Time budget evaluation for image-based reconstruction of sewer shafts

Sandro Esquivel[a] and Reinhard Koch[a] and Heino Rehse[b]

[a]Christian-Albrechts-University, Kiel
[b]IBAK Helmut Hunger GmbH & Co. KG, Kiel

ABSTRACT

In this paper we propose a robust real-time image and sensor based approach for automatic 3d model acquisition of sewer shafts from survey videos captured by a downward-looking fisheye-lens camera while lowering it into the shaft. Our approach is based on Structure from Motion adjusted to the constrained motion and scene, and involves shape recognition techniques in order to obtain the geometry of the scene appropriately. We perform a time budget evaluation for the components of an existing off-line application based on previous work and design a real-time application which can be applied during on-site inspection. The methods of our approach are modified so that they can be executed on the GPU. Expensive bundle adjustment is avoided by applying a simple and fast geometric correction of the computed reconstruction which is capable of handling inaccuracies of the intrinsic camera calibration parameters.

Keywords: sewer inspection, 3d scene reconstruction, structure from motion, spherical camera system

1. INTRODUCTION

Automatic 3d reconstruction from video has been a topic of research in photogrammetry and computer vision for a long time and has been largely studied. Recent systems approach real-time reconstruction by implementing algorithms on the graphics processing unit (GPU) which is capable of highly efficient parallel computations.

An important application for automatic scene reconstruction is the inspection of sewers and sewer shafts. Remotely controlled inspection devices such as mobile robots are commonly used for this task since the structures under observation are often not directly accessible for humans or access is difficult to achieve. Conventional systems are remote controlled and deliver visual data which is analyzed by experts on the site or afterwards from stored video. In order to facilitate the surveillance process, commercial sewer inspection systems demand for measuring the local geometry of the scene which can also be used for later visualization. Near real-time applications which can be used on site are preferable in order to support decisions timely. While there are systems using active sensors such as laser scanners available, purely visual reconstruction approaches are interesting to enhance existing systems and reduce production costs.

In this paper we will propose a real-time approach for image-based reconstruction of sewer shafts with a specific camera setup. First, we will give an overview over the setting and related work. In the main part, we perform a time budget evaluation of an existing off-line approach proposed by us[1] and modify the methods used in order to achieve near real-time processing such that the reconstruction method can be used on site during image acquisition.

Further author information: (Send correspondence to S. Esquivel)
S. Esquivel: E-mail: esquivel@mip.informatik.uni-kiel.de, Telephone: +49 431 880-4448

1.1 Problem specification and setting

The setting of our work is illustrated in figure 1: A fisheye-lens camera designed for sewerage survey is lowered vertically into a sewer shaft which is specified to be vertical with arbitrary basic shape, but often rectangular shafts or shafts with elliptical profile are found. The camera is looking downwards into the shaft. Images are captured in fixed translation intervals (here: every 5 cm) which can be measured accurately from the feed of the conducting cable. In the given setting, the camera moves up to 35 cm/s, but a flash light ensures sharp images every 5 cm. Hence, images are captured with a maximum rate of 7 fps, and our system must be capable to process at least at such frame rate, which gives a time budget of 142 ms per frame. The camera is additionally equipped with an inertial sensor which measures the differential rotation around the viewing axis for each image. We will use this information to compensate roll rotation in the images in order to facilitate feature point matching. While it is assumed that the camera is looking approximately along the axis of the shaft, the exact position of the camera (i.e. the off-center position and orientation with respect to the shaft) is unknown. The camera will also inevitably oscillate around the cable axis. The task is to recover sparse local geometry and the positions of the camera within the shaft, and to classify and measure the cross-sectional shape of the shaft at the camera positions robustly during image acquisition. These data can be used e.g. by an interactive viewer to inspect the shaft virtually in 360°, or to measure distances within the camera image. Up to now, the reconstruction is computed off-line after the inspection session and is analyzed by an expert afterwards in the office.

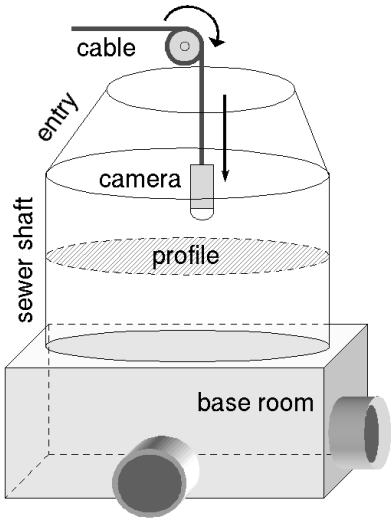

Figure 1. Setup for shaft inspection.

Figure 2 shows typical input images captured by the fisheye-lens camera while lowering it into a sewer shaft through the manhole. Apparently, the task of visual reconstruction is not trivial: Illumination and visibility decrease rapidly towards the center of the image, the hanging camera is rotating and oscillating significantly around its view axis, there are reflections especially on fronto-parallel parts of the shaft surface and obscuring structures such as stairs and branching pipes, and vision is very poor in larger rooms where the camera is located off-center.

