




Methodology for capturing contextual information and preserving and interlinking semantic information for video processing

Authors

Ondrej Klima, Pavel Smrz (Brno University of Technology)

May 2014

This work was partially supported by the SCAPE Project. The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137).

This work is licensed under a CC-BY-SA International License 

Executive Summary

This deliverable deals with methodology for processing and interlinking video content in external computation facilities. It focuses on analysing video frames in two particular sets of experiments – deriving 3D model from the scene captured in the video and annotating separate video frames that capture large-scale natural environments, such as a whole range of mountains. We briefly introduce basics of employed video processing methods that are necessary for understanding identified risks, especially those arising from performing the large-scale experiments in external data centre facilities. The methodology also addresses reproducibility of large-scale experiments and preservation of experiment environments and circumstances. After identifying possible risks, we propose a general strategy and identify software tools which minimize risks' impact. The software solution takes advantage of the SCAPE platform, particularly benefitting from its scalability and performance. In order to demonstrate functionality and usability of the software solution, two data sets related to the 3D reconstruction and the video annotation, respectively, are identified and described too. We discuss executable workflows, providing analysis of the mentioned data sets, while using the software preservation solution.

Table of Contents

Executive Summary	3
1 Introduction	6
1.1 Outline	6
2 Contextual and semantic analysis of video content	7
2.1 3D Model Reconstruction	7
2.1.1 Feature extraction and feature tracking	8
2.1.2 Bundle adjustment	9
2.2 Annotation of videos captured in natural environments	10
2.2.1 Rendering synthetic panorama and extracting features.....	11
2.2.2 Aligning video frames to synthetic panorama	12
3 Experimental data sets for video semantic analysis	13
3.1 The Málaga Stereo and Laser Urban Data Set	13
3.2 The Alp Mountains Image Data Set.....	14
4 Identified preservation risks of video processing in remote data centres	16
4.1 Data centre environment threats	16
4.1.1 Threats related to static properties	17
4.1.2 Risks related to dynamic characteristics	20
4.2 Input and output of experimental data	21
4.3 Non-technical attributes of experiments.....	22
4.4 Links between source data, cluster environment and results.....	23
5 Tools for minimizing impact of identified preservation risks	24
5.1 Preservation tool requirements.....	24
5.1.1 Solution generality	24
5.1.2 Ability to process large data	24
5.1.3 Comfortable user interface	25
5.2 Software architecture	25
5.2.1 C++ library	25
5.2.2 Hadoop tools	26

5.3	Toolkit functionality	26
6	Taverna workflows related to semantic video analysis	27
6.1	3D reconstruction workflow	28
6.2	Workflow of annotations of natural environments.....	29
7	Conclusions.....	30

1 Introduction

This deliverable describes a methodology for preserving large-scale experiments performed in external data centres. A key objective is to identify how particular data centre settings influence results and to guarantee reproducibility of experiments. Video semantic processing is employed as a typical representative of resource-intensive computation task that can benefit from specialised data centres. In particular, two sets of experiments conducted in our work deal with deriving 3D models of actual scenes from video data and with annotating videos of large-scale natural environments. Relevant video analysis methods are briefly presented in the text.

The methodology first identifies preservation risks related to performing video analysis in remote clusters that are not under control of the institution responsible for the experiments, risks related to currently used cluster infrastructure and risks and performance characteristics related particularly to the 3D reconstruction and video annotation. The second part describes a software solution minimizing the identified preservation risks. Requirements on such a software solution, its architecture and functionality are detailed. The presented methodology provides a basis for actual deployment and integration of preservation scenarios within the data centres that will be reported in deliverable D23.2.

1.1 Outline

The next section introduces the methods for large-scale 3D model reconstruction and for annotation of large-scale natural environments. These methods are currently a subject of intensive research. To process large video data, they typically consume significant amounts of system resources and thus benefit from “off-loading” to external data centres. Experimental applications used in *Taverna* workflows discussed in the next section are implementations of these methods.

The third section discusses two data sets that will be used together with experimental applications in executable *Taverna* workflows for demonstrating the functionality of deployed software solution.

The fourth section essentially covers the first part of the preservation methodology. Individual categories of identified risks are detailed in related subsections – risks related to the cluster environment and infrastructure, risks related to losing context of experiments, and risks related to input and output data.

Fifth section describes architecture and functionality of software tools minimizing the preservation risks. Benefits of using the SCAPE platform are also discussed together with implementation details relevant for successful deployment.

The sixth section contains *Taverna* workflows related to 3D-reconstruction and video-annotation experiments.

We conclude with a summary of the proposed methodology.

2 Contextual and semantic analysis of video content

This section discusses basics of 3D-scene-reconstruction and video-annotation methods that are used in experimental applications demonstrating usage and functionality of the deployed software solution for large-scale experiment preservation based on the SCAPE platform.

2.1 3D Model Reconstruction

Inferring an actual shape of a scene captured in video is commonly referred to as the *structure from motion problem*. It is closely related to *SLAM – Simultaneous Localization and Mapping* – explored in the robotics research. Results of the semantic video analysis correspond to a sparse point cloud representing a shape of a captured object and a trajectory of the camera. Example of a point cloud resulting from such an analysis is shown in **Figure 1**.

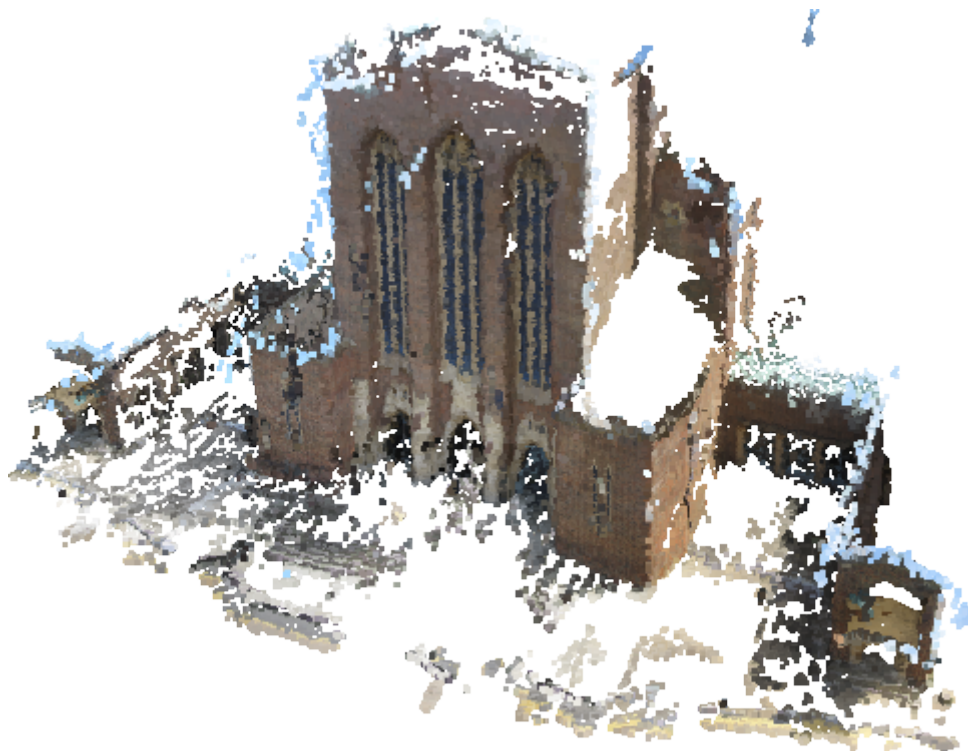


Figure 1: 3D reconstruction of a cathedral in Scotland

Resulting 3D models are used in digital cinematography, in advanced robotics, and in many other areas. Modern cinematography particularly employs 3D models when creating visual effects. The special effects are based on rendering animations created using 3D models – thanks to that it is possible to create effects like natural disasters, when the asteroid destroys a city, mountains sink to an ocean, etc. It is significantly cheaper to obtain a needed 3D model of a city or of mountains automatically using the structure from motion approach than to create it manually in 3D modelling software.

