Real-time scene flow using a depth sensor

Mohamed Chaïk Bakkay, Walid Barhoumi and Ezzeddine Zagrouba

Research Team on Intelligent Systems in Imaging and Artificial Vision (STIVA), RIADI Laboratory
ISI, 2 Rue Abou Rayhan Bayrouni, 2080 Ariana, Tunisia
med.chaik@hotmail.fr, walid.barhoumi@laposte.net, ezzeddine.zagrouba@fsm.rnu.tn

Abstract. Scene flow is the 3D motion field of the scene. It is important for many applications such as object segmentation, obstacle avoidance and analysis of dynamic scenes. In this paper, we present an automated method for real-time scene flow estimation using Microsoft’s depth sensor “Kinect”. The main contribution of the proposed method is that the estimation is fast while avoiding over-smoothing objects boundaries, error accumulation and sensor noise. In particular, while filtering the motion in depth direction, we avoid that depth values tend to be unstable and inaccurate on objects boundaries.

Key words Kinect, scene flow, RGB-D camera, depth-map.

1 Introduction

Scene flow is the 3D motion field of the scene. Although it is more difficult to estimate than optical flow [9], it is used in many applications such as object segmentation, obstacle avoidance, analysis of dynamic scenes, tracking, autonomous robot navigation and virtual reality. Many methods were proposed to estimate scene flow. However, most of these methods perform an optimization on a global energy function which is generally slow. Moreover, the used smoothness constraint by these methods can lead to over-smooth the discontinuities (such as object boundaries) in the estimated scene flow. Also, most of these methods suffer from the accumulation of errors. Furthermore, few methods use modern depth sensors to achieve real-time performance.

The method proposed in this paper uses Microsoft’s depth sensor “Kinect” to provide real-time and accurate scene flow. It avoids over-smoothing of structure and motion by using Farneback’s optical flow algorithm [3]. However, the depth-map provided by Kinect may contain large areas of invalid values and the edges in this map are not always stable. In fact, there is a systematic problem common to all structured-light approaches which use a camera offset: It is not possible to estimate the depth in regions where the projected pattern is shadowed by foreground objects. Thus, the dense depth-map is interpolated from discrete values measured at the positions of projected point patterns. This interpolation makes depth values tend to be unstable and inaccurate on objects boundaries. The proposed method avoids this problem by filtering the motion in depth direction.

The rest of this paper is structured as follows. The section 2 is devoted to a brief summary of the most relevant works on scene flow estimation. In Section 3, we present the proposed method. Then some experimental results are presented in section 4 to demonstrate the effectiveness of our method. Finally, section 5 concludes the paper and presents some directions for future works.
2 Related works

Scene flow is often estimated using an optimization on a global energy function that includes brightness constancy constraint and some regularization constraints. Additional elements can be added to the energy function such as camera extrinsic parameters estimation [7] and constraints on stereo matching [12]. This optimization is generally slow because of its high computational cost. Thus, in order to achieve fast performance, authors in [1] used a GPU implementation. In addition to the slowness of the previous mentioned methods, the used regularization causes over-smoothing of object boundaries in both structure and motion estimates. This effect can be reduced by segmenting the input images and applying smoothness constraints only within segments [8]. However, the segmentation step needs an additional computational cost. Moreover, the brightness constancy matching will be less accurate due to projective distortions of segments during the matching process. In [10], authors formulated the problem as a point cloud in the 3D space. This formulation allows the system to easily support any number of cameras. Nevertheless, it makes smoothness constraints less applicable. Another 3D formulations based on meshes instead of point cloud are presented in [6, 11, 4]. However, these formulations reduce the application cases number of the generated motion and structure. Devernay et al. [2] estimated sparse motion fields instead of the dense scene flow estimation which is reached by most proposed methods. This motion is obtained from tracking surfsels. The sparse estimation is more precise than the dense one. However, it has less coverage of the represented scene.

On the other hand, only few works [9, 5] uses direct depth sensors to make the estimation of the scene flow faster. Hadfield et al. [9] presented a novel formulation for scene flow estimation in which a collection of moving points in 3D space is modeled using a particle filter that supports multiple hypotheses. This filter does not over-smooth the motion field. In [9], depth-map, which contains unstable and inaccurate values on objects boundaries, is directly used. This will reduce the accuracy of the obtained scene flow. Authors in [5], resolve this problem by eliminating boundary value from the depth map. However, this makes the estimated scene flow incomplete.

