Hasselt University
transnationale universiteit Limburg

Video-Based Animation : Articulated Video Sprites and Augmented Panoramic Video

Dissertation submitted for the degree of

Doctor of Philosophy in Computer Science

at Hasselt University

to be defended by

Cedric Vanaken

on May 18, 2011

Advisor: Prof. dr. Frank Van Reeth

2005 – 2011

Abstract

With digital photo and video cameras currently taking an indispensable part of many domestic households, a vast amount of research and software development has been aimed towards tools for facilitating the manipulation of images and videos. Ranging from foreground segmentation to deblurring, many powerful tools have been made available.

In this dissertation we mainly focus on the use of video sprites in so called image- and video-based animation, synthesis and augmentation techniques. These techniques use one or more images as input, analyse this data and animate, synthesize or augment it to obtain new images inspired by the original. The best known video-based synthesis example is the *Video Textures* algorithm from Schödl et al. [Schödl 00b]. Starting from a relatively short video sequence, a new video is created by rearranging the input frames in such a way that the resulting video can be infinitely looped while adhering to the appearance of the original content and avoiding visually harsh transitions.

An important extension to this work was provided in 2002 with the *Video Sprites* technique of Schödl et al. [Schödl 02]. Instead of playing back whole frames, images of a character are extracted from video and concatenated to form new animations. Where video textures are mostly limited to work with videos that inherently feature repetitive visual content like natural phenomena, the video sprites algorithm allows the animation of small characters like hamsters or flies. With both techniques allowing only a limited amount of possible input videos, we introduce two video based animation and synthesis techniques that are aimed at videos containing different kinds of subjects.

We first present a novel technique to synthesize videos featuring traffic scenes. Given an input traffic video, we are able to reproduce a traffic scene from the same viewpoint in which the configuration of the vehicles has been altered by the user. The main application of our technique is the validation and training of camera-based traffic analysis systems, e.g. for accident, queue and presence detection.

Secondly, we present a new approach to video sprites that allows for animating articulated characters such as animals and humans. Articulated characters possess an underlying structure in the form of skeletons. We exploit this by using 2D skeletal representations instead of the visual appearance of the subject. Having a desired animation defined as a sequence of skeletons, we look for the best matching input frames and arrange these in such a way that the subject performs the desired movement.

When solely using input frames to create new animations, the quality and variety of the results is limited by the diversity of the input data. To extend the amount of available input data, we introduce a technique that facilitates the creation of novel human poses by synthesizing images. Existing approaches commonly deform one single image, often resulting in a distorted image due to texture and illumination artifacts. We present a novel image-based pose synthesis technique that accurately reconstructs texture details by combining information from multiple photographs. Given a user-specified 2D target pose, our solution merges different parts of the input photographs in order to conform to the desired pose, solely using 2D operations. We illustrate how novel poses can be generated from only a few example images, requiring little user intervention. Combining this technique with our articulated characters aimed approach to video sprites allows for a larger variety of possible target poses to animate the filmed subject.

In the final part of this dissertation we take a side-step from these animation algorithms and discuss a video editing and mosaicing system called *Augmented Panoramic Video*. We focus on a common type of video sequence, in which the videographer shoots a scene by rotating the camera to capture the entire panorama, possibly zooming into areas of particular interest that can contain dynamic subjects such as people or animals. We wish to re-display and manipulate such sequences in a meaningful way, presenting a technique that gives the user control over the camera's motion and field of view. The presented results show this technique produces high quality panoramas without parallax artifacts, seams or blurring, while retaining repetitive dynamic elements.

Acknowledgments

In my experience, the first skim through a PhD-thesis is mostly restricted to looking at the images in the Results sections and reading the Acknowledgements, not necessarily in that order. Please go ahead and check out all the full-color images, I hope you won't be disappointed. If you already did, then thank you for staying. Chances are that you're a colleague or a relative of mine, so let me start off by thanking you for your help and/or support. Thanks!

Traditionally, the supervisor gets acknowledged first in this section. In my case, that would be Prof. dr. Frank "ziet maar dat ge mij in uw dankwoord zet!" Van Reeth. Frank fully deserves to be acknowledged. In my second year of Computer Science studying, it was Frank who triggered my interest with his Computer Graphics classes. I could not have wished for a better supervisor. When necessary, Frank always provides me with good ideas and insights. Whenever I come up with a far fetched solution, Frank takes the time to think it through with me, mostly ending up with a straight forward approach, for which I'm very thankful.

Second on the list of acknowledgements is Prof. dr. Philippe Bekaert. I would have never ended up in this research area without the master thesis proposals that Philippe provided. I am very happy that I was allowed to work on my "Free Viewpoint Video" master thesis (together with Karel Frederix, I'll thank you later, don't worry) in the graphics lab of Frank and Philippe in the Expertise Centre for Digital Media. Philippe's passion for computer vision research has been very inspiring to me since I followed his Image Processing classes and keeps on inspiring our entire research group. Frank and Philippe, thanks for giving me the chance to work in the graphics group, most definetely the coolest research group of the EDM!

My PhD committee, Prof. dr. Karin Coninx and Prof. dr. Philippe Bekaert, together with external reviewer John Collomosse: thank you for taking effort in reading this dissertation and for your much appreciated comments. I would also like to thank the chair of the jury, Prof. dr. Marc Gyssens, and the other members of the jury, Prof. dr. Rae Earnshaw and Prof. dr. ir. Christophe De Vleeschouwer for evaluating my work.

Prof. dr. Eddy Flerackers, also known as managing director of the EDM, thank you for your annual speeches. Even more, thank you for making sure that I was not lacking any resources during my stay at the EDM. I would also like to thank the rest of the academic staff of the EDM: Prof. dr. Wim Lamotte, Prof. dr. Kris Luyten, Prof. dr. ir. Luc Claesen and Prof. dr. Fabian Di Fiore along with Peter Vandoren for providing an inspiring and productive working environment. This inspiring and productive working environment would never be the same without Ingrid Konings. For me, Ingrid is indispensable at the EDM. With Ingrid and Roger Claes in charge, the EDM has a superb administration team.

Thanks dr. Peter Quax and Tom De Weyer for keeping the EDM network online 24/7.

Next in line are my dear colleagues. While studying Computer Science, lots of lunch breaks were passed at Demerstrand Skatepark together with Karel Frederix. Many years later, I'm glad to see that we're still sharing lunch breaks.

When I started working at the EDM, I had the honour of sharing an office with dr. Tom Mertens, Tom Haber and Mark Gerrits. Tom, Tom and Mark have been a huge inspiration and support for me, inside and outside the office.

I would also like to thank the current and former members of the graphics group. We've had lots of interesting discussions and great game/movie nights! Thank you Codruta Ancuti, dr. Cosmin Ancuti, dr. Koen Beets, Tom Cuypers, dr. Bert De Decker, Maarten Dumont, Prof. dr. Fabian Di Fiore, dr. Yannick Francken, dr. Jan Fransens, Karel Frederix, Mark Gerrits, Patrik Goorts, Tom Haber, dr. Chris Hermans, dr. Erik Hubo, Steven Maesen, dr. Tom Mertens, Johan Nulens, Sammy Rogmans, Johannes Taelman, dr. William Van Haevre and dr. Tom Van Laerhoven, great job! I especially thank Tom Haber for proofreading this dissertation and Chris "Wolf" Hermans for the great collaboration.

Thanks mom & dad for having an endless amount of faith in me and for always being around when needed. I am pretty sure that I have the best parents in the world. Thanks dad for the Commodore 64 and everything that followed after that. Thanks mom for your never ending flow of love, worries and food. Thanks Anouk for calming down mom and dad whenever the kid inside me takes control. I am pretty sure that I have the best sister in the world.

Thanks friends, for hanging on while work was very/too important to me. A lot of time and energy was put into this PhD. Some friends got lost out of sight during the years, but without this job at the EDM I wouldn't have met Open Garments project partner Katrien Dreessen. Thanks Katrien, I am pretty sure that I have the best girlfriend in the world.

Thanks again Tom Mertens, Tom Haber and Mark Gerrits for fueling my interest for digital photography. Thanks Kurt Bosmans, I wouldn't have become a photographer without your support. Also thanks to Hans Machiels, Jeroen Delodder, Kristof Claes, Kim Mathijs, Edwin Korver, Nina De Man, Luc Carpentier and Ralf Leesen for believing in "c vangt licht" and Sure Shot.

Pol, Niem, Lola, Jack Bauer, life would not have been the same without the best cats in the world. And fortunately, I also have the best neighbours in the world, thanks Mary, Jan and your lovely kids.

Andy Decroos, Jochen Vandorpe, Michelle Palumbo, Veerle Verbakel, Bart Bulen, Line Vanvoorden, Bart Neven, thanks for doing what you do.

Off to the next challenge!
Diepenbeek, May 2011.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Problem Statement

Animating and rendering images and videos in a convincing and realistic way has always been an important goal in computer graphics research. One of the best known photorealistic rendering techniques is called *ray tracing* [Dutré 03]. Ray tracing algorithms can render scenes with a high degree of photorealism, this due to the fact that they translate nature's illumination rules into mathematical formulas. With ray tracing, simulated rays of light are traversed throughout a scene, starting from a virtual camera that can be customized by the user. The mathematical calculations take account of reflection and refraction physics as well as shadows. Unfortunately, these kind of calculations can be complex and mostly yield large computational costs. With recent improvements in computer hardware and ray tracing research, ray tracing has finally become interactive. However, most of these techniques only support walk-through applications for static scenes.

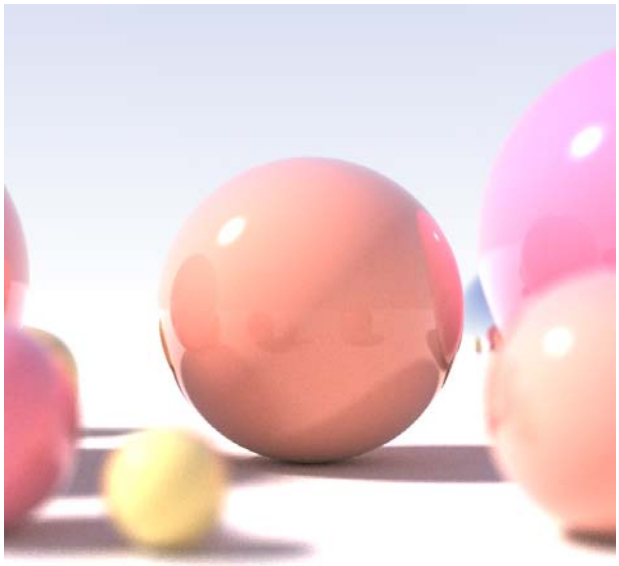


Figure 1.1: Ray traced scene featuring spheres of different sizes.

An image produced by a ray tracer is shown in Figure 1.1. This image looks highly realistic, which is not only the merit of the technique, but also of the provided scene input. Scene input can be represented in different ways. One can use point clouds, geometry descriptions, surfaces, etc, which can be accompanied by material properties of the objects as well as lighting information of the scene. Coming up with an accurate description for objects is not always straight-forward, the quality of the results is therefore often highly dependent of the designers or 'artists'. The shown figure also has an artificial feeling, which occurs often with ray traced images.

When considering dynamic scenes, physically correct rendered natural phenomena can fairly diminish the artificial feel and boost the perceived level of realism. Examples of this are the wind blowing through trees, a flowing river, smoke, etc, which unfortunately are all very hard to model visually correct.

Next to photorealistic renderings, ray tracing techniques have also been used in the production of quite a lot of Hollywood animation productions. Figure 1.2 shows a ray traced image from the *Pixar* animation movie *Cars* [Christensen 06]. However, when the goal is to render a scene in which a real-life actor performs a (physically

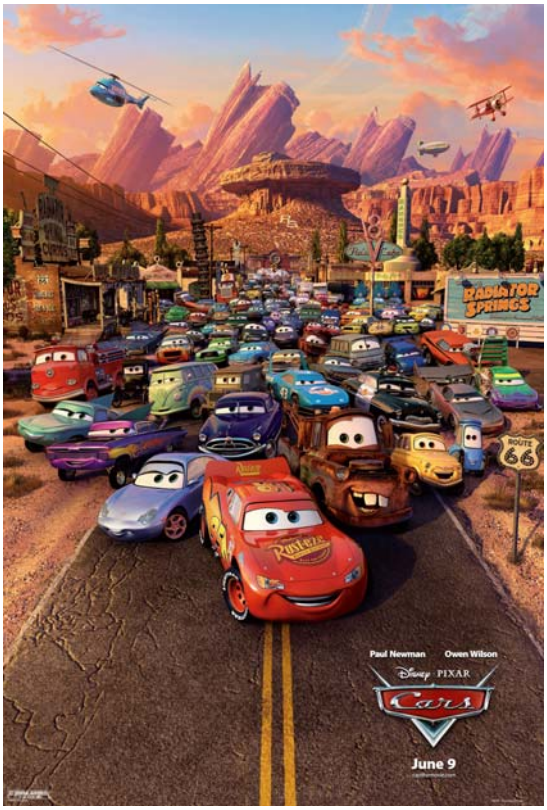


Figure 1.2: Ray traced poster for the *PIXAR* movie *Cars*.

difficult) motion in a realistic looking way, computer graphics research is not necessarily the right domain to look into.

Recently, computer vision researchers have come up with many interesting image- and video-based rendering techniques [Schödl 00a, Schödl 02, Agarwala 05, Liu 05]. Compared to ray tracing algorithms, these vision techniques are fast and computationally inexpensive, with computations being fully independent of the complexity of the scene. These techniques display the scene as recorded, inherently eliminating the artificial feel, however not providing any degree of freedom on changing camera and lighting parameters in post-production. In addition, since little or no geometric information of the captured scene is known, adjusting the scene content in post-production can be quite cumbersome.

Figure 1.3: Camera setup used in the *Matrix Trilogy*, used to create the "bullet time" rotating camera effects. (http://whatisthematrix.warnerbros.com/)

The *Matrix Trilogy* is an example of a big Hollywood production using computer vision algorithms. Bullet-time effects have been obtained by using a large amount of expensive high quality video cameras, positioned close to each other in a circle. This camera setup is illustrated in Figure 1.3.

Over the last decade, digital photo and video cameras have appeared in many domestic households. With captured images and videos being inherently photorealistic, a very accessible alternative to standard modeling and rendering techniques has arised. Many researchers and software developers lately have been creating tools for facilitating the manipulation and editing of photographs [Oh 01, Chuang 05, Lalonde 07, Avidan 07], with Adobe Photoshop being one of the best known. These tools range from seamless cloning [Pérez 03a] to automatic image-based rendering [Hoiem 05], with the work of Joshi et al. [Joshi 10] illustrated in Figure 1.4. Joshi et al describe a framework for improving the quality of personal photos by using a person's favorite photographs as examples.

By adding an extra dimension to the image data representation, video allows for the use of temporal as well as visual information in rendering algorithms. This invites researchers to extend standard image processing techniques to be applicable to video, not always straightforward, but most likely a very rewarding challenge.

On consumer level, multi-camera setups can be quite expensive and are furthermore not always straightforward to assemble, calibrate or synchronize. Therefore

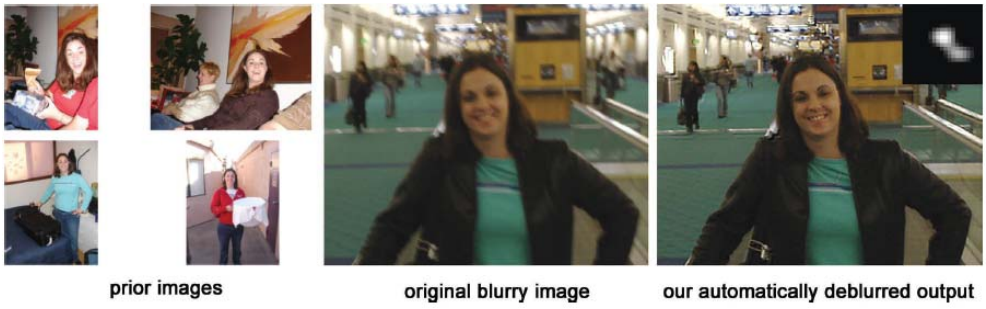


Figure 1.4: Illustration of the **Personal photo enhancement using example images** paper of Joshi et al [Joshi 10]. Personal photos are improved by using a person's favorite photographs as examples. Face detection is used to align faces between "good" and "bad" photos. Deblurring, sharpening, in-painting and white-balancing operations are used to further improve the resulting pictures.

the focus of this dissertation will be shed on single-camera based applications.

Video-based techniques use a sequence of images as input, analyse the data and animate, synthesize or augment it to obtain a new video that is inspired by the original. The best known video-based synthesis example is the *Video Textures* algorithm of Schödl et al. [Schödl 00b]. Starting from a relatively short video sequence, a new video is created by rearranging the input frames in such a way that the resulting video can be infinitely looped, while adhering to the appearance of the original content and avoiding visually harsh transitions. An important extension to this work was provided in 2002 with the *Video Sprites* technique of Schödl et al. [Schödl 02]. Instead of playing back whole frames, images of a character ("sprites") are extracted from video and concatenated to form new animations. Where video textures are mostly limited to work with videos inherently featuring repetitive visual content like natural phenomena, the video sprites algorithm allows the animation of small characters like hamsters or flies. Since both techniques allow only a limited amount of possible input videos, we introduce two video based animation and synthesis techniques that are aimed at videos containing different kinds of subjects.

Furthermore, while average camera users not always have the same experience and skills as trained professionals, an interesting field of research and development lies in augmenting captured images and videos. Useful applications include video

Figure 1.5: A frame generated by the **panoramic video textures** algorithm of Agarwala et al. [Agarwala 05].

stabilization [Matsushita 05] - where unintended shaky motion is removed from a video - and restructuring images and video, where images can be resized in a content-aware manner [Avidan 07] or seamless panoramas can be built [Brown 07b]. These tools provide consumers a large array of possibilities to easily and intuitively augment home-made recordings. We take this one step further by introducing a technique aimed at creating panoramic videos featuring foreground and background sprites.

Related to our work is the *Panoramic Video Textures* algorithm from Agarwala et al. [Agarwala 05] (see Figure 1.5). Starting from a video segment filmed by panning a camera across a dynamic scene, they combine looping segments of a constant duration in order to construct a single panoramic video texture. While our work is aimed to work with arbitrary input videos, the method of Agarwala et al. is restricted to horizontal panning sequences. Our method furthermore allows dynamic foreground elements that feature more than just looping elements.

1.2 Contributions

We will now briefly describe the main contributions of this dissertation, followed by an organized overview of the dissertation in the next section.

- We present a novel technique to synthesize videos featuring traffic scenes. Given an input traffic video, sprites are created for the vehicles in a semi-automatic manner. Using these sprites, we are able to reproduce a traffic scene from the same viewpoint in which the configuration of the vehicles has been altered by the user.

- We introduce a new approach to the Video Sprites [Schödl 02] technique that allows for animating articulated characters such as animals and humans. Articulated characters possess an underlying structure in the form of skeletons. We exploit this by using 2D skeletal representations in our algorithm instead of the visual appearance of the subject.

- We present a technique that facilitates the creation of novel human poses by synthesizing images. We accurately reconstruct texture details by combining information from multiple input photographs. Given multiple images featuring a character in different poses, a new image can be created with this character standing in a new pose by combining information from the input images. Combining this technique with our articulated characters aimed approach to video sprites allows for a severely larger variety of possible target poses to animate the filmed subject.

- We propose a novel way of re-displaying video sequences featuring dynamic backgrounds and moving foreground subjects, by giving the user control over a virtual camera frame using a full panoramic representation. The virtual camera can have an enlarged field-of-view and a controlled camera motion. This technique is able to process videos with complex camera motions, reconstructing high quality panoramas without parallax artifacts, visible seams or blurring, while retaining repetitive dynamic elements.

1.3 Overview of the Dissertation

The text is organized as follows:

- Chapter 2 discusses the first contribution of this dissertation. Our first attempt to video-based synthesis is aimed at input videos featuring traffic scenes. Vehicle “sprites” are extracted from the input footage and resynthesized in a controllable fashion. The main application of this technique is the validation and

training of camera-based traffic analysis systems, e.g. for accident, congestion and presence detection.

- In Chapter 3 we present two novel techniques that can be used in the context of video-based character animation.
  The first part of Chapter 3 introduces our articulated video sprites algorithm, this is a new approach to video sprites that allows animation of articulated characters, such as animals or humans. The key to our technique is a matching algorithm that focuses on high-level 2D skeleton models instead of the character's visual appearance. This matching algorithm does not evaluate absolute positions of skeleton joints, but implicitly includes 3D information by comparing angles and ratios between adjacent limbs.

  The second part of Chapter 3 presents our image-based pose synthesis technique, which facilitates the creation of novel human poses by synthesizing images. Multiple input images featuring a character in different poses are combined into a new image that shows the character in a brand new pose.

- Chapter 4 discloses the final contribution of this dissertation, namely our work related to Augmented Panoramic Video. Starting from a panoramic video, captured by rotating the camera on a static position, we reconstruct high quality panoramas without parallax artifacts, visible seams or blurring, while retaining repetitive dynamic elements and allowing the user to control the motion and field of view of the virtual camera.

- Chapter 5 concludes this dissertation with some final thoughts and an outlook on future work.

- In Appendix A, a list of contributions and publications related to this dissertation is given.

- Appendix B contains a Dutch summary of this thesis.

Chapter 2

Video-Based Synthesis for Rigid Objects

Contents

2.1 Introduction

Video-based rendering methodologies have proven to be adept at synthesizing photo-realistic video sequences from sparse real world data, with the best known example being the Video Textures of Schödl et al. [Schödl 00b]. Video Textures

are derived from video by changing the order in which the recorded frames are played, while assuring seamless transitions between the video frames. By playing frames out of the original order, only where it is unnoticeable for the viewer, a finite duration input clip can be transformed into a smoothly playing infinite video. This concept has been extended to Video Sprites [Schödl 02], where images of a filmed character are extracted from video and concatenated to form new animations.

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, congestion and presence detection [Traficon]. Given the high percentage of daily commuters on the road, fast and accurate detection of accidents can be critical to avoid long traffic james on the highways. These camera based traffic systems cannot afford to have a high margin of error and thus require a wide variety of initial test sequences. Because these input video 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-consuming task.

As an alternative, one might synthesize video sequences directly using traditional modeling and global illumination techniques [Dutré 03]. 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. Using rendered scenes, one needs to be able to reproduce natural landscapes in the surroundings, including all kinds of different weather behaviours. Keeping weather in mind, hard to render phenomena like rain-drops falling on cars and splashing on the road are required to create test scenes with a high degree of realism. Furthermore, camera imperfections should also be simulated.
By using recorded footage from the targeted camera system, we achieve our goals immediately.

Akin to Debevec et al.'s image based modeling approach [Debevec 96], one might construct approximate geometry for each vehicle of interest. However, this is a labor-intensive task, since this process has to be carried out for every existing car separately. Alternatively one might come up with a parameterized model for vehicle geometry which can be fitted to the image data [Blanz 99]. 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 [Schödl 00b, Schödl 02]. 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 fairly straight path, while Video Sprites [Schödl 02] 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 its compression rate makes it useful for data storage. In our case, only 20 % of the actual input vehicle data is stored.

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. [Schödl 00b, Schödl 02] 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 [Jojic 01] copes with inter-sprite occlusions efficiently. For our purposes this is less of an issue, but static obstructions like lamp posts will need to be dealt with.

The outline of our system is presented in Figure 2.1 on page 12. Starting from an input video and some minor user assistance (semi-automatic vehicle segmentation, see Section 2.3.1.3), the analysis phase outputs a background, an "occlusion map" and extracted vehicles with associated trajectories. This data is used as input for the

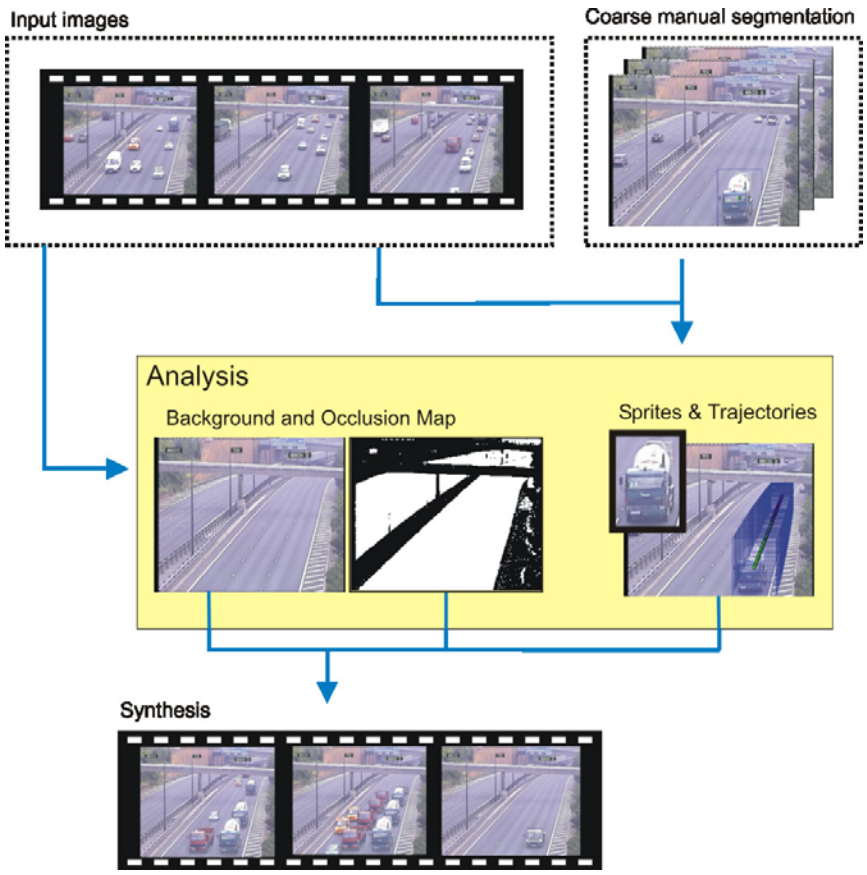


Figure 2.1: **Outline of our traffic animation system**. Starting from an input video and some user assisted vehicle extraction, the analysis phase outputs a background, an "occlusion map" and extracted vehicles with associated trajectories. This data is used as input for the synthesis phase, where the vehicles are drawn onto the background to obtain a new animated video.

synthesis phase, where the vehicles are drawn onto the background to obtain a new animated video.

