Received 25 August 2022, accepted 8 September 2022, date of publication 14 September 2022, date of current version 22 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3206471

RESEARCH ARTICLE

Improved Mesh Reconstruction With an Edge Quality Enhancement Using Multiple Inward Depth Streams

SASADARA B. ADIKARI[1], NALEEN GANEGODA[2], RAVINDA MEEGAMA[3], AND INDIKA L. WANNIARACHCHI[1]
[1]Department of Physics, University of Sri Jayewardenepura, Nugegoda 10250, Sri Lanka
[2]Department of Mathematics, University of Sri Jayewardenepura, Nugegoda 10250, Sri Lanka
[3]Department of Computer Science, University of Sri Jayewardenepura, Nugegoda 10250, Sri Lanka

Corresponding author: Indika L. Wanniarachchi (iwanni@sjp.ac.lk)

This work was supported by the University of Sri Jayewardenepura, Sri Lanka, under Grant ASP/01/RE/SCI/2019/65.

ABSTRACT This paper presents a complete 3D model reconstruction of an object with edge quality enhancements using multiple inward depth sensors to create closed 3D model. In the reconstruction pipeline, a pattern of incorrect depth information was consistently observed at the edges of the mesh generated by each sensor stream, which we refer to in this paper as a ''drift-effect''. In order to mitigate this, we introduced a filtering approach with a localized threshold value that is used to remove drift faces from a mesh. We also present a mesh stitching technique incorporating Laplacian mesh smoothing to generate a closed 3D model from the smoothened multi-view meshes. The primary objective of this research was to implement a system that could capture a static physical object with a minimum scan time and at a low cost while retaining accurate details in the model. For the demonstration, we used four Intel RealSense D435 depth sensors to capture a clothing article that can be imported into a virtual dressing room application. We captured the entire object within three seconds, which is quicker than traditional techniques such as table rotation and sensor rotation. The final results indicate that the system is able to provide a satisfactory reconstruction of a clothing model which can be used in a live virtual dressing room application.

INDEX TERMS 3D reconstruction, augmented reality, depth sensors, point-cloud, RGB-D, virtual reality.

I. INTRODUCTION

In the present-day augmented and virtual reality applications are used in various fields, including in arts, computer games, training platforms, virtual dressing rooms, virtual heritage, etc. Commonly, 3D models are built using 3D modeling tools such as Maya, 3DS Max, or Blender [1]. However, this process involves tedious and time-consuming work and requires 3D modeling engineers who are specially trained for such work. Learning how to work with the above tools is cumbersome for non-specialized professionals [2] who are focused on various other fields. Additionally, there are applications that require the frequent creation of many 3D models of multiple physical objects. From a practical standpoint, it is not efficient to use 3D modeling software when creating such a large number of 3D models from physical objects. Instead, this issue can be addressed using a 3D reconstruction system.

We can find various 3D reconstruction techniques including image-based, single, and multiple depth sensors. However, such 3D reconstruction implementations are expensive [3], less accurate, [4], [5] and take time to scan [6] and reconstruct the 3D models, which significantly limits the accessibility of 3D reconstruction technology. As an example, the frequency of use for such a reconstruction system would be high in an apparel store that needs to generate 3D scanned clothing objects every time a new article is added to their inventory. The aforementioned factors further drive up the cost of reconstruction for such use cases.

The associate editor coordinating the review of this manuscript and approving it for publication was Michele Nappi.

In the process of 3D reconstruction, there are limitations to removing noise and incorrect depth data from the raw sensor output. This paper describes the additional techniques we innovated to increase the quality of the 3D reconstruction using a generic depth sensor array configuration. The implemented system is compatible with any provided color and depth streams as it only relies on an RGB-D image array. A major advantage of using the RGB-D image array is the ability for this system to be connected to any type of sensor, provided that the sensor can capture and provide the color and depth streams.

When generating a 3D object using physical objects, we identified the sequence of steps that needed to be carried out, including the depth data acquisition, manipulation, and conversion of depth data into three-dimensional meshes, and the subsequent transformation of the end result into a merged 3D mesh. This process becomes tedious because, even with precise calibration and position alignment of sensors, the data may become contaminated with invalid data points. This occurs mainly due to the hardware limitations of various sensor technologies and the sensor manufacturer.

As most 3D reconstructions are based on RGB-D cameras [7], we used four stereoscopic sensors (Intel RealSense D435) as our primary depth sensor module for 3D reconstruction. The Intel RealSense D435 sensor was one of the latest sensor models that were available at the time this research was conducted. The sensor uses infrared rays to capture depth information unlike legacy stereoscopic sensors, which use visible light. We used a parallel data acquisition technique to speed up the depth capturing process which was facilitated by the fact that the selected sensor model supported parallel connections with the host out of the box.

The reconstruction process consists of three main stages: depth data acquisition, mesh creation transformation and alignment, and final 3D object creation using multi-view mesh stitching. In the sensor pose estimation stage, we calculated the real-world sensor coordinates using image processing to determine the translation and rotation of each depth sensor placed at different angles. Then we focused on depth noise filtering and the removal of invalid depth data from the output sensor streams to generate a satisfactory point-cloud. Depending on the sensor model, the output of depth points and the generated mesh has a tendency to be noisy, justifying the use of multiple filters to improve depth data quality. One of the issues we observed in the mesh was the drift-effect which we were able to reduce at the post-processing stage. Finally, we proposed an algorithm to create a stitched and closed 3D model from the meshes generated by each sensor.

From this paper, we contribute a novel algorithm for 3D model reconstruction using depth sensors to refine the drifted edges of a mesh and a stitching technique to create a closed 3D model. In this paper, the related work section will discuss the various 3D reconstruction research that has been implemented in previous literature. Then the methodology section will discuss how we implemented the system. Finally, we present the data that was obtained along with the conclusions that were made from this research.

II. RELATED WORK

A. SENSOR AND TABLE ROTATION TECHNIQUE

It has been reported in recent literature that most 3D scanning and reconstruction procedures are carried out using a single sensor with the aid of a rotation table. According to Haleem *et al.* [8], the rotation scanner they implemented had an operation time of approximately ten seconds and had a weight limit. Table rotation is a common technique used in 3D reconstruction, and several other attempts have been implemented using the same technique to capture a 3D model [9], [10]. In 2014, Popescu and Raluca implemented a 3D reconstruction technique using a Kinect sensor, with a mechanism that utilized sensor rotation instead of table rotation [11]. However, for each of these implementations, the usage of table and sensor rotations inherit multiple problems, including the complexity of the scanning setup and its mechanical components. Rotating the sensor around the object can also cause issues due to the sensor being mobile. Furthermore, considerable human effort is required to capture all aspects of the physical object when considering handheld sensor rotation techniques. These scanning techniques also take longer to capture the entire object that needs to be scanned. As a result, the above methods are less attractive for a use-case involving scanning numerous physical objects (i.e., clothing articles for virtual dressing rooms).

