Video-Based Rendering Of Traffic Sequences

Cedric Vanaken, Tom Mertens and Philippe Bekaert

Hasselt University — Expertise Centre for Digital Media
and transnationale Universiteit Limburg — School of Information Technology
Wetenschapspark 2, 3590 Diepenbeek, Belgium
{cedric.vanaken, tom.mertens, philippe.bekaert}@uhasselt.be

ABSTRACT

Video-based rendering is a viable alternative to traditional realistic image synthesis techniques. It avoids the burden of time-consuming modeling and expensive global illumination simulation. In this paper we propose a video-based synthesis and animation system for fixed viewpoint scenes that feature rigid objects. We examplify this by using traffic video sequences. In a first stage, we extract vehicles and their trajectories from the example footage using an intuitive semi-automatic segmentation technique. Subsequently, the resulting vehicle "sprites" are dimensionally reduced using PCA in order to effectively extrapolate trajectories from incomplete sequences. Background, occlusions and shadows cast by vehicles are extracted as well. We show that convincing new footage can be synthesized readily from a single input video. Any number and variety of cars can be inserted, and their trajectories can be edited to simulate such traffic scenarios as lane changes and traffic jams.

Keywords: Video-based Rendering, Animation, Traffic, Video Sprites.

1 INTRODUCTION

Video-based rendering methodologies have proved to be adept at synthesizing photo-realistic video sequences from sparse real world data, e.g. Schödl et al. [SSSE00]. In the same spirit, we present a novel technique to synthesize traffic scenes. Given an input traffic video, we are able to reproduce a traffic scene from the same viewpoint in which the configuration and trajectories of the vehicles have been altered by a user.

The main application of our technique is the validation and training of camera-based traffic analysis, e.g. for accident, queue and presence detection [Tra]. These systems cannot afford to have a high margin of error and thus require a wide variety of initial test sequences. Because these sequences are unique for each camera placement, they usually have to be acquired by shutting down highways and filming all desired scenarios *in situ*. This is an expensive and time-intensive task. As an alternative, one might synthesize video sequences directly using traditional modeling and global illumination techniques [DBB03]. However, this is an overwhelming task, both in terms of manual labor and computational requirements. Moreover, one would need to achieve a degree of realism that is hardly practical using current modeling and rendering tools (e.g. to reproduce natural landscapes, weathered surfaces, etc...). Also, it is unclear how imperfections of the camera system would be simulated. By using recorded footage from the targeted camera system, we achieve both goals immediately. Aside from validation of camera-based traffic detection systems, we believe that the techniques developed for this particular problem are general enough to be applied to video-based sprite animation [SE02].

Akin to Debevec et al.'s image based modeling approach [DTM96], one might construct approximate geometry for each vehicle of interest. However, this will turn out to be a labor-intensive task, since this process has to be carried out for each car separately. Alternatively one might come up with a parameterized model for vehicle geometry which can be fitted to the image data [BV99]. It is unclear whether such a model can be general enough to deal with the wide variety of shapes found in real life.

The main challenge is to extract vehicle sprites from the input footage and resynthesize them in a meaningful and controllable fashion [SSSE00, SE02]. These sprites are 2D images that represent an object of interest that can afterwards be integrated at different locations in the input scene, or inserted into a novel scene. We assume vehicle sprites travel along a straight path, while Video Sprites [SE02] aims at reconstructing arbitrary motions. This prior information is exploited in our segmentation and easily allows for parameterizing a vehicle's trajectory and appearance. In contrast, Video Sprites are a non-parametric representation based on searching and copying the most suitable frame from the input data. The parameterized representation facilitates easy extrapolation of incomplete trajectories (e.g. when the vehicle is not completely visible), and is friendly toward storage.

Conference proceedings ISBN 80-86943-03-8
WSCG'2006, January 30 - February 3, 2006
Plzen, Czech Republic.

In addition, we need to extract the background, which we assume to be static, and deal with possible occlusions along a vehicle's trajectory (e.g. caused by a bridge or lamp post). Schödl et al. [SSSE00, SE02] record sprite images and accompanying shadows using chroma keying. In our setting, this information cannot be extracted under such controlled conditions.

Jojic et al.'s video sprite model [JF01] copes with inter-sprite occlusions efficiently. For our purposes this is less of an issue, but static obstructions like lamp posts have to be dealt with.

