How can we exploit typical architectural structures to improve model recovery?

Petko Faber and Bob Fisher
Division of Informatics, University of Edinburgh,
Edinburgh, EH1 2QL, UK
npf|rbf@dai.ed.ac.uk

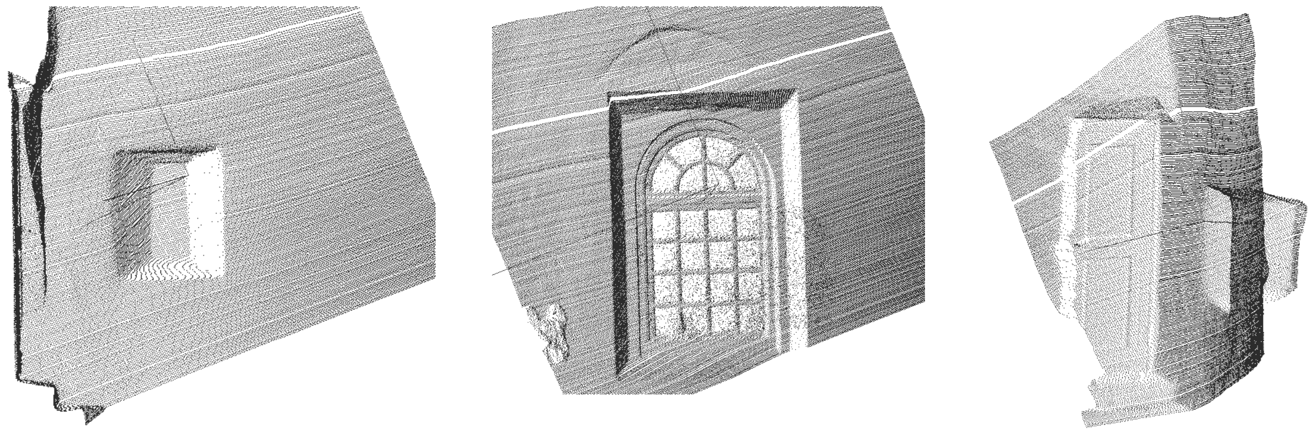


Figure 1. Examples from the Bornholm data set.

Abstract

In this work we address the question of how to exploit typical architectural structures to improve accuracy for CAD modeling of built environments from 3D data. In doing so we have examined the applicability of the GENOCOP III *algorithm to the model fitting. The algorithm uses explicit domain knowledge, specifically geometric constraints in the form of parameterized surface models and an Euclidean fitting of geometric primitives that describe the parameterized volumetric models. Beside some results fitting parameterized volumetric models to real 3D datasets, example times for convergence and comparison with known ground truth are given.*

1 Introduction

In many areas of engineering, medical sciences or art there is a strong demand to create an appropriate computer representation of existing 3D objects from huge sets of measured data points. While this topic has been an important branch in computer vision, due to the advances in laser scanning, now it has become a realistic expectation in the geometric modelling community to generate accurate and topologically consistent models, ready to be used in CAD/CAM.

Typical applications include reproducing and redesigning objects or only parts, when no original drawings or documentation are available. The modeling of real world environments covers objects such as buildings, industrial plants and historical buildings like churches, castles, and towers where it is hard to get data for the whole scene.

In areas where the design is of utmost importance such as in the automobile industry or in making of prostheses for human body parts a free-form modeling is required. However, the shape of the objects we are dealing with here is not arbitrary, rather it conforms to architectural coventions expressed through geometrical and topological relations between feature primitives. Usually, man-made objects can be represented by a set of relatively simple geometric shapes with characteristic features. For example doorways have a standard range of sizes and the vertical sides are usually nearly parallel, windows are often included in walls and arranged in a row, walls interact with specific angles, pipes have specific (relative) orientations in industrial factories etc. These properties are formulated from specific domain knowledge. But what is needed to preserve these specific instances of the properties in the reconstructed models?

This paper focuses on the use of constraints, rather then the selection of constraints. We assume that the appropriate constraints are chosen already (by hand here). The main issues are how to incorporate these constraints into the modeling process and what strategy should be used for managing these constraints.

Constraints can be used as a-priori information to reduce the search space between, for example, the model features and the extracted features, e.g. [2, 12, 13]. Constraint-based optimization algorithms have been examined in numerous applications, e.g. [4, 7, 17, 21]. A review of the main research in the CAD community as well as in the vision community revealed that the exploitation of geometric constraints has not been fully investigated [20]. Nevertheless, the application of classical constraint-based algorithms requires a reasonable initialization to guarantee convergence and it is hard to formulate objective function and constraints especially when variable models in an unknown environment are used. These characteristics make classical methods not appropriate and favour evolutionary algorithms. The popular argument of the time-consuming characteristic of evolutionary algorithms takes only a secondary role in reverse engineering where real time is not required and becomes less important due to rapidly increasing machine speed.

Fig. 1 shows three subsets from a real 3D data set[1]. It is normally a hard problem to segment 3D data interactively or semi-automatically because the segmentation boundaries can be influenced by sparse data, noise from inter-reflections, data from large slants, mixed data points from edges and small features compared to the laser footprint. Additionally the geometry of extracted shapes tend to be distorted due to misregistration errors early in the processing.

