3D Garment Digitisation for Virtual Wardrobe Using a Commodity Depth Sensor

Dongjoe Shin
School of Creative Technologies
University of Portsmouth
Portsmouth, PO1 2DJ, UK
dongjoe.shin@port.ac.uk

Yu Chen
Metail Ltd.
50 St. Andrew's Street
Cambridge, CB2 3AH, UK
yu@metail.com

Abstract

A practical garment digitisation should be efficient and robust to minimise the cost of processing a large volume of garments manufactured in every season. In addition, the quality of a texture map needs to be high to deliver a better user experience of VR/AR applications using garment models such as digital wardrobe or virtual fitting room. To address this, we propose a novel pipeline for fast, low-cost, and robust 3D garment digitisation with minimal human involvement. The proposed system is simply configured with a commodity RGB-D sensor (e.g. Kinect) and a rotating platform where a mannequin is placed to put on a target garment. Since a conventional reconstruction pipeline such as Kinect Fusion (KF) tends to fail to track the correct camera pose under fast rotation, we modelled the camera motion and fed this as a guidance of the ICP process in KF. The proposed method is also designed to produce a high-quality texture map by stitching the best views from a single rotation, and a modified shape from silhouettes algorithm has been developed to extract a garment model from a mannequin.

1. Introduction

A 3D model plays an important role in improving consumers' perception of virtual clothing. For example, the topology of a model can be used for physically based cloth simulation [7], a deformable garment model can be trained for draping it on various body shapes [8, 4], and the normal vectors of a model can help to deliver realistic rendering of fine details of garments in conjunction with an appropriate BRDF model [27]. More importantly, the recent development of VR technologies allows us to develop new user interfaces for immersive garment browsing. Therefore, the demand for practical garment digitisation solutions is expected to increase.

However, complex geometries found in a garment, such as creases, knots, and small buttons, make shape reconstruction challenging for vision based approaches, and in some cases it appears not feasible for a passive optical sensor, *e.g.* reflective sequins or veil and tulle textures. Image-based rendering proposed in [9] can be an alternative solution as it can synthesise a novel view at an arbitrary viewing position, but the lack of depth information makes it difficult to implement the interactions with other garments for future VR applications. Close range scanning with a hand-held active sensor could produce a fine model but this approach was ruled out because it normally requires a longer scanning time and the quality can be operator dependent.

We, therefore, believe the best outcome from a practical garment digitisation is a combination of an approximated (but complete) 3D geometry with a high-quality 2D texture map, and design algorithms to achieve this goal in an automated manner with a minimal hardware setup, *e.g.* simply pressing a button to accumulate depth maps and images from a rotating mannequin instead of using a multi-Kinect system used in [24]. Another aspect we considered in this project is the robustness of the reconstruction algorithm; we want to develop a robust algorithm that can work under a fast camera motion to facilitate the digitisation process. Also, the compatibility with a 2D garment digitisation process based on static photographs, where a few colour images are captured from discrete positions, has been taken into consideration as well, since it can reduce the required memory significantly and 2D garment segmentations can be exploited for 3D garment digitisation.

Multi-view stereo algorithms [23, 21, 25] can be one possible solution for garment digitisation. A problem of this is that it does not work well with feature-less garments (*e.g.* plain white top) and repetitive patterns as this system basically relies on a feature matching between different views. Cloud-based computing could be used with multi-view algorithms to ease the computing power and speed constraints

but we have found that many small and medium sized fashion retailers consider this as running cost which makes difficult to scale up the operation.

Shape from Silhouettes (SfS) [13] is another classic option for garment digitisation. This approach could fit well with a conventional digitisation process based on 2D photography, as a silhouette image can be extracted for each input image. However, it cannot capture concave regions. In addition, the voxel based volume reconstruction used in SfS will take a long processing time to produce an appropriate result for fashion garment. Alternatively, we can combine colour information with a classic SfS technique as in [26]. In theory, this can address some of the SfS limitations but it takes even longer processing time and requires many keyframe images to produce a reasonable result, which is against our requirement for scalability.

