

Web-based 3D reconstruction service

Maarten Vergauwen · Luc Van Gool

Received: 14 December 2005 / Accepted: 15 March 2006 / Published online: 25 May 2006
© Springer-Verlag 2006

Abstract The use of 3D information in the field of cultural heritage is increasing year by year. From this field comes a large demand for cheaper and more flexible ways of 3D reconstruction. This paper describes a web-based 3D reconstruction service, developed to relieve those needs of the cultural heritage field. This service consists of a pipeline that starts with the user uploading images of an object or scene(s) he wants to reconstruct in 3D. The automatic reconstruction process, running on a server connected to a cluster of computers, computes the camera calibration, as well as dense depth (or range-) maps for the images. This result can be downloaded from an ftp server and visualized with a specific tool running on the user's PC.

Keywords 3D reconstruction · Webservice · Cultural Heritage

1 Introduction

The possibility to acquire accurate and realistic 3D representations of objects and scenes is highly valued by cultural heritage professionals as it offers possibilities to better document, study, preserve, restore and present sites and artefacts. The European Network of Excellence EPOCH is a network of about a hundred European

cultural institutions joining their efforts to improve the quality and effectiveness of the use of Information and Communication Technology for Cultural Heritage. The development of versatile and cost-effective 3D acquisition technologies is part of its research agenda.

Two EPOCH partners, the ESAT-PSI lab of K.U.Leuven (Belgium) and the Visual Computing Lab of CNR-ISTI (Italy) have been active for a long time in the field of 3D reconstruction, albeit each in their own specific sub-area, passive reconstruction from images, mesh decimation and integration, respectively. They decided to combine their strengths and set up a low-cost 3D reconstruction pipeline to be used in the cultural heritage field. The idea is that only a digital photo-camera and an Internet connection are necessary for a user to reconstruct scenes in 3D.

Figure 1 shows a schematic overview of the complete pipeline. It consists of three services that are performed on servers, accessible via the Internet and fed by data from the user and two local applications. Images of the object of a scene to be reconstructed are first checked for quality, then uploaded to the first web service which computes dense range maps. A second web service is capable of merging these range maps. The user can also align range maps from different image sequences locally and merge them. The third web service is in charge of mesh creation and simplification.

This paper describes the first web service of Fig. 1: the computation of dense range (or depth-) maps from images uploaded by the user. The outline of the paper is as follows. Section 2 gives an overview of the client-server system that allows users to upload images to a server which computes 3D information from these images. Section 3 describes the automatic reconstruction pipeline running on the server. Sections 4, 5, 6 and 7

M. Vergauwen (✉)
K.U.Leuven ESAT-PSI
Kasteelpark Arenberg 10, 3001 Leuven, Belgium
e-mail: maarten.vergauwen@esat.kuleuven.be

L. Van Gool
ETH Zürich D-ITET-BIWI
Gloriastrasse 35, CH-8092 Zürich, Switzerland
e-mail: vangool@vision.ee.ethz.ch

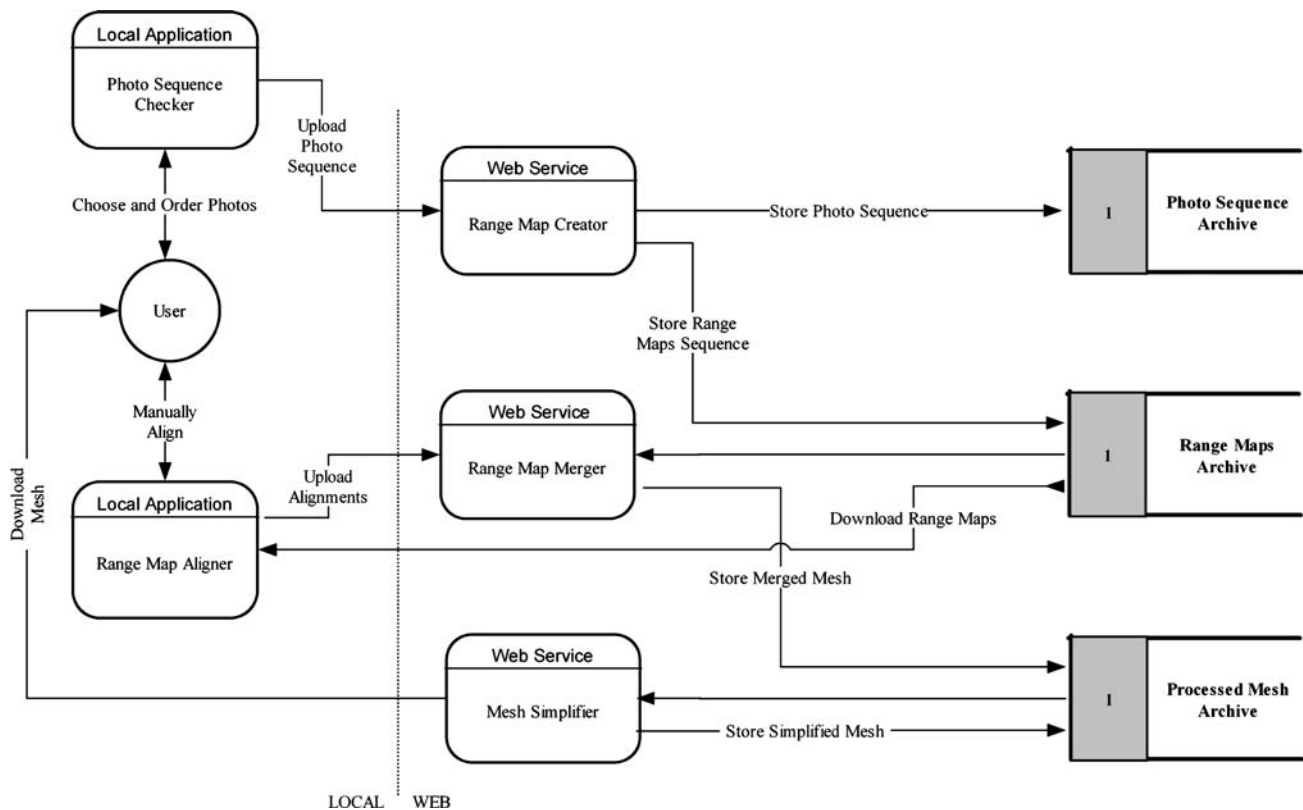


Fig. 1 The main architecture of the combined 3D system: a local application prepares and checks the photos collected by a user and uploads them to a web service; the web service creates and

stores range maps; these range maps are then aligned by the user and finally merged, simplified and exported to various formats

give more detailed information on several core aspects of this pipeline. Section 8 shows some results obtained from uploaded images and Sect. 9 concludes the paper.

2 System overview

Figure 2 shows a schematic overview of the client–server setup of the 3D webservice. The client- (or user-) part is located at the top, the server side at the bottom. On his PC, the user can run two programs, the upload tool and the modelviewer tool, indicated with **A** and **B**. These tools are explained in more detail in Sects. 2.1 and 2.2.

In the upload tool, images can be imported that were taken with a digital camera. Once authenticated, the user can transfer these images (**C**) over the Internet to the EPOCH server at ESAT. There, a fully automatic parallel process is launched which computes dense 3D information from the uploaded images. The parallel processes are run on a cluster of Linux PC's (**D**). When the server has finished processing, the user is notified by email and the results can be downloaded from the ESAT server by FTP. These results consist of dense depth maps

for every image and the corresponding camera parameters (**E**).

The modelviewer tool allows the user to inspect the results. Every reconstructed depth map in the image set can be shown in 3D, unwanted areas can be masked out and the meshes can be saved in a variety of formats.

2.1 Upload tool

The first tool a user of the 3D webservice encounters is the upload tool. This is the program that uploads images to the server via the Internet. Figure 3 shows a screenshot of the GUI of this tool. First, the user selects the images which he wants to upload. Thumbnails of these images are shown on the left and a large version of the image is shown in the main panel when a thumbnail is selected. Images can be removed from the list or extra images can be added. On the bottom left, information about the image such as the number of pixels, the size on disk, and the format are shown.

