3-D Reconstruction of a Dynamic Environment With a Fully Calibrated Background for Traffic Scenes

Karsten Müller, *Member, IEEE*, Aljoscha Smolić, Michael Dröse, *Member, IEEE*, Patrick Voigt, and Thomas Wiegand

***Abstract*—Vision-based traffic surveillance systems are more and more employed for traffic monitoring, collection of statistical data and traffic control. We present an extension of such a system that additionally uses the captured image content for 3-D scene modeling and reconstruction. A basic goal of surveillance systems is to get a good coverage of the observed area with as few cameras as possible to keep the costs low. Therefore, the 3-D reconstruction has to be done from only a few original views with limited overlap and different lighting conditions. To cope with these specific restrictions we developed a model-based 3-D reconstruction scheme that exploits *a priori* knowledge about the scene. The system is fully calibrated offline by estimating camera parameters from measured 3-D–2-D correspondences. Then the scene is divided into static parts, which are modeled offline and dynamic parts, which are processed online. Therefore, we segment all views into moving objects and static background. The background is modeled as multitexture planes using the original camera textures. Moving objects are segmented and tracked in each view. All segmented views of a moving object are combined to a 3-D object, which is positioned and tracked in 3-D. Here we use predefined geometric primitives and map the original textures onto them. Finally the static and dynamic elements are combined to create the reconstructed 3-D scene, where the user can freely navigate, i.e., choose an arbitrary viewpoint and direction. Additionally, the system allows analyzing the 3-D properties of the scene and the moving objects.**

***Index Terms*—Kalman filter, multiview 3-D reconstruction, surveillance, traffic modeling, videostreaming.**

I. Introduction

VISUAL surveillance systems are more and more employed for observation and protection of public and private areas. Often many cameras have to be used to get a good coverage of the area to be monitored. For this purpose we have developed a multiview video streaming system that can include an arbitrary number of cameras that are connected wirelessly to a central station. The system is designed as a multiserver/single client system and employs state of the art MPEG-4 video coding, RTP steaming and control mechanisms to adapt to varying network conditions.

Besides simple monitoring advanced visual surveillance systems employ image processing to extract additional data from the video. This can be for instance automatic detection and tracking of moving objects and recognition of certain events, etc. A special case is traffic surveillance where intelligent surveillance systems can be used for detection and tracking of vehicles, recognition and classification of vehicles, traffic analysis, acquisition of statistical data, traffic control and handling of emergency situations.

Such systems fall into two separate scenarios: static surveillance systems are installed at certain places to monitor roads, crossings or other important areas. They can include automatic vehicle tracking, which can be parameter-based [20] or object-based [16]. Also automatic object classification can be part of such systems [5]. The second group of traffic surveillance systems is associated with driver assistance, where cameras are installed within vehicles to monitor traffic parameters to adjacent vehicles, as shown in [13].

Such intelligent surveillance systems include several processing steps, starting from the segmentation of static background and moving objects, which is widely studied in prior work. One suitable approach, which is employed in the work described in this paper, applies an adaptive Kalman filter to separate background and moving objects, even in case of changing background conditions [17]. Similar ideas for background extraction have been proposed in [9].

The next step is tracking of the extracted objects over time, which is also widely studied in the literature. Intelligent traffic surveillance systems require tracking algorithms, which are robust against object occlusions, since the vehicles in traffic surveillance scenes very often occlude each other. We employ a tracking algorithm based on [17] that includes occlusion reasoning and is, therefore, very well suited for the problem at hand. Other methods for vehicle tracking with occlusion handling in static surveillance scenarios have been proposed in [8], [19], [23], and [10]. The problem of object occlusions is also an issue in driver assistant systems as described in [12] and [15]. Object detection and tracking is also required for extracting parameters for traffic control purposes as shown in [3], [6].

As an extension of such intelligent traffic surveillance systems we present an advanced approach that adds the functionality of 3-D scene reconstruction from multiple camera views. The contribution of this paper lies in the combination of existing algorithms for segmentation and 2-D object tracking with adapted algorithms for 3-D tracking and reconstruction to the specific requirements of such scene setups. A 3-D model is generated that allows observing the traffic scene from an arbitrary viewpoint and direction. This enables a better visualization and

Manuscript received May 5, 2003; revised August 29, 2003. This work was supported by the German Federal ministry of Education and Research under Grant 03WKJ02C. This paper was recommended by Associate Editor E. Izquierdo.

K. Müller, A. Smolić, and T. Wiegand are with the Image Processing Department, Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut, Berlin 10587, Germany (e-mail: kmueller@hhi.de).

M. Droese is with the Technical University of Berlin, Berlin 10587, Germany.

P. Voigt is with Siemens AG Power Generation, 10553 Berlin, Germany.

Digital Object Identifier 10.1109/TCSVT.2005.844452

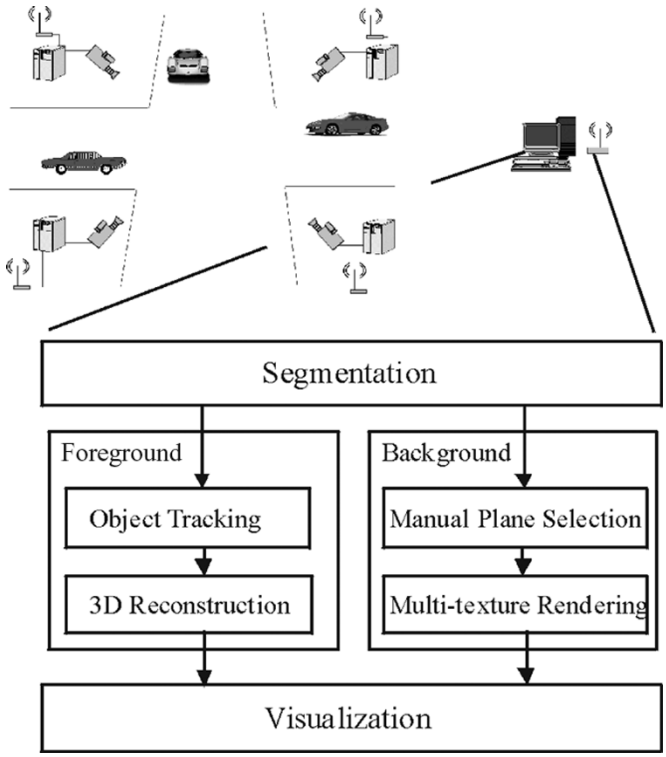


Fig. 1. Surveillance setup with four wireless servers and a central client for traffic monitoring and scene rendering.

analysis of the traffic, e.g., after a crash or in an emergency situation. It also allows analyzing the 3-D properties of the scene and the moving vehicles (e.g., evaluation 3-D motion trajectories).

A basic goal of surveillance systems is to get a good coverage of the observed area with as few cameras as possible to keep the costs for installation and maintenance of cameras and transmission channels and complexity in scene calibration reasonable. Therefore, the 3-D reconstruction has to be done from only a few original views with limited overlap and different lighting conditions. Hence, some 3-D scene modeling methods, like voxel-based reconstruction [22], light fields [18], or disparity-based reconstruction [7] cannot be used in general for such a scenario, because to obtain reasonable reconstruction results they rely on a rather dense camera grid with much overlapping image content. Nevertheless, some approaches have been undertaken to extract optical flow fields from images with large displacements [1].

Our approach consists of a model-based reconstruction method requiring *a priori* knowledge about the scene geometry and camera calibration. The scene is separated into static parts (streets, sidewalks, buildings, etc.) that are modeled semi-automatically and dynamic objects (vehicles, pedestrians, bicycles, etc.), which are modeled automatically using predefined geometric primitives. When mapping the 2-D views from all cameras into the 3-D scene, original lighting conditions should be preserved to present original information at the appropriate viewpoints. All components are combined, using a 3-D compositor that allows free navigation within the scene. As a result, normal traffic as well as emergency situations can better be observed and evaluated. The system setup is shown in Fig. 1.

The remainder of this paper is structured as follows. Video streaming is presented in Section II. Section III describes scene segmentation and calibration. The description of multitexture background reconstruction is presented in Section IV. Section V explains moving foreground object reconstruction. The 3-D scene integration is presented in Section VI. Finally Section VII summarizes and concludes the paper.

II. Multiview Video Streaming

The multiview video transmission system consists of two parts, an autonomous server application at each surveillance camera, and a client application to receive multiple video streams at a central computer, see Fig. 2.