Shape reconstruction based on machine learning techniques looks promising. This approach basically learns shape priors from training data (*i.e.* a PCA of variant shapes of a 3D model) and uses the shape descriptors to estimate an individual model, *e.g.* a deformable 3D face model [15]. A similar idea has also been successfully used for creating a parameterised 3D body model [17] and 4D body model [19], and Pons-Moll *et al.* have recently proposed a new method that can extract a cloth model from 4D pose scan [18]. However, it is not clear how to define a few template models to cover a wide range of garment shapes, and deforming a template model seems not effective when a garment contains fine details. Another novel idea proposed by Berthouzoz *et al.* can construct a 3D garment model automatically from a set of 2D sewing patterns [1]. This can be seen as an advanced version of 3D cloth modelling using a CAD model such as [5]. This approach can create a cleaner 3D model, but it needs a complete set of 2D sewing patterns as an input and the error for parsing sewing pattern of their automatic algorithm is 32% which will be an issue for designing a practical processing pipeline.

Therefore, we have explored solutions using active vision sensors. Kinect is a cost-effective active depth sensor, which can deliver a high definition image with a good depth map without any special video processing hardware. A Kinect basically creates a 3D model by accumulating a sequence of depth maps from a different viewing angles. This means that we should be able to estimate the relative position of a moving sensor. Kinect Fusion (KF) [14] solves the camera tracking problem using a modified Iterative Closest Point (ICP) which exploits both a normal map and a vertex map to define initial correspondences for the ICP process [20]. Another advantage of using KF is that it addresses depth outliers appropriately by adopting a volumetric representation [3] and the fusion process can be done in real-time based on their parallel algorithm [10].

However, one practical problem we have found with a conventional KF is that its camera tracking only works well when the spatial separation between adjacent frames is small, *i.e.* the best output is expected when the camera translates and/or rotates slowly. This is mainly because the ICP process is heavily reliant on the initial guess. Therefore, a fast camera movement or any frame drops could make the iteration converge to a local minimum.

Figure 1. An example of conventional KF reconstruction: (a) actual garment image captured at a starting camera position; (b) rendered colour images using a model (c) with noticeable colour leakage from the background; (c) 3D geometry of (a) reconstructed by a default KF with a slow rotation (about 0.1 rpm).

Another problem of KF is poor quality of reconstructed colour. A conventional KF is basically optimised for capturing approximated shape rather than for creating better colours. Default KF only supports rudimentary colour fusion which simply averages aligned vertex colours without checking colour outliers. This means that a single colour outlier from a noise corrupted vertex could create noticeable colour distortion [see Fig. 1(b)]. Even if there are no outlier colours, fused results are generally blurry due to the insufficient spatial resolution of the voxels used in the reconstruction process. Therefore, the proposed solution is developed to tackle the two issues, *i.e.* a) fast and reliable reconstruction; b) better texture map construction.

2. Guided Kinect Fusion

2.1. System configuration

The proposed system basically consists of three components: a RGB-D sensor, a turntable, and a control unit. The system could be easily modified to have a separate imaging sensor, if a higher texture map is required.

A mannequin is placed on a turntable so that a user can style a garment properly before digitisation, and both the speed and the direction of rotation can be controlled through the control unit. The proposed system is also capable of reading the rotation angle in real time, but there might be some time latency between the triggering signal and the actual data reading, and this delay could be noticeable when the turntable rotates fast (*e.g.* greater than 5 rpm). Therefore, either software or hardware synchronisation is

required for a fast rotation.

2.2. Camera model

The camera geometry of the proposed system is relatively equivalent to that with a rotating camera around a fixed object. Therefore, we can parameterise the camera matrix in terms of a rotation angle.

