A Differential Alignment Approach for the Extraction of Photo Textures

Matthias Buck
Daimler-Benz AG
Research Center Ulm

Abstract

In this paper a method is presented to extract photo textures from perspective views of a real object and to map these textures on a geometry model of the same. The main task to be solved here is to recover the parameters of the perspective projection, which determine the appearance of the textures in the reference image. We propose to use a differential alignment approach to determine these parameters in a very user friendly and intuitive manner. The method bases on the superposition of the reference photo and a perpective projection of the known object geometry in one image. By a few simple 2d mouse operations, the projection parameters are optimized such that the object geometry is aligned to the reference view. Once this is achieved, the photo textures can be extracted and mapped onto the geometry model automatically.

1 Introduction

The generation of realistically looking models which can be rendered efficiently is a demand for any realtime and interactive computer graphics application, in particular in the field of virtual reality (VR). Such models typically consist of a geometric model of limited complexity, which represents the relevant 3d shape, while most of the details are described by appropriate photo textures which are mapped onto the geometry (see fig.1).

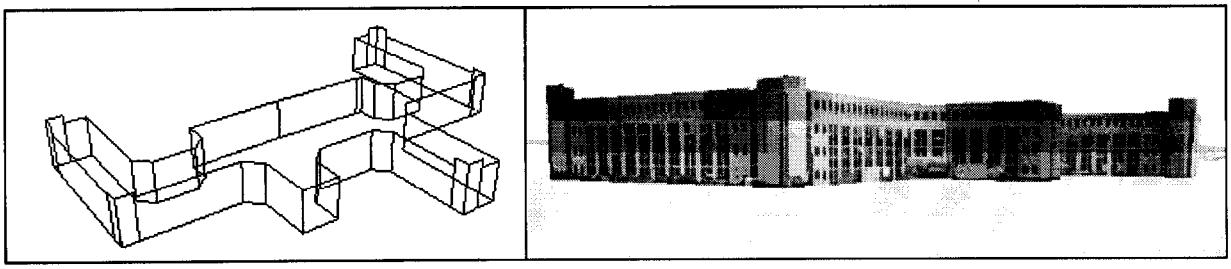

Figure 1: *On the left the geometry of a building as wire frame, on the right the same object with texture maps.*

This paper addresses the task to extract undistorted texture maps for a 3d model from 2d image material (e.g. photos) which shows the real object to be modeled. We describe a method which assumes the object geometry to be known, and which uses a set of still photographs of the real object which contain all relevant texture information. The external and internal camera parameters can be different for each photo and do not need to be known. The geometry model is projected onto the image plane of each photo, and aligned such that corresponding features (vertices, faces etc.) coincide. Once sufficient correspondence is achieved, the texture information can be extracted automatically for

all model faces which are visible in the respective view. It is important that this correspondence can be achieved as easily and intuitively as possible. In our approach, only a few simple 2d manipulations in the image plane are required to adjust the 3d model to the 2d image, to extract the textures, and to map them onto the geometry. Generally several photographs of the same object from different views have to be processed in the described way to obtain all necessary texture maps.

2 Related work

The extraction of geometric and photometric data from perspective views has a long history in the field of photogrammetry. Photogrammetric methods have been considered by the computer vision community for the reconstruction of 3d computer models from perspective images of real objects (for example [8]). A number of publications treats the extraction of textures from photographic image material, using various methods to determine the projection parameters which are necessary to recover undistorted texture maps. In [9], the acquisition of image sequences is strictly controlled, and the reference objects are rotated on a turn-table. The perspective distortion of parallel object structures in the image plane is employed by Buck [3] to recover object geometry and projection parameters for the reconstruction of buildings. An interesting approach for modeling architectural scenes is presented be Debevec et.al. [4]. It uses parameterized geometric primitives, which are matched against photographs of the real objects. A different approach is used by Azarbayejani and Pentland [1], who exploit the motion of manually marked features along video sequences to recover camera parameters and object properties. These approaches require either certain object shapes, or the objects to be modeled have to be either of a limited size or of a limited geometric complexity. The differential approach we are presenting here avoids all these limitations. It does not cope with the recovery of the geometry however, but focuses on the extraction of the texture maps. The mathematical tools we are using for this purpose are very similar to the ones used by Gleicher [6] to control the interactive manipulation of objects in 2D and 3D drawing tools. In [7], Gleicher and Witkin proposed to control a virtual camera with such a differential approach. The geometry of texture mapped models frequently is simplified to some degree (e.g. the facades of a building are approximated by planes even if structures like windows are not perfectly flat). In [4], view dependent texture mapping and model based stereopsis are proposed to produce improved renderings in case of such model inaccuracies. It is not clear though if such methods are suitable for real-time applications.

