2015

Creating Simplified 3D Models with High Quality Textures

Song Liu
University of Wollongong, songl@uow.edu.au

Wanqing Li
University of Wollongong, wanqing@uow.edu.au

Philip O. Ogunbona
University of Wollongong, philipo@uow.edu.au

Yang-Wai Chow
University of Wollongong, caseyc@uow.edu.au

Publication Details

Liu, S., Li, W., Ogunbona, P. & Chow, Y. (2015). Creating Simplified 3D Models with High Quality Textures. 2015 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2015 (pp. 264-271). United States of America: The Institute of Electrical and Electronics Engineers, Inc..

Creating Simplified 3D Models with High Quality Textures

Abstract

This paper presents an extension to the KinectFusion algorithm which allows creating simplified 3D models with high quality RGB textures. This is achieved through (i) creating model textures using images from an HD RGB camera that is calibrated with Kinect depth camera, (ii) using a modified scheme to update model textures in an asymmetrical colour volume that contains a higher number of voxels than that of the geometry volume, (iii) simplifying dense polygon mesh model using quadric-based mesh decimation algorithm, and (iv) creating and mapping 2D textures to every polygon in the output 3D model. The proposed method is implemented in real-Time by means of GPU parallel processing. Visualization via ray casting of both geometry and colour volumes provides users with a real-Time feedback of the currently scanned 3D model. Experimental results show that the proposed method is capable of keeping the model texture quality even for a heavily decimated model and that, when reconstructing small objects, photorealistic RGB textures can still be reconstructed.

Disciplines

Engineering | Science and Technology Studies

Publication Details

Liu, S., Li, W., Ogunbona, P. & Chow, Y. (2015). Creating Simplified 3D Models with High Quality Textures. 2015 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2015 (pp. 264-271). United States of America: The Institute of Electrical and Electronics Engineers, Inc..

Creating Simplified 3D Models with High Quality Textures

Song Liu, Wanqing Li, Philip Ogunbona, Yang-Wai Chow
Advanced Multimedia Research Lab
University of Wollongong, Wollongong, NSW, Australia, 2522
{sl796,wanqing,philipo,caseyc}@uow.edu.au

***Abstract*—This paper presents an extension to the KinectFusion algorithm which allows creating simplified 3D models with high quality RGB textures. This is achieved through (i) creating model textures using images from an HD RGB camera that is calibrated with Kinect depth camera, (ii) using a modified scheme to update model textures in an asymmetrical colour volume that contains a higher number of voxels than that of the geometry volume, (iii) simplifying dense polygon mesh model using quadric-based mesh decimation algorithm, and (iv) creating and mapping 2D textures to every polygon in the output 3D model. The proposed method is implemented in real-time by means of GPU parallel processing. Visualization via ray casting of both geometry and colour volumes provides users with a real-time feedback of the currently scanned 3D model. Experimental results show that the proposed method is capable of keeping the model texture quality even for a heavily decimated model and that, when reconstructing small objects, photorealistic RGB textures can still be reconstructed.**

I. Introduction

Generating 3D models based on real-world environments and with high quality textures is of great significance to many fields including civil engineering, 3D printing, game design, movie, virtual reality and preservation of cultural heritage artefacts. Various computer vision-based approaches have been proposed to create 3D models and deal with the associated classical problems such as simultaneous localization and mapping (SLAM), and structure-from-motion (SFM). To date, impressive progress has been made in this domain [6][2][8]. Largely, many approaches use visual key points to build 3D models leading to sparse point cloud based 3D reconstruction. Conventional dense reconstruction method [9][28] on the other hand usually require professional sensors such as high-fidelity laser scanners or time-of-flight (ToF) depth cameras which are very expensive.

The release of commodity RGB-D cameras such as the Microsoft Kinect™ and Asus Xtion™ has made dense 3D reconstruction possible at an affordable cost. This, along with the KinectFusion algorithm [17][15] , has enabled real-time dense 3D reconstruction using a low-cost RGB-D camera and GPU parallel processing. Subsequent efforts by other researchers have led to the development of several KinectFusion-based methods [26][21][4][18] that allow efficient 3D reconstruction on a large scale and with higher reconstruction quality. However, current KinectFusion-based methods tend to deliver 3D models with high quality geometry but low quality texture. In other words, the works on improving model textures are less advanced. Moreover, 3D models created by dense 3D reconstruction usually contain significant redundant information, which would increase the model complexity and lower the rendering efficiency. In many cases, it is necessary to simplify a dense 3D model to achieve higher rendering efficiency, especially for large scale rendering or on platforms with limited processing powers such as cell phones and tablets. It is noteworthy that for many 3D models generated from existing 3D reconstruction systems, the quality of model texture is directly related to the model complexity. Furthermore, decimation of the polygon mesh models will degrade the model texture.

II. Related Work

The problem of reconstructing geometry and texture of real world has remained an active challenge in the field of computer vision for decades. We now review some of the extant aproaches and the associated results.

Conventional 3D reconstruction approaches usually do not consider model texture information, or represent model texture in a simple way. Chen and Medioni [5] average overlapping range images and connect points based on simple surface topology to create polygon mesh models; model textures are totally ignored. Turk and Levoy [23] propose mesh zippering as an extension to Chen and Medioni's work; they stitch polygon meshes to create 3D model without textures. Some point-based 3D reconstruction methods [20][24][12][22] use simple unstructured point representations that are directly captured from many range imaging devices. These methods do not model connected surfaces which usually requires post processing to generate polygons. In most popular point-based 3D model rendering techniques [11][19][30], textures are simply represented by colours attached to each point in the model.