B. MULTI-VIEW 3D RECONSTRUCTION USING IMAGES

Structure From Motion (SFM) is a common passive image-based [12], [13] technique that is used to capture an object and create a point-cloud representation of it. This method uses an array of images taken at different angles of the object. It is a low-cost and accurate technique that can be applied to most use cases. In 2022, according to Mi and Gao, ‘‘image-based 3D reconstruction is more widely used due to its low environmental requirements’’ [14]. In 2017, Merras *et al.* [2] implemented a 3D reconstruction system using genetic algorithms with the help of the SFM technique. The research shows quality output compared to previous literature related to SFM. Yet, this method could not be applied to objects with similar features/textures unless we provide the rotation and translation of each camera system. This problem was addressed to some extent by El Hazzat *et al.* [15]. Also, image-based method estimations are computationally expensive and can consist of many outliers [16].

C. MIRRORS FOR 3D RECONSTRUCTION

In 2018, Nguyen implemented a 3D reconstruction method using a single depth sensor along with multiple mirrors [17]. However, in such implementations, the occlusion effect can cause a certain level of unreliability when scanning physical objects. Furthermore, the inclusion of mirrors in the infrastructure introduces additional complexity to the system due to variables such as object size, external reflections, mirror size, etc. Our proposed method uses multiple sensor arrays to eliminate the need for any sensor movement during the data acquisition phase. This method does not contain any mechanical components as seen in some previous implementations. A similar implementation was created by Hu *et al.* [18] and a full-body scanner using mirrors was implemented by Xie *et al.* [19]. Tan *et al.* noted that mirrors introduce a significant number of errors to depth sensing and 3D reconstruction [20].

D. RECONSTRUCTION WITH MULTIPLE SENSORS

3D model reconstruction using multiple sensor arrays produces different sets of problems, including the calibration and identifying the sensor poses along with each other and the merging of different meshes into a single closed mesh. According to Auvinet *et al.* [21], two-camera configurations did not provide sufficient details of a person's surroundings. Therefore, we utilized four inward sensor setups to increase the visibility of the object we were interested in. In 2012, Tong *et al.* [22] proposed a 3D scanning system using multiple Kinects positioned at varying heights. In this research, the author used several sensors to capture different object levels instead of all angles at once, demanding the use of a rotation table. Eventually, the object needed to be in a static pose in order to be fully captured, which resulted in the process being time-consuming.

E. MESH ALIGNMENT FOR MULTIPLE SENSORS

To align a point-cloud generated by each sensor, we attempted to use the most common mesh aligning method. Iterative Closest Point (ICP) is a common mesh alignment technique that computes the similarity of points in the mesh and uses this to align each of the meshes [23]. However, its main drawback is that it cannot align the meshes as expected if the sensor-generated meshes do not contain any similar depth points (vertices). This can happen if the sensor provides incorrect depth points in the region of interest (ROI). Research that was conducted by Kim *et al.* implemented a multi-view 3D reconstruction system with an edge line calibration system [16]. This research demonstrated good results and involved the use of Poisson surface reconstruction to create a closed mesh. The Poisson reconstruction is not suitable for every single object model as it introduces incorrect shapes into the final model. Taking these facts into consideration, we used an image reference-based calibration method to align meshes generated from a multiple-sensor array. Similar work has been completed by Kowalski [24]. They developed a fast and inexpensive 3D data acquisition system with multiple Kinect V2 systems using a combination of image-based calibration and ICP. The use of ICP, in this case, did not negatively impact the results due to the accuracy of the Kinect V2. However as mentioned above, the ICP algorithm could potentially deform the model if provided with less accurate depth data. In their implementation, it is necessary to calibrate and estimate the camera poses prior to the object scanning phase. In contrast, our calibration technique only utilized image processing to automate the entire reconstruction process to be performed at the data acquisition stage by executing the calibration and the object scanning simultaneously.

F. GENERATE CLOSED STITCHED MODEL

To use a model in an augmented or virtual reality application, it is necessary to create a stitched and closed 3D model. If the model contains unstitched points, there is a chance for it to display unnatural deformations during interactions inside the application. We used a merging technique similar to the work done by Brandao *et al.* [25], but with the addition of Laplacing mesh smoothing [26] to generate a better output than the above methods.

G. SENSOR PLUGIN MODE

We also incorporated the ability of our system to plug in any depth sensor instead of being limited to a single sensor model. This advantage introduced the possibility of swapping out the sensor for a better depth sensor module and utilizing the new sensor's superior performance to generate an improved clothing model. We designed our implementation to operate using parallel data acquisition instead of series data acquisition to increase the time efficiency of 3D scanning. As a prerequisite, the sensor model should support the parallel handshake with the computer, and it should be able to capture depth and color frames from all the sensors at once. Most sensor manufacturers have defined these values beforehand. Our selected Intel RealSense D435 sensor can perform parallel data acquisition [27] with up to four sensors without a reduction in frame rate.

III. METHODOLOGY

We wanted to be able to use the 3D models generated by our 3D reconstruction implementation for an augmented reality application. We chose a clothing article as the physical object and our goal was to generate a 3D model of it for use in a virtual dressing room application that we had already implemented [28]. This section discusses the process of creating a 3D model that we followed to generate the 3D clothing article. Figure 1, shows the flow diagram of the implemented system.

A. CAPTURE DEPTH & COLOR FRAMES ACQUISITION

1) SENSOR CALIBRATION

Calibration of the sensor is a necessary step for any 3D reconstruction system [29]. In most sensor variants, a hardware design specification indicates the calibrating methods. For our implementation, the guidelines provided by Intel were used to calibrate the Intel RealSense D435 sensors [30].

