Realtime fusion of depth maps with a film camera in a popular game engine framework for pre-visualization compositing

Timothée de Goussencourt, Pascal Bertolino

REALTIME FUSION OF DEPTH MAPS WITH A FILM CAMERA IN A POPULAR GAME ENGINE FRAMEWORK FOR PRE-VISUALIZATION COMPOSITING

Timothée de Goussencourt

SolidAnim,
GIPSA-lab, Grenoble Alpes University

Pascal Bertolino *

GIPSA-lab, Grenoble Alpes University

ABSTRACT

The global context of our work is the virtual production film industry. We present an efficient framework to merge a low resolution depth map sensor with a high resolution film camera. The depth sensor used is a Kinect 2, based on time of flight technology. Our method is especially designed for film production requiring live (pre) visualization. To achieve real-time performance we are not using a specific customized solution software but the very popular game engine *Unity 3D*. Our method is directly implemented into this game engine to give the user all the facilities of a traditional game engine.

Index Terms— Depth map, Real-time, Compositing, Virtual Production, Previz on-set, *Unity*

1. INTRODUCTION

The main goal of this article is to present an efficient fusion method of a low resolution depth sensor with a high resolution film camera. Adding depth information to color frame is a huge benefit for film industry, especially for virtual production where real contents need to be mixed with digital assets (compositing). Different techniques that achieve this goal are presented in [1].

Some techniques to mix high definition color frames with depth sensors have already been developed. For instance [2] uses a trifocal rig to generate high precision depth map: The authors combine a stereo system with a monocular depth sensor, similar to [3, 4, 5, 6]. Then, a complex global optimization workflow is needed to merge data. [7] uses a graph-cut optimization whereas [8] formulates a convex optimization problem to make depth image upsampling. All of these methods use custom rig. The *Arri* company develops a prototype coupling professional video sensor with a ToF depth sensor called *Arri Alexa Scene* (`http://www.arri.com/news/prototype-motion-scene-camera/`). Conversely, the method we propose can deal with every film camera.

Our method calibrates sensors in a classic way and remaps the depth data into the main film camera referential. A depth comparison between real and virtual objects of the scene is performed to compose the final view. An existing custom implementation is described [9]. [10] is also an interesting work using a depth sensor to produce mixing reality. Our main contribution is to demonstrate the use of a very popular game engine, *Unity 3D* (`http://unity3d.com/public-relations`) as a live previewing tool. With the later, scene edition is available in real-time and the user can benefit of all the advanced features available in a game engine to really focus on creativity for mixing virtual with reality.

This paper is outlined as follows: in 2 we discuss about important choices we made for our system and compare them with relative work on this field. Section 3 describes the main points of our method and how they are technically implemented into the game engine. The paper will be concluded with an experiment of our method in 4.

*The authors thank the French FUI for supporting the PREVIZ project.

2. BACKGROUND

A prerequisite of our method is the depth information of the real scene. In the last years, several techniques have been developed to achieve this goal. Multicamera system around the scene reconstruct a 3D volume of it. This approach is really effective but requires an important setup. Other approaches based on stereo could recover the depth of the scene but are limited by the amount of texture details available on the scene. To handle this limitation, some active stereo techniques have been developed based on structured light. The first generation of Kinect is based on this technique [11]. The use of an infrared pattern also has the advantage not to disturb the visible spectral domain of the scene. [12] builds a custom system mixing stereo vision with an infrared pattern projected to the scene. Another technique using infrared is the time of flight (ToF) technology in which the time that it takes for an infrared ray to hit an object of the scene and go back to the sensor is measured, yielding the corresponding depth. The second version of the Kinect we are using in this work uses this technology.