The release of low-cost RGB-D cameras like the Microsoft Kinect™ and Asus Xtion™ opens up new opportunities to 3D reconstruction in terms of providing easy access to depth imaging. KinectFusion [17][15] adopts volumetric data structure to store reconstructed scene surface [7][14] and realise real-time reconstructions using GPU. Although model textures are not considered in the original KinectFusion algorithm, it inspires multiple volumetric 3D reconstruction methods using commodity RGB-D cameras that try to create dense polygon mesh with RGB textures. Among these KinectFusion-based methods, an open source C++ implementation of KinectFusion from Point Cloud Library (PCL) [1], Whelan et al. [26] and Bylow et al. [4] use a colour volume to store and update RGB texture information. In their methods, model textures on reconstructed 3D models are represented by colours on model vertices, and these colours are linearly interpolated within each polygon. This popular 3D model texture representation can also be easily found in many other 3D reconstruction methods [21][25][18]. While the texture representation is straightforward and easy to implement, simplifying the model inevitably degrades the texture quality because it is determined by the number of vertices in the model.

Zhou and Koltun [29] reconstruct 3D models with high quality colour textures using a commodity depth camera and an HD RGB camera. HD model textures are refined using optimized camera poses in tandem with non-rigid correction functions for all images. Because model textures are also represented by colours assigned to each vertex, the generation of high quality textures requires increasing the number of vertices and polygons of a 3D model. This results in increased model complexity.

In order to create simplified 3D models, Whelan et al. [27] decimate dense polygon mesh model by means of planar simplification and simultaneously preserve model colour information by 2D texture mapping. However, their work focuses on scene reconstruction where flat floors and walls tend to be over represented by millions of polygons. For 3D models of arbitrary shapes, planar simplification is not suitable. Moreover, their texture preserving method does not consider generating HD RGB textures with resolution beyond what a Kinect RGB camera currently provides.

In this paper, a KinectFusion-based method for creating simplified 3D models with high quality textures is presented. Texture information is stored and updated in a colour volume with a higher dimension than that of the truncated signed distance function (TSDF) volume. Two-dimensional texture images are extracted from colour volume and are mapped to reconstructed 3D models so that model texture can retain its quality even on a simplified 3D model with much fewer number of polygons.

III. Background

Our method is based on the open source C++ implementation of the KinectFusion algorithm provided by PCL [1]. In KinectFusion, retrieved RGB image and registered depth map are taken as inputs. A vertex map and normal map pyramid based on input depth map is calculated; the value is then used to estimate the camera motion using iterative closest point (ICP) algorithm [3] in conjunction with a predicted surface vertex map. The map is derived from currently stored depth information. With estimated camera motions, depth and RGB information from the input frame is used to update the current model. In the work of KinectFusion, scanned depth information is stored in a TSDF volume, in which each voxel stores the distance to its closest surface. Since TSDF volume contains geometric information of scanned 3D model and is updated using depth map, in our method, it is referred to as the geometry volume. The predicted surface vertex map used for camera tracking is obtained by ray tracing the geometry volume. The original KincetFusion does not consider reconstructing model textures. Its C++ implementation in PCL capture and save texture information using a separate colour volume that has the same dimension and size as the TSDF volume. In our paper, this colour integration method is adopted and extended. Given updated geometry and colour volumes, 3D polygon mesh model with textures can be extracted using marching cube algorithm [16]; a method that aims to produce triangle polygons for visualization.

IV. Improved Method

The workflow of our improved method is shown in Fig 1. The similarity with the KinectFusion process is noticeable. However, in order to create simplified 3D models with high quality textures, the following major improvements are made:

- HD RGB camera is added to achieve higher model texture quality;
- Colour volume integration scheme is revised to provide an asymmetrical colour volume;
- Dense polygon mesh model is simplified without losing much geometry information;
- Simplified model is textured and details are retained.

In subsequent subsections each major improvement is presented in more detail.

A. HD Camera Setup and RGB-D Calibration

The on-board RGB camera of the Kinect sensor can only return RGB images in VGA resolution (640×480) and this insufficient for generating high quality textures. This shortcoming is mitigated, in order to achieve high quality texture mapping, by rigidly attaching an external HD RGB camera to the Kinect sensor assembly. The resulting assembly delivers high-definition RGB images (see Fig 2).

Fig. 2. Kinect sensor with HD RGB camera

The HD RGB image and depth map are misaligned due to the different placements and field of views of the HD RGB camera and the depth camera of Kinect sensor. Hence high quality texture mapping will depend on accurate mapping relation between HD RGB camera space and Kinect depth camera space.

Herrera C's RGB-D calibration toolbox [13] is adopted to calculate camera intrinsic and extrinsic parameters. Camera intrinsic parameters include camera principal point (image centre) and focal length in pixel-related units which are used to project points between image space and camera space. Camera intrinsic parameters can be compactly represented in a matrix, $K \in \mathbb{R}^{3\times3}$. Let the coordinate vector in image space be denoted by $(x, y, z)^T$ and the coordinate vector in the camera space be $(X, Y, Z)^T$. The transformation from camera space to image space can be written as

$$(x, y, z)^T = K(X, Y, Z)^T. \tag{1}$$

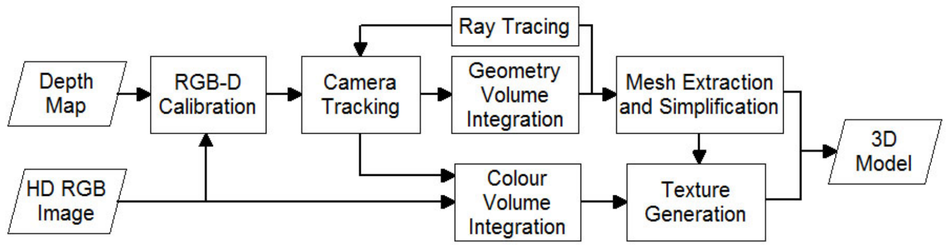


Fig. 1. Block diagram showing the workflow of improved method.