A separate frame grabber module was implemented which can be easily customized or exchanged. The module acquires uncompressed frames from a Firewire camera. First the uncompressed data must be converted to YC_bC_r 4:2:0, which is required by the internal MPEG-4 video encoder [14]. This is done by a special module, which is able to convert type, spatial and temporal resolution automatically for better usability. For real-time encoding a fast motion estimation is carried out within the video coder. The bitstream is transmitted using real-time transport protocol (RTP). Further, all transmitted data can be encrypted, including RTP packets and control commands.

A remote control mechanism was designed for server configuration and maintenance purposes through the central computer to avoid costly manual maintenance at each crossing. Thus, the client contains a module for transmitting server control commands over a separate backward TCP channel.

Theoretically, the client application can receive an arbitrary number of video streams, but the number is limited by the system performance parameters, mainly processor performance and network bandwidth. In the experiments we successfully tested a local net, where four servers transmitted up to two video streams each, using a common PC platform. The client contains one processing module for each stream, which includes an RTSP/RTP client, an MPEG-4 decoder and an appropriate rendering module. To adapt video streams to changing transmission conditions, packet loss rates are analyzed and new server configurations are set up, if necessary. Two mechanisms are applied at the server side to meet transmission conditions: gradual bit rate adjustment by rate-control and variation of temporal and spatial resolution.

For the traffic scenario, where an image resolution of 640 $\times$ 480 pixel and frame rate of 15 Hz is used, the transmitted data rate was about 1–1.5 MBits/s due to the coding of mainly static background and small moving objects. Since 802.11b as the transmission protocol of choice leaves approximately 6 Mbits/s, a maximum of four video streams could be transmitted to one single client. This number can be extended for 802.11 g with an effective bandwidth of about 20 Mbit/s. Since the 3-D reconstruction system only requires an image frame rate of 7.5 Hz, 50% bandwidth remains for retransmitting in case of RTP packet loss, which is automatically handled by the RTSP protocol.

If the system is used for observation purposes, i.e., to monitor traffic, premises, parking blocks, etc., the transmitted video data

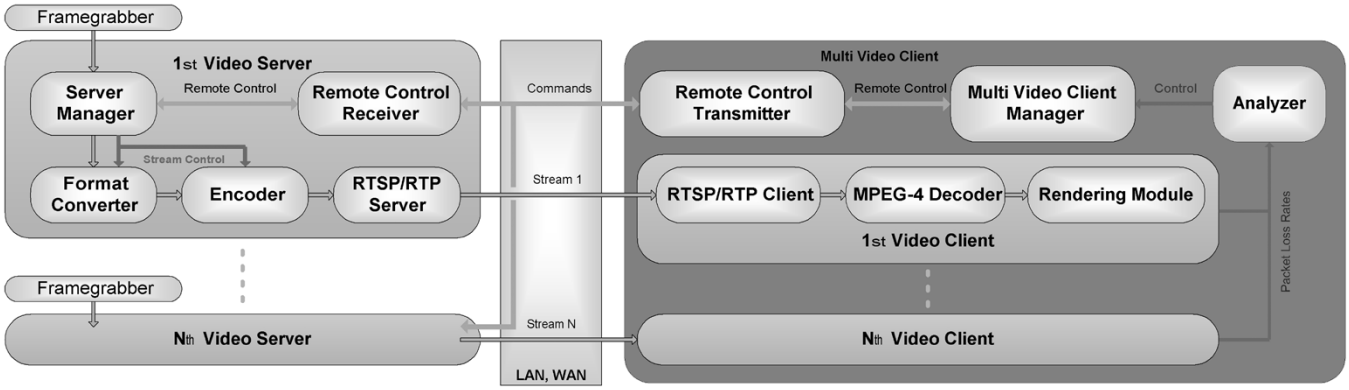


Fig. 2. Multiview video streaming transmission architecture.

can also be stored as MPEG-4 streams or uncompressed single frames to record important incidents. The application of this system is not restricted to surveillance. It can be used in any multiview video scenario, e.g., video conferencing with several partners.

III. Segmentation and Camera Calibration

The scene creation process starts with the separation of static and dynamic scene parts. Therefore, we first extract foreground objects from the static background, using a segmentation-by-motion approach with adaptive background estimation [21]. A simple Kalman filter is assigned to each pixel. The system state contains pixel intensity and its first derivative. To distinguish between foreground and background, the state is compared to a variable threshold value at each time step. In this approach we assume that a background pixel is characterized by an intensity value that remains either constant over a decent number of frames or only changes slowly due to environmental changes. The later condition is characterized by a low value of its first derivative. If the derivative of intensity is below the threshold, the pixel is recognized as background and the state is updated accordingly, taking into account the Kalman filter gain settings. Otherwise the pixel is recognized as foreground and only a minimum update of the state is carried out, also depending on different Kalman filter gain settings. The thresholds are updated dynamically and independently for each pixel taking into account a statistical analysis of a number of previous measurements. It is increased if a pixel is recognized as foreground, which allows adaptation to environmental changes, e.g., a parking car. For more details on the algorithm please refer to [21].

Fig. 3 shows a segmentation example for one input camera view. Moving objects are detected by combining all foreground pixels within a region by morphologic operation and surrounding the cluster by a contour. This contour is than approximated using cubic B-splines. The algorithm needs an initialization phase of a certain number of frames to extract a reliable background image where moving foreground objects are removed. As a side effect the extracted background images of all views are used in the 3-D background modeling algorithm as described in Section IV. Fig. 4 shows an example of an extracted background image.

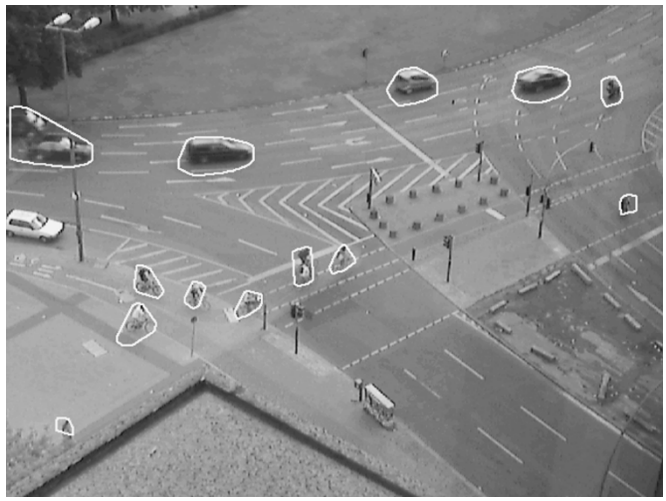


Fig. 3. Detected foreground regions highlighted by convex polygons.

Fig. 4. Background image after foreground objects have been removed.

One benefit of this method is the fast and robust adaptation to environmental changes, e.g., changes in lighting, fog, rain, snow, etc., since a background update is carried out for each new frame using a Kalman filter formalism for each pixel. Another advantage is the possibility to extract even nonmoving objects, like vehicles that stop for a small period of time due to congestion or red traffic lights. Furthermore, the filter formalism is robust to coding artifacts caused by the MPEG-4

video compression during transmission at the given transmission parameters, described in Section II. Coding artifacts in the textures are only visible in the reconstructed scene. One tradeoff in connection with the applied segmentation process occurs through shadows from moving objects. In this case the shadow areas within the background are associated with foreground objects. This problem and the very different textures in each view prevent the use of image-based alignment methods, e.g., disparity estimation, for 2-D data fusion from all views to a common 3-D object. Therefore, we decided to use a more robust nearest-point fusion approach as described in Section V. Finally, the segmented objects will be used for texture mapping onto real objects. Since the Kalman filter formalism is applied for each pixel, an object texture is formed by clustering the individual pixels and surrounding them by a contour.

Our model-based approach to 3-D scene reconstruction requires a fully calibrated system, i.e., knowledge about camera parameters that map the 3-D world to each 2-D camera view as well as homographies that map all 2-D camera views to each other. This is needed for instance for the projection of object textures from each view into a common 3-D plane and to properly position artificial objects in the 3-D scene. We assume that correspondences between a number of points the 3-D world and points in each camera view are available. For selection of appropriate points in the 3-D scene and 2-D views, an architectural map with original scene information was used. Another approach was the usage of fiducial markers within the scene, that where measured by GPS. The associated 2-D points within all images where selected manually. For scene calibration a total of eight points around the scene was enough for providing adequate projection matrices. By manually providing the point correspondences, environmental influences of different lighting conditions and shadow problems can thus be neglected.