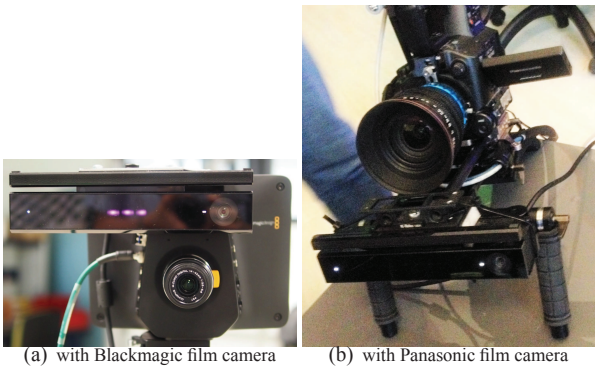
So the Kinect 2 choice is a compromise in terms of price and quality. Moreover Kinect 2 includes other interesting features like human pose recognition [13]. This sensor delivers a 512×424 pixels resolution, 16 bits per pixel, real-time depth map. The color sensor available on the Kinect 2 is not used for our experiment. So our current system is designed for indoor small scale scenes, in a controlled environment which may correspond to an important part of virtual production needs.

The Kinect is rigidly fixed to a high resolution color sensor with its lens (film camera) as shown in figure 1.

We make the choice to integrate our fusion method directly into the popular game engine *Unity 3D*. This platform was chosen because of its great performance, versatility and huge developers community.

3. METHOD

Our method is divided into two main blocks. First a classical offline process retrieves the intrinsics parameters of each camera. Then an extrinsic calibration is done to compute relative position of the depth sensor to the film camera. Second an online process is performed to back-project in real-time the 3D information from the depth sensor into the main camera screen space for each frame. Finally a compositing step mixes virtual and real assets. These two blocks are



(a) with Blackmagic film camera (b) with Panasonic film camera

Fig. 1. Our system using different film cameras

represented in figure 2. Each step of these blocks is presented in the sequel.

3.1. Calibration

One of the most important aspect of this application is the internal and relative calibration of the color camera and the depth camera. We consider the two sensors as classical perspective camera. The first step of our method is to retrieve intrinsics and distortion parameters of each camera sensor. We are using radial and tangential distortion models described in [14, 15] and a specific distortion model for fish-eye lenses described in [16]. A well-known technique for calibrating perspective camera is based on chessboard pattern recognition [17, 18]. This calibration is independently done for each camera, in color space for the film camera and in infrared for the depth sensor. This leads to a projection matrix M for intrinsics parameters (**??**) and a vector of distortion parameters for each camera. Whereas distortion for film camera completely depends of the used lens it appears that Kinect 2 is very few distorted.

The calibration algorithm also gives for each chessboard view the pose of the chessboard relative to the camera system coordinates. Combining the relative pose of each camera to the chessboard makes it possible to compute for each view an estimation of the rotation R and translation T between the color camera and the depth camera. We take several captures of the scene with a chessboard at different depths, simultaneously with the depth and the color camera. Each couple of images is processed and the measurement of the current pose is added to a stack. Finally an overall adjustment is performed using a robust Levenberg-Marquardt iterative algorithm to obtain an optimal extrinsic calibration [19].

It is possible to retrieve intrinsics and extrinsic parameters in the same time but it is sometimes more appropriate to use two different sets of chessboards views. Indeed the covered field of view may differ between sensors. It is better to maximize the pattern size in the camera view to compute intrinsics and distortion parameters, whereas we need to see the same chessboard entirely from each sensor point of view for extrinsic calibration.

This calibration is computed only once at the beginning, all calibration parameters are saved and loaded by the online process.

3.2. Depth map reprojection into the film camera

Each step of the online process is implemented in Unity. Everything in this game engine relies on the concept of gameobject. In our case

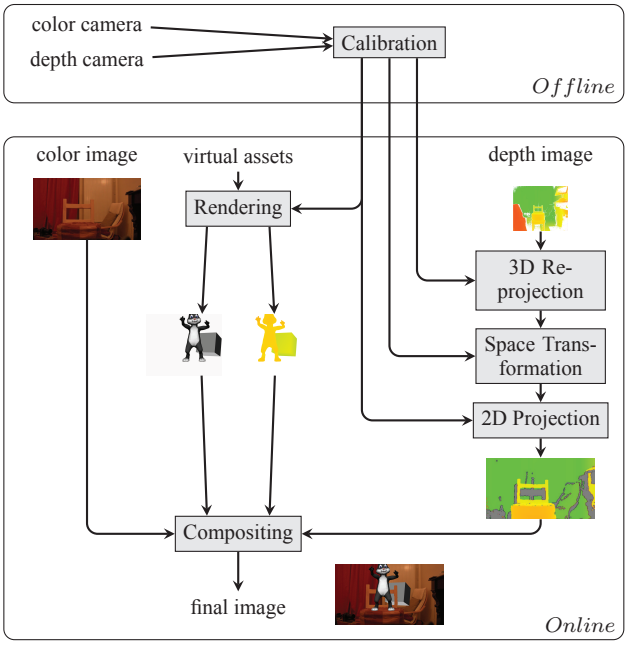


Fig. 2. Overall system workflow. Offline process is executed once at the beginning. Online process is executed in real-time for each frame.