Camera extrinsic parameters describe the relative position of the camera to another object (another camera in our case), and can be written as a rotation plus a translation. The extrinsic parameters from Kinect depth camera space to HD RGB camera space are denoted by rotation matrix $R_{calib} \in \mathbb{R}^{3\times3}$ and translation vector $t_{calib} \in \mathbb{R}^3$. Let the intrinsic parameters of Kinect depth camera and HD RGB camera be donoted as $K_c \in \mathbb{R}^{3\times3}$ and $K_{hd} \in \mathbb{R}^{3\times3}$ respectively.

Given calibrated camera intrinsic and extrinsic parameters, a mapping from depth image to HD RGB image is calculated as follows. Assume a point in real world is captured by the depth camera whose coordinate in depth image space is (x_c, y_c, z_c). Let its 3D coordinate in depth camera space be denoted as (X_c, Y_c, Z_c); the 3D coordinate in HD RGB camera space as (X_{hd}, Y_{hd}, Z_{hd}); and its coordinate in HD RGB image space as (x_{hd}, y_{hd}, z_{hd}). The equations governing the respective transformations can be written as follows:

$$(X_c, Y_c, Z_c)^T = K_c^{-1}(x_c, y_c, z_c)^T; \tag{2}$$

$$(X_{hd}, Y_{hd}, Z_{hd})^T = R_{calib}(X_c, Y_c, Z_c)^T + t_{calib}; \tag{3}$$

$$(x_{hd}, y_{hd}, z_{hd})^T = K_{hd}(X_{hd}, Y_{hd}, Z_{hd})^T. \tag{4}$$

The mapped 2D coordinate (x_{hd}, y_{hd}) in HD RGB image will be considered when updating colour volume.

B. Colour Volume Integration

In the implementation of KinectFusion from PCL, RGB texture information is saved in a separate volume but similarly dimensioned and sized as the TSDF geometry volume. This establishes one-to-one correspondence between voxels in colour volume and geometry volume. In our method, modifications are made to improve PCL's technique to deliver higher quality model textures.

Since all texture information is saved in the colour volume, a colour volume with higher dimension will result in better texture quality. As mentioned earlier, most KinectFusion-based methods tend to deliver 3D models with high quality geometry but low quality textures. Based on this observation and limited GPU memory availability, a strategy to achieve higher colour volume dimension is to make the geometry and colour volumes asymmetrical. Specifically, the dimension of colour volume is made higher than that of the geometry volume while their sizes remain the same. In this way, the actual space in real world taken by a colour voxel is much smaller than that is taken by a geometry voxel; this results in capturing more textures details.

The colour volume containing updated texture information from the 1^{st} frame to the n^{th} frame is denoted by $C_n(v)$, where $v \in \mathbb{N}^3$ is the 3D coordinate representing the location of a voxel in the colour volume. Each voxel of the colour volume stores a 3×1 RGB colour vector $C_n(v)$ and a weight $W_n(v)$.

When updating the colour volume, for each voxel v in the colour volume, its 3D coordinates (X_v, Y_v, Z_v) in the depth camera space of the first frame is considered. Given all camera motions between consecutive frames so far, by synthesizing all the motions, the camera motion (R_n, t_n) from the first frame to the current n^{th} frame can be calculated. Therefore, the coordinates of this voxel in the current camera coordinate system, (X_{vc}, Y_{vc}, Z_{vc}), can be calculated as

$$(X_{vc}, Y_{vc}, Z_{vc})^T = R_n(X_v, Y_v, Z_v)^T + t_n. \tag{5}$$

Using the depth camera intrinsic matrix K_c, 3D point (X_{vc}, Y_{vc}, Z_{vc}) can be mapped to its depth image coordinates using the equation,

$$(x_{vc}, y_{vc}, z_{vc})^T = K_c(X_{vc}, Y_{vc}, Z_{vc})^T. \tag{6}$$

Based on the 2D coordinate (x_{vc}, y_{vc}), if voxel v is outside the current depth camera frustum, the updating process terminates and the algorithm moves on to update the next voxel. If it is inside the current depth camera frustum, its actual depth value $D_n(x_{vc}, y_{vc})$ is retrieved from the n^{th} depth map D_n.

If $D_n(x_{vc}, y_{vc})$ is non-zero, the valid depth map pixel $(x_{vc}, y_{vc}, D_n(x_{vc}, y_{vc}))$ and its 3D point (X_{vd}, Y_{vd}, Z_{vd}) in the current camera coordinate system are calculated as

$$(X_{vd}, Y_{vd}, Z_{vd})^T = K_c^{-1}(x_{vc}, y_{vc}, D_n(x_{vc}, y_{vc}))^T. \tag{7}$$

If the Euclidean distance between (X_{vc}, Y_{vc}, Z_{vc}) and (X_{vd}, Y_{vd}, Z_{vd}) is greater than a threshold σ, the process terminates. In our experiments, the threshold σ is set to 20 (equivalently 20mm).

