building geometry (Figure 52). Finally, a reduced dense point is computed. The result is a reduced dataset which better corresponds to the asset the user wants to reconstruct if requested (Figure 53)



Figure 50: Overview Image Sequence Gendarmenmarkt from extracted keyframes



Figure 51: Overview Spatio-Temporal Building and Object Localization (STBOL) results identifying persons, sky and other non-building object types.



Figure 52: Overview image masks removing all non-building entities from the imagery.



Figure 53: Depth Map Generation from masked imagery.

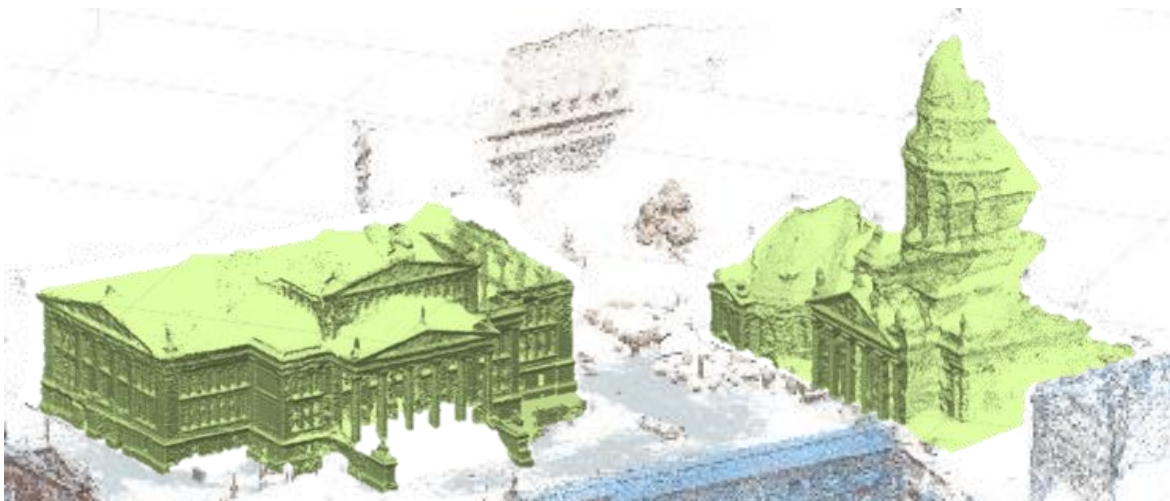


Figure 54: Overview resulting data reduction of the final reconstruction compared to the initial SfM pipeline.

For the sake of the experiment, the building point clouds are compared to the initial reconstruction. Table 10 shows the difference in computation time between both models. It can be clearly observed that a significant time reduction can be achieved through segmentation. Furthermore, some imagery does not contain any building geometry and thus

can be ignored during the dense reconstruction. Overall, no less than 29% of the scene could be removed based on the STBOL detection. Not only does this improve the data efficiency, it also supports further segmentation into separate buildings since a significant portion of the noise and clutter is removed from the point clouds.

Table 10: Overview comparison between initial reconstruction and segmented building point clouds

Dataset	Images	STBOL Buildings	Initial point cloud	Time [s]	Targeted Building Reconstruction point cloud	Time [s]	%Scene reconstruction
Gendarmenmarkt	1057	957	667 345	0:15:35	405 326	0:05:12	29%

6.2.4 Future work enhanced model extraction

The current segmentation targets building geometry driven by an image based interpretation framework. It is important to notice that the buildings class is an archetype and thus there is no differentiation between a specific building and the structure next to it. As a result, a user initiating a reconstruction based on imagery of the Gendarmenmarkt or even the Konzerthaus will be delivered a model that contains the target structure and all the building geometry surrounding it that happens to occur in the crawled imagery. A valuable extension is to further extend the segmentation based on the keywords entered by the user. For instance, the keyword Konzerthaus should only provide 3D results for the structure itself opposed to the entire site. To enable this segmentation, Geospatial databases may be employed. Given