3 Recovery of undistorted photo textures

The mapping of photo textures onto 3d geometry models for computer graphics or VR applications requires undistorted textures (i.e. orthogonal views of the original object). In general, such orthogonal views are not available, or too expensive to acquire. Instead, in most cases the required textures are contained in photos as perspective views, and in consequence are perspectively distorted. Thus, the task which has to be solved before such image material can be used for texture mapping is to correct the perspective distortion.

A perspective projection maps a point $\mathbf{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3$ in world space to an image location $\mathbf{b} = (b_1, b_2)^T \in \mathbb{R}^2$. To describe this relation, we embed $\mathbf{x}$ and $\mathbf{b}$

in homogeneous coordinates $\mathbf{x}_h = (x_1, x_2, x_3, 1)^T$ and $\mathbf{b}_h = (b_1 w, b_2 w, fw, w)^T$. The perspective projection now is given by the homogenous transformation

$$\mathbf{b}_h = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{x}_h \tag{1}$$

where $\mathbf{R} \in \mathbb{R}^{3\times3}$ and $\mathbf{t} \in \mathbb{R}^3$ describe the transformation between the world coordinate space and the camera coordinate space, and f is the focal distance of the perspective projection. This mapping contains 7 projection parameters, 6 from the camera transformation and one from the focal distance f. In the rest of this paper, we will refer to (1) using the abbreviation

$$\mathbf{b} = \mathbf{F}(\mathbf{x}, \mathbf{p}) \tag{2}$$

where $\mathbf{p} \in \mathbb{R}^7$ denotes the set of projection parameters.

In the following, we assume that the 3d geometry of the object, for which we want to extract texture maps, is known, e.g. in form of CAD data, or reconstructed from images as in [1, 3, 4, 8, 9]. Given a perspective image of the real object, and assuming that the seven projection parameters mentioned above are already known, each image location can be re-projected into the 3d space (see fig. 2), and the textures can be mapped correctly onto the geometric representation of the model.