The remainder of this chapter is organized as follows. We start by reviewing related work in the areas of traffic detection techniques and video based animation systems. Section 2.3 gives an outline of our system. Section 2.4 presents our results

and discusses practical issues. Finally, in Section 2.5 we will present our conclusions and suggestions for future work.

2.2 Related Work

This work touches on two major research areas: Vision Based Traffic Surveillance and Video Based Computer Animation.

2.2.1 Vision Based Traffic Surveillance

Considering the Traffic Surveillance part of the research, our work serves a somewhat different purpose than what has been mainly achieved so far. The most common approaches used for vision based traffic surveillance consist of a fast segmentation of the vehicles in the scene, together with an intelligent reasoning module capable of identifying, tracking and classifying the vehicles in dependency of the system goal [Cucchiara 00]. Real-time extraction of moving objects is an essential part of traffic surveillance [Zhang 03, Cucchiara 00, Coifman 98], and different surveillance systems obviously use different segmentation techniques.

The biggest difference between our work and common traffic surveillance systems is segmentation. In our case, precision comes before speed. A badly segmented vehicle will look equally bad when it is used in synthesis, while a badly segmented vehicle will still be counted or tracked as a vehicle in a surveillance system. Since our input videos don't exceed a time length of 5 or 10 minutes and global background features usually don't change abrupt in this timespan, we have opted for a static background instead of a dynamic one [Zhang 03, Cucchiara 00].

Zhang et al [Zhang 03] and Cucchiara et al [Cucchiara 00] both use dynamic backgrounds, with the former using an adaptive learning method and the latter employing statistic and knowledge-based backgrounds. A big issue related to the segmentation of vehicles is how shadows and other illumination effects are dealt with. Different detection systems, and more specifically the way that these systems handle shadows, are discussed by Prati et al in a comparative study [Prati 01]. As shown in Figure 2.2, it is of high importance for these surveillance systems not to misclassify shadows as moving objects or as parts of moving objects.

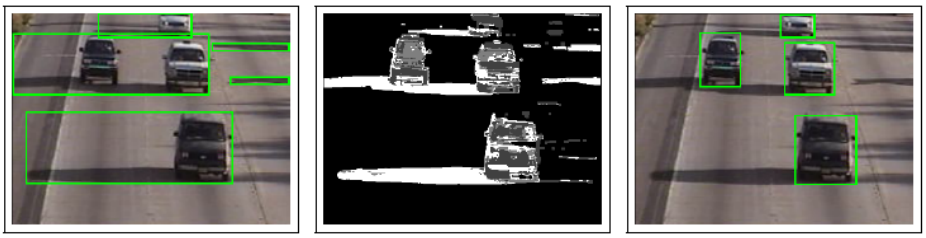


Figure 2.2: **Importance of shadow handling in traffic surveillance.** [Prati 01] Left : Detection without shadow suppression. Center : Shadow detection. Right : Detection with shadow suppression.

Our work also considers shadows as being part of the background, but shadows casted by vehicles are an important factor concerning the realism of the synthesis, so adversly, we extract these shadows from the input video instead of ignoring them.

The second major difference exposes itself in how the data is utilized after the extraction. A lot of different purposes can be served when it comes to traffic surveillance [Burns 04, Oh 03, ITS 11]. Our goal is to use the extracted vehicle-related data to compose new video sequences, which form a very effective means for the testing of these traffic surveillance systems.

To fulfill this ambition, we need to satisfy some requirements, from which the most important comprises of the realism of our synthesized videos. Therefore we have opted for a video based system using a very exact segmentation method. In our case, precision comes before speed.

2.2.2 Video Based Computer Animation

Video-based representation, rendering and animation techniques have received increasing interest in the graphics and vision community in the past years. The biggest benefit of these video-based techniques, is that when starting from an input video, one already obtains of a high degree of photo-realism to start with. Compared to modeling a complex scene, video-based techniques simply use a video of the required scene.

Figure 2.3: Left : **Video Textures** [Schödl 00b] example, with only a short recording of a flame, a seamless never-ending video is produced. Right : **Video Sprites** [Schödl 02] example, creating a video sprite of one fly, a synchronized fly-collective can be animated.

A recurring video-based technique consists of rearranging frames in an input video to create novel footage, either globally on a per-frame basis [Bregler 97, Schödl 00b, Agarwala 05], or locally on a per-sprite basis [Schödl 00a, Schödl 02]. Video sprites are obtained by filming an object and extracting it from the video, which can be done automatically using layers [Jojic 01] or chroma keying [Schödl 02], or manually [de Juan 04]. Schödl and Essa animate these sprites by rearranging the frames of the original video [Schödl 02], a technique successfully used by Video Textures [Schödl 00b]. This approach allows for efficient animation of e.g. natural phenomena or small animals, with examples shown in Figure 2.3.

Since we are restricted to using only the appearances of vehicles that are shown in our input video, we don't have the ability to let a vehicle take every possible turn and every available lane in a new animation. We can merely synthesize a vehicle in the same way it was extracted in the original footage, except for simulating speed changes. Nevertheless, this also means we are exempted from segmenting and analyzing an excessive amount of filmed data and calculating extensive graphs and associated cost functions of animations for this data.

Our approach furthermore differs by not reordering frames but rather building a simple parametric motion and appearance model for vehicle sprites, similar in spirit to Fitzgibbon [Fitzgibbon 01]. Consequently, less input is required and this representation even facilitates extrapolation. A different solution for sprite

Figure 2.4: **Active Appearance Models** [Cootes 01] are shape models generated from a training set of images. In this example 400 images were manually labelled with 122 landmark points around features. From this training data, an appearance model was generated with only 80 parameters. By changing these parameters, variations to the appearance can be produced.

representations can be found in Active Appearance Models [Cootes 01] using a statistical model of the shape and appaearance of an object of interest. Active Appearance models require an intensive training phase, where training images need to be manually annotated with hundreds of landmark points, as illustrated in Figure 2.4. This technique works very well related to medical imaging and face representation/recognition/detection, where the input subjects position and orientation towards the camera is relatively static. In our case, the vehicles more from far away until nearby the camera, forcing us to take perspective changes into account in our model.

Layered video models [Wang 94, Jojic 01] automatically extract layered sprites and moving parts from input footage by using a variational expectation maximization algorithm to learn a mixture of sprites from a video sequence. For this, the number of layers and sprites in the video have to be indicated manually by the user. 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 Motion Magnification algorithm from Liu et al [Liu 05] composites motion sprites as layers onto an average background texture. Subtle motions in a video sequence can be amplified to allow for visualization of deformations that would

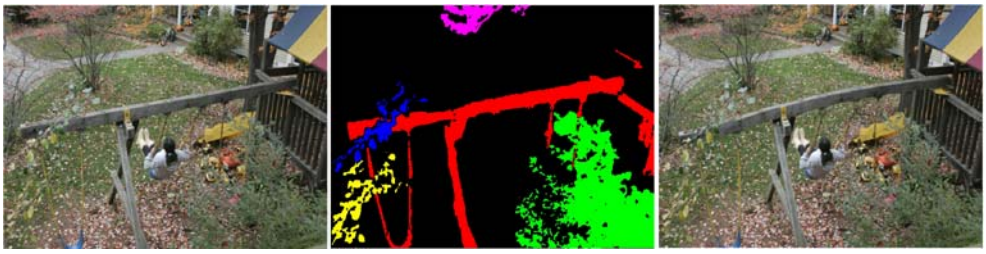


Figure 2.5: **Motion Magnification** [Liu 05], acts like a microscope for visual motion. After identifying different motions in a video sequence, users are allowed to choose a motion cluster and magnify it without objectionable artifacts.

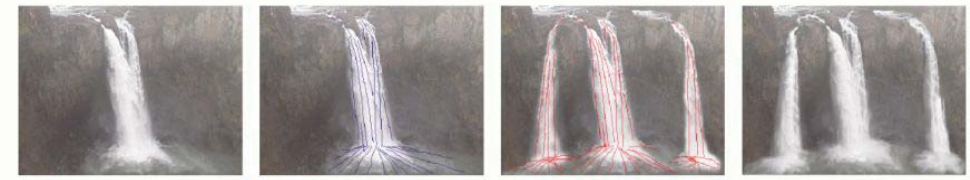


Figure 2.6: **Flow-based video synthesis** [Bhat 04], textured particles follow user-specified lines to synthesize new animations.

otherwise be invisible. The work in the Motion Magnification paper addresses and clusters all kinds of motions, ranging from small motions like a baby's breathing to large motions like someone playing on a swingset (see Figure 2.5). The sprites used in our work more or less follow the same trajectory and can be grouped as rigid sprites, allowing us to make useful assumptions and discarding the need for an extensive feature registration technique as introduced by Liu et al.

The flow-based video synthesis technique [Bhat 04], illustrated in Figure 2.6, 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 [Chan 05, Chan 06] model traffic flow from video using a holistic generative model. This is done by adopting an auto-regressive stochastic process, which encodes the appearance and the underlying motion of the video separately into two probability distributions. The 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 [Zhang 03, Cucchiara 00, Coifman 98, Kim 03], 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.

2.3 Overview

As illustrated in Figure 2.1 on page 12, the system consists of an analysis stage and a synthesis stage, which will both be detailed in the following sections.

2.3.1 The Analysis Stage

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

2.3.1.1 Background

We obtain an appropriate background as the per-pixel median intensity along the time dimension [Gloyer 95]. This simple technique works very well with our input video sequences, which have virtually static backgrounds. This rendered the implementation and testing of other, more complicated, techniques (e.g. [Stauffer 99]) unnecessary in our case. An example of a calculated background image is shown on the left side of Figure 2.7.

2.3.1.2 Occlusion Map

Similar to the occlusion buffers introduced by Collomosse et al [Collomosse 03], our occlusion map indicates which pixels remain unchanged during the entire length of the input video. The pixels that fluctuate significantly with respect to some threshold T can be considered “possibly foreground”, indicating that the background should always be drawn behind the sprites. A static pixel will either be an occluder

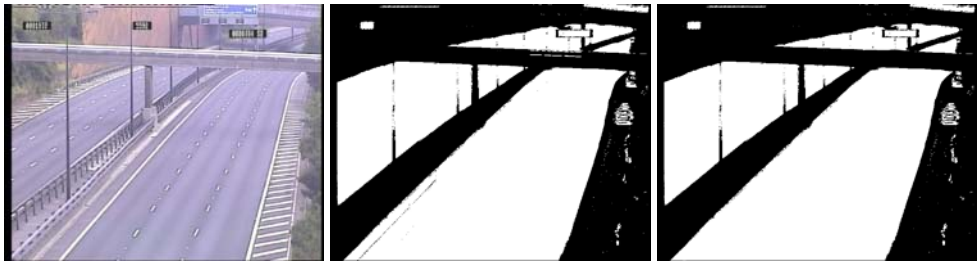


Figure 2.7: **Background image and occlusion map.** Left: extracted background image from an input video. Middle: calculated occlusion map. Static pixels (occluder or background) are labeled black and fluctuating pixels (possibly foreground) are labeled white. Right: occlusion map after applying minor manual corrections.

(for example the bridge in Figures 2.7 and 2.8) or background area without dynamic behaviour. Either way, no sprite layer should be drawn on top of these pixels.

The occlusion mask is calculated by checking for every pixel whether or not its color intensity varies wildly across the different input frames. Starting with the first frame, we compute the sum of squared differences between the colour intensities of each pixel and their counterparts in the already calculated background. If this difference stays beneath threshold T, the pixel probably belongs to the background and will be painted black in the occlusion map, otherwise it will be painted white. The value of threshold T was empirically estimated to an intensity value of 0.012 in our experiments, with pixel values scaled between 0 and 1.

This procedure is repeated for all input frames. Any pixel that was turned white in a previous frame won't be examined in the subsequent frames.
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. An example of such a constructed occlusion map is shown on the right side of Figure 2.7

2.3.1.3 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

Figure 2.8: **Vehicle trajectory calculation.** Windows drawn around a particular vehicle by the user, with on the right a calculated trajectory for the vehicle.

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 (see Figure 2.8).

In the next step, the user draws a mask for the vehicle in the three initially indicated windows, as illustrated in Figure 2.9.

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. [Kent 92], 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

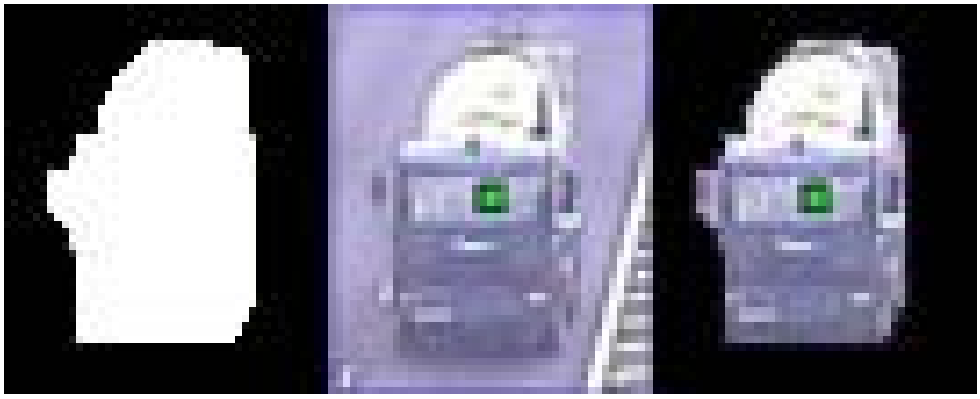


Figure 2.9: **Vehicle segmentation**. Left: user drawn mask. Middle: vehicle for which the mask is drawn. Right: result of the mask operation.

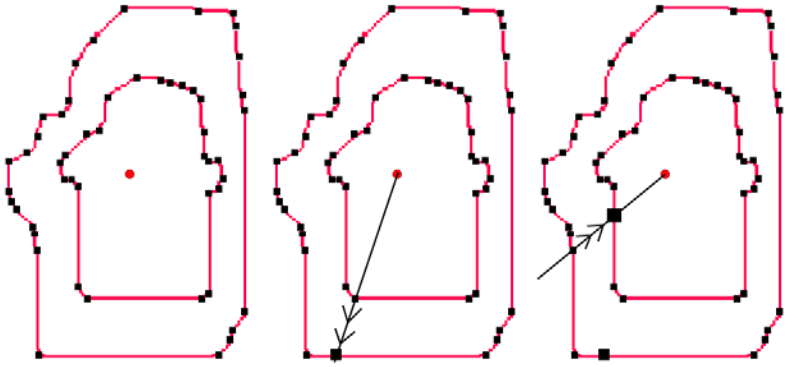


Figure 2.10: **Polygon interpolation.** Left: source polygon placed on target polygon. Middle: extra interpolation step. Right: inverse direction extra interpolation step.

be achieved by using more advanced matting techniques [Chuang 01, Sun 04].

2.3.1.4 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

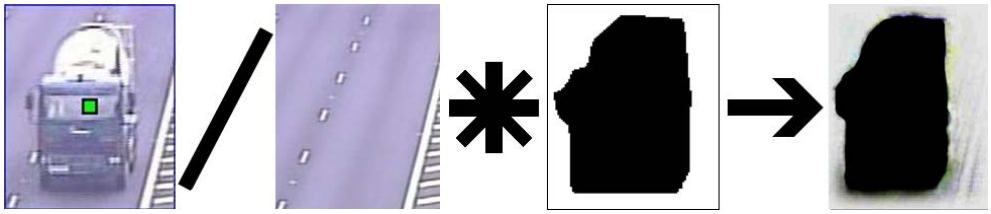


Figure 2.11: **Shadow map computation.** Shadow information is extracted by dividing the window content by the background (right image is contrast enhanced for presentation purposes).

time (t) by fitting the following simple parametric function to each of their corner positions x:

$$x(t) = \frac{(a \times t) + b}{t + c} \tag{2.1}$$

This function takes account of the position of the vehicle in time to model the vehicle's trajectory. When a vehicle drives towards the camera with perspective projection, its position will change faster while approaching the camera. Parameters a, b and c can be solved easily given 3 positions for x at a given time t.

The appearance of the vehicle inside its mask mainly evolves due to a small relative rotation with respect to the camera and environmental illumination (e.g. from street 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 2.11. 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, which is not done for the shadow map.

Given the full appearance of a vehicle at each frame, we reduce it using Principal Component Analysis (PCA). PCA allows us to identify patterns in data and to visualize the data in such a way as to highlight similarities and differences in the high dimensional data space. When the significant area of the high dimensional input data is found, one can compress it by reducing the number of dimensions, without much loss of information.

Before we apply PCA, 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 D. Usually D is quite large (e.g. 10^5) while the number of frames N is small (e.g. 100), yielding a $D \times N$ matrix $Y = [y_1, ..., y_N]$ that features an impractically large covariance matrix $\Sigma = \sum_{i=1}^{N} y_i y_i^T$.

We follow Matusik's variant on the PCA for high-dimensional data algorithm [Matusik 03, Bishop 06] to circumvent this problem. More precisely, we perform an eigenvalue decomposition of the $N \times N$ dimensional covariance matrix of the zero mean Y :

$$B = Y^T Y = V \Lambda V^T \tag{2.2}$$

where V is the orthonormal matrix of eigenvectors and Λ is the diagonal matrix of decreasing eigenvalues. The obtained eigenvectors are sorted by ascending eigenvalues λ, while the vectors with very low eigenvalues are thresholded. Finally, our PCA representation of the appearance of the vehicles consists of:

- transformation matrix: $U = YV(\Lambda^{-\frac{1}{2}})$
- PCA coefficients matrix: $X = U^T Y = \Lambda^{\frac{1}{2}} V^T$
- mean image of the frames: μ

with U an orthonormal basis $[u_1, ..., u_D]$ and $X = [x_1, ..., x_N]$ the coordinates of the input data in the new basis.

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, as illustrated in Figure 2.12.

Usually, the full trajectory of a vehicle cannot be captured 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 slightly

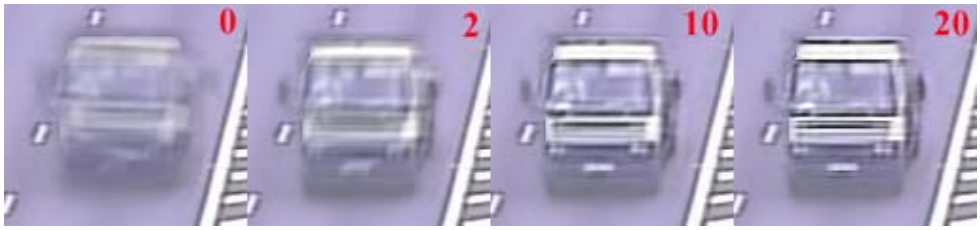


Figure 2.12: **Iterative reconstruction from PCA data.** The first image shows the mean appearance of the vehicle without extra detail added to it. The next images show the mean image with respectively 2, 10 and 20 PCA levels added. It is clearly visible that the first PCA levels contain the most important information, while the lower levels add only little detail.

Figure 2.13: **Vehicle trajectory extrapolation**. Left: extrapolated vehicle at the end of its trajectory. Right: Ground truth.

distorted PCA representation. 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 [Fitzgibbon 01, Schneider 01]. 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 2.13. From this we can conclude that the shape of the vehicle is approximated fairly well, while interlacing artifacts are partly smoothed out by the PCA algorithm.

2.3.2 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 requires 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.

Inter-vehicle occlusions are correctly resolved by rendering the sprites in back to front order, occlusions with the surrounding environment are taken care of by applying the occlusion map onto the rendered frames.

Step 1 consists of rebuilding the appearance of the vehicle for a frame t out of the PCA representation. As visualised in Figure 2.12 on page 24, where some examples of intermediate results are shown, we start off with the mean image μ, and iteratively add more detail :

$$Y(t) = \mu + \sum_i U(i,:) \times X(i,t); \tag{2.3}$$

This representation easily allows for interpolation to create in-between frames, or can be useful when the vehicle was occluded at some point in the original sequence. Interpolation between frames is straight-forward :

$$Y(t) = \mu + \sum_i U(i,:) \times \frac{X(i,t-1) + X(i,t+1)}{2} \tag{2.4}$$

2.4 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

Figure 2.14: **Extracted vehicles used in the renderings**, scaled to the same height for presentation purposes.

- 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 2.14.

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, a screenshot of a synthesized rendering is shown in Figure 2.15.

A different option, in the context of a commercial application, would be to introduce a database where the vehicles could be stored and queried in their compact representation.

Our synthesized videos suffer from such artifacts as interlacing and motion blur. These artifacts are already present in the input video (as can be seen in Figure 2.14), 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.

Figure 2.15: Synthesized image showing different vehicles in a small traffic jam.

Synthesized videos that look too polished provide an unrealistic training test for these systems, not corresponding with real world conditions.

Figure 2.16 shows all occlusions are automatically handled correctly. The videos show vehicles departing from underneath a bridge, where they are correctly hidden from view. 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 with respect to 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 a very small area in a few frames, rendering its impact on traffic detection systems negligible (see Figure 2.17).

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

Figure 2.16: Synthesized image illustrating occlusions are handled correctly, vehicles sprites are drawn underneath the bridge.

Figure 2.17: Example of an **edge artifact** that might appear around a vehicle when its orientation suddenly changes.

Figure 2.18: **Importance of introduced shadow maps.** Left: synthesized vehicle using a shadow map. Right: synthesized vehicle without using a shadow map.

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.

The effectiveness of our shadow maps in synthesizing the illumination effects around the vehicles is illustrated in Figure 2.18.

The illumination effects in our example input video example are restricted to shadows. We are confident that other illumination effects, such as headlights shining on the road, will be automatically included in our shadow maps if they are present in the input video. There is however a trade-off related to the shadow maps. The only illumination effects that will be added to the shadow map are those that are present inside the user-indicated windows of relevant information. The user can easily decide to increase the size of this window, but then the probability increases that other vehicles will intrude into the window and corrupt the shadow map. For these illumination effects, it is therefore recommended to extract only vehicles that remain at some distance from other vehicles in the input sequence. As applying the shadow maps is a multiplicative operation, the vehicles can be synthesized close together in the new videos without problems.

2.5 Conclusion and Future Work

In this chapter, we presented a video based method for rendering and animating rigid objects in fixed viewpoint video scenes. Our method was exemplified 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. This makes our method ideally suited for training traffic detection systems.

2.5.1 Future Work

Next to the data compression and the option to interpolate and extrapolate the data, our PCA vehicle sprite appearance model can furthermore be very useful to synthesize new vehicles. By means of analyzing the PCA data, one can create classifications for this data. For example, a class containing small vehicles can be created. Comparing the PCA data of this class with the PCA data of the vehicles that do not belong to this classification, some trait vectors can be identified, that are associated with corresponding parameters of the PCA model. Isolated trait vectors for different classifications related to size, color, etc, of the vehicles can then be replaced or interpolated to change the appearance of existing vehicles.

We would furthermore 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 representation, prediction and estimation techniques need to be investigated for this.

Chapter 3

Articulated Video Sprites and Image-Based Pose Synthesis

Contents

Figure 3.1: **Video Sprites** [Schödl 02] examples, synthesizing fish and hamsters.

3.1 Introduction

In this chapter, two new image/video based extensions to the well known Video Sprites [Schödl 02] technique are presented, allowing for animation of articulated characters, such as animals or humans. Video sprites are used to animate characters in a data-driven way by simply rearranging existing video frames, while maintaining smooth temporal coherence and the natural appearance present in the captured footage. The key to our technique is a matching algorithm that focuses on high-level 2D skeleton models instead of the character's visual appearance.

Traditional video sprites focus on simple characters without articulation (e.g. fish) or cases where the effects of articulation are negligible (e.g. flies or hamsters), from which some examples are shown in Figure 3.1. This focus limits the general usability of the video sprites technique and does not allow for a high level of control over the animation. Often it is necessary to control the animation in more detail (to the level of character poses), for instance to infuse emotion into a character's performance.

Articulated characters possess an underlying structure in the form of skeletons. When moving an arm or a leg of a character, underlying skeleton information can prove to be very useful. In the first part of this chapter, we exploit this property by using an articulated skeletal representation in our algorithms instead of the visual appearance of the subject.

When synthesizing an input video, a sequence of virtual skeletons is used to define a desired target animation. These skeletons are indicated by the user or can be acquired in a number of different ways, such as by motion capture or provided by an animator. The target skeletons are then matched with skeletons extracted from the source footage. The target skeletons only function as a guide for the new animation and do not force the input frames to match them exactly. This way, the animation will achieve the desired effect without sacrificing the natural movement and appearance of the filmed subject.

The second part of this section describes an extension to our work on articulated video sprites. We present a novel image-based pose synthesis technique that facilitates the creation of new human poses by synthesizing images, illustrated in Figure 3.2. When no input frame is found that matches a target skeleton in the articulated video sprites pipeline, a target-approximating image can be created using our image-based pose synthesis technique. This technique accurately reconstructs texture details by combining information from multiple input images. Given a user-specified 2D target pose, our solution merges different parts of the input images in order to conform to the desired pose, solely using 2D operations. Requiring little user intervention, image-based pose synthesis can create new poses to extend the input footage for the articulated video sprites.

Editing and creating photographs is becoming increasingly important, judging by the many powerful tools that are available today [Oh 01, Chuang 05, Lalonde 07, Avidan 07]. Image-based pose synthesis, which allows the creation of new human poses by synthesizing images, can be a useful component for some of these tools. The currently existing approaches create novel poses from single images by deforming a character using meshes [Igarashi 05, Hornung 07].