Suppose that a camera and a turntable have separate coordinate systems denoted by $\mathcal{F}_{t,i}$ and $\mathcal{F}_c$, respectively. The subscription c and t represent a camera and a turntable, and as each rotation can define a new coordinate system, we use i to represent the coordinate at the i-th rotation, *i.e.* $\mathcal{F}_{t,0}$ is a coordinate system with no rotation. Assuming that $\mathcal{F}_c$ is our reference coordinate system, we can describe the camera matrix at a starting point as follows,

$$P_0 = \begin{bmatrix} K & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix}, \tag{1}$$

where $\vec{0}$ is a 3×1 zero vector, K is a 3×3 intrinsic camera matrix. Since we use fixed values for K in (1) throughout any single digitisation process, we can assume that it is constant.

If we can model the rotation of a turntable is a single axis rotation around a y axis, the i-th camera matrix formed after θ_i rotation is described by

$$P_i = \begin{bmatrix} K & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & \vec{t}_0 \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} R_y(\theta_i) & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & -\vec{t}_0 \\ \vec{0}^T & 1 \end{bmatrix}, \tag{2}$$

where I, $\vec{t}_0$, and $R_y(\theta_i)$ denote a 3×3 identity matrix, a 3×1 translation vector from the origin of $\mathcal{F}_c$ to the origin of $\mathcal{F}_{t,i}$, and a 3×3 rotation matrix about the y axis of $\mathcal{F}_c$, respectively. Thus, we can easily derive (1) from (2) by inserting zero rotation, *i.e.* $\theta_i = 0$.

However, the rotation axis of a mannequin does not always align with the y axis of a camera coordinate system. Therefore, we need an additional rotation (R_0) to compensate this difference. Consequently, our parameterised camera model is defined as

$$P_i = \begin{bmatrix} K & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} R_y(\theta_i)R_0 & \vec{t}_0 \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & -\vec{t}_0 \\ \vec{0}^T & 1 \end{bmatrix}. \tag{3}$$

2.3. Calibration

Since $\vec{t}_0$ and R_0 in (3) are constant during the digitisation process, we can estimate them in a separate sensor calibration stage. Let $\vec{t}_i$ and R_i be a translation vector and a rotation matrix from $\mathcal{F}_{t,i}$ to $\mathcal{F}_c$ after the i-th rotation. In that case, we can derive the following by rearranging (3),

$$\begin{cases} R_y(\theta_i)R_0 = R_i \\ (I - R_y(\theta_i)R_0)\vec{t}_0 = \vec{t}_i. \end{cases} \tag{4}$$

As R_y is determined by actual angle reading from a turntable, our calibration process basically needs to find out 7 parameters, *i.e.* 3 parameters from $\vec{t}_0$ and 4 quaternion parameters of R_0.

Although conventional KF easily breaks down with a fast motion or significant shape change, a relative camera pose (*i.e.* R_i and $\vec{t}_i$ in (4)) can be estimated reliably with a slow motion (*e.g.* < 0.1 rpm). In our experiments, we normally collect reliable estimations from 0 to 45-degree rotation and use these to find the solution using least square fitting. For example, the first equation in (4) can be rewritten using a quaternion notation, *i.e.*

$$\begin{bmatrix} \cos(\frac{\theta_i}{2}) & 0 & -\sin(\frac{\theta_i}{2}) & 0 \\ 0 & \cos(\frac{\theta_i}{2}) & 0 & \sin(\frac{\theta_i}{2}) \\ \sin(\frac{\theta_i}{2}) & 0 & \cos(\frac{\theta_i}{2}) & 0 \\ 0 & -\sin(\frac{\theta_i}{2}) & 0 & \cos(\frac{\theta_i}{2}) \end{bmatrix} \vec{q}_0 = \vec{q}_i, \tag{5}$$

where $\vec{q}_0$ and $\vec{q}_i$ are a 4×1 quaternion representation of R_0 and R_i, respectively. Thus, if we have n estimations we can form a $4n \times 4$ matrix, so that it can be solved by SVD. Similarly, we can estimate $\vec{t}_0$ using least square fitting after substituting the first equation into the second equation of (4), *i.e.* $(I - R_i)\vec{t}_0 = \vec{t}_i$.