When the user is satisfied with the selected images, he can upload them to the reconstruction server. In order to do so, the user first needs to authenticate himself with his username and password. The upload is started and a

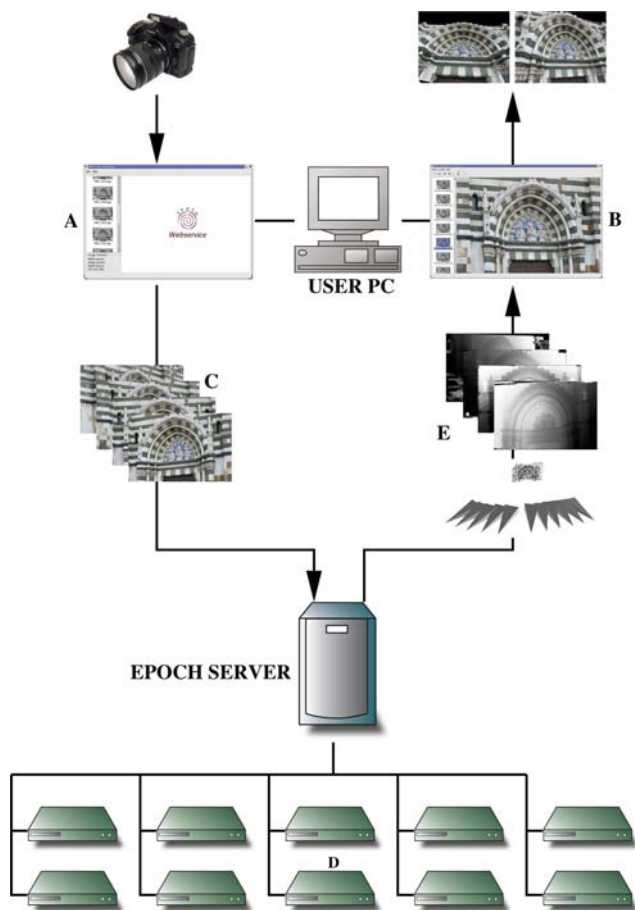


Fig. 2 Schematic overview of the client-server setup. Images (C) are uploaded from the upload tool (A) on the user side to the server. There they are processed on a PC cluster (D). The results (E) can be downloaded via ftp and visualized on the user PC with the modelviewer tool (B)



Fig. 3 Upload tool. Thumbnails of the images to be uploaded are shown as well as some statistics. When satisfied, the user uploads the images to the reconstruction server

progress dialog shows the speed of the upload. In order to limit the upload- and processing time, the user can decide to downscale the images by a certain percentage.

2.2 Modelviewer tool

Upon completion of the 3D reconstruction, the user is notified that the results are ready. They are stored in a zip file on the ESAT ftp server in a directory with a random name whose parent is not listable. This makes sure the user's reconstructions are save from prying eyes. The zip file contains the original images, the calibration of the cameras, the dense depth maps and quality maps for every image. The results can be inspected by means of the modelviewer tool, a screenshot of which is shown at the top of Fig. 4. The layout of the modelviewer tool purposely resembles that of the upload tool.

A thumbnail is shown for every image in the set. If selected, a large version of the image is shown in the main window. A triangulated 3D model can be created for the current image, using the depth map and camera of this image. Every pixel of the depth map can be put in 3D and a triangulation in the image creates a 3D mesh, using the texture of the current image.

Every depth map has its corresponding quality map, indicating the quality or certainty of the 3D coordinates of every pixel. The user can select a quality threshold. High thresholds yield models with fewer but more certain points. Lower thresholds yield more complete models including areas with a lower reconstruction quality. If required, certain areas of the image can be masked

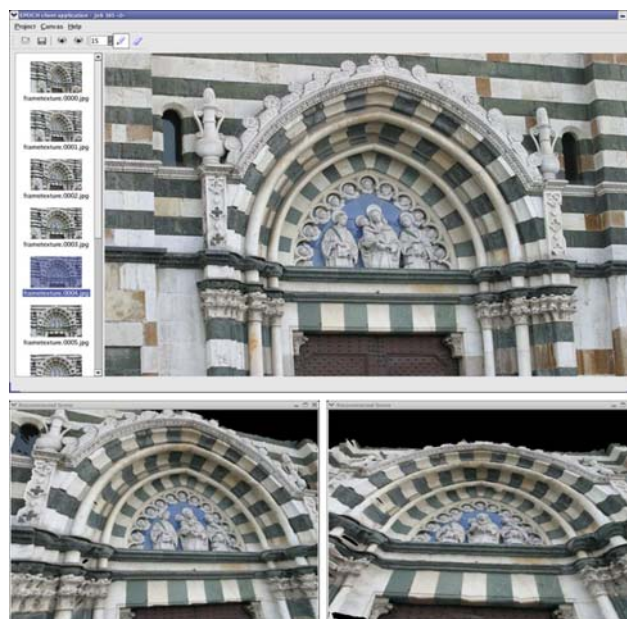


Fig. 4 Modelviewer tool. The layout of the tool (top) deliberately resembles that of the upload tool, with thumbnails on the left and a large image in the main panel. The tool allows the user to create textured wireframe representations of the depth maps. The bottom two pictures show different views of such a 3D representation

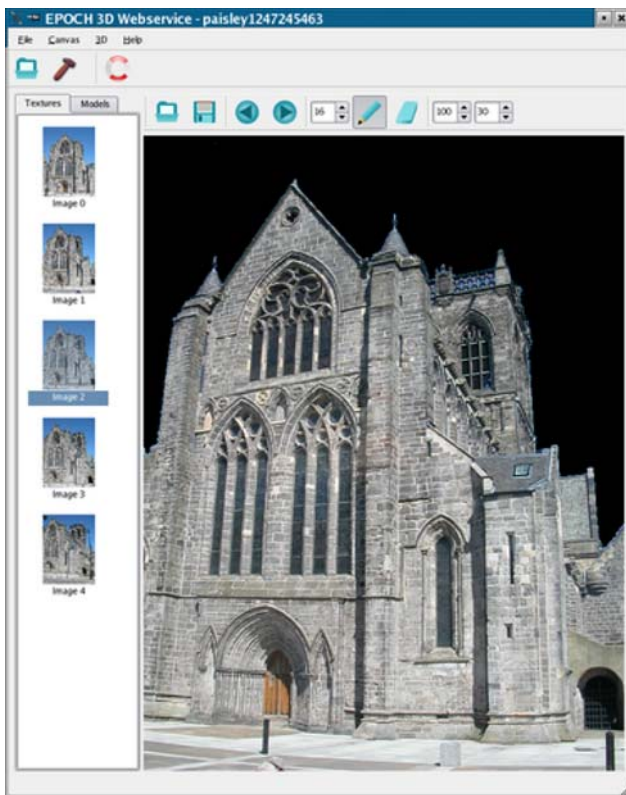


Fig. 5 View of the Paisley Abbey in Scotland. The operator has automatically masked the blue sky with a simple click inside the region

by drawing with a black brush in the image. Areas that are masked will not be part of the 3D model. Homogeneous regions like the sky yield bad reconstruction results and can automatically be discarded by the user. A simple click inside such a region starts a mask-growing algorithm that covers the homogeneous region, as can be seen in Fig. 5, where the sky above Paisley Abbey in Scotland is automatically covered by a mask. The resulting 3D model is displayed in an interactive 3D widget. Two views of a 3D mesh of the Cattedrale di Santo Stefano in Prato are shown at the bottom part of Fig. 4. The model can be exported in different 3D file formats such as VRML2, Open Inventor, OBJ or OpenSG's native file format. OpenSG [10] is the standard choice for the EPOCH consortium.