While deformation suffices for simple (rigid) objects and cartoon-like subjects [Haevre 05], it often results in distorted images when dealing with photorealistic images and human subjects, as it cannot reproduce changes in texture and illumination. Using multiple input images provides in higher realism for texture changes in local regions, like creases in fabrics at bent limbs. Standard deformation techniques do not account for these details.

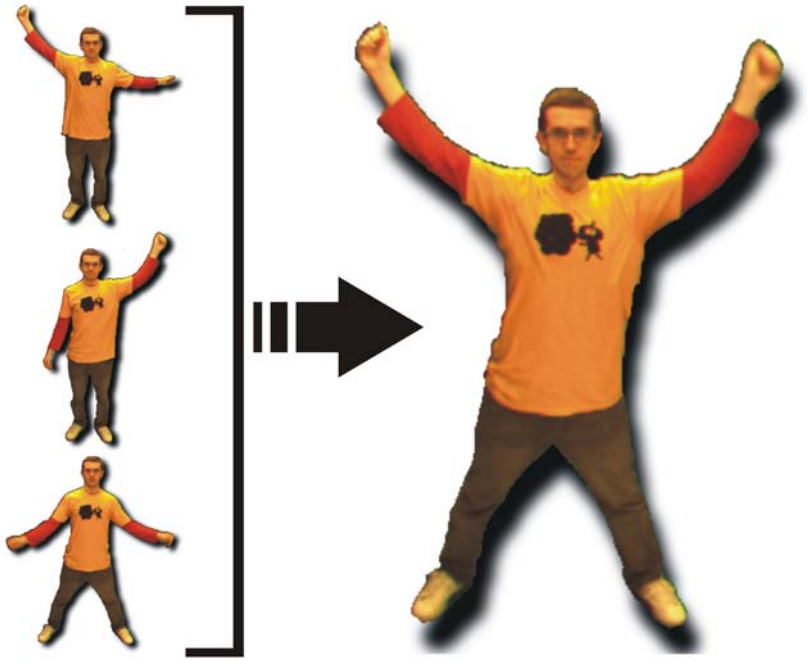


Figure 3.2: **Image-Based Pose Synthesis illustrated.** The images on the left hand side are used as input. The image on the right is the output of our technique, which basically reconstructs a new image using different parts of the input images and a target skeleton.

The main contribution of this work exists in relaxing the restriction of using only a single photograph for synthesizing novel poses in images, while balancing user-convenience and image quality. Synthesized images featuring characters in a user-specified pose are created from a small set of images (typically 2 to 4). Different parts from the sample poses are merged into a new whole, such that the desired user-specified pose is obtained.

In order to reach these goals, we provide an intuitive solution where the user annotates the input images with a simple approximate skeleton, which can be obtained through only a few mouse clicks. We also let the user draw a similar skeleton which serves as the target pose.

In addition, we also allow for mesh deformation in order to offer a larger degree of freedom concerning synthesizing images for a desired target pose.

Technically, there are two main challenges for our approach. First, we have to find correspondences between the character in the images and the bones of the user-specified skeleton. Afterwards, the character has to be subdivided into different body parts, which can be merged into a composite image.

Secondly, once we have a set of seperated bodyparts, they need to be fused seamlessly into a whole, while preserving texture details. When necessary, the synthesized image can finally be deformed to better match the user-specified target pose.

3.2 Related Work

Video-based representation, rendering and animation techniques have received increasing interest in the graphics and vision community in the past years. The biggest benefit of these video-based techniques, is that when starting from an input video, one already obtains of a high degree of photo-realism to start with. Compared to modeling a complex scene, video-based techniques simply use a video of the required scene.

A recurring video-based technique consists of rearranging frames in an input video to create novel footage, either globally on a per-frame basis [Schödl 00b], or locally on a per-sprite basis [Schödl 00a, Schödl 02]. Video sprites are obtained by filming an object and extracting it from the video, which can be done automatically using layers [Jojic 01],chroma keying [Schödl 02], or manually [de Juan 04]. Schödl and Essa animate these sprites by rearranging the frames of the original video [Schödl 02], a technique successfully used by Video Textures [Schödl 00b]. This approach allows for efficient animation of e.g. natural phenomena or small animals, with examples shown in Figure 3.3.

As these techniques are not able to work with more detailed, articulated characters, our work contributes to the video-based animation research domain by proposing a solution to cope with this limitation. Furthermore, we propose a distance measure aimed at working on high-level 2D skeletal representations of

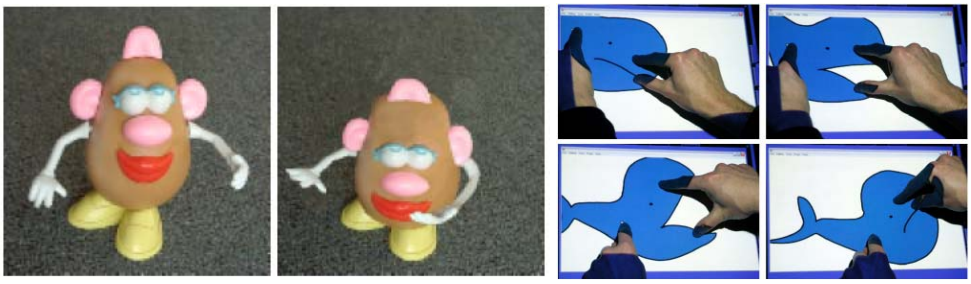
Figure 3.3: Left : **Video Textures** [Schödl 00b] example, with only a short recording of a flame, a seamless never-ending video is produced. Right : **Video Sprites** [Schödl 02] example, creating a video sprite of one fly, a synchronized fly-collective can be animated.

characters, instead of their visual appearance. Through a combination of different heuristics, we are able to animate a character according to a new sequence of target skeletons. Unfortunately tens or hundreds of input images might be needed to find a plausible match between input and target skeletons. Our Image-based Pose Synthesis technique significantly reduces the required number of input images.

Recently, researchers have proposed many advanced photo editing tools [Oh 01, Chuang 05, Lalonde 07, Avidan 07] based on computational techniques, ranging from seamless cloning [Pérez 03a] to automatic image-based rendering [Hoiem 05].

Techniques that allow for manipulating objects and characters in pictures are most relevant to our approach. Barrett et al. [Barrett 02] provide real-time animation and manipulation of static digital photographs, as illustrated in Figure 3.4(a). Individual image objects can be selected, scaled, stretched, bent, warped or even deleted (with automatic hole filling) - at the object, rather than the pixel level - using simple gesture motions with a mouse.

Igarashi et al. [Igarashi 05] (see Figure 3.4(b)) propose As-Rigid-As-Possible (ARAP) shape manipulation, which employs a mesh-based representation to deform objects in a photograph. They allow users to move and deform two-dimensional shapes without manually establishing a skeleton or freeform deformation (FFD) domain beforehand. The shape is represented by a triangle mesh and the user moves several vertices of the mesh as constrained handles. The ARAP system then



(a) Object-Based Image Editing (b) ARAP Shape Manipulation

Figure 3.4: Examples of **related character manipulation techniques** in images, with Object-Based Image Editing [Barrett 02] on the left and As-Rigid-As-Possible shape manipulation [Igarashi 05] on the right. As illustrated, these are most useful in cartoon-like images without highly detailed textures.

computes the positions of the remaining free vertices by minimizing the distortion of each triangle using a two-step closed-form solution that achieves real-time interaction.

Wang et al. [Wang 08] propose 2D shape deformation based on rigid square matching. Their method places a control mesh over the subject and uses a rigid shape matching method to find an optimal pure rotational transformation for each square in the control mesh. As shown in Figure 3.5(a), this shape deformation method is especially suitable for applications in cartoon character animation. Hornung et al. [Hornung 07] (see Figure 3.5(b)) propose a more elaborate method for deformation-based pose editing, and even allow for character animation using 3D motion data. In general, deformation-based techniques provide the user with flexible control over the shape of an object or character. Unfortunately, deformation by itself is unable to model changes in textures and illumination.

In this work, we take images from more than one pose into account, which allows for synthesizing more realistic results. At first sight our method may seem to require a significant amount of user interaction, because more than a single input photograph is used. However, specifying a very simple skeleton already leads to good results. The entire process typically requires less than a minute of work per

(a) 2D Shape Deformation Based on Rigid Square Matching

(b) Character animation from 2D pictures and 3D motion data

Figure 3.5: Examples of **related shape deformation techniques** in images, with the work of Wang et al. [Barrett 02] on the left and Hornung et al.[Hornung 07] on the right. These techniques can be used with more interesting characters, unfortunately, deformation by itself is unable to model changes in textures and illumination.

input image, as in our approach, one skeleton consists of only 21 joints.

Kavan et al. [Kavan 08] proposed a system of 2D polygonal impostors called *Polypostors* (see Figure 3.6). Characters are manually decomposed into bodyparts, which are overlayed with polygons. These polygons are used for character deformation and animation. One of the limitations of the Polypostors is that deformations can not deviate far from the inital key-frame, restricting the application to simple walk cycles.

Also taking several input images into account, Bregler et al [Bregler 02] provide a cartoon motion representation that allows to isolate the motion style of an existing cartoon animation and to apply the same style to a different subject/animation. Instead of working with skeletal models, user-identified key-shapes and hand made contours are used for retargeting purposes. Affine transformations on the contour are applied and in-between shapes are interpolated between the key-shapes. This technique is aimed at animating cartoon-like subjects. As shown in Figure 3.7,

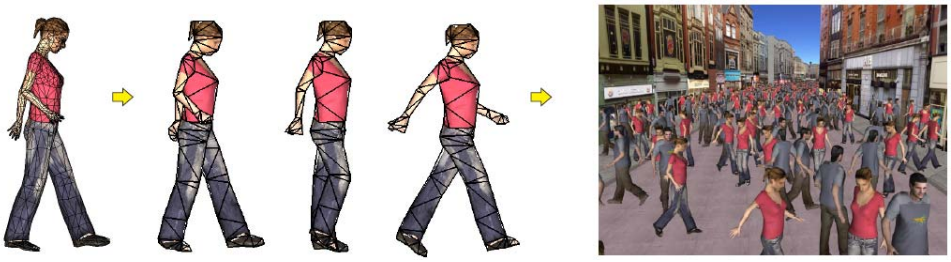


Figure 3.6: Example of 2D polygonal impostors called **Polypostors** [Kavan 08], they are manually decomposed into bodyparts and overlayed with polygons. This technique is limited to simple walk cycles.

Figure 3.7: Examples of the work of Bregler et al [Bregler 02], where a motion style of an existing cartoon animation is applied to a different subject. While this works convincingly for cartoon-like subjects, texture details of (photo)realistic characters are not taken into account in their key-shape interpolation.

animation works well on shapes that don't feature detailed textures, most of the examples only contain one or two colors and no shading or details at all. An advantage for this technique is that a cartoon animation is easily perceived as "correct" by the human eye. Working with photorealistic images (e.g. featuring human subjects), this kind of shape interpolation can be useful, but artifacts will be clearly visible when texture details are not taken into account. In our case, we simply reuse the existing input frames, assuming these input frames already look realistic.

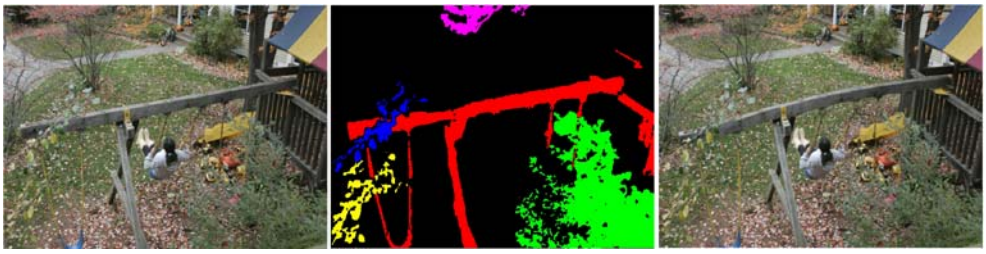


Figure 3.8: **Motion Magnification** [Liu 05], acts like a microscope for visual motion. After identifying different motions in a video sequence, users are allowed to choose a motion cluster and magnify it without objectionable artifacts.

Another important technique in the field of video based animation is the Motion Magnification work proposed by Liu et al [Liu 05]. Given an input video, the Motion Magnification algorithm can amplify subtle motions that would otherwise be invisible. All kinds of motions can be clustered and magnified, ranging from small motions like a baby's breathing to large motions like someone playing on a swingset. Where this technique shows promising results (as seen in Figure 3.8), one cannot deviate from the subtle movements that are already present in the video, only enlarge them. While our aim is to re-animate sprites from an input video given a novel target animation path, the work of Liu et al. aims to repeat the animation of the sprites from the input video, but in an exaggerated way without objectionable artifacts.

When considering multi-camera based pose synthesis and animation techniques like the work of Starck et al. [Starck 05, Starck 07] (see Figure 3.9), it is obvious that even though multi-camera setups allow for a higher sense of realism in reconstruction and animation, their cost and complexity currently makes them inaccessible office/living-room deployment. This is illustrated by Starck et al. [Starck 07], who use eight High-Definition Thomson Viper cameras and 8 recording PC's in a calibrated chroma-key studio environment to create a surface capture of a dynamic character. Our approach differs in that we start from two-dimensional footage instead of data acquired from multiple viewpoints. As such, a less restrictive recording setup is required for our method and we can even incorporate existing footage which was not originally shot to be reanimated.

Figure 3.9: Example of **multi-camera based pose synthesis and animation** from Starck et al [Starck 05]. These complex multi-camera setups allow for a higher sense of realism in reconstruction and animation

3.3 Articulated Video Sprites

Our method starts with a preprocessing step, in which sufficient video footage is collected. Since we are going to match skeletons later on, optimal results will be achieved if the video is shot with a static camera, keeping internal and external parameters unchanged. A video of a person standing still, shot with a moving camera, features 2D skeletons changing on the image plane, while they are inherently supposed to stay equal since the subject is not moving.

From the input video footage, 2D skeletons are semi-automatically extracted. Keeping occlusions in mind, a semi-automatic approach is the best and safest option compared to automatic tracking in 2D. This way, the user can intervene when automatic tracking drifts away from the subject. Next, we find the best matches between these skeletons and the target sequence of skeletons, incorporating several heuristics. Finally, our results are refined by taking into account blocks of distinct motions. Throughout this section, we will illustrate our techniques on a specific example. We start from a large source video in which the subject character performs several movements, amongst them the arm gestures for the letters Y, M, C and A. The target animation to which we want to match our source video consists of another person performing the famous YMCA routine.

3.3.1 Preprocessing

Similar to previous techniques [Schödl 00a, Schödl 02], we start by amassing a sufficient amount of video material of the subject that we wish to animate. This footage can be acquired in a studio, on location, or from already existing material. Because we will be reusing existing frames, every posture of the subject in the target animation must approximately appear in the source footage. A target animation featuring a person waving his hand will for example not find a good match in source material that only shows a person sitting down with crossed arms.

Skeletons are semi-automatically extracted from the source footage with user-assisted optical flow tracking. Aimed towards tracking and representing human characters, we choose a simple 2D skeleton, consisting of 21 nodes, as illustrated in Figure 3.10. The user indicates the skeleton joint positions in the first frame. These are then tracked across the entire video by an optical flow algorithm. We choose to use Lucas & Kanade's optical flow technique [Lucas 81b]. This algorithm assumes that the flow is essentially constant in a local neighbourhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighbourhood by the least squares criterion. Occlusions are easily handled in the GUI by having the user indicate problematic frames. The joint positions in these frames are then interpolated based on its positions in several frames before and after the occlusion.

The target animation is provided as a sequence of target skeletons. These skeletons can be produced by an animator, provided through motion capture, or extracted from video footage the same way as the source skeletons. The only requirement is that both sequences feature the same skeletal structure, avoiding matching a human skeleton with a reptile skeleton for example.

To broaden the input range for target skeletons, a conversion script was written to transform the skeleton data from the Carnegie Mellon Graphics Lab Motion Capture Database [CMU] to our own skeleton datastructure. The choice not to use the CMU skeletons for our user input is based on the fact that these skeletons are too detailed for our purpose, using four nodes for just one hand for example, which in our representation uses only a single node. The skeletons used in our algorithms are

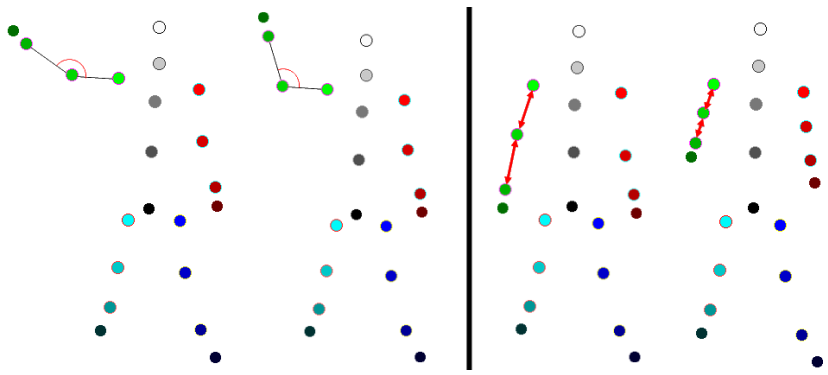


Figure 3.10: Illustration of the **2D skeletons** used in our technique. These skeletons consist of 21 nodes. As for skeleton matching, the angle between parent/child nodes as well as the length of the limbs are taken into account while the absolute positions of the nodes are discarded.

derived from this standard, but only use a subset of nodes.

3.3.2 Skeleton Matching

Once all input frames and skeletons are available, we can calculate the correct order of source frames matching the target animation. This is done by matching the skeletons extracted from the source material to the provided target skeletons. For our matching cost, we do not compare absolute positions of the skeleton joints, as these are dependent of the scene and body type of the actors or animation models. Instead, we compare the angles and length ratios between adjacent limbs L_i. These encode the 2D posture of the skeleton sufficiently and, through the ratios, implicitly include some 3D information. The matching cost $C(S,T)$ between two skeletons S and T with respective limbs L_{Si} and L_{Ti} is formulated, with α and β user-defined weights:

$$C_a(S,T) = \sum_{i=0:N-1} \left[angle(L_{Si}, L_{S(i+1)}) - angle(L_{Ti}, L_{T(i+1)}) \right]^2 \tag{3.1}$$

$$C_l(S,T) = \sum_{i=0:N-1} \left[\frac{|L_{Si}|}{|L_{S(i+1)}|} - \frac{|L_{Ti}|}{|L_{T(i+1)}|} \right]^2 \tag{3.2}$$

$$C(S,T) = \alpha C_a(S,T) + \beta C_l(S,T) \tag{3.3}$$

The incorporated implicit 3D information can be explained as follows. When a person's arms are held towards the camera, the visible length of the arms projected onto the image plane will be small. When held almost parallel to the image plane, the projection of the arms will be much larger. This way, the relative lengths of the projections can implicitely reveal useful 3D information related to the pose of the character.

For every target skeleton, matching costs are calculated with the available input skeletons. At the same time we also ensure that the chosen skeletons follow each other in a smooth temporal manner by forcing consecutively chosen source skeletons to have small matching costs.

Two perceptually disturbing artifacts can occur using this skeleton matching strategy. When a frame is chosen continuously for some time, for example because no better match is found for the target skeletons, the animation of the character will appear frozen. Furthermore, a visual stuttering in the animation can occur when the chosen frame oscillates between nearby frames. Using dynamic programming, we combine costs for these situations to find an optimal new frame order. Firstly, frames that are re-chosen consecutively will get an increased cost that is proportional to the length of the repetition. Secondly, winning frames that cause fluctuations in the selection (e.g. jumps from frame 1 to frame 5 to frame 2 to frame 6 ...) will also be penalized to ensure a smooth transition between selected frames. An outline of our algorithm is provided in Table 3.1 on page 46.

Given the number of target skeletons T and source skeletons S, a "Linkback" table is created with $T-1$ rows and S columns. Starting from target skeleton T_0, we calculate and temporarily store the matching cost with each source skeleton S_j in vector *Prev*. From now on, moving forward in time, for every target skeleton T_i and for every source skeleton S_j, we select source skeleton S_k that minimizes the following cost function where α and β are user-specified weights:

$$\begin{aligned} Cost(i,j,k) &= Prev(k) + C(T_i,S_j) + C(S_j,S_k) \\ &+ \alpha\, checkRepeat(i,j,k) + \beta\, checkFluctuation(i,j,k); \end{aligned}$$

This cost function can be summarized as follows:

- S_k should have a good match with the previous target skeleton T_{i-1}

- S_j should have a good match with T_i.
- S_k should have a good match with S_j
- repeatedly choosing the same source skeleton will be penalized
- fluctuations in skeleton selection will be penalized

The best k is assigned to $Linkback(i, j)$ and the best matching costs are added to the *Prev* vector for the next matching round for target skeleton T_{i+1}.

When the entire Linkback table has been filled in, the final frame selection is made from back to front by going to through the Linkback table, as described in Table 3.1.

The efficiency of the introduced penalties is illustrated in Figure 3.11 on page 47. The nodes on the graphs indicate which source frames (vertical axis) are chosen as best matches for the target animation frames (horizontal axis). As can be seen in Figure 3.11(a), the absence of the introduced penalties results in a fluctuating and rather unstructured frame order. Fluctuations clearly visible with the selected skeletons for the letter "C" and obvious frame freezes for the letters "M" and "A". A smooth animation selection obtained by incorporating the penalties is shown in Figure 3.11(b). This whole matching process is executed automatically without any manual intervention.

3.3.2.1 Matching Refinement

A further matching refinement is motivated by the fact that motion of articulated characters such as humans, can often be subdivided into a chain of distinct movements. In the YMCA example, four motion blocks can clearly be identified in the graphs from Figure 3.11, namely one distinct block per letter movement. These atomic movements are automatically detected by analyzing the sequential matching costs in the skeleton sequences. Their boundaries are detected as local maxima of this sequential matching function.

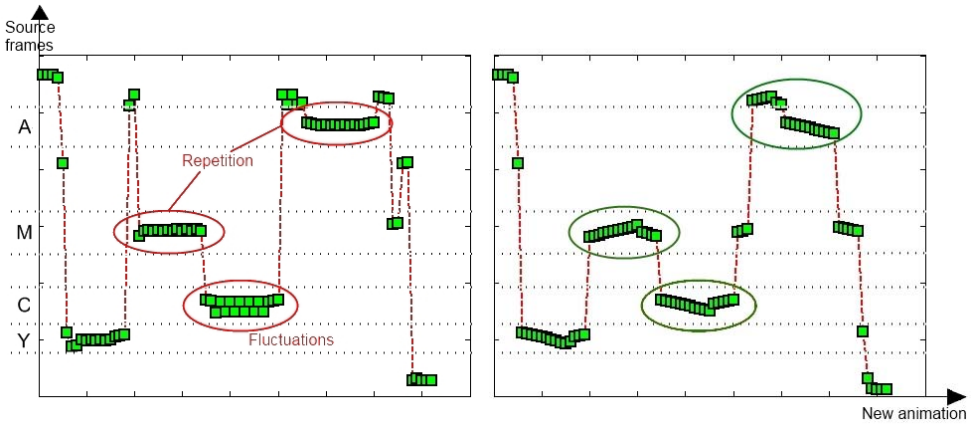
After a global pass of our matching algorithm, the chosen animation path is now refined by restricting the matching process to smoothly stay inside the blocks of distinct movements. When combining frames from several distinct motions to create a

1. %match first target skeleton with all source skeletons
   $Match := [C(T_0,S_0),C(T_0,S_1),C(T_0,S_2),...,C(T_0,S_N)]$
2. $\forall i = 0 : \#T - 1$
   (a) $min := 0$;
   (b) $Prev := Match$; %store the previous matching results
   (c) $\forall j = 0 : \#S$
      i. $\forall k = 0 : \#S$
         A. %match current target skeleton with current source skeleton
            $w := C(T_{i+1}, S_j)$;
         B. $w+ = checkRepeat(Linkback, i, j, k)$;
         C. $w+ = checkFluctuation(Linkback, i, j, k)$;
         D. %adapt cost to select source skeleton that matches best with S_j
            $cost := Prev[k] + w + C(S_j, S_k)$;
         E. `if` $cost < min : Match[j] := cost; Linkback[i, j] = k$;
3. $min := MAX$;
4. $\forall i = 0 : \#S$
   (a) `if` $Match[i] < min : chosenframes[\#S] := i; min := Match[i]$;
5. $\forall i = \#S - 1 : 0$
   (a) $chosenframes[i] := Linkback[i][chosenframes[i+1]]$;

%checkRepeat recursively checks if $Linkback[i-1][k] == j$
%checkFluctuation recursively checks if $j - k$ and $Linkback[i-1][k]$ have the same sign

Table 3.1: Skeleton matching using dynamic programming

new animation, temporal artifacts may occur. For example, the movement of putting a person's arms in an "M" position initially looks similar to putting them in an "A" position. Mixing "M" and "A" frames together to match with a target movement could induce temporal artifacts if the subject is smiling in the "M" frames looks angry in the "A" frames.



(a) Without penalties (b) With penalties

Figure 3.11: **Results of the skeleton matching algorithm.** The nodes on the graphs indicate which source frames (vertical axis) are chosen as best matches for the target animation frames (horizontal axis). Comparing Figures (a) and (b), the introduced penalty costs clearly remove the freeze frames (frames chosen for the letters “M” and “A”) and oscillations (letter “C”) from the original skeleton selection.

Both the source skeletons and the target skeletons are divided into atomic movements. For every target movement block, the best matching source movement block needs to be selected. For every source skeleton that matched best with these target skeletons in the previous round, a counter is added to their source movement block. The source block with the highest number of chosen skeletons is then selected as best matching the target movement block. We now restart the matching process, adding the restriction that per target movement block, one can only match with skeletons sampled from the selected source movement block.

Since movements commonly are not all performed using the same velocity, the size of the matching movement blocks might differ. When a source movement block is smaller than its associated target movement block (the source movement was performed faster than the target movement), there simply are not enough source skeletons to sample from for each target skeleton. In this situation, the repeat restraining cost is softened.