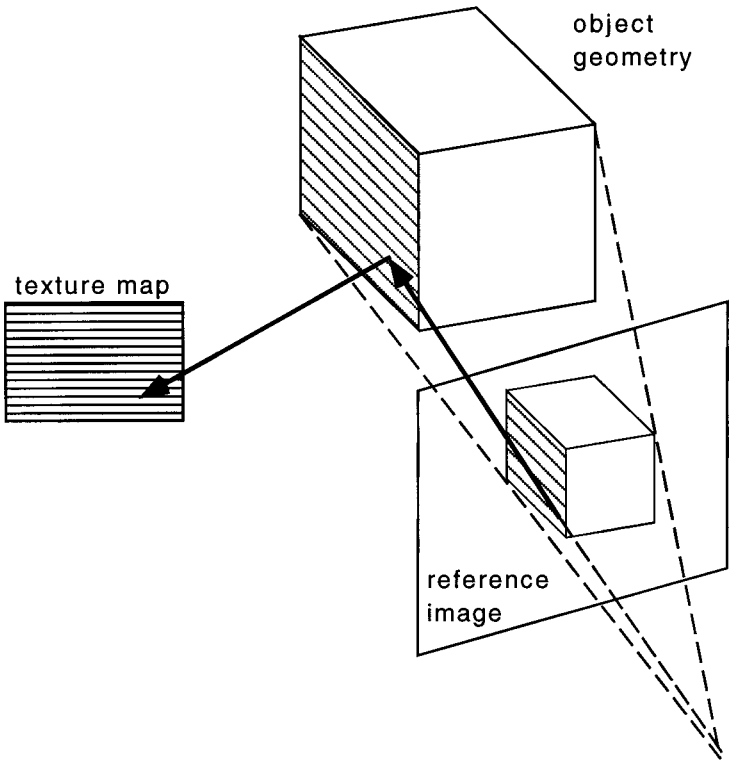

Figure 2: *Mapping between reference image, 3d object geometry and texture map.*

So the determination of undistorted texture maps requires to recover the set of projection parameters $\mathbf{p}$. The related problem of camera calibration has been addressed by many authors (e.g. [5, 2]). In most cases, images of calibrated reference objects are taken

with the used camera to recover the projection parameters. As we assume to know the geometry of the object for which we want to extract texture maps, we can use this object right away as calibration object. This requires to relate a number of reference points of the 3d geometry to the corresponding image plane locations where these points are projected to.

In the following chapter, we present a differential method which allows to specify these relations step by step in a very intuitive and user friendly way. It is semi-automatic, and the implied user interaction does not require special skills.

4 The interactive alignment process

Our strategy bases on an overlay technique: the geometry of the 3d model is projected onto the image plane and is overlaid with the photo of the real object, using the set of projection parameters we want to determine. As long as these parameters are not correct, the projected geometry and the photograph do not coincide. The technique we present in the following allows the user to select some robustly identifiable features of the object geometry and to drag them towards the locations where they appear in the photograph. This simple task has to be repeated for another three features, until the projected geometry and the photo are sufficiently aligned. At this point, the current projection parameters are the ones which were used when the photo was taken, and thus they are the ones we wish to determine to recover undistorted texture maps.

To develop our approach, we first extend (2) to a system of n equations of the same type,

$$\mathbf{u} = \mathbf{F}_n(\mathbf{y}, \mathbf{p}) \tag{3}$$

where $\mathbf{y} = (\mathbf{x}_1, \ldots \mathbf{x}_n)^T \in \mathbb{R}^{3n}$ collects the 3d points which are projected to the image locations $\mathbf{u} = (\mathbf{b}_1, \ldots \mathbf{b}_n)^T \in \mathbb{R}^{2n}$. Next we build the differential formulation of this equation system, because we want to derive the relation between the change rate $\dot{\mathbf{p}}$ of the set of projection parameters and the resulting change rate $\dot{\mathbf{u}}$ of the projected image locations:

$$\dot{\mathbf{u}} = \mathbf{J}\dot{\mathbf{p}} \tag{4}$$

where

$$\mathbf{J} = \frac{d\mathbf{F}_n(\mathbf{y}, \mathbf{p})}{d\mathbf{p}} \in \mathbb{R}^{2n \times 7}$$

is the Jacobian of the system. The most convenient way to compute $\mathbf{J}$ is to determine an aproximation by the method of finite differences (see [10]), which does not require to determine the partial derivatives of $\mathbf{F}_n$ analytically. The numerical precision of this approximation is sufficient for our purpose.

Now we need to determine the change rate $\dot{\mathbf{p}}$ from $\dot{\mathbf{u}}$. However, (4) is not invertible if $\mathbf{J}$ is not square and of full rank, which means physically that there is no unique and correct solution to this problem. Instead, we can use the so called pseudo-inverse $\mathbf{J}^I$, and obtain the approximative solution

$$\dot{\mathbf{p}} = \mathbf{J}^I\dot{\mathbf{u}} \tag{5}$$

The definition of $\mathbf{J}^I$ depends on the sign of $2n - 7$. If $2n > 7$, it is

$$\mathbf{J}^I = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T, \tag{6}$$

which minimizes the quadratic error functional

$$E = \left(\dot{\mathbf{u}} - \mathbf{J}\dot{\mathbf{p}}\right)^T\left(\dot{\mathbf{u}} - \mathbf{J}\dot{\mathbf{p}}\right) \tag{7}$$

If on the other hand $2n < 7$, the definition is

$$\mathbf{J}^I = \mathbf{J}^T\left(\mathbf{J}\mathbf{J}^T\right)^{-1}, \tag{8}$$

which minimizes $\dot{\mathbf{p}}^T\dot{\mathbf{p}}$ and thus the absolute parameter change rate.

