METHOD FOR ENHANCING LOW QUALITY DEPTH MAPS FOR 3D RECONSTRUCTION ON A EMBEDDED PLATFORM

Rajesh Narasimha *Karthik Raghuram* *Jesse Villarreal* *Joel Pacheco*

Texas Instruments Inc, 12500 TI Blvd, Dallas TX 75243

ABSTRACT

In the recent past, vast amounts of stereo and augmented reality based applications are being developed for hand-held devices. In most of these applications depth map is a key ingredient for acceptable user experience. Accuracy and high density of depth map are important along with meeting real-time constraints on an embedded system. There is an inherent tradeoff between depth map quality and speed and invariably performance is usually important for competing in todays high-definition video marketplace. In this paper we present a method that addresses depth map quality while still maintaining performance at video frame-rates. Specifically, we discuss a technique to enhance a low-quality depth map for 3D point cloud generation on an embedded platform. We provide performance metrics and estimates on a Texas Instruments (TI) OMAP embedded platform and show that using simple pre and post-processing techniques one can achieve both quality and performance. A preliminary version of our point cloud application developed has a frame rate of about 15fps, majority being display and rendering related overheads. The core algorithms including pre and post processing have a much higher frame rate of about 23-25fps. We estimate that with adequate mapping of the algorithms to various cores and accelerated kernels, the frame rate could reach real-time performance of 30fps.

Index Terms— Embedded Platform, depth quality, TI OMAP, multi-core, color/contrast match, 3D point cloud

1. INTRODUCTION

Depth estimation from multiple views is well studied and understood [1]. A commonly employed approach is to use a calibrated stereo camera. Several techniques exist to compute disparity and we consider one that has been implemented on the OMAP embedded processor [2]. The goal of the presented algorithm is to enhance the quality of the computed disparity map so that it could be used for applications such as gesture recognition, segmentation/tracking or generate a 3D point cloud of the scene. We shall not be concerned with methods for generating such disparity maps, but rather assume the presence of a simple technique for computing disparity. Commonly used methods of computing disparity can be found in [3] [4].

Given the real-time and power constraints to be met on embedded systems, along with the poor quality of the image sensors, the disparity maps generated often have regions in which the computed disparity cannot be computed reliably. Such regions may lack textures, have illumination artifacts such as shadows, and have mismatches between the stereo pair in terms of contrast and color. In this work, we describe techniques that pre-process the stereo pair to improve robustness of disparity estimation under such circumstances. Subsequent to any pre-processing, we employ a dynamic disparity range selection to improve disparity accuracy which tremendously reduces the search range and improves frame rates. In addition, we match the color and contrast in the stereo image using simple average RGB gains and a look up table (LUT) based contrast correction.

Further enhancement may still be needed after the above preprocessing to meet the quality demands of applications such as 3D point-cloud computation or gesture recognition. Lack of textures is a key reason for unreliable disparities. On the other hand, this also implies that such regions in the scene are rather smooth and hence suitable for interpolation. Through experimentation and keeping in mind the computational complexity, we interpolate the computed disparity map using a multi-resolution approach. As a second step, we perform a simple but selective linear interpolation on the generated disparity, with a simple algorithm to detect the presence of holes in the depth map. Finally, to generate a smooth disparity map, we implement a bilateral filter based on the combination of the algorithms provided by Yang et.al [5], Yu et.al [6] and Porikli [7].

The organization of the paper is as follows. Section 2.1 below describes automatic disparity range selection[1]. Section 2.2 describes a cross normalization algorithm for color and contrast in a stereo pair. Section 2.3 covers the multi-resolution interpolation approach. Section 2.4 discusses the hole identification and interpolation on the point cloud. A short description of bilateral filter is provided in section 2.5. Evaluation and performance estimates of a working prototype on TI OMAP platform is provided in section 3. Conclusions and Future are drawn in section 4.