A common direct linear transform (DLT) algorithm is used to calculate the projection matrices for each view [24]. Here, we assume a 3×4 projection matrix $\boldsymbol{A}$ that transforms a point $\boldsymbol{P}(x, y, z)$ from the 3-D world onto a point $\boldsymbol{p}(u, v)$ in the image plane [11]

$$\boldsymbol{p} = \boldsymbol{A}\boldsymbol{P} \quad \text{or} \quad \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{1}$$

The projection above yields two equations that are rearranged into the typical form for a certain point labeled i

$$\begin{pmatrix} x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i & -u_i \\ 0 & 0 & 0 & 0 & x_i & y_i & z_i & 1 & -v_i x_i & -v_i y_i & -v_i z_i & -v_i \end{pmatrix} \boldsymbol{a} = \boldsymbol{0}. \tag{2}$$

Here, $\boldsymbol{a} = (a_{11}, a_{21}, a_{31}, a_{41}, a_{12}, a_{22}, a_{32}, a_{42}, a_{13}, a_{23}, a_{33}, a_{43})^{\mathrm{T}}$ contains the rearranged elements of the projection matrix $\boldsymbol{A}$. Its calculation requires at least 6 point correspondences $\boldsymbol{p}_i$ and $\boldsymbol{P}_i$, but we use more correspondences $N > 6$ to obtain an overcomplete system. This allows a better approximation reducing the influence of measurement noise. Since the elements of $\boldsymbol{A}$ are the same for all corresponding points of a certain view, the 2×12 matrix from (2) is stacked for all $\boldsymbol{p}_i$ and $\boldsymbol{P}_i$ with $i = 1 \ldots N$ and (2) extends to a $2N \times 12$ matrix. With this stacked matrix, vector $\boldsymbol{a}$ is obtained by singular value decomposition. A homography matrix $\boldsymbol{H}$ describes the projection of the points in one camera view into another. The projection of a point in one camera plane $\boldsymbol{p}_1$ into a point in the second camera plane $\boldsymbol{p}_2$ is expressed by an equation, which is rather similar to (1)

$$\boldsymbol{p}_1 = \boldsymbol{H}\boldsymbol{p}_2 \quad \text{or} \quad \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix}. \tag{3}$$

Again, the elements of the 3×3 matrix $\boldsymbol{H}$ are calculated as described for the 3-D-2-D case above, using the DLT algorithm.

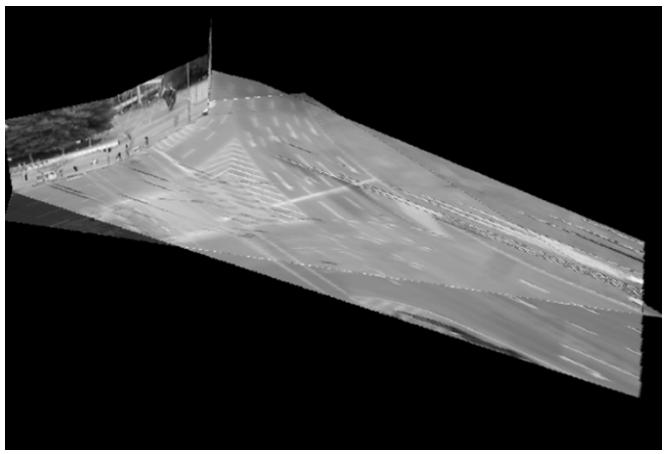


Fig. 5. Background reconstruction with multitexture ground plane and side plane objects.

IV. Static Background Processing

The 3-D model of the scene background consists of a ground plane of the traffic scene and additional side planes. These side planes either contain surrounding buildings or information that is far away from the place of interest. The original textures from all views are used to create the 3-D model. This is possible using camera calibration and homography information, which is estimated as described above. After the extraction of background images, the following processing stages are carried out:

A. Region-of-Interest Selection

In traffic surveillance scenarios, street and sidewalk areas are the most important parts, which are considered as part of a scene ground plane. The ground plane is typically surrounded by adjacent areas such as buildings etc, which are considered as part of side planes. Since the geometrical relationship of ground plane and side planes remains unchanged over time, we perform the geometrical modeling offline when setting up the system as follows: First the ground plane is selected with an interactive segmentation tool by manually drawing the appropriate area of interest as a polygonal contour in each original view. Then all adjacent side areas to that contour are presented and can be selected for modeling as side planes. These side areas are mapped onto planes that are perpendicular to the ground plane, as shown in the 3-D background model in Fig. 5.

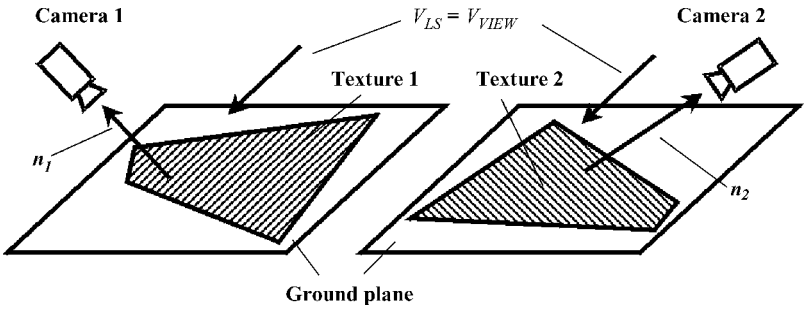


Fig. 6. Assignment of normal vector directions to different textures depending on original camera viewing directions.

In Fig. 5 the ground plane is composed of textures from all views and side plane textures from adjacent buildings from one view are added. One major problem that is also visible in the ground plane of Fig. 5 is the incorrect projection of static objects that are not part of the ground plane, e.g., traffic lights, flagpoles, or traffic signs. These elements cannot be eliminated in the initial segmentation-by-motion step since they are static. The elimination of such objects is currently under investigation using an interactive color/texture segmentation approach (watershed) that would also be performed offline when setting up the system.

B. Multiview Interpolation

After the selection of all regions-of-interest an interpolation of the ground plane textures from all views is done to create a common 3-D ground plane. Typically, large differences between camera positions and viewing angles of the scene cause different illumination of the same areas. In these cases, a texture might appear bright in one view, whereas in other views, it might appear completely dark. This effect has to be compensated to create a smooth transition between the original views during the user navigation through the scene. To preserve different lighting conditions for each individual view, all textures are kept separately in contrast to approaches, where one common texture is created before rendering.

Appropriate interpolation functionality of the single textures is already provided by today's graphics hardware and DirectX can be used to access these tools that enable interpolation and lighting compensation. Therefore, we use a DirectX-based rendering framework for the scene visualization. To enable interpolation, we use multitexture surfaces to model the ground plane as well as the moving objects. These multitexture surfaces allow mapping of a number of textures onto a single geometric primitive. Different normal vector directions $\boldsymbol{n}$ are assigned to each texture, depending on the original camera viewing direction. Light source direction $\boldsymbol{v}_{\mathrm{LS}}$ is identical to viewing direction $\boldsymbol{v}_{\mathrm{VIEW}}$ as illustrated in Fig. 6.

Then the surface lighting of each texture is determined by the combination of the actual position and viewing direction $\boldsymbol{v}_{\mathrm{VIEW}}$ and the normal vectors of the textures as illustrated in Fig. 7. The rendered view is automatically interpolated from all textures with their individual weight w depending on the actual position and viewing direction in the scene

$$w_{\mathrm{TEXTURE\ 1}} = -\boldsymbol{n}_1 \cdot \boldsymbol{v}_{\mathrm{VIEW}} \tag{4}$$

$$w_{\mathrm{TEXTURE\ 2}} = -\boldsymbol{n}_2 \cdot \boldsymbol{v}_{\mathrm{VIEW}}. \tag{5}$$

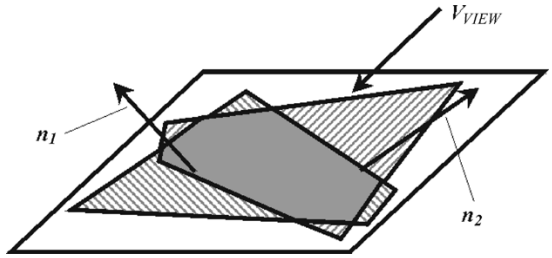


Fig. 7. Interpolation of intermediate view from multitexture surface.