Now that we have laid the mathematical basis, we can proceed to describe the alignment process in more detail. At any stage of the procedure, there is a number of n control points (collected in $\mathbf{y}$), which the user has selected, and which are projected onto the image plane at the locations contained in $\mathbf{u}$. These control points are dragged by the user towards the corresponding image locations, one at a time. However, what appears to be a simple dragging operation in fact is achieved indirectly by changing the camera parameters.

The procedure is as follows. The user specifies a desired motion $\Delta\hat{\mathbf{b}}_i$, $1 \leq i \leq n$, for the projection of one of the control points, while the remaining control points shall remain unchanged. The desired change vector is

$$\Delta\hat{\mathbf{u}} = (\mathbf{0}, ..., \Delta\hat{\mathbf{b}}_i, ..., \mathbf{0}). \tag{9}$$

With (5), the set of projection parameters is updated according to

$$\Delta\mathbf{p} = \mathbf{J}^I\Delta\hat{\mathbf{u}} \tag{10}$$

The final displacement of the control points in the image plane occurs only due to the changed projection parameters $\mathbf{p}$.

We want to point out how the update of the projection parameters depends on the two cases of the determination of $\mathbf{J}^I$. If $2n < 7$, there are more free parameters than equations, and the resulting displacement $\Delta\mathbf{u} = \mathbf{J}\Delta\mathbf{p} = \Delta\hat{\mathbf{u}}$ is realized exactly, with the minimal possible change of $\Delta\mathbf{p}$. If $2n > 7$, the number of free parameters is not sufficient to realize any desired displacement $\Delta\hat{\mathbf{u}}$ of the n control points exactly. Instead, an approximation is determined which minimizes the quadratic error (7). This implies that the previously aligned reference points may be slightly displaced again, so that the user might need to re-align them. If the physical camera would match exactly the underlying simplified camera model, a perfect correspondence would exist, which standard photogrammetric methods could determine more efficiently, than the proposed interactive method which can require a few re-alignment steps. Yet in practice the physical camera differs slightly from the camera model, and a perfect alignment cannot be achieved by any means. In this situation, the interactive differential alignment process allows the user to determine a solution which is optimal according to the overall visual judgement of the resulting correspondence between geometry and image.

The alignment proceeds in several steps. First, it is convenient to align the projected geometry and the reference image approximately by direct manipulation of the projection parameters. This can be done for example by rotating and moving the virtual camera

