Available online at www.sciencedirect.com
ScienceDirect
Image and Vision Computing 25 (2007) 1154–1164
www.elsevier.com/locate/imavis

View synthesis by the parallel use of GPU and CPU

Indra Geys [a,*], Luc Van Gool [a,b]

[a] Katholieke Universiteit Leuven, ESAT/PSI-VISICS, Kasteelpark Arenberg 10, 3001 Leuven, Belgium
[b] Swiss Federal Institute of Technology, ETH/BIWI, Sternwartstrasse 7, 8092 Zürich, Switzerland

Abstract

We present an algorithm for efficient depth calculations and view synthesis. The main goal is the on-line generation of realistic interpolated views of a dynamic scene. The inputs are video-streams originating from two or more calibrated, static cameras.

Efficiency is accomplished by the parallel use of the CPU and the GPU in a multi-threaded implementation. The input images are projected on a plane sweeping through 3D space, using the hardware accelerated transformations available on the GPU. A correlation measure is calculated simultaneously for all pixels on the plane and is compared at the different plane positions. A noisy 'virtual' view and a crude depth map result in very limited time. We apply a min-cut/max-flow algorithm on a graph, implemented on the CPU, to ameliorate this result by a global optimisation.

Keywords: View synthesis; Multi-camera; Graphical board; Depth map; Graph-cut

1. Introduction

Given multiple video-streams of a dynamic scene, an algorithm is presented to create image sequences from novel viewpoints at interactive rates. A depth map is calculated on the fly and textures are rendered onto this map to create the 'virtual' view. The input video-streams come from two or more static calibrated cameras. The work aims at the development of more advanced tele-working, tele-teaching and video-conferencing environments. Remote collaboration, e.g. on product design, becomes much easier. No moving cameras are needed. Instead, camera movements are simulated and the viewpoint can be optimised.

Several approaches can be used to synthesise novel views. One set of image-based methods is called morphing. A rectification step, feature matching, image warping and linear colour interpolation are used to generate the new view out of other views. One of the first systems was presented in 1993 by Chen and Williams [1]. Geometrically valid morphs between views were proposed by Seitz and Dyer [2] and Werner et al. [3]. Lhuillier and Quan [4] reconstruct matched planar patches using a homography. After that, a joint view triangulation is defined to handle partially occluded areas. The scene is supposed to consist of one static object. The recent work of Criminisi et al. [5] presents the generation of a cyclopean[1] view of a stereo pair of an upper body. Their system is based on a dynamic programming algorithm for the generation of the disparity map.

Light field and lumigraph rendering [6,7] form a second group of methods. A large collection of 2D images are used to reconstruct a function that characterises the flow of light through the 3D space. Once this function is known, view synthesis is quite straightforward as illustrated by Schirmacher et al. [8]. However obtaining, transmitting and processing such a very dense sampling of the environment render these methods impractical for dynamic scenes. Apart from this, the many cameras would also block the view for a local audience (often present in tele-teaching applications).

As an example of a third possible solution, Matusik et al. [9] compute and shade visual hulls [10], from images of four calibrated cameras. Multiple computers are used

* Corresponding author. Tel.: +32 0 16 32 10 61; fax: +32 0 16 32 17 23.
E-mail address: igeys@esat.kuleuven.be (I. Geys).
URL: http://www.esat.kuleuven.be/~igeys (I. Geys).

[1] Cyclopean means mid-way between left and right input cameras.

0262-8856/$ - see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.imavis.2006.07.023

for the computation and the rendering. Similarly, polyhedral visual hulls [11] are based on epipolar geometry and provide view-independent rendering through a mesh and a texture representation. A visual hull needs a large number of different views for recovering geometric details and it cannot recover concave regions no matter how many images are used. This poses problems for faces, which are extremely important for our intended applications.

A fourth group of approaches consist of the reconstruction of a 3D model, and then rendering the model from the desired view. Debevec et al. [12] use view dependent texture-mapping, in which the textures are projected onto the geometry using a 'view map' for every polygon. Recently, Zitnick et al. [13] developed a system to create even higher quality interpolations. They can process recorded video-streams of moving objects. However the depth calculation remains off-line. Only the visualisation of the high-quality intermediate views is done at interactive rates.