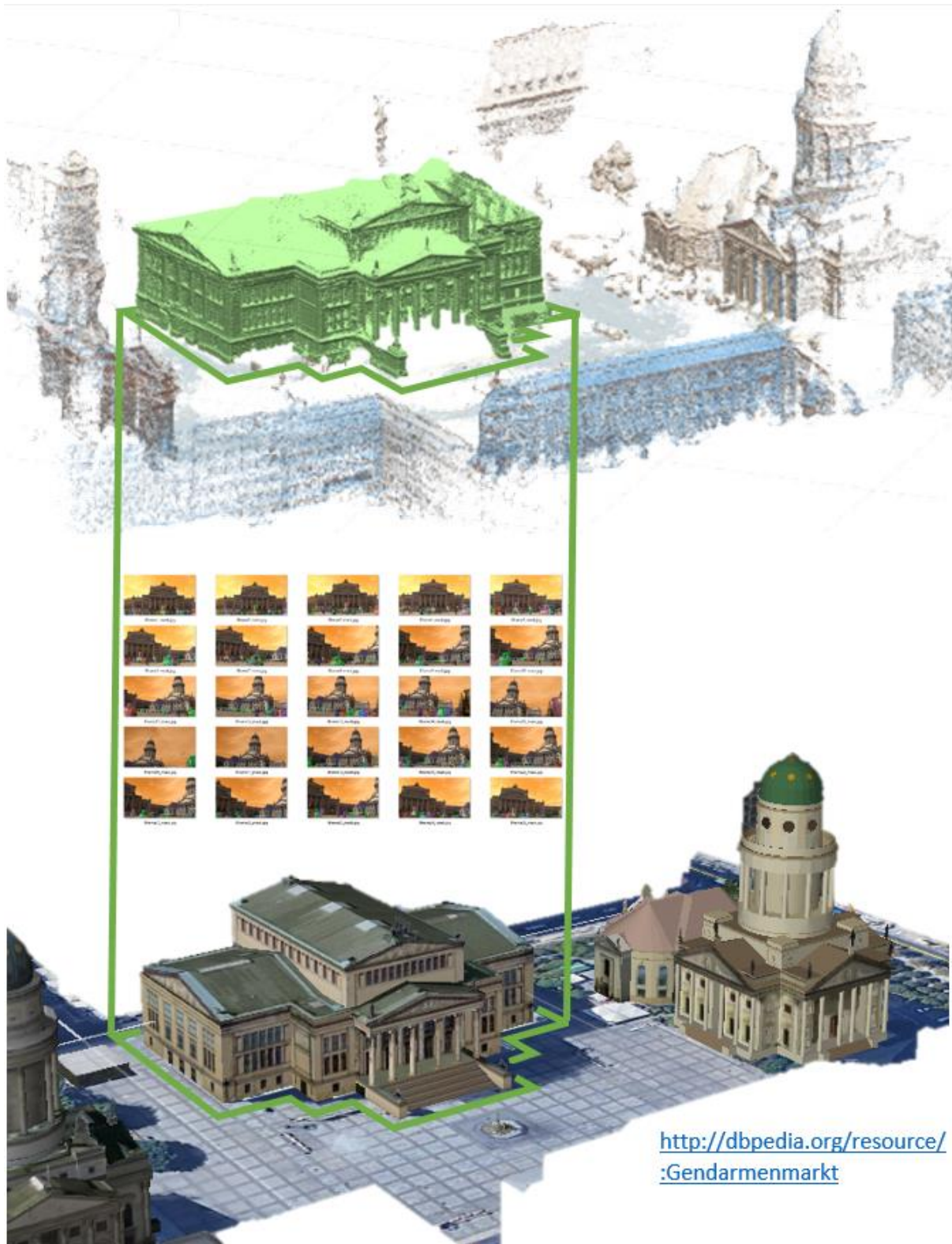


Figure 55: Overview future work enhanced model extraction through the use of Geospatial databases.

the geolocated position of the target asset, a spatial segmentation can be performed in addition to the building detection by STBOL. This is under the assumption that the crawled footage can be spatially linked to existing geolocated resources (Figure 55). If this is the case,

the building footprint or the bounding box can be used to isolate only those building points that are part of the intended structure. Alternatively, Semantic Web technologies can be used to the same effect. Given geolocation triples on dbpedia or other resources, similar spatial search queries can be initiated to segment the data. However, licensing permissions are mandatory in order to use online geospatial repositories. WP9 focusses on licensing and IP that will enable this extension.

