Online Generation of Dynamic Shape Models for Industrial Optical Quality Control

Lars Dornheim[1] Steffen Sauer[1,2], Erik Trostmann[2], Klaus D. Tönnies[1], and Dirk Berndt[2]

[1] Institut für Simulation und Graphik, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, Germany
[2] Abteilung Intelligente Sensorsysteme, Fraunhofer-Institut für Fabrikbetrieb und -automatisierung Magdeburg, Germany

Abstract. In an industrial optical quality control system the presence and correctness of mounted parts shall be verified. For that purpose, these parts´ current state (given by camera images) is compared to their reference state (given by rendered CAD data images). Thereto, the parts have to be searched and identified in the camera images, which is achieved by a shape-model-based technique, the Stable Mass-Spring Model (SMSM) approach.
We present a new method for the automatic generation of individual dynamic models for the robust search of objects in 2D images. For each object to be searched, an SMSM is constructed automatically from a reference image to represent its characteristic features. Using this model, a search can successfully be performed by a local dynamic fitting technique, even if the object is partly obscured or deformed.
We show that using these models a robust search could be performed in a quality control task from airplane fabrication. We achieve no false negatives and only 16 % false positives in an overall control check test of the presence and correctness of mounted hull parts. No special adaption of the production process was necessary, since our model generation operates online on the existing CAD data.

1 Introduction

Manual quality control in the industry is nowadays often replaced by optical quality control. Generally, this requires a fix and appropriate optical measurement process. But sometimes this cannot be guaranteed. An example for this is the hull fabrication for airplanes, where a lot of different parts have to be mounted in the correct positions and orientation. Airplanes are built in rather small series (e.g. Hundreds), which normally also change slightly within a series. Therefore, specifically taylored, classical systems for optical quality control are not efficient here, as they have to be adapted to every airplane model and all its mounted parts and need to be updated every time the airplane design changes. For these reasons, this kind of quality assurance is currently performed manually.

In our application case, a flexible optical quality control system is needed, which relies only on the current airplane design (e.g. given as complete CAD

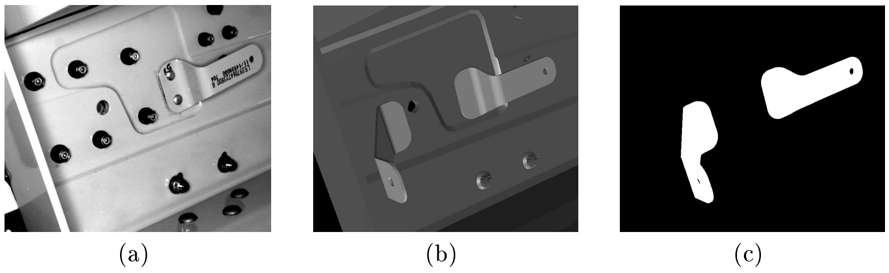


Fig. 1. (a) shows a real photographic image of the current state of a hull part with mounted parts. It can be seen that color infomation is not appropriate to find the mounted parts, but contour information is. **(b)** shows the same view rendered from the according CAD data. It can be seen that the left mounted part is not existant in (a) due to a production failure. **(c)** shows the same view in a binary image, where the mounted parts are marked (white).

data), which is always available. It should take various pictures of the mounted parts with a camera that can navigate through the 3D scene of the airplane hull fabrication. The system should then compare the *current state* (pictures of hull areas with mounted parts, figure 1(a)) with the respective *reference state* (given by a rendering of the CAD data of the hull and the mounted parts, figure 1(b)). This requires a method to detect the mounted parts in the current state images, even if they are slightly displaced. Furthermore, the parts should be identified, if they are distorted (due to image acquisition) or partly hidden (e.g. by labels or tags). This requires a system that:

1. segments the parts, even if they are displaced, distorted or partly occluded (in some degree) and
2. compares the segmented objects to the corresponding parts already marked in the reference state images.

This paper focuses on the first point of these: *part segmentation*. The second point is mostly independent (see [1] for more on that).