Considering the characteristics mentioned before we assume that a conventional segmentation will not work well enough to ensure a reliable space recovery. In this paper we describe a new approach to exploit constraints using parameterized surface models. We use an evolutionary algorithm to extract a good representation for typical architectural features in noisy 3D data subset such as from the Bornholm data set.

2 Model fitting

Model fitting is a common subproblem in computer vision and can be described as follows: Given a set of 3D data points p_i, $i \in [1, n]$, probably belonging to the same object, find the model that approximates the 3D data best. The literature on fitting 3D data points to parametric models is extensive and is only briefly discussed. An early paper concerning model fitting in 3D data [16] is focused on the extraction of edges and regions to retrieve a set of models from an object-model library that are similar to the objects in the 3D data. The main concern in [11] was obstacle avoidance for autonomous vehicle navigation, whereby a simple hierarchical model fitting was applied. In [3] an approach to range image object recognition was proposed that combines surfaces, edges, and points that have been estimated independently.

Focusing on fitting of 3D data to complex 3D models is of utmost importance because accurate 3D models for built environments are a key problem in computer vision. Consequently, some requirements that a model fitting algorithm should satisfy are:

1. The algorithm should produce accurate and repeatable results.

2. The results have to be robust with respect to outliers in the input data.

3. The algorithm must be able to deal with an unknown number of models.

While in [6] a framework for the fitting of multiple parametric models is presented, in [18] the possibility to extract parametric models from poor quality 3D data was examined.

[1] The range image was taken from a 12th century round church on Bornholm Island (Danish, off the southern coast of Sweden).

But, because parameter estimation and data classification influence each other it is hard to meet all requirements in one go. If the classification problem is solved so that we know which data points support which model, parameter estimation is easy. The other way round, if a good parameter estimation is given, the data can be classified straightforward. This means, we need a *classify-while-fit* algorithm. Fortunately, the model structure is a-priori known.

2.1 Parametrized surface models

One of the most useful building representations is the parametrical model. They enable a fixed formulation of building-specific geometrical and topological constraints. Different geometric instances of the objects can be simply produced by variation of the parameters. Parametric models are particularly suitable for interactive systems with models selected from a data base and parameters changed interactively. Nonetheless, the use of parametric models is limited if the number of possible specific models becomes large. In this case generic modeling forms are to be preferred.

Parameterized surface models were employed by, among others [5] and [19]. 3D objects with complex topology can be modeled by connecting parameterized 3D primitives in a generic way. There, each primitive is described by its own set of shapes as well as position parameter. Here the parameterized surface model is defined by three sets: the variable list $\mathcal{V} = \{v_k, k \in [1, l]\}$, the surface set $\mathcal{S} = \{S_j, j \in [1, m]\}$ and the set of local positions $\mathcal{T} = \{T_j, j \in [1, m]\}$.

2.1.1 Surface types

The surface types we investigated here are limited to three primitives: rectangular, (straight circular) cylindrical and (straight circular) conical segments, because the shape of most buildings and industrial factories is not arbitrary and can be described by geometric primitives.

- Rectangular segment: A normalized rectangular segment (Fig. 2) can be simply described by two parameters: width x and height y. Both x and y can be defined by either a constant value or an index into the variable list $\mathcal{V}$.

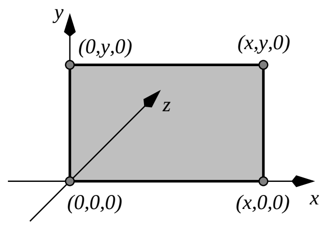


Figure 2. Normalized rectangular segment

- Cylindrical segment: A normalized cylindrical segment (Fig. 3) can be described by three parameters: width x, height y and depth z. All three parameters can be defined by either a constant value or an index into the variable list $\mathcal{V}$.

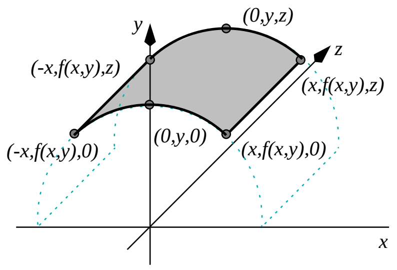


Figure 3. Normalized cylindrical segment

- Conical segment: A normalized cylindrical segment (Fig. 4) can be described by three parameters: width x, height y and depth z. All three parameters can be defined by either a constant value or an index into the variable list $\mathcal{V}$.

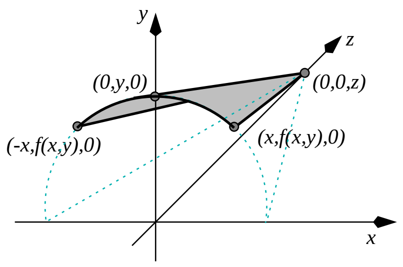


Figure 4. Normalized conical segment

2.1.2 Local position

The set of local positions $\mathcal{T} = \{T_j, j \in [1, m]\}$ describes the geometrical and topological relations between the surfaces $S_j \in \mathcal{S}$, $j \in [1, m]$. Because each surface has six degrees of freedom the local position of T_j is defined by rotation matrix $\mathbf{R}_j \sim (r_x, r_y, r_z)_j$ and translation vector $\mathbf{t}_j \sim (t_x, t_y, t_z)_j^T$.

