On Multi-view Texture Mapping of Indoor Environments using Kinect Depth Sensors

Luat Do, Lingni Ma, Egor Bondarev and Peter H. N. de With
Eindhoven University of Technology P.O. Box 513, 5600 MB, Eindhoven, The Netherlands

Keywords: Muli-view Processing, Texture Mapping, Kinect, 3D Reconstruction.

Abstract: At present, research on reconstruction and coloring of 3D models is growing rapidly due to increasing availability of low-cost 3D sensing systems. In this paper, we explore coloring of triangular mesh models with multiple color images by employing a multi-view texture mapping approach. The fusion of depth and color vision data is complicated by 3D modeling and multi-viewpoint registration inaccuracies. In addition, the large amount of camera viewpoints in our scenes requires techniques that process the depth and color vision data efficiently. Considering these difficulties, our primary objective is to generate high-quality texels that can also be rendered on a standard hardware setup using texture mapping. For this work, we have made three contributions. Our first contribution involves the application of a visibility map to efficiently identify visible faces. The second contribution is a technique to reduce ghosting artifacts based on a confidence map. The third contribution yields high-detail texels by adding the mean color and color histogram information to the sigma-outlier detector. The experimental results show that our multi-view texture mapping approach efficiently generates high-quality texels for colored 3D models, while being robust to registration errors.

1 INTRODUCTION

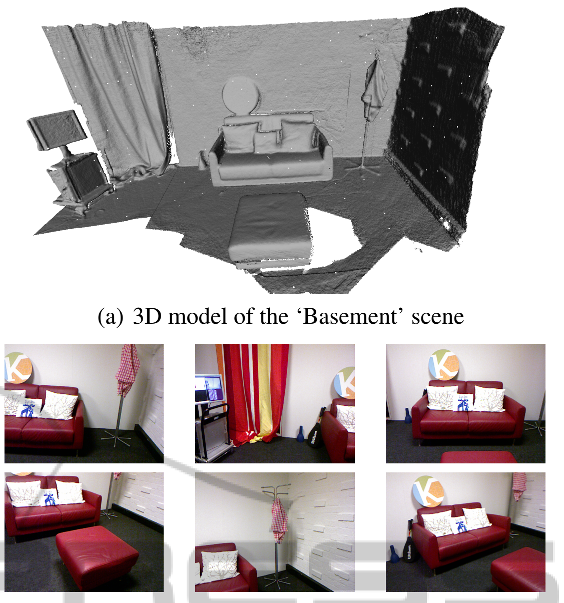
With the growing availability of low-cost depth sensing devices, 3D scanning platforms are rapidly becoming broadly accepted in the consumer market. This trend has accelerated research and development in the area of serious gaming, 3D printing and photo-realistic 3D models. The most popular 3D scanning device nowadays is the Kinect depth sensor, which can create reasonably accurate 3D models despite its low cost. To build a color 3D model of the surroundings, a common approach is to employ multiple color images that are captured at various viewpoints and map parts of these images onto the depth signal (range) generated by this sensor. This fusion of range and color vision data to provide colored 3D models is a challenging task due to various undesirable errors. First of all, textures are usually of low-quality, caused by the limited camera resolution, slight motion blur and lens distortion. Secondly, it is difficult to obtain perfect intrinsic and extrinsic camera calibration due to modeling and related algorithmic inaccuracies. Thirdly, the obtained 3D model contains modeling inaccuracies because of the limitations in sensor quality and scanning procedures.

Bearing these errors in mind, we aim at exploring a multi-view texture mapping (Heckbert, 1986) approach, where textures from various viewpoints are blended to create a photo-realistic textured model of indoor environments. We have adopted the referred texture mapping, since it is a well-known technique in computer graphics supported by hardware-accelerated processing. Specifically, this technique assigns a texture patch (*textel*) to every polygon (face) by defining a one-to-one mapping from the 2D image to the 3D (TIN) model (*mesh*). For meshes with large-sized polygons, such mapping efficiently creates textured models. Unfortunately, the separation between geometry and appearance complicates the reuse of color information for other processing tasks, such as object segmentation and classification. Therefore, other methods have been developed to obtain a colored model with more flexibility by directly associating colors to the geometry. Point-based rendering (Zwicker et al., 2001; Sainz and Pajarola, 2004) and mesh colors (Yuksel et al., 2010) are two recent techniques that associate colors to the vertices. In the case of mesh colors, also color samples are interpolated for edges and faces such that their appearance can be better presented even with large triangles. However, both techniques require much denser vertices than the referred texture mapping to yield a high-quality visualization.