Real-time calculation of depth without the use of dedicated hardware recently became feasible [14–17]. Ansar et al. [15] use bilateral filtering, while MMX-optimised code of a three-dimensional similarity accumulator is used by Schmidt et al. [16]. A plane sweeping algorithm [18] is described by Yang and Pollefeys [17].

In this research we try to overcome some of the limitations of the aforementioned systems. We want to generate synthetic views of dynamic scenes on-the-fly. This implies that per frame *off-line preprocessing or depth calculation* is not an option. The process from input images to synthesised views should be performed on-line. Therefore, the *execution speed* is of primary importance. In this work we focus on achieving a balanced load between CPU and GPU. We also strive for an *increased level of automation*. In contrast to earlier work [17], which mainly proved feasibility of GPU accelerated plane sweeping, we make this process fully automatic and more robust. The search range for the sweep is adapted from frame to frame without manual intervention. As a side effect, this also provides us with a rough histogram-based 3D tracking of the foreground (in our case typically a person) throughout the video-streams. Last but not least we try to strike a balance between computation time and quality.

The outline of the remainder of the paper is as follows. In Section 2 we provide a general overview of the system. Section 3 explains the segmentation algorithm, while Section 4 covers radial distortion correction and sparse correspondence search. Section 5 focuses on the GPU-based plane sweep and Section 6 on the graph-cut based regularisation. The actual generation of new views is highlighted in Section 7. Experimental results are shown in Section 8, followed by a conclusion in Section 9.

2. Overview of our system

One of the major challenges is to compute a reasonably accurate depth map given the available time between consecutive frames. To accomplish this we make a distinction between foreground and background. A *segmentation algorithm* separates both. For the foreground, the computation time is critical. A more accurate, but slower depth calculation can be applied to the background, since it is assumed to be static.

The problem of generating a dense depth map in limited time is addressed by a combined use of *a plane sweep algorithm implemented on the GPU*, followed by *a graph-cut based regularisation on the CPU*. Unlike most stereo algorithms, the depth map is not calculated with respect to one of the input cameras. Instead, the calculations are performed relative to the desired 'virtual' camera.

The same depth calculation algorithm can be used for the foreground and the background. However, due to its larger area and its more extended depth range, the computation time for the background is higher than for the foreground only. Therefore, we opt to recalculate only the foreground depth map every frame. For the background, only the texture is adjusted, to compensate for illumination changes.

Once the depth map is determined, *a triangle mesh* is sampled and rendered in OpenGl. Novel view synthesis is accomplished by *view dependent blending of the textures* on this mesh.

In practice, the algorithms running on the CPU and on the GPU can be executed in parallel by using a multi-threaded implementation. As such the next frame can already be processed by the plane sweep, while the graph-cut regularisation still runs on the previous one. Fig. 1 shows the consecutive steps of the pipeline from an algorithmic point of view, ignoring the parallelism due to the multi-threading.

3. Foreground segmentation

The implementation of the segmentation algorithm is based on a technique for illumination-invariant change detection in a sequence of images, based upon an appropriate model and a statistical decision criterion [19]. A descriptor vector x is defined for a 3×3 neighbourhood of every image pixel, by concatenating the 27 colour values (3 channels for 9 pixels). During on-line segmentation, for every pixel p_i this vector x_i in the current frame is compared to the corresponding vector b_i of a reference background by checking for their collinearity. Fig. 2 illustrates this principle in 2D. It was shown that minimisation of $d^2 = (d_1^2 + d_2^2)$ is a more robust test for collinearity than evaluating the angle [19]. This problem corresponds to solving a 2×2 matrix eigenvalue decomposition, which can be implemented efficiently. The dependence of the distance d on the length of the vector, reduces its values for darker objects, however. Therefore dark objects will be considered background too easily. Division by the distance for the vectors under 45 deg. from each other, compensates for this.