Automatic reconstruction methods are inspired by the biological phenomenon by which people recover the actual shape of a 3D object moving it in front of their eyes. Computer vision uses a similar approach. The camera goes systematically around a scene to capture every part of the surrounding environment, as it is illustrated in **Figure 2**.

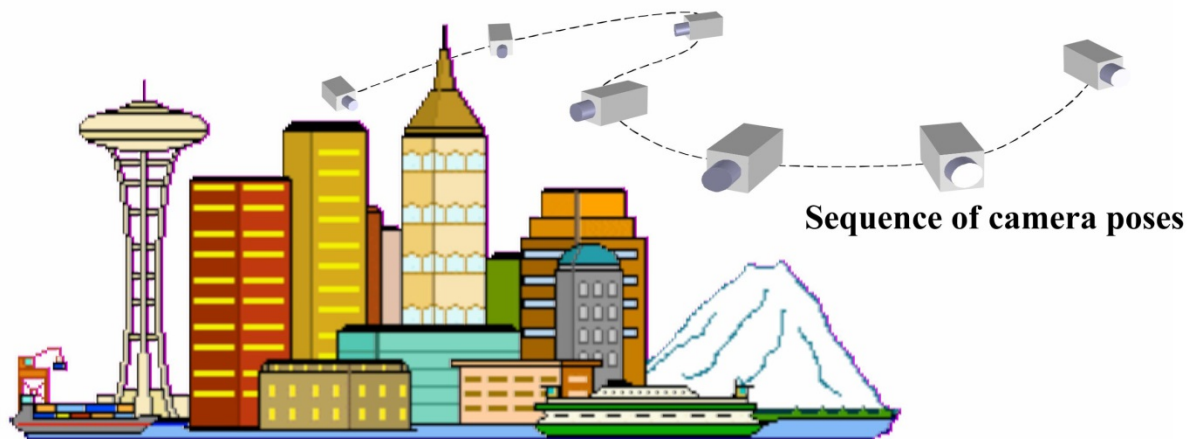


Figure 2: Camera going systematically around a captured scene

There are three essential steps that have to be taken in order to obtain reconstructed sparse point cloud from captured video frames:

1. **feature extraction,**
2. **feature tracking,**
3. **bundle adjustment**

2.1.1 Feature extraction and feature tracking

Feature extraction is the first step of the whole structure from motion analysis. Desired features are typically points in significant places, which characterize the geometry of captured scene. They usually correspond to corners of objects present in the scene. The features are extracted from separate frames of processed video. Typically, the *SIFT* algorithm (*Scale-Invariant Feature Transform*) is used to extract local feature points. It is also the case of the experimental application used in the executable *Taverna* workflow discussed in this deliverable.

Once the feature points are extracted, it is necessary to find correspondences between points in different video frames, so each feature point can be tracked across the frames in some neighbourhood. Not all feature points can be found in all frames of captured video because of occlusions. That is true especially in the case of large-scale video with huge amounts of separate frames. Special mechanisms need to be applied to cope with these characteristics. A large sparse matrix containing 2D coordinates of each feature point across video frames is established in the end of this step.

2.1.2 Bundle adjustment

The final step of the structure from motion analysis leading to the resulting point cloud identifies (recovers) 3D points from the matrix of tracked 2D feature points introduced in the previous subsection. Bundle adjustment is a state-of-the-art method for the 3D reconstruction that was firstly used in photogrammetry. It can be formulated as an optimization problem minimizing re-projection error of estimated 3D points coordinates and refining the resulting model.

The optimization is usually carried out by the *Levenberg–Marquardt* algorithm (*LMA*). It typically consumes a significant amount of time and system resources. Moreover, when dealing with large scenes, the final quality of the 3D reconstruction generally depends on the time granted for the process of optimization and on the performance of currently used computer(s).

Various modifications of the basic bundle adjustment method, like *incremental bundle adjustment* or *sparse bundle adjustment*, have been devised to reduce time- or memory-consumption caused by the optimization process. Large-scale experiments usually run on computers equipped by advanced GPUs that have the potential to significantly speed up the whole process.

Current research focuses on reconstructing large-scale scenes in a parallel manner by dividing the scene into smaller parts that can be reconstructed independently. Computer clusters (GPU clusters) are employed to reconstruct individual components that are then merged to form the complete 3D model of the large scene. This setting is also explored in our preservation experiments.

2.2 Annotation of videos captured in natural environments

The second contextual video analysis deals with annotating video frames capturing large-scale natural environments, such as ranges of mountains. An example of the annotation can be found in Figure 3 taken on the top of the Großglockner Mountain, the highest mountain of the Austrian part of Alps.