While the parameters of $\mathbf{R}_j$ are defined a-priori by constant values because we are not allowing arbitrary assemblies, the parameters of $\mathbf{t}_j$ can be defined by either constant values or indices into the variable list $\mathcal{V}$. The parameters of $\mathbf{R}_j$ could be extended to allow a limited use of variables, for example doors would be able to swing and windows can be opened.

2.2 Genocop III

Once appropriate model descriptions are generated the next step is to fit the 3D data to the models with the objective to establish a correspondence between the 3D data and the model(s). This step is strictly model-driven and follows the paradigm of hypothesize and verify. It is performed in two steps: (1) a correspondence is established between the two sets of descriptions and (2) using the established correspondences, a geometrical transformation, usually a rotation matrix and a translation vector, is derived such that the model may transformed to the orientation of the 3D data set. Using a robust penalty-driven constraint approach an accurate parameterization can be generated from degenerated, complex and noisy data.

One method is to use some form of iterative surface growing algorithm, e. g. [3, 10], but this depends on accurate dense data. We use an evolutionary algorithm to fit 3D data to parameterized surface models because we had very fragmentary, sparse and noisy data, as well as unknown scale and data-to-model correspondence. The GENOCOP III algorithm, developed by Michalewicz and Nazhiyath [15], was extended by adding a complex evaluation function. Because the internal working of GENOCOP III is beyond the scope of this paper, it seems sufficient to say that it is an evolutionary optimization algorithm that aims at finding a global optimum (minimum or maximum) of a function. Additional constraints may be specified, as well as domain constraints. For a detailed description of the algorithm and the functionality we refer to [14, 15].

Because all linear and non-linear constraints available can be defined within GENOCOP III linked with the used parametric models, only constraints on the range of values for the parameters are defined and we can apply the GENOCOP III algorithm straightforwardly[2]. Accordingly, we focused on the two aspects:

1. How should a chromosome be formulated?
2. How should the evaluation function be defined?

2.2.1 Chromosome

In standard genetic algorithms a binary encoding forms the chromosomes, but in an evolutionary algorithm a chromosome is set of real-valued concatenated genes. Thus, the parameterized surface model is represented by the list of model parameters $\mathcal{V} = \{v_k, k \in [1, l]\}$. Additionally the parameters of the global transformation Σ (rotation by $\Phi \sim \{\varphi_x, \varphi_y, \varphi_z\}$ and translation by $\Delta \sim \{\delta_x, \delta_y, \delta_z\}$) have to be estimated. The chromosome $\mathcal{C}$ consists finally of genes for Σ and $\mathcal{V}$.

φ_x	φ_y	φ_z	δ_x	δ_y	δ_z	v_1	v_2	$\ldots$	v_n

To perform the nonlinear optimization an initial population is required that forms the basis for the reference and search points that are then mutated. Designing a chromosome that fulfils the domain and relational constraints as well as builds an appropriate representation of the individual, concatenated genes can be complicated especially if we have a complex solution with complex constraints. Fortunately, when a parametric representation is used, generating the initial population becomes simple.

To narrow the search space for the genes and of course to improve the pose invariance of the fittings we *normalize* the 3D data in terms of general moments to affine transformations. In particular, we normalize by translating the data to the centroid, rotating the principal axes of the data to align with the coordinate axes, and applying anisotropic scaling to fit in the unit cube. Thus the domain for the global translation and the model parameters is defined: $\varphi \in [-\pi/2, +\pi/2]$, $\delta \in [-1, +1]$, and $v_k \in [0, +1]$. Considering the domains the initial population can be either a randomly choosen population or a single point where all individuals in the initial population are identical.

[2] The original GENOCOP III package was used that is available by `http://www.coe.uncc.edu/~zbyszek/gchome.html`. The evaluation function provided was completely re-defined as well as the domain and constraint specifications were re-adjusted.

2.2.2 Evaluation function

The evaluation function $\mathcal{E}$ used in our application is more complex than the original GENOCOP III algorithm. It is based on the Euclidean distance of each data point p_i to the closest surface s_j of the 3D model instance that is defined by the chromosome. The objective is to minimize

$$\mathcal{E} = f_\tau(D) \tag{1}$$

where $f_\tau(D)$ is the squared sum of the distance of the $\tau\%$ of the points that are closest to each $s_j \in \mathcal{S}$. We use a percentage (typically $\tau = 0.95\%$) to provide a robust measure against outliers, noise and incorrect surface assignments. Let d_{ij} be the distance $d(p_i, s_j)$ of the point p_i, $i \in [1, n]$ to the surfacce s_j, $j \in [1, m]$. Let $q_i = \text{argmin}_j d(p_i, s_j)$ be the closest surface. Let $d_i = d(p_i, s_{q_i})$ be the closest distance. Then

$$D = \{(d_i, q_i), i \in [1, n]\} \tag{2}$$

is the *best* distance and surface match for each point p_i.