3 Automatic reconstruction pipeline

The 3D webservice is meant to create 3D reconstructions from a wide variety of images. Because no user interaction is possible once the images have been uploaded, an important prerequisite is the need for robustness and autonomy on the server part. In order to avoid frustra-

tion and multiple uploads at the client side, the server should maximize the chance of obtaining a good 3D result, even from image sequences that are not perfectly suited for the purpose. This requirement has led us to the development of a hierarchical, massively-parallelized and opportunistic 3D reconstruction scheme.

3.1 Pipeline overview

A schematic flowchart of the reconstruction pipeline is shown in Fig. 6. In this figure, rectangles represent procedures or actions. Parallelograms represent data structures. The input of the pipeline is found at the top-left and consists of the set of images the user has uploaded to the server using the upload client. At the bottom right, the result can be seen consisting of dense 3D depth (or range) maps. The pipeline consists of roughly four steps:

1. A step that computes a set of image pairs that can be used for matching, including the Subsampling and Global Image Comparison steps. In this step, the images are first subsampled (hence the hierarchical nature of the pipeline). Since images can be uploaded in non-sequential order, we have to figure out which images can be matched. This is the task of the Global Image Comparison algorithm which yields a set of image pairs that are candidates for pairwise matching.
2. A step that performs the Pairwise and Projective Triplet Matching and the Self-Calibration. In this step, feature points are extracted from the subsampled images. All possible matching candidates of step 1 are now tried. Based on the resulting pairwise matches, all image triplets are selected that stand a chance for projective triplet reconstruction. This process is performed and the results are fed to the self-calibration routine which finds the intrinsic parameters of the camera.
3. A step that computes the Euclidean reconstruction and upscales the result to full resolution. In this step, all image triplets and matches are combined into one 3D Euclidean reconstruction.
4. A step that is responsible for the dense matching, yielding dense, i.e., pixelwise, depth maps for every image.

Sections 4, 5, 6 and 7 describe our approach for each of these steps respectively.

3.2 Opportunistic pipeline

Classic uncalibrated Structure from Motion (a.k.a. Structure And Motion) pipelines make use of the fact that the

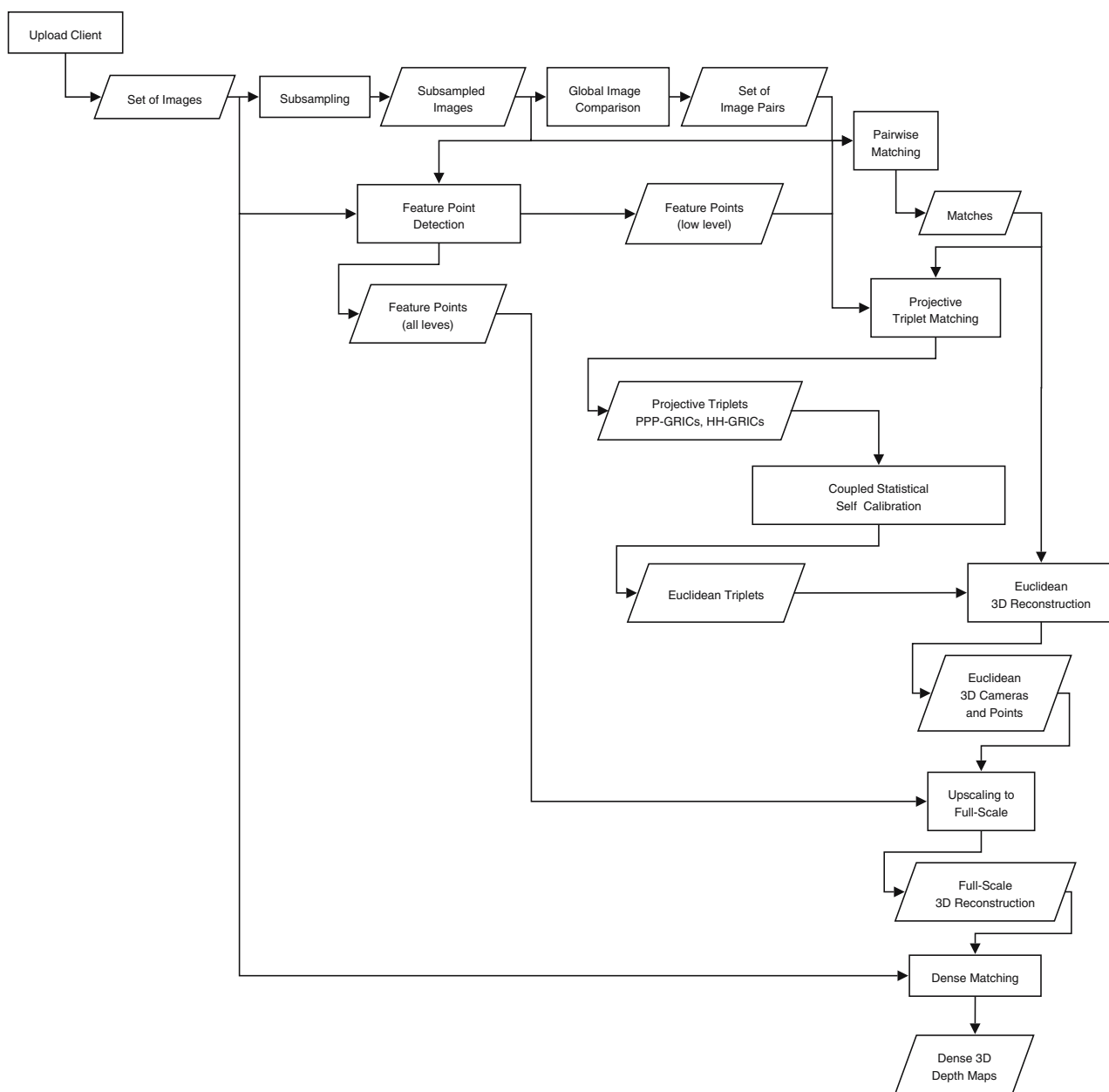


Fig. 6 Global flowchart of the Reconstruction Pipeline

set of input images is taken in a sequential manner. This helps the reconstruction process tremendously because only consecutive pairs of images must be matched for 3D reconstruction. Moreover, when the input consists of video, subsequent frames are very similar and matching is therefore relatively easy. Unfortunately, the 3D Webservice described in this paper can not rely on this assumption. Users can upload images in non-sequential order or even use images that were taken in a random fashion. The system has to actively look for opportunities to turn 2D data into 3D models. This has an impact

on the matching step, the reconstruction step and the dense matching step.

3.3 Hierarchical pipeline

In general, the quality and accuracy of the resulting depth maps is proportional to the size of the input images. However, computing feature points and matches on large-scale images is very time consuming and not so stable a process. That is why all incoming images are

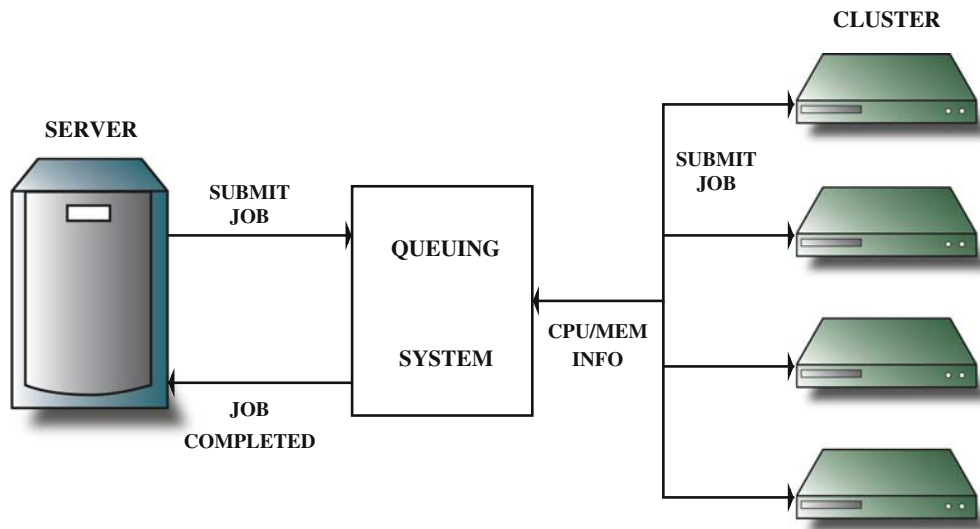


Fig. 7 Queuing system. The server sends a job to the system. The queue has access to a PC cluster and sends the job to any of the nodes that becomes free