[1] We do not discuss the conversion of disparity to depth as it is fairly standard. We use pre-calibrated tables for efficient disparity to depth conversion

2. AUTOMATIC DISPARITY SEARCH RANGE SELECTION

The block diagram of the proposed technique is shown in Figure 1. In the following sections we will describe the algorithm in detail. The computational cost of the disparity map algorithm is proportional to the disparity search range. Therefore narrowing this search range dynamically based on relatively fast object detection may lower the overall computational time of the desired disparity map calculation. For most image segmentation problems, we are only interested in a narrow disparity range, and that this disparity range may change over time because the camera, or the objects of interest may move toward or away from the camera. If there are no objects of interest found, the application may decide to either search the full available range, or move to a lower power state, depending on the application goals.

Next, we propose methods for detecting a limited search

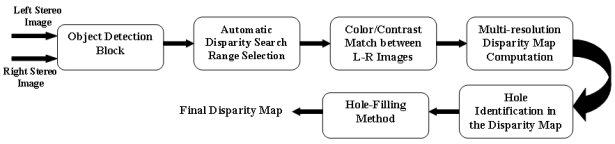

Fig. 1. Block diagram of the proposed method to enhance a low-quality depth map for 3D reconstruction

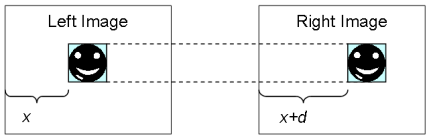

Fig. 2. Disparity computation of the object of interest

range using object detection algorithms. When the object(s) of interest in the scene can be located from each of the stereo pair images independently using object detection algorithms, then the disparity range of this object can be used to narrow the disparity search range of the disparity map computation. For example, if there is an algorithm that returns a bounding box of a face in each 2D image from the stereo pair, then the disparity of this face is calculated by simply subtracting the horizontal offset of this face in one of the images from the horizontal offset of the same face in the other image. For multiple objects in a scene we choose the nearest object and in our use case the nearest face. Figure 2 above depicts the disparity, d, of this face between the two images of a stereo pair. The scenario with multiple faces needs much more sophistication for disparity calculation and we do not discuss it here. Based on the known inverse relationship between disparity and depth, we fit the following model to parametrize the stereo system at hand. If d denotes the depth, D denotes the disparity, we find parameters a, b such that $\left\| d - (\frac{a}{D} + b) \right\|$ is minimum. With known depth measurements along the tape measure, and the corresponding disparities from the stereo pair, we setup a least-squares system and solve for a, b. Once obtained, we use the same relationship to obtain depth d from a disparity D from the stereo correspondences. Although this can be applied to any object that can be detected, the example of a face is used because face detection in digital cameras is now a mature technology, and many camera platforms have hardware accelerated face detection, such that it can be calculated virtually for free in terms of CPU cycles. In the case that the object detection can only be performed on one of the images from the stereo pair, then the disparity can still be calculated by comparing the size of the object in the image to the expected size at different distances from the camera. This can be done using a predefined equation or look up table. Once the distance is known, then this directly translates to a disparity value. Figure 3 shows an example of a distance to disparity function obtained from the TI Blaze software development platform, which features a stereoscopic camera pair on the front of the unit. The object size to distance, and distance to disparity calculations/tables can be merged into a single calculation/table, especially in the cases where there are a small number of objects being detected. When multiple objects are being detected, it may be efficient to have the distance to disparity function calculated separately. With several measurements, one may perform a least square reciprocal fit to convert disparity to depth as prescribed by the stereo optics.