At present, high-quality visualization based on texture mapping is obtained mostly for small objects and/or manually registered images from a few camera viewpoints. It is our objective to create high-quality mapping for large indoor environments with automatic camera pose estimation. In literature, multi-view texture mapping consist of two major steps. First, the visible faces are determined for each view. Then, for each face a texture patch is generated by weighted blending of the available views. The multi-view texture mapping approaches in (Rocchini et al., 1999; Rocchini et al., 2002; Alshawabkeh and Haala, 2005; Grammatikopoulos et al., 2007) selects first a master image for each face using the smallest intersection angle between the face normal and the image ray. Faces that are partly projected onto different images are blended to reduce the discontinuity in color differences of the images. The image views are manually registered to the 3D model, so that registration errors are rather small. To test the visibility of faces ray-casting (Whitted, 2005) or Z-buffering (Catmull, 1974) is used.

For 3D reconstruction in larger spaces, the literature is limited. For example, Whelan *et al* (Whelan et al., 2012) and Bondarev *et al* (Bondarev and Heredia, 2013) employ the Kinect depth sensor to scan a (large) room. In their process, color images are captured from various viewpoints (25–33 views) and automatically registered to the 3D model. For further improvements towards photo-realism, our approach need to handle many more views with larger registration errors. In addition, in contrast to rendering small objects, rendering 3D indoor scenes shows background/foreground ghosting. Furthermore, since the light conditions of the viewpoints are severely different from each other, it is not possible to employ a master image. Given these difficulties, the major issues that now need to be addressed are (1) the selection of visible faces for each viewpoint, (2) the color bleeding especially at object boundaries due to registration errors and (3) the photo-realistic blending of the multiple viewpoints. Our experimental results will show that visually appealing colored models can be obtained by employing efficient Z-buffering and applying simple outlier selection using only the geometric and color properties of the faces.

The remainder of this paper is organized as follows. Section 2 gives an overview of our off-line multi-view texture mapping approach which consists of the visibility test and the texture blending procedure. In Section 3, we describe our visibility test with Z-buffering and introduce the *confidence* map which is used as a weight during the texture blending. In Section 4, we analyze the selection procedure for each face and the texture blending process. The experimental results are visually presented in Section 5 and finally, Section 6 discusses the conclusions and recommendations for improving the multi-view texture mapping of indoor environments.


(a) 3D model of the ‘Basement’ scene

(b) Multi-view textures (6 samples out of 33 in total).

Figure 1: Example of input data (Basement) for our system.

2 OVERVIEW SYSTEM

In this work, we assume that the input of our multi-view texture mapping consists of a triangular mesh and a set of multi-view textures with calibration parameters. Fig. 1 depicts an example of an input dataset. To color the 3D model, we generate for each face an appropriate texture patch (*textel*) from the multi-view textures and store it in a larger texture image. For high-quality model generation, the textels must be such that the effects of registration errors are less visible and that the 3D textured model is contiguous and contains no seams. Since each face is processed individually and the visual registration errors are small, the smoothing between adjacent faces is not performed explicitly. Our proposed multi-view texture mapping approach is depicted in Fig. 2 and consists of three stages. In the first stage, the visible faces for each camera view are identified by exploiting a visibility map using Z-buffering. Within the same process, we create a confidence map, which serves as a weighting during texture blending. In the second stage, we select the most reliable views for

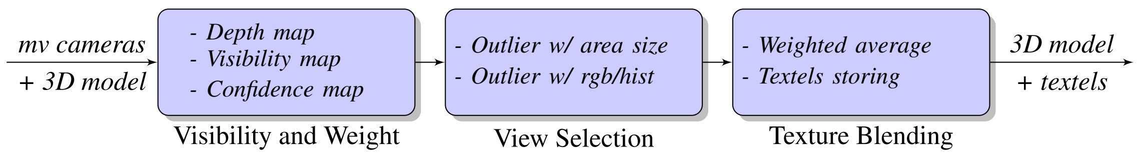


Figure 2: Multi-view texture mapping approach with processing stages and principal functions enclosed.

blending from the visible faces using outlier detection. Ideally, faces have a large projected area size, having a small angle with respect to the camera ray, so that more details are available and less texture distortion exists. In the final stage, the remaining views after outlier selection are blended using the confidence as a weight. The previous aspects and functions are captured as principal functions and listed in each processing stage within Fig. 2. In the next Sections, these functions are described in more detail.