2 Related work

For the segmentation of displaced, distorted or partly hidden parts, knowledge about the expected shape is necessary. We employ Stable Mass-Spring Models (SMSMs, [2]), which use prototypical shape knowledge (a deformable sample model) with no further training necessary. This is appropriate for our searching purpose, since we want to segment an individual structure we know and not a class with shape variation. Furthermore, SMSMs perform a local search, starting at a given point, which is also desirable here, as we expect the parts on certain positions (given by the CAD data) and want a local search from there.

A related model-based segmentation technique was successfully used for search of structures before in [3]. The major difference to our application was the fixed and small set of simple objects, so that the search models could easily be constructed by hand. We have a lot of more complex parts to be searched for, that can change after a while in the CAD data, so that the corresponding shape models should be constructed automatically and on demand.

2.1 Model generation

For most applications, dynamic segmentation models are created manually [4]. An automatic generation of 3D models is limited to an initial adapation of a general surface mesh to an object for segmentation or reconstruction like in [5]. So, no model knowledge is incorporated permanently into the model, which could be used for the segmentation of similar structures later on.

Two recent approaches create dynamic surface models ([6] by means of growing cell neural networks and [7] by means of 3D deformable surface meshes) for intermediate use, which are intended to work on binary segmentation masks and not on real data. The emphasis in these two approaches lies on the establishing of point correspondences for statistical segmentation methods.

To our knowledge, the only method that focuses on the construction of explicit deformable models (active contours, mass-spring models, etc.) is introduced in [8]. There, complex medical structures were modeled as SMSM prototypes using connected submodels representing contour and appearance (intensity distribution) information. This approach did not work well for our task, because appearance does not help in our case (see figure 1). As we used totally other types of sensors (no edge and intensity sensors, but sensors for different geometric primitives), it is not clear, how to adapt this method for our task. This is, because sensor distribution and according model construction should not be independent of the used sensor types.

3 Model generation

The model to generate is a 2D Stable Mass-Spring Model (SMSM). It shall be generated automatically from a reference state image, where the part of interest was marked using the CAD data (figure 1(c)). The model uses sensors that are attracted by geometric primitives' attributes (contour normal direction and contour curvature, see [1] for details). These sensors are connected to mass points of the model, so they have to be placed carefully in the model construction process to represent the characteristic object features.

3.1 Contour extraction for the reference object

To generate an SMSM that prototypically models the shape features of an object in a binary marked reference state image, its contour has to be extracted. This is done by a standard contour sampling algorithm (e.g. from [9]) working on binary pixel images and ensures closed contour lines of one pixel width.

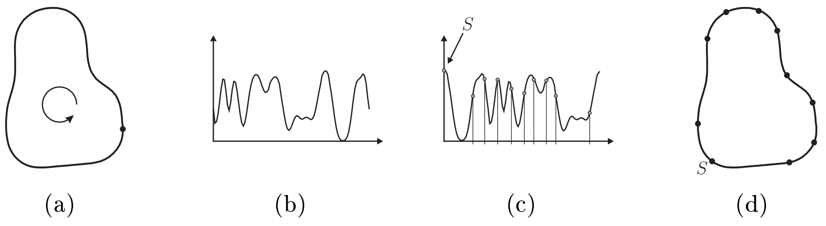


Fig. 2. Mass point distribution generation. **(a)**: oriented, closed curve with start point. **(b)**: according curvature function K along the curve. **(c)**: periodical translation of K (global maximum S is now $K(0)$) and subdivision in equal subintegrals by mass points. **(d)**: resulting mass point distribution on the curve. Regions of high curvature have denser mass point distribution.

3.2 B-spline description of the object contour

We need to know the curvature of the contour in each point of the contour. It will be used for the distribution of masses along the contour and to describe the target curvature of the curvature sensors. So we use B-splines for local abstraction and curvature calculation of contour segments, because of good fitting properties.