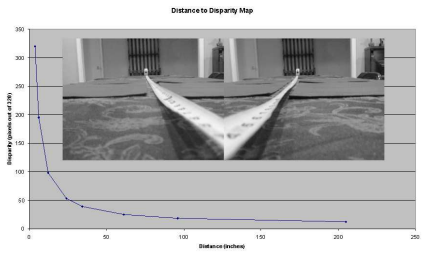

Fig. 3. Distance to Disparity Function

3. ENHANCEMENT OF LOW-QUALITY DEPTH MAP

The reference (left) and the right stereo image pairs used in our experiments are shown in Figure 4. This is the typical output from a stereo sensor on consumer 3D digital cameras and cellphone cameras. One can see a clear color and contrast mis-match between the left and right stereo pair. The results of depth map accuracy with and without disparity range selection is shown in the bottom row of Figure 4.

3.1. Color/Contrast Matching for Stereo Images

The mismatch in color and contrast between the left and right stereo pairs occurs due to low cost sensors, mismatches in the imaging pipeline or in the mechanical elements of the camera such as auto focus, aperture and auto exposure. Correcting for such mismatch will improve the depth image and thereby help in various post processing algorithms such as segmentation and view synthesis. Moreover, the algorithm is well suited for embedded platforms where usually low complex stereo matching algorithms such as sum of absolute differ-

ence or sum of squares of absolute difference based matching are used.

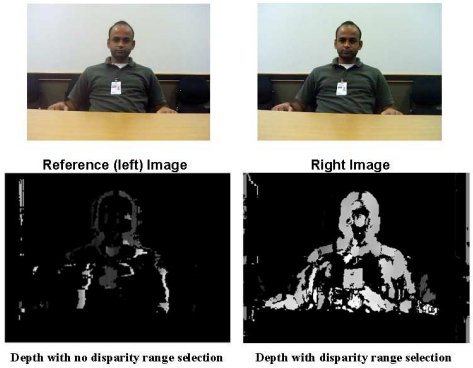

Fig. 4. (Top row) Brightness and contrast mis-match between L-R stereo pair, (Bottom row) Depth maps before and after disparity range selection

3.1.1. Color Matching

In this step, we compute the average R, G, and B of a sufficiently large area in the left image which is our reference image. For a VGA image size of $640 * 480$ we choose a block size of $64 * 64$. The intention is to match the color of the right image to the reference image. We then search for the matching area in the right image using simple sum of absolute difference (SAD) between the left and right areas and find a matching area. Once we find a matching block we compute the average R, average G and average B of the reference and the matched block. Alternately, we can compute the average R, G, B of the entire left and right image or compute local block averages and match the R, G, B values locally. To minimize the computation we compute the average R, G, B of the entire image that does not require matching. Then we compute the ratio between the right frame and the reference frame as follows,

$$\begin{aligned} \textbf{Rgain} &= Ravg_{right}/Ravg_{ref}, \quad \textbf{R} = R/Rgain \\ \textbf{Ggain} &= Gavg_{right}/Gavg_{ref}, \quad \textbf{G} = G/Ggain \\ \textbf{Bgain} &= Bavg_{right}/Bavg_{ref}, \quad \textbf{B} = B/Bgain \end{aligned} \tag{1}$$

In equation 1 we match the average R, G, B of the right image by scaling the R, G, B values by their respective gains.

3.1.2. Contrast Matching

The contrast matching algorithm is based on matching the histogram of the left and right image pair. We compute the luminance histogram of the left and right images. Here again we are matching the right luminance histogram to the left luminance histogram. In Figure 5, one can see the difference between the two histograms in terms of the brightness and contrast. We compute a mapping function that matches the right luminance histogram to the left luminance histogram in the form of a look up table (LUT). The procedure is to compute the cumulative distribution function (CDF) from the left and right luminance histograms and generate a mapping LUT to match the right CDF to the left CDF. We then check for monotonicity of the mapping LUT. The top row of Figure 5 depicts the left and right histograms mis-match that has been compensated for contrast mismatch as shown in the top right figure where the left-right histograms overlap. The results of the left/right stereo pair color and contrast matching on the density of the depth maps is shown in the bottom row of Figure 5. One can see that before matching the depth map is sparse as in the bottom right of Figure 4 where as after matching the depth map is denser which will facilitate easy foreground segmentation.