2) EQUIPMENT SETUP

It is necessary for the sensor setup to cover all the views of the object since this is an essential step when capturing each surface of the physical object from all angles. Our system used

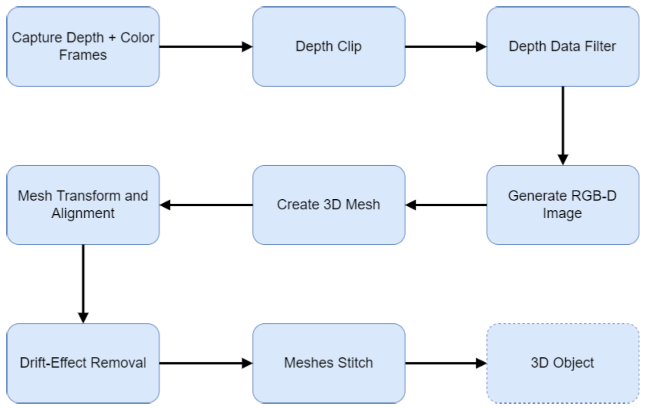


FIGURE 1. Flow-diagram of the 3D reconstruction system.

four Intel RealSense D435 sensors in 360° view positions placed approximately 90° apart and kept the object within 1.5 *m* of all the sensors (see Figure 2). Increasing the number of sensors causes the frame rate to reduce, [27] and in order to obtain the required number of frames, we need to expose the object to the sensor for a longer period of time. However, it is important to consider that longer exposure times introduce the possibility of unintended effects on the sensor output. This resulted in an effort to find the minimum number of frames required to obtain saturated results. With that, we collected the required color and depth frames to generate the 3D mesh.

B. GENERATE THE 3D MESH

To create the 3D mesh, it is necessary to generate the RGB-D images using depth and color frames. However, each frame contains depth information that is not necessary, and processing this data would consume additional computing resources. This is accomplished by removing unwanted depth points from the depth frame and color points from the color frame. We removed these unnecessary depth points by selecting a depth clipping distance threshold and using a simple depth clipping algorithm 1 as shown below.

```
Algorithm 1 Algorithm for Depth Clipping
Input: Depth frame, color frame
Output: Clipped → depth frame, color frame
1: Initialization: n = height × width
2: for $i = 0$ to $n - 1$ do
3:     if ($z \geq clip\_threshold$) then
4:         Set $color\_frame[i] = RGB(0, 0, 0)$
5:         Set $depth\_frame[i] = Z(0)$
6:     end if
7: end for
```

Thereafter, we generated the RGB-D image using the clipped depth and color frames. We created the 3D mesh using the RGB-D image by converting it into a colored point-cloud using Open3D library [31] and mapped each point with the color frame's pixel indices to generate the faces accordingly.

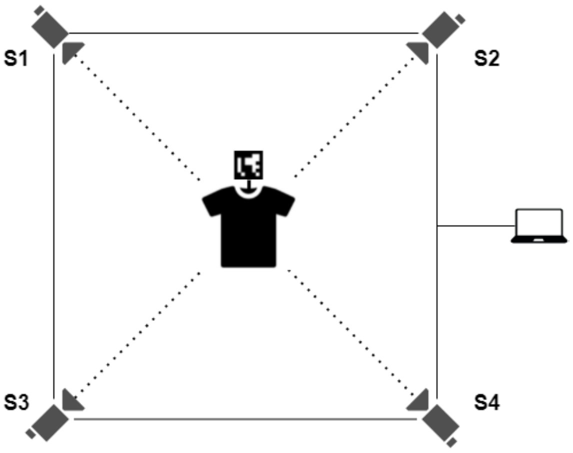


FIGURE 2. Multiple inward sensors arrangement.

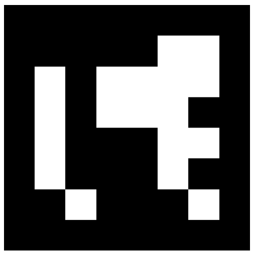


FIGURE 3. ArUco marker (Dictionary 6 × 6 250 - id 1) used in calibrating sensor position.

C. MESH TRANSFORMATION AND ALIGNMENT

When the scanning platform is equipped with multiple stereo depth sensors, it is necessary to estimate the positions of all the sensors according to their location and rotation relative to a reference plane.

For the sensor pose correction, it was imperative to detect the relative positions of each sensor and correct the offset using a specific transformation matrix using rotation and translation matrices for each sensor. In a generic multi-view construction system, if a sensor's position or alignment is changed, it becomes necessary to re-calibrate and re-estimate the poses of the sensor system. However, we used a simple placement strategy for sensors to mitigate this problem. We used two ArUco markers (dictionary 6 × 6 250 with id '1' as shown in Figure 3) on the two sides of a plane with an exact 180° rotation and complete overlap of the two images as seen in Figure 4. At the bottom of ArUco marker, we placed the object we needed to scan. Figure 5 shows a view of the actual setup of the system. This marker and sensor placement is helpful to capture the ArUco markers from all the sensors at the same time. Each sensor's pose estimation is performed at the moment data is collected for reconstruction instead of using a separate calibrate phase.

The displayed images and its four corners were identified from each sensor's color stream using an image processing technique. These corner points were then converted into 3D

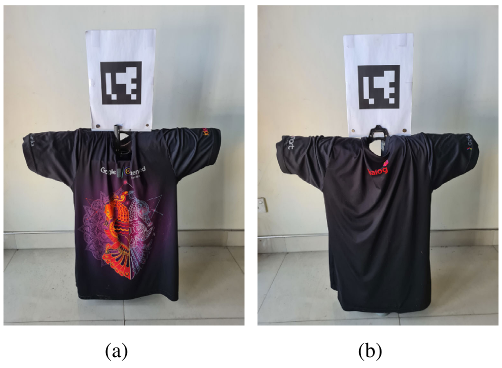


FIGURE 4. (a) front (b) back, of the clothing article with ArUco markers.