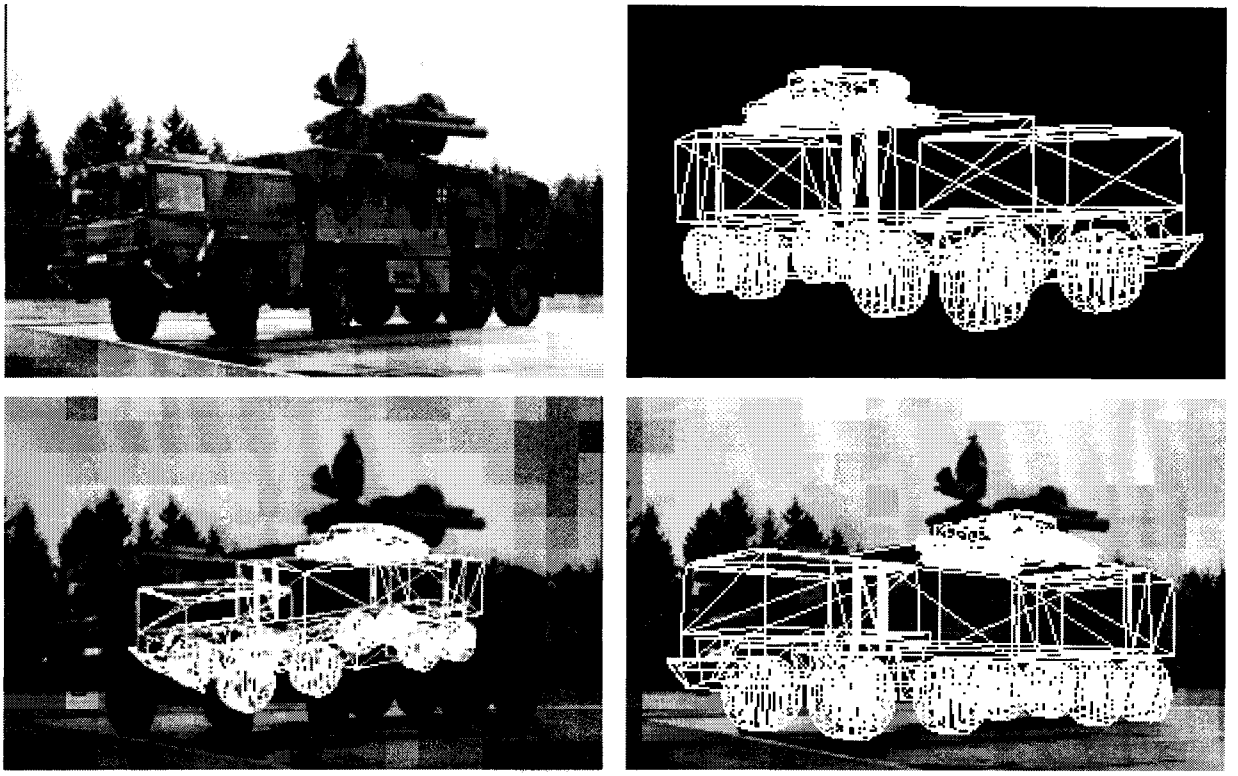

Figure 3: *Alignment between the 3d object geometry and a reference image. A reference photo (upper left) and the object geometry (upper right) are overlaid (lower left) and aligned until optimal correspondence is achieved (lower right).*

using a 6-dof input device like a space ball. After that, the differential approach is used to achieve the fine alignment (see fig.3). To do so, the user first selects one control point (i.e., $n = 1$), and drags it to the correct position. Next, a second control point is selected ($n = 2$), and moved to the correct image location. The first control point remains unchanged at this stage. This procedure is repeated with a third and fourth control point. If the overall alignment is not satisfactory at this stage, any of the four control points can be re-aligned appropriately. Of course, during the alignment process, not only the control points are visible, but all of the geometry (which is visualized as shown in fig. 3 as a wireframe model). This allows to judge the overall alignment between geometry model and reference image. If the camera which was used to take the reference image did not deviate too much from the underlying model of perspective projection (e.g. no fish eye lens was used), a good correspondence can be obtained with this method. Of course the model geometry has to match the geometry of the real object with sufficient accuracy. In the example of fig. 3, this is true except for the antenna device on top of the truck and the fromt wheels, which cannot be textured in the described manner.

4.1 Selection of control points

The user should choose control points which can be clearly identified in the geometry model as well as in the reference image. This makes the alignment process easy and reliable. There is one more requirement: The set of control points must be linearly independent, which means that any three of them do not belong to a common line,

respectively all four points do not belong to a common plane. Otherwise the pseudo-inverse would not exist, and hence no solution for (10). For the sake of numerical stability, it is even desirable that any three points are not in the proximity of a common line, and that all four points are not in the proximity of a common plane. In our implementation, these rules are checked when the third control point $\mathbf{x}_3$ and the fourth control point $\mathbf{x}_4$ are selected, according to the following conditions:

$$|\mathbf{n}| > \lambda|\mathbf{x}_2 - \mathbf{x}_1||\mathbf{x}_3 - \mathbf{x}_1| \quad (11)$$

$$|\mathbf{n}(\mathbf{x}_4 - \mathbf{x}_1)| > \lambda\sqrt{|\mathbf{n}|^3} \quad (12)$$

where $\mathbf{n} = (\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)$, and a good value for λ is $\lambda = 0.1$. If condition (11) does do not hold for $\mathbf{x}_3$, or condition (12) does do not hold for $\mathbf{x}_4$, the chosen point is not accepted as control point.