6.3 Texture enhancement

The proper retrieval of color information for the final mesh is considered an instance of texture optimization. Initially, texture was retrieved using vertex rendering methods. However, current methods nearly all rely on reprojecting images. Typically, all imagery is considered for the texturing but as stated above, this is subideal for crawled imagery. As a solution, weighted texturing is proposed (Baumberg, 2002). This is not applicable to the V4Design data unless the weights of some textures are considered 0, and also this method is prone to texture inconsistencies (Zhang, 2017). As a solution, priority imagery is proposed to incrementally texture portions of the mesh taking into consideration the visibility conditions between the images and the 3D surface model. As a criterion, the highest image quality in terms of Ground Sampling Distance is proposed.

$$GSD = \frac{D * CCD_{i,j}}{f * res_{i,j}}$$

Where D is the distance to the objects of interest, $CCD_{i,j}$ the sensor width [mm] along the width or height of the sensor, f the focal length [mm] and $res_{i,j}$ the number of pixels along the width or height of the image. Iwaszczuk et al. (Iwaszczuk, 2015) extend this with completeness of the texture, projection accuracy, viewing angle, and geometric resolution. This operation requires the processing of the candidate images to produce uniform brightness, contrast and saturation and to select the portions of the partial UV map corresponding to the central portion of the respective image to reduce the influence of the residual radial distortion (Caroti, 2015). One drawback of these blending approaches is their sensitivity to ghosting and blurring artefacts when textures are misaligned (Figure 56).



Figure 56: Example of misaligned textures resulting in a blurry texturing of the mesh object (Gal et al., 2010)

To overcome the artifacts introduced by per-triangle texturing, Lempitsky et al. (Lempitsky, 2007) consider the paradigm as a global optimization problem that can be solved with Markov Random Fields. Each surface triangle is projected back onto the images from which it is visible. A minimum set of image is sought that minimizes image blur across triangles. Gal et al. (Gal, 2010) further extend this by compensating for calibration and reconstruction errors, showing promising results.

6.3.1 Methodology

In this section, the implementation of the texturing is discussed. In the prototype SfM pipeline, the Open-source texture functions of Meshroom are used as implemented by Levy et al. (Levy et al., 2002). The Least Squares Conformal Maps for Automatic Texture Atlas Generation including the segmentation, parameterization and image projection are discussed below.

Chart Segmentation

The first step in segmentation is to divide the mesh into a set of homeomorphic parts, referred to as charts. The combination of texture charts is referred to as the texture atlas, which can be stored in well-known file formats such as .jpg or .png. The size of each chart is either user driven or dependent on the average texel size of the textured model. The latter is preferred since the texturing can then be formulated as a function of the input image resolution and the triangle size.

$$\text{texture quality} = \frac{\text{target output pixels per triangle unit}}{\text{average input pixels per triangle unit}}$$

An advantage of this method is its invariance to scale, which is crucial in the V4Design pipeline, as predominantly unscaled models will be produced. Also, it automatically takes the distance from the camera to the model into consideration since the average texel size is used. Given a user defined quality setting for the model, the total size of texture space is computed. This feature space is evenly distributed into 2048, 4096 or 8192 texture files to form the texture atlas.

Chart Parameterization

Each chart is unfolded with respect to its subset in the mesh. Meshroom employs conformal maps where the tangent vectors to the iso-u and to the iso-v curves are orthogonal and have the same length (Figure 57).