3 FACE VISIBILITY AND WEIGHT LABELING WITH Z-buffering

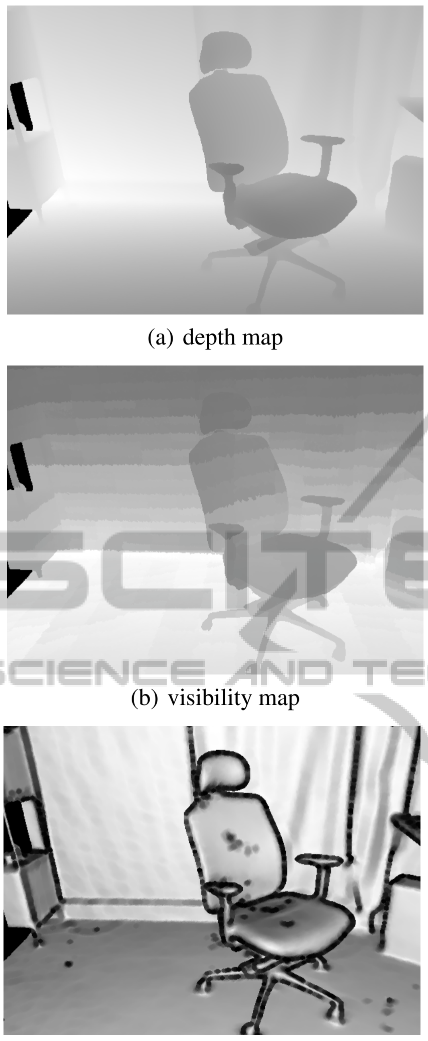
To color the 3D model, the faces are projected onto the 2D image and color values are extracted from it. To prevent background or occluded faces from being projected onto the 2D image, the visibility of each face needs to be determined for each camera view. A very effective method is to employ Z-buffering. This entails the creation of a depth map by projecting the depth values of the faces to the camera viewpoint. When multiple 3D points are projected onto the same 2D coordinates, we only keep the nearest value to the camera, thereby creating a so-called Z-buffer for that camera view. To this end, we have implemented an efficient algorithm to identify visible faces, extract projected area sizes of faces and create a confidence map using Z-buffering.

To determine the visibility of a face, we could test if this face resides on the surface of the depth map. However, this is a costly process, since we would have to iterate through multiple faces and compare the depth values of the vertices of the face with the depth map. Instead, we adopt an approach to create a visibility map in parallel to the depth map, by writing the face number into the image. Having this visibility map, it is only required to iterate through every pixel of the map. If a face number exists on the visibility map, we label this face as visible for this camera view. This method is definitely more efficient, because the number of pixels is usually much lower than the number of faces. Moreover, the test whether a face resides on the surface of the depth map can be fully omitted. As a bonus, the projected area size of each visible face can be easily determined by simply accumulating the number of pixels of each visible face.

From the visible faces, new textels need to be generated. Since not every projected face is equally reliable, we need to assign a weight value to each face. We consider a face as reliable when (1) it has a large projected area size, and (2) if its angle (θ) with respect to the camera ray is small. A smaller θ usually results in less texture distortion. For assigning weights to visible faces efficiently, we create a confidence map similar to the visibility map, by using the Z-buffer as a guide. Instead of depth values, we append the value of $\cos(\theta)$ of each face to this map as a unity-interval reliability indication. Since the camera viewpoints have registration errors, it is desirable to lower the weight of the faces that are projected near object boundaries. This is performed by dilating the confidence map with a structural element of 7×7 pixels. In Fig. 3 an example of the depth, visibility and confidence map are depicted. The values are scaled to [0–255] gray values. The depth map (Fig. 3(a)) represents the visible surface. The visibility map (Fig. 3(b)) is used to identify visible faces and to extract the area size of those faces. The confidence map (Fig. 3(c)) is used to assign weights to faces. We can clearly observe that faces projected to objects boundaries are assigned lower weights (darker), as is desired. To extract the weight for a particular face, we project this face on to the 2D image and calculate the average value of the projected face in the confidence map.