first subsampled a number of times until they reach a typical size in the order of $1,000 \times 1,000$. As can be seen in Fig. 6 most of the Structure And Motion processing is performed on these subsampled images. It is only in the upscaling step (at the bottom right) that the result is upgraded from the low resolution to the high input resolution. This hierarchical approach combines a number of advantages. First, it is more stable and has a higher chance of success than direct processing of the high resolution images. Indeed, it is easier to find matching features between small images because the search range is smaller and therefore fewer false matches are extracted. On lower-level images feature points are typically more dominant than on higher levels where features also appear in noisy and cluttered areas that are very hard to match. Furthermore, processing times decrease when dealing with smaller images. The upscaling step receives a very good initialization from the lower levels. This means that only a small search and optimization needs to be performed for the higher resolutions. The upscale step is followed by a search for extra features and correspondences at the highest level.

3.4 Parallel pipeline

Several operations in the reconstruction pipeline have to be performed many times and independently of each other. Image comparison, feature extraction, pairwise or triplet matching, dense matching, etc are all examples of such operations. The pipeline is implemented as a python script which is automatically triggered by the SQL database when a new job arrives. The script has to go through several steps and every step can only be

started when the previous one has finished. Inside one step, however, the processing is parallelized. The server on which the script runs has access to a queuing system of our own making, as shown in Fig. 7. When an operation must be performed, the server sends the job to the queuing system which returns a job id. The queuing system has access to a PC cluster and continuously checks the memory and CPU load of each of the nodes. When a node is free, the job on top of the queue is sent to that node. When the job has finished, the server is automatically notified by the creation of a file the name of which contains the job id.

4 Global image comparison

The images that are uploaded to the reconstruction server can be taken and uploaded in any random order. This means of course that it no longer suffices to process consecutive images. Unfortunately, matching every possible pair of images is a very time-consuming step. That's why we want to find out which images can be matched first.

All possible pairs of images are fed to the Global Image Comparison algorithm. This algorithm is based on comparing the two images under scrutiny with a comparison function like Normalized Cross-Correlation (NCC). The goal of this step is to quickly come up with pairs of images for which matching and computation of epipolar geometry is likely to succeed.

Figure 8 shows a schematic overview of the algorithm. First, the center part of the first image is selected. A window with the same size slides over the second image and the NCC value of both windows is evaluated.

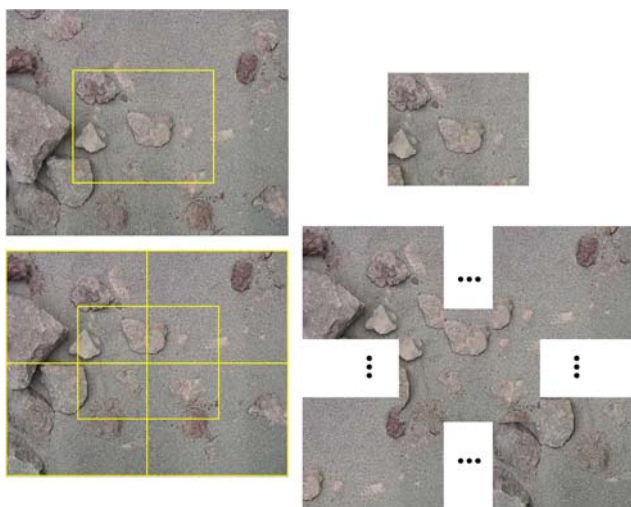


Fig. 8 Comparing two images. The global overlap between the two images is computed as well as the resulting correlation value

The window position for the largest NCC value yields the relative displacement and the amount of overlap. For efficiency reasons, the images are subsampled a couple of times in addition to the subsampling already mentioned in paragraph until a size of about 100×100 has been reached. A beneficial side effect of the subsampling is that the chance of local maxima of the NCC values is greatly reduced. All possible image pairs are processed in parallel (as explained in Sect. 3.4) and only those pairs with a NCC value that is sufficiently high are kept for real pairwise matching. The shift between the windows that yielded this highest NCC value is stored and used later to facilitate the search for matches between the images.

5 Self-calibration

5.1 Classical SaM techniques

Classic Structure and Motion systems either use a priori information on internal camera calibration parameters or go through the process of first reconstructing the scene in a projective framework and then upgrading this reconstruction to metric using self-calibration and bundle-adjustment techniques [3, 7].

First feature points are extracted and consecutive images are matched. Epipolar geometry is retrieved and a projective frame is initialized. All consecutive images are matched to their predecessor and every new projective camera is computed in the projective frame of the first pair. As explained by Cornelis et al. [1], all projective reconstructions obtained in a sequential way

suffer from drift. In a worst-case scenario, the drift of the recovered solution w.r.t. the true solution can become so substantial that self-calibration is no longer possible because the recovered projective reconstruction is not consistent anymore.

The simple sequential reconstruction approach poses other problems as well. Very regularly images are taken of a scene or object which is planar or close to it. Obtaining a unique projective reconstruction of such a planar scene is mathematically impossible because multiple projective solutions exist. The appearance of different projective frames in the solution causes self-calibration algorithms to fail as well. Also, contrary to the classical techniques, our input images are not always taken in a purely sequential way. It is very disadvantageous if these non-sequential images are only processed sequentially. Images should not just be matched to their direct predecessor to reconstruct their cameras.

5.2 Triplet matching

In order to alleviate the problems mentioned in the previous paragraph, a new reconstruction and self-calibration algorithm has been developed which has proved to be stable and rewarding on image sequences that could not be processed with the sequential approach. As explained in Sect. 4 all images in the uploaded set can be compared to each other efficiently. This comparison yields a list of image pairs that are candidates for feature matching. The extracted feature points are matched between each of these image pairs and epipolar geometry is computed. If the matching yields a sufficiently good result, the result is stored. Possible image triplets are selected based on the matching results. Only triplets with enough matches between both images 1 and 2 and images 2 and 3 are withheld. For these triplets a projective reconstruction is computed, in the form of three projective cameras and a set of 3D points [3]. All these calculations (image pair matching and image triplet reconstruction) can be performed independently and thus all processing is done in parallel using the queuing system described in paragraph.

Our self-calibration will later make use of the reconstructed triplets. Not only do we want to reconstruct as many triplets as possible, but we also need to investigate if they are useful for self-calibration purposes. In order to do so, we make use of the Geometric Robust Information Criterion (GRIC) proposed by Torr [12], a mathematical description of Occam's razor which states that "one should not increase, beyond what is necessary, the number of entities required to explain anything". If different models exist to explain certain data, the GRIC paradigm selects the model with the lowest penalty. The

penalty of a model is obtained by summing two contributions. The first one is related to the goodness of the fit and the second one is related to the parsimony of the model. GRIC takes into account the number n of inliers, the residuals e_i , the standard deviation of the measurement error σ , the dimension of the data r , the number k of model parameters and the dimension d of the structure:

$$\text{GRIC} = \sum \rho(e_i^2) + (nd \ln(r) + k \ln(rn)). \tag{1}$$

For a triplet, we try to explain matches between images in two different ways, either as a structure with three projection matrices or as two times two images with one common image in the middle, related by planar homographies. The GRIC values for these two models are called PPP-GRIC and HH-GRIC, respectively. If a scene is near to planar or if only a planar part is in common between the three views of a triplet, the corresponding HH-GRIC will have a lower penalty than the PPP-GRIC, indicating that a model consisting of two homographies better explains the data than the model using three projection matrices. Such a triplet is therefore not suited for self-calibration and can only be correctly reconstructed if the internal camera parameters have already been recovered, e.g., from other parts of the data.