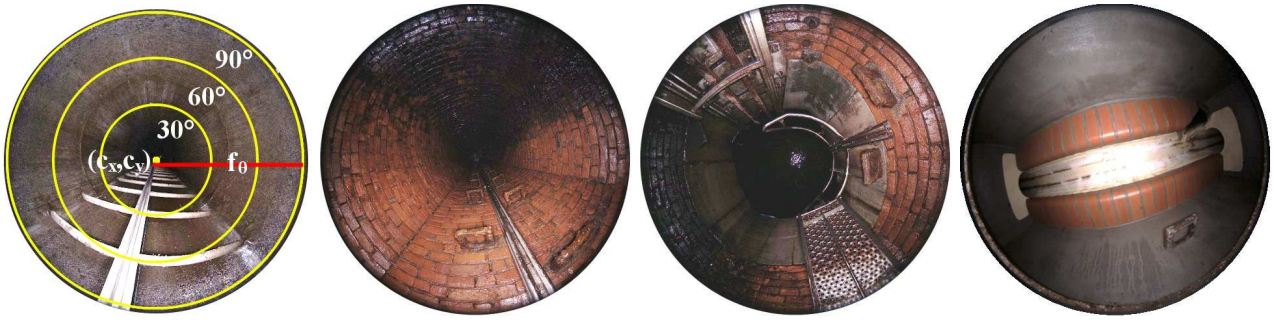

Figure 2. Input images captured by a hanging fisheye-lens camera during lowering.

1.2 Related work

Structure from Motion (SfM) techniques which simultaneously estimate the camera pose and sparse scene structure from corresponding points in subsequent images are widely used for 3d scene reconstruction from video sequences. Recently it has become popular to implement computer vision algorithms on the GPU. Utilizing the GPU, real-time Structure from Motion methods have become feasible, e.g. for large-scale urban reconstruction.[2] Careful algorithm design has also led to more efficient estimation methods such as the generic preemptive RANSAC scheme,[3] or more specific methods as the weighted RANSAC scheme proposed by Hedborg et al.

for visual odometry.[4] In general, real-time 3d reconstruction lacks accuracy since global bundle adjustment is not feasible. Mouragnon et al.[5] have surveyed different types of SfM techniques and have proposed a generic real-time approach which is based on incremental 3d reconstruction and local generic bundle adjustment which is also applicable to fisheye-lens cameras.

Specific reconstruction approaches have been proposed for sewer reconstruction and the reconstruction of cylindrical scenes. An early idea for recovering shape and camera pose relative to the sewer axis automatically from sewer survey videos was presented by Cooper et al.[6] Kannala et al.[7,8] considered a Structure from Motion approach for automatic 3d model acquisition from video sequences captured by a calibrated fisheye-lens camera moving through a sewer pipe. They recover camera positions and scene structure by computing calibrated multi-view tensors for image sub-sequences and merging the results hierarchically, which results in a point cloud approximating the scene structure as an initial 3d model. This approach suffers from error accumulation and sensitivity to inaccurate camera calibration resulting in bent and conical pipe reconstructions. Global bundle adjustment could be used to correct such errors partly.

Based on the results of Kannala et al.,[7,8] we have proposed an approach for the given problem which has been implemented and is successfully used in practice with the camera systems delivered by our industrial partner.[1] The prototype has been implemented using the BIAS software framework maintained at our work group,[9] and is not optimized for computational speed but for robustness and automation and is intended to be performed as an off-line post-processing step. The main idea of our approach is to incorporate a priori knowledge about the scene geometry and camera motion to facilitate the SfM method used. Instead of finding feature correspondences directly in the fisheye images, we perform a mapping to a virtual cylinder first. Sparse shaft geometry and camera positions are computed, and the shaft profile is measured at regular depth steps, as given by the camera positions (every 5 cm). The reconstruction is finally geometrically corrected using knowledge about the camera motion instead of using time-expensive bundle adjustment. Inspired by recent real-time SfM techniques,[2,5] we have evaluated the time budget for our implementation and modified it to approach real-time performance as described in the next section.

2. REAL-TIME 3D SHAFT RECONSTRUCTION

For the reconstruction of the shaft geometry, a Structure from Motion (SfM) approach is used which simultaneously estimates the camera pose and sparse scene structure from corresponding points in subsequent images. The original off-line approach consists of three phases which are processed subsequently: Image preprocessing of all camera images, reconstruction via Structure from Motion, and model creation from the reconstruction results performed as a post-processing step. The reconstruction phase consists of an initialization state (INIT) and a tracking state (TRACK). The original SfM pipeline is ordered as follows (see also figure 3):

Image preprocessing: Acquisition and preprocessing of all images.

1. Read images and synchronized rotation sensor data from disk and convert to grey-value float images.
2. Mapping of fisheye images onto virtual cylinder around camera axis.
3. Normalize brightness of cylinder images using the average brightness distribution in all images.

Reconstruction: Perform Structure from Motion for whole sequence.

1. KLT feature tracking with geometry-specific feature prediction (TRACK), or row-wise KLT feature matching for (re-)initialization of the reconstruction (INIT), and detection of new KLT feature points.
2. Robust pose estimation from 2d/3d correspondences (TRACK), or robust epipolar geometry estimation from 2d/2d correspondences and extraction of pose from essential matrix (INIT), using the traditional RANSAC scheme.[10]
3. Triangulation of new 3d points from 2d/2d correspondences, and update of existing 3d points via an Extended Kalman filter, using the estimated camera pose.

4. Reconstruction is re-initialized (TRACK → INIT) when the algorithm has lost track of too many feature points (e.g. when a different geometry segment begins and feature predictions do not hold anymore), or pose estimation fails.

Model creation: Refine whole point reconstruction and measure shaft profiles.

1. Simple geometric correction of the camera path and the scene geometry is capable of reducing the effect of camera calibration errors and camera drift, and replaces expensive bundle adjustment.
2. Local scene geometry is estimated by fitting 2d shapes to the ortho-projection of triangulated 3d points within vertical slices of 5 cm onto the ground plane.

In the following, these steps will be explained in detail. We will always discuss the off-line method first, describe the time budget restriction, and will then give a real-time solution.