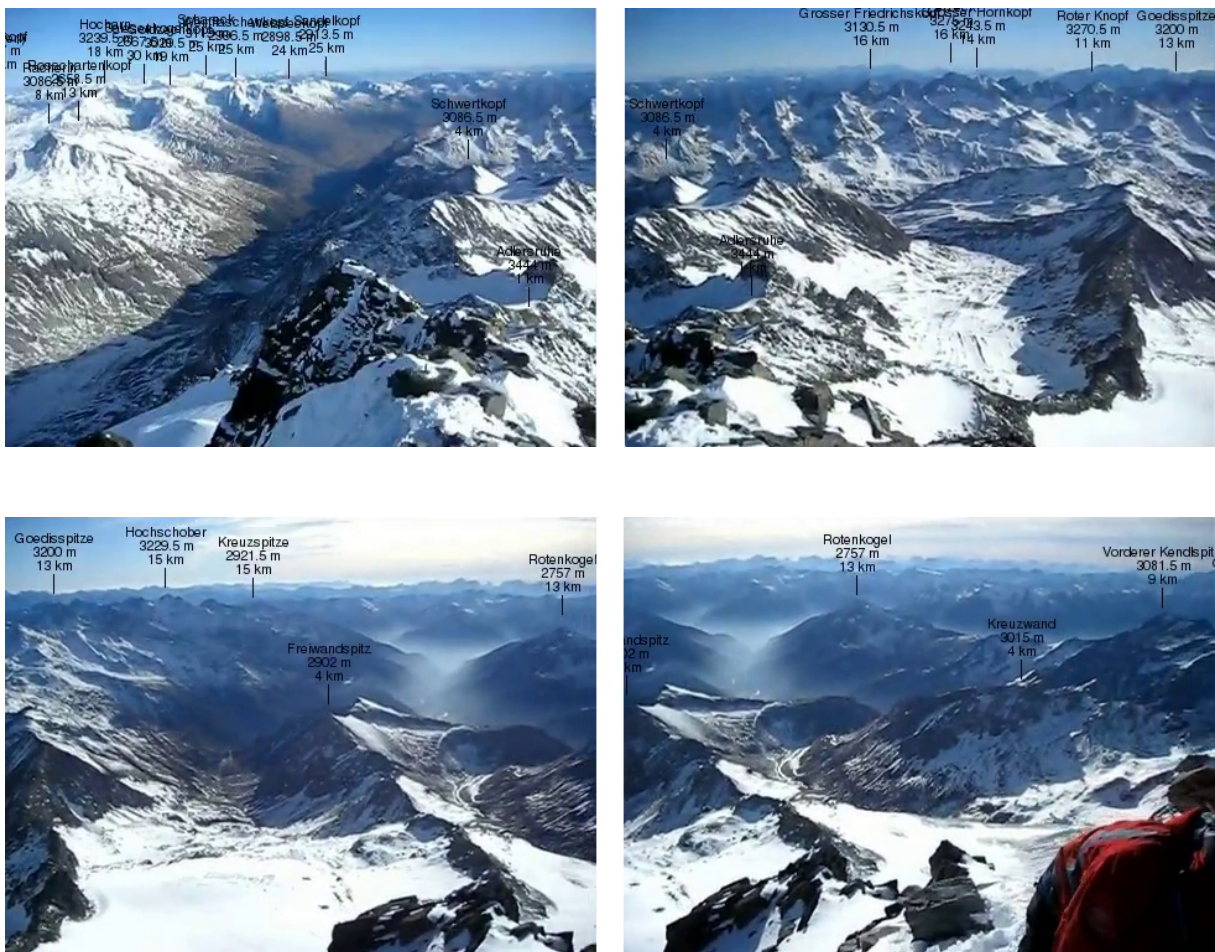


Figure 3: Annotated frames from a video captured on the top of the Großglockner Mountain

Automatic annotation is based on fitting real 2D images to a synthetic large-scale 3D model of the entire mountains. The 3D model is derived from a high-resolution *DEM* (*Digital Elevation Model*) of Alps. The model provides positions of prominent summits. After a video frame is aligned to the 3D model of mountains, annotations are copied from the model to the current image. Currently, it is assumed that the camera position is known in advance – modern cameras are equipped with the GPS module or the longitude and the latitude can be assigned manually.

There are four main steps that have to be done to annotate one frame of captured video:

1. **Rendering synthetic panorama from the 3D model,**
2. **Adjusting the video frame field-of-view,**
3. **Extracting features from the adjusted video frame,**
4. **Aligning the video frame to the synthetic panorama image,**
5. **Copying annotations from the panorama image to the aligned frame.**

2.2.1 Rendering synthetic panorama and extracting features

The polygonal model of mountains is derived from the Digital Elevation Model (DEM) first. The DEM is stored as a set of grayscale images in the PNG format. Each PNG image describes one tile of mountains relief; every tile captures an area which is one degree of latitude wide and one degree of longitude high. The lowest places of mountains are described by dark shaded colours; the highest are described by bright colours. Each PNG image has square format with width and height equal to 3600 pixels.

After the polygonal model is derived, the synthetic panorama for an individual viewpoint can be rendered. The viewpoint needs to correspond to the world coordinates where the real video was captured; therefore it is necessary to know the actual position of camera. Only contours of panorama are rendered as the further aligning is based on matching edges between the synthetic image and the captured video frame. The edges are extracted from polygonal model using the raycasting method. The polygonal model with annotations can be seen in **Figure 4a**, raycasted edges with annotations are shown in **Figure 4b**.

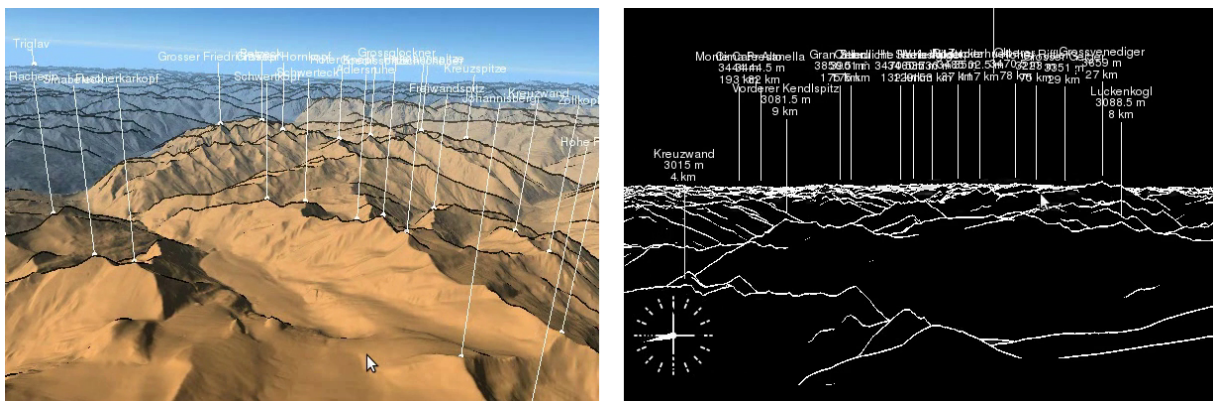


Figure 4: Polygonal model of mountains in the left (a) and rendered silhouettes in the right (b).

Before features can be extracted from a video frame, it is necessary to warp the frame in order to normalize the field-of-view of the real camera with the virtual camera used for rendering the synthetic panorama. After the warping, edges are detected and extracted. The *Compass* edges detector is typically used for the extraction. It is able to work directly with colour images while most of other detectors only deal with grayscale pictures. An example of the edge detection processes using the *Compass* method is shown in **Figure 5a**.

It can be seen that edges detected in the video frame contain a huge amount of noise when compared to the raycasted edges from the polygonal model. Detected edges therefore need to be further processed. First, the edges are compared to a certain strength threshold and those that are not strong enough are discarded. Remaining edges are thinned, binarized and finally vectorized. The amount of noise is significantly reduced, but not removed completely by this processing.

2.2.2 Aligning video frames to synthetic panorama

Vectorized edges from the polygonal model and a video frame are sampled with certain frequency. A special modification of the 2D cross-correlation is then used to compute a set of candidate solutions of the alignment. The cross-correlation can be computed fast by a sophisticated algorithm operating in the frequency spectrum.

To make this possible, the *FFT (Fast Fourier Transform)* is applied to convert edges to the desired frequency spectrum. Although the computation of cross-correlation is a fast process, it requires a lot of memory. The memory consumption and the quality of candidate solutions generally depend on the frequency of edge sampling. Higher frequencies increase the probability of the presence of a correct solution in the set of candidate alignments. In the opposite direction, memory demands as well as the processing time grow when increasing the frequency.

Every alignment candidate solution is then evaluated by a special metric. It is implemented using OpenGL which significantly improves performance of the computation by using hardware acceleration provided by the GPU. The candidate solution with the highest value of the metric is chosen as the final solution of the alignment. The final alignment is illustrated in **Figure 5b**.

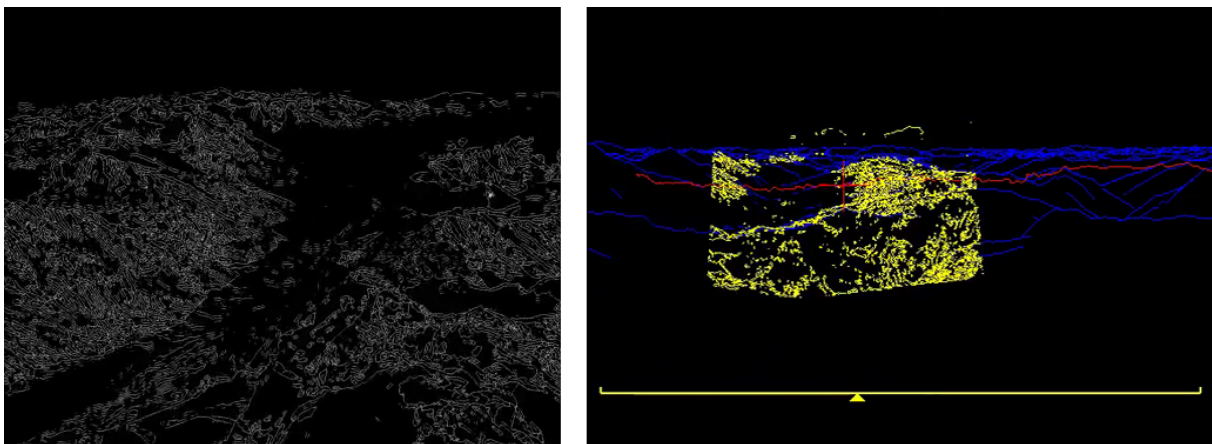


Figure 5: Thinned edges from a video frame in the left (a) and matching edges in the right (b).