This enables a smooth transition when navigating the scene. In Fig. 8, the background scene is shown from different viewpoints. Starting from the viewing position in Fig. 5, where ground plane textures are weighted equally, the viewing position is moved to the left and right, as shown in the left and right top pictures of Fig. 8, respectively. By moving the viewing position away from the middle, one of the textures becomes more visible than the other. When finally the original camera positions are reached, only the ground texture viewed from the corresponding camera is completely visible as shown in the bottom pictures of Fig. 8, whereas the texture views from other cameras are rendered transparent, i.e., they are not visible. Thus, smooth texture interpolation between different viewpoints is achieved.

V. Dynamic Object Tracking

For the tracking of dynamic objects, a framework was developed, which consists of a 2-D tracking stage in each view, followed by a 3-D tracking stage in the scene. Finally a 3-D object reconstruction is carried out. In the 2-D tracking stage object contours are described by closed B-splines based on [17]. The second tracking stage assigns the tracked 2-D objects from multiple views to 3-D objects at a higher semantic level and traces the 3-D trajectory with a linear Kalman filter [4]. Finally, a textured 3-D model is placed into the 3-D scene and its trajectory is updated each time step. The modeling starts with the segmentation described above, and continues with the following steps.

A. 2-D Tracking

The first step comprises the independent 2-D tracking of contours in each view. Two linear Kalman filters estimate the motion and shape of the contour. The objects are represented by convex polygons obtained in the motion segmentation step. To

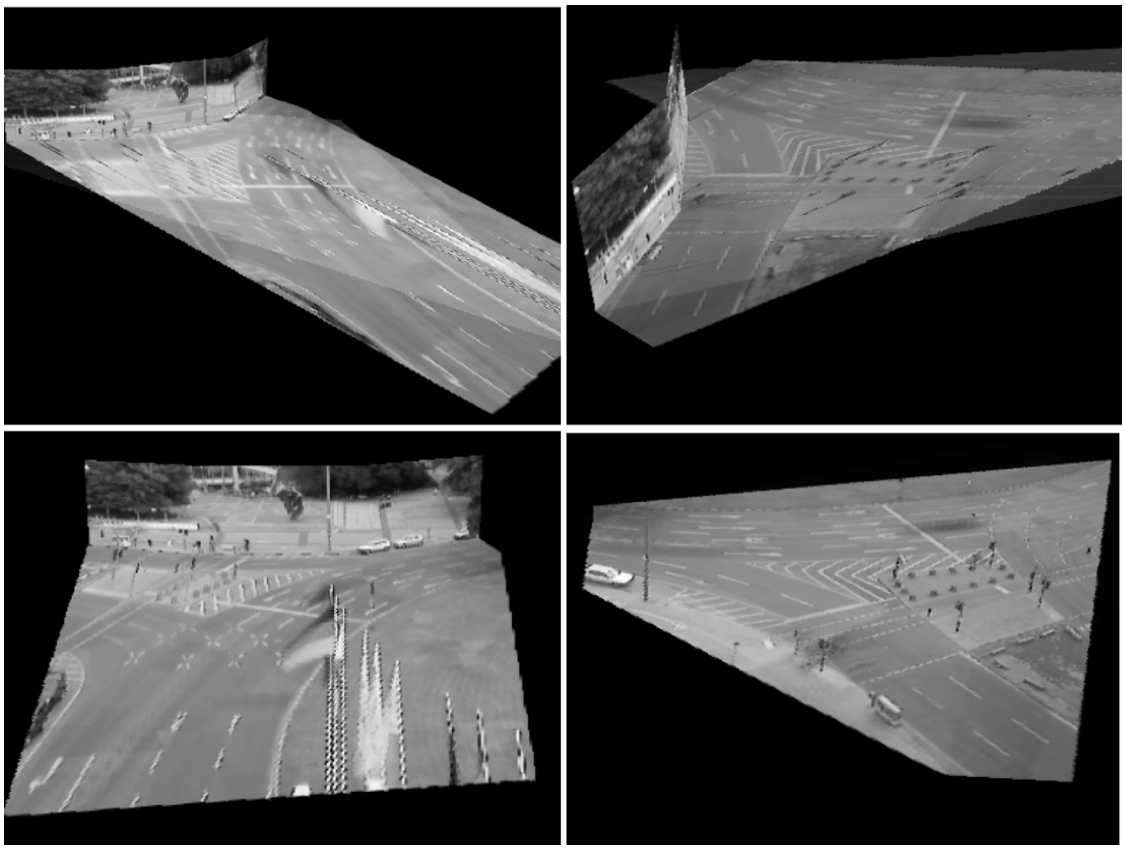


Fig. 8. Background scene from different viewpoints. Top: near the central viewpoint from the one shown in Fig. 5. Bottom: near the original camera viewpoints.

track these polygons over time, a fixed number of vertices is required, therefore, uniform cubic B-splines are used to approximate closed contours [2]. B-splines are invariant to translation, rotation and scaling, which is a necessary condition for applying affine transformations as described below.

The estimated shape state vector $\hat{\boldsymbol{x}}_{\mathrm{Shape},k}$ and the shape measurement vector $\dot{\boldsymbol{z}}_{\mathrm{Shape},k}$ at time instance k are specified by the N control vertices of a B-spline that approximates a closed contour

$$\hat{\boldsymbol{x}}_{\mathrm{Shape},k} = [[\hat{x}_1 \;\; \hat{y}_1] \;\cdots\; [\hat{x}_N \;\; \hat{y}_N]]^T|_k \tag{6}$$

$$\boldsymbol{z}_{\mathrm{Shape},k} = [[\widetilde{x}_1 \;\; \widetilde{y}_1] \;\cdots\; [\widetilde{x}_N \;\; \widetilde{y}_N]]^T|_k \tag{7}$$

where $[\hat{x}\;\hat{y}]_k$ are estimated and $[\tilde{x}_1\;\tilde{y}_1]_k$ are measured vertices. Equation (8) characterizes the prediction of the shape vector $\hat{\boldsymbol{x}}^{-}_{\mathrm{Shape},k}$ from the previous time instance (state update) before incorporating the measurement, using the matrix $\boldsymbol{B}_{\mathrm{Shape},k}$ which models the deterministic update of the state:

$$\hat{\boldsymbol{x}}^{-}_{\mathrm{Shape},k} = \hat{\boldsymbol{x}}_{\mathrm{Shape},k-1} + \boldsymbol{B}_{\mathrm{Shape},k}\,\hat{\boldsymbol{x}}_{\mathrm{Motion},k-1} \tag{8}$$

$$\boldsymbol{B}_{\mathrm{Shape},k} = \begin{bmatrix} \boldsymbol{I}_2 & \hat{\boldsymbol{x}}_{\mathrm{Shape},k-1,1} \\ \vdots & \vdots \\ \boldsymbol{I}_2 & \hat{\boldsymbol{x}}_{\mathrm{Shape},k-1,N} \end{bmatrix} \text{ with } \boldsymbol{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{9}$$

In the next step, the estimated shape vector $\hat{\boldsymbol{x}}_{\mathrm{Shape},k}$ is calculated incorporating the measurement vector $\boldsymbol{z}_{\mathrm{Shape},k}$. $\boldsymbol{K}_{\mathrm{Shape},k}$ represents the Kalman gain and is calculated taking into account the statistics of the system state and measurement noise processes

$$\hat{\boldsymbol{x}}_{\mathrm{Shape},k} = \hat{\boldsymbol{x}}^{-}_{\mathrm{Shape},k} + \boldsymbol{K}_{\mathrm{Shape},k}\left[\boldsymbol{z}_{\mathrm{Shape},k} - \hat{\boldsymbol{x}}^{-}_{\mathrm{Shape},k}\right]. \tag{10}$$

The relation of a contour between consecutive frames is approximated by an affine transformation. In our scenario vehicle rotation is in general restricted to rotation around the ground plane normal vector, which is limited between consecutive frames. We can, therefore, reduce the affine transformation to translation $\Delta\boldsymbol{t} = [t_x\, t_y]^T$ and scaling s. Other deformations are modeled as stochastic system input, which is handled robustly by the measurement update in the Kalman filter framework. With the center of gravity of a contour $\boldsymbol{p}_{C,k}$ we get an update of vertex positions $\boldsymbol{p}_{v,k+1}$ as

$$\boldsymbol{p}_{v,k+1} = s \cdot (\boldsymbol{p}_{v,k} - \boldsymbol{p}_{C,k}) + \Delta\boldsymbol{t} + \boldsymbol{p}_{v,k} \tag{11}$$

with $\Delta\boldsymbol{t}$ being the contour translation and s is the scale factor. With that the motion state vector $\hat{\boldsymbol{x}}_{\mathrm{Motion}}$ is given by

$$\hat{\boldsymbol{x}}_{\mathrm{Motion}} = [t_x \quad t_y \quad s]^T. \tag{12}$$

The measurement vector for motion $\boldsymbol{z}_{\mathrm{Motion},k}$ is modeled as difference of actually measured shape vertices $\boldsymbol{z}_{\mathrm{Shape},k}$ and previously estimated shape vertices $\hat{\boldsymbol{x}}_{\mathrm{Shape},k-1}$

$$\boldsymbol{z}_{\mathrm{Motion},k} = \boldsymbol{z}_{\mathrm{Shape},k} - \hat{\boldsymbol{x}}_{\mathrm{Shape},k-1}. \tag{13}$$