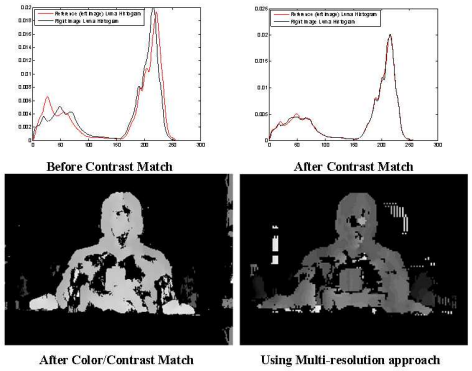

Fig. 5. (Top row) Contrast matching using histograms. (Bottom row) Left-Depth maps after left/right color and contrast matching, RightDepth using multi-resolution approach

3.2. Multi-resolution Depth Map Computation

While pre-processing is a low cost preferred approach to enhancing disparity, there may be several regions where disparity could not be reliably computed at the given resolution. In such cases, interpolation and hole-filling ideas are critical. In general, disparity computation in regions lacking textures is hard as a unique match between left and pixel neighborhoods is difficult if not impossible. However, one can use the smoothness of these regions (lack of textures) to interpolate the disparity values instead. A novel and effective interpolatory hole-filling procedure is described in what follows. We start by constructing a disparity maps using a multi-resolution approach. For example for a VGA frame (640x480) one could have disparities computed at 320x240 and 160x120 sizes. The interpolatory process now selectively fills the holes in the highest disparity image using a weighted combination of the available lower resolution disparities. Note that if no disparity is available at any lower resolution, the pixel remains a hole even after the post-processing. For purposes of demonstration, consider the stereo pair in Figure 4. The computed disparity map is as per the left figure in the bottom row of Figure 5. In addition, we also perform IIR filtering along with the interpolation process to fill in the holes over time and avoid noisy depth values.

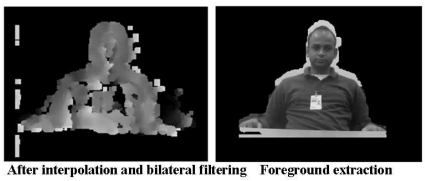

Fig. 6. (Left)Depth after linear interpolation and bilateral filtering, (Right)Foreground extraction

3.3. Hole Identification and interpolation

We describe a simple technique for identification of holes in the disparity and a linear interpolation of such regions. In the context of a point-cloud application, a dense disparity map is particularly important in the foreground region. We therefore employ adaptive thresholding of the relative depth to identify foreground regions which is a fuction of the dynamic range of the depth map. Since our depth map was 8-bit we chose a threshold of 245 to identify foreground and produce a binary map of every 8*8 block as foreground or not by counting the pixels exceeding a threshold by taking care of boundary conditions. We first perform erosion and dilation operation to fill holes that are trivial to fill in and use Bilateral Filter to fill the rest.

The advantage of the bilateral filter is that it has a range and domain filter that can be specified to obtain smooth depth maps which helps to avoid hole filling across boundaries. We implemented a combination of the algorithms proposed in [6-8]. The result of bilateral filtering along with binary morphology such as erosion and dilation is shown in left figure of Figure 6. We chose a 5x5 kernel for bilateral filter so that the decision from filling does not occur from non-similar neighbors. An example of foreground extraction is shown in the right figure of Figure 6.