The above-mentioned assumption of knowing the place from where the video has been captured (so the method could be also used for purposes of visual geo-location) can be removed at the expense of increased computation power. A simplified solution expects that the video was captured on a prominent summit and uses the 2D cross-correlation and a special edge metric not only to align the frame into the panorama, but also to choose the right mountain peak. Despite the simplification, this requires a significant amount of computation power. The second part of the large-scale video processing experiments corresponds to such computation in modern data centres.

3 Experimental data sets for video semantic analysis

This section describes two data sets that will be used together with experimental applications for video semantic analysis and will be directly involved in experiments with the executable Taverna workflows. The first data set consists of stereo images of an urban environment; the second one contains pictures taken in Alps and a large number of synthetic panorama pictures rendered from a polygonal model of Alps.

3.1 The Málaga Stereo and Laser Urban Data Set

The urban data set was captured by the University of Málaga and by the University of Almería in 2013. It is publicly accessible; a detailed description, links to related research papers and download links can be found at <http://www.mrpt.org/MalagaUrbanDataset>.

The data set was gathered in urban scenarios with a car equipped with one stereo camera and several laser scanners. Images and laser scans are geo-referenced. From the 3D reconstruction point of view, the most interesting part of the data set is formed by images grabbed using the stereo camera. Each image is 1024 pixels wide and 768 pixels high. The stereo-video was captured at a high rate of 20 video frames per second during a 36.8 km long drive through the town of Málaga. The image part of the data set contains 227 000 pictures stored in the JPEG file format. This part is 45 GB large in the ZIP compressed form; it takes around 60 GB after decompression. Examples of the data set content are shown in Figures 6 and 7.

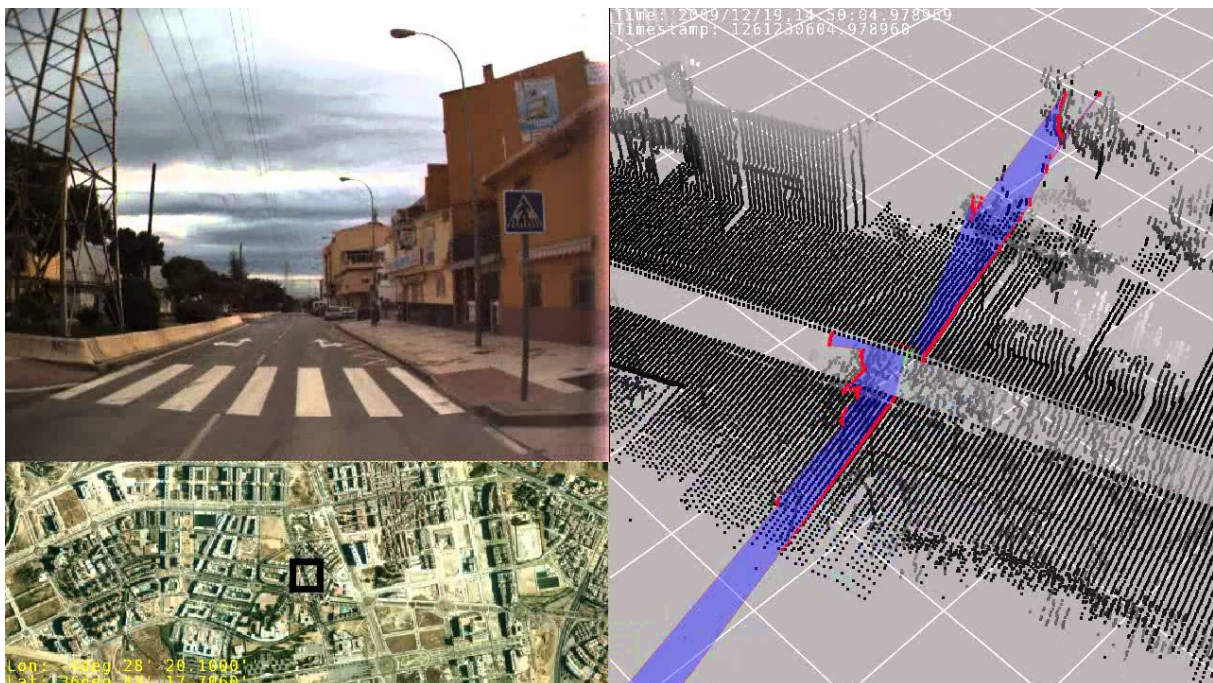


Figure 6: Relation between stereo images, laser scans and a position of the car.

Figure 6 demonstrates the correspondence between video frames recorded by the stereo camera and captured laser scans. Corresponding images and laser scans are geo-referenced, as it can be seen in the map in the bottom part of the image. The truth position of the car is marked by the black square in the map. **Figure 7** shows two actual frames from the stereo video taken in the same moment by the left and the right lens of the stereo camera.



Figure 7: A stereo image from the Málaga dataset

3.2 The Alp Mountains Image Data Set

The Alp Mountains image data set was gathered by the Brno University of Technology in 2013 and 2014. It comprises two parts; the first one contains real images taken from tops of significant summits of Alp Mountains. These images are equipped with geo-reference information. The roll, the yaw and the pitch of the camera, i.e. the ground truth alignment of the real photo to the panorama, are known as well.

The second part of the data set contains a big amount of rendered images of panorama views from prominent summits of the entire Alp Mountains. The prominent summits were chosen accordingly to the geo-relief of mountains by using an automatic approach based on geo-morphologic methods.

Each picture in the data set, a real photo or a rendered panorama, is stored in two formats. The first one is the PNG image format suitable for visualising results of the computed alignment, as illustrated in **Figure 8**, where the real photo¹ taken from the Finsteraarhorn summit is aligned to a rendered panorama.

The second format contains vectorized edges derived from the 3D polygonal model using raycasting, or edges derived from real images using the COMPASS edge detector. While the PNG files are usable mainly for visualisation of alignment results, files containing vectorized edges are used for computation of the alignment. **Figure 9** shows results of the automatic detection of prominent

¹ Notice that the photo is warped in order to normalize the camera field-of-view, as described in **Section 2.2.2**.

summits. Summits were detected using the *LandSerf* application. Detected peaks are higher than 250 metres and they are surrounded by a minimal drop of 250 metres.