For every contour pixel we fitted a B-spline to a contour segment of fixed length, which has the contour pixel as its middle point. Thereto, beside the middle and the end pixels of this segment additional pixels are chosen, such that all these pixels lie equidistantely on the segment. These pixels become the control points of the B-spline.

Now the curvature at the middle pixel of the segment can be calculated by means of the fitted B-spline. Thereto, the curvature of the point on the B-spline that lies nearest to the middle contour segment pixel is easily calculated.

3.3 Boundary mass points

With the generated B-spline curve approximation for each pixel of the object contour, the curvature can easily be calculated for every point in this description. Therefore, we can now define the curvature function $K : [0,1] \rightarrow \mathbb{R}$ along the contour, where $K(0) = K(1)$ is the curvature on the start point of the closed contour. This function K can now be used to spread a given number n of mass points evenly in terms of accumulated curvature over the contour, such that the same cumulated curvature A_n exists between two adjacent mass points (Figure 2 shows this in detail.).

$$A_n = \frac{1}{n} \int_0^1 K(s)ds$$

Thereto, the first mass point is placed at the highest curvature point S on the contour to ensure that this curvature extremum can later be modeled directly by a sensor at this mass point.

This process leads to a mass point distribution on the contour whose density is directly correlated to the curvature. This is desirable, since high curvature is often considered a significant local feature (by e.g. [10]), which needs more mass points (with connected sensors) to describe them.

With each of these mass points, two sensors will be associated. One is a sensor for object contours of a certain normal direction. The other sensor searches for object contours of a certain curvature. Both search for contour features and therefore have to be associated with mass points on the model contour. The contour feature values to be searched for (normal direction, respective curvature) are directly taken from the spline that models their local contour.

3.4 Inner mass points

Our application is more a search task than a segmentation task, so that high shape variability is not expected for each modeled target structure. So we do not want a sole local force propagation along the connected mass points on the modeled contour, which would provide a strong and detailed local model adaption. As we do not expect huge object deformations, we want the sensor force not only to deform the model locally, but also to move the model as a whole. Therefore, a direct force propagation throughout the whole modeled part is desirable.

Connecting all contour mass point with each other is one possibility to achieve this, but it has the disadvantage of a very rigid and complex model with long, crossing springs, which could lead to an uneven force propagation throughout the model (e.g. crossing force propagation waves).

In our opinion, a better possibility is the introduction of inner mass points, that lie evenly distributed in the contour model and act as pure *force propagators* without having any sensors attached. So, acting forces can spread as evenly as possible over the whole model resulting in a controlled and straight global movement besides the local adaption. Figure 3 compares this concept to the one mentioned before.

Distribution of inner mass points We start with an initial distribution, where the preliminary inner mass points are initialized in a grid with a given density inside the model boundary. Principially, other initialization schemes are possible (see section 4.1 for more).

Then, we use a kind of relaxation algorithm to rearrange the inner mass points to an even distribution within the model contour. Thereto, all mass points (including the ones on the model contour) are considered as point charges of the same polarity. Additionally, the contour mass points will be temporarily (for this relaxation) connected along the contour with line charges that have also the same polarity as the rest.

Now, we use Couloumb's law to simulate the mass point movement caused by the electrical forces. The masses of all inner mass points to relax are the same (e.g. $m = 1$), so that they will distribute evenly. The masses of the contour mass points and the connecting line charges are set to infinity, so they will not move. If some mass points with sensors for geometric primitives are used, they will also get an infinite mass, so that they will stay fix on their position. In this work, for example, ellipse sensors are used for detecting the boreholes in some of the mounted parts. These sensors are placed on the drill positions described in the CAD data. Finally, simulation is executed time discretely with a damping factor for safe convergence of this numerical approach (figure 6(a)).

3.5 Interconnection of the mass points

Once all mass points are generated at the correct positions, they have to be interconnected. As outlined before in section 3.4, a complete interconnection is not desirable, but an interconnection with short springs and no spring crossings is preferable.