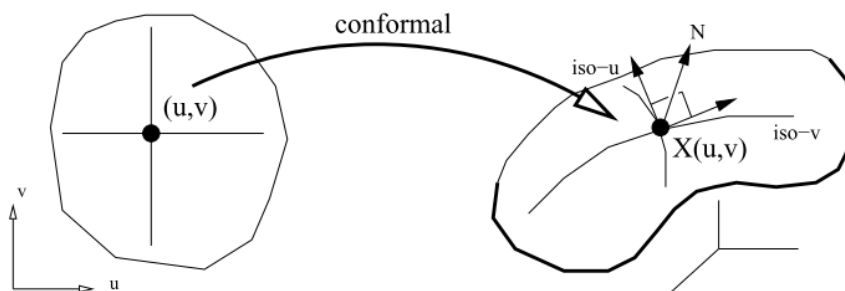


Figure 57: Conformal map relation to mesh surface as described by Levy et al. (Levy et al., 2002)

The mapping of the triangles to the conformal map is performed using a least squares optimization problem. A subsegmentation is performed that clusters triangles onto a portion of the chart to ensure that neighbouring triangles are textured using the same image. This significantly reduces artefacts in the final texturing and improves the computational efficiency of the image selection.

Image reprojection

Once a set of uv maps is constructed representing the mesh surface, the imagery is reprojected to texture each triangle. As discussed above, the best fit imagery is chosen for each triangle. Meshroom filters the cameras without a good angle to the surface to favor front-to-parallel cameras and finally averages the pixel values (Figure 57). A maximum of 3 images are considered for every triangle. The final texturing is computed by minimizing the error for the angle θ to the object, the distance D to the object and the offset from the camera center to the projected pixel.

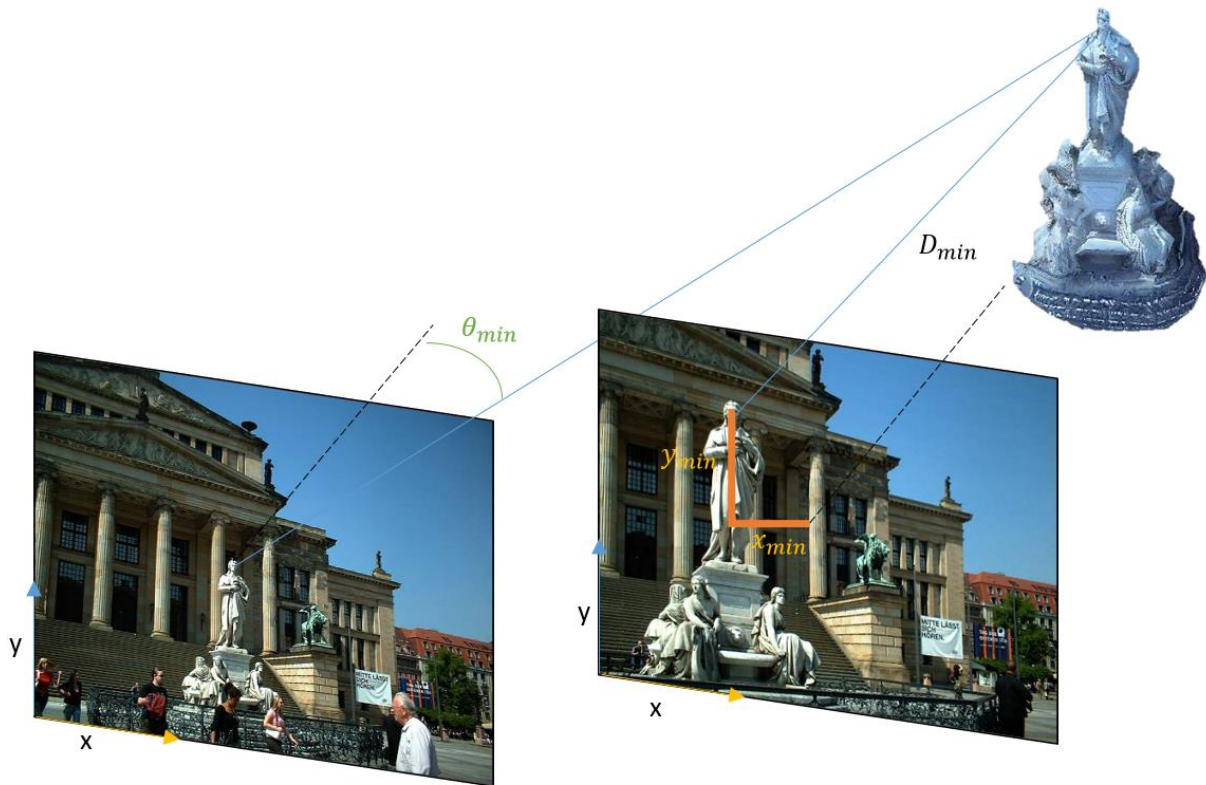


Figure 58: Image reprojection parameters for best fit texturing.