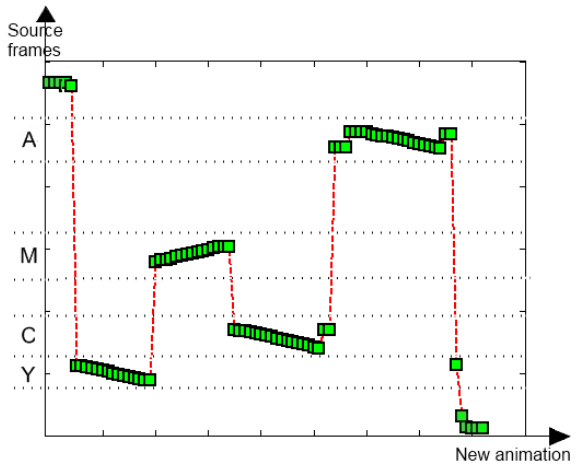


Figure 3.12: **Final frame selection after motion matching refinement.** Starting from the frame selection in Figure 3.11(b), these results are refined by restricting the matching process to smoothly stay inside the bounds of the distinct movement blocks and by imposing penalty costs on large sequential jumps inside these blocks.

When a source movement block is larger than its associated target movement block (the source movement was performed slower than the target movement), a large amount of source skeletons is available to sample from for each target skeleton. In this situation, small jumps from one frame to the other are allowed.

Furthermore, ensuring a continuous flow in frame selection, penalty costs are imposed on large sequential jumps inside the source movement blocks.

While in Figure 3.11(b), the letter “A” in the new animation was still composed from frames from the letters “M” and “A”, “A” has now become more consistent and coherent, reconstructed using only “A” frames, as illustrated in Figure 3.12.

3.3.3 Results

As shown in Figure 3.13, our algorithm correctly matches different poses even though the visual appearance differs greatly and the source and target skeletons have no exact matches. Furthermore, a large variety of poses to choose from were present in the original source footage. The relevant movements and poses were extracted

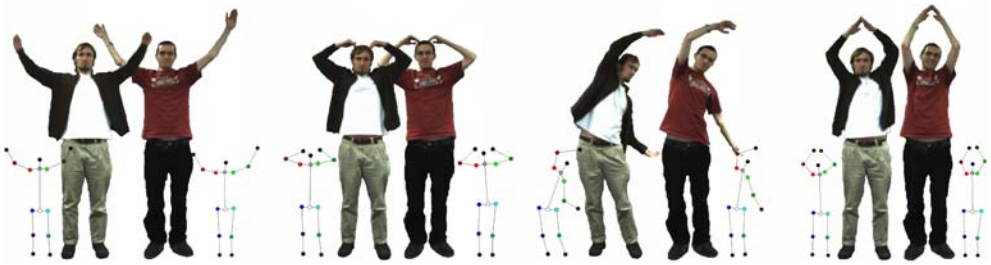


Figure 3.13: **Results of the Articulated Video Sprites algorithm.** Starting from a video clip of the person on the right, consisting of arbitrary movements, 2D skeletons are extracted semi-automatically, providing a high level description of the recorded motion. These skeletons can then be used to match frames to a user-specified 2D motion. In this example, we used the extracted skeletons of the person on the left to animate the person on the right.

from this large amount data, while the other poses were successfully filtered out. Our rendered video shows the full smooth animation from which the examples in the figure were extracted.

Already achieving promising results, this technique suffers from the restriction that good source frames matching the target animation are always required. Furthermore, to ensure a smooth result, slower movements are more useful than fast action since they present more frames to sample from. To loosen up this restriction and therefore providing more freedom related to choosing desired target animations, a pose synthesis technique has been developed. This technique allows combining parts of different input frames into novel character poses, increasing the size of the input pose space significantly.

3.4 Image-Based Pose Synthesis

The Image-Based Pose Synthesis technique can be seen as an extension on our work with Articulated Video Sprites [Vanaken 06a]. This technique facilitates the creation of new human poses by synthesizing images, as illustrated in Figures 3.2 (page 34), 3.14 (page 52) and 3.15 (page 53). When no input frame is found that adequately matches a target skeleton in the articulated video sprites pipeline, a target-approximating image can be created using our image-based pose synthesis technique.

This technique accurately reconstructs texture details by combining information from multiple input photographs. Given a user-specified 2D target pose, our solution merges different parts of the input photographs in order to conform to the desired pose, solely using 2D operations. Requiring little user intervention, image-based pose synthesis can create new poses to extend the input footage for the articulated video sprites.

A schematic overview of our approach is shown in Figures 3.14 and 3.15. This approach is now introduced briefly, the steps taken are detailed in the following paragraphs.

First, the user overlays 2D skeletons on each of the input images and specifies a desired target skeleton to which the resulting pose needs to adhere. This is done by simply marking the joints of the body in the input images using mouse input. Typically this process requires less than a minute of time per input image, as one skeleton consists of only 21 joints. In addition, if possible, we automatically extract the foreground sprite (i.e., the character) from each input image using a background subtraction technique [McIvor 00]. When background subtraction fails, semi-automatic or manual segmentation can be performed.

Given the input skeletons and segmented images, for each body part in the target skeleton a best match is sought using a skeleton-based distance measure, as described in Section 3.3.2. Afterwards the best matching body parts are extracted from the input images and merged to form the the resulting synthesized image.

For each input image, a mesh is overlayed on the character in an edge-aware fashion, using constrained Delaunay triangulation [Chew 87]. Hence, image regions

can be associated with the skeleton bones, and in turn also with each body part. When all possible combinations of input poses deviate too far from the desired target pose, the constructed mesh can be deformed to better match the target pose.

Once all useful input image regions have been automatically selected, they are overlayed to prepare the final image. Some parts of these regions might be overlapping in image space. Simple averaging could be performed to take care of these overlapping regions in the final image, yet this operation often introduces ghosting artifacts. We therefore choose to fuse overlaps while respecting the continuity of the image, as discussed in Section 3.4.2.

3.4.1 Matching Body Parts

In this section, we illustrate in detail how body parts are matched and extracted. At a later stage, these extracted body parts will be used to form the final pose, as described in Section 3.4.2.

3.4.1.1 Body Part Selection

As a first step, the given skeletons are subdivided into predefined body parts: legs, arms, head and torso. Skeleton matching will then be performed on each separate body part. For the matching itself, the skeleton matching technique described in Section 3.3.2 is used, comparing 2D angles of consecutive skeleton joints and length ratios of skeleton limbs instead of evaluating absolute positions. For each target body part, we keep the best match and discard the other input body parts.

Unfortunately, an absolute winner is not always found when matching. This situation can be detected by computing the differences between the matching costs. If these cost differences are below a given threshold, we can conclude there is no unique winner and therefore the best candidates are retained. These (overlapping) candidates will later on be combined into a single image (see Section 3.4.2). This often occurs with the torso body part for example.

3.4.1.2 Mesh Creation

In order to transfer the selected body parts to the final image, a link needs to be made between the input skeletons and the image data itself. To this end, background

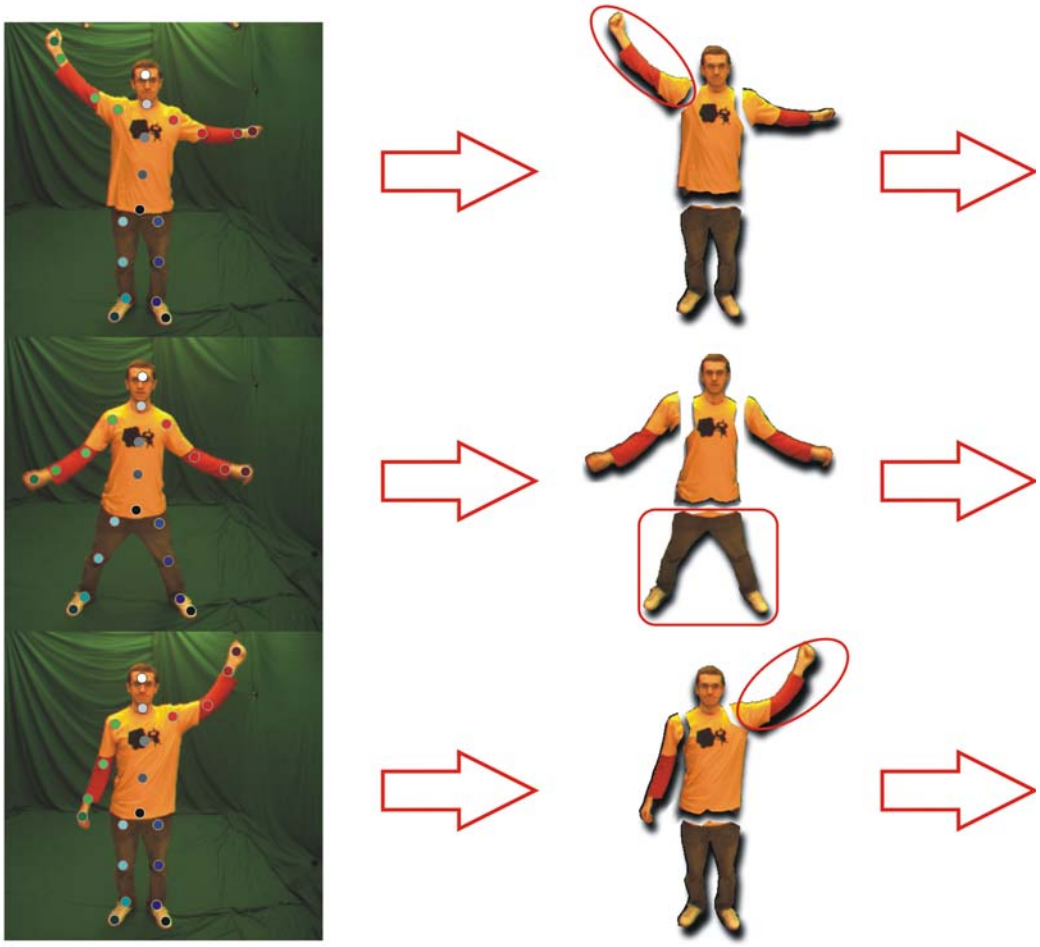


Figure 3.14: **Schematic overview of our algorithm, part I**. Starting from a desired target skeleton and a set of input images tagged with associated skeletons, we take those parts of the input poses which best match with the target pose. In this case, one arm from input image 1 matches best with the arm of the target skeleton, the legs from image 2 match best with the legs from the target, and one arm from image 3 matches best with the target. The torsos all match evenly well and need to be fused with the arms and legs. (see Figure 3.15)

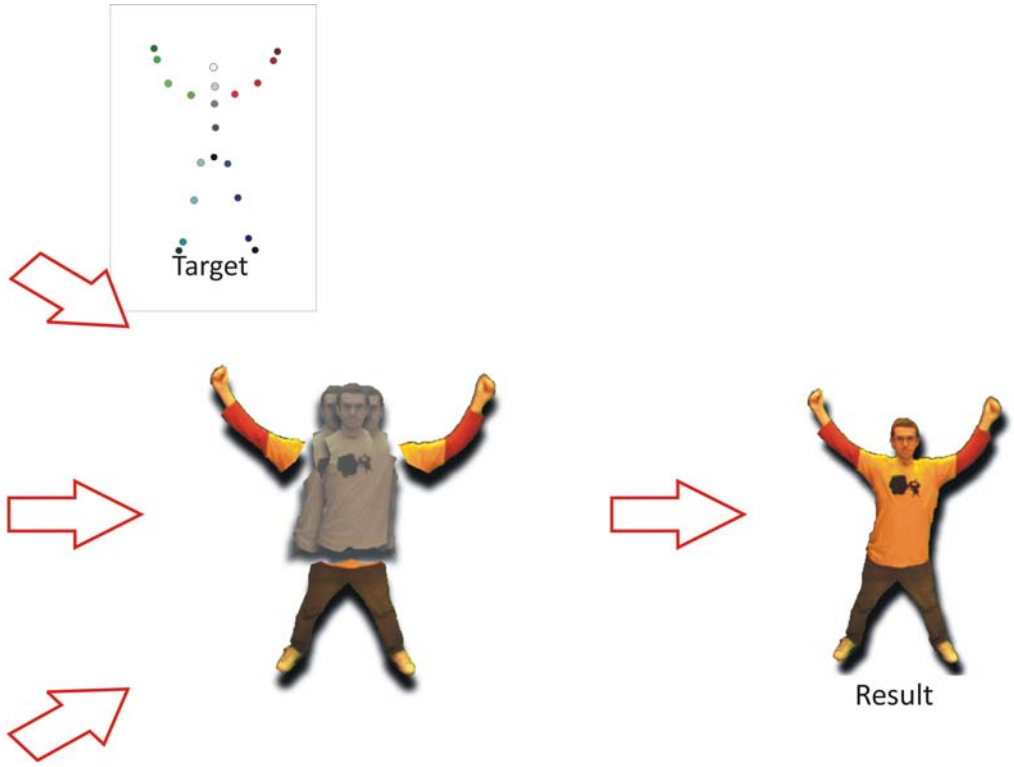


Figure 3.15: **Schematic overview of our algorithm, part II**. Following Figure 3.14, the segmented input sprites are subdivided related to the associated skeleton parts. The confident sprite parts (arms and legs) are transfered to the resulting image, while the best solution for the remaining body parts is infered using our fusing algorithm. The color code in the skeleton is used for visualization purposes only.

subtraction is performed in order to separate the subject from the background (from here on we refer to the resulting image region as a "sprite"). Now every pixel of the input sprites needs to be linked with the input skeletons.
Instead of simply calculating the Euclidian distance between pixels and skeleton joints, our distance measure is based on the skeleton structure as well as color and edge information of the sprites. For this, a mesh is placed over the sprite, combined with a per triangle distance.

Instead of using uniformly sampled meshes, we aim at obtaining meshes for which the triangle design represents certain meaningful parts of the sprite. This way, the sprite can be divided into different regions covering different body parts.
Therefore, a mesh is created influenced by the structure of the skeleton as well as by color information of the sprite. Color information of the sprite is used to ascertain that triangles will only contain pixels with a more or less equal intensity. This way high frequency areas in the image will be overlayed with smaller triangles, while low frequency areas will be collected in larger triangles, assuring the process of cutting and pasting triangles to be more trustworthy compared to when working with a uniform mesh that discards all color and skeleton information.

The mesh is furthermore constructed in an edge-aware fashion. Edge-awareness ensures that cuts occur where smooth transitions are needed, avoiding seams in the final composite image. Initially, the outer vertices of the mesh are placed on the silhouette of the sprite. The inner vertices are constrained to the bones of the input skeleton as well as to points on the edges of the sprite, obtained from an edge detector, for which we employed directional Sobel filters [Gonzalez 01]. See Figure 3.16 for an illustration of the mesh creation process.

To allow for a large variety of attainable target poses, the mesh can be deformed using the *As-Rigid-As-Possible shape manipulation* algorithm of Igarashi et al. [Igarashi 05], where the skeleton joints act as control points for the mesh. As motivated in Sections 3.1 and 3.2, using this deformation algorithm on a single image creates too large distortions. However, in this case it is merely used for small deformations to better match the target skeleton in the final result.

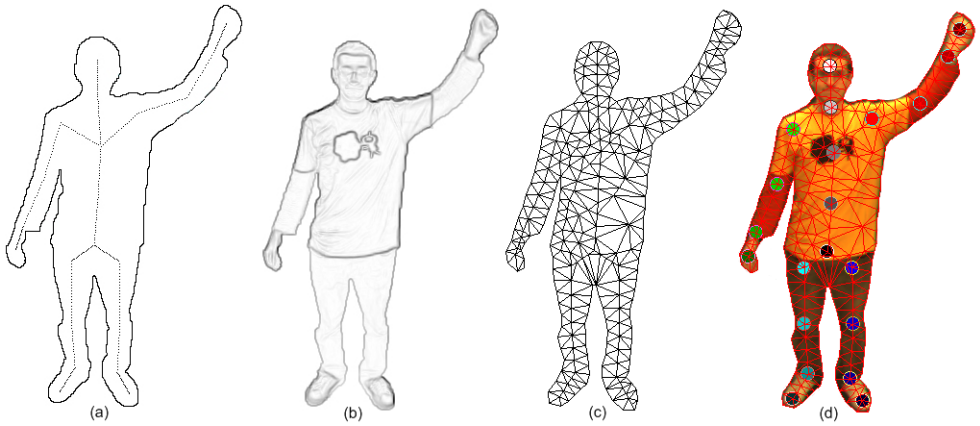


Figure 3.16: **The mesh creation process.** Using the silhouette of the sprite combined with the skeleton bones (a) and an edge image of the sprite (b), a mesh is created (c) that fits the input sprite (d). The colored circles in (d) represent the user-indicated skeleton joints.

3.4.1.3 Pixel Selection

For the purpose of collecting image regions that cover a given body part - or more specifically, collecting pixels that cover a given body part - each mesh-triangle is assigned to its nearest skeleton bone(s) as follows:
If one of a triangle's vertices is located on a skeleton bone, we assign it as being part of that bone's body part and mark its status as "confident". For the remaining triangles, we look for the two closest skeleton bones by comparing the L_2 distance between the triangles' centroid and the skeleton joints. If a triangle's two closest bones belong to the same body part (for example "upper arm" and "lower arm") then its status is marked as "confident", otherwise (for example "upper leg" and "lower back") it is marked "uncertain". An example for this classification is shown in Figure 3.17.

From the set of confident triangles, we keep those that belong to the required body part. The remaining triangles are those triangles that either belong to a body part for which we did not find a unique match, or those of which we are uncertain of if they belong to a required bodypart. These triangles are collected from the different input sprites, overlappingly placed on a grid, and merged with the fusing algorithm

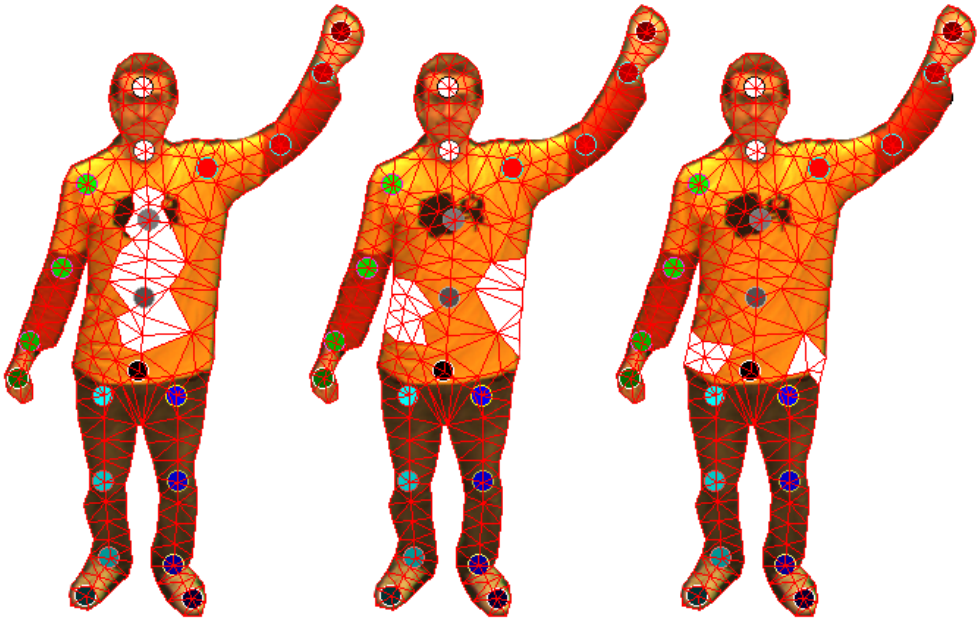


Figure 3.17: **Illustration of the triangle classification.** Triangles with a vertex located on a skeleton bone are classified as “confident” (left). Triangles of which the two closest bones belong to the same body part (the torso in this case) are also marked as “confident” (middle). Triangles that are close to several body parts are classified as “uncertain” (right).

as discussed in the following paragraph. For example, the arms and the legs from Figures 3.14 and 3.15 on pages 52 and 53 are classified as “confident”, while the torsos of the three input sprites are classified as “uncertain”. These torsos will need to be merged to form a consistent image together with the “confident“ body parts.

3.4.2 Fusing Body Parts

At this point, a set of “confident” and “uncertain” image regions has been collected from the different input images, conforming to the desired target pose. The final step is to fuse these regions into a consistent whole. In order to avoid mismatches, the individual input body parts are translated on the image plane using the target skeleton as a reference, as shown in Figure 3.18. When the selected body parts don’t all closely match the target skeleton, the As-Rigid-As-Possible shape manipulation

Figure 3.18: **Individual body parts translated to a reference image space.** A collection of "confident" (arms, legs) and "uncertain" image regions collected from different input images. When overlayed with a lattice of overlapping square patches, an optimal solution will be sought using belief propagation.

technique of Igarashi et al. [Igarashi 05] is applied on the triangles of these body parts to ensure that the resulting image better matches the target pose.

The final sprite could now be obtained by simply averaging overlapping regions. Unfortunately, this may lead to ghosting artifacts, as the input images can contain different texture details (e.g. due to wrinkles in clothing, see Figure 3.21 on page 62). These texture details are clearly important to preserve a sense of realism in the synthesized images. Therefore, a more elaborate approach is taken.

Given the sets of "confident" regions and "uncertain" regions and their binary masks, a visually coherent resulting image will be created by means of a discrete Markov Random Field (MRF). Aiming for a coherent image, our goal is to combine image regions that feature good overlap costs. Therefore we decide to simply use an MRF with a regular 2D lattice instead of working with non-overlapping triangles. The set of nodes N is defined by placing a regular grid of square patches on top of the resulting image space, all with an horizontal and vertical overlap.

Each node in the MRF is uniquely defined by their (x,y) coordinate in the image space, and the edges E of the MRF are defined by the 4-neighborhood of each individual node. The label set L consists of all possible $w \times h$ patches around every node $n_i \in N$, defining the labels $l \in L$ uniquely by the spatial coordinates (x,y) of their center pixel, and their input frame number $t \in [1,N]$. Note that these labels l are only sampled from the "uncertain" regions, as the "confident" images regions can be seen as fixed and no choice is needed to be made for those parts.

Every node $n_i(x_i,y_i) \in N$ has a maximum of N possible label candidates $l(x_l,y_l,t)$ where $(x_l,y_l) = (x_i,y_i)$. Also, a label (x,y,t) will only be considered a valid candidate if the full patch window is marked in the binary mask B, or if it is only partially marked but located on the border of an image region.

Nodes n_i located on the border of a "certain" image region - a fixed image region - will already contain some initial content. Therefore, any label $\hat{l}_i$ assigned to these nodes should retain as much intensity information present as possible. As such, the single node potential $V_i(l)$ of assigning label l to node n_i represents how well the intensity information of label l agrees with the intensities present in the window W around the center of node n_i:

$$V_i(l_t) = \alpha \left[\frac{1}{|W|} \sum_{(x,y) \in W} B_i(x,y) \left(I_i(x,y) - I_t(x,y) \right)^2 \right] \tag{3.4}$$

In case nodes i and j are spatial neighbors, the pairwise potential is defined by the normalized sum of squared differences (SSD) over the area of overlap A, assuring continuity from one patch to another:

$$V_{ij}(l,l') = \beta \left[\frac{1}{|A|} \sum_{(x,y) \in A} \left(I(x,y) - I'(x,y) \right)^2 \right] \tag{3.5}$$

Based on these formulations, where α and β are user-specified weights, a label $\hat{l}_i \in L$ should be assigned to each node n_i, so that the total energy cost $E(\{\hat{l}_i\})$ of the MRF is minimized, where:

$$E(\{\hat{l}_i\}) = \sum_{i=1}^{|N|} V_i(\hat{l}_i) + \sum_{(i,j) \in E} V_{ij}(\hat{l}_i, \hat{l}_j) \tag{3.6}$$

With the problem of image merging formulated as an energy minimization problem, we can now apply belief propagatoin to our energy function.

Belief propagation (BP) is an iterative inference algorithm that works by propagating local messages along the nodes of an MRF [Yedidia 01]. Messages sent from node n_i to n_j form a set $\{m_{ij}(l)\}_{l \in L}$, where element $m_{ij}(l)$ indicates how likely node n_i thinks that node n_j should be assigned label l. Furthermore, messages are updated (i.e. sent) until convergence as follows:

$$m_{ij}(l) = \min_{l_i \in L} \{V_i(l_i) + V_{ij}(l_i, l_j) + \sum_{k:k \neq j,(k,i) \in E} m_{ki}(l_i)\} \tag{3.7}$$

This update rule is associated with the min-sum version of BP, where the potentials are described in the *-log* domain. After convergence, a set of beliefs $\{b_i(l)\}_{l \in L}$ is computed for each node, where belief $b_i(l)$ is defined as follows:

$$b_i(l) = -V_i(l) - \sum_{k:(k,i) \in E} m_{ki}(l) \tag{3.8}$$

These beliefs approximate the max-marginal of the posterior at node n_i, and thus describes the likelihood that the label l should be assigned to that node. Based on this fact, a node is then assigned the label with the maximum belief, i.e. $\hat{l}_i = \arg\max_{l \in L} b_i(l)$.

3.4.3 Results and Discussion

In this section we discuss results obtained with our technique. Figure 3.2 on page 34 comprises of three input photos, taken in front of a green screen. Chroma-keying is used to extract the sprites. The target pose consists of legs and arms that are spread open. Our algorithm has automatically chosen the lower body from input image 3, while the arms in the result were taken from input image 1 and 2. The torso and the head were taken from all, and combined using our fusion method.

The second example (shown in Figure 3.19) uses two input images that were taken from a different camera position. This shows that our technique does not necessarily restrict the input images to originate from a static camera.