Figure 8: A photo from the top of the Finsteraarhorn Mountain aligned to a rendered panorama

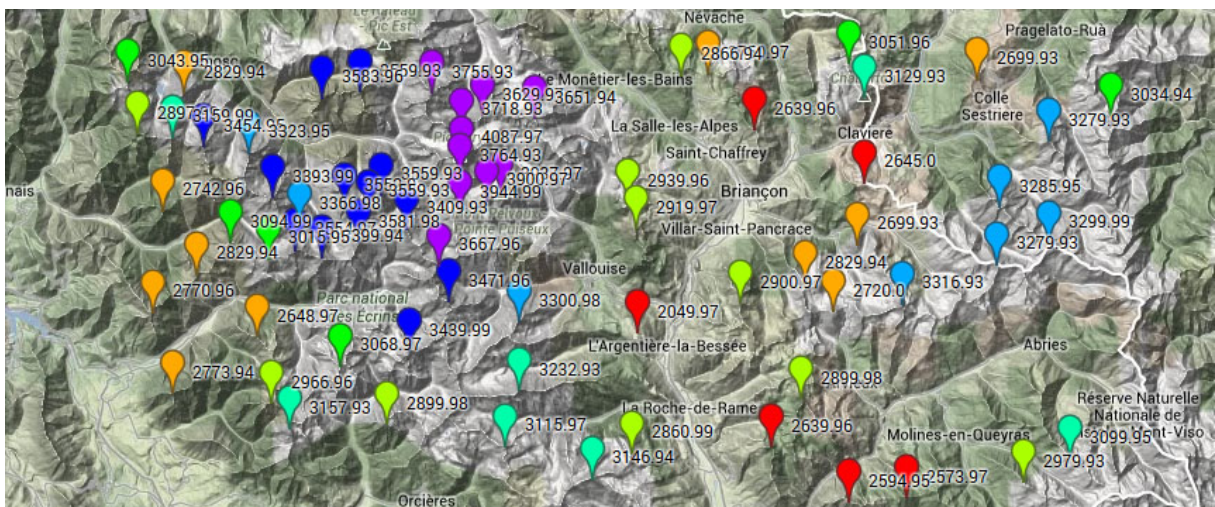


Figure 9: Visualisation of automatically detected summits in Dauphiné Alps, National Park of Écrins

4 Identified preservation risks of video processing in remote data centres

This section identifies preservation risks and actual deployment and integration of preservation scenarios that are related to large-scale video processing experiments in external data centres. The risks were divided into four main categories:

1. **Risks arising from external HPC infrastructure**
2. **Potentiality of input or output files corruption**
3. **Loss of experiment details**
4. **Loss of links between source data, HPC influence and results**

Each category is described in detail in following subsections.

4.1 Data centre environment threats

This category consists of the largest amount of identified risks that could directly affect the quality of video analysis results. A major part of them is closely related to varying performance of external data centres and their individual clusters. The different performance of HPCs may lead to different results of iterative video processing. This is true in both previously defined cases – in the 3D reconstruction of recorded scene and in the annotation of large-scale natural environments.

As it was stated in Section 2.1.2, bundle adjustment – the last and the key stage of the 3D reconstruction – is based on an optimization algorithm. Optimization algorithms generally depend on environment performance in general. When the wall time of computation is limited, the HPC with higher performance can process a higher number of optimization iterations compared to the processing with limited resources. This will lead to a more precise reconstruction in the case of a more powerful data centre, as the re-projection error is smaller and the probability of finding optimum grows with the rising number of optimization steps.

Similarly, one of the most important steps in annotating video frames consists in aligning them to the synthetically rendered panorama. This is done by computing the 2D cross-correlation between a video frame and a panorama, both represented as a set of sampled edges. When the sampling frequency of derived or detected edges increases, the probability of finding an optimal alignment grows as well. On the other hand, the amount of system memory needed grows with sampling frequency too. Thus the amount of memory available in the processing cluster computers can directly affect the probability of finding the optimal alignment.

Data centre risks can be further divided into two groups:

- **Risks related to static properties of the HPC environment,**
- **Threats related to dynamic properties of external data centres**

Static properties of the HPC environment are defined as properties, which cannot change in the time when an experimental application is running. For example, the general hardware setting of the infrastructure, such as the count of computation nodes, the presence of GPU accelerators, versions of available hardware drivers belong to static environment properties.

Dynamic properties are those that can change in the time of the experimental application execution. A list of processes running simultaneously with the experimental application, the amount of memory and the processor time consumed by other applications, and the amount of total available memory in each node belong to this category. When the application processing large data takes a significant amount of time, the variability of these parameters can significantly influence results of specific phases of the long-term processes.

4.1.1 Threats related to static properties

Static properties can characterize hardware infrastructure of external data centres as well as specific parameters of the HPC software employed.

Hardware properties

The hardware characteristics that should be preserved together with results of video processing experiments are given in **Table 1**.

Characteristics	Definition of the characteristics
Amount of memory	Amount of memory installed on each computational node. This characteristic should be preserved for every node that was used by the experimental application.
Processor type	Identification of processor(s) at each node of the remote computer cluster.
Local storage size	Size of the local scratch at each node of the HPC, if the local store is available to the application.
Available hardware accelerators	Graphic cards or hardware accelerators based on the GPGPU concept (General Purpose GPU) available in the used computational nodes.
Network characteristics	The topology and other parameters of the network that interconnects nodes of the external data centre.

Table 1: Hardware characteristics identified for preservation

Software properties

Software characteristics focus on the actual software of the external data centre, such as versions of installed shared libraries. Selected software characteristics that have shown to be relevant for the video processing applications are summarized in **Table 2**.

Characteristics	Definition of the characteristics
Versions of static libraries	Version numbers of static libraries that the application requires in order to get successfully compiled.
Versions of dynamic libraries	Version numbers of dynamic libraries that have to be present in the environment to allow the application to get compiled or that are needed by the compiled application at the run time.
Version of the C library	A version number of the GLIBC library which was used when the application or dynamic libraries were compiled.
Special hardware drivers	Versions of drivers that are available in the cluster environment at nodes equipped with dedicated hardware, such as GPGPU accelerators for general computations or graphic cards for image rendering.

Table 2: Properties of the HPC software

Versions of static libraries are important in the case when the experimental application is preserved in the form of source codes and compiled directly in the data centre. Usage of a required library in a version that significantly differs from the version with which the application was originally designed may cause unexpected behaviour of the application or may make it impossible to compile the application at all.

Versions of dynamic libraries should be preserved in both cases when the experimental application is compiled directly in external data centre or when it is distributed in its binary form. In the first case, the same problems as with static libraries mentioned previously can occur. Different versions of shared libraries may cause problems when the application is running even in the case when the application is distributed in its binary form.

Mentioned issue can be illustrated by the case of the PNG library. When the experimental application is compiled and run with different versions of the PNG library, the application generates warning messages about the different library versions. An example of such a message is shown in **Figure 10**.

```
libpng warning: Application built with libpng-1.6.7 but running with 1.5.15
```

Figure 10: Libpng warning message about wrong version of currently loaded library

Different versions can cause problems with storing image files, when a blank image is stored instead of an image with the actual content. Issues with the PNG library can be related to the experimental application providing annotations of the video content, which works with the data set containing PNG images and which visualizes results of the alignment in the same image format.

Problems may also occur when the experimental application, or some of the dynamic libraries that are required by the application, were compiled previously in another environment that differs in the version of the C library. In the worst case, such problems may completely stop the experimental process. To avoid this problem, it is possible to preserve the version of the original C library and to distribute the application with the original C library that was used during the compilation. Yet, issues with user privileges to execute the application with the custom C library may occur.