6.3.2 Texture Experiment

Two experiments are conducted for the texturing of the 3D reconstruction. The first test is a video sequence taken by a drone and the second is the crawling data dump which is also discussed in section 5.3. Both tests are discussed below.

Drone footage Bauhaus universität

The drone footage originates from a drone recon flight conducted in September 2017. It is a round flight of the Bauhaus in broad daylight. It is expected that proper texturing results are achieved for this data set due to the ideal circumstances. 103 frames are extracted using the GRIC algorithm after which the sparse reconstruction was performed in COLMAP and the dense reconstruction computed in Meshroom. The texturing was performed with the Meshroom Least Squares Conformal Maps for Automatic Texture Atlas Generation. A texture quality of 100% was selected for the colorization of the mesh.

Figure 59 shows one frame of the flight, along with the texture quality of the 3D reconstruction. It can be observed that the texture quality nearly equals the initial imagery. Upon more detailed inspection, there are some blending errors and artefacts but these are very limited. This is due to the consistent range at which the footage was acquired, the highly accurate exterior orientation and the uniform lighting in the imagery. It is stated that no texturing problems occur for these types of inputs.



Figure 59: Texturing results Bauhaus universität: (top) image from drone, (left) zoom in of same image, (bottom) textured mesh and (right) call out of textured mesh

Crawled imagery Gendarmenmarkt, Berlin

The second experiment investigates the ability of Meshroom to texture imagery from different repositories. The same image crawling batch is used as discussed in paragraph 5.3. Figure 60 and Figure 61 depict the wide variety of textures that occur in the data set. The colors range from near white all the way to near black.



Figure 60: Gendarmenmarkt image batch including images under varying lighting conditions: (left) normal daytime lighting and (right) dark evening light.

Figure 62 shows the resulting textured mesh of the structure. While the geometry of the structure is properly reconstructed, the texture is not. Some of the brick texture can still be distinguished but it is clearly observed that the texture has become a blur of the different image textures. Different shades coexist in the texture ranging from dark grey from the imagery taken at night to a near white texture during the day. This is expected due to the high texture variance of the inputs. Overall, it is stated that the texture quality is inconsistent in color and blurry. Texture enhancements are in order to improve the results.



Figure 61: Gendarmenmarkt image batch including images under varying lighting conditions: (left) warm evening light and (right) cold daytime.



Figure 62: Resulting texture using all aligned input imagery showing blurred colours.

6.3.3 Future Work texturing

The above presented method succeeds in compensating stitching errors between similarly textured images. However, the main issue with crawled imagery is that some textures cannot coexist. Depending on the acquisition conditions and the time period, subsets of the input data are not supposed to contribute to a single texturing of the model. There are several opportunities to deal with this problem. We propose the use of subsets of texture styles in the V4Design project, which can be used to create different models. This would comprise of an initial best fit texturing for the asset. Subsequently, the texture could be replaced by the different texture styles. As it is likely for texture subsets to lack sufficient imagery to texture the entire asset, the style can be detected and superimposed on the initial texturing to fill the gaps. The style can also originate from fictional inputs such as paintings. As a result, different model textures are computed for the mesh model.

The AE&TP (D3.2) algorithm can be extended to interpret the building textures and segment them into preset styles. As input, the full images should be used since the surroundings of a building give distinct cues about the style of the representation. For instance, the sky is a vital clue for the identification of day or night imagery. Following textures could be identified.