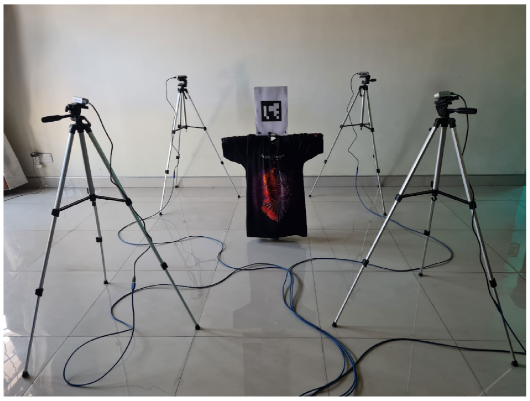


FIGURE 5. Actual view of the sensor setup with a clothing article.

space coordinates using (1),

$$\vec{p} = \left(\frac{(x - c_x) \times \vec{z} \times ds}{f_x}, \frac{(y - c_y) \times \vec{z} \times ds}{f_y}, \vec{z} \times ds \right) \quad (1)$$

where (x, y) is a point of the color frame. The z value refers to the depth of point (x, y) in the depth frame along the $x - axis$ and $y - axis$. The focal lengths of the camera are f_x and f_y, while c_x, and c_y denote the camera's optical center in pixels along the $x - axis$ and $y - axis$. ds is the depth scale of the sensor module. We can derive each focal and optical center of the sensor using the sensor's depth intrinsic parameters

Using the above corner derivation, we could generate all four corners per sensor, (P_0, P_1, P_2, P_3) as displayed in Figure 6. Yet, these corner values were not consistent due to the sensor depth error with time. To resolve this, we collected more than one frame from each sensor and calculated the cumulative moving average (CMA) using (2) for each depth pixel while removing invalid outliers if $any[p_n] = 0$. We used that CMA filter to find accurate space coordinates for all four corners, including the center point of the ArUco marker.

$$C_n = \frac{(n-1).C_{n-1} + p_n}{n} \quad (2)$$

Here, C_n is the CMA of n frames where $(n > 1)$ and p_n is a corner value of the n^{th} frame.

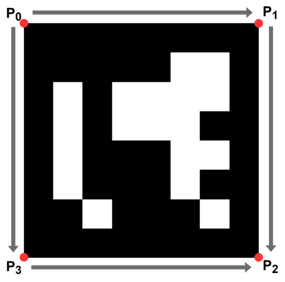


FIGURE 6. ArUco marker with four corners and defined vector directions.

Vectors x, y, and z for rotations are calculated as:

$$\vec{x} = P_3 - P_0 \quad (3)$$

$$\vec{y} = P_1 - P_0 \quad (4)$$

$$\vec{z} = \frac{\vec{x} \times \vec{y}}{|\vec{x} \times \vec{y}|} \quad (5)$$

From the four corners P_0, P_1, P_2, P_3, we calculated the center point P_c of the ArUco marker. With this, we can write the transformation matrix M as,

$$M = \begin{bmatrix} x_1 & y_1 & z_1 & 0 \\ x_2 & y_2 & z_2 & 0 \\ x_3 & y_3 & z_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 & -P_{cx} \\ 0 & 1 & 0 & -P_{cy} \\ 0 & 0 & 1 & -P_{cz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

For the individual sensor, we calculated the transformation matrix (M) using the rotation (R) and translation (T) matrices derived from each sensor. Here, the translation matrix (T) can be determine by the distance from the sensor to the center of the ArUco marker and the rotation matrix (R) can be determined using the orthogonal Procrustes problem method [32].

During this process we captured and generated the 3D point-clouds of the object that connected to the ArUco marker from each sensor. We used the color image pixel indices along with point-cloud indices to generate the mesh (point-cloud with faces) from the point-cloud. Then the matrices (M) from all the sensors were saved and the mesh transformation was applied for each mesh generated by its respective sensor to bring all the meshes into one reference plane. Additionally, we added 180° rotation components to sensors S3 and S4 as they were located on the opposite side of the sensors S1 and S2 (See Figure 2). This generated four aligned but separated meshes similar to the physical object that was scanned.

D. DRIFT-EFFECT REMOVAL

After generating the transformed meshes, we observed an issue where each mesh had a tendency to "drift" on faces at the edge. In the Intel RealSense D435, this drift-effect can be seen prominently when the surface is located close to another surface. The principle of the depth measuring technique used in the RealSense D435 sensor is stereoscopy and it uses infrared rays (IR) to generate depth results. To minimize the above "drift-effect" on the mesh edge-line, we attempted to identify the parameter causing the drift. The proposed method

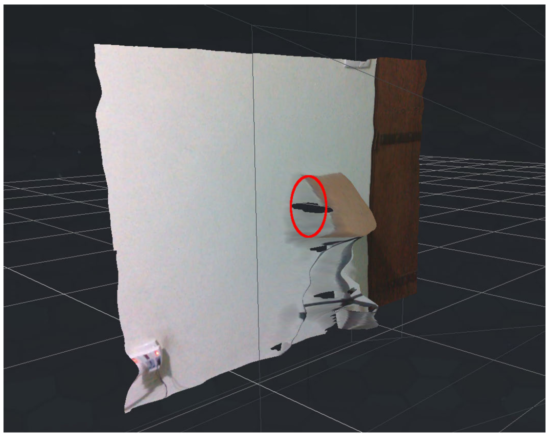


FIGURE 7. Scanning plane is near the wall. The red color oval shows the drift-effect of the depth data.

reduces the disparities of drift-effects and generates a filtered mesh by removing the unnecessary faces.

To remove these unnecessary faces in meshes we generate previous section, a localization threshold (Th) was associated with each face by comparing the distance between every three points in a face. We propose (7) to set the localization threshold as shown below.

$$\text{Th}_m = \text{SC} \times \sqrt{argmin\left[\left|z_q\right|, \left|z_r\right|, \left|z_s\right|\right]} \tag{7}$$

where Th_m is the threshold value for the m^{th} face, SC is the "Sensor Constant" which needs to be defined per sensor module, and z_q, z_r, and z_s denote the distances of the three vertices of the m^{th} face from the origin (0, 0, 0).

This value was compared with any edge size of the m^{th} face as shown in (8).

$$z_m = \text{argmax}\left(\left|z_q - z_r\right|, \left|z_s - z_r\right|, \left|z_s - z_q\right|\right) \tag{8}$$

Here, z_m is the maximum edge distance of the m^{th} face.

1) GENERATING THE SENSOR CONSTANT (SC)

To reduce this drift-effect, we used a localization-based threshold to create the facets and derived an equation with a term of "SC" which is specific to the sensor model. It is necessary to identify this "SC" value for a particular sensor.

A syntactic ground truth mesh is created using an angled plane kept near a wall to determine this drift-effect, as shown in Figure 7. A mesh is then generated by scanning the plane without applying any modifications. Subsequently, by removing the unwanted faces in the mesh, a syntactic mesh is generated for comparison with the resultant meshes created by varying the "SC"; at the point $z = \text{Th}$. For this purpose, we used a software called MeshLab [33] to generate the ground truth according to the actual measurements. The mesh is used to calculate faces and vertices to find the most suitable mesh generated by varying the "SC".

