# Interactive reconstruction of industrial sites using parametric models

Irene Reisner-Kollmann* VRVis Research Center
Anton L. Fuhrmann† VRVis Research Center
Werner Purgathofer Vienna University of Technology

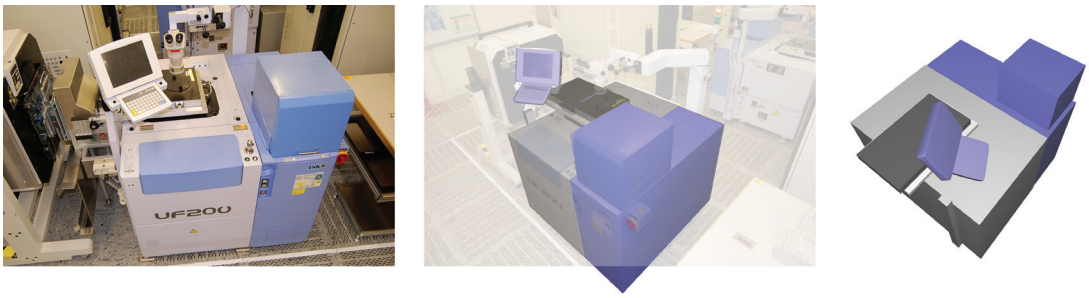


**Figure 1:** *The images show (from left to right) an input image and the resulting model overlaid to another input image and from a new perspective.*

## Abstract

We present a new interactive modeling technique for reconstructing 3D objects from multiple images. We specifically address the problems that arise in industrial environments during camera orientation, image segmentation and modeling. An accurate camera orientation is ensured by using coded markers and surveyed points from a total station. Interactive segmentations of edges and regions in the images are used as input for fitting parametric models to the scene. We provide an intuitive interface which allows modeling artificial objects without having extensive knowledge about 3D modeling or photogrammetry.

**CR Categories:** I.3.5 [COMPUTER GRAPHICS ]: Computational Geometry and Object Modeling — [I.4.8]: IMAGE PROCESSING AND COMPUTER VISION—Scene Analysis

**Keywords:** multi-view reconstruction, model fitting

## 1 Introduction

The modification of an industrial building requires the knowledge and analysis of the existing building. It is important to know the size of rooms as well as the size and shape of facilities and machines. With this information it is possible to plan the rebuilding and the placement of existing and new facilities. A 3D visualization of the plans is often desired for getting a better impression of the new building. It is necessary to efficiently create 3D models and accurately measure the size of industrial facilities. Image-based methods are useful because the actual facilities are only needed for taking a few photographs. If accuracy is important, the image-based methods can be supported by additional measurements, e.g. with a total station.

Several methods for reconstructing a scene automatically from a set of images have been presented in recent years, e.g. [Furukawa and Ponce 2007]. These methods usually create three-dimensional objects with a lot of polygons and depend on short camera baselines as well as diffuse surfaces. On the other hand, there are photogrammetric applications [ImageModeler; PhotoModeler] for manually creating 3D objects based on photogrammetric methods. They need a lot of user input, mainly creating point correspondences across multiple views, but in exchange they are independent to lighting conditions and camera baselines. Another advantage may be that the resulting 3D models are very simple and only have a few polygons. A drawback is that calculations are very error-prone if the user makes a mistake or the user input is not accurate enough.

In this paper, we present a new modeling technique which keeps the advantages of manual photogrammetric modeling but makes the workflow more convenient for the user. We combine the expertise of the user with automatic operations in photogrammetry and image processing. This will take workload off the user and increase the stability and accuracy of the reconstruction system. Figure 1 shows an input image and the result of our modeling system for a set of oriented images.

The application of multi-view reconstruction methods often depends on the input images. Therefore, we provide a set of different methods which can be used together by the user. If automatic approaches fail, it is important to have a fallback alternative, even if it requires more input from the user. The goal of our application is to make the workflow fast if it is possible, but it has to be successful in any event.

### 1.1 Problem statement

The computation of camera poses requires correspondences of points, lines or patches across multiple images. Industrial sites often contain objects with highly reflective surfaces or poorly textured objects. Bad lighting conditions or taking photos with a flash should not affect the camera orientation. This makes automatic feature detection and matching very difficult because the scene does not provide enough view-invariant patches. Automatic matching is further complicated if only a few images should be sufficient for the reconstruction and therefore the camera baselines are very wide.

If the accuracy of the reconstructed objects is very important a total station can be used for measuring several points in the scene. The survey points provide very accurate metric information about the scene. They should be integrated into the image-based tools and used for calculating the camera poses and 3D models. The image-based techniques further provide a control mechanism for the survey data. If a lot of measurements are done with a total station, it can easily happen that the user accidentally selects a wrong corner of an object. Such mistakes will show up with high re-projection

*reisner-kollmann@vrvis.at
†fuhrmann@vrvis.at

errors in the camera orientation process.