The estimation of d_i for rectangular segment is trivial. The estimation of d_i for the other surface types maybe complicated and often instead of the real Euclidean distance an approximation is used. A closed form expression exists for the Euclidean distance from a point p_i to each of the surface types used here. For a detailed description on how to estimate the Euclidean distance between a point p_i and a surface s_j in closed form we refer to [8]. If all d_i's corresponding to all s_j's are estimated the evaluation of the generated chromosomes becomes then relatively simple although computationally slow. The pseudocode for the evaluation algorithm follows:

```
∀ $p_i$,  $i \in [1, n]$
    Find closest surface $s_j$.
    Estimate Euclidean distance $d_i$.
∀ $s_j \in \mathcal{S}$,  $j \in [1, m]$
    Select $\tau\%$ smallest distances.
Compute $f_\tau(D)$.
```

3 Example: Doorway

In the previous section we described a) how to design an appropriate representation for the model and b) how to apply the nonlinear optimization algorithm GENOCOP III. In this section we summarize empirical testing of the proposed model fitting in terms of efficiency, correctness and robustness for both simulated and real data.

3.1 Parameterized surface doorway model

The two different parameterized surface models used to fit simulated 3D data are sketched in Fig. 5. They can be defined in terms of

- shape variables $\mathcal{V} = \{v_k, k \in [1, l]\}$,
- surfaces $\mathcal{S} = \{s_j, j \in [1, m]\}$ and
- transformations $\mathcal{T} = \{T_j, j \in [1, m]\}$

as follows:

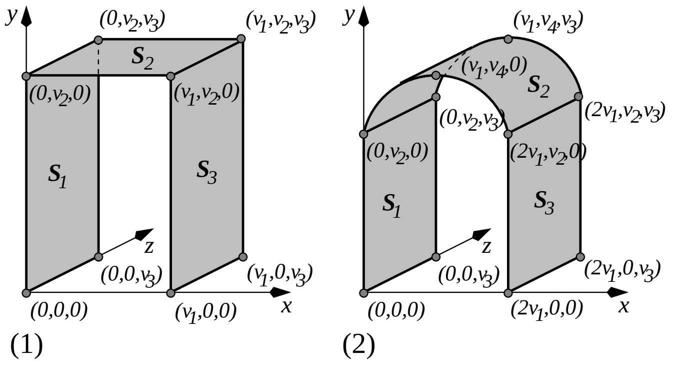


Figure 5. Parameterized surface doorway models.

Model (1) consists of the rectangular surfaces s_j, $j \in [1, 3]$ and can be modeled by the variables v_k, $k \in [1, 3]$ assuming that s_1 and s_3 are parallel and both are perpendicular to s_2.

$$s_1 \sim \begin{pmatrix} v_2 \\ v_3 \end{pmatrix}, \; s_2 \sim \begin{pmatrix} v_1 \\ v_3 \end{pmatrix}, \; s_3 \sim \begin{pmatrix} v_2 \\ v_3 \end{pmatrix}$$

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{t}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{R}_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{t}_2 = \begin{pmatrix} 0 \\ v_2 \\ 0 \end{pmatrix}$$

$$\mathbf{R}_3 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{t}_3 = \begin{pmatrix} v_1 \\ 0 \\ v_3 \end{pmatrix}$$

Model (2) consists of the rectangular surfaces s_j, $j \in \{1,3\}$ and the cylindrical surface s_2. It can be modeled by the variables v_k, $k \in [1,4]$ assuming that s_1 and s_3 are parallel and the orientation vector of s_2 is perpendicular to both orientation vectors of the s_j, $j = \{1,3\}$.

$$s_1 \sim \begin{pmatrix} v_2 \\ v_3 \end{pmatrix}, s_2 \sim \begin{pmatrix} v_1 \\ v_4 - v_2 \\ v_3 \end{pmatrix}, s_3 \sim \begin{pmatrix} v_2 \\ v_3 \end{pmatrix}$$

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{t}_x = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{R}_y = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{t}_y = \begin{pmatrix} v_1 \\ v_4 \\ 0 \end{pmatrix}$$

$$\mathbf{R}_z = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{t}_z = \begin{pmatrix} 2v_1 \\ 0 \\ v_3 \end{pmatrix}$$

3.2 Chromosome representation

We use a real-valued chromosome representation $\mathcal{C} = \{\Sigma, \mathcal{V}\} = \{(\Phi, \Delta), \mathcal{V}\}$ because each element of $\Sigma \subset \mathcal{C}$ corresponds to a transformation parameter and each element of $\mathcal{V} \subset \mathcal{C}$ corresponds to a variable. Thus, $\mathcal{C}$ consists of a part with fixed gene number (6) and a part with variable gene number (card($\mathcal{V}$)).

The chromosome representation for the two used parameterized surface models follows:

Model (1)

φ_x	φ_y	φ_z	δ_x	δ_y	δ_z	v_1	v_2	v_3

Model (2)

φ_x	φ_y	φ_z	δ_x	δ_y	δ_z	v_1	v_2	v_3	v_4

3.3 Model fitting results

This section details some of our experiences fitting parameterized surface models to both simulated and real 3D data sets. We summarize empirical testing of the proposed fitting method in terms of correctness and robustness as well as convergence.