After estimating R_0 and $\vec{t}_0$, we feed this information to KF with a rotation angle (θ_i) predicted by a linear regression from periodical turntable angle readings to improve the robustness of the tracking process and minimise the number of required depth maps. This proposed solution is designed to provide a better initial guess for the camera tracking whenever it is possible. We hence call this approach Guided Kinect Fusion (GKF).

3. Creating Seamless Texture

To complement KF with a proper colour reconstruction, we develop an additional workflow to create a unified texture map from multiple key-frame images in distinct camera views. This is basically similar to the seamless image stitching problem discussed in [6, 11], but slightly more complex as we need to solve the following sub-problems, *i.e.* a) how to extract a garment model from an initial foreground reconstruction; b) how to estimate vertex visibility (*e.g.* which image should not be used to define the colour of a vertex); and c) what is an optimal vertex classification that minimises the stitching seams. We have found that they are a closely related classification problem, and solve this using a graph-cut optimisation [26].

3.1. Garment model extraction

The first step of the seamless texture mapping is extracting a garment model from an initial reconstruction which normally contains some parts of a mannequin in addition to a true garment [see Fig. 2(a)].

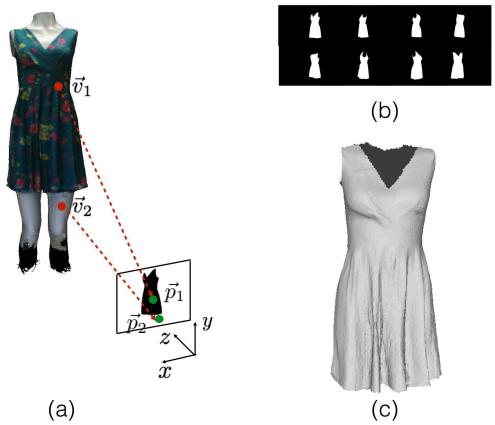

Figure 2. An example of an extracted garment model: (a) visibility test based on a garment mask obtained at $\mathcal{F}^c$; (b) example of garment masks from 8 viewing positions; (c) an extracted garment model from Algorithm 1.

One intuitive solution for the garment extraction is using a depth difference from a background depth map, which would be similar to background image segmentation [22], however this idea will not work with a tight-fit garment and it needs to store the background depth maps at every rotation. Instead, we have developed a simple algorithm using garment masks at a few different viewing positions [see Fig. 2(b)]. High quality garment masks are normally prepared in a conventional 2D digitisation process but the number of the masks are limited to a few keyframes. In our experiment, we used 8 garment masks from every 45° rotation.

Supposing that we have a projection matrix P_i and a garment mask $I_{g,i}$ for the i-th camera, it is simple to determine whether a vertex from an initial reconstruction $\mathcal{V}_{init}$ falls within the mask $I_{i,g}$ or not by back-projecting it using P_i [see $\vec{v}_1$ and its projection $\vec{p}_1$ in Fig. 2(a)].

Let $\mathcal{R}_i$ be a set of vertices whose projections are found inside the garment mask $I_{i,g}$, *i.e.*

$$\mathcal{R}_i = \{\vec{v}_i | \vec{v}_i \in \mathcal{V}_{init}, I_{g,i}(proj(P_i, \vec{v}_i) > 0)\}, \quad (6)$$

where $proj(P, \vec{v})$ returns a 2D pixel position by projecting a 3D vertex $\vec{v}$ using a projection matrix P.