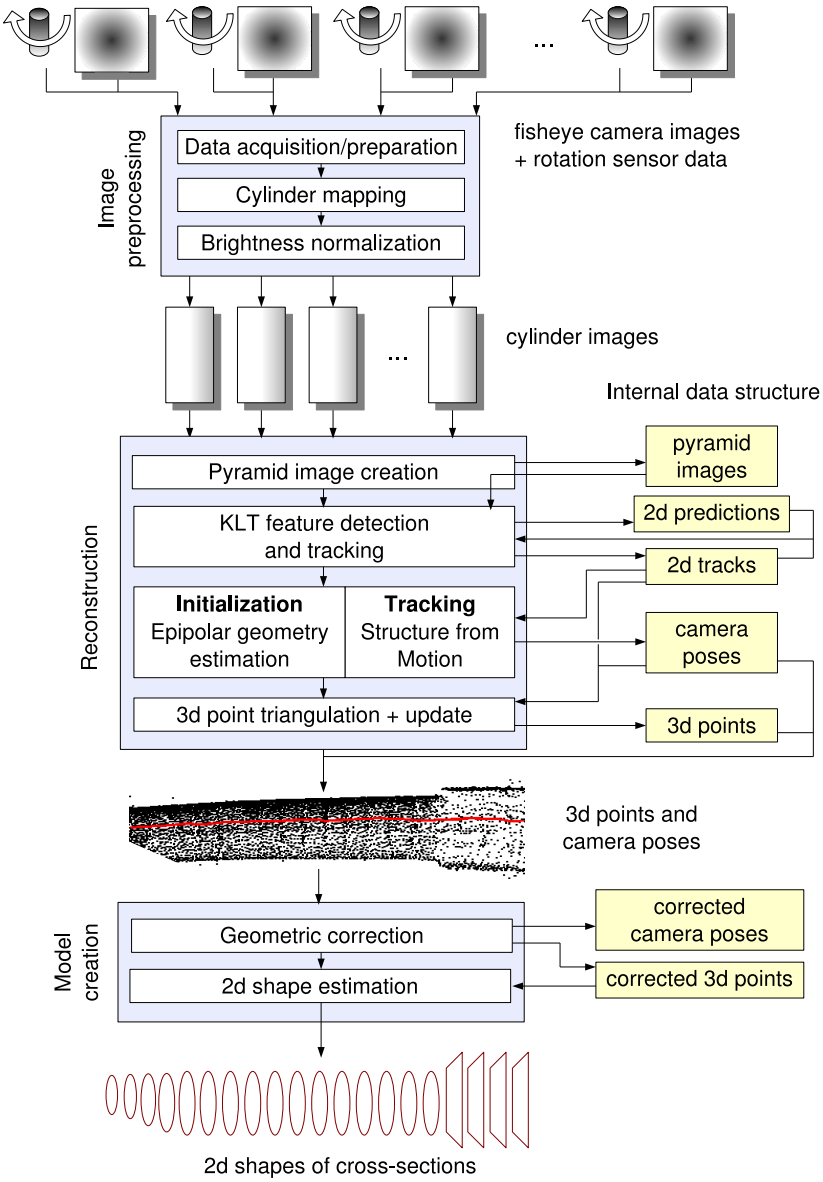

Figure 3. Processing modules and data flow of our original offline reconstruction approach.

2.1 Camera model

We employ an adaption of the sphere camera model proposed by Scaramuzza et al.[11] which is capable of describing single viewpoint catadioptric cameras and fisheye-lens cameras. The camera is assumed to be calibrated in general, e.g. by the Omnidirectional Camera and Calibration Toolbox for Matlab,[12] and has negligible radial distortion. Nevertheless, we also want to handle calibration inaccuracies.

According to the spherical camera model, 2d points in the fisheye image can be described by azimuth angle ϕ and distance d to center point (c_u, c_v), corresponding to 3d points within the camera coordinate system with spherical coordinates (ϕ, θ). Note that points with the same inclination angle θ are mapped onto concentric rings in the fisheye image which are ideally isoangular disregarding lens distortion. The radius of the 90° circle in the fisheye image in pel is denoted as f_θ in the following (see also fig. 2).

2.2 Image preprocessing

The fisheye-lens camera delivers JPEG-compressed color-valued images with 1040×1040 pel resolution. Images are converted into float grey-value images and mapped onto cylinder coordinates in order to facilitate the feature tracking process. Since image brightness decreases towards the image center due to the lighting conditions, we perform a brightness normalization of the image after cylinder mapping to facilitate feature point matching.

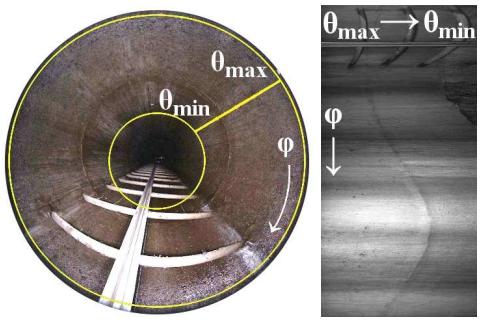

Figure 4. Fisheye image with region of interest, and cylinder-mapped image.