4 VIEW SELECTION AND TEXTURE BLENDING

In the previous section, we have identified the visible faces for each view and extracted their geometric properties such as the projected area size and weight (angle). However, due to registration and 3D modeling inaccuracies, not all visible faces are projected to the correct position in the 2D image. Therefore, we purposely select for each face a set of reliable views from the set of views in which this face is visible. We first use the projected area size as the selection metric. In the following step, we add the color information to


(a) depth map

(b) visibility map

(c) confidence map

Figure 3: Example of depth, visibility and confidence map for the 'Desk' scene.

our selection process.

Let us now describe our two-step selection procedure in more detail. In the first step, we use a sigma-outlier detection (Hodge and Austin, 2004) to remove small faces. With this approach a value is identified as outlier if it is clearly outside a predetermined interval about the mean, i.e. the mean $\pm\ n\times$the standard deviation of that variable. Small faces contain less details and/or are distorted due to a small angle (θ). Therefore, the use of these faces for generating a texture patch results in more blurring and distortion artifacts during rendering. In the second step, the color information of the faces is applied for outlier detection, considering that blending faces with similar visual appearances results in less blurring. We employ two methods for color outlier detection. The first outlier detection method uses the mean color of the face (C_m). Similar to the first step, we employ a sigma-outlier but now with the mean C_m within predetermined interval of the criterion. For the second outlier detection, we extract the color histogram (C_{hist}) for each projected face. Then for all histograms of the visible faces, we calculate the distance between each histogram and the average histogram (C_{hist_avg}). For the calculation of the distance of two histograms (C_{hist_dist}), we have experimented with the *correlation coefficient* (Imamura et al., 2011), the *chi-square* (Pele and Werman, 2010) and the *Bhattacharyya* distance (Bhattacharyya, 1943).

Ideally, the view selection procedure provides for each face a set of reliable views. However, due to different lighting conditions of the camera views, the color values can vary greatly. In addition, variations in the area size of the selected faces result in texture patches with different levels of detail. To reduce the variance in color and detail, we apply weighted averaging. Most previously mentioned multi-view texture mapping approaches use the projected area size of a face as a weight. This is logical, considering that a larger projected area size contains more detail. However, this approach does not consider the registration and 3D modeling errors which occur when performing automated camera pose estimation. Our approach employs the confidence map to assign weights to faces, as described in the previous section. The angle θ serves equally well in comparison to the projected area size, since $cos(\theta)$ is proportional to the area size. In addition, using the confidence map, color bleeding between background and foreground will be reduced considerably due to the lower weights of faces near boundary objects. With the selected faces and their weights, we now can generate the new textel for each face. Given N valid color patches, the blending function is defined by

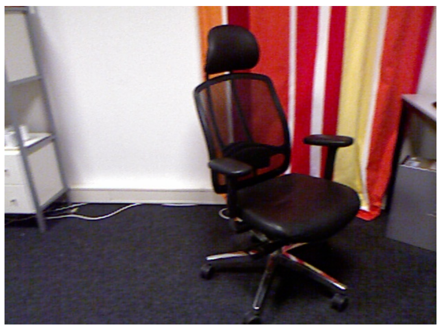
$$\mathcal{C} = \frac{\sum_{i=1}^{N} w_i \times \mathcal{C}_i}{\sum_{i=1}^{N} w_i}, \tag{1}$$

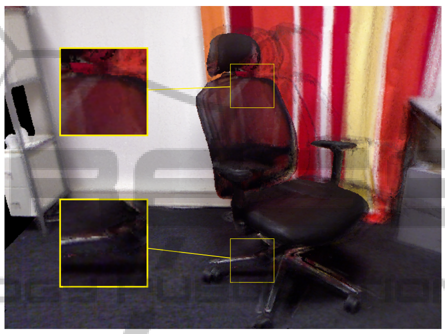
where w_i denotes the weight extracted from the confidence map and $\mathcal{C}$ represents the (R,G,B) color patch. With this blending function, we assume that a color patch is more reliable when the local surface is more orthogonal to the viewpoint and not near an object boundary. Since the projected patches of the camera views vary in size and position, it is necessary to generate a new textel on a pixel by pixel basis. This can be easily achieved with barycentric coordinates (Bradley, 2007). In our implementation, Equation (1) is performed for every pixel of the textel and the color value $\mathcal{C}_i$ of each camera view is retrieved using barycentric transformation. To reduce blurring, it is attractive to upper bound the number of valid camera views for blending. Once a new textel is generated, we store this textel in a larger texture file where they are packed in ascending face number.