3.3.1 Simulated data

In the case of simulated data we have generated 3D data sets that describe both models. The 3D data consists of up to 5000 3D data points generated by adding isotropic Gaussian noise $N_{\mu,\sigma}$ with $\mu = 0$, $\sigma = \{0.1, 1, 2, 5, 10\}$. In Fig. 6 examples for both model structures with added isotropic Gaussian noise $N_{0,2}$ are visualized.

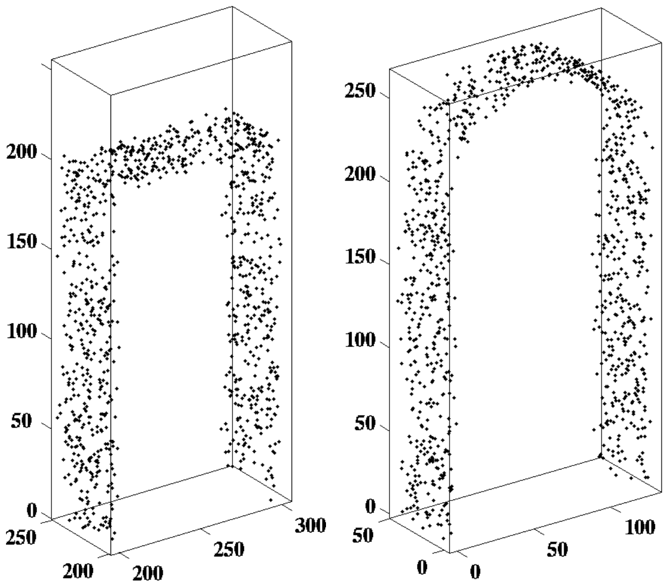


Figure 6. Simulated data with added isotropic Gaussian noise $N_{0,2}$ for both models.

It is obvious that the result of the model fitting should describe the 3D data set by the correct parameter model. That means that it should neither fit a false model structure nor a wrong parameter set $\{\Sigma, \mathcal{V}\}$ to the data. To test the reliability of the fitting result we repeated the experiment 100 times. The result of fitting model (1) is summarized in Fig. 7. As expected the result shows the success of the GENOCOP III algorithm for model fitting. The estimated variable values $v_k \in \mathcal{V}$, $k \in [1,3]$ are a) very stable and b) very close to the true values ($v_1 = 101$, $v_2 = 201.5$, $v_3 = 46.5$). While the distribution of v_1 and v_2 can be approximated by a normal distribution, the distribution of v_3

can be better characterized by an exponential distribution. The reason for this is simply that v_1 and partially v_2 can be estimated directly, but v_3 can be estimated only indirectly when the estimates for v_1 and v_2 are back-projected onto the 3D data. If more complicated parameterized surface models are used then v_3 can be estimated just as v_1 and v_2.

Beside correct fitting results that can be expressed by repeatable results, the result should be robust. Especially, the algorithm must degrade gracefully with increasing noise in the 3D data, with a decrease in the available relevant data, or with an increase in the irrelevant data.

In Fig. 8 the convergence behaviour of the algorithm fitting model (1) with added Gaussian noise $N_{\mu,\sigma}$ with $\mu = 0$, $\sigma = \{0.1, 1, 2, 5, 10\}$ is sketched. As expected, even with an increased noise level the algorithm converges but slower.

The third demand is that a good fitting algorithm has to be as efficient as possible in terms of run time and formal complexity. While the problem of computational cost here is no longer a really hard problem because of the rapidly increasing machine speed, we should guarantee the fitting has acceptable computational cost. Assuming the complexity of the GENOCOP III algorithm is as low as possible, the only part in the algorithm we have actively influenced is the evaluation function. As described in [9] the complexity of the implemented algorithm to estimate the Euclidean distance is low as possible. In Fig. 8 the convergence behaviour of the algorithm fitting model (1) as well as model (2) is sketched. As we can see the algorithm converges very quickly. After maximal 800 generations the fitting result seems stable enough in our experiments that the algorithm can be terminated.

All algorithms have been implemented in C and the computation was performed on a 360 MHz SUN workstation. The average computational costs for the model fitting was about 280 seconds per 1000 3D data points per 10^3 generations with 70 populations.

3.3.2 Real data

To evaluate the algorithm and the proposed parameterized surface models we used real 3D data from the Bornholm data set [1]. The 3D data set, sketched in Fig. 9, has originally $\approx 180k$ 3D data points and was