Triangulations have proven to be good topologies for such an interconnection ([8]). So we use a *Constrained Delaunay triangulation* ([11]) for this, which guarantees a triangulation with short links. The constrained variant of the Delaunay triangulation is necessary, because all potential springs along the object contour should be existent in the model, and the interconnection algorithm has to ensure that. They should always exist, because according to our experience, the possibility of direct force propagation along the contour is helpful for model fitting (e.g. model rotation along the contour or detail adaption to the object contour) and in the case of concave objects it is possible that Delaunay triangulation lacks some contour springs.

4 Evaluation

Model generation cannot be directly evaluated as a whole process. Therefore, we evaluate the critical steps of the model generation process directly. In addi-

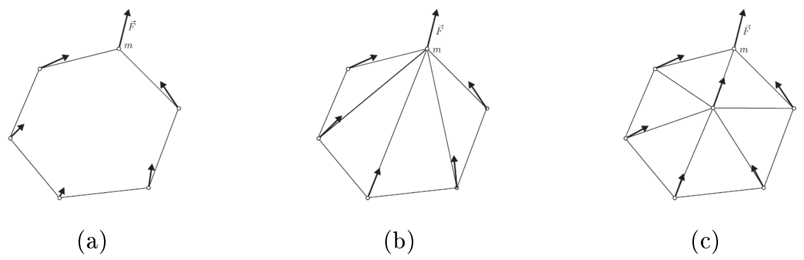


Fig. 3. Schematic influence of a local force on the whole model. **(a)**: stepwise force propagation along the border (no inner masses or springs). **(b)**: immediate force propagation over direct inner connections. **(c)**: indirect force propagation over inner mass points working as force propagators.

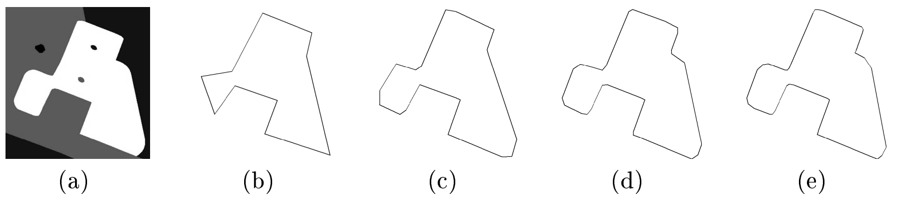


Fig. 4. Mounted part and derived model contour by n boundary mass points. **(a)**: part. **(b)**: $n = 10$. **(c)**: $n = 20$. **(d)**: $n = 30$. **(e)**: $n = 40$. There can be no significant contour enhancement seen above $n = 40$.

tion, the complete process will be evaluated indirectly by applying the generated model for the search process and evaluating the results.

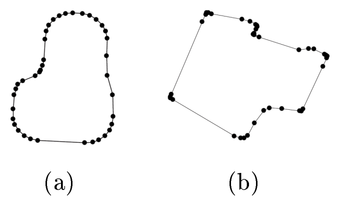
4.1 Critical steps of the model generation

B-spline contour fitting For the B-spline approximation of the pixel contour described in section 3.2, the B-spline curve has to be parameterized with the length l in pixels, the degree d and the number k of control points ($k \geq d$).

Short curve length, respective high curve degree and a high number of control points lead to a detailed local fitting. Long curve length respective low curve degree and low control point number do the opposite, a smooth abstraction. We want a good, but not too exact local adaption with not too complex B-splines for performance reasons. Therefore, we tried to balance local fitting and smooth abstraction in an appropriate way and determined empirically the values: $l = 50, d = 5, k = 5$.

Boundary mass distribution The quality of the generated model depends strongly on the masses on the boundary. Their position is determined automatically as described in section 3.3, but their number has to be specified manually. It should be as low as possible, but model all boundary curvature features. We generated model boundaries with different numbers of mass points for the part with the most complex contour. Figure 4 shows that 40 mass points were enough. We used this number of mass points for all parts to ensure that all relevant curvature features are modeled. Figure 5 shows schematic examples for two parts.