From this, the estimation vector $\hat{\boldsymbol{x}}_{\mathrm{Motion},k}$ is specified as

$$\hat{\boldsymbol{x}}_{\mathrm{Motion},k} = \hat{\boldsymbol{x}}_{\mathrm{Motion},k-1} + \boldsymbol{K}_{\mathrm{Motion},k} \times [\boldsymbol{z}_{\mathrm{Motion},k} - \boldsymbol{C}_{\mathrm{Motion},k}\hat{\boldsymbol{x}}_{\mathrm{Motion},k-1}] \quad (14)$$

with

$$\boldsymbol{C}_{\mathrm{Motion},k} = \begin{bmatrix} \boldsymbol{I}_2 & \hat{\boldsymbol{x}}_{\mathrm{Shape},k-1,1} \\ \vdots & \vdots \\ \boldsymbol{I}_2 & \hat{\boldsymbol{x}}_{\mathrm{Shape},k-1,N} \end{bmatrix} \quad \text{and} \quad \boldsymbol{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (15)$$

For initialization of an object, two consecutive frames with corresponding contours are required. To identify the same contour in both frames, a simple intersection of the frames is applied. The best fit is marked and recorded in a tracking list. Each tracked object is assigned with an internal label for its identification and further processing. This label is kept consistent for each contour that is tracked by the Kalman filter process from frame to frame. If an object is temporarily occluded by other foreground or background objects (e.g., trees, flagpoles, ...), its shape and motion will be predicted by the Kalman filter using previously estimated data. This process continues up to the point, where the initial object is again visible, its predicted position lies outside the image or a threshold is reached to restrict the number of prediction steps. The associated label is kept and reassigned, if the object is again visible. Otherwise the label is deleted from the object label list.

B. 3-D Tracking

The second part contains a common data fusion step followed by a Kalman filter for estimating the position and motion of an object in 3-D. The first task is the recognition of objects in all views. This is done by evaluating the locations of all tracked objects in all views using the previously estimated homography information. For a suitable fusion process we need to consider, that the initial object segmentation can be distorted due to shadow misclassification. Therefore, a robust approach was selected: Let M be the number of tracked objects in view labeled i, and N be the number of tracked objects in a second view labeled j. The center of gravity $\boldsymbol{p}_{C,m}$ of objects m in view i is projected into the second view j using the homography between these views $\boldsymbol{H}_{i,j}$. An error metric being the geometrical distance of this projected position to the centers of gravity of all objects n in the second view j is calculated. This metric is required, since the centers of gravity were calculated from possibly distorted contours. In our example, a distorted segmentation due to shadow influences resulted in a deviation of real object center and the center of the B-Spline contour of up to one third of an object's height. Therefore, the resulting projection errors occur mainly due to such segmentation problems rather than incorrect homography data. Using an error metric eliminates these projection errors. For a certain combination of objects m, n this metric can be expressed as

$$e_{m,n} = \|\boldsymbol{H}_{i,j} \cdot \boldsymbol{p}_{C,m} - \boldsymbol{p}_{C,n}\|. \quad (16)$$

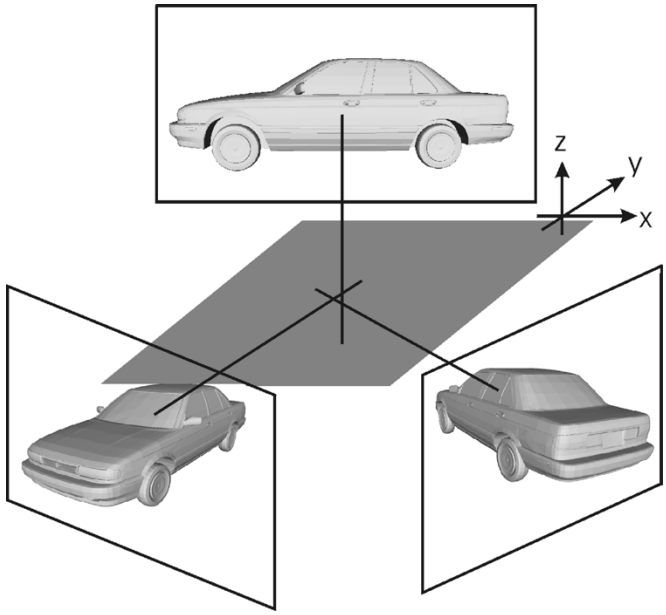


Fig. 9. Centers of gravity of 2-D objects are projected onto lines in 3-D, using camera calibration information.

This can be combined in an error matrix $\boldsymbol{E}_{i,j}$ for all objects tracked in view labeled i in relation with all objects in view labeled j

$$\boldsymbol{E}_{i,j} = \begin{bmatrix} e_{1,1} & \cdots & e_{1,N} \\ \vdots & e_{m,n} & \vdots \\ e_{M,1} & \cdots & e_{M,N} \end{bmatrix}. \quad (17)$$

To assign objects from one view to another the minimum error has to be located in each column, taking into account a maximum threshold to avoid wrong assignments due to segmentation and tracking errors. This is repeated for all combinations of views and the extracted information is integrated. Thus, all views of an object are associated with one unique object at a higher semantic level. Due to data fusion of multiple views, objects can be tracked even if they are occluded in other views.

The next step comprises the location and tracking of objects in 3-D. To position an object in 3-D the centers of projection from each available view are projected to lines in 3-D using camera calibration information [11] as shown in Fig. 9. These lines intersect the ground plane at certain positions, which should ideally be identical, but due to segmentation, tracking and sampling inaccuracies this situation is never given. Therefore, a simple least squares approach is used to calculate a point that minimizes the distance to all points of intersection. The coordinates of this calculated point are assigned to the object as center of gravity. This procedure is repeated for all objects.

Another linear Kalman filter is used to estimate the 3-D motion trajectory. The state vector $\boldsymbol{x}_{3D}$ is composed of the position and the temporal derivatives, whereas the measurement vector $\boldsymbol{z}_{3D}$ only consists of the calculated position

$$\boldsymbol{x}_{3D} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T \quad (18)$$