- Realistic daytime: The most used texture for building models is realistic daytime. This implies imagery taken during the day with sufficient quality and resolution. It serves as the basis best fit texture that can also be used for the texture weight factoring.
- Evening: A common class is imagery taken during the evening or nighttime. It is a valuable representation since game developers and Architects are interested in the unique lighting conditions of an asset during the night.
- Historic: Historic images differentiate from other imagery as they often do not contain color or depict the buildings without the pathologies that are commonplace for historic buildings. This is especially interesting for game development that often focuses on scenery from the past.
- Special: In addition to evening and daytime pictures, the image batch also contains special texturing such as images taken during a light festival and so on. These textures are fairly unique and thus hard to imitate. However, they need to be segregated since otherwise they would have a significant impact on the final texture mosaic.
- Painting styles: In addition to the styles extracted from the input imagery, the database is further expanded with fictional styles derived from paintings or other inputs. Concretely in the V4Design project, painter styles are learned from their respective works to publish creative content for the users.

The subsets are prone to leaving gaps in the texturing since the images batches aren't sufficiently large or do not cover the entire asset. As a solution, we propose the superimposition of the texture styles onto the initial texture. Similar to the painting styles, the categories of night imagery and historic imagery can be extracted from the subset of images. The initial best fit textures could be altered in such a way that they resemble the texture of the subset. However, they would be given a low weight so they can only be used to texture the gaps of the model.

7 CONCLUSION & FUTURE WORK

7.1 Conclusions

In this first iteration of the automated 3D reconstruction pipeline, we thoroughly assessed the State-of-the-Art of Structure-from-Motion methods for their ability to process video sequences and crawled image data sets and produce textured 3D mesh models. From the experiments, it was derived that the most promising method is to combine and extend existing algorithms. More specifically, COLMAP was implemented for the sparse reconstruction and Meshroom for the dense reconstruction. The algorithms from both softwares were integrated and automatically process any image batch when called upon by the message bus developed in WP7. Our method outperforms the individual State-of-the-Art approaches in terms of effectiveness and efficiency. This is especially true for the crawled imagery from varying sources with different lighting conditions, quality and from different time periods that severely obstruct conventional SfM pipelines.

There are several key contributions in the proposed 3D reconstruction pipeline. For the video processing, we extract keyframes through shot detection and GRIC baseline detection. We also assess the shots to dispatch degenerate parts and evaluate the blurriness of the resulting keyframes. In the image preprocessing, the STBOL outputs (D3.2) are used to compute masks to enhance the model extraction. More specifically, the method is made computationally more efficient by using the full images for the sparse reconstruction and the masked imagery for the dense reconstruction. The resulting reconstructions only depict objects from a single STBOL class i.e. buildings which is both faster and cleaner than the current reconstructions. The subsequent algorithms are governed by independent communication modules run on KUL servers that relay their respective process metadata through the message bus and thus allow swift and flexible processing. Furthermore, the communication modules are developed in such a way that we are able to batch process input messages and populate the knowledge base. A final contribution in the first iteration is the texture enhancement of the 3D models. The outputs of the AE&TP algorithm developed in D3.2 are used in combination with the native Meshroom texture functions to compute multiple textures for the same 3D models. The resulting high quality textures from reused imagery and fictional textures can be used for conceptual design as specified by the user requirements.

7.2 Future Work

There are several opportunities to upgrade the SfM pipeline in the operational prototype by M28. First of all, we plan on expanding the input filtering, as it is vital to the success of the method. It is worthwhile to notice that the Gendarmenmarkt and the BrandenBurger Gate, despite the large number of irrelevant images, are well documented assets for which numerous proper images exists. However, this is unlikely the case for many other key-word based crawlings. One suggestion is to develop a prior SfM evaluation tool that provides feedback to the user on the feasibility of the reconstruction given the initial results of the crawling. This proposition is based on the fact that the user is probably unfamiliar with which imagery is suitable for 3D reconstruction and thus is prone to engage the crawling with inconvenient key-words. It is therefore better to warn the user about the expected results

rather than disappointing an unexperienced user after the several hours SfM process fails to provide 3D results. Similarly, we are currently investigating the opportunities to enable process feedback during the reconstruction. More specifically, we are working on an online sparse and dense point cloud preview so users can evaluate their intermediate results. Another upgrade is the extension of the COLMAP image matching procedure that currently considers all inputs to be unordered data sets. However, some imagery contains EXIF data with location information that can be used to enhance the image matching. Furthermore, the location information allows the scaling of the 3D models that are currently predominantly unscaled. Single view reconstruction is also being considered since the scope of V4Design extends towards the processing of imagery of artwork e.g. pottery which is often documented with a single image.