the film and depth camera are represented by two camera gameobjects. The depth camera gameobject is parented to the film camera gameobject. The extrinsic calibration file computed during the offline process is loaded to correctly pose the depth camera object into the film camera object coordinates space.

Then a mesh gameobject is parented to the depth camera object. This mesh is a flat grid with the dimension of the depth map with $w \times h$ triangulated vertices. A $2D$ vector uv is generated for each vertex to correctly associate its corresponding depth texture pixel. The GPU computation of the $3D$ scene is realized with a shader. For each vertex passed to the vertex shader, we look up into the depth texture according to the uv coordinates to retrieve the associated depth data, $z = tex(uv)$. Using the depth camera inverse projective matrix M^{-1} it is possible to compute the current $3D$ vertex position P_{3D} as detailed in equ. 1. Then a geometry shader discards the triangle primitives where no depth information is present, to only display valid data. Finally a fragment shader is used to encode the interpolated depth value of the scene for each pixel. This way a $3D$ representation of the scene is computed in real-time using GPU capabilities. Figure 3 gives an illustration of the method.

$$P_{3D} = M^{-1} \cdot \begin{pmatrix} uv.x \cdot tex(uv) \\ uv.y \cdot tex(uv) \\ tex(uv) \end{pmatrix} \quad (1)$$

Considering a grid mesh more than a list of $3D$ points has the advantage to observe a dense surface. This $3D$ mesh is now observed according to the film camera point of view using the projection matrix of the color camera. Re-projection of the low resolution depth map becomes a dense interpolation fiting the resolution of the film camera. This provides a registered depth texture of the real scene.

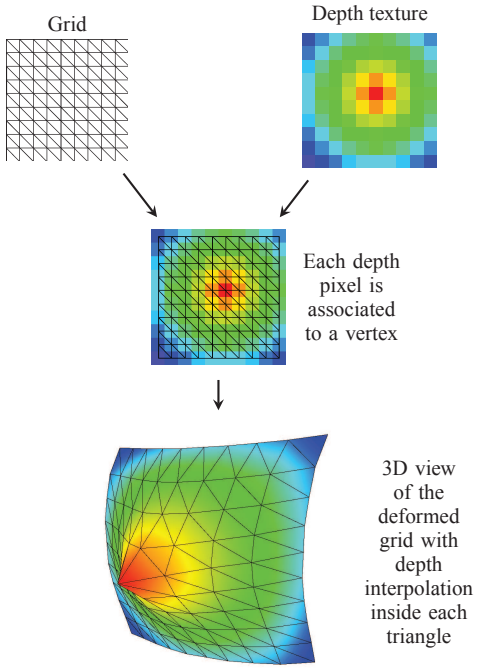


Fig. 3. 3D mesh reprojection method

3.3. Compositing

Then it is possible to compare the real scene depth map with the virtual scene depth map to generate a binary mask coding whether a pixel of the final rendering comes from the virtual asset frame or from the film camera frame. This way, original color image could be associated with its registered and distort depth image, making possible some compositing whatever occlusion between the virtual and real worlds.

4. EXPERIMENTS

Our system was tested with two different film cameras used traditionally for broadcast applications (see Figure 1). A live video stream is transferred from the camera to the computer with an SDI to USB3 acquisition card. The computer is a laptop with an Intel i7 2.3GHz processor, 16Go RAM and a Nvidia GeForce GTX 670M.

We use special asset *AVPro Live Camera* to pipe the HD color frames into *Unity*. Moreover *Microsoft* provides *Unity* plugins to access Kinect 2 functionality into the game engine. For each camera a frame buffer is set up to compensate the small delay between depth and color streams.