$$\boldsymbol{z}_{3D} = [x, y, z]^T. \quad (19)$$

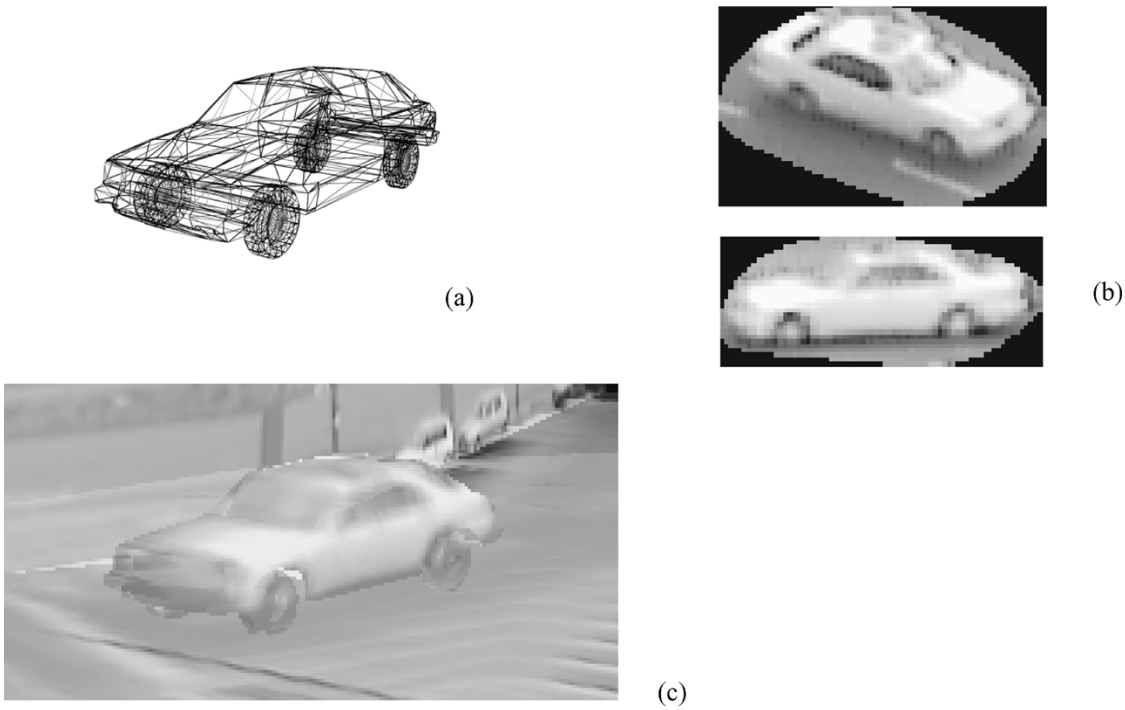


Fig. 10. (a) Synthetic 3-D object. (b) 2-D image patches obtained in tracking procedure. (c) Reconstructed vehicle placed in the scene background.

Using the temporal derivatives, the direction of movement of an object can be determined. As mentioned earlier in this paper, in our scenario the rotation of vehicles is restricted to rotation around the normal vector of the ground plane. Since we set the ground plane into the plane $z = 0$, we can neglect z and $\dot{z}$.

C. Dynamic 3-D Object Reconstruction

In the last step we assign a predefined 3-D model from a database to each tracked object. These 3-D models are positioned, scaled and oriented properly in the 3-D scene and the original textures from each view are mapped onto them. This yields the final 3-D representation of dynamic objects.

The position and motion direction have already been determined by the tracking procedure. We assume that vehicles move forward between successive frames. Therefore, the predefined 3-D model for a certain object has to be oriented in the same direction that was estimated in the tracking step. For that the predefined 3-D model is rotated by a matrix $\boldsymbol{R}$ for a given temporal derivative $(\dot{x}, \dot{y}, \dot{z})$ as follows:

$$\boldsymbol{R} = \begin{bmatrix} \frac{\dot{x}}{d} & -\frac{\dot{y}}{d} & 0 \\ \frac{\dot{y}}{d} & \frac{\dot{x}}{d} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{with} \quad d = \sqrt{\dot{x} + \dot{y}} \tag{20}$$

since the ground plane is located in the plane $z = 0$.

The last parameter to be set after position and orientation have been determined is the proper scale factor. Therefore, the size of the wire frame model is adjusted by projecting the artificial model into each view and comparing its 2-D bounding box with the bounding box of the segmented textures. The model is rescaled until it fits to the original textures. The last step applies the ground plane constraint to the 3-D object. Since the position of the object was calculated from possibly distorted 2-D contours, the position is now adjusted by placing the 3-D model on the ground plane.

After proper sizing, the original textures are mapped onto the object. Fig. 10 illustrates the 3-D reconstruction of a moving object, where a 3-D wire frame model and 2-D texture patches are connected to create the textured object. This object is finally integrated into the scene background, considering proper positioning and orientation on the ground plane. For the final models, the objective was to preserve original texture information. Therefore, lighting enhancement methods, e.g., radiometric self-calibration were not applied. This may result in visual distortion during the automatic interpolation of differently lighted textures, however the overlap of object textures is limited due to the camera positions being far apart. On the other hand objects strongly change their lighting conditions due to varying reflectance. Depending on the surface characteristics, strong reflection may suddenly occur in single areas or the entire objects, which would make it difficult to apply radiometric calibration methods that generate a constantly illuminated object texture during scene passage.

In contrast to vehicle modeling in 3-D, cyclists and pedestrians are modeled using a simple billboard technique. They are identified by their limited size and shape in comparison to vehicles, as reported in [20]. The textures from all views are mapped onto a plane perpendicular to the ground plane. This plane can only rotate around the z-axis as defined in Fig. 9, i.e., remaining always perpendicular to the ground plane, but facing the actual point of view in x- and y-direction. This simple technique can only be applied to very small objects, e.g., pedestrians or cyclists, since it leads to strong visual distortion due to the missing 3-D-information and sparse texture information.

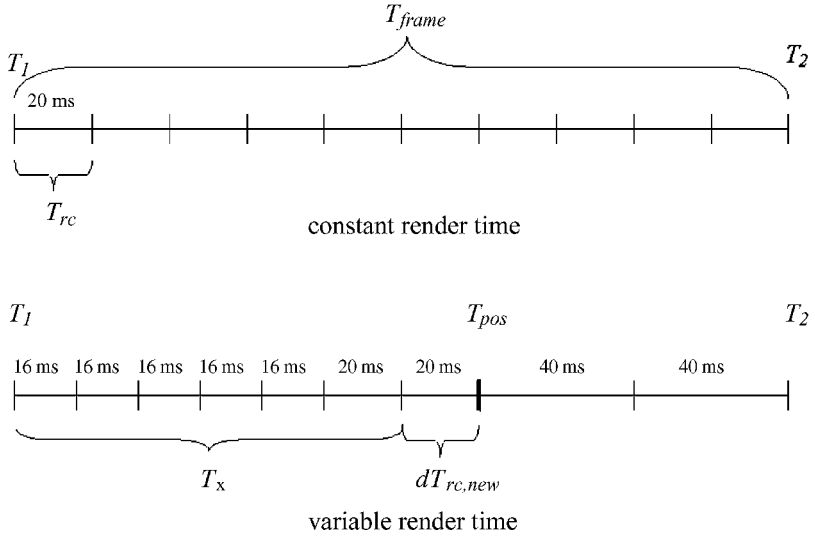


Fig. 11. Ideal constant render time intervals between two input frames (top) and example for realistic behavior of changing render time between two input frames (bottom).

VI. 3-D Scene Composition

Finally, the 3-D scene is composed of static background and dynamic objects. The different scene elements have different timing and complexity requirements. In the following we discuss the implications for the rendering framework. The scene consists of the following main elements.

A. Object Position

The position of each foreground object needs to be updated at each render cycle. This is the most time critical information but requires only very few parameters, i.e., the 3-D coordinates of the centers of gravity. A position update is required each render cycle, which is determined by the display system. The render cycle should be independent from the camera frame rate, since this can be very low in surveillance scenarios. A typical camera frame rate of around 5 Hz, would cause the objects in the 3-D scene to move rather jerky. Therefore, the position is interpolated for each render cycle between two camera frames. Consider the time between two frames T_1 and T_2 as T_{frame} and the time between two render cycles as T_{rc} as shown in Fig. 11 top.

If the render cycle time remains constant, the interval T_{frame} is split into N equally small intervals, where the object position is interpolated along a spline trajectory at ach time stamp T_{pos}:

$$T_{\mathrm{pos}} = T_1 + n \cdot T_{\mathrm{rc}} \text{ with } n = 1 \ldots N-1 \text{ and } N = \frac{T_{\mathrm{frame}}}{T_{\mathrm{rc}}}. \quad (21)$$

The problem is that the render time T_{rc} depends on a number of system parameters, e.g., processor utilization and thus can vary significantly between two camera frames. In Fig. 11 bottom a varying render frame rate is shown. Therefore, the above approach needs to be adapted by introducing a time-varying render cycle time dT_{rc} and the time stamp T_{pos} for object position interpolation is calculated as follows:

$$T_{\mathrm{pos}} = T_1 + T_{\mathrm{acc}} + dT_{\mathrm{rc,new}} \quad \text{with} \quad T_{\mathrm{acc}} = \sum_{\forall i} dT_{rc,i}, \quad \text{if} \quad T_{\mathrm{acc}} + dT_{\mathrm{rc,new}} < T_{\mathrm{frame}}. \quad (22)$$

Here, T_{acc} represents all accumulated previous render times dT_{rc} within the current frame interval T_{frame} and $dT_{\mathrm{rc,new}}$ the current render frame time (compare Fig. 11 bottom). Note, that a new object position at time T_{pos} is only interpolated as long as the accumulated render frame time T_{acc}, including $T_{\mathrm{rc,new}}$, does not exceed the frame time T_{frame}.

With this adaptation, varying frame rates, as well as render cycle rates can be handled and a smooth interpolation is provided.