Figure 3.20 shows two input images of supermodel *Jordan*, found online in the Starpulse Supermodels image gallery [Starpulse 08]. The user-specified target pose

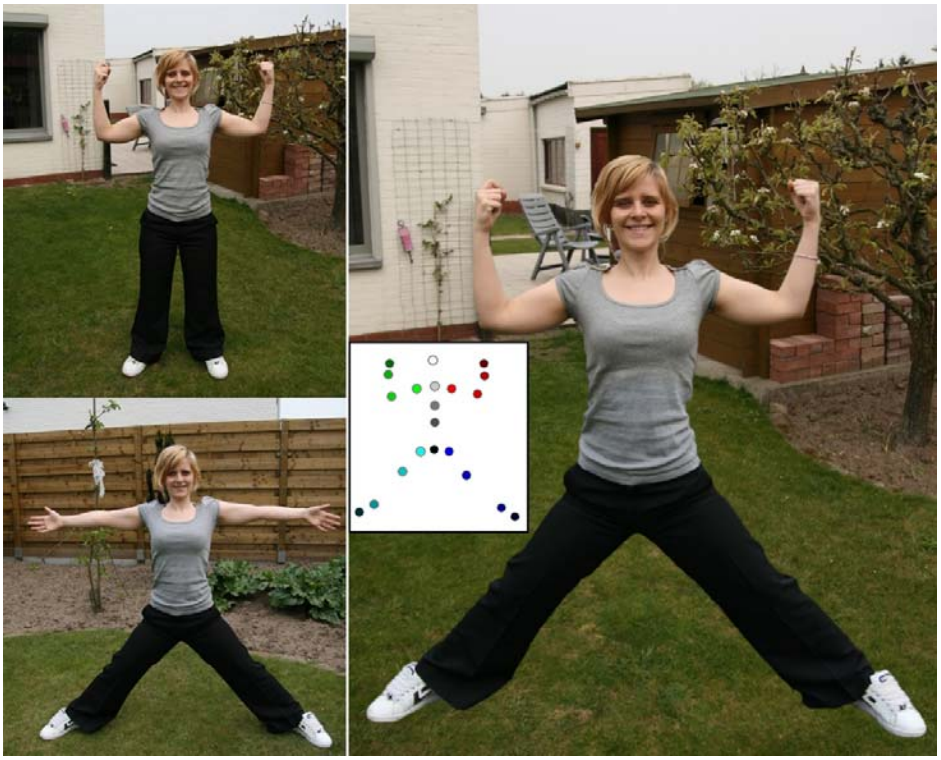


Figure 3.19: Synthesized image on the right hand side composed from two input images shown on the left. Notice that the input images are not taken from the same camera position.

cannot be composed out of the input poses, therefore the best available matches are automatically chosen and then deformed into the target pose. Since no background image was available for this scene, the input sprites were manually segmented, the background in the resulting image has been manually completed.

The example in Figure 3.21 consists of two input images. The first image shows a person sitting on a table, where the second image features this person standing up straight with arms spread. The result shows this person sitting on the table with his arms and legs open. To arrive at the desired target pose, skeleton and mesh deformation were performed. Notice the indicated areas, where a close-up of a fused overlap region is shown in the lower-right corner and compared with a close-up of the same

Figure 3.20: Synthesized image on the right hand side composed from two input images shown on the left. Note that the desired target pose can not be reached solely using the input poses. Mesh deformation is used to obtain the final result. Holes in the background were in-painted manually. Input images originate from the Starpulse Supermodels image gallery [Starpulse 08], shown at highest available image resolution.

region in the top-right corner by using averaging instead of the patch-based fusion approach. The ghosting artifacts in the averaged version are clearly not present in the fused version.

The last example uses a target skeleton in which both feet of the subject are lifted off the ground. Four input images are used, with one input image per leg, one for the head and one for the arms (see Figure 3.22). The final result is ob-

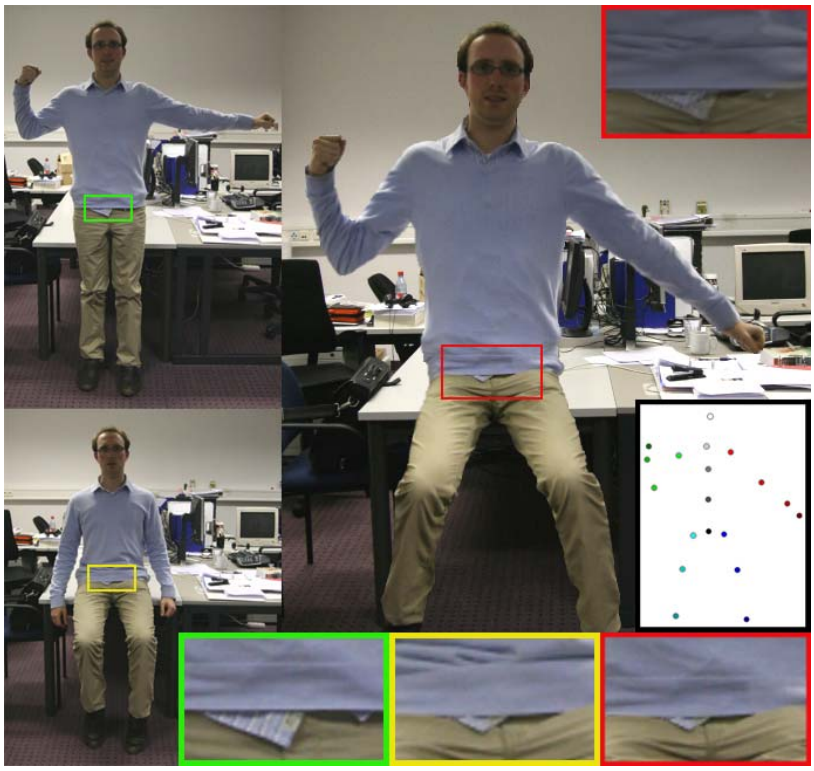


Figure 3.21: Pose synthesis example with input images on the left and result on the right. The inset red rectangle in the bottom right shows a close-up of our patch-based fusion approach, with simple averaging of the overlap input regions in the top right red rectangle. Close-ups of the same region in the input images are shown in the green and yellow rectangles. Notice how the averaged version exhibits ghosting artifacts on the shirt near the seam of the sweater. The final result was obtained using mesh deformation.

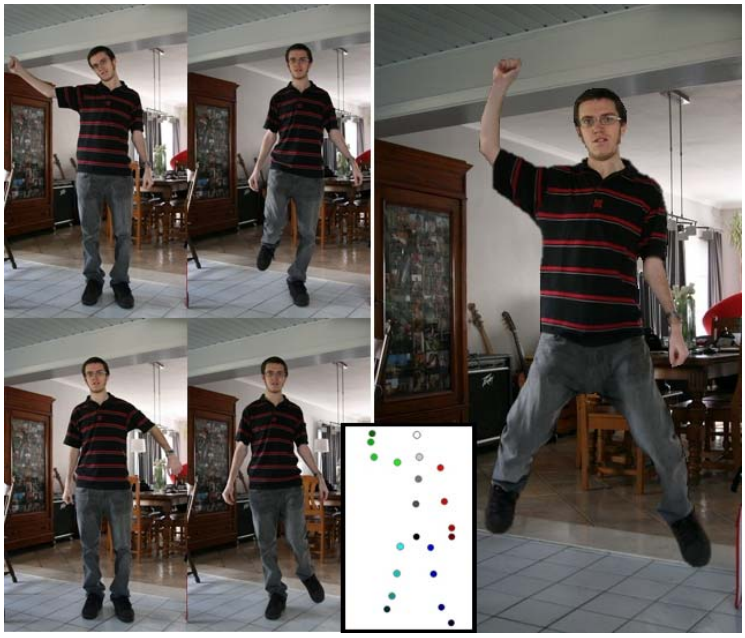


Figure 3.22: Synthesized image on the right hand side composed from four different input images shown on the left. The resulting image was deformed to match the target skeleton.

tained by using mesh deformation to better fit the legs and arms to the target skeleton.

For the results discussed in this section, the amount of time spent by the user was limited to indicating the skeleton joints on the input images and specifying a target skeleton. Marking the joints in the skeleton can be done with only a few mouse clicks, whereas a target skeleton is created starting from an input skeleton and then dragging and dropping the joints of that skeleton to a new position.

The precision of the positions of these joints is not highly important, as long as the used joint semantics are consistent throughout all input and target poses. When the joint positions are not placed consistently in all poses, for example if the hand is indicated once near the wrist and once at the end of the fingers, the mesh deformation algorithm might stretch the associated limb in an unnatural way.

Depending on the size of the overlap-areas in the fusing part of our solution, the unoptimised minimisation algorithm implemented in C++ required two to ten

minutes of calculation time. Optimising this code and running it on state-of-the-art hardware could decrease this bottleneck to less than a minute.

3.5 Conclusion and Future Work

In this chapter, techniques were presented to synthesize and animate articulated characters in an image/video-based manner. The key to our technique is a matching algorithm that focuses on high-level 2D skeleton models instead of the characterŠs visual appearance, granting an accurate frame matching and a high level of control over the animation.

When synthesizing an input video, a sequence of virtual 2D skeletons is used to define a desired target animation. The target skeletons are then matched with skeletons extracted from the source footage. The target skeletons only function as a guide for the new animation and do not force the input frames to match them exactly. This way the animation will achieve the desired effect without sacrificing the natural movement and appearance of the filmed subject.

Expanding the input pose space for the articulated video sprites algorithm, a novel technique was introduced to synthesize new poses from a set of input frames. This technique is based on selecting and merging different body parts into a desired pose. Only little user input is required to specify the poses (2D skeletons) of the input images and the target pose. For each body part in the target skeleton, best matches are computed in the input poses, and the associated image parts are transfered to the final image. A triangle mesh based distance function is used to identify which pixels belong to which body part. Overlapping regions in the resulting image are merged while respecting the continuity of the image.

Even though this method is able to generate a wide variety of poses from only a small set of images, a target pose can only be met approximately. More variety is obtained by incorporating mesh deformation.

3.5.1 Future Work

Potential improvements on this work can be suggested for different parts of the algorithm. Automatic skeleton extraction could reduce the required user interaction even more. User guided matching can also be an interesting feature that ensures

certain frames to be matched the way the user desires. Furthermore, instead of only using skeleton information to construct our matching cost function, we would like to look into the combination of both skeletal and pixel information to improve the matching between source and target frames.

Since this algorithm currently is unable to cope with situations where body parts occlude other ones, as well as with images where the subject is shot under large perspective differences, the availability of 3D skeletons and/or multi-camera information would be of great value when dealing with these problems. If this information is available, this technique would be highly suitable for use in 3D character animation applications [Starck 05, Starck 07].

Furthermore, an hierarchical skeleton model could be introduced to allow for an adaptable level of detail in the skeletons, for instance by switching to a detailed skeleton of a person's hand in a close-up.

Chapter 4

Augmented Panoramic Video

Contents

4.1 Introduction

Up until this part of the thesis, input videos have all been recorded with a static camera. We now extend this limitation and describe a video augmentation algorithm supporting videos featuring panning movements as well as zooming.

A first class of video augmentation applications for these kind of videos is video stabilization: the task of removing unintended and therefore unwanted shaky motion from a video. This is commonly achieved by computing and smoothing the motion path, either by making global adjustments commonly using a reference frame [Litvin 03] or by smoothing out local displacements [Matsushita 05]. Either way, the stabilized video will have gaps due to the warping of the original content. Instead of simply cropping the result, mosaicing [Litvin 03] or motion inpainting [Matsushita 05] (see Figure 4.1) can be applied to fill in the missing information.

Another class of applications deals with restructuring the image or video dimensions, while preserving a maximum amount of salient information. A recent example in the image domain, which allows for content-aware image resizing using seam carving, can be found in the work of Avadan et al. [Avidan 07] (see Figure

Figure 4.1: **Video stabilization.** Results from the algorithm of Matsushita et al. [Matsushita 05]. The top row shows the original input sequence, whereas the middle row shows the stabilized version, which shows missing parts of the image. Using motion inpainting, the missing areas are completed, which produces a sequence such as the bottom row.

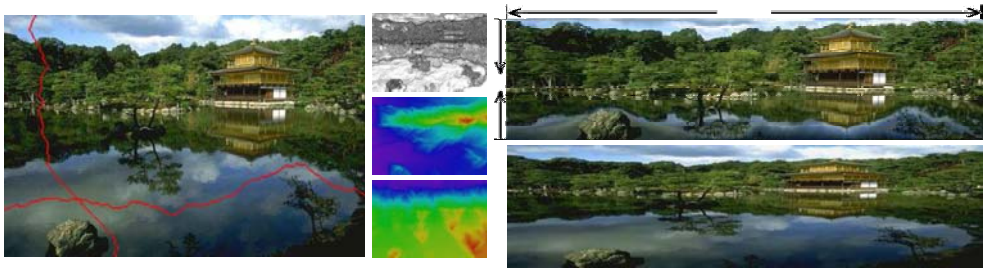


Figure 4.2: **Seam carving.** This algorithm by Avadan et al. [Avidan 07] poses an alternative to traditional cropping/stretching an existing image/video, preserving salient content in the process. Seam Carving operates on *seams*, i.e. sequences of orthogonally or diagonally adjacent pixels that run from one side of the image to the other. Removing all pixels in a seam reduces the height or width of the image by one row or column. In order to maximally preserve content, an importance function is defined to establish a cost for each pixel. Dynamic programming is used to establish the seam with the minimal cost.

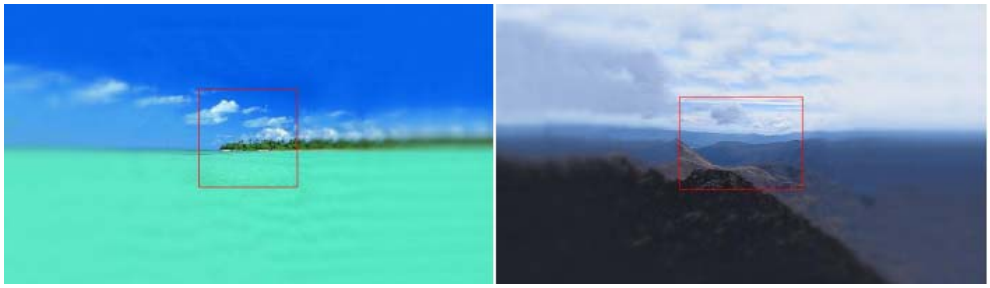


Figure 4.3: **Multiscale Ultrawide Foveated Video Extrapolation.** This algorithm by Aides et al. [Adies 11] introduces a multi-scale method which combines a coarse to fine approach to imitate the behavior of the human fovea. The box in the middle marks the original content for the current frame, the surroundings have been extrapolated. A proposed application for this is the Ambilight television system, where the wall around the television screen is illuminated in real-time with light that matches the colors at the margins of each frame.

4.2). The Multiscale Ultrawide Foveated Video Extrapolation work of Aides et al. [Adies 11] tries to extrapolate a video beyond its original field of view. They introduce a multi-scale method which combines a coarse to fine approach to imitate the behavior of the human fovea. A proposed application for this is the Ambilight television system, where the wall around the television screen is illuminated in real-time with light that matches the colors at the margins of each frame (see Figure 4.3).
A work more closely related to our own is the video retargeting algorithm by Liu et al. [Liu 06]. Illustrated in Figure 4.4, their paper aims at adapting a video sequence to fit a different display size than the one originally intended. As this introduces virtual pans and cuts, their approach is designed to minimize the loss of important information.

As mentioned, we will focus on a common type of video sequence, in which the videographer shoots a scene by rotating the camera to capture an entire panorama and possibly zooming into areas of particular interest. Typical video sequences furthermore even feature dynamic subjects, such as people or animals. We wish to re-display and manipulate such sequences in a meaningful way, presenting a technique that gives the user control over the camera's motion and field of view. As

Figure 4.4: **Video retargeting.** This algorithm, proposed by Liu et al. [Liu 06], adapts a video sequence to fit the spatial dimensions of a different display type than the medium it was originally recorded for. This introduces virtual pans and cuts, which must be optimized to minimize the loss of important information.

we are usually interested in increasing our field of view, this work can be seen as an inverse case of the video retargetting algorithm of Liu et al. [Liu 06].

Several assumptions are made. Most importantly, the video should be recorded from approximately a single location in the scene, i.e. the camera may only undergo a rotational motion. Significant translation would introduce severe parallax effects, which would require a more elaborate scene analysis with full 3D information. However, as it is practically impossible to avoid parallax, this is compensated for in our technique. Contributions to the field are twofold:

1. the idea of editing a panning/rotating video sequence using a full panoramic representation;

2. a robust video mosaicing algorithm that produces high quality panoramas without parallax artifacts, seams or blurring, while retaining repetitive dynamic elements.

4.2 Related Work

In this chapter we will cover several subdomains of computer graphics and computer vision which are related to the work presented in this part: video registration, texture

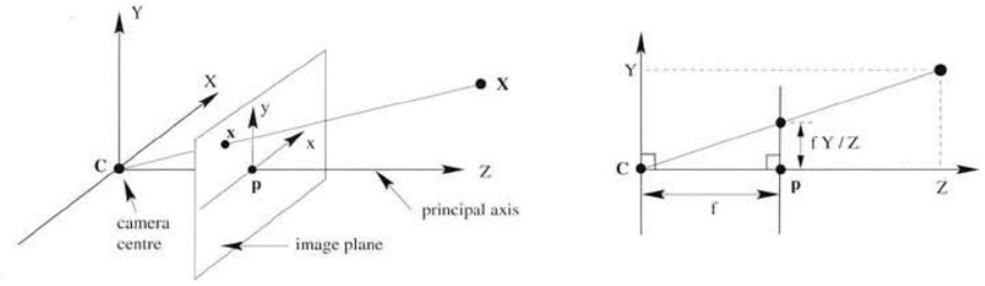


Figure 4.5: **Pinhole camera geometry.** **C** is the camera centre and **p** the principal point. The camera centre depicted in this image is placed at the origin of the coordinate system.

synthesis, and image & video completion. Before we take a look at these topics however, we first perform a quick review of the classical pinhole camera.

4.2.1 The Pinhole Camera Model

We consider the central projection of points in space onto a plane. Let us place the centre of projection at the origin of a Euclidian coordinate system, and consider the plane $(Z = f)$, which is called the *image plane* or *focal plane*. Under the pinhole camera model, using homogeneous coordinates, a point $\mathbf{X} = (X,Y,Z,1)^T$ in space is mapped to the point $\mathbf{x} = (fX, fY, Z)$ on the image plane where a line joining the point $\mathbf{X}$ to the centre of projection meets the image plane. This is illustrated in Figure 4.5.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.1}$$

The centre of projection is called the *camera centre*. The line from the camera centre perpendicular to the image plane is called the *principal axis* of the camera, and the point where the principal axis meets the image plane is called the *principal point*. The plane through the camera centre parallel to the image plane is called the *principal plane* of the camera.

4.2.1.1 The Camera Projection Matrix

For an arbitrary camera in a projective coordinate system, the pinhole camera model equation in Equation 4.1 generalizes to $\mathbf{x} = \mathrm{P}\mathbf{X}$, where P is the 3×4 homogeneous *camera projection matrix*. This projection matrix can be decomposed in two independent components, which are responsible for two seperate steps of the imaging process.

$$\mathrm{P} = \mathrm{K}[\mathrm{R}|\mathbf{t}] = \mathrm{K}\mathrm{R}[\mathrm{I}|-\mathbf{C}] \tag{4.2}$$

4.2.1.2 Extrinsic Parameters

The matrix $[\mathrm{R}|\mathbf{t}] = \mathrm{R}[\mathrm{I}|-\mathbf{C}]$ contains the camera's *external* or *extrinsic* camera parameters, encoding the position of camera centre $\mathbf{C}$ and the camera's orientation R. Using this information, we can compute the coordinates of a space point $\mathbf{X}$ within the camera's own coordinate frame.

$$\mathbf{X}_{cam} = \begin{bmatrix} \mathrm{R} & -\mathrm{R}\mathbf{C} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R} & -\mathrm{R}\mathbf{C} \\ 0 & 1 \end{bmatrix} \mathbf{X} \tag{4.3}$$

The problem of estimating the parameters of the matrix R and the vector $\mathbf{t}$ is commonly referred to as *extrinsic camera calibration*. This occurs when the intrinsic parameters of the camera (which are discussed in the next section) have already been acquired by another method.

4.2.1.3 Intrinsic Parameters

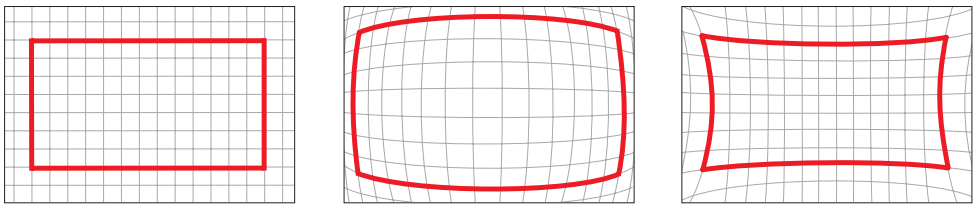
Once the position and orientation of the camera and the coordinates of the point $\mathbf{X}_{cam}$ are known, the space point can be projected onto the image plane.

$$\mathbf{x} = \mathrm{K}\mathbf{X}_{cam} \tag{4.4}$$

The matrix K is called the *camera calibration matrix* and has the following form:

$$\mathrm{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

where



(a) no distortion (b) barel distortion (c) pincushion distortion

Figure 4.6: Imaging a red rectangle under the effect of **radial lens distortions**.

- α_x is the scale factor in the x-coordinate direction,
- α_y is the scale factor in the y-coordinate direction,
- s is the skew,
- (x_0, y_0) are the coordinates of the principal point.

The parameters of the camera calibration matrix are also referred to as the camera's *internal* or *intrinsic* camera parameters.

4.2.1.4 The Non-linear Distortion Model

The pinhole camera model is usually insufficient to fully describe the imaging process of real cameras, which show deviations from this linear model. The scene point **X**, the distorted image point $\mathbf{x}_d$ and the camera centre **C** are no longer collinear, and straight lines are no longer imaged as straight lines. The most common such deviation is a radial distortion (illustrated in Figure 4.6), which can be modeled by a polynomial distortion model.

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = L(\tilde{r}) \begin{bmatrix} x_u - x_c \\ y_u - y_c \end{bmatrix} \tag{4.6}$$

where:

- (x_d, y_d) is the pixel position, after radial distortion
- (x_u, y_u) is the undistorted pixel position
- (x_c, y_c) is the centre of the radial distortion
- $\tilde{r}$ is the radial distance $\sqrt{(x_u - x_c)^2 + (y_u - y_c)^2}$ from the centre for radial distortion

- $L(r)$ is the distortion function, approximated by a Taylor expansion: $L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + ...$

For other distortion models, we refer to the work of Devernay and Faugeras [Devernay 01]. Furthermore, for the remainder of this chapter, we will assume that we are dealing with sets of undistorted projective image points $\{\mathbf{x}_u^i\}$.

4.2.2 Video Registration

The task of properly aligning partially overlapping images captured by a camera is commonly referred to as video registration. Assuming we restrict ourselves to the class of *physically plausible* registration methods, in which the computed transformations correspond to a plausible camera displacement/rotation (this excludes non-linear grid deformations), this task corresponds to the calibration of a set of cameras with colocated camera centres.

4.2.2.1 Introducing the Homography

In the case of physically plausible video registration, we are thus computing the (linear) relation between sets of image points $\{\mathbf{x}\}$ and $\{\mathbf{x'}\}$ of the corresponding set of scene points $\{\mathbf{X}\}$ for different frames in the video,

$$\begin{aligned} \mathbf{x} &= \mathrm{K}\left[\ \mathrm{I}\ \middle|\ \mathbf{0}\ \right]\mathbf{X} \\ \mathbf{x'} &= \mathrm{K'}\left[\ \mathrm{R}\ \middle|\ \mathbf{0}\ \right]\mathbf{X} = \mathrm{K'RK}^{-1}\mathrm{K}\left[\ \mathrm{I}\ \middle|\ \mathbf{0}\ \right]\mathbf{X} = \mathrm{K'RK}^{-1}\mathbf{x} \end{aligned} \tag{4.7}$$

so that $\mathbf{x'} = \mathrm{H}\mathbf{x}$ with $\mathrm{H} = \mathrm{K'RK}^{-1}$. This constitutes the homography H, which uniquely defines the linear relationship between point sets $\{\mathbf{x}\}$ and $\{\mathbf{x'}\}$. Using a sufficient amount of image correspondences, a homography H can be computed for each set of subsequential video frames.

4.2.2.2 Panning Sequences

The most common video sequences shot from a single camera centre are one-dimensional panning shots. In such a simple scenario, the intrinsic camera matrix K remains the same throughout the entire sequence, and R takes the very simple form of a rotion around the Y-axis. As such, this can be translated into a very simple minimization problem [Hartley 04].

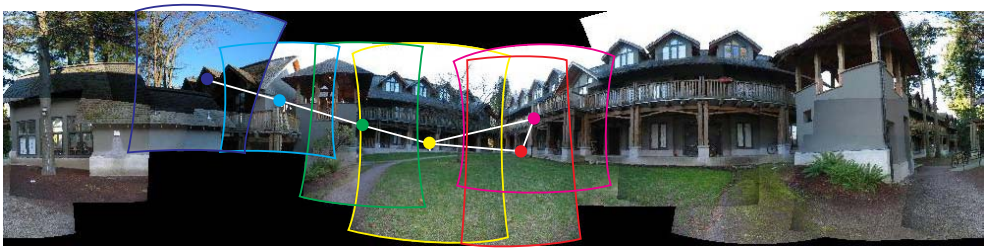


Figure 4.7: A set of input frames warped to a common reference frame, taken from the work of Brown et al. [Brown 03]. Superimposed, we have displayed a subset of the frame topology. The topology is depicted by a graph, where the nodes correspond to warped images, and edges exist between nodes corresponding to warped images with sufficient overlap.