The weight of a point, W, is the dot product between the surface normal at this point and the viewing direction from the camera to this point. The normal vector for the current point (X_{vd}, Y_{vd}, Z_{vd}) is computed using neighbouring retrojected points. Assuming $(X_{1vd}, Y_{1vd}, Z_{1vd})$ and $(X_{2vd}, Y_{2vd}, Z_{2vd})$ are the camera coordinates of $(x_{vc}+1, y_{vc}, D_n(x_{vc}+1, y_{vc})$ and $(x_{vc}, y_{vc}+1, D_n(x_{vc}, y_{vc}+1)$, the normal is calculated using the formula,

$$\begin{aligned} N_v =& ((X_{1vd}, Y_{1vd}, Z_{1vd}) - (X_{vd}, Y_{vd}, Z_{vd}))\times \\ & ((X_{2vd}, Y_{2vd}, Z_{2vd}) - (X_{vd}, Y_{vd}, Z_{vd})). \end{aligned} \tag{8}$$

If the weight, W, exceeds the previous weight times 0.8 (i.e. $0.8 \cdot W_{pre}$), the voxel is updated. Since HD RGB frames are used to deliver high-resolution RGB images, for colour voxel

v to be updated, its corresponding pixel in HD RGB image $HD(x_{hd}, y_{hd})$ can be located considering camera intrinsic and extrinsic parameters from RGB-D calibration (Section IV.A). Then, the new colour C_{new} in this colour voxel is updated as

$$C_{new} = \frac{C_{pre} \cdot W_{pre} + W \cdot HD(x_{hd}, y_{hd})}{W_{pre} + W} \tag{9}$$

where C_{pre} is the previous RGB value in this colour voxel and $HD(x_{hd}, y_{hd})$ is the RGB value of pixel (x_{hd}, y_{hd}) in the HD RGB image. The new weight, W_{new}, is updated as,

$$W_{new} = W_{pre} + W \cdot (1 - W_{pre}) \tag{10}$$

Finally, the new RGB value C_{new} and the new weight W_{new} are saved in the colour voxel v.

C. Mesh Extraction and Simplification

The marching cube algorithm is used to extract 3D polygon mesh models from TSDF volume in KinectFusion. However, dense polygon mesh models obtained directly from marching cube algorithm tend to contain a large number of redundant polygons, which can increase the model complexity without providing much geometry details. When memory and rendering efficiency are considered, simplification of the reconstructed dense polygon mesh is necessary in many circumstances. In order to achieve model complexity reduction the quadric-based mesh decimation is adopted[10]. Fig. 3 shows the results of simplifying a dense chair model. The original dense chair model (Fig. 3(a)) is decimated to contain different numbers of polygons. It can be observed that all simplified models (see Fig. 3(b) - Fig. 3(f)) can still keep their shape as a chair. The results indicate that quadric-based mesh decimation algorithm is effective in simplifying polygon mesh model while preserving geometric features.

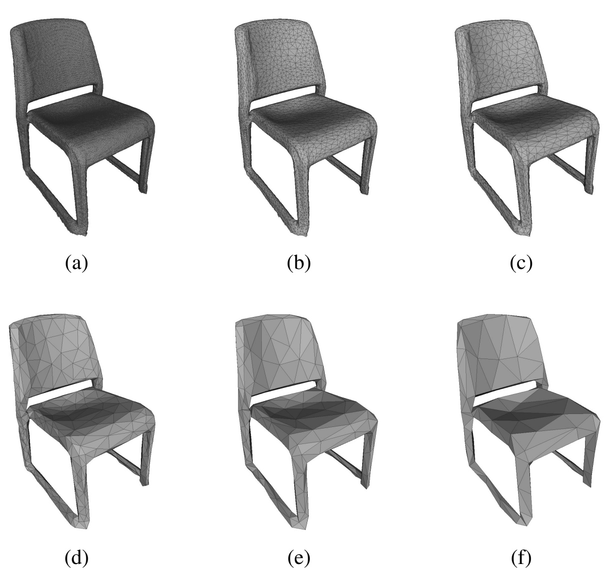


Fig. 3. Simplification of a chair model using quadric based mesh decimation. (a): original dense model containing 137037 polygons. (b): simplified model containing 13703 polygons (10%). (c): simplified model containing 6851 polygons (5%). (d): simplified model containing 1370 polygons (1%) (e): simplified model containing 685 polygons (0.5%). (f): simplified model containing 343 polygons (0.25%).

In passing, note that the reduction rate parameter of the decimation algorithm specifies the degree to which the current mesh model will be decimated. For instance, reduction rate of 10% implies that the simplified mesh model after polygon reduction will contain 10% of polygons as the original model. This reduction rate can be adjusted according to the actual requirements of 3D model generated.

D. Texture Generation

1) 2D Texture Map Generation for Each Polygon: Compared with a dense polygon mesh, a simplified model contains much less polygons. Texturing by coloured vertex method will incur losses of texture details which in turn reduces the texture quality. To maintain texture quality on a model with less polygons, each polygon should be textured with more RGB details. To this end, different 2D texture maps are required to be generated and mapped to all polygons in a mesh model; one 2D texture map for one polygon.