The outline of our system is presented in Figure 1.

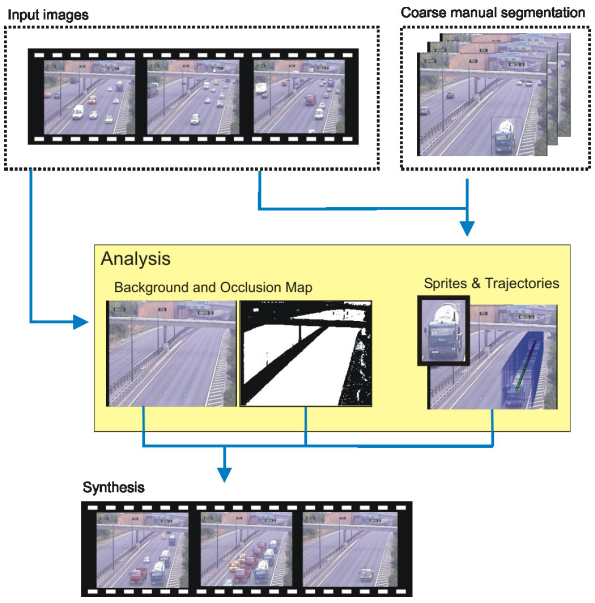

Figure 1: Outline of our traffic animation system. Starting from an input video and some user assistance, the analysis phase outputs a background, an "occlusion map" and extracted vehicles with associated trajectories. This data is used as input for the synthesis phase, where the vehicles are drawn onto the background to obtain a new animated video.

The rest of the paper is organized as follows. We start by reviewing related work in the areas of traffic detection techniques and video based animation systems. Section 2 gives an outline of our system. Section 3 presents our results and discusses practical issues. Finally, in section 4 we will present our conclusions and suggestions for future work.

1.1 Related Work

Video-based representation, rendering and animation techniques have received increasing interest in the graphics and vision community in the past few years. A recurring technique is rearranging frames in an input video to create novel footage, either globally on a per-frame basis [BCS97, SSSE00, AZP+05], or locally on a per-sprite basis [SSSE00, SE00, SE02]. This approach allows for efficient animation of e.g. natural phenomena or animals. Our approach differs by not reordering frames but rather building a simple parametric motion and appearance model for vehicle sprites, similar in spirit to Fitzgibbon [Fit01]. Consequently, less input is required and this representation even facilitates extrapolation.

Layered video models [WA94, JF01] automatically extract layered sprites and moving parts from input footage. In our particular problem the layers are fixed: background, vehicle sprites and occluders (such as a lamp post). Background and occluders are automatically recovered, while a user assisted process is employed to extract the vehicles.

The flow-based video synthesis technique [BSHK04] analyzes the motion of textured particles in the input video along user-specified flow lines, and synthesizes video of arbitrary length by enforcing temporal continuity along a second set of user-specified flow lines. The main difference between this approach and ours is that we not only redraw input pixels, but also capture and reuse the entire appearance of the objects in the input video.

Auto-regressive stochastic processes [CV05, CV06] model traffic flow from video. This method does not require segmentation or tracking, but lacks the per-vehicle control that our approach offers.

Segmentation and tracking of vehicles is a central problem in traffic analysis [ZCSP03, CGPP00, CBMM98, KM03], which has to be fully automated. We opted for a simple and robust semi-automatic system, though in a more general settings, these techniques could be used as well.

2 SYSTEM OVERVIEW

Our system consists of an analysis stage and a synthesis stage, which will be detailed in the following sections.

2.1 The Analysis Stage

The analysis stage extracts a background image, an occlusion map and vehicle sprites together with their trajectories.

Background We obtain an appropriate background as the per-pixel median intensity along the time dimension [GASK95]. This simple technique works very well on our video sequences, which have virtually static backgrounds. This rendered the implementation and testing of other, more complicated, techniques (e.g. [SG99]) unnecessary for us. An example of a calculated background image is shown on the left side of Figure 3.