The workflow at the actual scene should be kept very simple and time efficient in order to keep costs low. A common approach is that the reconstruction experts measure a few important points with a total station and take photographs. The interactive calculation of the camera orientations shows if enough points and images are available. Additional photographs can be taken by non-experts. These are necessary if some details needed for modeling are not available in the existing photographs. Another possibility is that small changes occurred at the scene and should be incorporated into the 3D reconstruction.

A common goal of multi-view reconstruction is to get information about the size and general shape of industrial machines. This information is used for calculating the required space for machines as well as for visualizing rebuilding plans. Small details of the facilities are often not important and sometimes even disturbing for planning the arrangement of machines. Deciding which objects are important strongly depends on the reconstruction goals and therefore should be left to the user.

Individual elements of artificial objects are often quite simple and can be approximated with geometric primitives like cubes or cylinders. These objects should be reconstructed as simple as possible because dense vertex geometries are often not desired in the CAD-workflow. Small elements, e.g. small switches, usually can be included only with texturing in order to obtain the simplicity of the models. On the other hand, if they are important, the user should be able to explicitly reconstruct these small objects.

Another common type of objects in factories are tubes and pipes. It is hard to automatically match features of tubes across multiple views because they often have very homogeneous surfaces. Just like other objects, the geometry of tubes should be kept as simple as possible.

The following list summarizes the requirements of our image-based modeling system:

- simple workflow on-site
- wide camera baselines
- objects may contain reflective surfaces
- should work with textureless objects
- independent from lighting conditions
- fast and accurate camera orientation
- integration of survey points
- models with simple geometry
- intuitive user interface

The workflow of the reconstruction system can be divided into camera orientation, image segmentation and image-based modeling. Camera orientation uses coded markers in order to achieve short computation time and accurate results. Detailed information about camera orientation can be found in Section 2. The next step is to segment regions and edges of objects or object parts in multiple images with interactive image segmentation techniques. On the basis of oriented cameras and image segmentations it is possible to fit parametric models to the scene. The user selects an appropriate model type, multiple image segmentations and optionally additional constraints and the model parameters are automatically fit to these requirements. Complex objects can be modelled by fitting primitives to individual parts of the object. Section 3 describes the model fitting process in detail.

### 1.2 Related work

A similar modeling approach has been used for reconstructing objects from single images [Lowe 1987]. For a set of known correspondences between points on a 3D object and 2D points in the image, the model and projection parameters are estimated with an iterative optimization technique. The parameters include translation and rotation of the model and focal length of the camera. Edges and line segments are detected automatically in the images. Multiple initial matches between 3D and 2D points are created and verified between objects and line segments grouped by collinearity or parallelism.

The Façade [Debevec et al. 1996] system uses geometric primitives for image-based modeling with multiple views. The user defines line segments in the images and manually links them to the corresponding edge of a model. The parameterized models are then fitted to these correspondences with non-linear minimization.

A system for interactively reconstructing scenes by fitting models to multiple views has been presented by [van den Hengel et al. 2006]. The basic model is a bounded plane from which more complex models can be constructed. The models are fitted to a dense point cloud which is created during the structure-and-motion calculation. Therefore, short camera baselines and diffuse surfaces are needed. The interactive modeling takes advantage of regular scenes by replicating already optimized models to other positions.

[Sinha et al. 2008] present an interactive modeling system for architectural scenes which is supported by automatically detected vanishing lines. Global directions are extracted from the vanishing points in multiple images. The user outlines planar polygons in images whose pose in 3D is estimated by 3D points obtained during structure-from-motion calculations. Line segments near a global direction are snapped to exactly be parallel to the direction.

## 2 Camera orientation

An accurate camera orientation is important for successive reconstruction steps because errors and inaccuracies will propagate through the reconstruction pipeline. Additionally, the calculation of extrinsic camera parameters has to be fast in order to get the orientation results on-site. If additional photographs are needed they should be taken as soon as possible. Longer time steps between taking photographs increase the possibility of undesired changes in the scene.

We use pre-calibrated cameras in order to increase the accuracy and stability of the camera orientation calculations. Coded markers are used for localizing a planar calibration pattern in the images. The correspondences between 2D and 3D positions are used for determining the intrinsic camera parameters as well as the radial and tangential distortion parameters.

We use point-to-point correspondences for calculating the extrinsic camera parameters which are position and orientation. Coded markers in the scene make the matching of points across multiple views very accurate and simple. Points can be manually marked in the images if coded markers cannot be placed in the scene. If the objects in the scene are textured and they have diffuse surfaces, it is possible to automatically match natural features, e.g. with SIFT points [Lowe 2004]. Of course, coded markers, manual points and automatic feature points can also be used together. In this case the contribution of coded markers is weighted higher because they are less error-prone than other points.

We first describe the detection of coded markers in the images and then how the point-to-point matches are used for the camera pose estimation.