For example, when trying to run a Qt enabled application with libraries compiled with the GLIBC version different from the one that is currently available, error messages such as illustrated in **Figure 11** appear. The application is prevented from a successful execution.

```
./pano: /lib/libc.so.6: version `GLIBC_2.9' not found (required by /home/root/libQtGui.so.4.8.3)
./pano: /lib/libc.so.6: version `GLIBC_2.9' not found (required by /home/root/libQtOpenGL.so.4.8.3)
./pano: /lib/libc.so.6: version `GLIBC_2.9' not found (required by /home/root/libQtCore.so.4.8.3)
./pano: /lib/libc.so.6: version `GLIBC_2.9' not found (required by /home/root/libQtXml.so.4.8.3)
```

Figure 4: Error messages generated by Qt libraries about a different version of the GLIBC library

It can be very useful to preserve versions of software drivers for specialized hardware in the cluster. This is true even in the case when the application runs in different environments with the same hardware characteristics. It is caused by the fact that functionality of the available hardware, particularly GPUs, depends on the currently used drivers.

The OpenGL standard defines an API for rendering 3D graphics. This standard is commonly implemented using hardware acceleration that is provided by the present graphics card. The version of a currently available OpenGL API depends on the version of the installed graphics driver; therefore it can differ even in the case when the hardware equipment is the same.

There are also purely software drivers for OpenGL which do not require any GPU at all, but, without the hardware acceleration, they render 3D graphics with significantly poorer performance. Mesa 3D Graphics Library, supporting the OpenGL 3 standard, is an example of this approach.

Lower performance of rendering can have an impact on the quality of experimental results, e.g. in video frame annotations. As mentioned above, the special metric for evaluating candidate solutions is implemented using OpenGL shaders. When the time for computation and the overall OpenGL performance are limited, the frequency of edges sampling must be lowered. Low sampling frequency then decreases the probability of correct alignment of the photo to the panorama picture.

4.1.2 Risks related to dynamic characteristics

Dynamic parameters of the environment are strongly related to fluctuation of the data centre performance. These characteristics have to be preserved repeatedly on each node of the cluster. This is obvious from the definition of cluster dynamic properties – their main feature is that they are varying in the time of experiment computation. The list of dynamic characteristics is captured in **Table 3**.

Characteristics	Definition of the characteristics
Memory consumption by other running processes	This characteristic describes the amount of memory consumed on each computational node by other processes running simultaneously to the experimental application.
Memory consumption of the experimental application	This attribute describes the amount of memory allowed to be consumed by the experimental application on each node.
Processor time consumption by other running processes	This stands for the processor time consumed by processes running simultaneously to the experimental application.
Processor time consumption by the experimental application	The amount of processor time allowed to be consumed on each node by the experimental application in each phase.

Table 3: Dynamic characteristics of data centre environments

Memory and processor time consumption can temporarily decrease performance of the external data centre. This can affect the quality of results of video processing in the same way as described in the previous sections. These characteristics need to be preserved, as they appear and affect the cluster performance randomly; they cannot be estimated from any other characteristics of the data centre environment.

Preservation of resource consumption caused by the experimental application itself can be useful for reproducibility and profiling of the experimental platform.

4.2 Input and output of experimental data

To guarantee reproducibility of experiments, it is necessary to ensure that all files that are related to the experiment will be preserved and that they will not be corrupted. Input files have to be saved so the experiment can be run again later. Results generated as an outcome of the experiment have to be preserved for the purpose of comparison with results of repeated experiments.

Files can be corrupted in several ways. As the experiments are supposed to be processed in external data centres, the most serious risk of file corruption is present in the moment when the input files are being transferred from the institution that owns the data to the cluster storage facilities and when the output files are transferred from the data centre back to the research institution. When transferring through the network, the content of files can be changed or truncated due to transfer failures. Another way how the files can be corrupted is their manual change, which can be done by mistake, for example, by editing a wrong file.

Basic characteristics related to preserving experimental data are stated in **Table 4**.

Characteristics	Definition of the characteristics
List of all experiment-related files	List containing all files that are related to the experiment including preserved details, such as file sizes or times of the last modification.
Current file size and hash	The size and a hash (e.g., MD5) of the current file.
Time of modification	The last time when the content of the current file was modified.

Table 4: Characteristics related to input and output data preservation.

By comparing the file size and the hash preserved in the list of experiment-related files with the same characteristics of the actual file, one decide if the file was truncated during the file transfer through the network or modified in another way.

Preserved time of the last modification of the current file can be compared with the actual modification time to see if the file wasn't changed after the experiment has finished.

4.3 Non-technical attributes of experiments

The development process of video semantic analysis methods involves performing a high number of experiments. This is caused by the fact that these methods usually rely on many different parameters whose optimal values are set empirically.

However, the high number of preserved experiments may lead to a confusion that can potentially make preserved experiments useless. Therefore, it is beneficial to preserve non-technical details about the experiments, such as the purpose of each individual experiment etc. A list of suggested details is summarized in **Table 7**.

Attribute	Definition
Responsible persons and their contacts	Persons who have planned and executed the experiment, their e-mail and other contacts.
Datetime	When the preserved experiment was performed.
Data set	Identification of the data set which was used in the experiment. This identification should also contain a link to the data set and other details.
Description	Description of the experiment. The description should identify the main purpose of the experiment, what was to be verified or proved by the experiment.

Table 5: Non-technical aspects of preserved experiments.

Preserving these details helps to minimize the risk of the loss of experiment meaning. Adequate details should be filled in to guarantee understanding in a long term.

4.4 Links between source data, cluster environment and results

The last group of identified preservation threats deals with interlinking previously identified characteristics together. This aims at preserving relations between the cluster infrastructure and performance, current work load, input files and results of large-scale experiments. Key relations that contribute to the success of the preservation scenario are given in **Table 8**

Relation	Description
Input and output files	<p>This relation describes which input files were used for creating each individual output file, or, in the opposite way, which output files were created with contribution of a particular input file.</p> <p>These two cases typically correspond to one-to-one or one-to-many relationships.</p>
Dynamic characteristics and output files	<p>This relation links together resource consumption caused by the experimental application and generated output files of the experiment.</p>
Static characteristics and output files	<p>This relation captures links between created output files and hardware and software characteristics of the cluster.</p> <p>It can describe which output file was created on which computation node, which hardware accelerators were used during its creation, which drivers were used, etc.</p>
Static and dynamic characteristics	<p>Relationships between hardware and software of the external data centre and resource consumption caused by both the experimental application and other simultaneously executed processes.</p>

Table 6: Relations between characteristics appropriate for interlinking.

5 Tools for minimizing impact of identified preservation risks

This section describes the second part of the proposed preservation methodology focusing on large-scale video processing experiments. It discusses the development of a software solution which minimizes the risks and threats described in the previous section by preserving selected characteristics of the external data centre environment and details about execution of the experimental application. These characteristics will be preserved together with results as experimental metadata and interlinks.

Requirements on such preservation software are described first. Then the architecture of the software solution is discussed with respect to previously stated requirements and in relation to the SCAPE platform. Software functionality and implementation recommendations are given in the last part.

5.1 Preservation tool requirements

Three major requirements have been identified as key enabling factors for software general applicability and acceptability in the whole range of potential areas of large-scale experiments involving data centres:

- **Solution generality**
- **Ability to process large-scale data**
- **Comfortable user interface**

5.1.1 Solution generality

The primary requirement on the software is its usability in the greatest possible range of experimental applications. This can be difficult to fulfil, as every experimental application may have very different design and there is no general assumption about the experimental application architecture.

Researchers must be able to specify only selected preservation characteristics that are relevant to their experiments and to use dedicated tools for further analysis of previously stored experimental metadata.

5.1.2 Ability to process large data

The proposed solution must not significantly influence performance of experimental applications and it has to be able to process stored experimental metadata in an efficient way.