Occlusion Map The occlusion map indicates which pixels remain unchanged during the entire length of the input video. Those pixels that fluctuate significantly w.r.t. some threshold T can be considered "possibly foreground", here the background should always be

Figure 2: Windows drawn around a particular vehicle by the user, with on the right a calculated trajectory for the vehicle.

drawn behind the sprites. A static pixel will either be an occluder (for example the bridge in Figures 2 and 3) or just background, and should be drawn on top of the sprite layer. We empirically estimated T to an intensity value of 0.012 in our experiments. If necessary, the occlusion map can be edited easily with any image editing tool to clean any impurities. We have made use of this feature to correct a very small number of pixels that slipped through the threshold. See Figure 3 for an example occlusion map.

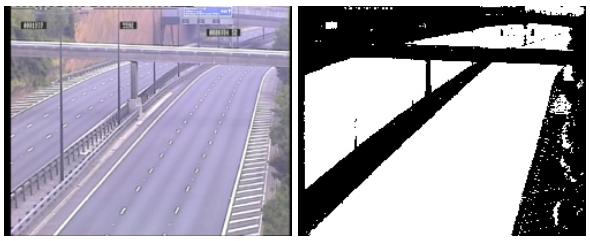

Figure 3: Left: extracted background image from an input video. Right: occlusion map. Static pixels (occluder or background) are black and fluctuating pixels (possibly foreground) are white.

Segmentation Extracting a vehicle from the input video is a semi-automatic process, which starts with a small amount of user interaction. The user takes three frames in which a particular vehicle can be seen at different positions: one frame where the vehicle has initiated its trajectory, one where it is approximately half way, and one near the end. The user indicates a window around the vehicle in each of these three frames. The content of this window captures all relevant information about the vehicle: its appearance and surrounding illumination effects (i.e. shadows). This window does not need to be drawn very precisely, it is only necessary that the vehicle and illumination effects are included and that no other vehicles or parts of other vehicles appear inside the window. The position of the window also defines where the vehicle is located at a known instance in time (fig 2).

Vehicle Sprite Appearance Model In this section we detail the vehicle appearance model, which is based on the user masks and pixel intensities in the window.

Assuming that the vehicle moves at a near constant speed and travels along a fairly straight line, we can interpolate the windows for the intermediate frames in time (t) by fitting the following simple parametric function to each of their corner positions x:

$$x(t) = \frac{(a \times t) + b}{t + c}$$

Thus, we can model the vehicle's trajectory with a simple function that takes into account perspective projection.

In the next step, the user draws a mask for the vehicle in the three initially indicated windows (see Figure 4).

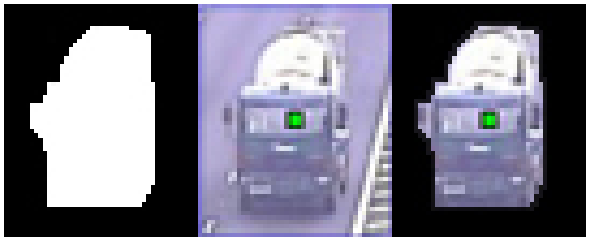

Figure 4: Left: user drawn mask. Middle: vehicle for which the mask is drawn. Right: result of the mask operation.

In a more general setting, this manual intervention may be replaced with an automatic technique.

In order to obtain the masks for intermediate frames, we convert the masks to polygonal shapes and interpolate them. However, the number of vertices for different masks is not guaranteed to be the same. Inspired by the morphing algorithm of Kent et al. [KCP92], we solve this problem as follows. Let A, B and C be the 3 user defined mask polygons. Polygon A is placed on B and we project and add each of A's vertices to B. This step is repeated from A to C, B to A, etc...We are able to reconstruct the mask for each frame simply by rasterizing the corresponding interpolated polygon. In addition, we soften the edges of the binary masks using convolution to avoid possible seams between vehicle and background. The results of this simple matting technique, are qualitatively the same for our input videos as results that can be achieved by using more advanced matting techniques [CCSS01, SJTS04].

The vehicle's appearance inside the mask mainly evolves due to a small relative rotation w.r.t. the camera and environmental illumination (e.g. from street

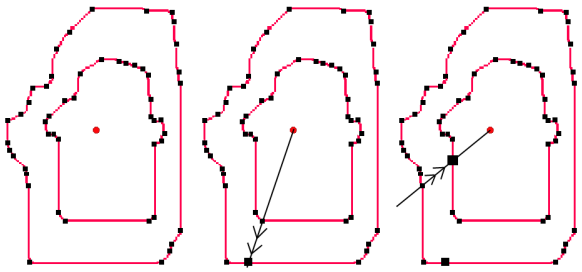

Figure 5: Left: source polygon placed on target polygon. Middle: extra interpolation step. Right: inverse direction extra interpolation step.

lights). In addition to pixels belonging to the vehicle itself, shadow information is extracted by dividing the window content by the background. This yields a background-invariant multiplicative "shadow map", as seen in Figure 6. Naively storing full windows is dependent on the background, and might corrupt the appearance when altering the location of a vehicle. We therefore also blur out all detail surrounding the mask in the vehicle sprite (not done for the shadow map).

