Image and Vision Computing 28 (2010) 1386–1395

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

From rendering to tracking point-based 3D models

Christophe Dehais *, Géraldine Morin, Vincent Charvillat

IRIT–ENSEEIHT, 2 rue Charles Camichel, B.P. 7122, 31071 Toulouse Cedex, France

ARTICLE INFO

Article history:
Received 10 September 2008
Received in revised form 8 February 2010
Accepted 2 March 2010

Keywords:
Visual tracking
Point-based model
Surface splatting
GPGPU

ABSTRACT

This paper adds to the abundant visual tracking literature with two main contributions. First, we illustrate the interest of using Graphic Processing Units (GPU) to support efficient implementations of computer vision algorithms, and secondly, we introduce the use of point-based 3D models as a shape prior for real-time 3D tracking with a monocular camera.

The joint use of point-based 3D models together with GPU allows to adapt and simplify an existing tracking algorithm originally designed for triangular meshes. Point-based models are of particular interest in this context, because they are the direct output of most laser scanners.

We show that state-of-the-art techniques developed for point-based rendering can be used to compute in real-time intermediate values required for visual tracking. In particular, apparent motion predictors at each pixel are computed in parallel, and novel views of the tracked object are generated online to help wide-baseline matching. Both computations derive from the same general surface splatting technique which we implement, along with other low-level vision tasks, on the GPU, leading to a real-time tracking algorithm.

1. Introduction

Low-level vision tasks such as extraction of visual features in images consume a large portion of the overall computation time of 3D visual tracking algorithms compared to higher level tasks such as pose estimation. For real-time applications, this is an area of focus for increasing performance. Faster algorithms are not always available and might not behave as well in real-life scenarios. Another possibility is to exploit the computational capabilities of modern Graphic Processing Units (GPU) which are nowadays common even in the embedded computing realm. On workstations and desktop systems, they often outperform the main CPU.

This paper presents a real-time visual tracking framework which makes a heavy use of the GPU at nearly all stages of computation. Acknowledging the advantages of the GPU for vision tasks, we are able to propose the use of dense point-based models (PBM) to represent the tracked object. While PBM and related GPU-based techniques have been quite well studied by the computer graphics community [1], to our knowledge they have never been used in the context of visual tracking.

Commonly used object models are based on 3D meshes that naturally lead to edge-based tracking techniques. In his pioneering work, Lowe [2] extracts image lines that are matched and fitted to those of the model. One can avoid prior edges extraction by actively searching for strong image gradients along normals of the projected model edges [3] from a limited set of control points sampled along the edges [4]. These techniques are very well suited for industrial objects that exhibit strong straight edges, but may fail for natural or smooth objects.

To enrich the model with object appearance information, a common approach is to apply textures onto the model surface [5]. The popular active appearance models [6] use a compact texture representation derived from the analysis of the dense object appearance in a (possibly large) number of poses. Some objects (like faces) do not show strong texture information everywhere on their surface. Sparse texture representations using for example interest points help to address this issue. A representative technique is the one of Vacchetti et al. [7]; our work is strongly inspired by their tracking algorithm.

Bringing further the idea of models based on sparse salient features yields to collections of visual features without explicit topology. The hyperpatches of Wiles et al. [8] are attached to 3D points and centered on a priori selected salient features in the image. Rothganger et al. [9] successfully applied a similar 3D object representation using texture descriptors to object recognition. Munoz et al. [10] use a set of small planar textured patches along with shape and texture bases that support the deformations and the appearance changes of the model.

We use a model similar in nature, made of a collection of unconnected points. However, instead of carefully located features, we use a set of unconnected points to model the object to track. Such point clouds are of particular interest since they are the typical

* Corresponding author. Tel.: +33 5 61 58 83 78; fax: +33 5 61 58 83 06.
E-mail address: christophe.dehais@enseeiht.fr (C. Dehais).
URL: http://dehais.perso.enseeiht.fr/ (C. Dehais).

0262-8856/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2010.03.001

output of most 3D scanning devices and the initial structure recovered by most vision-based reconstruction techniques. Of course, dense point clouds introduce a somewhat bigger volume of data than standard approaches, but this concern is reduced by the fact that, as we show in this paper, most of the treatment is offloaded to the GPU. The intermediate values required by our tracking algorithm cannot, to date, be performed on a standard CPU. Since these computations are highly parallelisable, the GPU is well adapted to perform these computations.

Along with the introduction of point-based models (PBM) as a representation of interest for computer vision problems in general and for visual tracking in particular, this paper presents two main contributions. First, we adapt state-of-the-art techniques developed for point-based rendering to compute very efficiently intermediate values required by the tracking algorithm. In particular, apparent motion predictors are computed in parallel and at each pixel by the GPU. Second, we directly use our PBM rendering algorithm to generate novel views of the tracked object as necessary in order to avoid tracking drift. As a "side-effect", we come up with an elegant and somewhat simpler algorithm compared to traditional tracking approaches.