5 EXPERIMENTAL RESULTS

To evaluate our multi-view texturing approach for indoor environments, we have performed a series of experiments on three scenes which differ in complexity and number of camera views. The input datasets are acquired from an extended RGB-D (Whelan et al., 2012; Bondarev and Heredia, 2013) reconstruction system with a Kinect sensor, which provides a triangular mesh and a set of 640×480-pixels textures with calibration parameters. In the first series of experiments, we have generated textels using only geometric properties of the scene. To show the effectiveness of our confidence map, we have compared rendered views (using software rendering) with the projected area size as a weight, against rendered views with a confidence value as a weight. Fig. 4(c) shows that employing a confidence map reduces ghosting errors. However, blurring and ghosting artifacts are still noticeable.

In the second series of experiments, we have supplied the color information to our outlier detector and the confidence term is used as a weight for texture blending. Let us now examine these rendering results and compare them with the previous results. From Fig. 5, we can observe that by adding color information the ghosting errors are reduced drastically. We have also found that the rendered views are sharper, giving the impression of more details. The usage of color histograms (5(b)) instead of the mean color value (5(a)) yields a slightly sharper image. However, the perceptual improvements are marginal. This can be readily understood, since the projected area size of the average face is very small, consisting of only a few pixels. For such small faces, the outlier detector for histograms and mean values have almost identical performance. From the previous experiments and results, we can distinguish three ways of textel generation: (1) using only geometric properties, (R_{geo}), (2) using outlier detection of the mean color value (R_{rgb}) and (3) using outlier detection of the color histogram (R_{hist}) with three variations in histogram distance measurement. In Fig. 6, these three ways of textel generation are compared side by side, for the 'Couch' scene. We can observe that the differences in textel generation methods show a similar trend to that of the 'Desk' scene. Textures from the R_{hist} method (Fig. 6(c)) contain more detail than the other two methods. However, this method also introduces slightly more noise (left side of the couch). In Table 1, we have summarized the average PSNR quality of the three scenes using R_{geo}, R_{rgb} and R_{hist}. The average PSNR is obtained by first rendering the views at the identical position as the reference camera views and then calculate the average PSNR over all the views. We can observe that the three methods for calculating the histogram distance perform (almost) equally well. The overall obtained average PSNR quality is relatively low, which is explained by the distributed amounts of blurring and the 3D modeling and registration errors. By inspecting the depth maps and color images of the individual scenes more closely, we have observed that the 'Couch' scene contains the


(a) reference view, camera 4


(b) rendered view with projected area size as weight

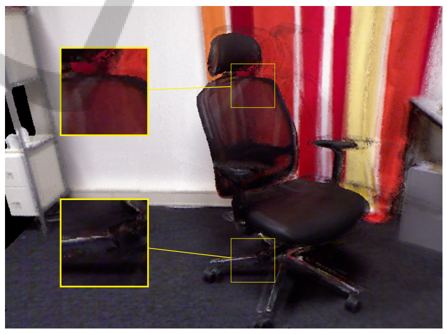
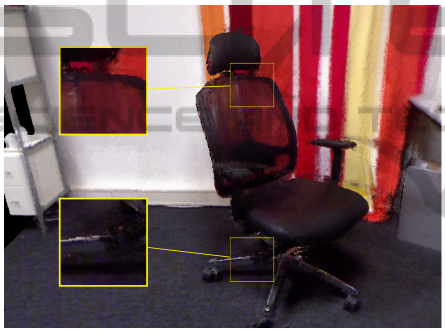

(c) rendered view with confidence value as weight

Figure 4: Comparison of rendered views for the 'Desk' scene (29 camera views) using only geometric properties.

Table 1: Average PSNR quality measured for three scenes and various metrics.