For the enhanced model extraction, the following opportunities are under investigation. The segmentation can be significantly enhanced by integrating location information in the process either through Linked Data Resources or the Geolocation of the imagery. Also, several texture implementations are proposed to enhance texturing results and allow users to choose the best suited texture for their applications.

REFERENCES

- J.L. Schönberger, and J-MFrahm, „Structure-from-Motion revisited“, In proc. Conference on Computer Vision and Pattern Recognition, pp. 4104-4113, 2016
- D. G. Lowe, “Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, 60 (2004), pp. 91–110.
- H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF)”, Computer Vision and Image Understanding, 110 (2008), pp. 346–359.
- N. Ramakrishnan, T. Srikanthan, S.K. Lam, and G.R. Tulsulkar, “Adaptive window strategy for high-speed and robust KLT feature tracker” in Lecture Notes in Computer Science, 2016, vol 9431, pp. 355-367.
- J. L. Schönberger, T. Price, T. Sattler, J.-M. Frahm, and M. Pollefeys, “A vote-and-verify strategy for fast spatial verification in image retrieval”, In Asian Conference on Computer Vision (ACCV), 2016.
- D. Nistér. "An efficient solution to the five-point relative pose problem". IEEE Transactions on Pattern Analysis and Machine Intelligence. 26 (6): 756–777, 2004.
- B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, “Bundle Adjustment – A Modern Synthesis”, Springer, 2010.
- N. Jiang, Z. Cui, and P. Tan, “A global linear method for camera pose registration” In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 481–488, 2013.
- H. Cui, X. Gao, S. Shen and Z. Hu, “HSfM : Hybrid Structure-from-Motion”, In Proceedings of CVPR, pp. 1212–1221, 2017.
- F. Remondino, E. Nocerino, I. Toschi, and F. Menn, “A critical review of automated photogrammetric processing of large datasets”, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, 42(2W5):591–599, 2017.
- D. Shin, and J. Muller, “An explicit growth model of the stereo region growing algorithm for parallel processing”, in ISPRS Technical Commission V Symposium, XXXVIII, 2010.
- V. Kolmogorov and R. Zabih, “Multi-camera scene reconstruction via graph cuts”, Proc. ECCV 2002, pp. 8–40, 2002.
- D. Tingdahl, and L. Van Gool, “A public system for image based 3D model generation”, In Lecture Notes in Computer Science, 6930, pp. 262-273, 2011.
- M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson surface reconstruction”, in Symposium on Geometry Processing, 2006, pp. 61–70.
- V. Lempitsky and D. Ivanov, “Seamless mosaicing of image-based texture maps”, in Proc. of CVPR, 2007.
- FFMpeg-developers, “FFmpeg 4.0.2”, www.ffmpeg.org, 2019.
- N. V. Patel, and I. K. Sethi, “Video shot detection and characterization for video database”, in Pattern Recognition, 30(4), pp.583-592, 1997.