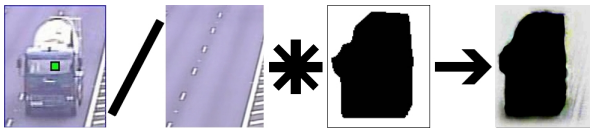

Figure 6: Shadow map computation (right image is contrast enhanced).

Given the full appearance of a vehicle at each frame, we reduce it using PCA. Before we do this, we need to scale our masks, sprites and shadow maps to a common size, for which we take the maximum window size of the indicated windows. Then, the polygon vertices relative to the midpoint, the sprites' pixel intensities and the shadow map intensities are each concatenated into a long vector with dimensionality V. Usually V is quite large (e.g. 10^5) while the number of frames N is small (e.g. 100), yielding a $V \times N$ matrix D that features an impractically large covariance matrix. We follow Matusik's variant on PCA [Mat03] to circumvent this problem. More precisely, we perform an eigenvalue decomposition of the $N \times N$ dimensional covariance matrix of the zero mean D_0. The obtained eigenvectors (*eivec*) are sorted by ascending eigenvalues (*eival*), while the vectors with very low eigenvalues are thresholded. Finally, our PCA representation of the appearance of the vehicles consists of:

- transformation matrix: $T = D_0 \times eivec \times (eival^{-\frac{1}{2}})$
- PCA coefficients matrix: $X = T \times D_0$
- mean image of the frames: μ

The eigenvectors corresponding with the highest eigenvalues contain the coarse details, while subsequent values express the finer details. Reconstruction quality can be traded off against the level of compression by discarding eigenvectors that have a small eigenvalue associated with it. We found that the 20 (out of 10^5 in total) largest eigenvalues and accompanying eigenvectors are sufficient to reconstruct a sprite's appearance.

Usually, we cannot capture a vehicle's full trajectory on the screen, because the sprite is cropped at the edge of the screen near the beginning and end, or possibly occluded. The inclusion of these sprites in the input data will result in a somewhat distorted PCA result. Therefore, we don't take these frames into account, but solve this problem by extrapolating the PCA coefficients that parameterize the vehicle's appearance, using autoregression [Fit01, NS04]. These "fitted" PCA coefficients will be used in the synthesis step to complete the trajectory of the vehicles at points in time where the vehicle was not segmented.

A comparison of our extrapolation technique with ground truth is shown in Figure 7. From this we can conclude that the shape of the vehicle is approximated fairly well, while interlacing artefacts are partly smoothed out by the PCA algorithm.

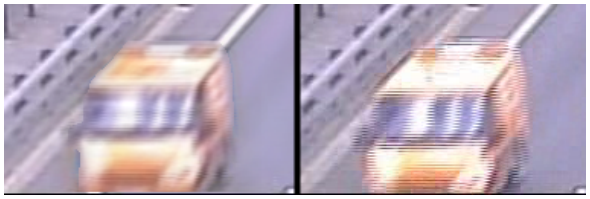

Figure 7: Left: extrapolated vehicle at the end of its trajectory. Right: Ground truth.

2.2 The Synthesis Stage

The synthesis step is relatively straightforward. We initialize the frame buffer with the background image. For each vehicle and a given frame we need to perform the following steps, of which only number 1 needs a more in-depth explanation.

1. Compute the vehicle's appearance.
2. Scale the vehicle to its original window size
3. Paste the vehicle onto the frame buffer
4. Multiply the frame buffer with the shadow map.

The sprites are rendered in back to front order to correctly resolve inter-vehicle occlusion.

Step 1 consists of rebuilding the appearance of the vehicle out of the PCA representation. We start off with the mean image μ, and iteratively add more detail. We do this by reshaping the next-in-line eigenvector from the transformation matrix T to an eigenimage, multiplying it with its associated PCA coefficient and adding

the result to the mean image. Figure 8 shows some examples of intermediate results of this iterative process. We have empirically fixed the number of used PCA levels to 20.