Cylinder mapping projects a part of the fisheye image onto a virtual cylinder with radius r_{cyl} along the camera axis. The forward mapping function is described by the mapping of image pixels (u, v) to rays (ϕ, θ) given in spherical coordinates within the camera coordinate frame using the known intrinsic parameters of the camera (eq. 1), followed by the computation of the intersections with the cylinder surface in cylinder coordinates (ϕ, z) which are mapped to cylinder image pixels (u', v') (eq. 2). By adding the absolute roll angle ϕ_{sensor} measured by the inertial sensor, rays are transformed into a globally aligned coordinate frame with respect to the world's z-axis, and we obtain rotation-invariant cylinder images. Typically, we create cylinder images with width $w_{\text{cyl}} = 512$ pel and $h_{\text{cyl}} = 1024$ pel, since radial resolution is greater than resolution in depth.

$$\phi = \text{atan2}(v - c_v, u - c_u) + \phi_{\text{sensor}}, \quad \theta = \sqrt{(u - c_u)^2 + (v - c_v)^2} \cdot f_\theta \tag{1}$$

$$z = \tan(\frac{\pi}{2} - \theta) \cdot r_{\text{cyl}}, \quad u' = \frac{z - z_{\min}}{z_{\max} - z_{\min}} \cdot w_{\text{cyl}}, \quad v' = \frac{\phi}{2\pi} \cdot h_{\text{cyl}} \tag{2}$$

$z_{\min} = \tan(\frac{\pi}{2} - \theta_{\max})$ and $z_{\max} = \tan(\frac{\pi}{2} - \theta_{\min})$ are given by the inner and outer inclination angles $\theta_{\min}, \theta_{\max}$ of the ring-shaped region of interest in the fisheye image that is used for featurepoint tracking. Typically, we use $\theta_{\max} = 90°$ and $\theta_{\max} = 33°$, i.e. we reject the inner circular area with $\theta < 33°$ in the fisheye images. The resulting depth range is about 2.25 times the radius of the shaft when the camera is centered. Refer to fig. 4 for an illustration of cylinder mapping.

The backward mapping is defined by the following equations (3, 4). Note that the backward mapping can easily be implemented on the GPU, e.g. using the Cg shading language.[13]

$$z = u' \cdot \frac{z_{\max} - z_{\min}}{w_{\text{cyl}}} + z_{\min}, \quad \theta = \frac{\pi}{2} - \text{atan2}(r_{\text{cyl}}, z), \quad \phi = v' \cdot \frac{2\pi}{h_{\text{cyl}}} \tag{3}$$

$$u = \cos\phi \cdot \theta \cdot f_\theta + c_u, \quad v = \sin\phi \cdot \theta \cdot f_\theta + c_v \tag{4}$$

As shown in table 1, the time consumed by image preprocessing is about 120 ms per frame on average which is mainly used by image acquisition (41 ms for reading JPEG image from disk and extracting sensor data from EXIF field) and by cylinder mapping (78 ms). Brightness adjustment which is mainly a scaling of the image values takes negligible time (1 ms/frame).

To approach real-time performance, we modified our approach so that reading images and inertial sensor data is performed by a separate thread. The thread also performs the cylinder mapping on the GPU which consumes on average 6.3 ms per frame as shown in the right column of table 1 which is a reduction of $> 90\%$. Since we will use a gain-adaptive KLT feature tracker later, we can also omit the brightness normalization. The resulting total duration for the image preprocessing thread is on average bounded by 50 ms/frame (20 Hz) which satisfies our real-time requirements.

2.3 Feature tracking

The first step of the reconstruction is finding corresponding feature points within pairs of images captured at subsequent time steps. For video sequences with sufficiently high capture frame-rate, the problem of finding local correspondences can be treated as a tracking problem rather than feature matching. The well-known KLT (Kanade-Lucas-Tomasi) approach[14,15] is commonly used for feature tracking. Image pyramids are used for tracking to overcome the limitation to subpixel motion.There are implementations of the KLT tracker which utilize the GPU and can hence be performed in real-time. One GPU-based KLT tracker implementation is proposed by Sinha et al.,[16] which has drawbacks on newer GPUs such as the Nvidia GeForce 8 series as pointed out by Zach et al.[17]

In the given setting, feature tracking is challenging due to the camera's framerate of 7 Hz at a speed of 35 cm/s which results in large feature point translations of up to 50 pel in subsequent frames. Nevertheless the feature point translation is very constrained due to the linear camera motion and cylindrical scene geometry, hence tracking is feasible by applying a prediction method for feature point positions in subsequent image frames. Without knowledge about the local scene geometry (e.g. diameter of the shaft), we apply a feature search approach which is only used for (re-)initialization of the tracking.

Another problem in tracking is the violation of the constant brightness assumption. Since the flashlight of the camera produces an inhomogeneous environment illumination, features on the shaft surface change their intensity between frames as the camera approaches them. Kim et al. proposed a gain-adaptive version of the KLT tracker[18] which has been modified to perform in real-time on the GPU by Zach et al.[17]

Due to the cylindrical scene geometry and the alignment of the camera, it is evident that featurepoints move mainly horizontally within the cylinder images. Small vertical motion is caused by oscillations of the camera. The horizontal offset between frames depends on the distance of the corresponding 3d point to the camera center. Given a cylinder image with width $w = 512$ pel and height $h = 1024$ pel, and a regarded camera view angle range of $\alpha = 66°$, we have shown in previous work[19] that for typical shaft diameters ranging from 50 to 250 cm we can expect maximal offsets of up to 50 pel and < 10 pel vertical offsets. Note that we can derive offsets from knowledge about the local scene geometry and vice versa. Without any knowledge about scene geometry and hence about feature offsets, feature correspondences are established by matching within a search window of 50×15 pel to the left of each feature point. Once feature correspondences have been found, offsets are averaged for each image row and stored as row-wise offset predictions for the next frame. For rows without predictions, offset predictions are interpolated from adjacent rows with predictions. Offset predictions are updated as featurepoints are tracked in new frames. For tracking we use a window of 15×15 pel. When the tracker has lost track of too many features, tracking is reinitialized with the matching method described above and offset predictions are reset. This happens typically when the shaft geometry is changing significantly, e.g. at the transition between shaft and base room or intermediate chambers.