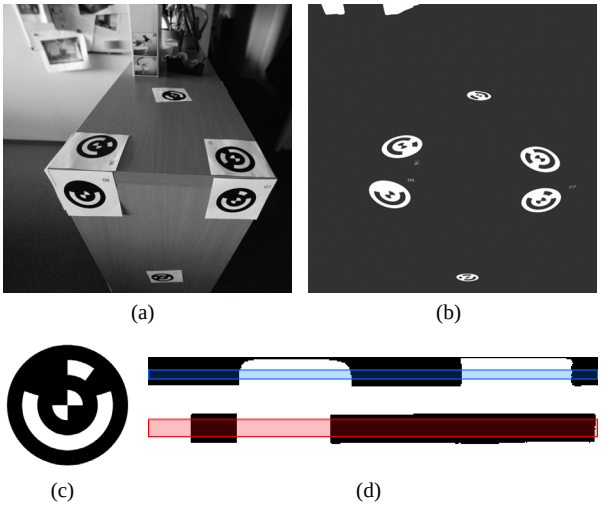



**Figure 2:** *Detecting coded markers in an image. (a) Input image for marker detection. (b) Detected MSERs. Only regions with the shape of an ellipse will be used. (c) Original marker which is placed in the scene. (d) Unwrapped marker: The top section contains the x-corner in the center. The bottom section contains the visual bit code of the marker.*

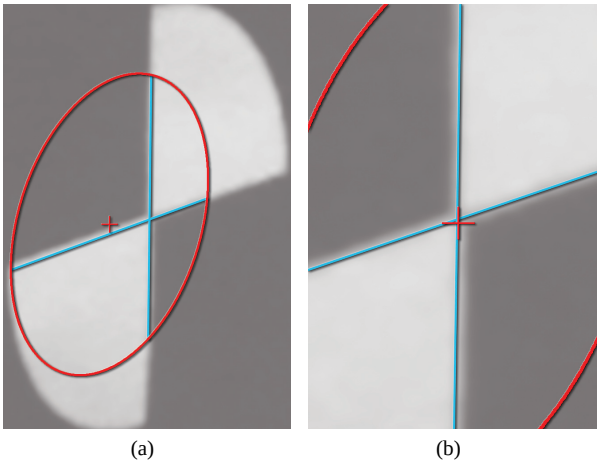



**Figure 3:** *Correction of marker position. (a) First iteration of the algorithm. The original position (center of the ellipse) is far from the actual center of the checkerboard. (b) Second iteration: The line crossing from the first iteration is used as center of the ellipse. It is already very near to the actual center.*

### 2.1 Coded markers

The subpixel-accurate position of coded markers can be easily detected in images, despite low resolution or bad lighting conditions. Each marker provides a visual bit code which can be assigned to a unique id. Therefore it is very easy and robust to track a marker across multiple images.

Figure 2(c) shows an example of our coded markers. A marker contains a central checkerboard pattern and a rotationally invariant bit code divided into ten black or white sections. In an image the marker appear as connected black region in the shape of an ellipse.

We use MSERs (maximally stable extremal regions) to detect possible marker regions which are significantly darker than the surrounding areas in the image [Nistér and Stewénius 2008]. If multiple regions are overlapping only the one with the highest stability is used. We take the outer contour of each region and try to fit an ellipse to it. Regions are only accepted if their outer shape is close enough to an ellipse (see Figure 2(b)).

The center of the ellipse is only an approximate position of the marker due to perspective distortion. An iterative algorithm is used for detecting the sub-pixel accurate position of the central checkerboard pattern, starting with the center of the ellipse. An ellipse with the same eccentricity as the outer ellipse is drawn around the current center inside the checkerboard pattern. All crossings from black to white and vice versa along the border of the ellipse are extracted with subpixel-accuracy.

If there are four crossings, the marker position can be refined by taking lines between oppositely positioned crossings. The intersection of the two lines is used as new center. This procedure is done iteratively and the size of the ellipse is reduced at every iteration. The algorithm is stopped after a fixed number of iterations. It converges very fast and the average value of multiple intermediate results is used to avoid influence of noise.

If there are more or less than four crossings the algorithm is stopped. In this case the algorithm probably failed and the original position of the ellipse is used as center. This happens if the original position is located nearer to the outer contour of the checkerboard than to the center of it.

The circular marker is unwrapped to a rectangle in order to retrieve the visual bit code. Figure 2(d) shows an example for an unwrapped marker. The bottom section in red contains the 10-digit visual bit code and the top section in blue corresponds to the central checkerboard. The sections are divided into ten respectively four parts and the average value is taken for each part. The marker is only accepted if the bit code as well as the central checkerboard can be easily identified, that means all values clearly point to zero or to one. The evaluation of the checkboard is an additional check to avoid the false identification of circular structures which are not coded markers.

### 2.2 Pose estimation

Pose estimation calculates initial values for the camera poses and the triangulated positions of all matched image points. We use RANSAC [Fischler and Bolles 1981] in order to detect incorrectly matched points, which is especially necessary for manually added point correspondences. It is important to point out possible mistakes or inaccuracies by the user which easily happen e.g. in the presence of repeating structures. Due to the small number of point correspondences only solutions with very few outliers can be accepted.