The current framework we describe here is used in the context of the PREVIZ project [20] dedicated to virtual production, especially previz on-set. In order to enlarge the compositing scenario range, we coupled our system to the *SolidAnim* camera tracking solution (SolidTrack) via a *Unity* plugin to retrieve for each frame the correct film camera pose into the $3D$ world. Thus it was straightforward to remove real background and replace it with a virtual one without using green or blue walls or to display a virtual character or object in the scene while respecting any occlusions between real and virtual worlds. Figure 4 shows our result.

5. CONCLUSION

We presented a reliable method to align a Kinect 2 depth sensor with a professional film camera. This method has been implemented into the popular game engine *Unity* and achieves realtime performances. This system is designed for indoor small scale scene. In this work we do not deal with camera tracking, but this feature could be added by implementing state of the art tracking algorithms.

In the future, using *Unity* will allow to simply make powerful and fast process with the GPU, like improving the quality and regularity of the depth maps or managing interactions between the virtual world and a human body. Actually, real-time skeleton tracking data provided by Kinect [13] are also available in *Unity 3D*.

6. REFERENCES

[1] G. Thomas, "Mixed reality techniques for tv and their application for on-set and pre-visualization in film production," in *International Workshop on Mixed Reality Technology for Filmmaking*, 2006, pp. 31–36.

[2] G. Boisson, P. Kerbiriou, V. Drazic, O. Bureller, N. Sabater, and A. Schubert, "Fusion of kinect depth data with trifocal disparity estimation for near real-time high quality depth maps generation," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2014, pp. 90110J–90110J.

[3] J. Zhu, L. Wang, R. Yang, and J. Davis, "Fusion of time-of-flight depth and stereo for high accuracy depth maps," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[4] U. Hahne and M. Alexa, "Combining time-of-flight depth and stereo images without accurate extrinsic calibration," *International Journal of Intelligent Systems Technologies and Applications*, vol. 5, no. 3, pp. 325–333, 2008.

[5] Y. Song, C.A. Glasbey, G.W. van der Heijden, G. Polder, and J.A. Dieleman, "Combining stereo and time-of-flight images with application to automatic plant phenotyping," in *Image Analysis*, pp. 467–478. Springer, 2011.

[6] V. Gandhi, J. Cech, and R. Horaud, "High-resolution depth maps based on tof-stereo fusion," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4742–4749.

[7] S. Patra, B. Bhowmick, S. Banerjee, and P. Kalra, "High resolution point cloud generation from kinect and hd cameras using graph cut.," in *VISAPP (2)*, 2012, pp. 311–316.

[8] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof, "Image guided depth upsampling using anisotropic total generalized variation," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 993–1000.

[9] B. Bartczak, I. Schiller, C. Beder, and R. Koch, "Integration of a time-of-flight camera into a mixed reality system for handling dynamic scenes, moving viewpoints and occlusions in real-time," in *Proceedings of the 3DPVT Workshop, Atlanta, GA, USA (June 2008)*, 2008.

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al., "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 559–568.