In this case, we can extract a set of vertices in a garment model $\mathcal{V}_g$ by intersecting all $\mathcal{R}_i$ from t_k number of camera positions.

$$\mathcal{V}_g = \bigcap_{i=1}^{t_k} \mathcal{R}_i. \quad (7)$$

This is a simple binary operation having linear complexity. However, it assumes that there is no hole in the input mask, $I_{g,i}$. This is because the projection of a single hole from one garment mask into a 3D space will create a cone, which removes many valid vertices belonging to a garment model. In general, a garment mask frequently contains such holes.

Algorithm 1: Iterative noise removal

Input: $\mathcal{M}_g, t_i, t_s, t_h$
Output: $\mathcal{M}'_g$

```
M'_g := M_g;
for i := 0 to t_i do
    identify connected components in M'_g;
    add all connected components to a queue Q;
    sort Q by the size in a descending order;
    while |Q| > 0 do
        c = Q.pop();
        if |c| > t_s then
            break;
        else
            collect a histogram of IDs of the
              neighbour of c;
            find max ID;
            if |h(max_ID)| > t_h then
                swap ID of c with max_ID;
                update the neighbour of c;
            end
        end
    end
end
```

To address this, we modify (7) to intersect $\mathcal{R}_i$ selectively according to vertex visibility.

Suppose that $\mathcal{S}_i$ is a set of surface vertices visible from the i-th view. $\mathcal{S}_i$ is different to $\mathcal{R}_i$ in that it does not include any occluded vertex. However, it is not a subset of $\mathcal{R}_i$ as a visible vertex can be a part of a mannequin, *e.g.* $\vec{v}_2$ in Fig. 2(a) is in $\mathcal{S}_i$ but not in $\mathcal{R}_i$. S_i is easily defined by checking a vertex against a depth buffer at each camera position.

In the proposed selective intersection method, we perform the intersection (*i.e.* space carving process) when a vertex is in $\mathcal{S}_i$ and it is confidently visible. We measure the confidence based on the angle difference between a normal vector of a vertex and the viewing direction of a current camera, *i.e.* the smaller difference the more confident. This selective intersection can avoid the incorrect space carving.

This modified SfS method can give a reasonable result in many cases, but it is prone to contain small holes depending on the input parameters such as t_k. Instead of repairing this with a heavy optimisation algorithm at this stage, we devise a simple greedy iterative algorithm.

During the initial garment segmentation, we also assign an image ID[1], $l_i \in \{0, 1, \cdots, t_k\}$, to each visible vertex $\vec{v}_i$. As a result, each vertex in $\mathcal{V}_g$ has an image ID that can give us the most confident visibility, whilst the rest of the vertices in $\mathcal{V}_{init}$ do not have image ID. This can be considered as

[1] An image ID represents one of the keyframe images.

an initial vertex classification result for estimating optimal visibility explained in Sec. 3.2.

Small holes after the garment segmentation therefore can be seen as small connected components that are surrounded by other connected components. To remove these, we basically swap the image ID of a small connected component to the dominant image ID from its neighbourhood.

This approach could create another small component at each swapping operation, so we perform this operation recursively until there is no small component in the queue. Pseudo code for this algorithm is found in Algorithm 1, where $\mathcal{M}_g$ is an initial garment model, $\mathcal{M}'_g$ is a resulting model after noise removal, t_i and t_s represent a threshold for the number of iterations, the minimum size of a connected component, and $h(\cdot)$ is a function that returns the frequency of an image ID appearing at the component boundary. A result of Algorithm 1 is shown in Fig. 2(c).

3.2. Optimal visibility

Each vertex after the initial garment segmentation will have an image ID which can give the best colour for the associated vertex. However, it is not optimised for creating a good texture map. For example, if we have multiple connected components around complex geometry, they are highly likely to produce noticeable stitching seam. To minimise this visual artefact from over-segmentation, we should optimise the initial segmentation in a way that the boundary between segments is defined at a place where the colour difference is minimal.

This is basically reassigning segment ID of a vertex in a graph structure, and we can model the problem as a Markov network problem, in which a segment ID of a current vertex is affected only by directly connected neighbour vertices to enforce smooth transition between segments. In general, solving this type of problems is time-consuming, so we adopt a graph-cut algorithm to address this.