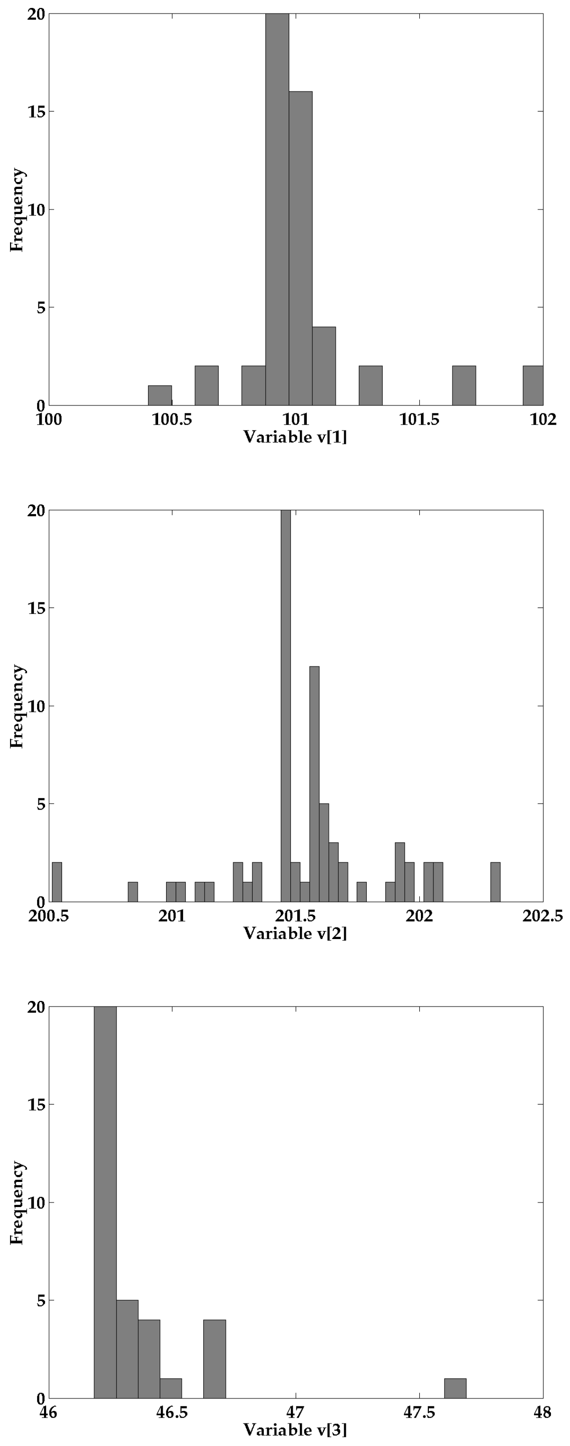
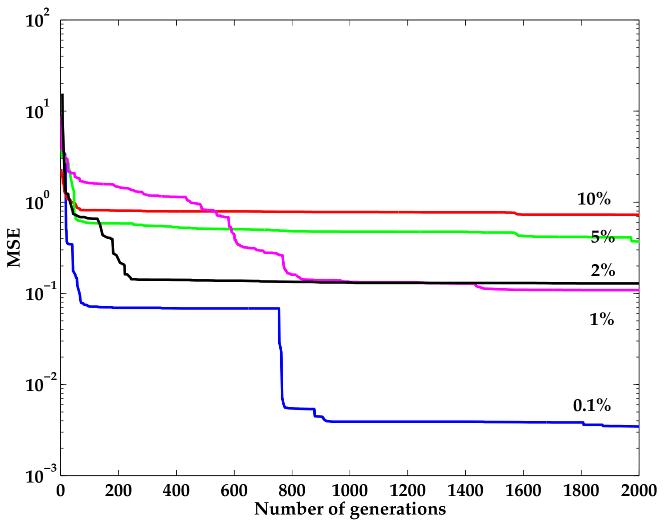


Figure 7. Distribution of variables v_1, v_2 and v_3 fitting model (1). The used histogram contains 100 equally spaced containers, but it was cutted at a frequency of 20. The number of trials was 100.



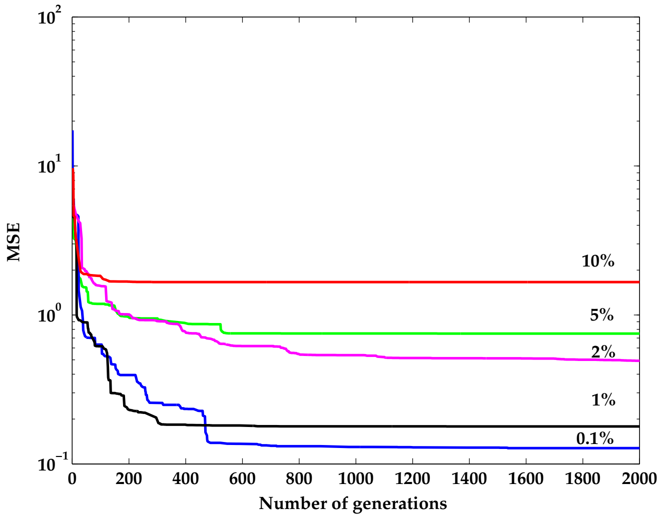


Figure 8. Example times for convergence fitting model (1) respectively model (2) with added isotropic Gaussian noise $N_{\mu,\sigma}$ with $\mu = 0$, $\sigma = \{0.1, 1, 2, 5, 10\}$.

subsampled by factor 100. Because we are interested in doorway fitting we segmented the 3D data set interactively. The final 3D data set used in our experiment consists of $\approx 7k$ 3D data points and is sketched in Fig. 10. As we can see, we have to deal with different problems: the (rectangular) segments are rough (standard deviation $s \approx \pm 5$), for the top as well as the left (rectangular) segment 3D data points are only partially available. Additionally the left segment is described by sparse data. We used model (1) for the parameterization of the 3D data.