4.2.3 Arbitrary Rotations and Zoom

However, in case the camera follows a motion pattern which is more sophisticated than this common 1D panning sequence, all available information needs to be exploited in order to assure a good overall registration. This information can usually be seen as an approximation of the frame topology (illustrated in Figure 4.7). Generally some form of global optimization is applied to ensure an overall consistent registration.

In the last decade, many approaches to global registration have been proposed. We will restrict ourselves to those most closely related to our own work, more precisely those that let topological information guide the registration process. A graph representation is commonly used to depict the topology, casting the problem as the identification of the shortest path [Kang 00, Marzotto 04, Sawhney 98].

We opted for an alternative graph-based approach: instead of weighing the edges with some confidence measure of choice, our algorithm is designed to minimize the number of intermediary nodes between each frame and the reference frame. This is based on the notion that we do not necessarily need to know *how good* every single edge in the graph is, only that they are *good enough*. Furthermore, the proposed approach is aimed at reducing computation time, minimizing the number of homography computations.

4.2.4 Image/Video Completion & Texture Synthesis

Image completion poses the problem of filling in missing pixels in large unknown regions of an image in a visually consistent way. This is very similar to the objective of texture synthesis, in which a large area of texture information needs to be generated, based on the limited intensity information available in a smaller sample.

Historically, exemplar-based techniques have proven to be the most successful in dealing with this problem, copying pixels or source patches from the observed part of the image [Efros 99, Criminisi 03, Drori 03, Kwatra 05]. The common drawback to these approaches is their greedy approach to filling the image, which can often lead to visual inconsistencies. Initial attempts to avoid this problem have taken a more global approach, using Expectation Maximization (EM)-like schemes for optimizing the process [Kwatra 03, Wexler 04]. However, EM is known to be particularly sensitive to initialization and can get trapped in poor local minima.

Other recent approaches have applied dynamic programming or belief propagation [Yedidia 01] to reach a more globally consistent image. Most of these algorithms guide the completion process by influencing the order by which the synthesis proceeds. This can either be done manually by user assistance, e.g. Jian Sun et al. [Sun 05] give priority to user-specified curves on which the most salient missing structures reside, or it can be deduced by the algorithm itself [Komodakis 06].

Recently some authors have extended the application range of their image completion and texture synthesis algorithms to the video domain. In one related work, Agarwala et al. [Agarwala 05] constructed ‘panoramic video textures’ (see Figure 4.8). Starting from a video segment filmed by panning a camera across a dynamic scene, they combine looping segments of a constant duration in order to construct a single panoramic video texture. Even though this work naturally relates to dynamic panoramic backgrounds, there are several issues that prevent us from applying this technique to our situation. Our augmented video has a predetermined finite duration, and contains pixel intensities that should remain unchanged to preserve the original content. We cannot discard pixels from the input sequence to create a better fit for the required constant looping time. Finally, while we would like to use arbitrary input videos, the method of Agarwala et al. is restricted to horizontal

Figure 4.8: A frame generated by the **panoramic video textures** algorithm of Agarwala et al. [Agarwala 05].

panning sequences.

Finally, a new representation for video editing has been presented as Unwrap Mosaics [Rav-Acha 08]. The representation introduced Rav-Acha et al. allows extracting a 3D surface model directly from uncalibrated video footage. The primary goal however is to recover the object's texture map rather than its 3D shape. The recovered texture map is accompanied by a 2D-to-2D mapping describing the texture map's projection to the input images and a sequence of binary occlusion masks. From this information a 3D shape can be recovered and simple image editing tasks can be performed directly on the "unwrap mosaic", as illustrated in Figure 4.9.

4.2.5 Background Estimation

The problem of estimating a consistent background is commonly addressed by applying a temporal mean or median filter to the video at pixel level. However, in case of stationary occlusors that persist for more than half the sequence length, or when dealing with the presence of parallax effects, these simple approaches fail. Spatial support is required as an additional cue to improve pixel-level algorithms.

The work most closely related to our own is that of Colombari et al. [Colombari 06] (see Figure 4.10). They present a region growing algorithm, which starts from patches that are always visible in the scene, gradually forming a

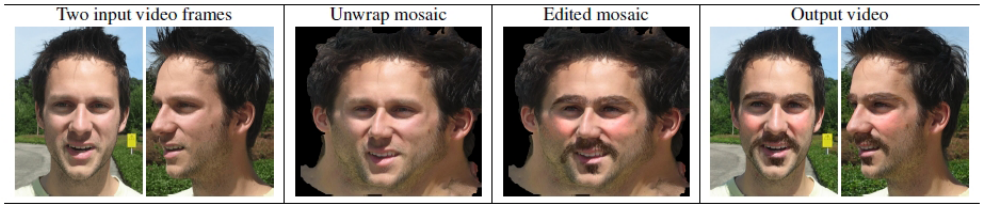


Figure 4.9: Overview of the **Unwrap Mosaics** representation of Rav-Acha et al. [Rav-Acha 08]. Starting from input video frames, a texture map is recovered together with a 2D-to-2D mapping and occlusion masks. The entire video can then be edited by simply editing the "unwrap mosaic".

Figure 4.10: Results of the **background estimation algorithm** of Colombari et al. [Colombari 06]. The first three colums are sample frames from the video, whereas the next two columns respectively show the median image and their computed background.

consistent background. This approach shows similarities to the early examplar-based image completion algorithms, and potentialy inherits their common drawback: its greedy approach can lead to visual inconsistencies when two regions come together. While we use similar cues to guide our background synthesis, our algorithm poses background estimation as an optimization problem with a well defined energy function. Our formulation also allows for sharper patches to be chosen over their blurred counterparts, reducing (if not completely removing) parallax effects if the intensity information is available in the original video.

4.3 Overview

Our video augmentation pipeline consists of five different processing steps.

1. In the first of these steps, we properly warp and align all input images in a common reference frame. This corresponds with the calibration step of our system (Section 4.3.2).

2. Based on the warped video, we proceed by computing a globally consistent static background, containing sharp details free of parallax artifacts (Section 4.3.3).

3. This background is then used to classify the dynamic foreground elements (actors) and the static or dynamic background elements (repetitive/quasi-repetitive motions, or complex stochastic phenomena with an overall stationary structure) (Section 4.3.4).

4. After the background elements are identified, the warped input video is extended with a dynamic background panorama (Section 4.3.5).

5. Finally, we warp the dynamic background panorama back to the original video, modified according to the user-controlled virtual camera (Section 4.3.6).

4.3.1 Notation

Two images of the same scene are related by a non-singular linear transformation of the projective plane in two cases: (a) if the scene is planar or (b) if the center of projection does not change, i.e. the only degrees of freedom are due to the orientation of the camera. In these cases we do not suffer from the effects of parallax, and the images can be composed together to form a mosaic.

Image points are represented by their homogeneous coordinates $\widetilde{\mathbf{x}} = (x, y, w)$, with $\mathbf{x} = (\frac{x}{w}, \frac{y}{w})$ being the corresponding Cartesian coordinates. A linear transformation of the projective plane, called a homography, is represented by a 3×3 matrix H when $\widetilde{\mathbf{x}}_j = \mathrm{H}_{i,j}\widetilde{\mathbf{x}}_i$, where $\widetilde{\mathbf{x}}_i$ and $\widetilde{\mathbf{x}}_j$ are corresponding points in frames i and j respectively.

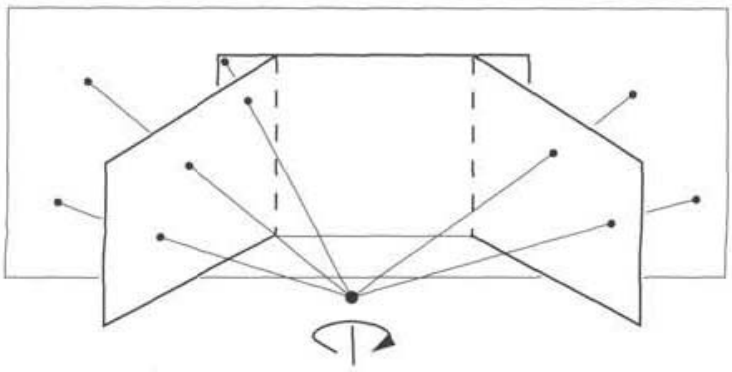


Figure 4.11: **Video registration.** The first task consists of aligning all the images to a common reference frame. In this illustration, three images acquired by a rotating camera are registered to the frame of the middle one, by warping the outer images to align with the middle one.

4.3.2 Video Registration

Our video registration pipeline is essentially a two-step process, with an optional bundle adjustement step. During the initial estimation step (as illustrated in Figure 4.11), we have no knowledge of the frame topology (the relative spatial positioning of the frames), so we rely on temporal information only. Using the results of this initial estimation, we subsequently take a graph-based approach, using the newly acquired spatial information. Finally, we can employ an optional bundle adjustment step.

4.3.2.1 Homography Computation

Feature detection and matching is done by employing the Kanade-Lucas-Tomasi tracker [Lucas 81a, Tomasi 91]. After proper normalization of the found correspondences [Hartley 97], we employ a RANSAC-based [Fischler 81] algorithm to compute homographies, using minimal sample sizes [Brown 07a]. When the RANSAC procedure has computed an initial homography and a matching set of initial inliers, we employ the method proposed by Kanatani and Ohta [Kanatani 99]. This method is based on a statistical renormalization technique, and determines a statistically optimal homography. This non-linear estimation is repeated until a stable amount of inliers is achieved.

4.3.2.2 Topology-independent Alignment

As a common first step in many graph-based registration algorithms, the inter-frame homographies between all neighboring frames are calculated. In order to establish an initial guess of the frame topology, we could recursively concatenate the homographies from each frame to a chosen reference frame r.

$$\begin{cases} \mathrm{H}_{r,r} = \mathrm{I} \\ \mathrm{H}_{i,r} = \mathrm{H}_{i+1,r}\mathrm{H}_{i,i+1} & \texttt{if } i < r \\ \mathrm{H}_{i,r} = \mathrm{H}_{i-1,r}\mathrm{H}_{i,i-1} & \texttt{if } i > r \end{cases} \tag{4.8}$$

These homographies are commonly computed from one frame of the input sequence to the next. However, for the purpose of image mosaicing, our experiments have indicated that computing these homographies on pre-warped images (using the target frame's homography as the warping function) results in more accurate estimates. The reasoning behind this is that the pixel-error is measured in the coordinate space of the final panorama directly.

Due to the recursive nature of the computation process, estimation errors will propagate down the homography chain. A coarse misalignment would immediately break the chain. Therefore, we propose a slightly different scheme, in which we use a sliding window of potential homography candidates instead of simply linking consecutive frames (see Figure 4.12). As mentioned before, computing homographies requires the target frame's warping function to be known. As such, we keep track of the frames that are already linked to the reference frame, and label them as committed.

Starting from the reference frame, we try to connect adjacent uncommitted frames to committed frames. Initially, only the reference frame itself is labeled as committed, as it is the only one whose warping function to the reference mosaic is already known. For each uncommited frame i, we attempt to compute the homography $\mathrm{H}_{i,j}$ to each commited node $j \in [i-d, i+d]$, starting with the closest neighbors. As soon as we find a homography with a confidence value above a predefined threshold, it is stored and the source frame is labeled as committed. We repeat this procedure until no more uncommitted frames remain.

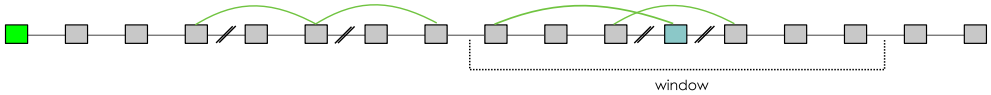


Figure 4.12: **Topology-independent Alignment.** A sliding window of potential homography candidates is checked instead of only linking consecutive frames. In this illustration, the boxes represent individual frames, whereas the edges correspond to computed homographies with sufficient inlier support. Frames connected by black edges are direct neighbors, whereas green edges correspond to short-cuts through the graph. The window is centered around the frame highlighted in blue.

There are two possible reasons why no more frames are committed: (a) either all the frames have a parent frame (a frame through which they are linked to the reference frame) and an associated homography to this frame, or (b) for each of the uncommitted frames i, no potential candidate j within the given window size has been found. In the latter case, we use the previously stored confidence values to find the homography with the highest confidence, add the source frame to the list of committed frames, and resume the previous procedure.

This produces a homography tree with constraints on the confidence values associated between the different nodes. Unfortunately, considering the depth of the tree, the propagated estimation errors will still result in a considerable misalignement at the end of the sequence. However, we now have a first estimate of the frame topology in the reference mosaic, which we can use to provide us with a more accurate registration algorithm.

4.3.2.3 Topology-dependent Alignment

In this stage the homography tree from the previous stage will be transformed into a new instance, taking into account the estimated topology information.

As stated before, our algorithm is designed to minimize the number of intermediary nodes between each frame and the reference frame, based on the notion that we do not necessarily need to know how good every single edge in the graph is, only that that they are good enough (see Figure 4.13). If we can guarantee a minimum level of confidence, the results will be usable for future computations. During our experiments, we have used the number of correspondences within predefined error

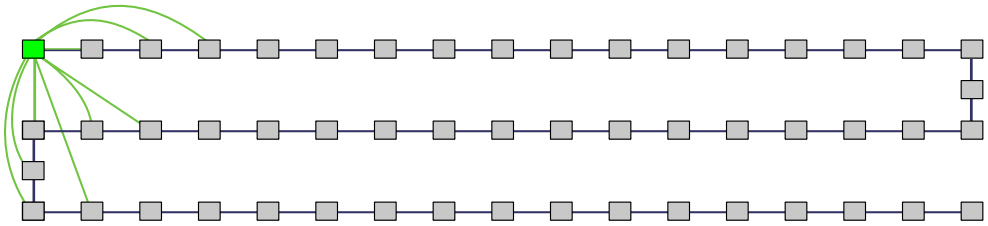


Figure 4.13: **Topology-dependent Alignment, Initiation Step.** Using topoligy information, all potential candidates for direct linking to the reference frame are computed. Those with sufficient support are added to the graph.

bounds as our confidence metric of choice. We provide an outline of our algorithm in Table 4.1

---

1. $U := \{i \mid i \neq r\}$; % unconnected frames ($\neq$ reference frame)
2. $C := \{r\}$; % previously connected frames
3. $N := \emptyset$; % newly connected frames
4. `while` $U \neq \emptyset$
   (a) $\forall i \in U$:
      i. Sort $j \in C$, according to $\|i - j\|$, closest first;
      ii. Find first $j \in C$, where $\#inliers(\mathrm{H}_{i,j}) \geq threshold$
      iii. If j found: $N = N \cup \{i\} \ \wedge \ U = U - \{i\}$
   (b) $\begin{cases} \texttt{if}\ N \neq \emptyset : C = N \\ \texttt{else} : C = \{k \mid \arg\max(\#inliers(\mathrm{H}_{k,j})), k \in U, j \notin U\} \end{cases}$
5. `end while`

Table 4.1: Topology-dependent Alignment

Note the similarities with the previous stage: we utilize a confidence threshold to decide if we add the child node i of the homography $\mathrm{H}_{i,j}$ to the set of connected (committed) frames. Also, once a node is connected to the rest of the graph, its warping function is known. This way, we can always use pre-warped images to perform the homography estimation.

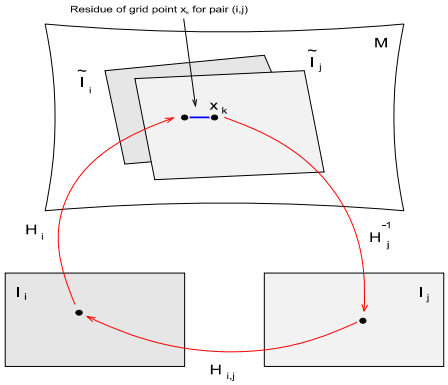


Figure 4.14: **Bundle adjustment.** We apply the bundle adjustment method of Marzotto et al. [Marzotto 04] to our graph, minimizing the residual error of a set of grid points over the parameter space of the homographies within the graph.

In principle, when iterating through the set of unconnected frames U, given the set of previously connected frames C, we could compute the homography between each pair $(i, j) \in U \times C$ to look for new additions to the graph. Homography computation however is an expensive operation, and should be avoided if a lack of inlier support is expected beforehand. Therefore, we will only consider frames with a significant degree of overlap as potential candidates for addition.

We use the available topology information to reduce the search space of potential edge candidates: in order to establish the degree of overlap, homographies $\mathrm{H}_{i,r}$ from the topology-independent alignment step are used as an approximation to the true registration matrices. As an overlap measure, we use the normalized distance between centroids:

$$\delta_{ij} = \frac{max(0, |\mathbf{c}_i - \mathbf{c}_j| - |d_i - d_j|/2)}{min(d_i, d_j)} \tag{4.9}$$

where $\mathbf{c}_i$, $\mathbf{c}_j$, d_i and d_j are the centroids and the diameter of the projection onto the mosaic of frames i and j, respectively.

4.3.2.4 Bundle Adjustement

As a final step we apply the bundle adjustment step proposed by Marzotto et al. [Marzotto 04], which finds the solution $\{\mathrm{H}_i\}$ that minimizes the total misalignment of a predefined set of m grid points on the mosaic (see Figure 4.14). Let x_k be a

grid-point and let E_k be the set of edges $(i, j) \in E$ so that x_k belongs to the overlap region between frame i and frame j. The error at the grid-point $\mathbf{x}_k$ is defined as:

$$E_k = \frac{1}{|E_k|} \sum_{(i,j)\in E_k} \left\| \mathbf{x}_k - \pi \left(\mathrm{H}_{i,r} \mathrm{H}_{i,j} \mathrm{H}_{j,r}^{-1} \tilde{\mathbf{x}}_k \right) \right\|^2 \tag{4.10}$$

where π transforms homogeneous coordinates into Cartesian (pixel) coordinates.

Since we want to minimize the error at all grid points simultaneously, we end up with a system of non-linear equations that can be cast as a least-squares problem.

$$\min_{\{\mathrm{H}_{i,r}\}} \sum_{k=1}^{m} E_k^2 \tag{4.11}$$

The Levenberg-Marquardt algorithm is used to solve Equation 4.11, using the previous set of $\{\mathrm{H}_{i,r}\}$ as the starting solution. Data standardization is carried out to improve the conditioning of the problem [Hartley 97].

4.3.3 Static Background Estimation

After we have properly registered all frames, the next step in our pipeline consists of computing a consistent static background. In an ideal situation, registration would be perfect and every background pixel would be visible for a sufficient period of time. Unfortunately, real-world footage is rarely perfect, and as a result we will have to deal with the effects of motion blur and parallax. For good measure, we will also be dealing with the possibility of actors that stay in a single place for a significant period of time, only exposing the true background for a few seconds.

As stated before, our background estimation algorithm shows some similarities to the region growing algorithm of Colombari et al. [Colombari 06]. However, unlike the greedy approach taken in their work, we have opted to pose the background estimation as a discrete global optimization problem.

4.3.3.1 Problem Statement

Given a set of warped input images I_i and their binary masks B_i (as illustrated in Figure 4.15 on page 87), the goal of our algorithm is to compute a visually plausible background by merging spatially consistent, but temporally varying patches into a consistent background image. To this end, we propose the use of a discrete Markov

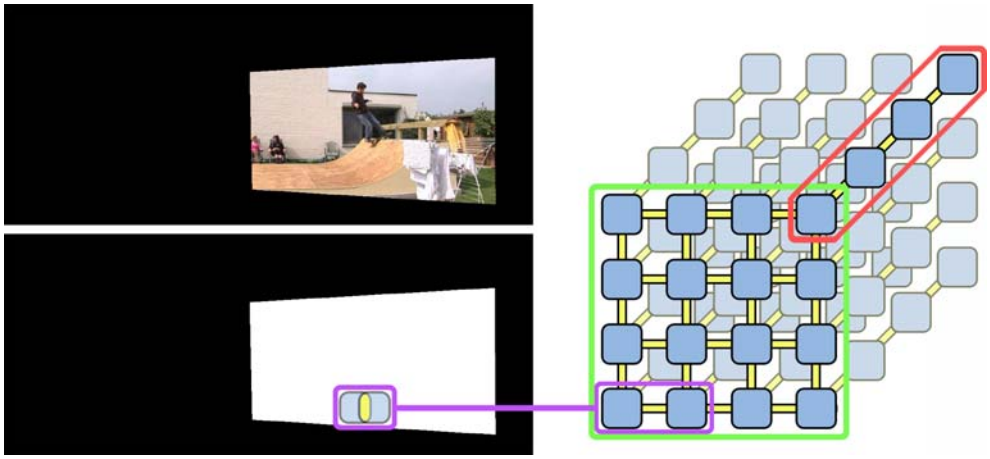


Figure 4.15: **Warped input frame + binary mask; Background estimation MRF.** *(left)* An example of a warped input frame, and the associated binary mask. *(right)* Background Estimation MRF: a 2D grid of nodes (green), with label patches in the temporal dimension (red). Every pair of connected nodes represents overlapping image patches (purple).

Random Field (MRF).

The nodes N of the MRF are defined by placing an image lattice over the total space occupied by the mosaic, with a horizontal and vertical spacing of $step_x$ and $step_y$ respectively. Each node is uniquely defined by their (x,y) coordinate on the mosaic, and the edges E of the MRF are defined by looking at the 4-neighborhood of each individual node. The total label set L consists of all possible $w \times h$ patches around every node $n_i \in N$. Thus, the labels $l \in L$ are uniquely defined by the spatial coordinates (x,y) of their center pixel, and their frame number $t \in [1,N]$. Note that $step_x$ and $step_y$ are set so that a region of overlap between neighboring patches of size $w \times h$ always exists.

Every node $n_i(x_i,y_i) \in N$ has a maximum of N possible label candidates $l(x_l,y_l,t)$ where $(x_l,y_l) = (x_i,y_i)$. Also, a label (x,y,t) will only be considered a valid background candidate if the full patch window $W = [x-\frac{w}{2},x+\frac{w}{2}] \times [y-\frac{h}{2},y+\frac{h}{2}]$ is marked in the binary mask B_t (see Figure 4.15), or if it is only partially marked but

located on the border of the mosaic.

The single node potential $V_i(l)$ for placing label l over node n_i will describe the likelihood of patch l being part of the background. This likelihood can be expressed in terms of the number of frames in which the patch is visible during the entire sequence. However, a single patch has no inherent information about the duration of its visibility, so we are required to perform an a priori clustering step. As in the work of Colombari et al. [Colombari 06], we apply single linkage agglomerative clustering [Jain 99] to group our labels l_t into clusters $C_t \subset L_i$. Every cluster C_t will be defined by choosing one of its labels l_t as the primary label, and each node will be assigned a set of clusters instead of a set of individual labels. Based on the size of these clusters, we can now define our single node potentials as:

$$V_i(C) = \alpha \left[1 - \left(\frac{|C|}{N} \right)^2 \right] \tag{4.12}$$

Lastly, the pairwise potential $V_{ij}(C, C')$ will measure how well these clusters agree on their region of overlap. We will define the pairwise potential by the sum of squared differences (SSD) of the mean labels from the respective clusters in this area of overlap A, divided by the amount of overlap pixels $|A|$:

$$V_{ij}(C, C') = \beta \left[\frac{1}{|A|} \sum_{(x,y) \in A} \left(\bar{I}(x,y) - \bar{I}'(x,y) \right)^2 \right] \tag{4.13}$$

Based on this formulation, where α and β are user-specified weights, our goal will now be to assign a cluster $\hat{C}_i \subset L$ to each node n_i, so that the total energy cost $E(\{\hat{C}_i\})$ of the MRF is minimized, where:

$$E(\{\hat{C}_i\}) = \sum_{i=1}^{|N|} V_i(\hat{C}_i) + \sum_{(i,j) \in E} V_{ij}(\hat{C}_i, \hat{C}_j) \tag{4.14}$$

4.3.3.2 Energy Minimization by Belief Propagation

As an advantage of formulating background estimation as an energy minimization problem, we can now apply belief propagation to our energy function.

Belief propagation (BP) is an iterative inference algorithm that works by propagating local messages along the nodes of an MRF [Yedidia 01]. Messages sent from

node n_i to n_j form a set $\{m_{ij}(l)\}_{l\in L}$, where element $m_{ij}(l)$ indicates how likely node n_i thinks that node n_j should be assigned label l. Furthermore, messages are updated (i.e. sent) until convergence as follows:

$$m_{ij}(l) = \min_{l_i \in L}\{V_i(l_i) + V_{ij}(l_i, l_j) + \sum_{k:k\neq j,(k,i)\in E} m_{ki}(l_i)\} \tag{4.15}$$

This update rule is associated with the min-sum version of BP, where the potentials are described in the *-log* domain. After convergence, a set of beliefs $\{b_i(l)\}_{l\in L}$ is computed for each node, where belief $b_i(l)$ is defined as follows:

$$b_i(l) = -V_i(l) - \sum_{k:(k,i)\in E} m_{ki}(l) \tag{4.16}$$

These beliefs approximate the max-marginal of the posterior at node n_i, and thus describes the likelihood that the label l should be assigned to that node. Based on this fact, a node is then assigned the label with the maximum belief, i.e. $\hat{l}_i = \arg\max_{l\in L} b_i(l)$. It is known that, for tree structured graphs, BP will always converge to the optimal solution, while for graphs with loops, it can only guarantee to find a local optimum.

4.3.3.3 Dual-step Energy Minimization

In order to reduce the computational time of our algorithm, we have opted to perform our background estimation in two seperate steps. During the initial step, we will try to assign a cluster $\hat{C}_i \subset L$ to each node n_i, minimizing the total energy cost $E(\{\hat{C}_i\})$ of the MRF (Equation 4.14). Here, the clusters take the role of labels in the BP algorithm.