In our original approach, feature detection on the CPU consumes on average 45 ms/frame with max. 1000 KLT features present in each image. Finding initial correspondence takes 386 ms/frame, and tracking with prediction takes 107 ms on average. Since the tracker proposed by Zach et al. is brightness-invariant and has better performance than the real-time KLT tracker proposed by Sinha et al.,[16] we use it in our approach, and modify it to support the adaptive feature point prediction described above. The resulting feature detection and tracking consumes on average 15 ms/frame for tracking and 46 ms/frame for initial correspondence creation which is a reduction of $> 90\%$ with respect to the our original approach.

2.4 Pose estimation

Pose estimation of the camera within the shaft at different time steps is divided into two states: During initialization only 2d/2d point correspondences are available. The classical approach to estimate the local pose with respect to the first camera position is to estimate the essential matrix describing the epipolar geometry, and extracting local orientation and translation from it.[20] Since the translation can only be recovered up to scale, we use knowledge about the amount of translation between the first two images from the cable feed (5 cm) to obtain the metric scaling. To achieve robustness, we use a RANSAC approach[10] for essential matrix estimation. After pose estimation, 3d points can be created from 2d/2d correspondences via triangulation. Once 2d/3d correspondences are known in the tracking state, the relative camera pose is estimated in combination with a RANSAC approach to achieve robustness. When pose estimation or feature tracking fails, the reconstruction process switches back to the initialization state.

In practical application, our offline-approach needed on average 196 ms for pose initialization from 2d/2d correspondences and 56 ms for pose estimation from 2d/3d correspondences. Computational effort is spent mainly by the repeated evaluations of the RANSAC. Since we expect high outlier ratios due to moving particles in the shaft, dirt and water drops on the camera etc., we evaluate 1000 samples for each RANSAC. In order to increase runtime, we make use of the preemptive RANSAC scheme proposed by Nistér[3] which is suitable for live Structure from Motion as shown by Pollefeys et al.[2] Preemptive RANSAC extends the traditional method by preemptive scoring and rejection of motion hypotheses and hence reduces the number of observation evaluations. Statistical evaluation has shown that the runtime can on average be reduced by at least 50%, resulting in an expected runtime of < 100 ms for pose initialization and < 30 ms for pose estimation during tracking.

2.5 Triangulation and update of 3d points

For robustness and adaptivity, we use a triangulation method which takes into account both the uncertainty of the 2d featurepoint positions and of the camera pose described by their covariance matrices.[21] 3d points with too large uncertainty are rejected. The Mahalanobis distance between 2d featurepoints and reprojected 3d points is evaluated to reject inconsistent 3d points. To increase accuracy, an Extended Kalman filter[22] is used to update the positions and covariance matrices of 3d points which are visible in multiple views, replacing the multi-view tensor from Kannala's work.[7] Typically, featurepoints will be lost after 3–8 images, depending on the diameter of the shaft and the oscillations of the camera. In our original approach, triangulation of new 3d points consumed on average 9 ms/frame with respect to max. 1000 featurepoints per image. Update of 3d points visible in multiple views via the Extended Kalman filter needed on average 36 ms/frame. During experiments, it became evident that we can not omit the 3d point update since reconstructions will often fail in this case. Since the computations in this steps demand for high accuracy, we leave them on the CPU without modification.

2.6 Geometric correction

As described in our previous work on sewer shaft reconstruction,[1] error accumulation in pose estimation resulting from short featurepoint tracks, as well as inaccuracies in the intrinsic camera calibration result in systematic reconstruction errors: We observed camera drift, resulting in bent and conical shaft geometry. Time-consuming bundle adjustment is typically used to account for such errors which is no choice due to real-time requirements. Instead, we use restrictions on the camera path to perform a simple geometric correction of the 3d points.

Since the camera is lowered into the shaft on a cable, the average camera path is expected to follow the vector of gravity (the z-axis of the world coordinate system by definition). Due to the image acquisition process, we know that the distance between the camera positions in subsequent images is approximately 5 cm in depth (z-coordinate). By mapping the estimated camera path to the expected camera path and translating 3d points with respect to the last camera they have been observed by, we can correct bending and scale change of the reconstruction. The distorted average camera path is modeled by fitting a second-degree 3d polynomial $P(t)$ to the camera centers $C_0, \ldots, C_N$. Given any 3d point X, let $P(t_X)$ denote the orthogonal projection of X onto $P(t)$. The projections of the camera centers onto the average path are mapped to the z-axis at fixed depth intervals of 5 cm. 3d points and camera center positions are corrected by computing their perpendicular distance $\|\Delta X\|$ to the average camera path polynomial and placing them at this distance perpendicular to the global z-axis. The local reconstruction scale is corrected by the ratio λ_k of the expected camera path (k times 5 cm)

and the estimated camera path length $\ell(t_{C_k})$ in the k-th image frame. The position X of 3d points and camera centers belonging to the k-th image are hence corrected to X^* as given by eq. 5:

$$X^* = \lambda_k \cdot \begin{pmatrix} \delta_X \cdot \Delta X_x \\ \delta_X \cdot \Delta X_y \\ \ell(t_X) \end{pmatrix} \text{ with } \lambda_k = \frac{k \cdot 5\,\text{cm}}{\ell(t_{C_k})} \;,\; \ell(x) = \int_0^x \|P(t)\|\,\mathrm{d}t \;,\; \Delta X = X - P(t_X),\;\; \delta_X = \frac{\|\Delta X\|}{\sqrt{\Delta X_x^2 + \Delta X_y^2}} \tag{5}$$

where $P(t_X)$ is the orthogonal projection of X onto $P(t)$, and C_k is the original position of the camera in the k-th image frame. Refer to fig. 5 for an illustration of the geometric correction procedure. For efficiency, the camera path length $\ell(t_X)$ to a 3d point X in image k is approximated as $\ell(t_X) \approx \sum_{i=1}^{i=k} \|P(t_{C_i}) - P(t_{C_{i-1}})\| + \|P(t_X) - P(t_{C_k})\|$ which can be precalculated efficiently for each frame up to the last term. Note that t_X is obtained from minimizing $\|X - P(t)\|^2$ and can be computed efficiently by a closed-form solution for finding the roots of a cubic polynomial when $P(t)$ is of degree 2.