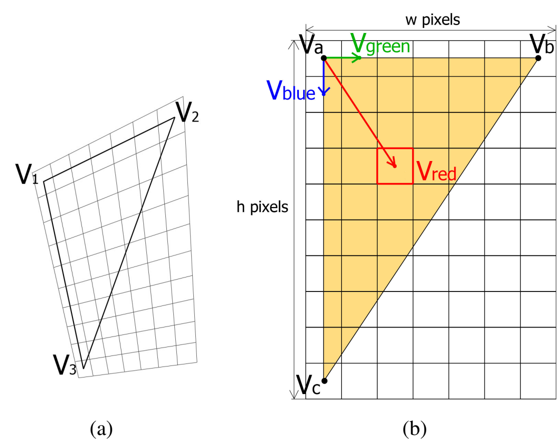


Fig. 4. Generating 2D texture map, each block represents one pixel in the 2D texture map. (a): a polygon in the model to be textured. (b): a 2D texture map for the polygon in (a).

In the reconstructed 3D models, each polygon is a triangle with 3 vertices. For an arbitrary polygon in a 3D model, assume its 3 vertices are (V_1, V_2, V_3) (Fig 4(a)); V_1, V_2, V_3 are points with 3D space coordinates in real world. When generating a 2D texture map (Fig 4(b)) for this particular polygon, the upper left triangle in the texture map ($\triangle V_a V_b V_c$ in Fig 4(b)) will be mapped onto this polygon. The mapping relations are defined as follows: $V_a \mapsto V_1$; $V_b \mapsto V_2$; and $V_c \mapsto V_3$

The resolution of the texture image is calculated as the actual size of polygon divided by a given pixel size parameter $Size_{pixel}$. The value of $Size_{pixel}$ represents the size of a pixel when mapped onto a 3D model, which should be set according to the HD RGB camera resolution and scanning distance. In Fig. 4(b), the resolution of 2D texture map is $w \times h$ pixels, where

$$w = \text{distance}(V_a, V_b)/Size_{pixel} \tag{11}$$

$$h = \text{distance}(V_a, V_c)/Size_{pixel} \tag{12}$$

distance(V_i, V_j) is the Euclidean distance between 3D coordinates V_i and V_j.

In order to complete the texture generation process, two unit vectors ($\overrightarrow{V_aV_{green}}, \overrightarrow{V_aV_{blue}}$) are required;

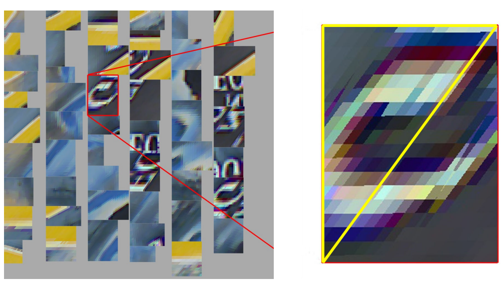
$$\overrightarrow{V_aV_{green}} = \overrightarrow{V_aV_b}/(w-1), \tag{13}$$

$$\overrightarrow{V_aV_{blue}} = \overrightarrow{V_aV_c}/(h-1). \tag{14}$$

For any pixel in the texture map, its 3D world position can be expressed as a combination of these two vectors. If the pixel marked with a red bounding box is considered for example, its 3D world position can be expressed as $V_a + \overrightarrow{V_aV_{red}}$. This is equivalent to $V_a + 2\overrightarrow{V_aV_{green}} + 3\overrightarrow{V_aV_{blue}}$. Given the 3D world positions, the RGB value of any pixel can be found by accessing the colour volume. By repeating the process above for every pixel in the texture map, a complete 2D texture map can be obtained.

2) 2D Texture Map Merging: A regular 3D model (including simplified ones) usually contains hundreds or thousands of polygons. The generation of 2D textures for each polygon will result in the production of very large number of 2D texture maps. Loading each texture map one at a time is time consuming and required a speedup strategy. An efficient method is to load multiple textures onto one or a few 2D images. This strategy is implemented in 2D texture merging scheme.

2D texture image that are supposed to contain texture maps for multiple polygons are created. Each texture map for one polygon is added and placed in the 2D image starting from the top-left corner to bottom-right corner in a column-wise order.



(a) (b)

Fig. 5. (a): a merged 2D texture image file. (b): a 2D texture map for one polygon. Best viewed in colour.

Fig. 5 depicts one 2D texture image file created by merging multiple texture maps. Fig. 5(a) shows the merged texture image and Fig. 5(b) is a 2D texture map for one polygon. The upper left triangle of each 2D texture map (yellow triangle in Fig 5(b)) will be rendered on a polygon as the actual model texture.

V. Experiments and Results

The HD RGB camera used in our experiment is a Point Grey GS3-U3-41C6C-C camera with an 8mm F1.4 wide angle lens. Our 3D reconstruction method is implemented in C++ and run on a PC with a Nvidia GTX680 GPU, 4GB graphical memory.

Experiments are carried out on 2 test cases to verify the effectiveness of our texturing method in terms of

- texture improvements from using 2D texture mapping compared with using coloured vertex; and
- texture improvements from using HD RGB images compared with using Kinect RGB images.

The test cases selected include a lunch bag and a backpack, which are small and medium-sized objects with textures of noticeable details (e.g. brand logo). In our experiment, the dimension of geometry volume is $384 \times 384 \times 384$ and the asymmetrical colour volume is set to $768 \times 768 \times 768$. Volume size is variable to suit the size of the object to be reconstructed.