Let $\vec{l}$ represent a vector of segment IDs for every vertex in $\mathcal{V}_g$. In this case, we can think that our optimisation problem is finding out an optimal label vector $\vec{l}_o$. The total cost of a label vector $E(\vec{l})$ is defined as a weighted sum of data cost $E_d(\vec{l})$ and local smoothness cost $E_{smooth}(\vec{l})$

$$E(\vec{l}) = E_d(\vec{l}) + \lambda E_{smooth}(\vec{l}), \tag{8}$$

where λ is a weighting coefficient.

The data cost $E_d(\vec{l})$ in (8) is a sum of all vertex costs $d(l_i)$ that measure the cost of being classified as a given segment ID l_i, *i.e.*

$$E_d(\vec{l}) = \sum_{i=0}^{|\mathcal{V}_g|-1} d(l_i), \tag{9}$$

where l_i is a segment ID for the i-th vertex, *i.e.* $\vec{l} = [l_0, \cdots, l_{|\mathcal{V}_g|-1}]$, $0 \leq l_i < t_k$ and $l_i \in \mathbb{Z}$.

$d(l_i)$ in (9) is defined as

$$d(l_i) = 1 - \left\{ \frac{\sum_{f_j \in N_f(\vec{v}_i)} a(f_j, l_i) T(f_j, l_i)}{|N_f(\vec{v}_i)| a_{max}} \right\}, \tag{10}$$

where $N_f(\vec{v}_i)$ is a set of facets sharing a vertex $\vec{v}_i$, $a(f_j, l_i)$ is the area of a face f_j on a garment mask I_{g,l_i}, a_{max} is the maximum size of all projected facets, and $T(f_j, l_i)$ represent a binary indicator for visibility, *i.e.*

$$T(f_j, l_i) = \begin{cases} 1, f_j \text{ is visible from } I_{g,l_i} \\ 0, \text{otherwise.} \end{cases} \tag{11}$$

This means that we penalise assigning l_i to a vertex v_i if the visible area of its neighbour facets is small. For simplicity, we use a binary visibility cost in (11), but this can be a continuous function, *e.g.* a product of the confidence values of visible vertices.

Similarly, we can define the smoothness cost in (8) from local neighbours, *i.e.*

$$E_{smooth}(\vec{l_i}) = \sum_{\vec{v}_i \in \mathcal{V}_g} \sum_{\vec{v}_j \in \mathcal{N}_v(\vec{v}_i)} s(l_i, l_j), \tag{12}$$

where $s(l_i, l_j)$ is a function that estimates the cost of local smoothness of v_i, and $N_v(\vec{v}_i)$ is a set of neighbour vertices connected to a vertex $\vec{v}_i$,

Unlike the data cost, we define $s(l_i, l_j)$ using the colour difference between the seed and its neighbour vertices,

$$s(l_i, l_j) = \frac{1}{|\Omega(l_i)|} \sum_{\vec{p} \in \Omega} \max_{k \in \mathcal{C}} \left\{ |I_{k,i}(\vec{p}) - I_{k,j}(\vec{T}_{i,j}\vec{p})|/255 \right\}, \tag{13}$$

where $\mathcal{C}$ represents a set of colour channels of an input image, $\Omega(l_i)$ is a rectangular bounding box that encloses the projection of the first ring neighbour of vertex $\vec{v}_i$, and T_{ij} is a linear transform that maps a pixel in the k-th colour channel of the i-th input image I_{ki} to the corresponding pixel in the j-th input image I_{kj}. We solve this optimisation using a classic multi-class graph cut (GC) algorithm developed based on the α-expansion [2].

4. Experimental results

To compare the shape reconstruction of GKF with a conventional KF method, we scan different types of garments under the same condition. In this test, we set the distance from the sensor to a garment between 1.5 and 2 metres and voxel resolution is fixed for all test garments. To minimise the effect from background scene explicitly, we capture the reference depth map during the sensor calibration and use it to mask out background depth values.