Finally, using the algorithm we removed the face if:

$$z \geq \text{Th} \tag{9}$$

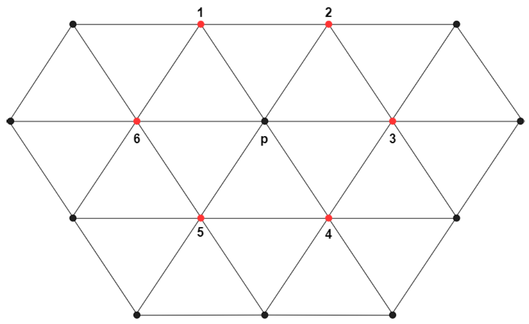


FIGURE 8. Neighbor points around vertex 'p'.

E. MESH MERGING - STITCHING ALGORITHM

In this phase, we fixed the separation of each mesh to create a single closed mesh. These separations can occur during the position calibration phase and due to disparities in-depth readings. To rectify this, we proposed a technique that identified the edge line of each mesh by searching for all the vertices that have less than six neighboring vertices, which were considered to be edge line vertices.

As seen in Figure 8, points 1 and 2 do not possess six neighboring vertices as seen in the others. This technique allows us to identify the edge line vertices when the mesh is built with a triangle mesh. Moreover, the KD-tree algorithm [34] is used to get the nearest point of the neighboring mesh and to find the closest four vertices from both meshes. As seen in Figure 9, the nearest vertex for A0 in the same mesh is B0. A1 and B1 are the nearest vertices for A0 and B0 from the neighboring mesh (See algorithm 2).

Algorithm 2 Algorithm for Stitching

```
Input: [$Meshes$], [$Edge\_line\_vertices$]
Output: [$faces$]
 1: Initialization:
 2: $n = length(Meshes)$,
 3: $g = Stitching\_separation$
 4: for $i = 0$ to $n - 1$ do
 5:     for $j = 0$ to $length(Edge\_line\_vertices[i]) - 2$ do
 6:         $v_1$ = Get nearest vertex of $j$ from $i + 1$ mesh
 7:         $v_2$ = Get nearest vertex of $j + 1$ from $i + 1$ mesh
 8:         $d_1 = D(j, v_1)$      ▷ $D(v_1, v_2)$ - Calculate distance
 9:         $d_2 = D(j + 1, v_2)$
10:         if $d_1 < Gap\ and\ d_2) \leq g$ then
11:             $faces \leftarrow [j, v_1, v_2]$
12:             $faces \leftarrow [j, v_2, j + 1]$
13:         end if
14:     end for
15: end forreturn $faces$
```

The identified vertex indices are used to create new faces for the mesh, and subsequently, all the meshes are concatenated and stitched by adding the new faces to the resultant

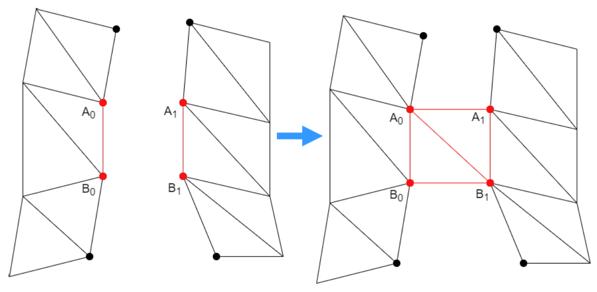


FIGURE 9. Identifying nearest vertices using KD-tree and creating the faces.

mesh. The final mesh is smoothened out by applying the Laplacian mesh smoothing algorithm.

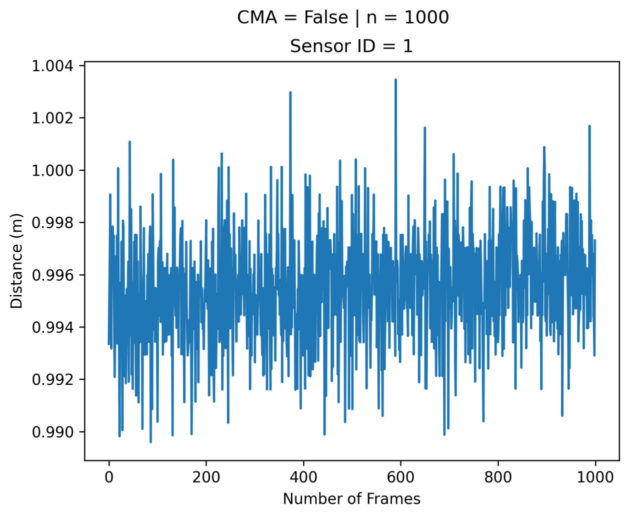
IV. RESULTS AND DISCUSSION

Parallel data acquisition with a multiple sensor system using a single USB port may slow down the actual frame rate captured from a single sensor at a time. In order to obtain an adequate number of frames/images with a higher frame rate from multiple sensor arrays using a single USB protocol, it is recommended to retain the maximum sensor count according to the provided technical specification. We used parallel data acquisition to capture data for 3D reconstruction because only four Intel RealSense D435 stereo depth sensors and a USB 3.1 connector can deliver the required frame rate (approximately 75-79 fps from any sensor for both color and depth frames). In this setup, we used a USB hub with an external power source that connects all four sensors since all the sensors draw power simultaneously. The Intel RealSense D435 sensor showed no adverse performance issues when connected in parallel with a single USB hub since this configuration is supported by the sensor's SDK. We observed that proper ambient lighting conditions must be maintained as poor light exposure can negatively affect the final 3D reconstruction, as mentioned in Intel's white paper [35].

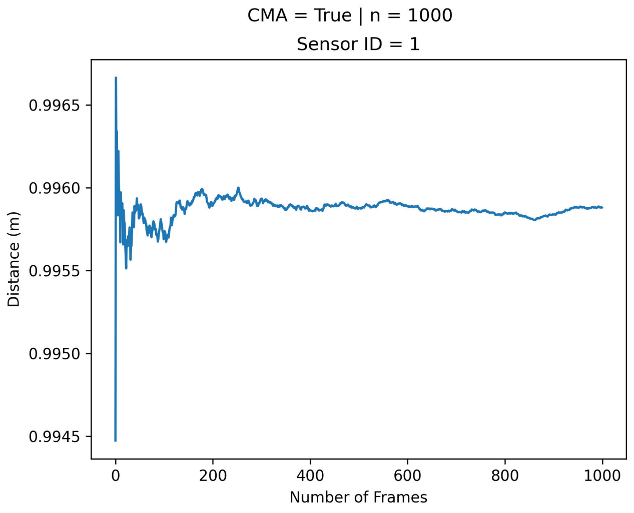
A. THE OPTIMAL NUMBER OF REQUIRED FRAMES

Generally, the sensors do not provide fixed values for the depth due to various factors affecting the sensor readings. With the CMA filter, we were able to minimize this anomaly by arriving at an optimum number of required frames (see Figure 10).