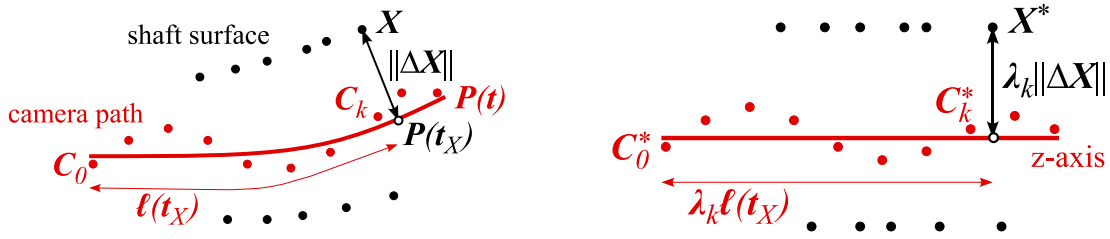

Figure 5. Camera path and reconstructed 3d points before (left) and after geometric correction (right).

Geometric correction is applied as a post-processing step in our original approach which takes all estimated cameras poses and 3d points as input. We have shown in our previous work that we can compensate calibration inaccuracies of the fisheye camera of up to 10 pel for f_θ. The time consumed is on average 1.3 ms/frame as seen in table 1. Almost the entire time is used for geometric correction of 3d points and camera positions according to eq. 5, the time needed for polynomial fitting and path length computation is negligible ($\sim$ 1 ms).

In our real-time modification, we can also consider to apply geometric correction after *each* step instead of as a post-processing step, e.g. to visualize the results frame by frame during image acquisition. In this case, only currently visible 3d points are refined (typically 100 – 200 3d points), which are used to measure the cross-sectional shape of the shaft at the current position of the camera (see section 2.7). The average time used for geometric correction of the partial reconstruction online is measured to be 6.8 ms/frame which is still sufficiently fast for real-time application.

Since each 3d point is corrected independently, this could also be executed very efficiently on the GPU. First experiments showed average computation times of 0.03 ms/frame when computed for the whole sequence, and 0.16 ms/frame for partial reconstructions in each frame. Nevertheless, the resulting 3d reconstruction has reduced accuracy due to the floating-point arithmetics of the GPU.

2.7 Measuring cross-sectional shape

After geometric correction of the reconstructed 3d points, the cross-sectional shape of the shaft at the camera position is measured by fitting 2d shapes to the ortho-projection of 3d points within a slice of 5 cm depth onto the ground plane (the x/y-plane of the world coordinate system by definition). Given are different shape classes which are commonly found as sewer shaft profiles (circle, ellipse, square, rectangle, ovoid). The shaft profile is classified by fitting one instance of each shape class robustly to the projected 2d points using a RANSAC approach, and choosing the instance with the highest score (i.e. highest inlier ratio and minimal distance to shape). Since inconsistent 3d point have been already removed at the pose estimation/triangulation step, we expect low outlier ratios with respect to the real profile shape. Shape classification and estimation consumes on average 8 ms/frame in our original approach which is sufficiently fast for real-time processing (see table 1). Nevertheless, we expect to reduce the time by using an preemptive RANSAC approach to at least 5 ms/frame.

2.8 Putting it all together

Our modified approach is finally outlined as follows (see also figure 6):

Input thread Handles image input and preprocessing.

1. Read current camera image and rotation sensor data.
2. Perform cylinder mapping on the GPU.

Reconstruction thread Performs SfM and model creation incrementally.

1. Detection of KLT features and brightness-adaptive KLT feature tracking on the GPU (TRACK), or row-wise KLT feature matching for (re-)initialization, as in the original approach (INIT). Pyramid images used by the tracking algorithm are also created on the GPU to gain speed.
2. Robust pose estimation (TRACK), or epipolar estimation (INIT), using the preemptive RANSAC.
3. Triangulation of new 3d points, and update of existing 3d points, using the estimated camera pose.
4. Geometric correction of camera path and scene geometry reconstructed so far is recomputed in each frame. The positions of 3d points are optionally corrected on the GPU for sake of accuracy. For efficiency, only the currently visible 3d points needed for profile measuring are processed.
5. 2d shapes are robustly estimated "on-the-fly" for visible 3d points, using the preemptive RANSAC.
6. The reconstruction can re-enter the initialization state (TRACK → INIT) as in the original approach.