Experimental applications can produce huge amounts of experimental results and metadata even if the original data set is not extremely large. This is particularly true in the cases when experiments are performed to empirically find optimal values of certain parameters. The computations are then executed many times while varying the value of the desired parameter.

5.1.3 Comfortable user interface

The software must be easy to use for people working in the field of the semantic video analysis. It does not necessarily mean the solution has to be equipped with an elaborate graphical user interface; the strongest focus should lie on practicality of the user interface.

5.2 Software architecture

The software solution was designed to meet the requirements listed above. With respect to the first one, it is assumed that most of experimental applications related to large-scale video processing are implemented using the C or C++ languages and that they are parallelized using the MPI mechanism (Message Passing Interface). This assumption allows the software solution to be used with a large amount of graphics applications, as most of image processing libraries are written in the C/C++ language – for example the OpenCV library by Intel, libraries related to the OpenGL standard or the Qt Toolkit by Digia.

The software is referred to as the EXPRES (Experiment Preservation) toolkit. It consists of two key components – a C++ library and a set of Apache Hadoop tools (in Java). The C++ library is to be used as a part of experimental applications. The set of the Hadoop tools helps to analyse stored experimental metadata. The design of the solution is shown in **Figure 12**.



Figure 5: EXPRES toolkit architecture.

It is supposed that the experimental application can take advantage of a GPU cluster. The application will be compiled together with the toolkit library that enables storing experiment metadata in the HDFS storage. Metadata is stored as a set of XML files.

The second part of the toolkit, tools for analysing produced metadata, can run on a different cluster, which will support the Apache Hadoop, but the cluster needs to have an access to the stored metadata, for example by calling the same HDFS.

5.2.1 C++ library

The C++ library is intended for instrumentation of the source code of experimental applications; the experimental applications must be compiled together with this library to be able to store the experimental metadata. As said in the requirement section, the library has to be able to save only the metadata which are relevant to the current experiment. Therefore, the library provides a set of functions corresponding to saving a particular set of characteristics and it is completely up to the

user, which characteristics and interlinks will be preserved, when they will be preserved or, in the case of dynamic characteristic, how often they will be stored.

5.2.2 Hadoop tools

Each Hadoop tool is dedicated to a specific task analysing the stored experiment metadata. There are tools that deal with interlinks, other tools for checking consistency of preserved metadata and files related to the experiment, etc.

The Hadoop tools are implemented as command line applications, so they can be easily used from Taverna workflows or from shell scripts. As the metadata from different nodes are stored in separate XML files, they can be effectively processed by these tools in the map-reduce manner.

5.3 Toolkit functionality

Functions of the toolkit can be divided into following groups:

- **Functions for preserving data centre environment details**
- **Functions for creating and analysing interlinks**
- **Functions and tools for preserving input and output files**
- **Functions for checking metadata consistency**

Functions for preserving cluster infrastructure and performance details make it possible to store and analyse static and dynamic characteristics which were defined in the previous section. For example, information about the amount of actually consumed memory in certain moments can be regularly stored and later analysed.

Functions dealing with interlinks allow users to see a global picture of inter-related characteristics that have been stored. For example, it can be preserved which input files were used to create certain results, how long did it take to create a result and what drivers were used during the computation.

Functions for checking related input and output files and metadata consistency are able to check whether any input or output file is missing, whether they were not modified after the experiment or whether the structure of stored metadata is valid.

6 Taverna workflows related to semantic video analysis

This section describes executable Taverna workflows that are related to video contextual analysis as it was described in previous sections. The workflows are designed to run in external data centre. A basic template workflow is shown in **Figure 13**.

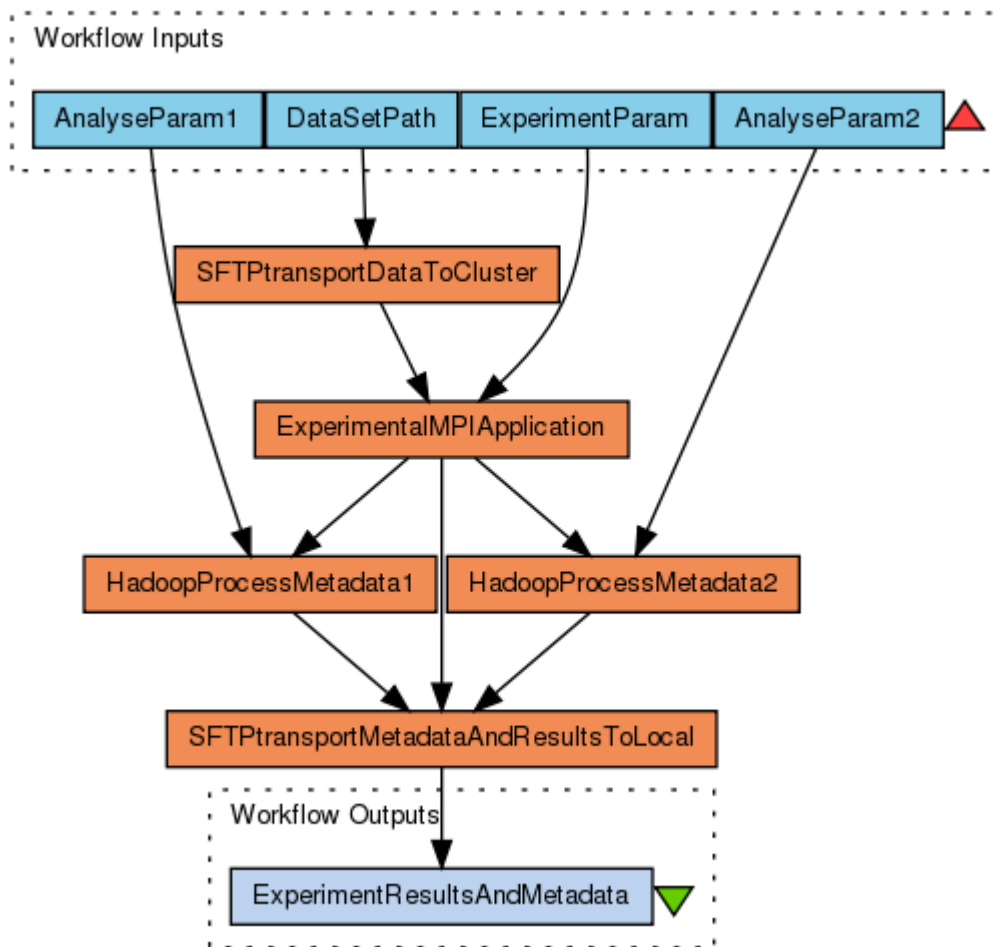


Figure 6: General workflow using the experiment preservation toolkit.

This workflow describes how the experiment preservation toolkit is typically used in remote data centres. The first step copies the data set from the storage facilities of the institution that owns the data to HDFS of the target HPC. This transfer can be done by using the SFTP protocol. In the next step, the experimental application performs certain computation and stores the metadata to the HDFS in the form of XML files. After the computation is finished, tools analysing the stored metadata can be executed. When the analysis of the metadata is finished, results of the experiment, its metadata and results of its analysis are transferred back to the facilities of institution that performed

the experiment. This workflow can be found online here: <http://www.myexperiment.org/workflows/4268.html>.