5.3 Coupled self-calibration

In [8] the concept of coupled self-calibration was introduced for a limited set of projectively reconstructed sequences which share the same intrinsic parameters. The approach is based on the projection equation for the absolute quadric [13]:

$$\mathbf{K}\mathbf{K}^\top \sim \mathbf{P}\Omega^*\mathbf{P}^\top, \tag{2}$$

where Ω^* represents the absolute quadric. In metric space $\Omega^* = \text{diag}(1, 1, 1, 0)$, in projective space Ω^* is a 4×4 symmetric rank 3 matrix representing an imaginary disc-quadric. When choosing $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$ for one of the projection matrices it can be seen from (2) that Ω^* can be written as:

$$\Omega^* = \begin{bmatrix} \mathbf{K}\mathbf{K}^\top & \mathbf{a} \\ \mathbf{a}^\top & b \end{bmatrix}. \tag{3}$$

Now any set of constraints on the internal parameters are translated to constraints on the absolute quadric and can be written as:

$$[\mathbf{C} \ \mathbf{D}] \begin{bmatrix} \mathbf{k} \\ \mathbf{a} \\ b \end{bmatrix} = 0, \tag{4}$$

where \mathbf{k} is a vector containing six coefficients representing the matrix $\mathbf{K}\mathbf{K}^\top$, \mathbf{a} is a 3-vector and b a scalar and \mathbf{C} and \mathbf{D} are matrices containing the coefficients of the equations. Their size is $n \times 6$ and $n \times 4$, respectively with n the number of constraints. Note that this can be done independently for every 3D subsequence. If the sequence is recorded with constant intrinsics, the vector \mathbf{k} will be common to all subsequences and one obtains the following coupled self-calibration equations:

$$\begin{bmatrix} \mathbf{C}_1 & \mathbf{D}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_2 & \mathbf{0} & \mathbf{D}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_n & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{D}_n \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \mathbf{a}_1 \\ b_1 \\ \mathbf{a}_2 \\ b_2 \\ \vdots \\ \mathbf{a}_n \\ b_n \end{bmatrix} = 0. \tag{5}$$

5.4 Statistical coupled self-calibration

Coupled self-calibration is a powerful extension of self-calibration of a single projective sequence. It allows to calibrate reconstructions in different projective frames in one step. This is of high interest if only short projective sequences are available. Such sequences do not entail enough information to recover all parameters of the calibration model in a stable way if used by themselves. Coupled in one computation, they allow to recover the internal parameters of the camera. The most extreme example of a short projective sequence is a projective triplet like the ones we recovered in. The calibration algorithm only yields good results if all reconstructions at the input are consistent in their projective frame. With only three views present in each triplet, there is a chance that a combination of several triplets contains at least one weakly reconstructed triplet. That is the reason why we opted for a statistical approach.

All reconstructed triplets are possible candidates for coupled self-calibration. First, the PPP- and HH-GRIC of each triplet is compared. Only triplets for which the PPP-GRIC value is lower than the HH-GRIC are retained because we are certain that there is enough 3D information in these triplets for self-calibration. From this set of triplets, we randomly pick a certain number (7 in the current implementation). The matrix on the left of (5) is built and we solve for the internal parameters. If the smallest singular value of the SVD which solves this system is small enough the solution is accepted and stored. This step is repeated many ($\sim 1,000$) times or until all triplet combinations have been tried, each time picking another random set of triplets. Each solution for the absolute quadric corresponds to a matrix

with intrinsic parameters \mathbf{K} . We retrieve the top left element of each \mathbf{K} , representing the focal length in pixels in x -direction and create a histogram of these values. A typical histogram is shown in Fig. 9. This figure shows the typical Gaussian-like distribution of the self-calibration results we obtained from a multitude of experiments. The maximum of the Gaussian is selected and the \mathbf{K} -matrix corresponding to this value is chosen. Of course, the self-calibration procedure is not meant to yield perfect calibration results but is rather used as a good initialization for bundle-adjustment steps that are executed later.

6 Reconstruction

Sections 5.2 and 5.4 explained how we retrieve projective reconstructions of many image triplets and how we use these triplets to compute the intrinsic parameters of the camera that took the images. A straightforward next step is to upgrade every triplet from its projective to a metric frame, using the intrinsic parameters \mathbf{K} . However, this procedure brings us only halfway towards the true goal: to retrieve the camera calibration and 3D points for all images in the same metric frame because each triplet is still reconstructed in its own metric frame.

In order to bring all information together, we start by searching for the triplet which is best suited for initializing our final 3D reconstruction. This triplet should have a large number of 3D points and have a large $(HH-GRIC)/(PPP-GRIC)$ value. When the triplet

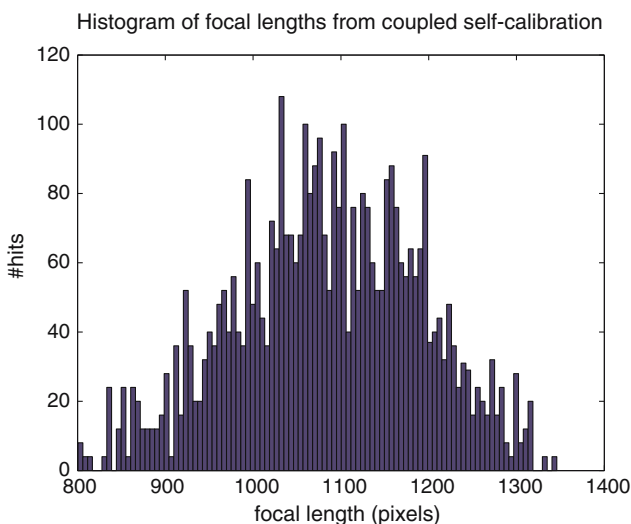


Fig. 9 Histogram of focal lengths for all solutions of coupled self-calibration from a set of triplets

which maximizes a function combining both values¹ is found, our reconstruction is initialized. Other images can now be added to the reconstruction one at a time. This is not done in a purely sequential way where only matches to the previously added image are employed but rather using all matches with all already added images. Using these matches a robust pose-estimation algorithm is executed which computes the camera of the newly added image in the metric frame of the reconstruction. The fact that the pose-estimation can be done in metric space is an advantage because now also images which only observe a planar part of the scene can be reconstructed.

6.1 Upscaling the result

All previous computations are performed on the sub-sampled images, both for efficiency and stability reasons. It is now time to upgrade the result to the full-scale images that were uploaded, a process we have called upscaling. An iterative algorithm was implemented for that purpose. It is executed as many times as needed to go from the small low-level images to the highest level of the original size images. The simplest way to upscale the result to the original image size is to upscale the resulting calibration. Every upscaling with a factor of two corresponds to scaling the intrinsic parameters of the lower level (\mathbf{K}_0) to the higher level (\mathbf{K}_1) for every image i . The extrinsic parameters (\mathbf{R} and \mathbf{t}) of every camera i and the position of every 3D point j (M_j) can stay the same. This yields the new cameras \mathbf{P}_i .

$$\mathbf{K}_{0i} = \begin{bmatrix} f_{x0i} & s_{0i} & u_{x0i} \\ 0 & f_{y0i} & u_{y0i} \\ 0 & 0 & 1 \end{bmatrix}, \tag{6}$$

$$\mathbf{K}_{1i} = \begin{bmatrix} f_{x1i} & s_{1i} & u_{x1i} \\ 0 & f_{y1i} & u_{y1i} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2f_{x0i} & s_{0i} & 2u_{x0i} \\ 0 & 2f_{y0i} & 2u_{y0i} \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R}_{1i} = \mathbf{R}_{0i},$$

$$\mathbf{t}_{1i} = \mathbf{t}_{0i},$$

$$M_{1j} = M_{0j},$$

$$\mathbf{P}_{1i} = \mathbf{K}_{1i} \left[\mathbf{R}_{1i}^T | - \mathbf{R}_{1i}^T \mathbf{t}_{1i} \right].$$

This solution can only serve as an initialization for an optimization process because all errors of the low-level reconstruction are upscaled as well. As can be seen in Fig. 6 we computed feature points in the images on all resolution levels, including the original images. The

¹ In the current implementation the maximum of $(w(HH-GRIC)/(PPP-GRIC) + N)$ is computed with w a weighing factor, set to 100 and N the number of 3D points.