To obtain a visual evaluation of texture quality, reconstructed 3D models with textures are loaded and visualized in MeshLab. A snapshot from a fixed view point is taken for each model. An objectve evaluation of our algorithm measures the level of texture quality. For each test case, image patches are cropped from the same location where patterns containing detailed texture could be found (e.g. brand logo). An image patch from high quality textures tend to be sharp, so that more texture details can be captured. Therefore, image sharpness of the image patch is measured as the indicator of texture quality using gradient magnitude; that is, the sum of all gradient norms divided by number of pixels. A higher gradient magnitude implies more sharpness and more texture details.

A. RGB-D Camera Calibration Results

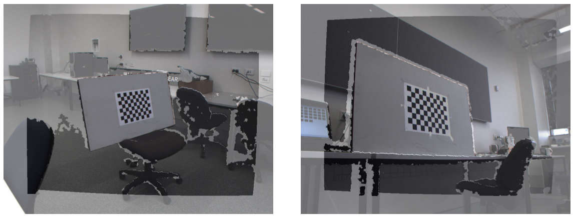
Efforts to ensure correct RGB texture mapping require an accurate calibration between HD RGB camera and the Kinect depth camera. An evaluation on RGB-D camera calibration is performed before our experiments.

In the process of RGB-D calibration, 100 sample shots were taken which include 100 HD RGB images and 100 depth maps. These samples can be further divided into 3 groups according to the average shooting distance (distance between the camera and the chess board pattern), which include short range(0.5M-1M), middle range(1M-2.5M) and long range(2.5M-4M). In each group, there are more than 30 RGB-D image pairs that were taken from different view points of the chess board. RGB-D calibration is then performed using all sample shots.

The accuracy of RGB-D camera calibration is ascertained a straightforward way by visually checking the overlapped HD RGB image and visualised depth map for misalignment. The overlayed image can be generated by overlaying the aligned semitransparent depth map onto its corresponding HD RGB image (Fig. 6). Judging by the overlayed images, it can be observed that the depth map and HD RGB images are reasonably well aligned.

B. 3D Model Texture Results

1) Improvements from using 2D texture mapping: Most Kinectfusion-based methods texture the reconstructed models using coloured vertex; assignment of an RGB value to each vertex in the model. The texture of a polygon is the linear interpolation of 3 RGB value on its vertices, and this can lead to loss of details due to mesh simplification. On the other hand, our proposed texture method generate 2D texture maps directly from the colour volume and this has maintained texture details even on a simplified 3D polygon mesh.



(a) (b)

Fig. 6. Overlapped HD RGB image with registered depth map

Fig. 7 and Fig 8 depict simplified models that are textured with these two methods. Table I presents texture qualities of reconstructed models as measured by gradient magnitudes of their image patches. From the table, the quality of texture generated using coloured vertex drops drastically as the level of model simplification increases. However, our 2D textured mapping method is able to maintain the texture quality at a relatively high level with only a small texture degradation.

Assessments based on visual check also support the texture results shown in Table I. It can be observed that texture details on coloured vertex models are greatly "washed" away by mesh simplification. Texture details such as letters and brand logos are difficult to distinguish on those models. On the other hand, the letters and brand logos are clear to see and easy to distinguish from models textured using our 2D texture mapping.

Based on the results shown, it can be verified that our proposed texture method is effective in terms of preserving texture qualities on a simplified polygon mesh model.

Image Patch	Gradient Magnitude	Image Patch	Gradient Magnitude
Test Case: Lunch Bag			
Coloured Vertex		2D Texture Mapping	
Fig 7(g)	1.1292	Fig 7(j)	2.1064
Fig 7(h)	0.6633	Fig 7(k)	1.9707
Fig 7(i)	0.0870	Fig 7(l)	1.8888
Test Case: Backpack			
Coloured Vertex		2D Texture Mapping	
Fig 8(g)	3.9389	Fig 8(j)	5.7487
Fig 8(h)	2.6984	Fig 8(k)	5.4861
Fig 8(i)	0.4366	Fig 8(l)	5.3571

TABLE I. TEXTURE QUALITIES (COLOURED VERTEX VS. 2D TEXTURE MAPPING)

2) Improvements from using HD RGB Image: HD RGB camera can deliver RGB images with a higher resolution than that from the Kinect RGB camera. Thus such a camera is able to capture more details of real world. Based on this observation, model texture generated by using HD RGB image should also contain more texture details. To verify this assumption, the following experiment is carried out.

For each test case, a model textured using Kinect RGB image and a model textured using HD RGB image are generated and compared. The geometry of these two models are exactly the same and both models are textured using 2D texture mapping method to keep high texture qualities.

Table II shows the difference of texture quality between using Kinect RGB and HD RGB. For both test cases, textures generated by using HD RGB possess higher quality. Similar conclusions can also be made by checking model textures visually. As shown in Fig. 9 and Fig 10, the product brand logos on models (Fig. 9(b), Fig. 10(b)) with HD textures are clearer and sharper to visually. However, the logos reconstructed using Kinect RGB image are quite blurry and hard to recognise especially when the logo size is small (letters in Fig. 9(a)).

It is also worth noting that when reconstructing small-sized objects, photorealistic textures can be achieved. Fig 11(a) and Fig 11(c) are reconstructed model textures using HD RGB image while Fig 11(b) and Fig 11(d) are the corresponding parts directly cropped from HD RGB images. Similar level of details can be observed from both model texture and cropped HD RGB image.

(a) (b)

Fig. 9. Textured model and cropped image patch of a lunch bag (a): texture using Kinect RGB. (b): texture using HD RGB.