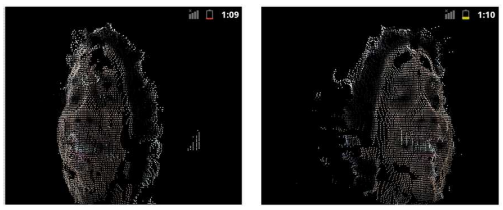

Fig. 7. Screen shots from a sample point cloud application used for 3D reconstruction

3.4. Application: 3D point Cloud Generation

One practical application for the proposed method is in the generation of a point cloud (a collection of X, Y, and Z coordinates representing the location of objects in 3D space) from a stereoscopic image pair. Using the techniques described in this paper to improve the density of generated depth maps, we can improve the accuracy and quality of the resulting point clouds, making them suitable for 3D reconstruction applications. Figure 7[2] below contains screen shots extracted from an example application Android application that recreates what is seen through a stereoscopic camera.

[2]Please note that the figure 7 is different from the torso figures used throughout the paper since it is a screen shot from the application running on the OMAP platform

4. PROTOTYPING ON TI OMAP PLATFORM

The TI OMAP platform has a dedicated camera pipe infrastructure which encapsulates several hardware acceleration features including histogram computation, image resizing and face-detection. The OMAP4430 SDP was running the ARM cores at $1GHz$, the DSP Core at $466MHz$ and the Dual Image Co-Processors at $100MHz$each. Usage of such hardware accelerated features greatly reduces the CPU load. The object detection which in the presented use case is a face and hence the use of face detection hardware does not affect the frame rate. The color matching is also free in terms of cycles since we have pre-computed block averages using the hardware. The contrast matching uses a hardware histogram module and is a low-cost LUT which again is negligible in terms of cycles per pixel. In general, the block matching based disparity estimation takes $40ms$ for a 320x240 frame for 32 disparities on the DSP along with accelerated kernels. Since we make use of the disparity search range selection we can reduce the number of disparities to 16, which is executed in $25ms$. We then use multi-resolution depth maps at 160x120 and 80x60 frame sizes which is equivalent to running 1.3125 times the original size and consumes $34ms$. The counting of foreground pixels for the purpose of identifying foreground blocks is accelerated by executing the process on a SIMD (single instruction multiple data) processor. For example, multiple rows in the block-divided depth map is processed in parallel using a DSP. To further reduce computational load, we build the foreground mask at a lower rate than the video frame rate. In doing this, the foreground mask can be reused for subsequent frames until a new mask is generated and we generate a mask for every 2 to 4 video frames. The bilateral filter, in general is an expensive operation since the range filter coefficients needs to computed dynamically. However, recent advances (see [5], Yu et.al [6] and Porikli [7]) have proposed constant time bilateral filters w.r.t to kernel size, which is suitable for acceleration on OMAP and we can parallelize the computation due to the dependency only on the kernel size.

5. CONCLUSIONS AND FUTURE WORK

We have discussed simple but effective pre and post proccesing techniques to enhance a low-quality depth map on a embedded platform in real-time. As extensions, the pre and post-processing stages may be mapped to the most suitable core, and in our preliminary studies, point to real-time frame rates of 30fps for QVGA frames. Thus, given reasonable acceleration, the pre-post processing to improve depth map quality are essentially free in terms of cycles per pixel, and the overall frame rate is dominated by the standard disparity computation algorithms at about $25 - 30$ fps.

6. REFERENCES

[1] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[2] Texas Instruments OMAP, ," *www.ti.com/omap*.

[3] Daniel Scharstein and Richard Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision*, vol. 47, no. 1-3, pp. 7–42, Apr. 2002.

[4] Myron Z. Brown, Darius Burschka, and Gregory D. Hager, "Advances in computational stereo," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 25, no. 8, pp. 993–1008, 2003.

[5] Qingxiong Yang, Kar-Han Tan, and N. Ahuja, "Real-time o(1) bilateral filtering," *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 557–564, June 2009.

[6] Wei Yu, Franz Franchetti, James C. Hoe, Yao-Jen Chang, and Tsuhan Chen, "Fast bilateral filtering by adapting block size.," *Image processing, 2010. ICIP 2010. IEEE Conference on*, pp. 3281–3284, 2010.

[7] F. Porikli, "Constant time o(1) bilateral filtering," *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008.