3 Proposed method

In this section we present the different steps of the proposed method for scene flow estimate beginning from calibration of the Kinect sensor to finally compute motion in depth direction.

3.1 Calibration

It is essential that the color and depth image information at a certain location belong to the same object point. Thus, a stereo-calibration of the two (RGB and depth) images of the Kinect sensor is necessary, in a first step, in order to find the intrinsic and extrinsic parameters, which will be used for data alignment. This calibration step begins with the acquisition of a set of images from a calibration chessboard. These images are taken from different viewpoints by the two cameras in the same lighting conditions. The parameters of the used Kinect are estimated as follows (1):

\[
K_{\text{RGB}} = \begin{pmatrix} 533 & 0 & 320 \\ 0 & 533 & 240 \\ 0 & 0 & 1 \end{pmatrix}, \quad K_m = \begin{pmatrix} 533 & 0 & 320 \\ 0 & 533 & 240 \\ 0 & 0 & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 0.999 & -0.019 & 0.015 \\ 0.020 & 0.999 & 0.004 \\ -0.011 & -0.002 & 0.999 \end{pmatrix} \quad \text{and} \quad T = \begin{pmatrix} -3.12 \\ -0.43 \\ 0.44 \end{pmatrix}.
\]

The rotation matrix \( R \) is practically equal to the identity matrix which is expected, since the axes of the two cameras are parallels (no rotation). \( T \) is the translation vector. \( K_{\text{RGB}} \) and \( K_m \) are the intrinsic matrix of RGB and depth cameras.
3.2 Data Alignment

In the second step, the previously calculated calibration parameters are used to rectify the RGB and the depth images of the two cameras (Fig. 1.a, Fig. 1.b). Once the images are rectified, homologous pixels representing the same physical point are computed by triangulation to find its 3D point cloud. In the case of our RGB-D camera, this is done using the baseline $b$ between the infrared (IR) emitter and IR camera sensor.

(a) 
(b) 
(c)

Figure 1: Optical flow estimation: a) depth image, b) RGB image, c) optical flow.

3.3 Optical flow estimation

The Gunnar Farneback [3] dense optical flow algorithm was used, as implemented in OpenCV, in order to estimate accurately and rapidly the optical flow. This algorithm uses polynomial expansion to approximate the neighbors of a pixel. The expansion could be seen as a quadratic equation with matrices and vectors as variables and coefficients. The proposed method solves the over-smoothness problem by combining the algorithm with a simultaneous segmentation procedure [3]. Fig. 1.c shows the resulting dense optical flow (sampled by 16 pixels only for visualization).

3.4 Estimation of the motion in depth direction

Finally, motion in depth direction is estimated, using the obtained optical flow, as the difference between the two depth values of each two homologous pixels in two consecutive RGB images. Then, this motion is filtered using a threshold representing the maximal speed of the human body. This speed is generally around 20m/s and the frequency of color image and depth data of Kinect is about 30 frames/s. Thus, the threshold used between two consecutive frames is $20/30=0.6m$. This filtering step reduces the effects of Kinect noise. Another advantage of the proposed method is that the depth map, which is often noisy, is not considered during the computation of the motion in $x$ and $y$ directions. Also, motion obtained in each frame is not used to initialize the motion of the next frame, what will avoid the accumulation of errors.

4 Results

In this section, we present qualitative results of the proposed method. Unfortunately, there is no publicly available database with ground truth range flow fields for Kinect data. Hence, no generally agreed and methodology for quantitatively analyzing our method is available. A moving hand sequence has been recorded from a Microsoft Kinect. The motion field (Fig. 2) shows the velocity
estimated at each structure point, as a flow line. The background in the scene is stationary, and has therefore little estimated motion in any direction. Plausible motion estimates can be seen on hands. Also, the method is able to operate fast without parallelization (5 frames/s on a standard machine).

![Figure 2: Estimated dense scene flow.](image)

## 5 Conclusion

In this paper, a real-time method for 3D motion estimation was proposed and provides comparable performance to the current state-of-the-art methods. The proposed method is also capable of making use of modern depth sensor technology such as the Kinect™, rather than relying on stereo matching algorithms which are generally slow. In particular, proposed method avoids the noise of the depth maps of the Kinect using a filtering step. We propose in future works to ameliorate the depth of Kinect using Kalman filtering and to use the estimated scene flow for real-time segmentation.

## References