(a) (b)

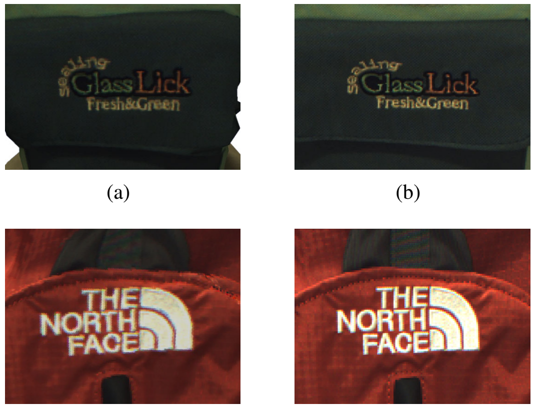
Fig. 10. Textured model and cropped image patch of a backpack (a): texture using Kinect RGB. (b): texture using HD RGB.

Image Patch	Gradient Magnitude	Image Patch	Gradient Magnitude
Test Case: Lunch Bag			
Kinect RGB		HD RGB	
Fig 9(a)	0.9467	Fig 9(b)	1.8462
Test Case: Backpack			
Kinect RGB		HD RGB	
Fig 10(a)	4.3045	Fig 10(b)	5.8296

TABLE II. TEXTURE QUALITIES (KINECT RGB VS. HD RGB)

VI. CONCLUSION AND DISCUSSION

To the best of our knowledge, our method is the first aimed at creating simplified 3D models with high quality textures. To deliver high quality model texture, an HD RGB camera is added to work with the depth camera of Microsoft Kinect™



(a) (b)

(c) (d)

Fig. 11. Textures using HD RGB camera vs. actual HD RGB image (a): HD texture of a lunch bag. (b): HD image of a lunch bag. (a): HD texture of a backpack. (b): HD image of a backpack.

sensor. Improved texture update scheme on an asymmetrical colour volume with a higher dimension than the geometry volume is presented. Given an output decimated 3D model, our 2D texturing method is able to maintain a high-level texture quality despite the degree of model simplification.

However, the texture quality is still limited by the dimension and size of the colour volume which is again constrained by the GPU memory. Our future work includes improving texture quality especially for large scene reconstruction and exploring better 2D texture map generation methods to achieve higher model rendering efficiency.

Acknowledgement

This work is supported by Smart Services CRC, Australia.

References

[1] Point Cloud Library. http://pointclouds.org/, 2015. [Online; accessed 24-June-2015]. 1, 2

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011. 1

[3] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 2

[4] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Proceedings of Robotics: Science and Systems*, 2013. 1

[5] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, Apr. 1992. 1

[6] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. 3-d motion and structure from 2-d motion causally integrated over time: Implementation. In *Proceedings of the European Conference on Computer Vision*, volume 1843, pages 734–750. 2000. 1

[7] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 1

[8] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1691–1696. IEEE, 2012. 1

[9] C. Fruh and A. Zakhor. 3d model generation for cities using aerial photographs and ground level laser scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–31–II–38 vol.2, 2001. 1

[10] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997. 4

[11] M. Gross and H. Pfister. *Point-Based Graphics*. Morgan Kaufmann Publishers Inc., 2007. 1

[12] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *Proceedings of the RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2010. 1

[13] D. Herrera C., J. Kannala, and J. Heikkil. Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 34(10):2058–2064, 2012. 2

[14] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In B. Buxton and R. Cipolla, editors, *Proceedings of the European Conference on Computer Vision*, volume 1064 of *Lecture Notes in Computer Science*, pages 117–126. Springer Berlin Heidelberg, 1996. 1

[15] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the ACM symposium on User interface software and technology*, pages 559–568, 2011. 1

[16] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS*, 21(4):163–169, 1987. 2

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. 1

[18] M. Niessner, M. Zollhofer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32(6):169:1–169:11, 2013. 1, 2

[19] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 335–342, 2000. 1

[20] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. *ACM Trans. Graph.*, 21(3):438–446, July 2002. 1

[21] F. Steinbrucker, C. Kerl, D. Cremers, and J. Sturm. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3264–3271, 2013. 1, 2

[22] J. Stuckler and S. Behnke. Integrating depth and color cues for dense multi-resolution scene mapping using rgb-d cameras. In *Proceedings of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 162–167, 2012. 1

[23] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 311–318, 1994. 1

[24] T. Weise, T. Wismer, B. Leibe, and L. Van Gool. In-hand scanning with online loop closure. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1630–1637, 2009. 1

[25] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013. 2

[26] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. In *Proceedings of the RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012. 1

[27] T. Whelan, L. Ma, E. Bondarev, P. H. N. de With, and J. McDonald. Incremental and batch planar simplification of dense point cloud maps. *Robotics and Autonomous Systems (RAS) ECMR '13 Special Issue*, 2014. 2

[28] Y.-Q. Yang, Q. Xiao, and Y.-H. Song. The investigation of 3d scene reconstruction algorithm based on laser scan data. In *Proceedings ot the International Conference on Machine Learning and Cybernetics*, volume 2, pages 819–823, 2010. 1

[29] Q.-Y. Zhou and V. Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics*, 33(4):155:1–155:10, 2014. 2