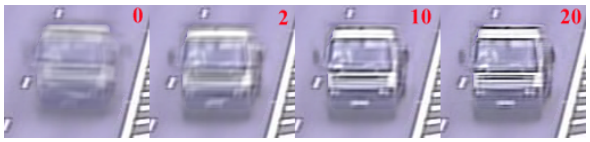

Figure 8: Iterative reconstruction from PCA data: the first image shows the mean appearance of the vehicle without extra detail added to it. The next images show the mean image with respectively 2, 10 and 20 PCA levels added. It is clearly visible that the first PCA levels contain the most important information, while the lower levels add only little detail.

3 RESULTS AND DISCUSSION

We extracted five different vehicles from a short input video and out of this data synthesized four videos that display different animated traffic situations:

- normal traffic
- a traffic jam
- a vehicle that suddenly stops while the rest of the traffic continues on the other lanes
- a vehicle that drives backwards while the rest of the traffic drives normally on the other lanes

The extracted vehicles that were used for the rendering of these new videos are shown in Figure 9.

Figure 9: Extracted vehicles, scaled to the same height for presentation purposes.

The amount of user-interaction involved in the creation of these results is fairly low. In the analysis stage, three rectangles have to be drawn around each vehicle, which typically takes a few seconds. The most time-consuming step is drawing a mask for each rectangle. This task may require a couple of minutes per mask for an unexperienced user. Note that these steps need to be performed only once per segmented vehicle.

Due to memory restrictions in MATLAB we were unable to simultaneously load more than five vehicles into memory. As a quick workaround for this problem, we resorted to using the same vehicles more than once in our synthesis progress. Another option, in a commercial context, would be to introduce a database where the vehicles could be stored and queried in their compact representation.

Our synthesized videos suffer from such artefacts as interlacing and motion blur. These artefacts are already present in the input video (as can be seen in Figure 9), so it is only logical that they also occur in our output videos. As our algorithm was developed for detection systems, this actually becomes a major advantage. Synthesized videos that look too polished provide an unrealistic training test for these systems that does not correspond with the real world conditions.

In the accompanying videoclips (available on the CD-ROM), we can clearly see that thanks to our occlusion map, all occlusions are handled correctly. The videos show vehicles departing from underneath a bridge, where they are correctly hidden from view thanks to the occlusion map. In frames where the vehicles are only partially visible, we apply extrapolation of the PCA components to obtain an approximation of the entire vehicle. The shape and appearance of remote vehicles are accurately estimated using our autoregression algorithm as their orientation barely changes. Closer to the camera however, the orientation of the vehicles can vary rapidly w.r.t. the camera. Autoregression has a hard time estimating changes before they occur because its prediction is based on previous frames. This may cause a slightly thicker edge around the vehicle. This problem however only occurs on a few synthesized vehicles and is visible in just a small area in a few frames, rendering its impact on traffic detection systems negligible.

Figure 10: An edge that might appear around a vehicle when its orientation suddenly changes.

The trajectories of the vehicles in the synthesized videos are restricted to their original trajectories in the input video. This is because the camera has a different perspective view on each lane. If a vehicle is synthesized on a different lane than the one it originally occupied, it will be perceived to be sliding on the road surface instead of driving down the road. However, slight deviations are allowed.

The effectiveness of our shadow maps in synthesizing the illumination effects around the vehicles is illustrated in Figure 11.

Figure 11: Left: synthesized vehicle using a shadow map. Right: synthesized vehicle without using a shadow map.

4 CONCLUSION AND FUTURE WORK

In this paper, we presented a novel video based method for rendering and animating rigid objects in fixed viewpoint video scenes. Our method was examplified by using traffic video sequences. A parameterized sprite appearance model is central in our approach. It describes how the sprite evolves in shape and pixel intensity, and also allows for interpolation, extrapolation and compact storage. Using this information, we can synthesize new videos that feature these sprites, in such a way that the videos exhibit animated traffic situations.

We would like to explore approximate geometric representations to more rigorously represent the relative rotation of the vehicles. Furthermore we believe that variable weather conditions and intricate illumination effects like vehicle headlights can be a valuable addition to our framework.