As seen in Figures 10 and 11, the sensor's resultant distance converges at different values as the measuring distance varies. Figure 10(b) shows the convergence at a distance of $1m$, while Figure 11(a) and Figure 11(b) display the convergences for $1.25m$ and $1.5m$ respectively. For all depths, the depth results converge approximately after the 200^{th} iteration. At this point, we limited the maximum depth to 1.5 m because our data was collected within 1 m to 1.5 m depths. We assumed this is common for all four sensors and choose the 200^{th} depth frame as the minimum number of depth frames required for the Intel RealSense D435 sensor to perform optimally.



(a)



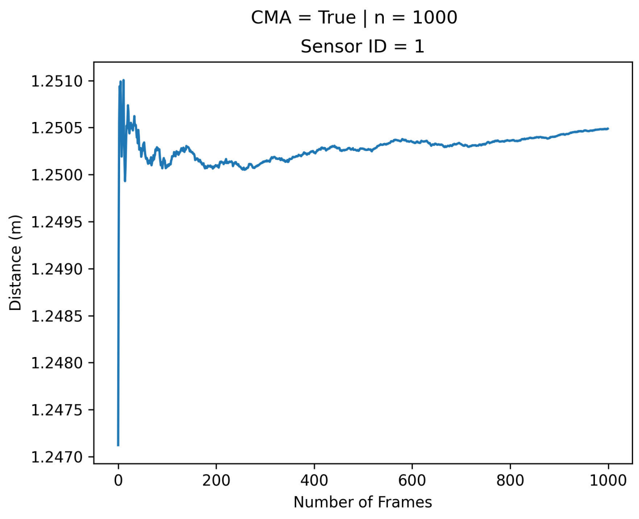
(b)

FIGURE 10. **1 *m* distances vs number of frames (a) without CMA filter (b) with CMA filter.**

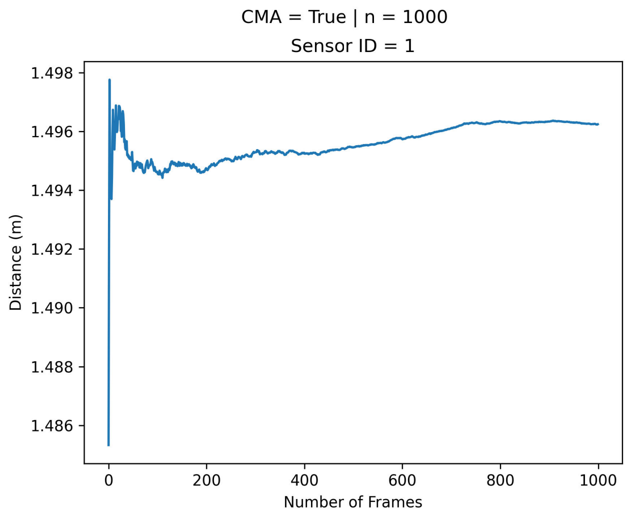
It is evident from Figure 12 that acquiring a higher number of frames (or long exposure) can affect depth accuracy. Using the minimum required number of frames is helpful in the reconstruction process since it can minimize the variation in accuracy that potentially occurs due to the effect of sensor heat or any other external factors (i.e. room temperature fluctuations, variations in humidity). The depth data acquisition is commenced by setting the maximum depth distance from each sensor to the object of interest to be 1.5 m as there is a higher error rate at longer distances.

B. DRIFT-EFFECT REMOVAL OUTPUT

In the graph plotting *SC* vs inverse-loss (see Figure 13), the board refers to the angled board plane we used for the ground truth and the base is the white background drift that was observed between the wall and the board in Figure 7. These

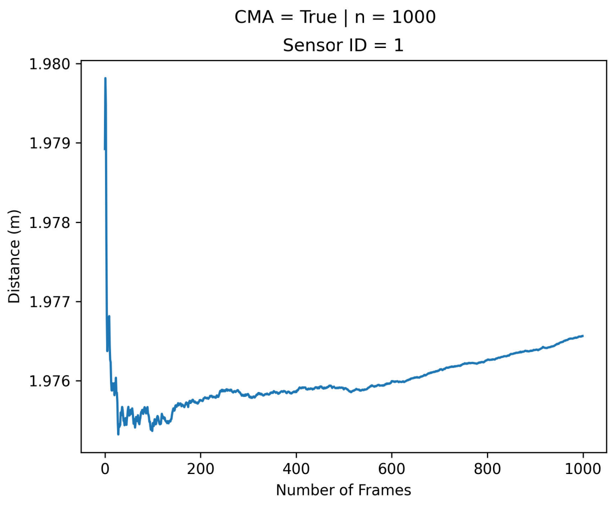


(a)

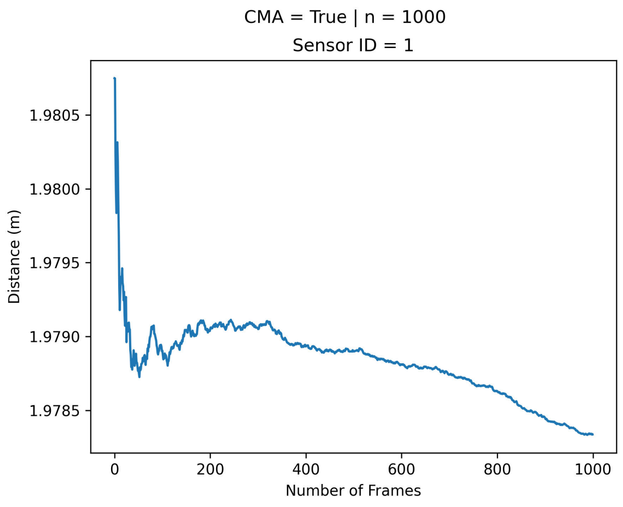


(b)

FIGURE 11. Distance vs number of frames with the CMA filter. (a) 1.25 *m* (b) 1.5 *m*.