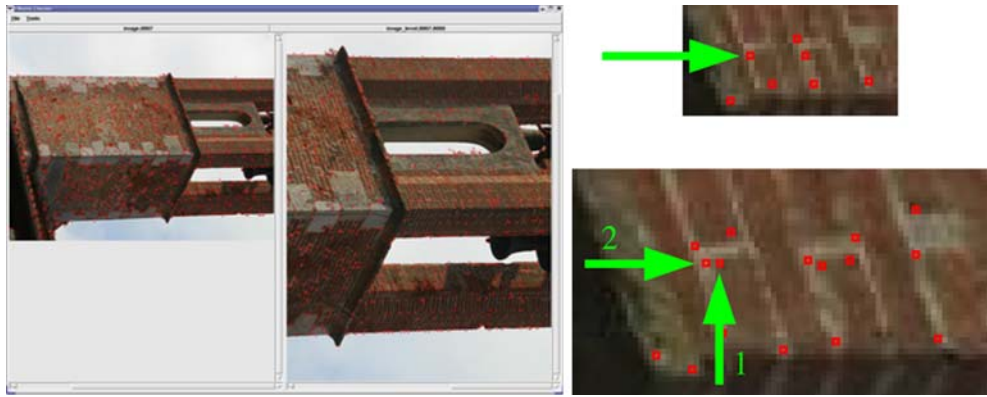


Fig. 10 Subsampled and original image with extracted features are shown *on the left*. *On the right* one part of the images has been cut out. The arrow at the top points to a feature with a 3D point in the low-level image. At the bottom *arrow 1* points to the feature

in the high-resolution image which is closest to the reprojection of the 3D point using the initialization (6). *Arrow 2* points to the feature that is selected after an iteration of the 3D points and camera parameters

number of extracted features is increased for every consecutive level. These features help to optimize the result of (6). The upscaling algorithm goes as follows.

1. Compute an initialization for the cameras \mathbf{P}_{li} and points M_{lj} of the new level l based on the level below $(l - 1)$ as explained in (6).
2. The 3D–2D correspondences that indicate in which images every 3D point is visible remain identical.
3. For every 3D point M_{lj} perform the following operation:
 - Project M_{lj} in the images where it is visible using the upscaled cameras \mathbf{P}_{li} .
 - Search the extracted features of the image at level l for the feature that is closest to the projected point.
 - Change the 3D–2D correspondence to this closest feature.
4. Perform a Robust Euclidean Bundle Adjustment on all 3D–2D correspondences. This step computes updated values for the 3D points M_{lj} and cameras \mathbf{P}_{li} . This step is explained in Sect. 6.2.
5. Iterate steps 3 and 4. Because of the updated correspondences, the updated values of the points and cameras will yield new projections. Some of these will now be closer to another feature in the feature set. Figure 10 shows an example. This implies a new change to the 3D–2D correspondences and hence again an updated solution. The iteration is stopped when no more updates of the 3D–2D correspondences are made. This is typically reached after 5–6 iterations.

The upscale step described above computes the Euclidean projection matrices and 3D points for the full-

scale images. Non-linear radial distortion parameters can be estimated as well. The technique has one drawback however. Only originally matched points will be upscaled and no extra points are added to the reconstruction. The reconstructed points are those that are inliers to the epipolar geometry of image pairs. Since at the time of computation of this geometry no information on the radial distortion is available, typically few points near the borders of the image are matched if the images suffer from this kind of distortion (radial distortion has the largest effect on points far away from the center). After the upscaling step, however, we have the parameters describing the distortion to our disposal. More matching points can be searched, taking into account this distortion. Figure 11 shows an example. The top two images show an image pair matched at the smallest level without taking radial distortion into account. The bottom image pair shows the matches on the full size images after accounting for radial distortion. Especially the foreground region of the images shows much more matched features.

6.2 Robust Euclidean Bundle adjustment

The Robust Euclidean Bundle Adjustment of step 4 is an extension of the traditional bundle adjustment techniques [14]. In these least-squares estimators the global reprojection error is minimized. It has been shown in numerous studies that least-squares estimators are vulnerable to outliers. Even a single bad sample may perturb these least-squares estimates completely. In the upscaling step outliers are inevitable. A robust version of the classical bundle adjustment technique is called for. A popular technique to deal with possible outliers is the so-called *M-estimators*. Their use and applicability is well

Fig. 11 The *top two images* show an image pair matched at the smallest pyramid level without taking radial distortion into account. The *bottom image pair* shows the matches on the full size images accounting for radial distortion. Especially the *foreground region of the images* shows much more matched features

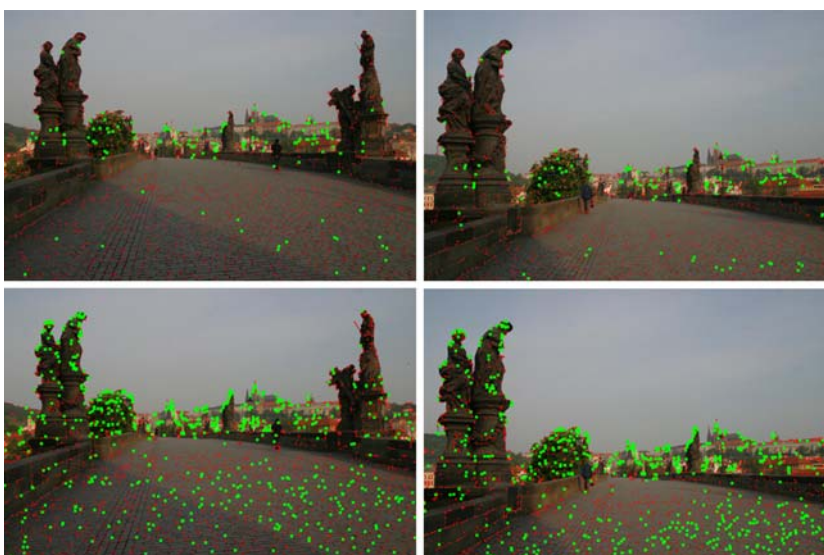


Fig. 12 Reconstruction of points and cameras for the (sequential) “Prato” example

described in [16]. In short, M-estimators try to reduce the effect of outliers by replacing the squared reprojection errors by another function of these residuals. In our implementation, we chose Cauchy’s function, also known as the Lorentzian function given in (7). Instead of minimizing the sum of the residuals $\sum r_i^2$ we now minimize $\sum \rho(r_i)$. The parameter c of (7) is set to 1 which means that the M-estimator starts limiting the influence of outliers when they are more than one pixel away from their estimate.

$$\rho(x) = \frac{c^2}{2} \log \left(1 + \left(\frac{x}{c} \right)^2 \right). \tag{7}$$

Although the resulting 3D points and calibration from the upscaling algorithm are accurate, the number of 3D points can be rather low. As will be explained in Sect. 7 however, dense matching approaches can use reconstructed feature points as seeds. It is therefore important to keep the number of 3D points sufficiently high. The recovered calibration is used to search for new correspondences in very localized areas (along the epipolar line for the first two images and in a neighborhood around the projection of the 3D point in consecutive images).