Scene	# faces	# views	R_{geo} (dB)	R_{rgb} (dB)	R_{hist} (dB) correlation	R_{hist} (dB) chi-square	R_{hist} (dB) Bhattach.
Desk	500,000	29	20.1	20.0	19.5	19.8	19.7
Couch	2,109,558	25	24.9	24.8	24.4	24.7	22.5
Basement	1,634,591	33	20.4	20.4	19.8	20.2	20.0

least amount of registration errors. This results in a higher average PSNR score compared to the other two scenes. The relatively low PSNR values also indicate that the realism of the scene is deteriorated. This could be expected in advance because the view-dependent color values of each camera are replaced by the weighted average values. The 3D visualization of the three scenes using meshlab is illustrated in In Fig. 7. We can clearly observe that our colored models give natural appearance representations.


(a) rendered view with C_m

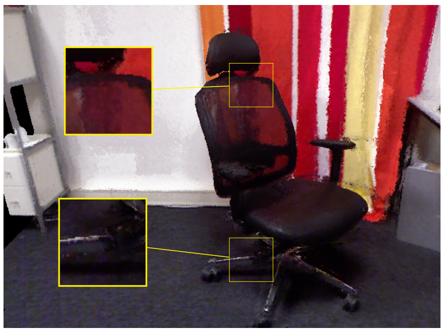
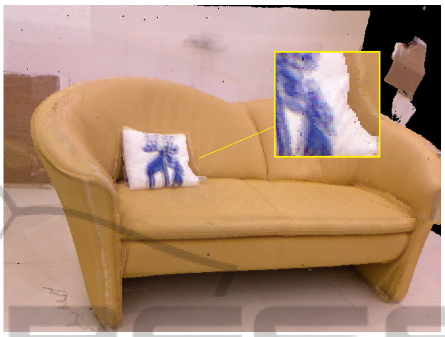
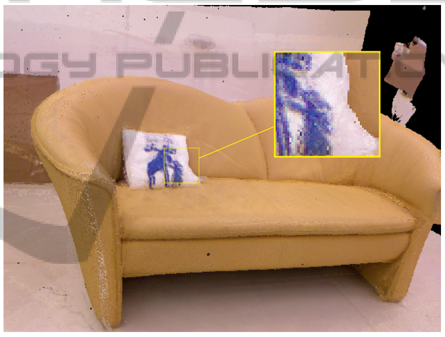

(b) rendered view with C_{hist} and correlation coefficient

Figure 5: Comparison of rendered views for the ‘Desk’ scene with C_m and C_{hist}.


(a) rendered view using R_{geo}


(b) rendered view using R_{rgb}

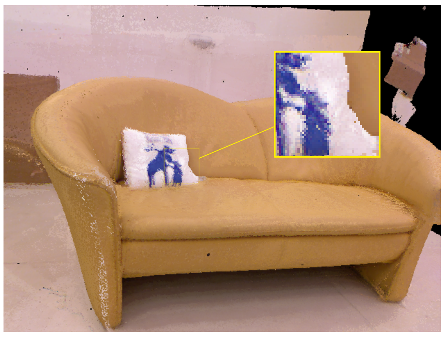
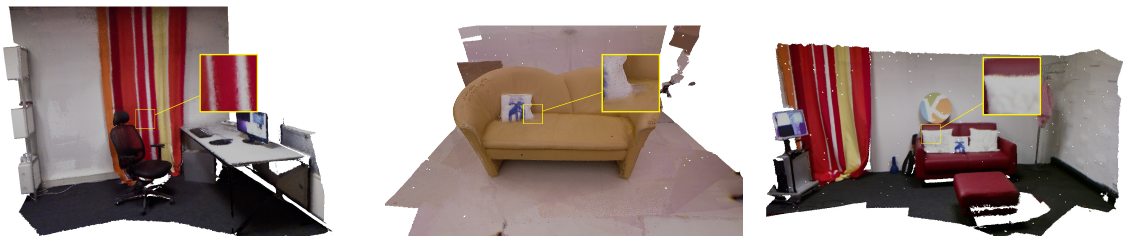