(a)



(b)

FIGURE 12. Distances vs long exposure. 2 *m* vs 1000 frames (a) perspective I (b) perspective II.

inverse-loss values for the board plane were calculated using the ratios between the number of vertices that needed to be retained and the number of vertices in the ground truth. The white base inverse-loss values were calculated by the ratios between the number of vertices that needed to be removed and the number of vertices in the ground truth.

Figure 14 shows how the SC works with the algorithm. It is not recommended to use the algorithm with lower value which results in a loss of detail or a higher value which results in incorrect details to be retained in the final output.

According to Figure 13, SC $= 0.016m^{1/2}$ was identified to be the most compatible for use with the Intel RealSense D435 sensor. This value was chosen as it provides the maximum inverse-loss for the most interesting vertices (Board vertices) while minimizing the information loss from valid vertices, since it is recommended to retain valid/invalid vertices rather than losing valid vertices. This value is used in (9) to filter out unwanted faces at the edge and to generate a more accurate 3D model from the scanned object, as seen in Figure 15. We also found that this drift-effect occurs due to light diffraction at the edges of the object surfaces with respect to the sensor viewing point. Figure 16 depicts a similar phenomenon that was observed in the Kinect V2 sensor, which occurs in most generic depth sensors (Time of Flight, Structured light) as they utilize a similar technology (IR emitters) to generate the depth data.

We compared our algorithm's output with the "CMU Panoptic Dataset" [36], [37] which is a generic accepted dataset for the Kinect v2. This public dataset contains complex raw 3D points to validate our algorithm. This data was converted to a mesh and the implemented algorithm was applied to the mesh with the given calibration (See Figure 17).

As seen in Figure 17(b), our algorithm was able to fairly smoothen the edges of the input mesh in Figure 17(a).

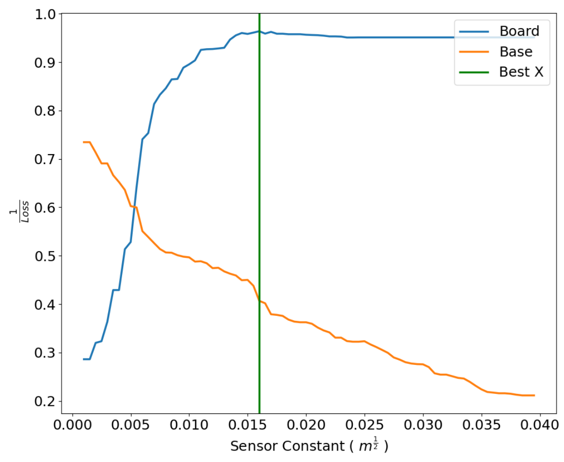


FIGURE 13. Determine the Sensor Constant (SC) - Best X line shows the maximum of inverse-loss point of the "board"

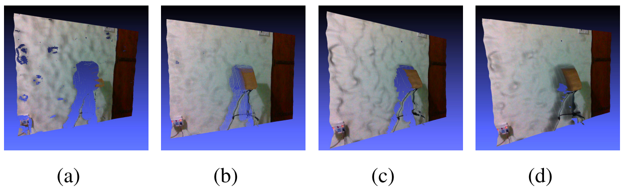


FIGURE 14. Applying SC for Figure 7 and generated outputs when (a) SC = 0.001 $m^{1/2}$ (b) SC = 0.007 $m^{1/2}$ (c) SC = 0.012 $m^{1/2}$ (d) SC = 0.027 $m^{1/2}$.

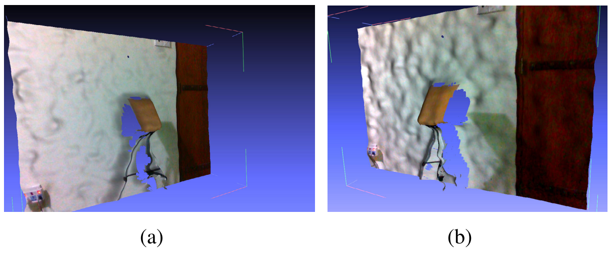


FIGURE 15. Maximum drift-effect removal observed when SC = 0.016 $m^{1/2}$ from Figure 7 sensor output (a) left View (b) right view.

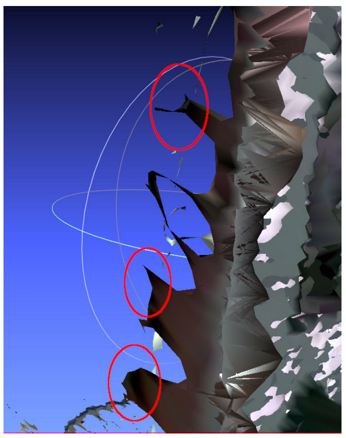


FIGURE 16. Kinect V2 sensor demonstrating the drift-effect at the edge.

C. FINAL STITCHED 3D MODEL

As shown in Figure 18, the center separation was stitched using our stitching algorithm. Finally, we used the Laplace mesh process to apply the final smoothing to the generated model. This process can be slow if the model is complex. However, since this is a post-processing technique, it is not necessary for the sensor to remain turned on. The vertices are not always in order, and it is essential to keep track of each vertex and face indices when adding or removing features from the mesh. Figure 19 shows the final result of our scanned clothing article, which can now be imported into an augmented or virtual reality application.

Figure 20 shows the usage of a 3D reconstructed clothing model in an implemented virtual dressing room.

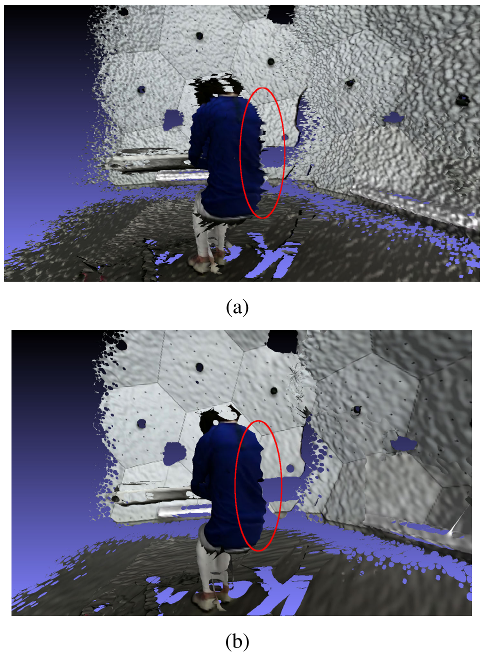


FIGURE 17. CMU Panoptic Data - Piano-4 (a) a view of original data mesh reconstructed (b) a view of applying the algorithm to mesh in (a).