The next section lays the basis of this work. We first informally identify a few good properties of point-based models (Section 2.1). We then introduce the visual tracking framework we build upon (Section 2.2) and finally present the splatting algorithm used for rendering and computing attributes on the PBM (Section 2.3). Section 3 presents the extension of this algorithm to the computation of dense apparent motion predictors. Section 4 details our GPU implementations, including those of low-level tasks. Section 5 presents a qualitative and performance evaluation of the proposed algorithm. Section 6 concludes the paper.

2. GPU-supported point-based models for computer vision

A 3D model of the object of interest is often helpful in many computer vision tasks such as object recognition [9], object tracking [11,7] and visual servoing [12]. In face tracking, many face models have been devised: a cylinder, an ellipsoid, a generic parameterized face model, etc. Thus, a natural question arises: what makes a good object model? Task-related and specific constraints notwithstanding, a set of good properties can be pointed out.

2.1. What is a good 3D model?

We believe a *good model* should verify as much as possible the following four properties. First, it should be easy to acquire, ideally output directly by a 3D digitization device or reconstruction algorithm. Second, it should be easily manipulated in 3D, e.g. transformed, simplified or combined. Third, computer vision problems often mandate easy access to the model from image space, in particular visibility computation, back-projection of image features and synthesis of novel views. Fourth, the model should be as general as possible and hold, along with 3D coordinates, any other geometric or photometric attributes relevant to the application (e.g. surface color and normal, deformation fields, saliency descriptors).

Some of these properties can actually conflict. A compact parametric model is lightweight and easily processed in 3D space, yet is not generally easy to acquire and extend with attributes. The often used 3D meshes are known to be tedious to acquire from inherently point-oriented raw data. Computing a motion model at any point of the projection of a 3D mesh involves a lot of indirection levels: from a given image pixel to the facet, from the facet to the vertices, where the normals are stored, and finally to the 2D motion induced by the facet plane. The increasing ubiquity of GPUs makes an alternative solution possible. Instead of sparse facet-based models, dense unstructured PBM can be considered and indeed satisfy the above-mentioned properties thanks to recent works from the computer graphics community [1]. The model creation is simplified because the output of a laser scanner or of a Structure-from-Motion algorithm can be used with minimal processing. Both 3D and 2D manipulations can be GPU assisted as shown in Section 2.3. In addition, PBM are expandable: along with 3D points, our models include static attributes such as color and normal and dynamic, motion-related attributes (see Section 3).

This work argues in favor of PBM supported by GPU implementations for computer vision. We think that these models provide an elegant answer to the following general issues:

- dense computation in image space of various model attributes, such as normals, or colors and
- dense generation of intermediate pose-dependent values necessary to vision algorithms.

In the following of this paper, we show how we apply the approach described in this section to the model-based visual tracking problem.

2.2. 3D visual tracking using a point-based model

The proposed tracking algorithm implements the approach introduced in the previous section and follows the framework initially designed by Vacchetti et al. [7], which combines iterative and keyframe-based tracking.

2.2.1. Iterative tracking

The rigid 3D motion between two successive images I_{t-1} and I_t is described by a 3D euclidean transformation, made of three elementary rotations and a translation, and modeled by a 4×4 matrix δE. Hence $E_t = \delta E\, E_{t-1}$. We use the standard pinhole camera model with 3×4 projection matrix $P_t = KE_t$, where K is the (known) constant matrix of intrinsic parameters.

Tracking the object iteratively means updating the current pose estimation knowing the previous pose E_{t-1} and a set of motion measurements made on the successive images. In this work, we track feature points detected with the Harris criterion [13] and matched using a standard correlation-based technique. Those steps are performed on the GPU as detailed in Section 4.2. Let $\mathbf{m}_{t-1}^i$ be a feature point detected on image I_{t-1}. If this point lies on the object image, then $\mathbf{m}_{t-1}^i$ is the image of a 3D point $\mathbf{M}^i$ on the object surface. The iterative tracking problem can be formulated as

$$\widehat{\delta E} = \underset{\delta E}{\operatorname{argmin}} \sum_{i=1}^{k} \left\| \Psi_{\mathbf{M}^i}\left(\delta E, \mathbf{m}_{t-1}^i\right) - \mathbf{m}_t^i \right\|^2, \quad (1)$$

where $\Psi_{\mathbf{M}^i}$ is the apparent motion model that relates $\mathbf{m}_{t-1}^i$ to $\mathbf{m}_t^i$ and depends on $\mathbf{M}^i$. It can also be seen as a motion predictor in the sense that it predicts the next position of $\mathbf{m}_i^{t-1}$ given a small 3D displacement. Lepetit and Fua [14] surveys various formulation for $\Psi_{\mathbf{M}^i}$ depending on both image measurements (optical flow, image gradient, feature points, etc.) and underlying 3D models.

2.2.2. Adaptation to PBM

At this point, an efficient 3D model for tracking will allow the easy interpretation of *any* 2D–2D correspondence $\mathbf{m}_{t-1}^i \leftrightarrow \mathbf{m}_t^i$ in terms of a 3D pose update. Those correspondences can come for example from feature point matching or from optical flow estimation. We should have for any feature $\mathbf{m}_{t-1}^i$ a motion model $\Psi_{\mathbf{M}^i}$ induced by the object model.