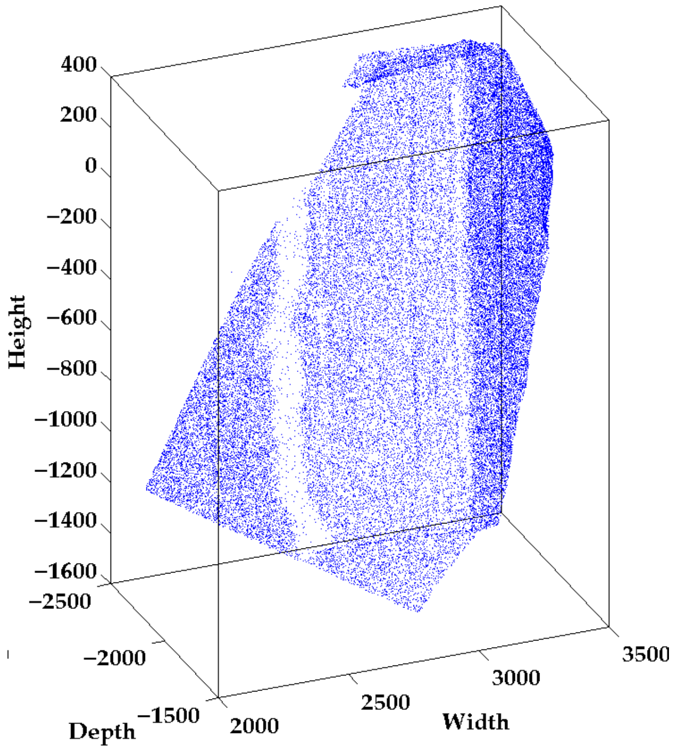


Figure 9. Real 3D data taken form the Bornholm data set. The subset includes parts of two walls, stairs, a doorway and a (closed) door.

As final result we have after 10^4 generation the chromosome $\mathcal{C} = \{0.036, 0.506, -0.989, 0.023, -0.058, 0.966, 0.057, 0.305, 0.808\}$. The RMS error of this fit is $\mu = 0.512$. After re-normalizating the chromosome (see 2.2.1) the parameterization is: $v_1 = 147.715$ [mm] (depth), $v_2 = 1651.17$ (height) and $v_3 = 948.267$ [mm] (width). However, because the ground truth for the example is unknown, we can evaluate the fitting result only subjectively. Therefore, the resulting representation for the doorway was back-projected onto the 3D data in Fig. 11. The estimated variable values seems reliable compared to the 3D data.

4 Conclusions

First and foremost, this paper shows that a parameterized surface model can be successfully invoked in an evolutionary algorithm to approximate 3D data by

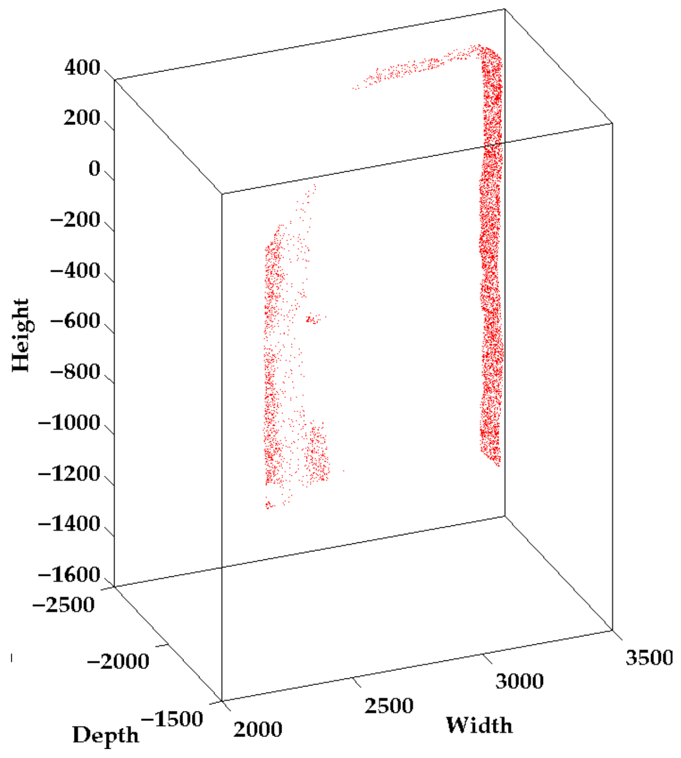


Figure 10. Used 3D data set describing only the doorway in Fig. 9.