(c) rendered view using R_{hist} and correlation coefficient

Figure 6: Comparison of rendered views for the ‘Couch’ scene w/o color information.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have explored the problem of generating colored 3D models of indoor scenes using multi-view texturing, with a focus on inexpensive reconstruction platforms. We have developed algorithms for colored 3D models of indoor scenes that: (a) can effectively process a large number of camera views, (b) are robust to multi-viewpoint registration errors and 3D modeling inaccuracies. Our contribution is threefold. Firstly, we have applied a visibility map to efficiently identify and extract the visible faces. Secondly, a confidence map is employed to assign



(a) 'Desk' scene with R_{rgb} (b) 'Couch' scene with R_{rgb} (c) 'Basement' scene with R_{rgb}

Figure 7: Meshlab 3D visualization using method R_{rgb}.

weights to faces with an additional function to reduce ghosting errors at object boundaries. Thirdly, color histograms are used in combination with sigma-outlier detectors to remove unreliable generation is obtained with high details, so that this leads to a sharper rendering quality. However, from the overall low PSNR quality we can conclude that the textel quality is highly dependent on the 3D modeling and registration errors. For our scenes, the visual difference of textel quality between the use of color histogram and mean color is minimal, because the average projected area size of a face is rather small (a few pixels). For such scenes, the benefit of employing color histograms does not out weight the involved extra computational costs. To improve textel quality, we could explore the concept of super resolution (Goldluecke and Cremers, 2009), employ a distributed weighting scheme (Wang and Kang, 2001) or perform local matching of the faces (Rocchini et al., 2002). For improving registration and thereby reducing blurring originated from registration errors, image-based camera calibration prior to textel generation is recommended.

REFERENCES

Alshawabkeh, Y. and Haala, N. (2005). Automatic multi-image photo-texturing of complex 3D scenes. *XVIII CIPA Int. Symposium, Torino, 27 Sept*, pages 1–6.

Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of Cal. Math. Soc.*, 35(1):99–109.

Bondarev, E. and Heredia, F. (2013). On photo-realistic 3D reconstruction of large-scale and arbitrary-shaped environments. pages 621–624.

Bradley, C. (2007). *The Algebra of Geometry: Cartesian, Areal and Projective Co-Ordinates*. Highperception Limited.

Catmull, E. (1974). A subdivision algorithm for computer display of curved surfaces.

Goldluecke, B. and Cremers, D. (2009). Superresolution texture maps for multiview reconstruction. *IEEE International Conference on Computer Vision (ICCV)*.

Grammatikopoulos, L., Kalisperakis, I., Karras, G., and Petsa, E. (2007). Automatic multi-view texture mapping of 3d surface projections. *3D Virtual Reconstruction & Visualization of Complex Architectures*, pages 12–13.

Heckbert, P. (1986). Survey of texture mapping. In *IEEE Computer Graphics and Applications*, pages 56–67.

Hodge, V. and Austin, J. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126.

Imamura, K., Kuroda, H., and Fujimura, M. (2011). Image content detection method using correlation coefficient between pixel value histograms. In *Signal Processing, Image Processing and Pattern Recognition*, volume 260, pages 1–9. Springer Berlin Heidelberg.

Pele, O. and Werman, M. (2010). The quadratic-chi histogram distance family. *ECCV*, (1):1–14.

Rocchini, C., Cignoni, P., and Montani, C. (1999). Multiple Textures Stitching and Blending on 3D Objects. *Eurographics Rendering Workshop*.

Rocchini, C., Cignoni, P., Montani, C., and Scopigno, R. (2002). Acquiring, stitching and blending diffuse appearance attributes on 3D models. *The Visual Computer*, 18(3):1–24.

Sainz, M. and Pajarola, R. (2004). Point-based rendering techniques. *Computers & Graphics*, 28(6):869–879.

Wang, L. and Kang, S. (2001). Optimal texture map reconstruction from multiple views. *Computer Vision and pattern recognition (CVPR)*, pages 1–8.

Whelan, T., Kaess, M., and Fallon, M. (2012). Kintinuous: Spatially Extended KinectFusion.

Whitted, T. (2005). An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349.

Yuksel, C., Keyser, J., and House, D. H. (2010). Mesh colors. *ACM Transactions on Graphics*, 29(2):1–11.

Zwicker, M., Pfister, H., van Baar, J., and Gross, M. (2001). Surface splatting. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, pages 371–378.