In a subsequential step, we will unpack these clusters and assign a label $\hat{l}_i \in \hat{C}_i$ to each node n_i, minimizing another energy cost $E(\{\hat{l}_i\})$ associated with individual labels rather than clusters:

$$E(\{\hat{l}_i\}) = \sum_{i=1}^{|N|} V_i(\hat{l}_i) + \sum_{(i,j)\in E} V_{ij}(\hat{l}_i, \hat{l}_j) \tag{4.17}$$

The single node potential function $V_i(l)$ of label l estimates the level of blur of the corresponding window W:

$$V_i(l) = -\frac{1}{|W|}\sum_{(x,y)\in W}\left(I(x,y) - \bar{I}(W)\right)^2 \tag{4.18}$$

where $\bar{I}(W)$ symbolizes the mean of window W. By defining this single node potential, we encourage the use of the sharper labels in each cluster over their more blurry counterparts. The pairwise potential $V_{ij}(l, l')$ computes the SSD of the labels in their respective area of overlap A, divided by its size $|A|$.

Choosing this two-step approach over a single minimization step decreases computation times due to the reduced number of labels in each step. In addition, it will also increase the robustness of the algorithm, as false positives are most likely to have been removed from the label set after the clustering step.

4.3.3.4 Belief Propagation Optimizations

For sequences of limited length and resolution, standard BP will be able to cope with the size of the MRF. However, for problems with a large set of potential labels per node, additional optimizations are required. One approach, specifically designed for this purpose, is the application of priority-based message scheduling and the associated label pruning, as described in the work of Komodakis et al. [Komodakis 06]. For the exact details, we refer to the paper.

We advise using a very conservative pruning threshold for the initial cluster minimization step, and a more aggressive threshold for the following label minimization step. The reasoning behind this is the following: if we wish to consider even the smallest background exposures, we want to avoid aggressive first pass label pruning. During the first pass, the pruning relies heavily on the single-node potentials to compute the relative beliefs, and as a result the smallest clusters will be pruned away. Also, our initial clustering already significantly reduces the amount of labels, so our need for pruning is somewhat alleviated.

During the second (label minimization) step, we can use a more aggressive pruning approach: first pass pruning can filter out many labels that are considered much more blurry than the other labels in the cluster. It should be noted that the proposed pruning scheme makes use of the label's relative beliefs, so a cluster that contains only blurry or textureless patches will not be pruned away in its entirety.

4.3.4 Foreground Segmentation

The purpose of the foreground segmentation component is to identify the actors (dynamic foreground elements) in our scene. For every warped input frame we need to decide which pixels belong to the (static or dynamic) background and which pixels belong to the foreground.

To do this, we use a classifier that is based on the X84 outlier rejection rule [Hampel 86]. Every pixel of each warped frame is compared with its associated pixel in the static background image calculated in section 4.3.3. To make a robust classification based on the difference between these pixel values, we incorporate the median of absolute deviations (MAD) into the computations (Figure 4.16(a)). The MAD is a statistical measure that is commonly used to describe the variability of data with outliers.

$$MAD(x,y) = med_i\{|I_i(x,y) - bg(x,y)|\} \tag{4.19}$$

An input pixel (x,y) belongs to a foreground element if

$$\frac{|I_i(x,y) - bg(x,y)|}{MAD(x,y)^2} > \chi_3^{-1}(\alpha) \tag{4.20}$$

where $\chi_3^{-1}(\alpha)$ is the inverse-chi-square distribution with 3 degrees of freedom and a confidence value of α. The resulting segmentation images S_i are cleaned up by using standard morphological filtering operations. A resulting foreground/background classification is shown in Figure 4.16.

4.3.5 Dynamic Background Estimation

Our process of creating dynamic background content is structured in a fashion similar to our background generation component.

4.3.5.1 Single-step Energy Minimization

Given a set of warped input images I_i, their binary masks B_i and a foreground/background segmentation S_i, the goal of this component is to compute a visually plausible panoramic video by merging spatially consistent, but temporally varying patches into a consistent video panorama. To this end, we propose the use of an additional Markov random field, which expands our static background estimation Markov

Figure 4.16: **Classification of foreground/background elements.** We use the median of absolute deviations to: (a) segment out foreground elements, and (b) classify background regions as static or dynamic (see labels)).

random field into the temporal dimension.

The nodes N of this new field are once again defined by placing an image lattice over the total space occupied by the mosaic, with the addition of a temporal dimension t. Each node is thus uniquely defined by a set of coordinates (x,y,t). The edge set E is acquired by connecting all available elements within the 6-neighborhood of each individual node. The label set L is a subset of the one we used for static background estimation. We remove the labels containing the foreground object (encoded in the segmentation images S), which leaves us with both the static and dynamic background labels.

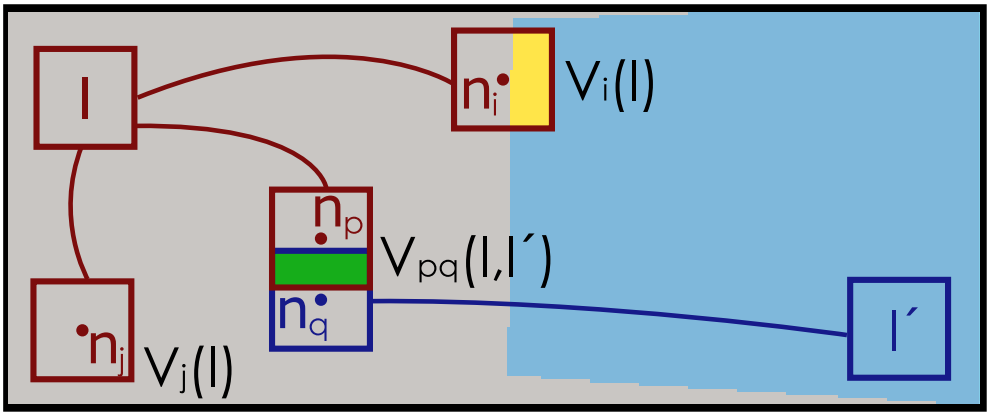


Figure 4.17: **Dynamic Background Potentials.** For boundary node n_i its single node potential $V_i(l)$ will be an SSD over the yellow region, while for nodes n_p, n_q their pairwise potential $V_{pq}(l,l')$ will be an SSD over the green region. Non-boundary node n_j has zero single node potential.

4.3.5.2 Single Node Potentials

Nodes n_i located on the border of a warped input image will already contain some initial content. Therefore, any label $\hat{l}_i$ assigned to these nodes should retain as much intensity information present as possible. As such, the single node potential $V_i(l)$ of assigning label l to node n_i represents how well the intensity information of label l agrees with the intensities present in the window W around the center of node n_i (Figure 4.17):

$$V_i(l_t) = \alpha \left[\frac{1}{|W|} \sum_{(x,y)\in W} B_i(x,y) \left(I_i(x,y) - I_t(x,y)\right)^2 \right] \tag{4.21}$$

Lastly, the pairwise potential $V_{ij}(l,l')$ must be defined in a way that provides us with both spatial and temporal consistency.

4.3.5.3 Spatial Pairwise Potentials

In case nodes i and j are spatial neighbors, the pairwise potential is defined by the normalized SSD over the area of overlap A:

$$V_{ij}^{S}(l,l') = \beta \left[\frac{1}{|A|} \sum_{(x,y)\in A} \left(I(x,y) - I'(x,y) \right)^2 \right] \tag{4.22}$$

4.3.5.4 Temporal Pairwise Potentials

When dealing with temporal neighbors, we want to encourage temporal continuity for dynamic elements. However, not all dynamic background movement is caused by actual scene movement, as some of it originates from parallax artifacts. Because incorporating this unwanted movement in our results leads to visual artifacts, we need to subdivide the label set L into a subset of static (L_S) and dynamic labels (L_D). This subdivision is based on thresholding, using the MAD values calculated in the foreground segmentation step.

$$\kappa(l) \;=\; \left(\frac{1}{|W_l|} \sum_{(x,y)\in W_l} MAD(x,y) \right)^2 \tag{4.23}$$

If $\kappa(l)$ exceeds a predefined threshold, label l will be considered dynamic (see Figure 4.16(b)).

Depending on which subset two labels l and l' are in, temporal costs are chosen to either encourage temporal continuity (for the dynamic elements), or to increase temporal coherence (for the static elements). In the first case, we will assign a penalty to subsequential labels in the output video that are not subsequential in the original sequence:

$$V_{ij}^{TD}(l,l') = \gamma \quad \texttt{if} \quad \left[t(n_i) - t(n_j) \right] \neq \left[t(l) - t(l') \right] \tag{4.24}$$

In case of static background elements, the pairwise potential is defined by the normalized SSD over their common spatial window W:

$$V_{ij}^{TS}(l,l') = \lambda \left[\frac{1}{|W|} \sum_{(x,y)\in W} \left(I(x,y) - I'(x,y) \right)^2 \right] \tag{4.25}$$

4.3.5.5 Total Energy Cost

Based on these formulations, where α, β, γ and λ are user-specified weights and $t(l)$ returns the label's frame number, a label $\hat{l}_i \in L$ should be assigned to each node n_i, so that the total energy cost $E(\{\hat{l}_i\})$ of the MRF is minimized, where:

$$E(\{\hat{l}_i\}) = \sum_{i=1}^{|N|} V_i(\hat{l}_i) + \sum_{(i,j)\in E} \left[V_{ij}^{S}(\hat{l}_i, \hat{l}_j) + V_{ij}^{T^*}(\hat{l}_i, \hat{l}_j) \right] \tag{4.26}$$

4.3.6 Gradient-domain Image/Video Compositing

When compositing the patches assigned to nodes of the respective MRFs of the background estimation and the video completion step, the resulting image can still exhibit visual artifacts (e.g. from exposure variations in the original video). In order to deal with these problems, we composite the patches in the gradient domain, as described in the work of Pérez et al. [Pérez 03b], using Dirichlet boundary conditions to guide the integration process. In order to avoid color flashing and flickering artifacts when applying the patches during video completion, we recommend using Wang's method [Wang 04] to integrate the image gradients in 3D, which is based on the weaker Neumann boundary conditions.

4.3.7 Visualization

In the end, the goal of our system is to re-dispay video sequences with a controlled camera motion, field of view and zoom.

4.3.7.1 Camera motion

Warping the dynamic video panorama back to the coordinate system of the original input footage can be done by simply applying the inverse of the homographies $\{\mathrm{H}_{r,i}\} = \{\mathrm{H}_{i,r}^{-1}\}$, computed during the registration step, to each respective frame of the dynamic panorama. If we want to control the rotational motion performed by the virtual camera, we first need to recover the original camera motion. To achieve this, we will need to calibrate the camera, seperating the intrinsic and extrinsic camera parameters. Assuming that not all rotations are about the same axis, we can linearly decompose the homographies $\mathrm{H}_{r,i}$ as described by de Agapito et al. [de Agapito 99]:

$$\mathrm{H}_{r,i} = \mathrm{K}_i \mathrm{R}_{r,i} \mathrm{K}_r^{-1} \tag{4.27}$$

Pre-multiplying or replacing rotation matrix $R_{r,i}$ with a user-controlled rotation R' will allow direct access to the virtual camera.

$$H'_{r,i} = K_i R' R_{r,i} K_r^{-1} \tag{4.28}$$

Also, as we know the translations of all pixels for each pair of neighboring frames, we have the option of adding blur to pixels that were not part of the original input video. This can easily be achieved by convoluting the selected pixels with an appropriate kernel.

4.3.7.2 Field of View

Besides the ability to control camera motion, we also allow the user to adjust the field of view. This brings up several complications: (a) by expanding the field of view, empty parts of the panorama can become visible when the camera reaches the edge of the panorama, and (b) when the rotation of virtual camera is adjusted, parts of the foreground element(s) may no longer be visible. To cope with these situations, we iteratively adjust the rotation R' and if needed the focal length of K_i. During this computation, we treat the absence of gaps in our output frame as a hard, and the visibilty of the actors as a soft constraint.

4.4 Results and Discussion

We have applied our algorithm to a variety of input sequences, chosen specifically to test individual components of our algorithm. For example, in order to test our registration algorithm, a skateboarding sequence with recurring loops in the frame topology was computed. Our waterfall scene contains both structured (miniature water wheel) and unstructured (waterfall) dynamic background elements.

In general, the augmentation of the original video sequences generates convincing results (depicted in Figure 4.18). Careful examination however will reveal occasional artifacts, in the form of 'popping' effects. These artifacts are usually the product of aperiodic background elements, or background elements without a full visible cycle, labeled as dynamic background. Their temporal continuity ends abruptly, resulting in the popping artifact.

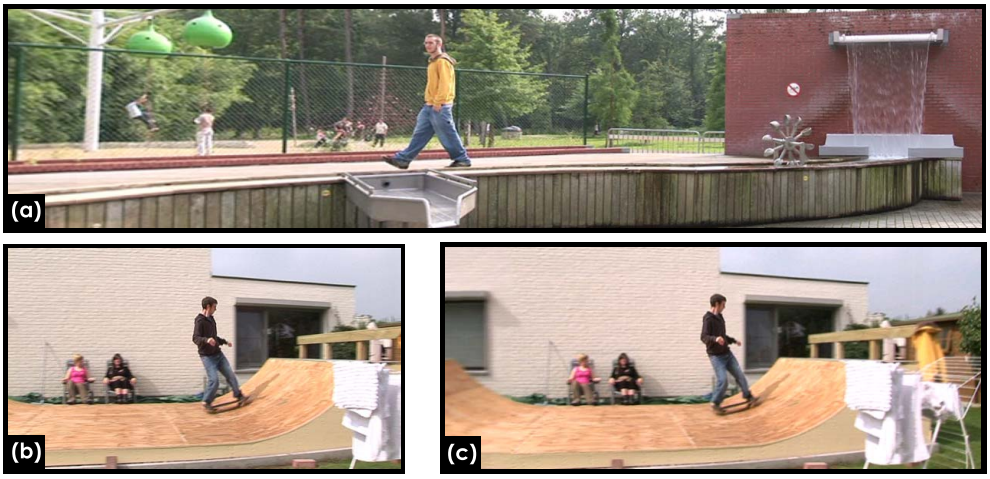


Figure 4.18: **Results produced by our Augmented Panoramic Video algorithm.** *(a)* A cropped panorama frame from our waterfall scene. *(b-c)* An input frame of our skateboarding sequence, and the processed frame with an expanded field of view. Motion blur has been added in an additional post-processing step.

4.4.1 Individual Component Analysis

The automatic registration of the video frames consistently provides us with accurate results, unless the underlying inter-frame warping procedure breaks down. This happens in two cases: (a) when comparing a severely blurred image with an undistorted one, and (b) when dealing with stochastic dynamic regions filling nearly the entire input image. We will look into the recent work of Yuan et al. [Yuan 07] to deal with the first issue, but we are unaware of any methods that can deal with the second.

Our static background estimation component produces high-quality static panoramas, under the assumptions that parallax effects stay within reasonable bounds and that all sharp background pixels are visible at least once within the entire sequence. A comparison of our technique to standard background subtraction methods is shown in Figure 4.19. It should be noted that this stage can be used as a stand-alone application for background estimation in cluttered scenes.

Our dynamic background estimation component generates convincing results, with the exception of the popping artifacts which we mentioned earlier. However,

Figure 4.19: Comparison of *(a)* temporal mean filtering, *(b)* temporal median filtering, and *(c)* our background estimation algorithm.

there are some limitations that need to be taken into account when applying our technique, e.g. BP algorithms tend to use large amounts of memory. This requires us to take several measures to make sure our algorithm does not unnecessarily squander its resources.

Whereas precomputing the single-node potentials relieves us from storing binary masks and segmentation information, label clustering and label pruning [Komodakis 06] reduce the amount of pairwise potentials that needs to be computed. It should be noted that the pairwise potentials only depend on intensity information stored in the labels. As a result, in case the amount of labels is sufficiently reduced

in number, it is possible to pre-compute and store all pairwise potentials in memory, without the need to retain the intensity images themselves. Storing the potentials in memory also reduces the required computation times from a number of days to a few hours, depending on the scene.

4.5 Conclusion and Future Work

Besides presenting the idea of editing panning/rotating video sequences using a full panoramic representation, we present a robust video mosaicing algorithm that produces high quality panoramas without parallax artifacts, seams or blurring, while retaining repetitive dynamic elements. Our technique allows the user to control the camera of a panning/rotating video in a post-processing step, allowing for a seamless change of aspect ratio or camera motion path. Furthermore, this technique also facilitates other post-processing steps such as adding motion blur or video stabilization.

4.5.1 Future Work

During our experiments, all video frames were warped to the image plane of the reference frame. This effectively reduces the resolution of the background pixels in the outer regions of our background panorama. In the future, we would like to test the effectiveness of our approach on other parametrizations of the scene intensities, such as cylindrical or spherical pixel coordinates.

Another interesting area for future work could be devising a hierarchical approach to our dynamic background estimation procedure. Building on the results from a lower resolution level, we might be able to narrow down the number of candidate labels for each new iteration.

Chapter 5

Conclusions and Future Work

Contents

In this dissertation, we investigated new methods to animate, synthesize and augment data in a video-based manner. These methods involve video-based synthesis for rigid objects applied on traffic sequences, articulated video sprites and pose synthesis illustrated on human characters and augmented panoramic video featuring dynamic foreground subjects.

These main parts of the dissertation can be summarized as follows:

5.1 Video-Based Synthesis for Rigid Objects

We presented a novel technique to synthesize videos featuring 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 the user.

A parameterized sprite appearance model is central in this approach. The model describes how the sprite evolves in shape and pixel intensity, and also allows for interpolation, extrapolation and compact storage. Using this information, new videos can be synthesized that feature the sprites, in such a way that the videos exhibit novel animated traffic situations. Synthesized videos featuring traffic congestions and suddenly stopping vehicles make this method ideally suited for training traffic detection systems.

Next to the data compression and the option to interpolate and extrapolate the data, our PCA vehicle sprite appearance model could furthermore be very useful to synthesize new vehicles. By means of analyzing the PCA data, one can create classifications for this data. For example, a class containing small vehicles can be created. Comparing the PCA data of this class with the PCA data of the vehicles that do not belong to this classification, some trait vectors can be identified that are associated with corresponding parameters of the PCA model. Isolated trait vectors for different classifications related to size, color, etc, of the vehicles can then be replaced or interpolated to change the appearance of existing vehicles.

While this method works well for synthesizing new videos, the resulting animations are limited by the fact that the sprites need to follow the same path as captured from the input sequence. Due to perspective projection of the camera, a sprite drawn on a different position will appear unrealistic. Therefore we would like to explore approximate geometric representations to more rigorously represent the relative rotation of the vehicles. A rigid geometric sprite representation could also allow to combine sprites extracted from different input sequences into a synthesized video. Furthermore we believe that the simulation of variable weather conditions and intricate illumination effects like vehicle headlights can be a valuable addition to our framework.

When the input videos can be captured in a controlled environment, an alternative exists in filming the vehicles driving on several different lanes and/or in different weather conditions. New trajectories for the sprites could then be synthesized by interpolating the different appearances.

5.2 Articulated Video Sprites and Image-Based Pose Synthesis

In this dissertation, two techniques were presented to synthesize and animate articulated characters in an image/video-based manner. The key to our technique is a matching algorithm that focuses on high-level 2D skeleton models instead of the visual appearance of the character, granting an accurate frame matching and a high level of control over the animation.

When synthesizing an input video, a sequence of virtual skeletons is used to define a desired target animation. The target skeletons are then matched with skeletons extracted from the source footage. The target skeletons only function as a guide for the new animation and do not force the input frames to match them exactly. Hence, the animation will achieve the desired effect without sacrificing the natural movement and appearance of the filmed subject.

At the moment, these skeletons are user-indicated in a semi-automatic manner. Automated skeleton extraction could resolve this and could furthermore deliver more consistent skeletons. Different users can for example indicate the “hand”-joint of a skeleton near the wrist or near the fingers. Consistency in these kind of choices can be an advantage related to skeleton matching.

While several heuristics try to enforce a smooth animation, the skeleton matching results sometimes still might not be fullfilling to the end-user. Since our dynamic programming approach is slightly dependent on the matching choice made for the last frame (from which it starts backtracking), user guided matching could be an interesting feature that ensures certain frames to be matched according to the desires of the user. Starting from user-“locked” frames, the matching algorithm could grow into an animation adhering to the underlying wishes of the user. Furthermore, instead of only using skeleton information to construct our matching cost function, we would like to look into the combination of both skeletal and pixel information to improve the matching between source and target frames.

Expanding the input pose space for the articulated video sprites algorithm, a novel technique was introduced to synthesize new poses from a set of input frames. This technique is based on selecting and merging different body parts into a desired pose. Only little user input is required to specify the poses (2D skeletons) of the input

images and the target pose. For each body part in the target skeleton, best matches are computed in the input poses, and the associated image parts are transfered to the final image. A triangle mesh based distance function is used to identify which pixels belong to which body part. Overlapping regions in the resulting image are merged while respecting the continuity of the image.

Even though this method allows for generating a wide variety of poses from only a small set of photographs, a target pose can only be met approximately. More variety is obtained by incorporating mesh deformation.

Our algorithm currently is unable to cope with situations where body parts occlude other ones, as well as with images where the subject is shot under large perspective differences. The availability of 3D skeletons and/or multi-camera information would be of great value when dealing with these problems. If this information is available, this technique would be highly suitable for use in 3D character animation applications [Starck 05, Starck 07].

Furthermore, an hierarchical skeleton model could be introduced to allow for an adaptable level of detail in the skeletons, for instance by switching to a detailed skeleton of a person's hand in a close-up.

5.3 Augmented Panoramic Video

The last part of this dissertation discussed a video augmentation algorithm supporting videos featuring panning movements as well as zooming. Besides presenting the idea of editing panning/rotating video sequences using a full panoramic representation, we presented a robust video mosaicing algorithm that produces high quality panoramas without parallax artifacts, seams or blurring, while retaining repetitive dynamic elements. Our technique allows the user to control the camera of a panning/rotating video in a post-processing step, allowing for a seamless change of aspect ratio or camera motion path. It also facilitates other post-processing steps such as adding motion blur or video stabilization.

During our experiments, all video frames were warped to the image plane of the reference frame. This effectively reduces the resolution of the background pixels in the outer regions of our background panorama. In the future, we would like to test

the effectiveness of our approach on other parametrizations of the scene intensities, such as cylindrical or spherical pixel coordinates.

The automatic registration of the video frames consistently provides us with accurate results, unless the underlying inter-frame warping procedure breaks down. This happens in two cases: (a) when comparing a severely blurred image with an undistorted one, and (b) when dealing with stochastic dynamic regions filling nearly the entire input image. We will look into the recent work of Yuan et al. [Yuan 07] to deal with the first issue, but we are unaware of any methods that can deal with the second.

Our static background estimation component produces high-quality static panoramas, under the assumptions that parallax effects stay within reasonable bounds and that all sharp background pixels are visible at least once within the entire sequence.

Our dynamic background estimation component generates convincing results, with the exception of the popping artifacts which we mentioned earlier. However, there are some limitations that need to be taken into account when applying our technique, e.g. BP algorithms tend to use large amounts of memory. This requires us to take several measures to make sure our algorithm does not unnecessarily squander its resources.

Whereas precomputing the single-node potentials relieves us from storing binary masks and segmentation information, label clustering and label pruning [Komodakis 06] reduce the amount of pairwise potentials that needs to be computed. It should be noted that the pairwise potentials only depend on intensity information stored in the labels. As a result, in case the amount of labels is sufficiently reduced in number, it is possible to pre-compute and store all pairwise potentials in memory, without the need to retain the intensity images themselves. Storing the potentials in memory also reduces the required computation times from a number of days to a few hours, depending on the scene.

Another interesting area for future work could be devising a hierarchical approach to our dynamic background estimation procedure. Building on the results from a lower resolution level, we might be able to narrow down the number of candidate labels for each new iteration.

5.4 Overall

While the presented algorithms have proven useful for already existing input data, one might wonder if the restriction of only using a single camera will still be an applicable restriction in the near future. Nowadays, high-tech hardware like stereo cameras are becoming available off-the-shelf, allowing consumers to obtain depth information useful for all kinds of post processing operations like refocusing and deblurring.

Furthermore, the recently disclosed open-source drivers for Microsofts Kinect [Microsoft 10] have opened up a broad range of interesting applications. The Kinect features an RGB camera, depth sensor (infrared laser projector combined with a monochrome CMOS sensor) and multi-array microphone running proprietary software, which provides full-body 3D motion capture, facial recognition and voice recognition capabilities. These systems will certainly grow to be an important part of computer graphics and vision research and should not be ignored.

By incorporating the Microsoft Kinect setup in our Articulated Video Sprites framework, the process of sprite segmentation and user assisted skeleton extraction could be automated and could even deliver 3D skeletons.

Appendices

Appendix A

Scientific Contributions and Publications

The following list of publications, presented at scientific international conferences, contains work that is part of this dissertation:

[Vanaken 06b] *Cedric Vanaken, Tom Mertens & Philippe Bekaert.* Video-Based Rendering of Traffic Sequences. *In Proceedings of Winter School of Computer Graphics (WSCG), pages 161–168, 2006*

[Vanaken 06a] *Cedric Vanaken, Mark Gerrits & Philippe Bekaert.* Articulated Video Sprites. *In Proceedings of Eurographics, pages 69–72, 2006*

[Hermans 08] *Chris Hermans, Cedric Vanaken, Tom Mertens, Frank Van Reeth & Philippe Bekaert.* Augmented Panoramic Video. *Computer Graphics Forum, vol. 27, no. 2, pages 281–290, 2008*

[Vanaken 08] *Cedric Vanaken, Chris Hermans, Tom Mertens, Fabian Di Fiore, Philippe Bekaert & Frank Van Reeth.* Strike a Pose: Image-Based Pose Synthesis. *In VMV, pages 131–138, 2008*

The following work is not part of this dissertation:

[Cuypers 08] *Tom Cuypers, Cedric Vanaken, Yannick Francken, Frank Van Reeth & Philippe Bekaert.* A Multi-Camera Framework for Interactive Videogames. *In GRAPP '08: International Joint Conference on Computer Vision and Computer Graphics Theory and Applications, pages 443–449. INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2008*

[DiFiore 08] *Fabian DiFiore, Peter Quax, Cedric Vanaken, Wim Lamotte & Frank Van Reeth.* Conveying Emotions through Facially Animated Avatars in Networked Virtual Environments. *Motion In Games (MIG08), Lecture Notes in Computer Science LNCS series, pages 222–233, 2008*

[Cuypers 09] *Tom Cuypers, Yannick Francken, Cedric Vanaken, Frank Van Reeth & Philippe Bekaert.* Smartphone Localization on Interactive Surfaces Using the Built-in Camera. *In Procams 2009: IEEE International Workshop on Projector-Camera Systems, 2009*

Bijlage B

Samenvatting (Dutch Summary)

Vermits digitale foto- en videocameras tegenwoordig een onmisbaar deel uitmaken van talrijke huishoudens wordt er ook meer en meer onderzoek en software ontwikkeling gericht op het ontwerpen van tools die het manipuleren van beelden en videos vereenvoudigen. Veel krachtige tools zijn reeds beschikbaar, gaande van voorgrond segmentatie tot het 'ontblurren' van beelden.