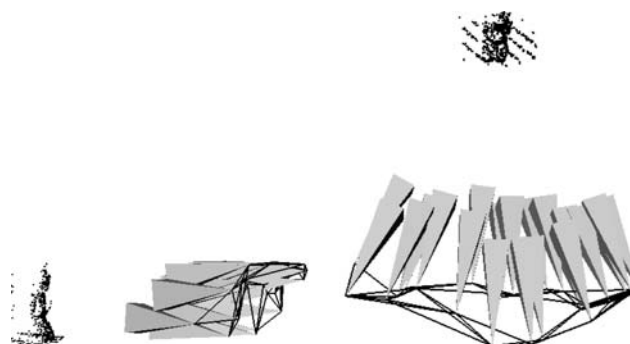


Fig. 13 Reconstruction of points and cameras for the (non-sequential) “Thai statue” example

A typical result, consisting of calibrated cameras and a set of reconstructed 3D points is shown in Figs. 12 and 13, showing a sequential and non-sequential image set, respectively. The calibration of the non-sequential image set of Fig. 13 makes use of non-



Fig. 14 Point reconstruction of the *skull* sequence. The images are clearly recorded in a non-sequential order

Fig. 15 Reconstruction of the *skull* sequence made with the PDE-based dense matching method

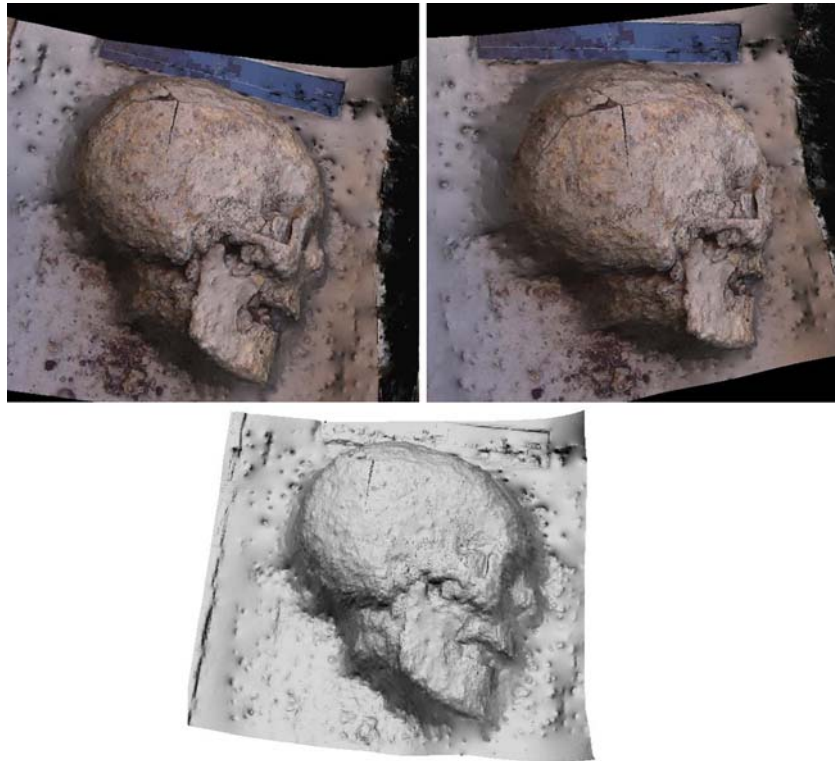
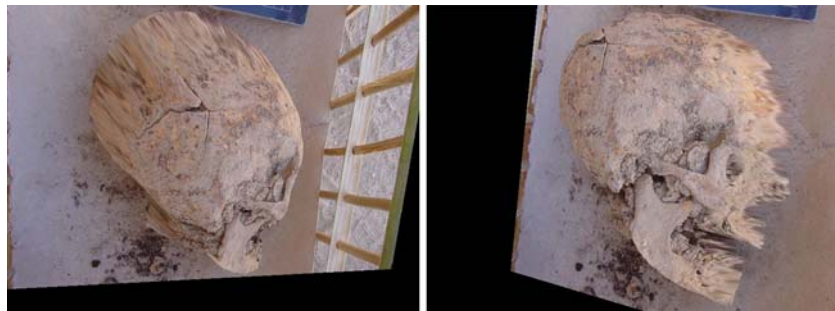


Fig. 16 Reconstruction of the *skull* sequence made with the GPU-based dense matching method



sequential links between images. After the initialization with the best triplet, new cameras are added using all matches with all already added images. The links that were used in this specific example are shown as lines connecting the camera centers.

7 Dense matching

The result of the Euclidean reconstruction explained in the previous sections is a set of points and camera parameters, as shown in Fig. 12. This sparse set of points typically is not enough. Dense, textured 3D models is what is usually required. In order to deliver these, a dense matching process between the input images is necessary.

7.1 Linked pairwise stereo

In previous publications [7], a technique for dense matching using linked pairwise stereo has already been thoroughly explained. This strategy entails matching all consecutive pairs of an image sequence with a stereo algorithm [6]. The resulting disparity maps between the image pairs are then combined into depth maps for every image using a statistical algorithm that checks for consistency of matched pixels over multiple pairs. This approach has proven its merits for sequential image sets, like the Prato example of Fig. 12. Quality maps (see Sect. 2.2) indicating the confidence we have in the 3D reconstruction of every pixel are part of the output. The more consecutive views in which a pixel is matched consistently, the more confident we are of its 3D position, hence the higher the quality measure for this pixel.



Fig. 17 Reconstruction of the “Prato” example. The *top two views* show textured models, *the bottom view* shows a non-textured version

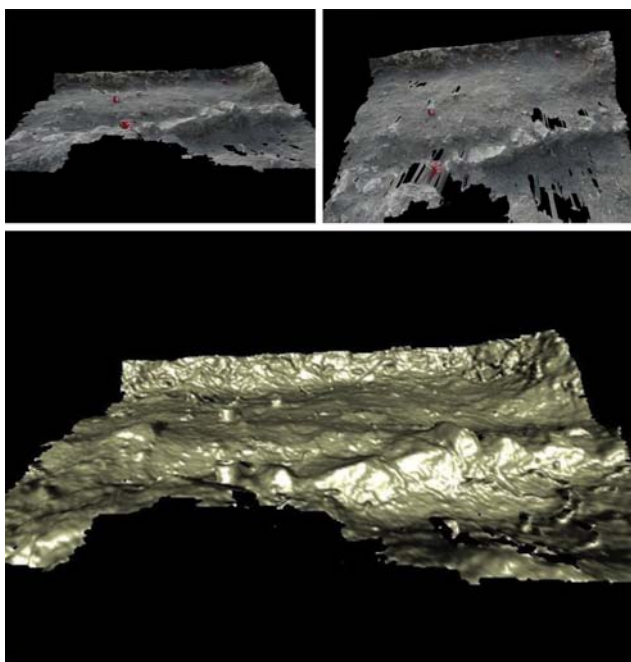


Fig. 18 Reconstruction of a stratigraphic layer in Sagalassos. The *top two views* show textured models, *the bottom view* shows a non-textured version

As already mentioned, we cannot assume that the uploaded image sets have indeed been recorded and uploaded sequentially. Using the linked pairwise stereo method can easily lead to failure with such non-sequential image sets. If we just process the images

pairwise as before the chances of encountering an image pair with extreme disparities or even hardly any overlap are very real. This would immediately break the link between the pairs, resulting in inferior depth maps. A possible strategy would be to search for a sequence connecting the images in the best order. This is not an easy task. We have to decide which criterion will be used to assess whether two images should be next to each other in the sequence or not. For this, one could use the number of pairwise matches (a result of the algorithm described in Sect. 5.2). Unfortunately, while this number gives a good indication of the matching quality between every pair, it does not give any information on the quality of the resulting 3D reconstruction which could be very bad for close views which have a small baseline. Furthermore, finding the optimal path in the image set boils down to solving a Traveling Salesman problem, which is $n-p$ hard and scales very badly. That is why, for non-sequential image sets, we want to change the linked pairwise stereo method for a true multi-view stereo approach.