4.2 The interaction interface

As described before, the user selects control points in the image plane and drags them towards the correct locations in the reference image. The motion of these points is realized indirectly by an adaptation of the parameters of the perspective projection with the differential approach discussed above. In the derivation of this approach, we identified $\Delta\mathbf{u}$ with $\dot{\mathbf{u}}$, which is only allowed for a small displacement $\Delta\hat{\mathbf{b}}_i$ of the projection of the selected control point. To avoid numerical instabilities, in our implementation we limit the magnitude of $\Delta\hat{\mathbf{b}}_i$ for each update step to $|\Delta\hat{\mathbf{b}}_i| < d_{max}$:

$$\Delta\hat{\mathbf{b}}_i = \begin{cases} \hat{\mathbf{b}}_i - \mathbf{b}_i & \text{if} \quad |\hat{\mathbf{b}}_i - \mathbf{b}_i| < d_{max} \\ d_{max}\frac{\hat{\mathbf{b}}_i - \mathbf{b}_i}{|\hat{\mathbf{b}}_i - \mathbf{b}_i|} & \text{else} \end{cases}$$

where $\hat{\mathbf{b}}_i$ is the desired new position of the selected control point, and $\mathbf{b}_i$ is its current position. This corresponds to an automatic subdivision of large parameter changes into sufficiently small ones.

If the specified motion exceeds the above limitation, the projection $\mathbf{b}_i$ of the selected control point will lag behind the desired motion, but after some iterations finally converge to the desired location, as $\Delta\hat{\mathbf{b}}_i$ is always directed towards the desired position $\hat{\mathbf{b}}_i$.

5 The texture extraction

Once the projection of the 3d model is aligned sufficiently with the reference image, the texture information for individual object surface elements can be extracted. As the textures are perspectively distorted in the photo, this distortion must be reversed using the parameters of the perspective projection to obtain rectified texture maps. Since these parameters have been determined in the previous step, the mapping from image coordinates to texture coordinates is known, and the texture data can be extracted from the photo using equation (3), and can be mapped onto the 3d model geometry automatically. It is possible to do this either for all faces which are visible in the current image, or to select a particular subset interactively. Automatic texture mapping is of particular interest in the case of rather complex 3d object geometries with many faces, where manual texturation would be impractical. In this case, if the object geometry is

given e.g. from CAD data, only a few camera parameters have to be estimated, which is easily done with the described alignment method.

In general, multiple views are required to obtain all necessary textures for a model. The alignment and texture extraction process has to be repeated with each photograph, and the texture maps from the different views have to be mapped onto the geometry model.

There are two potential problems in this context which need special attention. If one face (or part of it) is visible in more than one photograph, a strategy has to be included to decide which image data to use, or how to mix the different image data in one texture. At present, in our realization the texture map for one face has to be extracted entirely from one single photograph.

A second type of problem can occur if the model is not convex, such that in a given view some parts of the geometry can occlude other parts of it. In this case, some faces are only partially visible, and no complete texture maps can be obtained for such faces from the present view. The missing parts have to be complemented from other photographs which are taken from different views. In the current realization, such occlusion effects are not yet addressed. Methods to solve both mentioned problems however have already been described e.g. by [4].