[30] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 371–378, 2001. 1

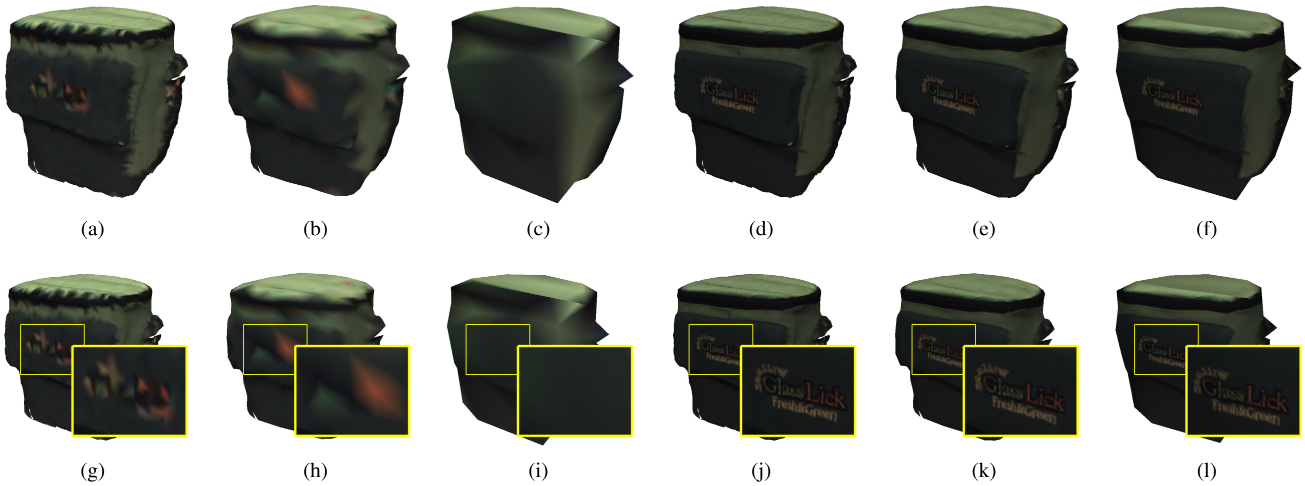


Fig. 7. Texture results of a lunch bag model (a): coloured vertex model with 13360 polygons. (b): coloured vertex model with 1336 polygons. (c): coloured vertex model with 134 polygons. (d): 2D textured model with 13360 polygons. (e): 2D textured model with 1336 polygons. (f): 2D textured model with 134 polygons. (g): image patch of (a). (h): image patch of (b). (i): image patch of (c). (j): image patch of (d). (k): image patch of (e). (l): image patch of (f).

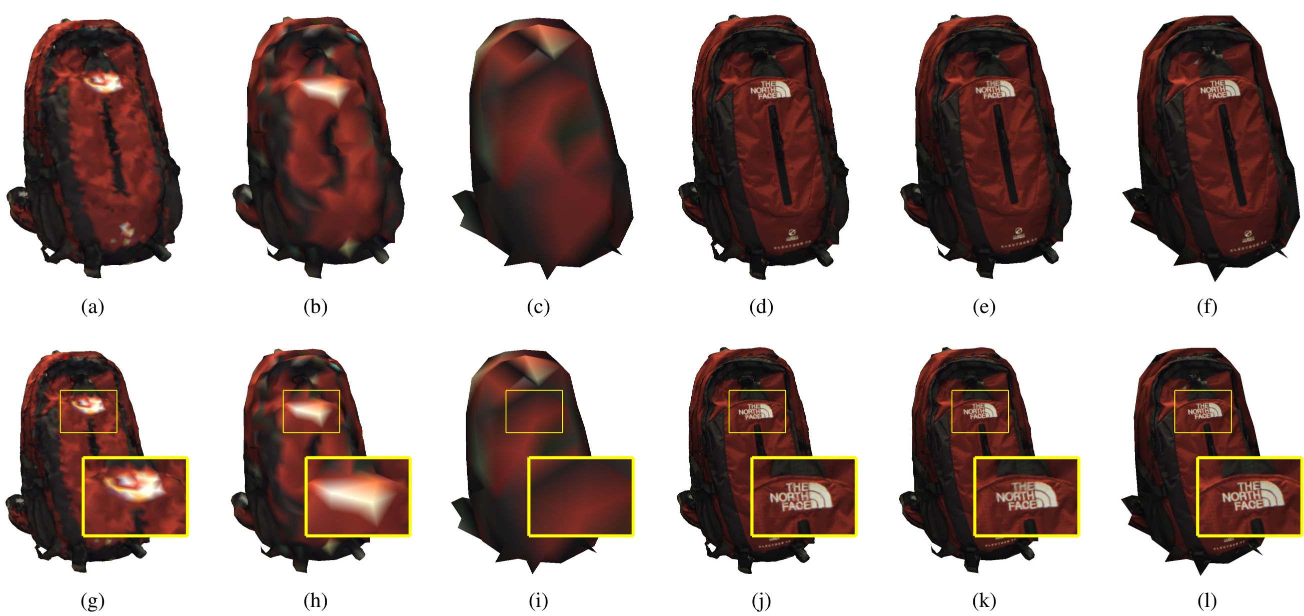


Fig. 8. Texture results of a backpack model (a): coloured vertex model with 39176 polygons. (b): coloured vertex model with 3918 polygons. (c): coloured vertex model with 392 polygons. (d): 2D textured model with 39176 polygons. (e): 2D textured model with 3918 polygons. (f): 2D textured model with 392 polygons. (g): image patch of (a). (h): image patch of (b). (i): image patch of (c). (j): image patch of (d). (k): image patch of (e). (l): image patch of (f).