In figure 5(a), the mass points are well distributed, while lying denser in regions of higher curvature. In figure 5(b), this effect is much stronger, due to the very high curvature in the corners and the low curvature values along the edges. Such parts with an extreme mass point distribution were rare and no problem in our application. If they occur more often and more extreme in another task, we recommend to connect only curvature sensors to these mass points and create another set of equidistant mass points on the contour for the normal direction sensors. This way, large boundary areas without sensor forces are avoided and mass points keep lying on boundary positions with relevant



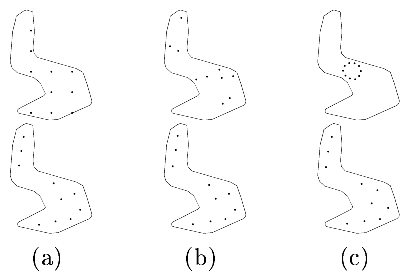


Fig. 5. Schematic boundary mass point distribution for the contours of example parts. In contour regions with high curvature mass point density is higher.

Fig. 6. Different point initialization schemes (above) and resulting relaxation (below). **(a)**: grid initialization. **(b)**: randomly uniformly distributed initialization. **(c)**: local initialization.

features for their sensors. But in this case, more sensors are needed for the boundary and therefore more springs are necessary to connect them with the interior mass points, which results in a more complex model.

Inner mass distribution Two questions arise for the evaluation of the inner mass point distribution:

1. How does the inner mass point distribution depend on the start point configuration before the relaxation process starts?
2. Does the relaxation process really lead to a homogeneous distribution?

To investigate the first question, we applied the relaxation process from section 3.4 to three very different start point configurations on 10 very different objects with always the same number of inner mass points to distribute. The points were positioned on a grid (figure 6(a)), randomly uniformly distributed (figure 6(b)) and cumulatively in a small section of the object (figure 6(c)).

As can be seen in figure 6, the point distribution always converges to nearly the same arrangement. This is underlined by the fact that the Euclidean Hausdorff of between the resulting point sets always (except once) lies below 9 % of the model diameter for tests with 10 inner mass points. The only case with a higher Hausdorff distance was the part shown in figure 7, which has a bottleneck hindering the points moving from one object part to another. In such cases it should be ensured that every region of the object is filled with some points in the initialization step. We therefore used and recommend initialization by grid. If the grid is too wide to cover some thin object parts, then these parts need no inner points according to the used inner mass point number according to our tests. For all other object parts, a good initialization with points is ensured using the grid method. We choose a grid width empirically that resulted in approximately 10 inner mass points per part, which proved sufficient to cover all regions of the different parts in our application well.

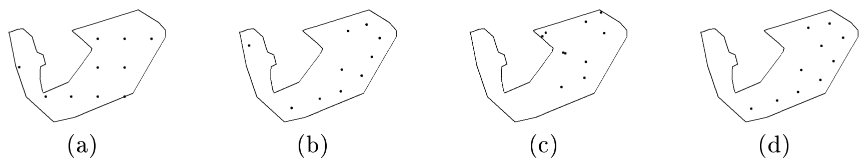


Fig. 7. Dependance of relaxation result from the initialization for a part with bottleneck. **(a)**: grid initialization with result shown in **(b)**. **(c)**: equal random initialization with result shown in **(d)**.

To investigate the second question, we compared the homogeneity of the inner mass point distribution before (after grid initialization) and after the relaxation process indirectly by the standard deviation of the spring lengths resulting from the subsequent triangulation. For the 10 above-selected objects it was found, that this standard deviation always decreases (in average by 43 %).

4.2 Model application

We applied the generated models to the search task in our application. The above-described model generation was performed automatically as the first step in the automatic search process. We tested the models on 13 different parts that were photographed from several directions, so that alltogether 57 pictures (1392 x 1036 pixels) were available. On 10 pictures the parts were missing or incorrectly mounted (rotated by 180 degrees).