7.2 Multi-view stereo

Multi-view stereo algorithms seek to reconstruct the 3D shape of a scene directly from a combination of more than two images. Again, the most difficult step is to find the correspondence between points in different viewpoints. One can make a distinction between two types of techniques. The first are the so-called volumetric algorithms. They start with an approximation of the volume of the 3D scene. This can be computed from reconstructed feature points or, more common, with visual hull techniques. The initialization of the volume evolves towards the true shape by enforcing photo-consistency between points that are seen in different images [9, 15].

Other techniques do not make use of a volumetric description of the scene but try to find multi-view correspondences from the images directly [4, 11], possibly initialized from reconstructed feature points. The technique of [11] builds up a probabilistic model of the correspondences between all views and a reference view, taking into account the possibility of occlusions, color changes, . . . and maximize the probability of the solution. This result can then serve as initialization to a pixel-based PDE approach.

Fast direct multi-view methods have been implemented on the basis of GPUs. Graphics boards have become fully featured processing units with dedicated high-performance memory and up to 16 parallel pipelines. The introduction of vertex and fragment programs caused a major breakthrough, giving the programmer almost complete control over the GPU. A

Fig. 19 Four of the five uploaded images of the Templo de los Mascarones at the archaeological site of Edzna in Yucatan



technique which uses these GPUs for multi-view stereo was explained in [2].

A non-sequential image set of a skull was recorded and uploaded to the webservice. A point-based reconstruction is shown in Fig. 14 which shows the non-sequential nature of the recording. Two textured and one non-textured views of the result of the PDE based method are shown in Fig. 15. Different views of the result of the GPU-based method are shown in Fig. 16.

These are two techniques for dense matching provided by the webservice. The first puts the emphasis on precision, the second on speed.

8 Results

At the time of submission of this paper, the 3D web-based reconstruction service was in beta-testing. Several image sets have been uploaded to the service by various users in the cultural heritage field. In this section, we show some results.

Figure 17 depicts some reconstructions of the facade of the Cattedrale di Santo Stefano in Prato, Italy. The image set was recorded and uploaded sequentially as can be seen in Fig. 12.

Figure 18 shows resulting depth maps for the stratigraphy at an archaeological excavation. Instead of using textual descriptions of each layer and measuring positions of artefacts with a ruler, the webservice offers a more efficient alternative. Each layer is recorded with a photo camera and these pictures are uploaded to the service. The results describe the layer in 3D, including textures. The second screenshot in Fig. 18 shows a top

view. The image for which this depth was computed, was taken from the side. It is clear that occluded areas behind rocks and other obstacles have not been reconstructed and show up as holes in the top view. These should be filled by merging depth maps. This is the goal of the steps in Fig. 1 that follow the reconstruction pipeline described in this paper.

Finally, Fig. 19 shows four out of the five uploaded images of the Templo de los Mascarones (Temple of the Masks), at the Archaeological site of Edzna, which is a Maya site in the country of Yucatan, near Campeche in Mexico. Figure 20 shows two textured and one non-textured views of the resulting reconstruction.

9 Conclusion and future work

In this paper, we described the first part of a complete 3D reconstruction system for the cultural heritage field. The system under scrutiny is in charge of computing the camera calibration and dense depth maps from images of an object or a scene. These images are uploaded to a processing server which automatically computes the results, making use of a cluster of PCs, and makes these results available on an ftp server. Several specifics of the upload-server paradigm have to be taken into account, such as the fact that no user interaction is possible and that input images might not be taken or uploaded in a sequential order. These constraints drove the design of the pipeline which is automatic, opportunistic, hierarchical and tries to execute as many processes in parallel as possible.



Fig. 20 Two textured and one non-textured views of the reconstruction of the Templo de los Mascarones. The latter gives an especially clear view of the complicated geometry of the scene

Several aspects of the reconstruction service can still be improved. In order to create a usable tool for the cultural heritage sector, the integration of our reconstruction system with the tool which aligns and integrates

different depth maps will have to be improved. An important aspect of this will consist of better and more automatic segmentation of foreground and background and the choice of depth maps that will represent the data in a sequence. On the client site, the level of self-diagnosis of the system can be improved. Regularly image sets are uploaded that do not meet the requirements for 3D reconstruction and therefore fail. Some simple algorithms can be envisaged which check the feasibility of the image set a user is about to upload. On the processing site, the feedback to the user must be improved. In the case of failure, analysis of the errors should make sure to supply specific guidelines to the user on how to improve his image data. Also, we plan to make use of features and matching methods that allow to overcome wider baselines than is currently possible. Some early experiments with SIFT [5] and other features have shown that the positional accuracy of these features is rather low for 3D reconstruction purposes. We are looking into alternative features.

The 3D webservice is intended for use by the Cultural Heritage community. User accounts for professionals active in this domain can be applied for via the website at <http://homes.esat.kuleuven.be/~visit3d/webservice/html>

Acknowledgements The authors wish to thank Christoph Strela and Nico Cornelis for their dense reconstruction of the *skull* sequence, Glyn Matthews for his contributions to the webservice interfaces, Frank Verbiest for algorithmic support and Maurizio Forte of CNR-ITABC in Rome for letting us use the results of his uploaded recording of the skull and the Templo de los Mascarones. The authors gratefully acknowledge support by the EC 6FP NoE EPOCH.

References

1. Cornelis, K., Verbiest, F., Gool, L.V.: Drift detection and removal for sequential structure from motion algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(10), 1249–1259 (2004)
2. Cornelis, N., Gool, L.J.V.: Real-time connectivity constrained depth map computation using programmable graphics hardware. In: *CVPR* (1), pp. 1099–1104 (2005)
3. Hartley, R.I., Zisserman, A.: *Multiple View Geometry in Computer Vision*, second edn. Cambridge University Press, Cambridge, ISBN: 0521540518 (2004)
4. Jin, H., Soatto, S., Yezzi, A.: Multi-view stereo reconstruction of dense shape and complex appearance. *Int. J. Comput. Vision* **63**(3), 175–189 (2005). DOI 10.1007/s11263-005-6876-7. URL <http://dx.doi.org/10.1007/s11263-005-6876-7>
5. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **60**(2), 91–110 (2004)
6. Meerbergen, G.V., Vergauwen, M., Pollefeys, M., Gool, L.V.: A hierarchical symmetric stereo algorithm using dynamic programming. *Int. J. Comput. Vision* **47**(1), 275–285 (2002)
7. Pollefeys, M., Gool, L.V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. *Int. J. Comput. Vision* **59**(3), 207–232 (2004)

8. Pollefeys, M., Verbiest, F., Gool, L.V.: Surviving dominant planes in uncalibrated structure and motion recovery. In: Proceedings of the 7th European Conference on Computer Vision-Part II, pp. 837–851 (2002)
9. Pons, J.P., Keriven, R., Faugeras, O.D.: Modelling dynamic scenes by registering multi-view image sequences. In: CVPR (2), pp. 822–827 (2005)
10. Reiners, D.: Opensg. Ph.D. thesis, TU Darmstadt, Fachbereich Informatik (2002)
11. Strecha, C., Fransen, R., Gool, L.V.: Wide-baseline stereo from multiple views: A probabilistic account. In: Proceedings of the CVPR 2004, pp. 552–559 (2004)
12. Torr, P.: An assessment of information criteria for motion model selection. In: Proceedings of the CVPR97, pp. 47–53 (1997)
13. Triggs, B.: The absolute quadric. In: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition, pp. 609–614 (1997)
14. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment: a modern synthesis. *Vision Algorithms: Theory and Practice*, LNCS **1883**, 298–372 (2000)
15. Vogiatzis, G., Torr, P., Cipolla, R.: Multi-view stereo via volumetric graph-cuts. In: Proceedings of the CVPR 2005, pp. 391–398 (2005)
16. Zhang, Z.: Parameter estimation techniques: A tutorial with application to conic fitting. *Image Vision Comput. J.* **15**(1), 59–76 (1997)