Four garments of different types (*i.e.* dress, shorts, trousers, and top) are scanned at different rotating speeds

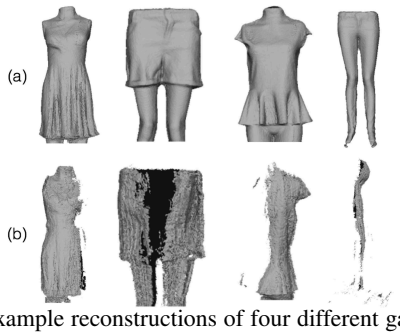

Figure 3. Example reconstructions of four different garments at 5 rpm: reconstructions of GKF (our approach) (a) and conventional KF (b)

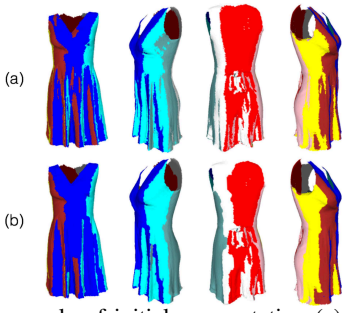

Figure 4. An example of initial segmentation (a) and optimised segmentation (b).

(*i.e.* 1, 3, and 5 rpm) to demonstrate how robust the proposed algorithm is. Some of the reconstruction results obtained from 5 rpm can be found in Fig. 3. When the rotation is slow[2] (*i.e.* $\leq$ 1 rpm), both methods can manage to deliver good reconstructions, in which there is no considerable difference between GKF and KF by visual inspection. However, KF starts to break down at 3 rpm and it is worse at 5 rpm [see Fig. 3(b)], whilst the proposed method can produce a good reconstruction result regardless of the high rotating speed.

As mentioned earlier, the proposed method need to extract a true garment model from an initial reconstruction. This process can also give us an initial vertex classification result as a by-product. Figure 4(a) shows an example of an initial garment segmentation, which was optimised by the proposed graph cut method; see Fig. 4(b), where each colour on a garment model represents a different keyframe image ID and the corresponding. It is worth noting that GC does not smooth seam line but it minimises the colour tone difference across the stitching seam. Thus, the coarser segmentation is generally the better.

Final rendering results of a reconstructed 3D garment model (*i.e.* colour and shape reconstruction of a garment) are shown in Fig. 4. The final results with and without the seam optimisation are shown in Fig. 4(a) and (b), and the actual colour image captured from Kinect is also found in

[2] 1 rpm is slow considering a 2D garment digitisation process where a team of trained operators can capture all required images in 30-90 sec.

Figure 5. Rendering example of a final 3D garment model at different view: (a) after the optimisation (see the difference around knots); (b) before the optimisation; (c) actual image from Kinect

Fig. 4(c). It is observed that the seam optimisation step can reduce the colour inconsistency around the subtle regions of the garment models, *e.g.* knots. Please also note that there is a slight geometry difference between Fig. 4(a) and (b), because of the final mesh smoothing applied to smooth out the jaggy meshes at the boundary of a 3D model.

5. Conclusions

This paper presents a practical solution for 3D garment digitisation. In order to retrieve the garment model rapidly and robustly, we adopt a Kinect sensor and expand its reconstruction pipeline.

The proposed method linearly approximates the camera motion, and calibrates a sensor accordingly before 3D garment digitisation. During the digitisation, we can predict a new camera position from the previous angle readings and feed this information to the default KF workflow. This additional information makes an ICP-based camera tracking more robust. A texture map for the reconstructed garment model is created by stitching a few texture images. In order to minimise the stitching seam, we convert the stitching process to another optimisation process in a graph structure and solve this using a multi-class graph cut algorithm.