On a standard PC (1 GHz Pentium III), the complete model generation process never took longer than 0.3 seconds. The models consists of approximately 50 mass points and 120 springs in average. Using these specifically generated models, all failures in part mounting were detected (see figure 8(d)). Correctly mounted parts were correctly classified in 84 % of the case (see figure 8(b)). The 16 % false positives were not caused by problems in the model fitting phase, but in the analysis of the deformation and fitting degree of the fitted model. So no direct failure in model generation can be deduced.

5 Conclusion

We introduced a technique to automatically construct SMSM models for the use in dynamic search tasks. The construction process relies only on an object sample given as a binary pixel image. After extracting the contour, it is approximated by B-splines to place mass points with contour normal and curvature sensors on characteristic postions. Inner mass points are added for the ellipse sensors. Furthermore, inner mass points are added and interconnected as homogeneously as possible for a smooth force propagation supporting the search task.

We showed that the parts of the model construction process do, what they were designed for. Additionally, we showed by an industrial application that the models are well constructed for the search and fitting of the parts they model.

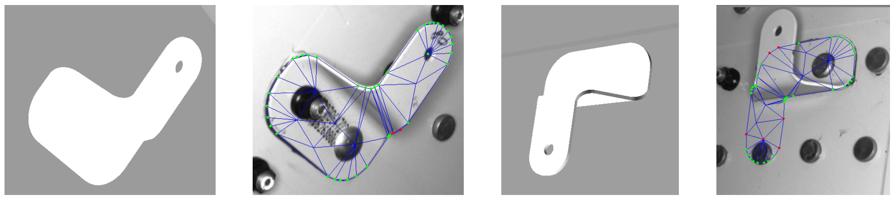

(a) (b) (c) (d)

Fig. 8. Model fitting. **(a)**: rendered and marked CAD model for the search in (b). **(b)**: Model has completely found the part. **(c)**: rendered and marked CAD model for the search in (d). **(d)**: Model has not completely found the part, because it was mounted incorrectly (rotated by 180 degree).

By use of this model construction technique, it is easily possible to visually compare the actual state of an object to a given reference state using the SMSM approach. With the specifically generated models, it is possible to find the matching object (if it is there) in images of the actual state, even if occlusions or deformations occur. Then comparison can take place by appropriate means (e.g. analysis of model deformation and fitting degree).

References

1. Sauer, S.: Modellbasierte, optische Prüfung der Vollständigkeit von montierten Baugruppen. Diplomarbeit, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg (2006)
2. Dornheim, L., Tönnies, K.D., Dornheim, J.: Stable dynamic 3D shape models. In: ICIP. (2005)
3. Bergner, S., Al-Zubi, S., Tönnies, K.: Deformable structural models. In: ICIP. (2004)
4. Pohle, R.: Computerunterstütze Bildanalyse zur Auswertung medizinischer Bilddaten. Habilitationsschrift, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg (2004)
5. Delingette, H.: Initialization of deformable models from 3D data. In: ICCV. (1998)
6. Ferrarini, L., Olofsen, H., van Buchem, M.A., Reiber, J.H., Admiraal-Behloul, F.: Fully automatic shape modelling using growing cell neural networks. In: MICCAI. (2005)
7. Kaus, M.R., Pekar, V., Lorenz, C., Truyen, R., Lobregt, S., Weese, J.: Automated 3d pdm construction from segmented images using deformable models. IEEE Transactions on Medical Imaging **22** (2003) 1005–1013
8. Dornheim, L., Dornheim, J., Tönnies, K.D.: Automatic generation of dynamic 3d models for medical segmentation tasks. In: SPIE: Medical Imaging. (2006)
9. Pavlidis, T.: Algorithms for Graphics and Image Processing. Computer Science Press (1982)
10. Cohen, I., Ayache, N., Sulger, P.: Tracking points on deformable objects using curvature information. In: ECCV. (1992) 458–466
11. Shewchuk, J.R.: Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In: WACG. (1996) 203–222