In deze thesis concentreren we ons voornamelijk op de zogenaamde beeld- en videogebaseerde animatie-, synthese- en 'uitbreidings'technieken. Deze technieken nemen één of meerdere beelden als invoer, analyseren deze data om ze vervolgens te animeren, synthetiseren of vermeerderen om zo nieuwe beelden te bekomen die geïnspireerd zijn door de oorspronkelijke invoer. Het best gekende video-gebaseerde synthese voorbeeld is het *Video Textures* algoritme van Schödl et al. [Schödl 00b]. Vertrekkend van een relatief korte video sequentie kan een nieuwe video geproduceerd worden door de invoer frames te herordenen. Deze frames worden op zo een manier herschikt dat de resulterende video oneindig kan herhaald worden in een lus, men trouw blijft aan de oorspronkelijke inhoud en dat visueel storende transities vermeden worden.

Een belangrijke uitbreiding op dit werk kwam er in 2002 met de *Video Sprites* techniek van Schödl et al. [Schödl 02]. In plaats van volledige frames te hergebruiken, worden nu beelden van een gefilmd onderwerp uit de video geknipt en achteraf aan mekaar geplakt om nieuwe animaties te bekomen. Waar we met Video Textures

voornamelijk gelimiteerd waren tot het werken met videos die inherent repetitieve inhoud bevatten zoals gevisualiseerd door natuurlijke fenomenen, laat het Video Sprites algoritme toe om kleinere onderwerpen zoals hamsters of vliegen te animeren. Beide technieken werken op een beperkt aantal types van invoer videos. Deze thesis introduceert twee video gebaseerde animatie en synthese technieken die gericht zijn op videos die verschillende soorten onderwerpen bevatten.

Allereerst presenteren we een nieuwe techniek om videos die auto-verkeer visualiseren te synthetiseren. Vertrekkende van een invoer verkeers-video kunnen we een verkeers-scène reproduceren vanuit hetzelfde standpunt, waarin de configuratie en de trajecten van de voertuigen gewijzigd zijn door de gebruiker. De belangrijkste applicatie van deze techniek bevindt zich in het valideren en trainen van camera-gebaseerde verkeers-analyse systemen, bijvoorbeeld ongevallen of file detectie.

Ten tweede introduceren we een nieuwe aanpak voor de Video Sprites techniek, waarbij gearticuleerde onderwerpen zoals dieren en mensen geanimeerd kunnen worden. Gearticuleerde onderwerpen bezitten namelijk een onderliggende skeletstructuur. We buiten dit gegeven uit in ons algoritme door gebruik te maken van een 2D skelet-voorstelling in plaats van de visuele voorstelling van het onderwerp in kwestie. Wanneer we beschikken over een gewenste doel-animatie, gedefinieerd door een opeenvolging van skeletten, gaan we op zoek naar de invoer frames die hier het best mee overeenkomen en ordenen we deze frames op zulk een manier dat het gegeven onderwerp de gewenste beweging uitvoert.

Wanneer enkel de gegeven invoer frames gebruikt worden om nieuwe animaties te bekomen, worden zowel de kwaliteit als de verscheidenheid van de resultaten gelimiteerd door de diversiteit van de invoer data. Om de hoeveelheid beschikbare invoer data uit te breiden, introduceren we een techniek die het mogelijk maakt om nieuwe poses te creëren door invoer afbeeldingen te synthetiseren. Reeds bestaande aanpakken vervormen meestal één enkel beeld, wat regelmatig resulteert in een beeld dat fouten vertoont, voornamelijk door zichtbare artefacten in textuur en belichting. We presenteren een nieuwe beeld-gebaseerde pose-synthese techniek die details in textuur accuraat reconstrueert door informatie van verschillende foto's te combineren. Gegeven een 2D pose die door de gebruiker wordt aangegeven als de gewenste pose, voegt onze oplossing verschillende delen van de invoer foto's samen op zulk een manier dat het resultaat overeenkomt met deze pose, enkel gebruik makend van

2D operaties. We illustreren hoe nieuwe poses gegenereerd kunnen worden met behulp van slechts enkele voorbeeld afbeeldingen en dit zonder dat er een grote mate aan gebruikers-interactie verwacht wordt. De combinatie van deze techniek met het eerder vermeldde Articulated Video Sprites algoritme verzekert een groot en divers aanbod aan mogelijke poses om het gefilmde onderwerp mee te animeren.

Voor de finale contributie van deze uiteenzetting nemen we een zijstap van de animatie algoritmes en bespreken we het *Augmented Panoramic Video* videobewerking en 'mosaicing' systeem. We mikken hierbij op een veel voorkomend type van video sequenties, waarbij de cameraman een scène en terwijl de camera roteert om zo een panorama te registreren en mogelijks inzoomt naar specifieke gebieden die dynamische onderwerpen kunnen bevatten zoals mensen of dieren. We willen zulke videosequenties hertonen en manipuleren op een betekenisvolle manier en presenteren een techniek die de gebruiker controle geeft over zowel de beweging als het gezichtsveld van de camera. De gepresenteerde resultaten tonen aan dat deze techniek panorama's van hoge kwaliteit produceert zonder de zogenaamde parallax effecten of zichtbare naden of onscherpe gebieden, en terwijl repetitieve dynamische elementen uit de originele invoer video behoudt.

Bibliography

[Adies 11] Amit Adies, Tamar Avraham & Yoav Schechner. *Multi-Scale Ultrawide Foveated Video Extrapolation*. In IEEE-ICCP (International Conference on Computer Photography), 2011.

[Agarwala 05] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David Salesin & Richard Szeliski. *Panoramic video textures*. ACM Trans. Graph., vol. 24, no. 3, pages 821–827, 2005.

[Avidan 07] Shai Avidan & Ariel Shamir. *Seam carving for content-aware image resizing*. ACM Trans. Graph., vol. 26, no. 3, page 10, 2007.

[Barrett 02] William A. Barrett & Alan S. Cheney. *Object-based image editing*. In SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pages 777–784, 2002.

[Bhat 04] Kiran S. Bhat, Steven M. Seitz, Jessica K. Hodgins & Pradeep K. Khosla. *Flow-based video synthesis and editing*. In SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, pages 360–363, 2004.

[Bishop 06] Christopher Bishop. Pattern recognition and machine learning (information science and statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[Blanz 99] Volker Blanz & Thomas Vetter. *A morphable model for the synthesis of 3D faces*. In SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 187–194, 1999.

[Bregler 97] Christoph Bregler, Michele Covell & Malcolm 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.

[Bregler 02] Christoph Bregler, Lorie Loeb, Erika Chuang & Hrishi Deshpande. *Turning to the masters: motion capturing cartoons*. ACM Trans. Graph., vol. 21, pages 399–407, July 2002.

[Brown 03] Matthew Brown & David G. Lowe. *Recognising Panoramas*. In ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision, page 1218, Washington, DC, USA, 2003. IEEE Computer Society.

[Brown 07a] Matthew Brown, Richard I. Hartley & David Nister. *Minimal Solutions for Panoramic Stitching*. In CVPR '07: Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 1–8, june 2007.

[Brown 07b] Matthew Brown & David G. Lowe. *Automatic Panoramic Image Stitching using Invariant Features*. Int. J. Comput. Vision, vol. 74, pages 59–73, August 2007.

[Burns 04] Burns. *Mike Burns, Christopher DeCoro and Anaya Misra, Speed Trap* `http://www.cs.princeton.edu/~cdecoro /traffic/`, 2004.

[Chan 05] Antoni Chan & Nuno Vasconcelos. *Classification and retrieval of traffic video using auto-regressive stochastic processes*. Proceedings of IEEE Intelligent Vehicles Symposium (IEEEIV), 2005.

[Chan 06] Antoni Chan & Nuno Vasconcelos. *Layered Dynamic Textures*. In Y. Weiss, B. Schölkopf & J. Platt, editeurs, Advances in Neural Information Processing Systems 18. MIT Press, Cambridge, MA, 2006.

[Chew 87] Paul L. Chew. *Constrained Delaunay triangulations*. In SCG '87: Proceedings of the third annual symposium on Computational geometry, pages 215–222, New York, NY, USA, 1987. ACM.

[Christensen 06] Per H. Christensen, Julian Fong, David M. Laur & Dana Batali. *Ray Tracing for the Movie 'Cars'*. In Proceedings of the IEEE Symposium on Interactive Ray Tracing, pages 1–6, 2006.

[Chuang 01] Yung-Yu Chuang, Brian Curless, David H. Salesin & Richard Szeliski. *A Bayesian Approach to Digital Matting*. In Proceedings of IEEE CVPR 2001, volume 2, pages 264–271. IEEE Computer Society, December 2001.

[Chuang 05] Yung-Yu Chuang, Dan B Goldman, Ke Colin Zheng, Brian Curless, David H. Salesin & Richard Szeliski. *Animating pictures with stochastic motion textures*. In SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pages 853–860, 2005.

[CMU] CMU. *Graphics Lab Motion Capture Database, http://mocap.cs.cmu.edu/*.

[Coifman 98] Ben Coifman, David Beymer, Philip McLauchlan & Jitendra Malik. *A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance*. Transportation Research: Part C, vol. 6, no. 4, pages 271–288, 1998.

[Collomosse 03] John Collomosse, Dave Rowntree & Peter Hall. *Video Analysis for Cartoon-like Special Effects*. In In 14th British Machine Vision Conference, pages 749–758, 2003.

[Colombari 06] Andrea Colombari, Andrea Fusiello & Vittorio Murino. *Background Initialization in Cluttered Sequences*. 5th Workshop on Perceptual Organization in Computer Vision, page 197, 2006.

[Cootes 01] Timothy Cootes, Gareth Edwards & Christopher Taylor. *Active Appearance Models*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 23, pages 681–685, June 2001.

[Criminisi 03] Antonio Criminisi, Patrick Perez & Kentaro Toyama. *Object removal by exemplar-based inpainting*. In CVPR '03: Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages II: 721–728, 2003.

[Cucchiara 00] R. Cucchiara, C. Grana, M. Piccardi & A. Prati. *Statistic and knowledge-based moving object detection in traffic scenes*. In in Proceedings of IEEE IntŠl Conference on Intelligent Transportation Systems, pages 27–32, 2000.

[Cuypers 08] Tom Cuypers, Cedric Vanaken, Yannick Francken, Frank Van Reeth & Philippe Bekaert. *A Multi-Camera Framework for Interactive Videogames*. In GRAPP ’08: International Joint Conference on Computer Vision and Computer Graphics Theory and Applications, pages 443–449. INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2008.

[Cuypers 09] Tom Cuypers, Yannick Francken, Cedric Vanaken, Frank Van Reeth & Philippe Bekaert. *Smartphone Localization on Interactive Surfaces Using the Built-in Camera*. In Procams 2009: IEEE International Workshop on Projector-Camera Systems, 2009.

[de Agapito 99] Lourdes de Agapito, Richard I. Hartley & Eric Hayman. *Linear calibration of a rotating and zooming camera*. In CVPR ’99: Proceedings of the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 15–21, 1999.

[de Juan 04] Christina de Juan & Bobby Bodenheimer. *Cartoon textures*. Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 267–276, 2004.

[Debevec 96] Paul E. Debevec, Camillo J. Taylor & Jitendra Malik. *Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach*. In SIGGRAPH ’96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pages 11–20, 1996.

[Devernay 01] Frederic Devernay & Olivier Faugeras. *Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments*. Mach. Vision Appl., vol. 13, no. 1, pages 14–24, 2001.

[DiFiore 08] Fabian DiFiore, Peter Quax, Cedric Vanaken, Wim Lamotte & Frank Van Reeth. *Conveying Emotions through Facially Animated*

Avatars in Networked Virtual Environments. Motion In Games (MIG08), Lecture Notes in Computer Science LNCS series, pages 222–233, 2008.

[Drori 03] Iddo Drori, Daniel Cohen-Or & Hezy Yeshurun. *Fragment-based image completion*. ACM Trans. Graph., vol. 22, no. 3, pages 303–312, 2003.

[Dutré 03] Philip Dutré, Philippe Bekaert & Kavita Bala. Advanced global illumination. AK Peters, 2003.

[Efros 99] Alexei A. Efros & Thomas K. Leung. *Texture Synthesis by Non-Parametric Sampling*. In ICCV '99: Proceedings of the International Conference on Computer Vision - Volume 2, page 1033, 1999.

[Fischler 81] Martin A. Fischler & Robert C. Bolles. *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Comm. of the ACM archive, vol. 24, pages 381 – 395, 1981.

[Fitzgibbon 01] Andrew W. Fitzgibbon. *Stochastic Rigidity: Image Registration for Nowhere-Static Scenes*. Proceedings of International Conference on Computer Vision, pages 662–669, 2001.

[Gloyer 95] Brian Gloyer, Hhamid K. Aghajan, Kai-Yeung S. Siu & Thomas Kailath. *Video-based freeway monitoring system using recursive vehicle tracking*. In Proceedings of SPIE, volume 2421, pages 173–180, February 1995.

[Gonzalez 01] Rafael C. Gonzalez & Richard E. Woods. Digital image processing. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[Haevre 05] William Van Haevre, Fabian Di Fiore & Frank Van Reeth. *Uniting cartoon textures with computer assisted animation*. In GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia, pages 245–253, New York, NY, USA, 2005. ACM.

[Hampel 86] Frank Hampel, Elvezio Ronchetti, Peter Rousseeuw & Werner Stahel. Robust statistics: The approach based on influence functions. Wiley, 1986.

[Hartley 97] Richard I. Hartley. *In Defense of the Eight-Point Algorithm*. IEEE Transactions On Pattern Analysis And Machine Intelligence, vol. 19, no. 6, pages 580–593, 1997.

[Hartley 04] Richard Hartley & Andrew Zisserman. Multiple view geometry in computer vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[Hermans 08] Chris Hermans, Cedric Vanaken, Tom Mertens, Frank Van Reeth & Philippe Bekaert. *Augmented Panoramic Video*. Computer Graphics Forum, vol. 27, no. 2, pages 281–290, 2008.

[Hoiem 05] Derek Hoiem, Alexei A. Efros & Martial Hebert. *Automatic photo pop-up*. In SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pages 577–584, 2005.

[Hornung 07] Alexander Hornung, Ellen Dekkers & Leif Kobbelt. *Character animation from 2D pictures and 3D motion data*. ACM Transactions on Graphics, vol. 26, no. 1, page 1, 2007.

[Igarashi 05] Takeo Igarashi, Tomer Moscovich & John F. Hughes. *As-rigid-as-possible shape manipulation*. In SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pages 1134–1141. ACM, 2005.

[ITS 11] ITS. *Europe ITS - Intelligent Transport Systems and Services* `http://ec.europa.eu/transport/its/index_en.htm` , 2011.

[Jain 99] Anil K. Jain, M. Narasimha Murty & Patrick J. Flynn. *Data clustering: a review*. ACM Computing Surveys, vol. 31, no. 3, pages 264–323, 1999.

[Jojic 01] Nebojsa Jojic & Brendan Frey. *Learning Flexible Sprites in Video Layers*. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2001.

[Joshi 10] Neel Joshi, Wojciech Matusik, Edward H. Adelson & David J. Kriegman. *Personal photo enhancement using example images*. ACM Trans. Graph., vol. 29, pages 12:1–12:15, April 2010.

[Kanatani 99] KenIchi Kanatani & Naoya Ohta. *Accuracy Bounds and Optimal Computation of Homography for Image Mosaicing Applications*. In ICCV (1), pages 73–78, 1999.

[Kang 00] Eun-Young Kang, Isaac Cohen & Gerard Medioni. *A Graph-based Global Registration for 2D Mosaics*. In Proceedings of lnternational Conference on Pattern Recognition 2000, pages 257–260, 2000.

[Kavan 08] Ladislav Kavan, Simon Dobbyn, Steven Collins, Jiří Žára & Carol O'Sullivan. *Polypostors: 2D polygonal impostors for 3D crowds*. In SI3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games, pages 149–155, New York, NY, USA, 2008. ACM.

[Kent 92] James R. Kent, Wayne E. Carlson & Richard E. Parent. *Shape transformation for polyhedral objects*. In SIGGRAPH Comput. Graph., volume 26, pages 47–54, New York, NY, USA, July 1992. ACM.

[Kim 03] ZuWhan Kim & Jitendra Malik. *Fast Vehicle Detection with Probabilistic Feature Grouping and its Application to Vehicle Tracking*. In ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision, page 524, 2003.

[Komodakis 06] Nikos Komodakis & Georgios Tziritas. *Image Completion Using Global Optimization*. In CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 442–452, 2006.

[Kwatra 03] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk & Aaron Bobick. *Graphcut textures: image and video synthesis using graph cuts*. ACM Trans. Graph., vol. 22, no. 3, pages 277–286, 2003.

[Kwatra 05] Vivek Kwatra, Irfan Essa, Aaron Bobick & Nipun Kwatra. *Texture optimization for example-based synthesis*. ACM Trans. Graph., vol. 24, no. 3, pages 795–802, 2005.

[Lalonde 07] Jean-François Lalonde, Derek Hoiem, Alexei A. Efros, Carsten Rother, John Winn & Antonio Criminisi. *Photo clip art*. In SIGGRAPH '07: ACM SIGGRAPH 2007 papers, volume 26, 2007.

[Litvin 03] Andrew Litvin, Janusz Konrad & William C. Karl. *Probabilistic video stabilization using Kalman filtering and mosaicking*. In IS&T/SPIE Symposium on Electronic Imaging, Image and Video Communications and Proc., September 2003.

[Liu 05] Ce Liu, Antonio Torralba, William Freeman, Frédo Durand & Edward Adelson. *Motion magnification*. In ACM SIGGRAPH 2005 Papers, SIGGRAPH '05, pages 519–526, New York, NY, USA, 2005. ACM.

[Liu 06] Feng Liu & Michael Gleicher. *Video retargeting: automating pan and scan*. In MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia, pages 241–250, 2006.

[Lucas 81a] Bruce D. Lucas & Takeo Kanade. *An Iterative Image Registration Technique with an Application to Stereo Vision*. In IJCAI '81: Proceedings of the 7th International Joint Conference on Artificial Intelligence, pages 674–679, April 1981.

[Lucas 81b] Bruce D. Lucas & Takeo Kanade. *An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)*. In Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), pages 674–679, April 1981.

[Marzotto 04] Roberto Marzotto, Andrea Fusiello & Vittorio Murino. *High Resolution Video Mosaicing with Global Alignment*. CVPR '04: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pages 692–698, 2004.

[Matsushita 05] Yasuyuki Matsushita, Eyal Ofek, Xiaoou Tang & Heung-Yeung Shum. *Full-Frame Video Stabilization*. In CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 50–57, 2005.

[Matusik 03] Wojciech Matusik, Hanspeter Pfister, Matt Brand & Leonard McMillan. *A data-driven reflectance model*. ACM Trans. Graph., vol. 22, no. 3, pages 759–769, 2003.

[McIvor 00] Alan M. McIvor. *Background subtraction techniques*. In Proceedings of Image and Vision Computing, Auckland, New Zealand, 2000.

[Microsoft 10] Microsoft. *Kinect, http://www.xbox.com/en-US/kinect/*, 2010.

[Oh 01] Byong Mok Oh, Max Chen, Julie Dorsey & Frédo Durand. *Image-based modeling and photo editing*. In SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 433–442, 2001.

[Oh 03] Cheol Oh & Stephen G. Ritchie. *Anonymous Vehicle Tracking for Real-time Traffic Surveillance and Performance on Signalized Arterials*. Transportation Research Board, 2003.

[Pérez 03a] Patrick Pérez, Michel Gangnet & Andrew Blake. *Poisson image editing*. In SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, pages 313–318. ACM, 2003.

[Pérez 03b] Patrick Pérez, Michel Gangnet & Andrew Blake. *Poisson image editing*. ACM Transactions on Graphics (SIGGRAPH), vol. 22, no. 3, pages 313–318, 2003.

[Prati 01] Andrea Prati, Ivana Mikic, Costantino Grana & Mohan M. Trivedi. *Shadow Detection Algorithms for Traffic Flow Analysis: a Comparative Study*. In Proc. IEEE Intelligent Transportation Systems Conf, pages 340–345, 2001.

[Rav-Acha 08] Alex Rav-Acha, Pushmeet Kohli, Carsten Rother & Andrew Fitzgibbon. *Unwrap Mosaics: A new representation for video editing*. ACM Transactions on Graphics (SIGGRAPH 2008), August 2008.

[Sawhney 98] Harpreet S. Sawhney, Steve Hsu & Rakesh Kumar. *Robust Video Mosaicing through Topology Inference and Local to Global Alignment*. In ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume II, pages 103–119, 1998.

[Schneider 01] Tapio Schneider & Arnold Neumaier. *Algorithm 808: ARfit—a matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models*. ACM Trans. Math. Softw., vol. 27, no. 1, pages 58–65, 2001.

[Schödl 00a] Arno Schödl & Irfan Essa. *Machine Learning for Video-Based Rendering*. In Proceedings of NIPS, pages 1002–1008, 2000.

[Schödl 00b] Arno Schödl, Richard Szeliski, David H. Salesin & Irfan Essa. *Video Textures*. Proceedings of SIGGRAPH, pages 489–498, 2000.

[Schödl 02] Arno Schödl & Irfan Essa. *Controlled Animation of Video Sprites*. SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 121–127, 2002.

[Starck 05] Jonathan Starck, Gregor Miller & Adrian Hilton. *Video-based character animation*. In SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 49–58, New York, NY, USA, 2005. ACM.

[Starck 07] Jonathan Starck & Adrian Hilton. *Surface Capture for Performance Based Animation*. IEEE Computer Graphics and Applications, vol. 27, no. 3, pages 21–31, 2007.

[Starpulse 08] Starpulse. *Starpulse Supermodels image gallery, http://www.starpulse.com/Supermodels/*, 2008.

[Stauffer 99] Chris Stauffer & Eric Grimson. *Adaptive background mixture models for real-time tracking*. Computer Vision and Pattern Recognition, vol. 2, pages 246–252, 1999.

[Sun 04] Jian Sun, Jiaya Jia, Chi-Keung Tang & Heung-Yeung Shum. *Poisson matting*. ACM Trans. Graph., vol. 23, no. 3, pages 315–321, 2004.

[Sun 05] Jian Sun, Lu Yuan, Jiaya Jia & Heung-Yeung Shum. *Image completion with structure propagation*. In SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pages 861–868, 2005.

[Tomasi 91] Carlo Tomasi & Takeo Kanade. *Detection and Tracking of Point Features*. Rapport technique CMU-CS-91-132, Carnegie Mellon University, April 1991.

[Traficon] Traficon. *http://www.traficon.com/*.

[Vanaken 06a] Cedric Vanaken, Mark Gerrits & Philippe Bekaert. *Articulated Video Sprites*. In Proceedings of Eurographics, pages 69–72, 2006.

[Vanaken 06b] Cedric Vanaken, Tom Mertens & Philippe Bekaert. *Video-Based Rendering of Traffic Sequences*. In Proceedings of Winter School of Computer Graphics (WSCG), pages 161–168, 2006.

[Vanaken 08] Cedric Vanaken, Chris Hermans, Tom Mertens, Fabian Di Fiore, Philippe Bekaert & Frank Van Reeth. *Strike a Pose: Image-Based Pose Synthesis*. In VMV, pages 131–138, 2008.

[Wang 94] John Y.A. Wang & Edward H. Adelson. *Representing Moving Images with Layers*. IEEE Transactions on Image Processing Special Issue: Image Sequence Compression, vol. 3, no. 5, pages 625–638, September 1994.

[Wang 04] Hongcheng Wang, Ramesh Raskar & Narendra Ahuja. *Seamless Video Editing*. In ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3, pages 858–861. IEEE Computer Society, 2004.

[Wang 08] Yanzhen Wang, Kai Xu, Yueshan Xiong & Zhiquan Cheng. *2D Shape Deformation Based on Rigid Square Matching*. In Computer Animation and Social Agents (CASA2008), Journal of Computer Animation and Virtual Worlds, 2008.

[Wexler 04] Yonatan Wexler, Eli Shechtman & Michal Irani. *Space-Time Video Completion*. In CVPR '04: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 1, pages 120–127, 2004.

[Yedidia 01] Jonathan S. Yedidia, William T. Freeman & Yair Weiss. *Understanding belief propagation and its generalizations*. In International Joint Conference on Artificial Intelligence 2001 Distinguished Lecture Track, 2001.

[Yuan 07] Lu Yuan, Jian Sun, Long Quan & Heung-Yeung Shum. *Blurred/Non-Blurred Image Alignment using Sparseness Prior*. In ICCV '07: Proceedings of the International Conference on Computer Vision, 2007.

[Zhang 03] Chengcui Zhang, Shu-Ching Chen, Mei-Ling Shyu & Srinivas Peeta. *Adaptive Background Learning for Vehicle Detection and Spatio-*

Temporal Tracking. Proceedings of the Fourth IEEE Pacific-Rim Conference On Multimedia, pages 1–5, 2003.