- S. Tsekeridou, and I. Pitas, “Content-based video parsing and indexing based on audio-visual interaction”, in IEEE Transactions on Circuits and Systems for Video Technology, 11(4), pp. 522-535, 2001.
- Z. Cernekova, and I. Pitas, “Information Theory-Based Shot Cut/Fade Detection and Video Summarization”, in IEEE Transactions on Circuits and Systems for Video Technology, 16(1), pp. 82-91, 2006.
- E. Rosten, R. Porter, and T. Drummond, “Faster and better: a machine learning approach to corner detection” in IEEE Trans. Pattern Analysis and Machine Intelligence, 2010, vol. 32, pp. 105-119.
- J. Shi and C. Tomasi, “Good Features to Track”, IEEE Conference on Computer Vision and Pattern Recognition, pages 593–600, 1994.
- P. Torr, “An assessment of information criteria for motion model selection”, Proc. CPVR, pp47-53, 1997.
- R. Hartley, and A. Zisserman, “Multiple View Geometry in Computer Vision”, Cambridge University Press, ISBN 0521540518, 2004.
- R. Arriew, “Ockham’s razor: a historical and philosophical analysis of Ockham’s principle of parsimony”, PhD thesis, University of Illinois, 1976.
- H. Akaike, “A new look at the statistical model identification”, IEEE Transactions on Automatic Control, 19(6), pp 716-723, 1974.
- J. Rissanen, “Modeling by the shortest data description”, in Automatica, vol 14, pp. 465-471, 1978.
- J. Aldrich, “R. A. Fischer and the making of maximum likelihood 1912-1922”, in Statistical Science, 12(3), pp. 162-176, 1997.
- Rashad, M., Khamiss, M., & Mousa, M. (2017). A review on image segmentation techniques. *International Journal of Engineering and Innovative Technology*.
[https://doi.org/10.1016/0031-3203\(93\)90135-J](https://doi.org/10.1016/0031-3203(93)90135-J)
- Baumberg, A. (2013). Blending Images for Texturing 3D Models. In *Proceedings of the British Machine Vision Conference*, 38.1-38.10. <https://doi.org/10.5244/c.16.38>
- AliceVision. (2019). Meshroom. AliceVision. <https://alicevision.github.io/#meshroom>
- RealityCapturing. (2017). Capturing Reality. <https://www.capturingreality.com>
- Pix4D (2016) Pix4D: Professional photogrammetry and drone mapping software.
<https://www.pix4d.com/>
- Agisoft. (2018). Metashape. Retrieved from <https://www.agisoft.com/>

- Schonberger, J. L., Zheng, E., Pollefeys, M., & Frahm, J.-M. (2016). Pixelwise View Selection for Unstructured Multi-View Stereo. *European Conference on Computer Vision (ECCV)*. <https://demuc.de/COLMAP/>
- 3Dflow. (2014). 3DF Zephyr. <https://www.3dflow.net>
- Levy, B., Petitjean, S., Ray, N., & Maillot, J. (2002). Least Squares Conformal Maps for Automatic Texture Atlas Generation Bruno. *ACM SIGGRAPH 2005 Courses, SIGGRAPH 2005*. <https://doi.org/10.1145/1198555.1198581>
- Caroti, G., Martínez-Espejo Zaragoza, I., and Piemonte, A. 2015. Range and image based modelling: a way for frescoed vault texturing optimization, *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XL-5/W4, 285-290, doi:10.5194/isprsarchives-XL-5-W4-285-2015, 2015.
- Iwaszczuk, D., Hoegner, L., & Stilla, U. (2015). Quality-based building-texture selection from different sensors. *2015 Joint Urban Remote Sensing Event, JURSE 2015*, 1–4. <https://doi.org/10.1109/JURSE.2015.7120352>
- Zhang, W., Li, M., Guo, B., Li, D., & Guo, G. (2017). Rapid texture optimization of three-dimensional urban model based on oblique images. *Sensors (Switzerland)*, 17(4). <https://doi.org/10.3390/s17040911>
- Schmid, C., Verbeek, J., Revaud, J., & Oneata, D. (2014). *Computer Vision – ECCV 2014. ECCV 2014 - European Conference on Computer Vision* (Vol. 8691). <https://doi.org/10.1007/978-3-319-10578-9>
- Jodoin, P. M., Piérard, S., Wang, Y., & van Droogenbroeck, M. (2014). *Background Modeling and Foreground Detection for Video Surveillance: Overview and benchmarking of motion detection methods*. CRC Press. <https://doi.org/10.1201/b17223>
- Rashad, M., Khamiss, M., & Mousa, M. (2017). A review on image segmentation techniques. *International Journal of Engineering and Innovative Technology*. [https://doi.org/10.1016/0031-3203\(93\)90135-J](https://doi.org/10.1016/0031-3203(93)90135-J)
- Nelson, R., Polana, R.: Qualitative recognition of motion using temporal texture. *CVGIP: Image Understanding* 56, 78 (1992)
- Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.