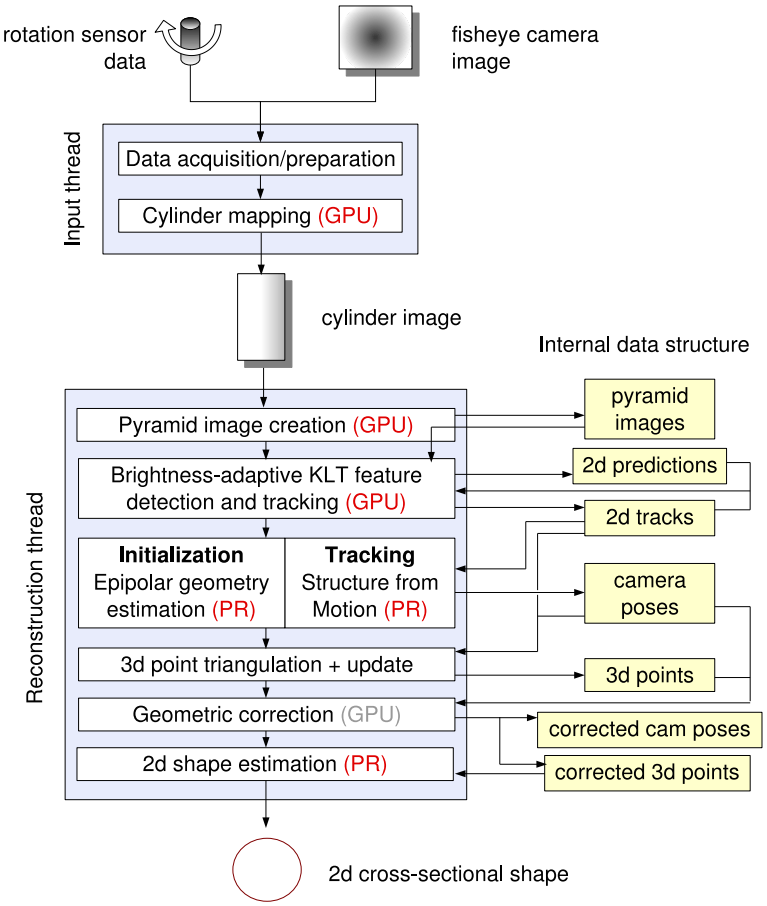

Figure 6. Processing modules and data flow of our new real-time reconstruction approach.

3. EXPERIMENTS AND RESULTS

To evaluate the performance, our industry partner has provided us with 49 video sequences captured during sewer shaft inspection with the presented camera system. The observed shafts show a great variety in depth, diameter, and shape. The length of the sequences ranges from 28 to 390 images, corresponding to shaft depths of 1.5 m up to 20 m. An example for the results of our approach (3d point cloud, camera path, and estimated shaft profile at certain depth) is shown in fig. 7 for a sewer shaft of 3 m depth.

We have performed a time budget analysis for our original approach per image frame and compared the times for each step with implementions of the modified real-time methods as proposed. All experiments have been performed on a Intel quad-core CPU at 2.66 GHz with 8 GB RAM, using a GeForce 9800GTX graphics card. The results are listed in table 1. We derive the time budget for a real-time application using our modifications and show that reconstruction within the given time limit is feasible.

The off-line approach consumes on average 410 ms per frame which exceeds the time limit of 142 ms given by the image capturing rate of 7 Hz by large. Decoupling image preprocessing and Structure from Motion to different threads, using the preemptive RANSAC scheme, and processing cylinder mapping on the GPU results in an expected average frame time bounded by 137 ms per frame, i.e. a frame rate of at least 7.3 Hz which meets our real-time requirements. Geometric correction (optional on the GPU) and profile shape estimation are used as a post-processing step as in the off-line approach here.

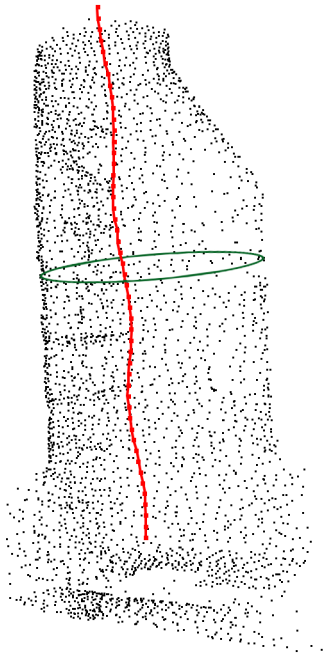

Figure 7. Example for 3d reconstruction of sewer shaft with camera path.

To approach reconstruction during image acquisition, the model creation step has to be applied in each frame instead of as a post-processing step on the whole reconstruction. Experiments show an increased average time spent for geometric correction of 6.8 ms/frame on the CPU (< 1 ms on the GPU). This results in an average frame rate of 7.04 Hz which is still satisfying the time budget. Nevertheless, reconstruction during image acquisition

	offline approach	**real-time approach**	
task	**time/frame**	**time/frame**	**comment**
image acquisition	41.2 ± 11.3 ms	41.2 ± 7.3 ms	not modified
cylinder mapping	78.1 ± 7.3 ms	6.3 ± 0.7 ms	GPU
brightness adjustment	1.02 ± 0.33 ms	–	omitted
total preprocessing	120.4 ± 14.8 ms	< 50 ms	1st thread
KLT detection	45.4 ± 8.7 ms	–	included in tracking
KLT tracking (INIT)	386.4 ± 172.8 ms	46.07 ± 1.5 ms	GPU (rare)
KLT tracking (TRACK)	106.7 ± 35.2 ms	15.1 ± 1.1 ms	GPU
pose estimation (INIT)	196.4 ± 69.1 ms	< 100 ms	preempt. RANSAC (rare)
pose estimation (TRACK)	55.9 ± 26.5 ms	< 30 ms	preemptive RANSAC
triangulation	9.2 ± 4.07 ms	9.2 ± 4.07 ms	not modified
3d point update	36.2 ± 14.5 ms	36.2 ± 14.5 ms	not modified
total reconstruction	290.2 ± 128.6 ms	< 130 ms	2nd thread
geometric correction	1.3 ± 0.4 ms	1.3 ± 0.4 ms / 0.03 ± 0.05 ms (6.8 ± 1.6 ms / 0.16 ± 0.1 ms	post-processing / GPU for each frame / GPU)
2d shape estimation	8.0 ± 2.2 ms	< 5 ms	preemptive RANSAC
total post-processing	$9.6 + -2.5$ ms	< 7 ms (< 12 ms)	2nd thread
total time	410.07 ± 168.8 ms	< 137 ms (< 142 ms)	

Table 1. Time budget evaluation for original off-line approach and modified real-time approach per frame using 49 test sequences with 28–390 images each (~ 4800 images in total). INIT steps are only called 2.48 ± 1.6 times per sequence.

is still difficult since it implies to meet the time budget of 1/7 s as an upper bound for each frame rather than an average bound. As shown in table 1, the initialization steps are several times more time consuming than the tracking steps, and standard deviations of frame times are quite large, which requires buffered load-balancing over the sequence. But since initialization steps occur on average only 2–3 times for each sequence, their impact on the total running time is low. With respect to the results, we achieve an expected total runtime within the range of the image capturing process which is suitable for application on site as demanded.