B. Object 3-D Geometry

A 3-D geometry needs to be assigned, if an object occurs the first time by either entering the scene or appearing after occlusion. The 3-D model is selected from a limited number of synthetic objects from a database. A further improvement would be an automatic object selection based on 2-D shape properties of the segmented object textures from multiple views.

C. Object Texture

A texture needs to be updated, if an object enters the scene or if significant new information is revealed in case of exposure or a change of direction. Currently, texture information is only assigned when the object occurs the first time. A further improved algorithm could also analyze the texture of a moving object during its entire lifetime within the scene to identify, whether significant new information becomes visible. This situation occurs for example, if a vehicle changes direction at a crossing and reveals image information that was previously invisible in a certain camera view.

D. Background

This is the scene part with the least demanding real-time requirements. Only if the whole scene drastically changes, e.g., due to strong lighting changes, an update of texture is required. Therefore, the modeling algorithm can be performed completely offline. If the topology of the observed scene changes, a new offline modeling has to be performed.

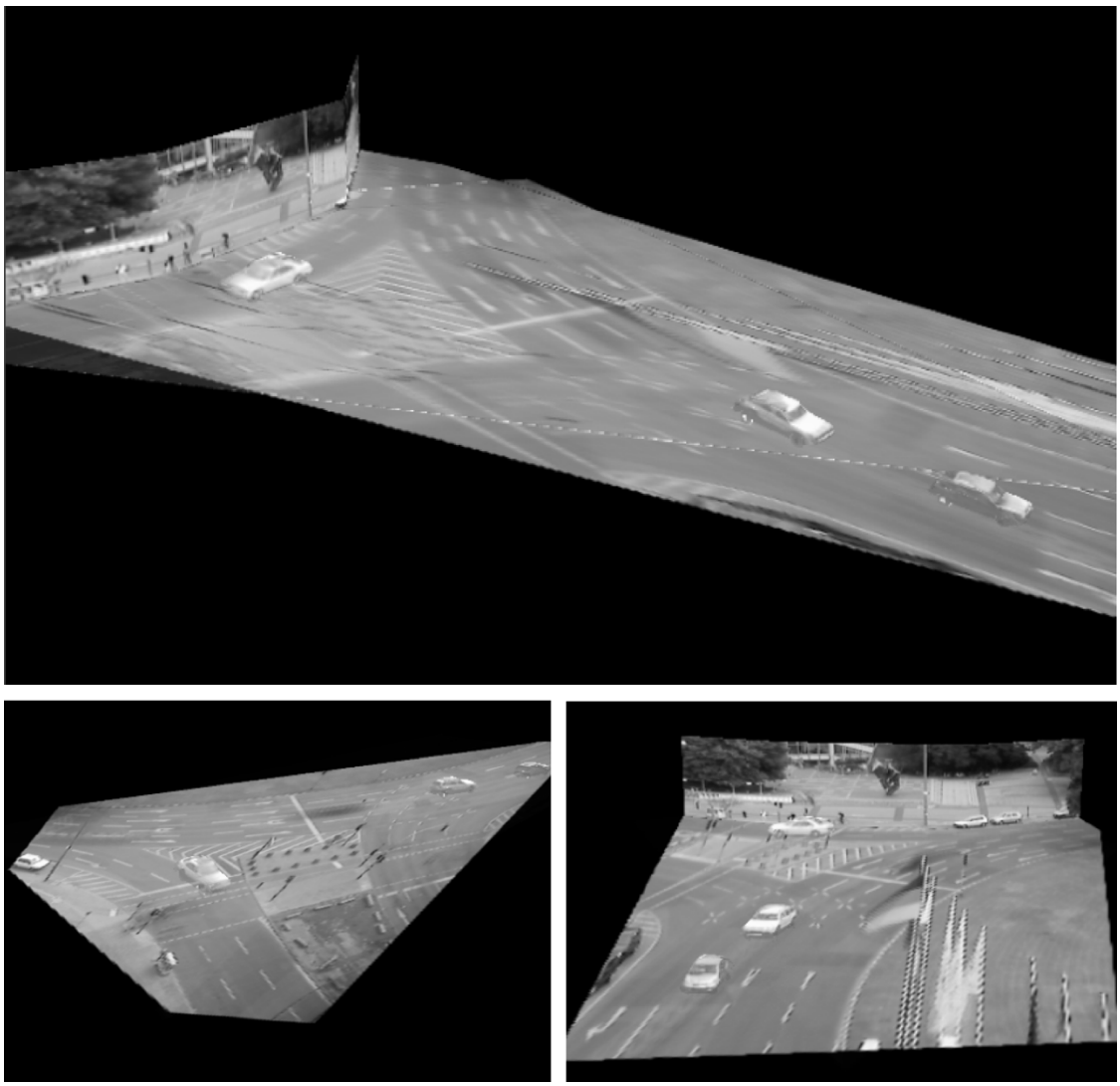


Fig. 12. 3-D scene views with reconstructed dynamic objects from different directions.

Since only the object position needs to be updated at every render cycle, more complex algorithms can be applied for processing the other scene elements such as 3-D object fitting, texture adaptation and, most of all, for background modeling. Furthermore, since intermediate object positions are interpolated from their motion trajectory, the render cycle time becomes independent of the camera frame rate and the scene can be rendered as fast as possible. The position interpolation for every render cycle is also useful in cases of jitter of the camera frame rate. Fig. 12 shows the final scene with three reconstructed cars. Again the central view is shown in the large image, whereas in the small images below a viewpoint near the original camera positions was selected.

Besides the obtained results, there are some drawbacks of the proposed approach. First, the scene needs to be fully calibrated and exact homographies have to be available. Otherwise, ground planes do not fit correctly, as mentioned in Section III. One major problem results from incorrect segmentation, especially from shadows that are segmented as foreground objects and, therefore, disturb the visual impression of the rendered textures. Although even strong changes in lighting conditions have no influence on background processing, however, they influence the segmentation process by changing shadow appearance. Currently, the segmentation routine cannot separate vehicles moving close together at the same velocity until the Kalman filter tracking approach splits these objects once an object is correctly identified.

VII. Summary and Conclusions

In this paper we presented a system that allows dynamic 3-D scene reconstruction from a limited number of input cameras. To cope with the limitations resulting from such a setup *a priori* knowledge about the scene is exploited, such as plane background areas and ground plane constraint for foreground objects. Further camera calibration is assumed to be available. The system includes video transmission from all cameras to a central client via a streaming architecture that provides scalability features and remote control of all camera servers from the central client.

After transmission, the video is segmented to extract moving objects and to create a background image for each view. This

approach is well suited for the shown traffic surveillance scenario, since static cameras are assumed and, therefore, a stationary background is obtained. After segmentation, static background and dynamic foreground are processed separately. From the background images ground planes and additional side planes are selected to create the 3-D background geometry. Ground plane textures are mapped onto a common 3-D plane to create a multitexture surface, which enables automatic weighting of the single textures according to the actual viewpoint and direction, when navigating the scene. This functionality is automatically performed by the graphics hardware.

Dynamic objects are processed by mapping all textures of an object onto a common synthetic 3-D model. The model is selected from a database by comparing its 2-D projections into the initial views with the original textures. For smooth animation, the object position needs to be updated each render cycle, which is often much higher than the typical frame rate of the input surveillance cameras. Therefore, object positions have to be interpolated along the estimated 3-D motion trajectory. Finally static and dynamic parts are combined to create the final scene, which allows free navigation for better traffic investigation and 3-D analysis.

We created a system with all components for traffic surveillance and 3-D reconstruction from a few cameras and demonstrated the appropriate operation on today's hardware platforms. Further improvement of the component is still possible, including better segmentation, automatic 3-D model selection and dynamic texture updates in case of revealed image content. The usage of this system is not restricted to traffic surveillance. In can be used for any application where a reliable dynamic 3-D scene reconstruction from a limited number of original views is necessary.

REFERENCES

[1] L. Alvarez, J. Sanchez, and J. Weickert, "Reliable estimation of dense optical flow fields with large displacement," *Int. J. Comput. Vis.*, vol. 39, no. 1, pp. 41–56, 2000.

[2] R. H. Bartles, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. San Francisco, CA: Morgan Kauffmann, 1987.

[3] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *Proc. Computer Vision and Pattern Recognition*, 1997, pp. 495–501.

[4] J. Black and T. Ellis, "Multi camera image tracking," in *Proc. 2nd IEEE Int. Workshop PETS*, Kauai, HI, Dec. 2001, CD-ROM.