We start with two images that have many point correspondences and compute the relative orientation between them. The five point algorithm [Nistér 2004] is used because the images are calibrated. 3D positions of the point correspondences are calculated by triangulation which is the intersection of backprojected rays from the camera centers through the image positions. If the angle between the oriented images is too small, the images are rejected and another image pair is used for creating a camera network. All other images are sequentially added to the network with the three-point algorithm for correspondences between 2D points and known 3D points. If an image cannot be oriented to the existing network, it is put to the end of the unconnected image list. Maybe another image will provide enough 2D-3D correspondences for adding previously

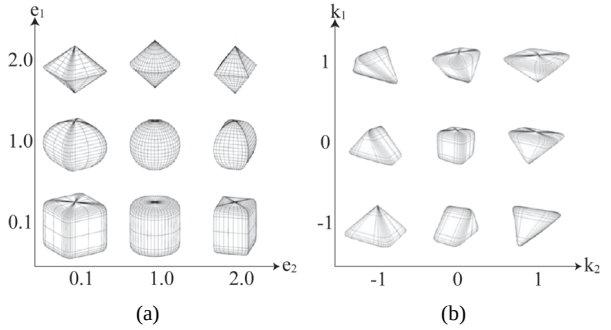



**Figure 4:** *(a) Basic shapes of superellipsoids with uniform scaling. (b) The effect of tapering on a cube-shaped super-ellipsoid.*

rejected images.

The resulting networks are finally optimized with bundle adjustment [Hartley and Zisserman 2004]. The triangulated positions of point correspondences and the camera parameters are optimized with a sparse Levenberg-Marquardt algorithm. The optimization minimizes the distances between reprojected triangulated positions and the according image positions.

The accuracy of pose estimation can be enhanced by using measurements from a total station. The survey points are matched to image points, either manually or the survey points have already been attached to coded markers. The absolute pose of the cameras can be calculated directly with these 2D-3D point correspondences without using the relative orientation between two starting cameras.

# 3 Model fitting

Our image-based modeling technique fits parametric models to different constraints that come from image segmentation or other user input. The user creates an object from a specific model type and adds image segmentations and other constraints for optimizing this model. Complex objects are reconstructed by combinining multiple primitives. The connectivity between these primitives can be ensured by applying constraints to multiple models.

We first describe the possible models and the constraints. Then we describe how these elements are combined during model fitting. The last section covers some special considerations for the reconstruction of tubes.

## 3.1 Models

Many industrial components can be approximated by simple geometric primitives. We provide models for the geometric primitives cube, box, sphere, cylinder, pyramid and frustum. Additional shapes are provided by superellipsoids. More complex objects can often be created by combining multiple primitives. A special case are tubes which are represented by 3D spline.

All models are defined by a set of parameters which are modified during the model fitting process. The parameters include the pose of the model in world space and the actual shape of the model. The pose in world space is defined by position, scale and rotation, each represented by three parameters. The scale and rotation of some models can be defined with less parameters. Table 1 summarizes the number of parameters for each model type.

Superellipsoids [Jaklič et al. 2000] can be used for modeling a wide range of shapes including spheres, cubes, cylinders as well as shapes in between and non-uniform scaled versions. Superellipsoids contain two parameters $e_1$, $e_2$ in addition to the common parameters position, scale and orientation. A global deformation transformation called tapering [Barr 1984] is used to represent even more shapes including pyramids, cones, frustums of pyramids and cones, wedges and all shapes in between. Tapering needs two additional parameters $k_1$ and $k_2$ which reduce the size of the shape along two directions. Possible shapes formed by superellipsoids can be seen in Figure 4.

Tubes are represented by a 3D spline and a diameter defining its thickness. A 3D curve is interpolated through a set of control points. These control points can be altered during the model fitting process. The number of control points is automatically defined depending on the length of the reprojected curves in the images. More details about the reconstruction of tubes can be found in Section 3.5.

| Name | p | s | r | |
|---|---|---|---|---|
| Cube | 3 | 1 | 3 | 0 |
| Box | 3 | 3 | 3 | 0 |
| Sphere | 3 | 1 | 0 | 0 |
| Cylinder | 3 | 2 | 2 | 0 |
| Pyramid | 3 | 2 | 3 | 0 |
| Frustum | 3 | 2 | 3 | 4 (smaller base position/scale) |
| Superquadric | 3 | 3 | 3 | 4 (coefficients and tapering) |
| Tube | 3 | 1 | 0 | n (control points) |

**Table 1:** *Number of parameters per model type: position p, scale s, rotation r, and additional parameters*

## 3.2 Image constraints

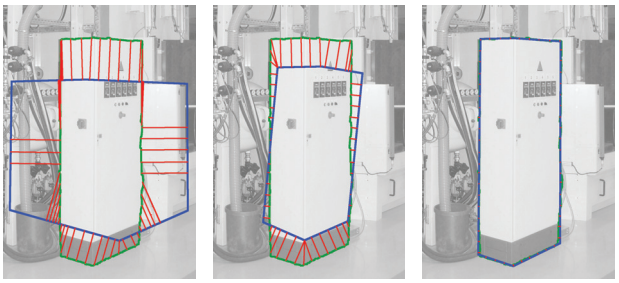
The image projections of a model provide several constraints in order to find the correct parameters of the model. These constraints are created by points, edges and regions from multiple images. Edges and regions are defined by interactive image segmentation, points were earlier used for camera orientation (see Section 2).