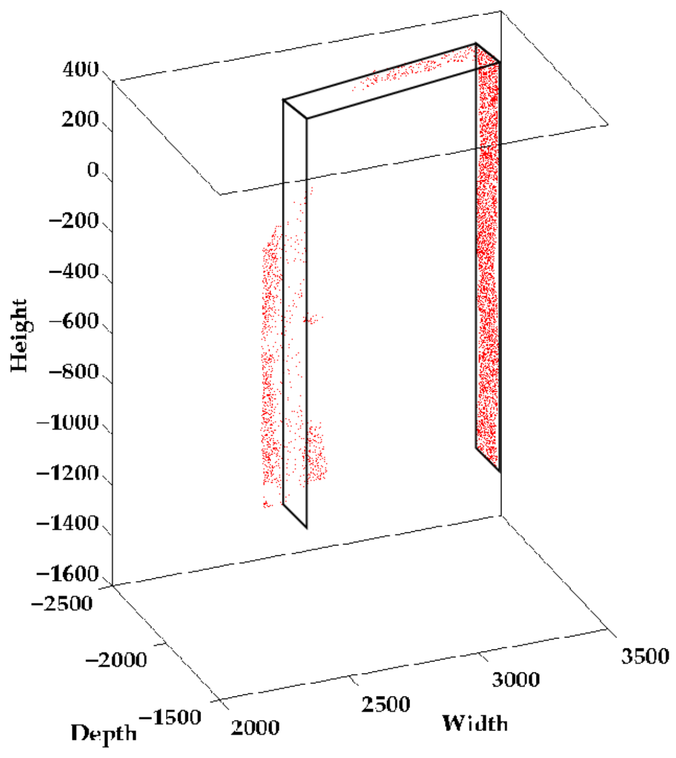


Figure 11. Back-projected result fitting the doorway (Fig. 10).

3D models. All of the experiments in the last section show that architectural conventions given by geometrical and topological relations between feature primitives can be incorporated into the modeling process to improve model accuracy.

The presented algorithm has the advantage of avoiding the problem of constraint formulation for complex 3D models. It also yields in some instances (especially where we have sparse data sets) superior estimations for the transformation parameters and the variables.

Acknowledgement

The work was funded by the CAMERA (*CA*d *M*odelling of Built *E*nvironments from *R*ange *A*nalysis) project, an EC TMR network (ERB FMRX-CT97-0127).

References

[1] http://www.dai.ed.ac.uk/homes/rbf/CAMERA/.

[2] R. Anderl and R. Mendegen. Modelling with constraints: Theoretical foundation and application. *Computer Aided Design*, 28(3):155–168, 1996.

[3] P. J. Besl. *Surfaces in range image understanding*. Springer, Berlin-Heidelberg-New York, 1988.

[4] R. M. Bolle and D. B. Cooper. On optimally combining pieces of information, with application to estimating 3-d complex-object position from range data. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 8(5):619–638, 1986.

[5] C. Brechbuehler, G. Gerig, and O. Kuebler. Parametrisation of closed surfaces for 3-d shape description. *Computer Vision and Image Understanding*, 61(2):154–170, 1995.

[6] G. Danuser and M. Stricker. Parametric model fitting: from inlier characterization to outlier detection. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 20(3):263–280, 1998.

[7] J. De Geeter, H. V. Brussel, J. D. Schutter, and M. Decreton. A smoothly constrained kalman filter. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 19(10):1171–1177, 1997.

[8] P. Faber and R. B. Fisher. Euclidean fitting revisited. In *4th Int.l Workshop on Visual Forms*, pp. 165–175. Springer LNCS 2059, 2001.

[9] P. Faber and R. B. Fisher. Pros and cons of Euclidean fitting. In *23th DAGM Symposium*, pp. 414–420. Springer LNCS 2191, 2001.

[10] R. B. Fisher, A. Fitzgibbon, and D. Eggert. Extracting surface patches from complete range descriptions. In *Proc. Int'l. Conf. Recent Advances in 3-D Imaging and Modeling*, pp. 148–155, 1997.

[11] D. Gennery. Object detection and measurement using stereo vision. In *DARPA'80*, pp. 161–167, 1980.

[12] W. E. Grimson. *The Role of Geometric Constraints*. MIT Press, London, 1990.

[13] D. G. Lowe. Fitting parameterized three dimensional models to images. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991.

[14] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, Berlin-Heidelberg-New York, 3 edition, 1996.

[15] Z. Michalewicz and G. Nazhiyath. Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Proc. 2nd Int'l. Conf. Evolutionary Computation*, pp. 647–651. IEEE, 1995.

[16] R. Nevatia and T. O. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8(1):77–98, 1977.

[17] J. Porill. Optimal combination and constraints for geometrical sensor data. *Int'l. J. of Robotics Research*, 7(6):66–78, 1988.

[18] C. Robertson, R. B. Fisher, N. Werghi, and A. P. Ashbrook. Fitting of constrained feature models to poor 3d data. In *Proc. Adaptive Computing in Design and Manufacture*, pp. 149–160, 2000.

[19] L. Staib and J. S. Duncan. Boundary finding with parametrically deformable models. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 14(11):1061–1075, 1991.

[20] T. Várady, R. R. Martin, and J. Cox. Reverse engineering of geometric models - an introduction. *Computer Aided Design*, 29(4):255–269, 1997.

[21] N. Werghi, R. B. Fisher, C. Robertson, and A. Ashbrook. Object reconstruction by incorporating geometric constraints in reverse engineering. *Computer Aided Design*, 31(6):363–399, 1999.