New trajectories for vehicles could be synthesized if control over the input of the videos is available. A vehicle filmed driving on several different lanes may then be interpolated horizontally afterwards.

Further research is also needed for solving the thick edges that sometimes appear around vehicles near the end of their trajectories. Different prediction and estimation techniques may need to be investigated for this.

ACKNOWLEDGEMENTS

The authors acknowledge financial support on a structural basis from the ERDF (European Regional Development Fund), the Flemish Government and the Flemish Interdisciplinary institute for BroadBand Technology (IBBT), and from research grants by the EU (IST-2-511316-IP "Racine-IP") and the IWT (innovation feasibility study 040658 with Traficon NV). We would like to thank Traficon for making available their traffic video sequences.

Furthermore we would like to thank our colleagues for their help and inspiration, especially Mark Gerrits, Tom Haber and Erik Hubo.

REFERENCES

[AZP+05] A. Agarwala, K.C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic video textures. *ACM Trans. Graph.*, 24(3):821–827, 2005.

[BCS97] C. Bregler, M. Covell, and M. Slaney. Video rewrite: driving visual speech with audio. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 353–360, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[BSHK04] K.S. Bhat, S.M. Seitz, J.K. Hodgins, and P.K. Khosla. Flow-based video synthesis and editing. *ACM Trans. Graph.*, 23(3):360–363, 2004.

[BV99] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH'99 Conference Proceedings*, 1999.

[CBMM98] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research: Part C*, 6(4):271–288, 1998.

[CCSS01] Y. Chuang, B. Curless, D.H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *Proceedings of IEEE CVPR 2001*, volume 2, pages 264–271. IEEE Computer Society, December 2001.

[CGPP00] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Statistic and knowledge-based moving object detection in traffic scenes. *Proc. of ITSC-2000 - The 3rd Annual IEEE Conference on Intelligent Transportation Systems*, pages 27–32, Oct. 1-5, 2000.

[CV05] A.B. Chan and N. Vasconcelos. Classification and retrieval of traffic video using auto-regressive stochastic processes. *IEEE Intelligent Vehicles Symposium*, 2005.

[CV06] A. Chan and N. Vasconcelos. Layered dynamic textures. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, 2006.

[DBB03] P. Dutré, P. Bekaert, and K. Bala. *Advanced Global Illumination*. AK Peters, 2003.

[DTM96] P. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH Proceedings*, 1996.

[Fit01] A. Fitzgibbon. Stochastic rigidity: Image registration for Nowhere-Static scenes. *Proceedings of International Conference on Computer Vision*, pages 662–669, 2001.

[GASK95] B. Gloyer, H. Aghajan, K.-Y. Siu, and T. Kailath. Video-based freeway monitoring system using recursive vehicle tracking. In *Proceedings of SPIE*, February 1995.

[JF01] N. Jojic and B. Frey. Learning flexible sprites in video layers. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[KCP92] J.R. Kent, W.E. Carlson, and R.E. Parent. Shape transformation for polyhedral objects. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 47–54, New York, NY, USA, 1992. ACM Press.

[KM03] Z. Kim and J. Malik. Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking. In *In Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2003.

[Mat03] W. Matusik. *A Data-Driven Reflectance Model*. Massachusetts Institute of Technology, 2003.

[NS04] A. Neumaier and T. Schneider. Arfit, a matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Trans. Math. Softw.*, 27:27–57, 2004.

[SE00] A. Schödl and I. Essa. Machine learning for video-based rendering. In *Proceedings of NIPS*, 2000.

[SE02] A. Schödl and I. Essa. Controlled animation of video sprites. *Proceedings First ACM Symposium on Computer Animation*, July 2002.

[SG99] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[SJTS04] J. Sun, J.J., C. Tang, and H. Shum. Poisson matting. *ACM Trans. Graph.*, 23(3):315–321, 2004.

[SSSE00] A. Schödl, R. Szeliski, D.H. Salesin, and I. Essa. Video textures. *Proceedings of SIGGRAPH*, pages 489–498, 2000.

[Tra] Traficon, website: `http://www.traficon.com/`.

[WA94] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, 3(5):625–638, September 1994.

[ZCSP03] C. Zhang, S-C. Chen, M-L. Shyu, and S. Peeta. Adaptive background learning for vehicle detection and spatio-temporal tracking. *IEEE Pacific-Rim Conf. on Multimedia (PCM'03)*, 2003.