We provide a set of interactive segmentation techniques for defining edges and regions. We made positive experiences with interactive graph cuts [Boykov and Jolly 2001] for region-based segmentation and with intelligent scissors [Mortensen and Barrett 1998] for defining edges and contours. Any other segmentation technique can be used as well and depending on the scene it may also be possible to use automatic segmentation techniques.

The goal of segmentation is to define outlines and edges of the desired 3D objects. It is not necessary to define all possible segmentations in one image, but rather to provide enough constraints across multiple images for fitting a model. Not all types of segmentation have to be provided, especially because outlines are often hard to segment due to occlusions from other objects. If the user is not satisfied with a fitted model, it is possible to add more constraints and further optimize the model. Experiments have shown, that rough segmentations in many images lead often to better results than very exact segmentations in only few images.

### 3.2.1 Outlines

Outlines are a very useful constraint for defining the position and scale of an object. The outline of a specific model is computed analytically or it is retrieved with an image-based technique. Sample points are taken from the user-defined contour at equal distances. Using all points of the contour would increase computation time, whereas the result is nearly unaffected because nearby points have almost the same effect for model fitting. On the other hand, using



(a) (b) (c)

**Figure 5:** *Optimization of an outline constraint. The distances between samples (red) of the user-defined outline (green) and the model outline (blue) are minimized during optimization.*

not enough samples increases the influence of outliers in the user-defined region.

The distances between sample points and the nearest points on the model projection contour are minimized with

$$\min \sum_I \sum_p d_2(p, outline(I))^2,$$

where $d_2$ is the Euclidean 2D image-based distance, $p$ denotes a sample point on the user-defined contour and $outline$ calculates the outline of a model for a specified image $I$. Figure 5 shows the distances at various minimization iterations.

#### 3.2.2 Edges

Edges either correspond to the projections of model edges or to the outline of a model in an image. It does not matter if only parts of an edge are segmented, or if multiple edges are segmented at once. Minimization is done in a similar way as for outlines. All visible edges of the model as well as the outline are calculated for the camera of the current image. The distances between sample points of the user-defined edges and the nearest projected object edge are minimized with

$$\min \sum_I \sum_p d_2(p, visEdge(I) \cup outline(I))^2.$$

#### 3.2.3 Points

Triangulated points from point correspondences can be used as constraint for the corners of a model. Corners are often selected as input points for calculating the camera poses because they can be seen and identified easily across multiple views. The 3D positions of the 2D points have already been calculated and optimized by triangulation and bundle adjustment and form a very accurate constraint for the model (see Section 2.2).

Points define a constraint in three-dimensional space in contrast to edges and regions which define constraints in image space. For each 3D point the nearest corner of the model is detected. The model is fitted by minimizing the distance between the points and the model corners with

$$\min \sum_P d_3(P, P_M)^2,$$

where $d_3$ denotes the Euclidean distance in 3D space, $P$ is a triangulated point from image point correspondences and $P_M$ is a corner of the model. Fitting to the nearest model corners can result in an undesired local minimum. Section 3.4 describes the initialization of the models in order to avoid such local minima.

### 3.3 User-defined constraints

The user can provide high-level information in addition to the image segmentations. These constraints form relations between multiple models or they demand certain shape properties of a model. A very simple constraint is to set the parameters of different models to the same value. For example, if two identical objects appear in a scene, they can be modeled by applying this constraint to all parameters except position and orientation.

#### 3.3.1 Up vector

This constraint ensures that objects are oriented in the same direction, but the rotation around this direction can vary. We do not provide a global up-direction in the reconstruction system, because the cameras may be registered to an arbitrary coordinate system. Hence, the up-vector is only defined by the models assigned to this constraint and do not need to point to a certain direction. The constraint minimizes the angle between the up-vectors of all models with each other.

Some models can be parameterized with different direction vectors and keep the same shape. For example, the orientation of a box can be interchanged in all main axes as long as the according scale values are adapted. During initialization we re-parameterize all models so that they have approximately the same direction vector while their shape is not affected.

#### 3.3.2 Ground plane

Objects are often located at the same ground plane. The ground plane is computed for all models assigned to the constraint by taking the lowest point of each model along its up-vector. The ground plane is fitted to these points and updated at every iteration during minimization. The constraint is then satisfied by minimizing the perpendicular distance between the plane and the lowest point of each model along the plane normal.

### 3.4 Initialization and optimization

We use Levenberg-Marquardt [Lourakis 2004] for the nonlinear optimization of several constraints. It is important to find a good initial model as Levenberg-Marquardt only converges to a local minimum. The position of a model can be found quite easily by triangulating the centers of segmentations in every image. For estimating the scale of the object we compute the median distance between the center and the outer contour of all segmentations in an image. This distance is projected back to the model position and the average value from all images is taken as uniform scale value. This is obviously a very rough initialization, especially for elongated objects, but the scale values usually converge very fast. Initializing a model with anisotropic scale values would increase the effort for finding the right model orientation.