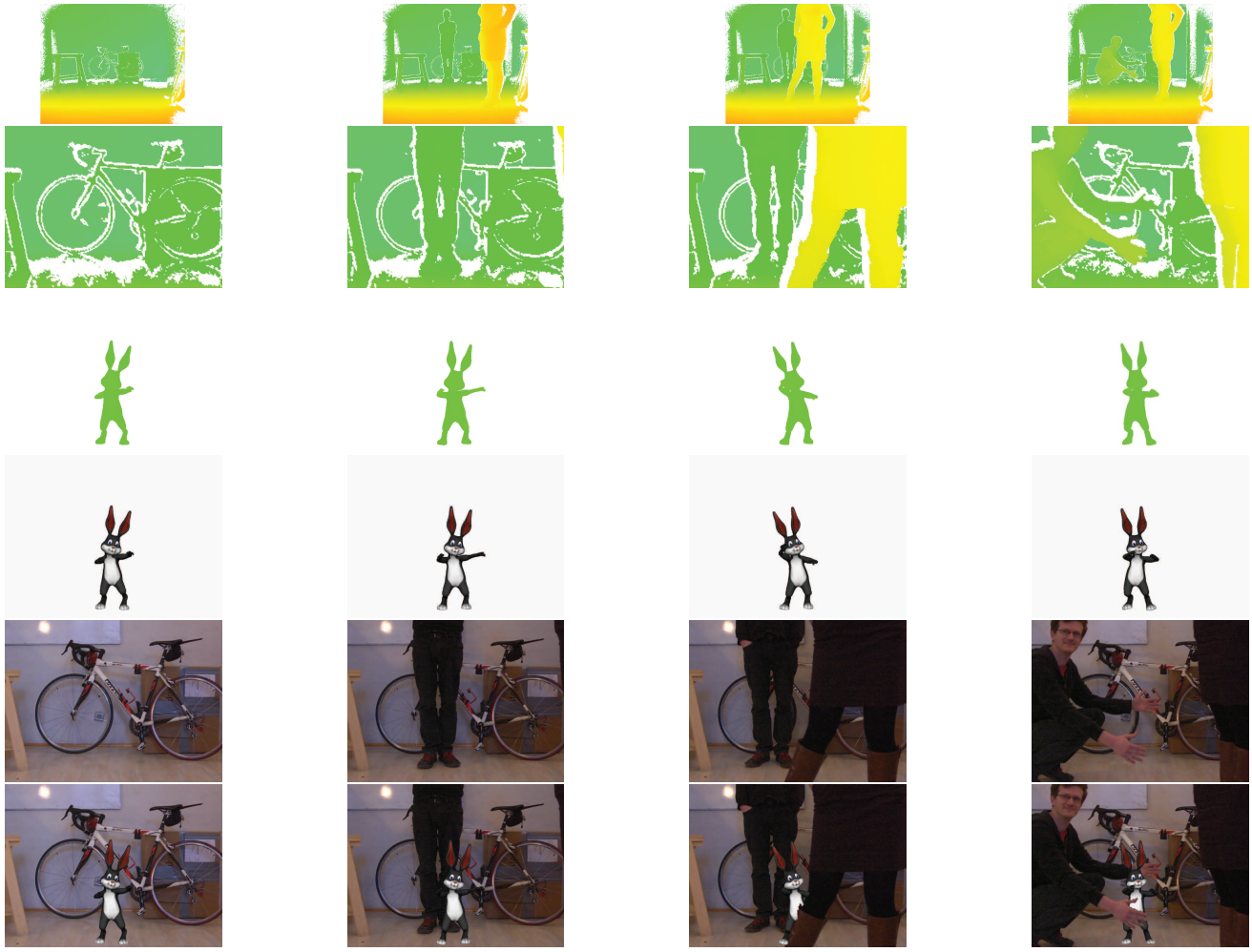


Fig. 4. Examples of compositing. First row: depth map. Second row: registered depth map. Third row: virtual characters depth map. fourth row: virtual character frame. Fifth row: film frame. Sixth row: final compositing

[11] A. Shpunt and Z. Zalevsky, "Three-dimensional sensing using speckle patterns," Mar. 5 2013, US Patent 8,390,821.

[12] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, et al., "Real-time non-rigid reconstruction using an rgb-d camera," *ACM Transactions on Graphics, TOG*, vol. 4, 2014.

[13] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[14] D.C. Brown, "Close-range camera calibration," *Photogramm. Eng*, vol. 37, pp. 855–866, 1971.

[15] J.G. Fryer and D.C. Brown, "Lens distortion for close-range photogrammetry," *Photogrammetric engineering and remote sensing*, vol. 52, no. 1, pp. 51–58, 1986.

[16] F. Devernay and O. Faugeras, "Straight lines have to be straight," *Machine vision and applications*, vol. 13, no. 1, pp. 14–24, 2001.

[17] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. IEEE, 1999, vol. 1, pp. 666–673.

[18] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.

[19] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*, " O'Reilly Media, Inc.", 2008.

[20] G. Briand, F. Bidgolirad, J.F. Zlapka, J.M Lavalou, M. Lanouiller, M. Christie, J. Lvoff, P. Bertolino, and E. Guillou, "On-set previsualization for vfx film production," in *International Broadcasting Convention (IBC)*, Amsterdam, Netherland, 2014.