4. CONCLUSION

We have proposed a robust real-time approach for sewer shaft reconstruction and cross-sectional shape measurement using a specific camera setup which can be used on site within the same time as the image capturing process takes, rather than remote in the office. To gain real-time performance, we have analyzed the time budget for an existing off-line approach developed by us previously, and adopted recent GPU implementations of computer vision algorithms and real-time approaches for robust estimation. Although the approach has been implemented only in parts by now, we are confident to develop a prototype in cooperation with our industrial partner, which can be used successfully in practice in order to facilitate the sewer surveillance task. Future work will also be done to enable the reconstruction procedure to run online during image capturing.

ACKNOWLEDGMENTS

This work has partially been funded by the "Zukunftsprogramm Schleswig-Holstein (2007-2013)" with funds from the European Commission (EFRE) and Land Schleswig-Holstein, Germany, as part of the Initiative KoSSE, project 122-09-048, and by our industrial partner IBAK Helmut Hunger GmbH & Co. KG, who has also supported this work with their expertise and provided us with a large set of test sequences.

REFERENCES

[1] Esquivel, S., Koch, R., and Rehse, H., "Reconstruction of sewer shaft profiles from fisheye-lens camera images," *Lecture Notes in Computer Science* **5748**, 332–341 (2009).

[2] Pollefeys, M., Nistér, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewénius, H., Yang, R., Welch, G., and Towles, H., "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision* **78**, 143–167 (2008).

[3] Nistér, D., "Preemptive RANSAC for live structure and motion estimation," *Machine Vision and Applications* **16(5)**, 321–329 (2005).

[4] Hedborg, J., Forssén, P.-E., and Felsberg, M., "Fast and accurate structure and motion estimation," *Lecture Notes in Computer Science* **5875**, 211–222 (2009).

[5] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P., "Generic and real-time structure from motion," *Proc. British Machine Vision Conference* (2007).

[6] Cooper, D., Pridmore, T. P., and Taylor, N., "Towards the recovery of extrinsic camera parameters from video records of sewer surveys," *Machine Vision and Applications* **11**, 53–63 (1998).

[7] Kannala, J., *Measuring the Shape of Sewer Pipes from Video*, Master's thesis, Helsinki University of Technology, Helsinki (2004).

[8] Kannala, J., Brandt, S. S., and Heikkila, J., "Measuring and modelling sewer pipes from video," *Machine Vision and Applications* **19(2)**, 73–83 (2008).

[9] MIP, CAU Kiel, Germany, "BIAS: **B**asic **I**mage **A**lgorithm**S** Library." Available at: http://www.mip.informatik.uni-kiel.de/BIAS.

[10] Fischler, M. A. and Bolles, R. C., "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM* **24(6)**, 381–395 (1981).

[11] Scaramuzza, D., Martinelli, A., and Siegwart, R., "A flexible technique for accurate omnidirection camera calibration and structure from motion," *Proc. International Conference on Computer Vision Systems*, 45 ff. (2006).

[12] Scaramuzza, D., Martinelli, A., and Siegwart, R., "A toolbox for easy calibrating omnidirectional cameras," *Proc. IEEE International Conference on Intelligent Robots and Systems* (2006).

[13] Mark, W. R., Glanville, R. S., Akeley, K., and Kilgard, M. J., "Cg: A system for programming graphics hardware in a C-like language," *Proc. SIGGRAPH* (2003).

[14] Lucas, B. D. and Kanade, T., "An iterative image registration technique with an application to stereo vision," *Proc. International Joint Conference on Artificial Intelligence*, 674–679 (1981).

[15] Shi, J. and Tomasi, C., "Good features to track," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 593–600 (1994).

[16] Sinha, S. N., Frahm, J.-M., Pollefeys, M., and Genc, Y., "GPU-based video feature tracking and matching," *Proc. EDGE 2006 Workshop on Edge Computing Using New Commodity Architectures* (2006).

[17] Zach, C., Gallup, D., and Frahm, J.-M., "Fast gain-adaptive KLT tracking on the GPU," *Proc. CVPR Workshop on Visual Computer Vision on GPUs* (2008).

[18] Kim, S., Gallup, D., Frahm, J.-M., Akbarzadeh, A., Yang, Q., Yang, R., and Nistér, D., "Gain adaptive real-time stereo streaming," *Proc. International Conference on Vision Systems* (2007).

[19] Esquivel, S., Koch, R., and Boettcher, M., "3D-Vermessung des Schachtprofils aus Fisheye-Kamerasequenzen - Projektabschlussbericht," tech. rep. (2008).

[20] Hartley, R. and Zisserman, A., [*Multiple View Geometry in Computer Vision, 2nd Edition*], Cambridge University Press (2003).

[21] Förstner, W., [*Handbook of Geometric Computing*], ch. 15, 493–534, Springer Berlin Heidelberg (2005).

[22] Quine, B., Uhlmann, J., and Durrant-Whyte, H., "A new approach for filtering nonlinear systems," *Proc. American Control Conference*, IEEE Press (1995).