The orientation of the object as well as additional parameters are more difficult to define in advance. Therefore a set of models is initialized with random parameters. They are optimized for a few iterations and the best model is selected for further optimization.

We first optimize the individual cost functions from image segmentations with Levenberg-Marquardt which are independent for each

(a) (b) (c)

**Figure 6:** *Segmentation of tubes: (a) user-drawn curve, (b) region with skeleton, (c) two edges and intermediate curve*

model. Then the models are re-parameterized if it is necessary by constraints that define relations between multiple models. The last step is to fit all models simultaneously to all constraints.

The previous sections described the individual cost functions, which either come from image segmentations or from user-defined constraints. The goal of optimization is to minimize the sum of all individual costs. All constraints serve as soft constraints, that means solutions slightly off the constraint are also accepted. Weights can be assigned to constraints in order to increase their importance compared to others.

Models with many parameters, e.g. superellipsoids, usually require more constraints, i.e. more image segmentations, than models with only a few parameters. Models which depend highly on a specific rotation often need more initial guesses and therefore longer computation time.

### 3.5 Reconstruction of tubes

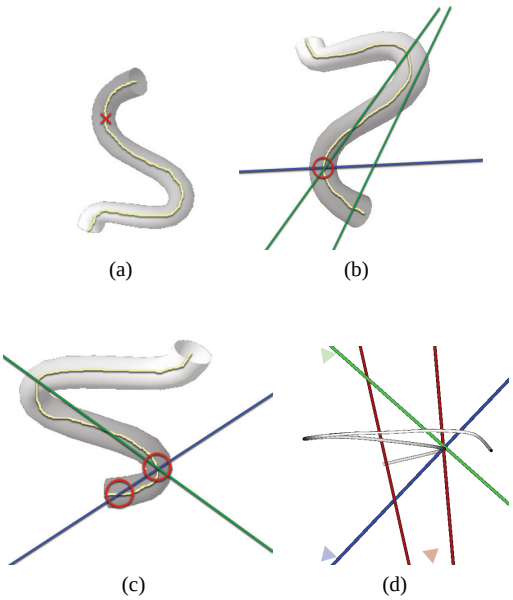
Tubes are represented by the control points of a 3D spline and a diameter for its width. The user input and the initialization of the models are different to other models. Once the initial parameters are available for the tube, optimization is done as for all other models.

There are three possibilities for defining image constraints for a tube. All methods lead to a 2D curve in the center of the tube and a value defining the width of the tube at a certain 2D position (see Figure 6).

- One user-drawn curve in the center of the tube and one line specifying the width of the tube.
- A region which corresponds to the outline of the tube. It is also possible to segment only a part of the tube, if it is partially occluded. The skeleton of the region is calculated and smoothed by sequential thinning with structuring elements [Sonka et al. 2007, p. 675]. The width of the tube is determined by the distance between the skeleton and the region border.
- Two edges along the outlines of the tube. The edges do not have to denote the whole tube, but the two edges should approximately mark the same segments of the tube. The central curve is approximated in the center of the input edges.

Initial 3D positions on the tube are created by triangulating point correspondences across multiple views. Tubes often do not provide any image features which could be used for matching points. Therefore, initial matches are created solely based on the input 2D curve and epipolar geometry.

A corresponding point in another image has to lie on the crossing of the 3D-projection of the tube and the epipolar line. If the epipolar line meets the 2D curve in another image only once, the probability



(a) (b) (c) (d)

**Figure 7:** *Initialization of tubes: (a) Sample point in image 1. (b) Image 2: The blue epipolar line from image 1 led to one corresponding point. This point is confirmed by a green epipolar line from image 3. (c) Image 3: The blue epipolar line from the first image led to two corresponding points, but the green epipolar line from image 2 validated only one point. (d) Reconstructed tube and epipolar lines from another viewpoint. The blue line originates from image 1, the green from image 2 and the red lines from image 3. It can be clearly seen that the intersection of the blue and red line is not located on the 3D curve.*

for a correct point match is very high. Of course, this is only true if the whole tube is visible and has been segmented by the user.

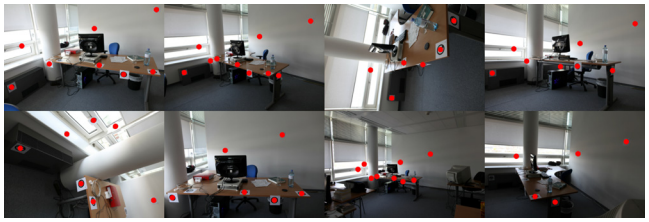
If the epipolar line crosses the 2D curve in more than one point, it is not possible to determine the correct match if there are only two images. If there are more images, all possibly correspondending points in all other images are computed. The corresponding points are validated by computing their epipolar line for every other image. Corresponding points are only accepted, if the epipolar lines from points from different images meet in the same point. This procedure is visualized in Figure 7.