From our experiments, we demonstrate that the proposed method can deliver a better rendering result. However, it cannot recover some concave areas, *e.g.* between legs, and the ICP is sensitive to the calibration results which might not be constant during rotation. To improve this, it is better to revise the ICP process. Since the proposed setting can reduce the number of unknowns in the ICP to 4 (*i.e.* a rotation normal and angle) this can help to improve the stability.

If a strong directional light is used, the proposed method might create visual tone differences between final segments. At the moment, we use a simple colour blending to smooth the difference out. This needs to be modified by either a proper light model, or advanced seamless image stitching techniques using colour gradients, such as Poisson image stitching [16] or Laplacian blending [12]. In addition, it should be noted that this paper is based on our preliminary test results to check the feasibility of using Kinect for 3D garment construction and more rigorous tests are left for the future research.

References

[1] F. Berthouzoz, A. Garg, D. M. Kaufman, E. Grinspun, and M. Agrawala. Parsing sewing patterns into 3d garments. *ACM Transactions on Graphics (TOG)*, 32(4):85, 2013.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 377–384 vol.1, 1999.

[3] 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*, SIGGRAPH '96, pages 303–312, New York, NY, USA, 1996. ACM.

[4] E. de Aguiar, L. Sigal, A. Treuille, and J. K. Hodgins. Stable spaces for real-time clothing. *ACM Trans. Graph.*, 29(4):106:1–106:9, July 2010.

[5] A. Divivier, R. Trieb, A. Ebert, H. Hagen, C. Gross, A. Fuhrmann, V. Luckas, et al. Virtual try-on topics in realistic, individualized dressing in virtual reality. In *Proc. Virtual and Augmented Reality Status Conf.* Citeseer, 2004.

[6] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 341–346, New York, NY, USA, 2001. ACM.

[7] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun. Efficient simulation of inextensible cloth. *ACM Trans. Graph.*, 26(3), July 2007.

[8] P. Guan, L. Reiss, D. A. Hirshberg, A. Weiss, and M. J. Black. Drape: Dressing any person. *ACM Trans. Graph.*, 31(4):35–1, 2012.

[9] S. Hauswiesner, M. Straka, and G. Reitmayr. Virtual try-on through image-based rendering. *IEEE transactions on visualization and computer graphics*, 19(9):1552–1565, 2013.

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

[11] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, July 2003.

[12] A. Levin, A. Zomet, S. Peleg, and Y. Weiss. Seamless image stitching in the gradient domain. In *Computer Vision-ECCV 2004*, pages 377–389. Springer, 2004.

[13] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 369–374, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[14] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[15] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. In *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, pages 296–301. IEEE, 2009.

[16] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics (TOG)*, 22(3):313–318, 2003.

[17] L. Pishchulin, S. Wuhrer, T. Helten, C. Theobalt, and B. Schiele. Building statistical shape spaces for 3d human modeling. *Pattern Recognition*, 67:276–286, 2017.

[18] G. Pons-Moll, S. Pujades, S. Hu, and M. Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM Transactions on Graphics,(Proc. SIGGRAPH)[to appear]*, 1, 2017.

[19] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (TOG)*, 34(4):120, 2015.

[20] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152, 2001.

[21] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3d. *ACM Transactions on Graphics*, 25(3):835–846, July 2006.

[22] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[23] C. Strecha, W. Von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[24] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan. Scanning 3d full human bodies using kinects. *IEEE transactions on visualization and computer graphics*, 18(4):643–650, 2012.

[25] D. Vlasic, P. Peers, I. Baran, P. Debevec, J. Popović, S. Rusinkiewicz, and W. Matusik. Dynamic shape capture using multi-view photometric stereo. *ACM Transactions on Graphics (TOG)*, 28(5):174, 2009.

[26] G. Vogiatzis, P. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 391–398 vol. 2, June 2005.

[27] S. H. Westin, J. R. Arvo, and K. E. Torrance. Predicting reflectance functions from complex surfaces. *SIGGRAPH Comput. Graph.*, 26(2):255–264, July 1992.