[5] R. Collins *et al.*, "A system for video surveillance and monitoring," Robotics Institute, Carnegie Mellon Univ., Tech. Rep. no. CMU-RI-TR-00-12, May 2000.

[6] D. J. Dailey and L. Li, "Video Image Processing to Create a Speed Sensor," University of Washington, Seattle, WA, ITS Research Program, Final Research Report, 1999.

[7] C. Fehn, E. Cooke, O. Schreer, and P. Kauff, "3-D analysis and image-based rendering for immersive TV applications," *Signal Process.: Image Commun. J., Special Issue on Image Processing Techniques for Virtual Environments and 3-D Imaging*, vol. 17, no. 2, pp. 705–715, Oct. 2002.

[8] T. Frank, M. Haag, H. Kollnig, and H.-H. Nagel, "Tracking of occluded vehicles in traffic scenes," in *Proc. 4th Eur. Conf. Computer Vision 1996 (ECCV'96)*, Cambridge, U.K., Apr. 1996, pp. 15–18.

[9] D. Gutchess, M. Trajkovic, E. Cohen-Solal, D. Lyons, and A. Jain, "A background model initialization algorithm for video surveillance," in *Proc. Int. Conf. Computer Vision*, Vancouver, Canada, Jul. 2001, pp. 733–740.

[10] M. Haag and H.-H. Nagel, "Combination of edge element and optical flow estimates for 3-D-model-based vehicle tracking in traffic image sequences," *Int. J. Comput. Vis.*, vol. 35, no. 3, pp. 295–319, 1999.

[11] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[12] F. Heimes and H.-H. Nagel, "Real-time tracking of intersections in image sequences of a moving camera," *Eng. Applicat. Artificial Intell.*, vol. 11, pp. 215–227, 1998.

[13] E. Ichihara and Y. Ohta, "Naviview: visual assistance using roadside cameras-evaluation of virtual views," in *Proc. IEEE Intelligent Transportation Systems*, 2000, pp. 322–327.

[14] *Text of ISO/IEC FDIS 14 496-2*, Document no. N2502, Oct. 1998.

[15] W. Kasprzak, "Adaptive road recognition and ego-state tracking in the presence of obstacles," *Int. J. Comput. Vis.*, vol. 28, no. 1, pp. 5–26, Jun. 1998.

[16] D. Koller, K. Daniilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *Int. J. Comput. Vis.*, vol. 10, no. 3, pp. 257–281, 1993.

[17] D. Koller, J. Weber, and J. Malik, "Robust Multiple Car Tracking With Occlusion Reasoning," University of California at Berkeley, Tech. Rep. no. UCB/CSD 93/780, 1994.

[18] M. Levoy and P. Hanrahan, "Light Field Rendering," in *Proc. ACM SIGGRAPH*, Aug. 1996, pp. 31–42.

[19] A. N. Rajagopalan and R. Chellappa, "Vehicle detection and tracking in video," in *Proc. Int. Conf. Image Processing*, vol. 1, 2000, pp. 351–354.

[20] P. Remagnino *et al.*, "An integrated traffic and pedestrian model-based vision system," in *Proc. Brit. Machine Vision Conf.*, vol. 2, 1997, pp. 380–389.

[21] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using Kalman-filtering," in *Proc. Int. Conf. Recent Advances in Mechatronics*, 1995, pp. 193–195.

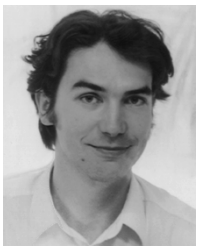
[22] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *Int. J. Comput. Vis.*, vol. 35, no. 2, pp. 151–173, 1999.

[23] R. Szeliski, "Robust shape recovery from occluding contours using a linear smoother," *Int. J. Comput. Vis.*, vol. 28, no. 1, pp. 27–44, Jun. 1998.

[24] P. H. S. Torr and D. W. Murray, "The development and comparison of robust methods for estimating the fundamental matrix," *Int. J. Comput. Vis.*, vol. 24, no. 3, pp. 271–300, September 1997.

Karsten Müller (M'98) received the Dipl. Ing. degree in electrical engineering from the Technical University of Berlin, Berlin, Germany, in 1997.

He has been with the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut, Berlin, Germany, since 1997, as a Member of the Research Staff, where he was working on projects focused on motion and disparity estimation, incomplete 3-D surface representation and coding, combined 2-D and 3-D object description, viewpoint synthesis and 3-D scene reconstruction with multitexture surfaces. His research interests are mainly in the field of representation, coding and reconstruction of 3-D scenes in free viewpoint video scenarios, multiview applications and combined 2-D/3-D similarity analysis. He has been involved in MPEG activities, where he contributed to visual MPEG-7 descriptors and MPEG-4 3-D scene representation. Currently, he is involved in applications for time-consistent 3-D mesh extraction and efficient coding.



Aljoscha Smolić received the Dipl.-Ing. degree in electrical engineering from the Technical University of Berlin, Germany, in 1996 and the Dr.-Ing. degree in electrical and information engineering from Aachen University of Technology (RWTH), Aachen, Germany, in 2001.

He joined the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institute (HHI), Berlin, Germany, as student in 1994. In 1996, he became a Research Assistant and has been employed as Project Manager since 2001. He has been involved in several national and international research projects, where he conducted research in various fields of video processing, video coding, computer vision and computer graphics. In this context, he has been involved in ISO standardization activities where he contributed to the development of the multimedia standards MPEG-4 and MPEG-7. In current projects he is responsible for research in video coding, object recognition and tracking, visual surveillance, 3-D object and scene reconstruction, free viewpoint video, and video-based rendering.

Dr. Smolić received the "Rudolf-Urtlel-Award" of the German Society for Technology in TV and Cinema (FKTG) for his dissertation in 2002. Frequently, he serves as reviewer for various IEEE and other journals and conferences. He co-chairs the MPEG ad hoc group on 3DAV.

Michael Dröse (M'03) was born in Neustadt am Rübenberge, Germany, in 1976. He received the Dipl.-Ing. degree in computer engineering from the Technical University of Berlin, Berlin, Germany, in 2003.

He joined the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut (HHI), Berlin, Germany, as a student in 2002, where he was involved in the development of algorithms for 3-D reconstruction of traffic scenes. Since 2003, he has been a Visiting Researcher at Nagoya University, Nagoya, Japan. Currently, he develops algorithms for multiview image interpolation for free viewpoint video applications. His research interests include computer vision, image processing, multiview imaging, compression and communication, image-based rendering and computer graphics.

Patrick Voigt received the Dipl.-Ing. degree in computer engineering from the Technical University of Berlin, Germany, in 2003.

He joined the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut (HHI), Berlin, Germany, as a student in 2002, where he was involved in the development of algorithms for video streaming in Single Server/Multi Client applications. In 2003, he joined Siemens AG as a sales engineer where he is currently responsible for power plant maintenance. His research interests include image processing, video coding, and video streaming.

Thomas Wiegand received the Dr.-Ing. degree from the University of Erlangen-Nuremberg, Nuremberg, Germany, in 2000 and the Dipl.-Ing. degree in electrical engineering from the Technical University of Hamburg-Harburg, Hamburg, Germany, in 1995.

Currently, he is the Head of the Image Communication Group in the Image Processing Department of the Heinrich Hertz Institute, Berlin, Germany. From 1993 to 1994, he was a Visiting Researcher at Kobe University, Kobe, Japan. In 1995, he was a Visiting Scholar at the University of California at Santa Barbara, where he started his research on video compression and transmission. Since then, he has published several conference and journal papers on the subject and has contributed successfully to the ITU-T Video Coding Experts Group (ITU-T SG16 Q.6-VCEG)/ISO/IEC Moving Pictures Experts Group (ISO/IEC JTC1/SC29/WG11-MPEG)/Joint Video Team (JVT) standardization efforts and holds various international patents in this field. From 1997 to 1998, he has been a Visiting Researcher at Stanford University, CA, and served as a consultant to 8×8, Inc., Santa Clara, CA. His research interests include image and video compression, communication and signal processing as well as vision and computer graphics.

In October 2000, Dr. Wiegand was appointed as the Associated Rapporteur of the ITU-T VCEG. In December 2001, he has been appointed as the Associated Rapporteur/Co-Chair of the JVT that has been created by ITU-T VCEG and ISO/IEC MPEG for finalization of the H.264/AVC video coding standard. In February 2002, he was appointed as the Editor of the H.264/AVC video coding standard.