It can happen that points are incorrectly matched, especially if the tube is only segmented in two images. Points which are neighbored in one image, are probably also neighbored in another image. Some points may be missing in another image, but in any case, a sequence of points must appear in the same order in all image. Points which occur in different sequences in different images are removed. The last step is to fill in additional points into large gaps along the 2D curves. The neighboring points provide clues for deciding between possible corresponding points. Finally the control points are re-parameterized at equal distances in 3D space. Figures 10(e) - 10(h) show the reconstruction of a tube.

## 4 Results

Figure 8 shows the input and result of camera orientation for images of an office room. We used eight images with 21 megapixels each. The detection and matching of coded markers needed 118 seconds[1],

[1] All tests were done on an Intel Q6600 2.4 GHz processor



(a)

(b)

**Figure 8:** *Camera orientation with coded markers. (a) Input images with detected markers (red points). (b) Oriented cameras and triangulated marker positions viewed from top of the scene.*

calculating camera poses and bundle adjustment was done in 2.4 seconds. In total, 266 MSERs were detected, from which 75 ellipses and 58 valid coded markers were extracted. 13 unique coded markers were detected across all images, from which 11 3D positions were triangulated during pose estimation. Two markers could not be triangulated because they were only visible in one image.

Figure 10 shows image segmentations and results for single models. An example where multiple models were used for reconstructing a scene can be seen in Figure 9. The scene was modeled with 19 primitives based on 9 images. The optimization contained 376 parameters and 21075 measurements from image segmentations and other constraints. The optimization of all models, i.e. the initialization and individual optimization of each model and the combined optimization of all models, was done in 9 minutes.

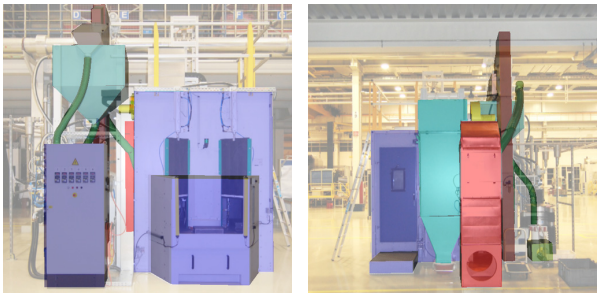
## 5 Conclusion and future work

We presented a new reconstruction system for modeling industrial facilities from a set of images. The system is easy to use and does not require any knowledge about 3D modeling or photogrammetry. Special solutions were presented for handling difficulties which arise in industrial environments.

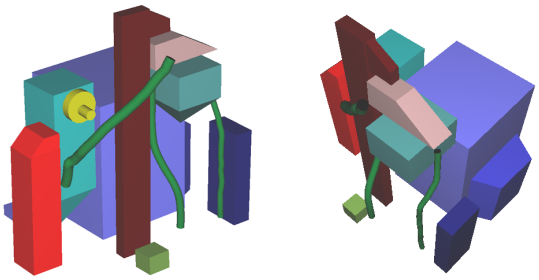
The calculation of accurate camera parameters and 3D points in the scene is important for determining the size of objects and for providing exact input information for the model fitting process. The camera orientation is supported by coded markers and survey points from a total station.

We use parametric models for reconstructing a scene with simple geometry. These models are either geometric primitives, superquadrics or tubes. The models are fitted to image constraints which correspond to outlines, edges or corners of the model and to user-defined constraints that contain high-level information about the scene.

For future work we will support the user during image segmentation. This part is the most time-consuming task for the user in the current system. We will propagate segmentations to other views with the help of photogrammetry and image processing techniques.



(a) (b)



(c) (d)

**Figure 9:** *This scene was reconstructed with 19 models based on segmentations in 9 images.*

## Acknowledgements

This work has been funded by Austrian Research Society (FFG) under the project *Reconstruction for Integrated Facility Planning* (FFG Basisprogramm 818114).

Special thanks to Michael Jenewein of vra Ziviltechniker GmbH.

## References

BARR, A. H. 1984. Global and local deformations of solid primitives. In *SIGGRAPH '84*, ACM, New York, NY, USA, 21–30.

BOYKOV, Y. Y., AND JOLLY, M.-P. 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *ICCV '01*, vol. 1, 105–112.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *SIGGRAPH '96*, 11–20.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6, 381–395.

FURUKAWA, Y., AND PONCE, J. 2007. Accurate, dense, and robust multi-view stereopsis. In *CVPR '07*, 1–8.

HARTLEY, R. I., AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518.

IMAGEMODELER. `http://www.imagemodeler.com`.

JAKLIČ, A., LEONARDIS, A., AND SOLINA, F. 2000. *Segmentation and recovery of superquadrics: computational imaging and vision*. Kluwer Academic Publishers, Norwell, MA, USA.