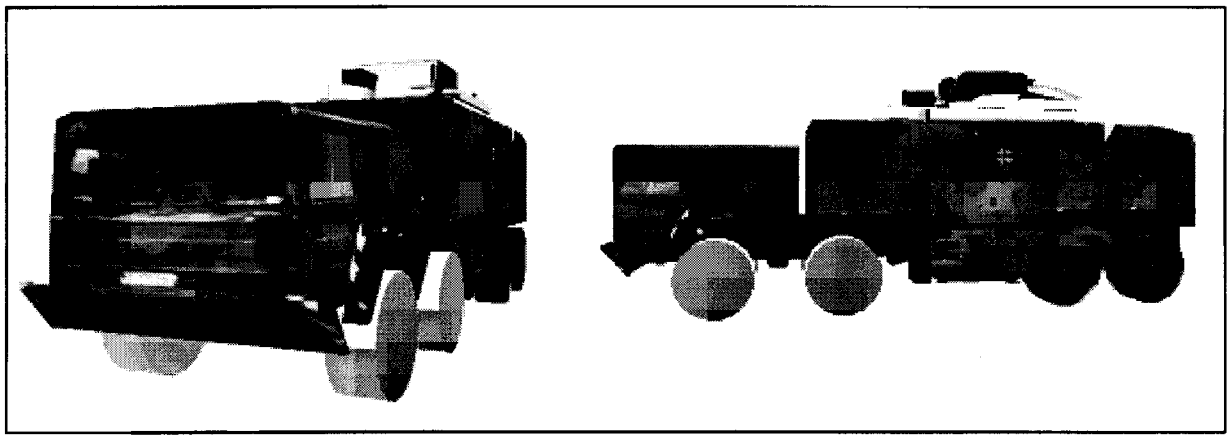

Figure 4: *Different views of the truck model from fig. 3 with attached texture maps.*

6 Conclusion and Future Work

A differential alignment approach for the extraction of photo textures has been presented. The described method has been successfully tested with several models. Fig. 3 shows the alignment process for one of the test objects. Different views of the resulting model with attached texture maps are shown in fig. 4. Although the whole process is not completely automatic, it is very user friendly and intuitive, and allows to optimize the resulting alignment even if the used camera model does not exactly match the physical camera parameters. As each alignment step is rather fast, the selected object vertices follow the mouse position without noticeable delay on a workstation with hardware texture acceleration, unless the mouse moves very fast.

The main focus of this contribution was laid on the alignment procedure. The texture extraction including the correction of the perspective distortion and the automatic

mapping onto the geometry model have been realized, however without taking partial occlusions into account, which should be done. Another planned extention of our approach is to allow the combination of multiple image data from different views to a single texture map. In this context, the challenge is to combine image material with different resolutions and from different illumination conditions without noticeable artifacts.

References

[1] A. Azarbayejani, A. Pentland: *Recursive Estimation of Motion, Structure, and Focal Length*, IEEE PAMI, Vol.17, No.6, pp.562-575, June 1995

[2] D. Ballard, C. Brown: *Computer Vision*, Prentice-Hall, 1982

[3] M. Buck: *Modelling Buildings from Single Images*, in Y. Paker, S. Wilbur (eds): Image Processing for Broadcast and Video Production, Springer, 1994

[4] P. Debevec, C. Taylor, J. Malik: *Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach*, Proceedings Siggraph '96

[5] O. Faugeras: *Three-Dimensional Computer Vision*, The MIT Press, Cambridge - London, 1993

[6] M. Gleicher: *A Differential Approach to Graphical Interaction*, Ph.D. Thesis, Carnegie Mellon University, CMU-CS-94-217, November 1994

[7] M. Gleicher, A. Witkin: *Through-the-lens camera control*, Computer Graphics 26(2), Proceedings Siggraph '92, pp. 331-340, July 1992

[8] P. Mulgaonkar, L. Shapiro, R. Haralick:: *Shape from Perspective: A Rule Based Approach*, CVGIP **36**, pp.298-320, 1986

[9] W. Niem: *Robust and Fast Modelling of 3D Natural Objects from Multiple Views*, SPIE Proceedings "Image and Video Processing II", Vol. 2182, pp.388-397, 1994

[10] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling: *Numerical Recipes in C*, Cambridge University Press.

[11] Y. Shan, Y. Koren: *Obstacle accomodation motion planning*, IEEE Transactions on Robotics and Automation, Vol.11, No.1, Febr. 1995