6.1 3D reconstruction workflow

This workflow aims at finding an optimal distribution of the reconstruction algorithm across the cluster of computation nodes. The experimental application will be executed many times with different numbers of nodes and processes running on each computation node. Time necessary for each step of the algorithm – the extraction of SIFT features, matching these features between corresponding stereo images and tracking them in neighbour images – will be preserved for each input picture in the XML metadata and then processed by Hadoop-based tools. Results of the metadata analysis will characterize the average computation time of each mentioned processing stage. The actual workflow is shown in Figure 14. It is available here: <http://www.myexperiment.org/workflows/4277.html>.

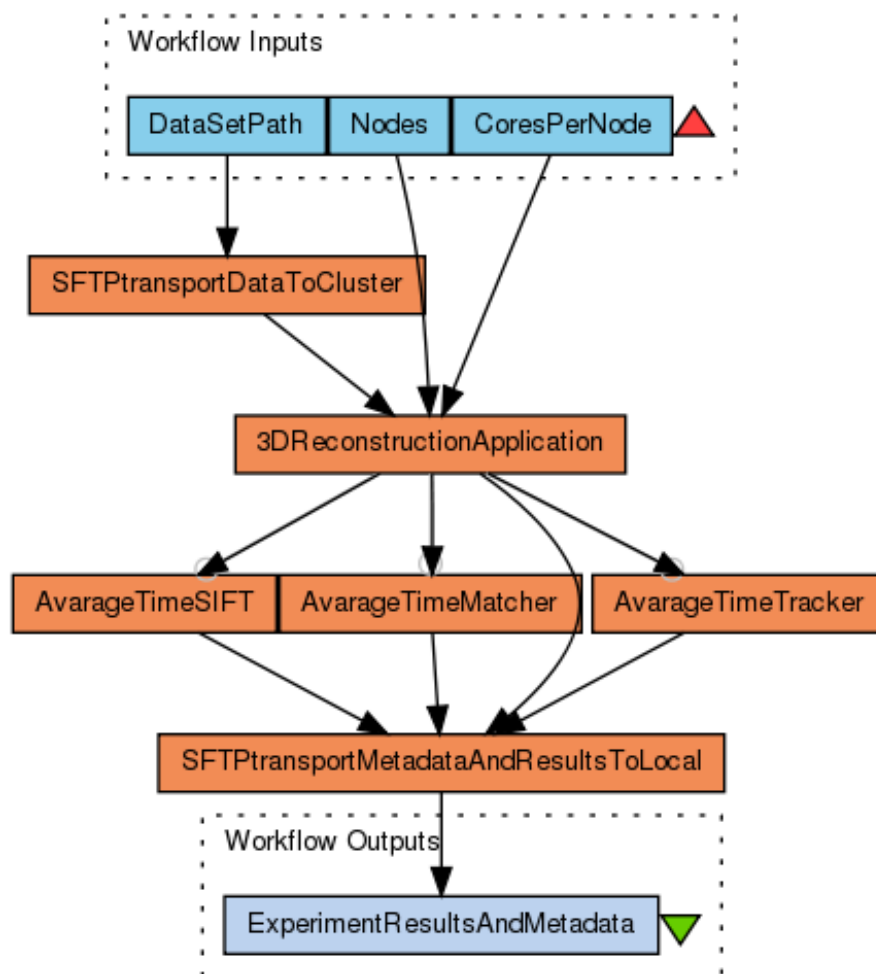


Figure 7: 3D reconstruction workflow.

6.2 Workflow of annotations of natural environments

This workflow corresponds to an experiment exploring relations among the frequency of edge sampling, the amount of memory and the time needed to compute the alignment with a certain sampling frequency and the corresponding alignment quality. This experiment produces a significant amount of metadata. Execution time needed for computing cross-correlations and the special metrics will be saved for each alignment, as well as the amount of memory consumed by the computation. This information will be preserved in the XML metadata files. The frequency of sampling will be varied and the experiments will be performed for each sampling value. The average value of memory and time consumption and the percentage of successfully aligned cases will be computed from metadata for each value of sampling frequency using the Hadoop tools. The workflow is illustrated on **Figure 15**. It is available online here: <http://www.myexperiment.org/workflows/4278.html>.

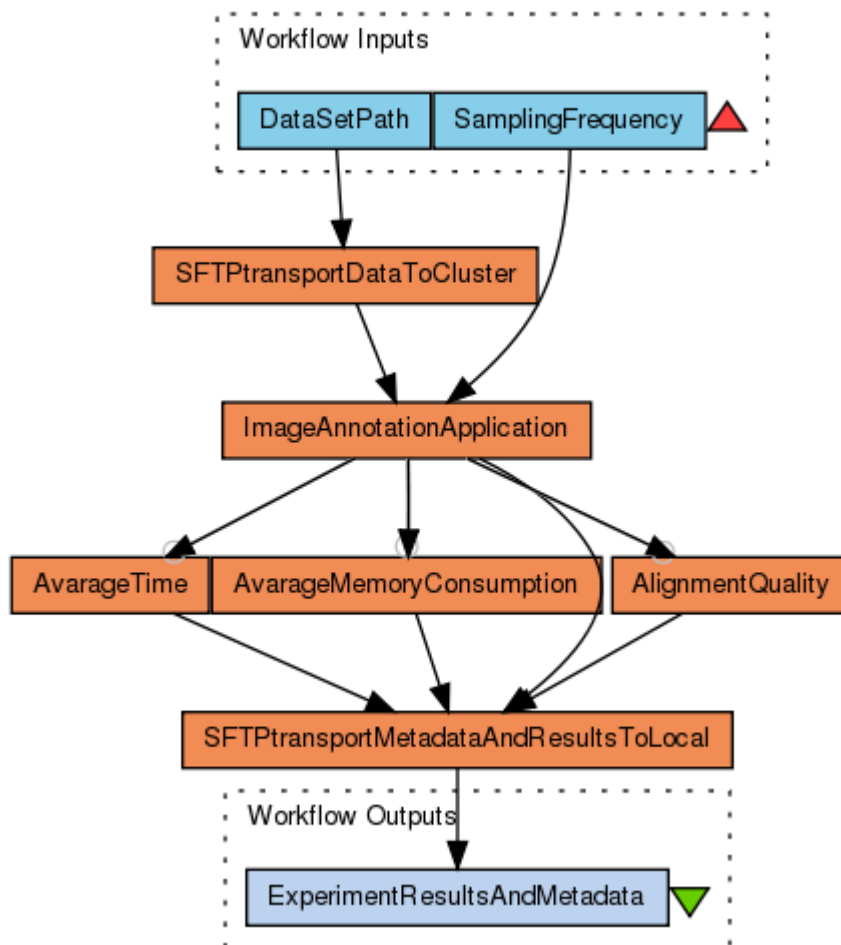


Figure 85: Annotation of natural environments workflow.

7 Conclusions

This deliverable lays the foundations of preservation actions relevant for large-scale experiments running in data centres. Two particular instantiations of the general video semantic analysis schema – the 3D model reconstruction and the annotation of videos captured in natural environments serve as representatives of typical use-cases for the complex preservation task. It was also shown that the two datasets briefly introduced in the text enable evaluating the devised preservation procedures on real data.

The main part of the deliverable focuses on the methodology of preserving results and circumstances of experiments related to the mentioned video analysis. Potential risks and threats arising from the usage of external data centre facilities were identified and discussed in detail. A new software solution which deals with these risks and provides easy-to-use preservation functions for this purpose was also proposed. It benefits from the SCAPE platform and adds components specific for the data centre setting. Taverna workflows and user scenarios were identified and described in order to demonstrate the functionality of proposed preservation solution.

The proposed methodology will be further validated in the process of full deployment and integration of the preservation scenarios within the data centres. Results will be reported by the end of the project in the final deliverable D23.2.