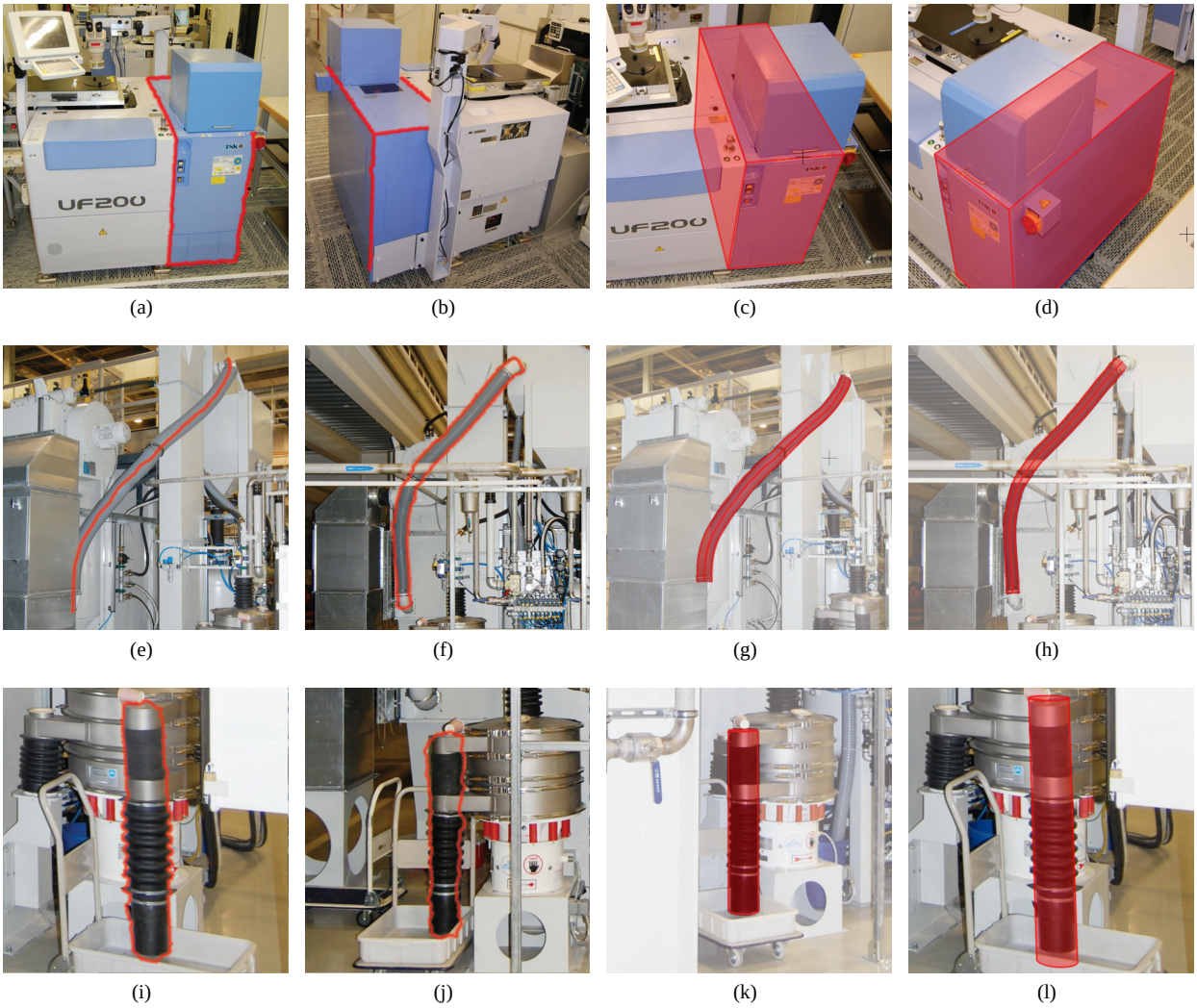



**Figure 10:** *(a) - (d) Edges in two images are used for modeling a box. The image in (a) provided 219 and the image in (b) 165 sample points. (e) - (h) A tube with 30 control points is modeled by an edge (181 sample points) and a region (277 sample points) in two images. (i) - (l) Outlines in two images with 106 and 126 sample points are used for modeling a cylinder.*


LOURAKIS, M., 2004. levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. `http://www.ics.forth.gr/~lourakis/levmar/`. Accessed on 24 Feb. 2010.

LOWE, D. G. 1987. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence 31*, 3, 355–395.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 91–110.

MORTENSEN, E. N., AND BARRETT, W. A. 1998. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing 60*, 349–384.

NISTÉR, D., AND STEWÉNIUS, H. 2008. Linear time maximally stable extremal regions. In *ECCV '08*, Springer-Verlag, Berlin, Heidelberg, 183–196.

NISTÉR, D. 2004. An efficient solution to the five-point relative pose problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence 26*, 6, 756–770.

PHOTOMODELER. `http://www.photomodeler.com`.

SINHA, S. N., STEEDLY, D., SZELISKI, R., AGRAWALA, M., AND POLLEFEYS, M. 2008. Interactive 3d architectural modeling from unordered photo collections. *ACM Trans. Graph. 27*, 5, 1–10.

SONKA, M., HLAVAC, V., AND BOYLE, R. 2007. *Image Processing, Analysis, and Machine Vision*, third ed. Thomson-Engineering.

VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. S. 2006. Building models of regular scenes from structure-and-motion. In *BMVC '06*, 197–206.