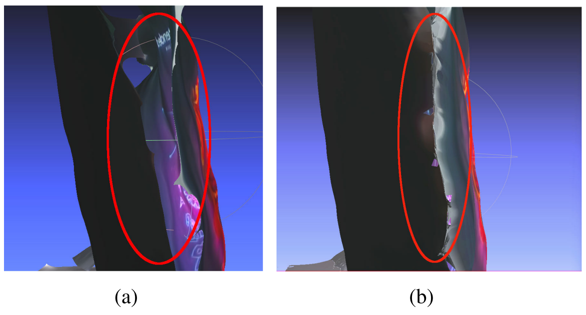


FIGURE 18. (a) before stitching meshes (b) after applying the stitching algorithm.

D. IMPLEMENTATION COMPARISON

In Table 1, we have compared and summarized features of the proposed technique with previous literature.

Our proposed technique is able to capture the entire object in less than three seconds as we have used a multiple parallel data acquisition technique instead of a single stream

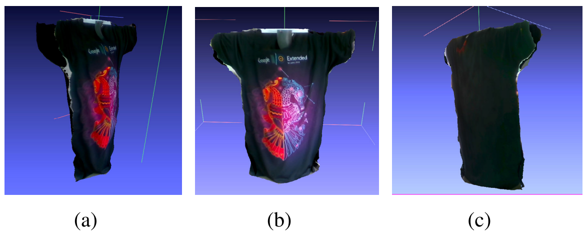


FIGURE 19. Generated final mesh (a) Front side view (b) Front view (c) back side view of the clothing article.

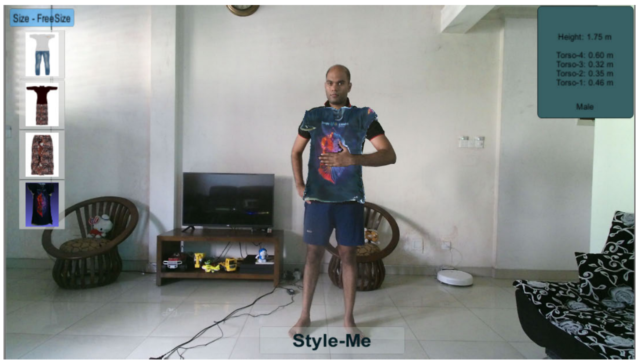


FIGURE 20. Virtual dressing room output with the scanned clothing article.

TABLE 1. Characteristics comparison of the implemented method.

Characteristic	Results in literature	Proposed work
Data acquisition speed - The speed of capturing required frames of the entire object	Takes more than 5 seconds (i.e. - Sensor rotation technique [8], [38], [39])	Less than 3 seconds
Implementation complexity - Hardware mechanical requirements and setup complexity of the systems	Sensors and additional hardware are required. Rotation mechanism (Motors/Rotation Tables) [8]–[11], Mirrors [17]–[19]	Minimal hardware - four static sensors
Size limitations - The dimensions and the weights limitations of the scanning object using the setup	Limited to rotation table limitations [8], such as weight, dimensions of the table	No weight limit and any object can be scanned up to 5 meters length and width
Mesh alignment process - Aligning the multiple meshes to create a single concatenate mesh	ICP-based alignments [24]	Image-based calibration
Feature similarity requirement	In methods such as SFM [14]–[16], [40], feature similarity required	Feature similarity is not required to concatenate meshes
Sensor scalability and changeability	Algorithm tests for predefined sensors (i.e. - Kinect Sensor [11], [22])	Based on color and depth frames. There is no limit to increasing the number of color and depth streams

with rotation tables or rotation sensors. We used only four fixed Intel RealSense D435 sensors in the experimental setup without any other tools or mechanics. Our system can be used to scan any object without a weight limitation as it does not require a specific table or a stand to support the object. However, as this specific implementation is connected with a USB C-type cable that is 5 meters long, the maximum distance at which an object can be placed is limited by the length of the USB cable. Further, we have used an image-based mesh alignment process which is promising compared to ICP because ICP-based mesh alignments do not perform satisfactorily if the mesh exhibits a considerable number of outliers. The proposed technique does not require any similarity in the scanning object as it uses RGB-D images and sensor poses to generate the mesh. Moreover, the system is scalable as it uses color and depth streams while the algorithm can be reused with any type of sensor module as long as it provides color and depth streams.

In addition to the improvements mentioned above, the proposed method consists of a drift-effect removal technique along with a stitching algorithm to create a closed-stitched 3D model.

V. CONCLUSION

In this research, we implemented an improved 3D reconstruction system that can 3D reconstruct any physical object using an inward multi-view depth sensor array. With this implementation, we proposed techniques to clip the unwanted depth information, detect the sensor pose, perform edge quality refinement, and generate closed, stitched 3D models of physical objects that can be used in any reality application. We used an array of four Intel RealSense D435 sensors in parallel, which was able to satisfy the optimal frame count by collecting more than 200 frames per sensor within three seconds. As a result, the proposed method can scan a complete 3D model within three seconds, which is a considerable reduction in scanning time when compared to alternative methods. Thereafter, it can generate the complete 3D model within one minute depending on the availability of compute resources. However, this takes place during the post-processing stage and so does not require the object to be visible. A well-calibrated system generates an elegant 3D mesh. However, the Intel RealSense D435 does not provide satisfactory depth data due to distortion caused by noise and factors such as the drift-effect. Furthermore, the D435 sensor encountered significant issues when distinguishing edges of nearby planes with different depths. We were able to successfully implement a solution for drift-effect removal by deriving a threshold (Th) value. This value was obtained using the Sensor Constant (SC) which was calculated specifically for the Intel RealSense D435 sensor. An accurate and detailed 3D model can be generated from the proposed technique if the system makes use of a more accurate depth sensor module. Since our proposed system only depends on color and depth frames as the input, the sensor can be swapped out with any number of alternative sensor modules while still using the same implementation, provided that the sensor modules in question meet the minimum requirements for the system. However, the drift-effect removal and mesh merging stitching algorithm is not directly applicable for real-time applications as it is computationally expensive. Our future work would focus on developing a more efficient real-time 3D reconstruction implementation.

ACKNOWLEDGMENT

The authors would like to thank the Department of Physics and the Apple Research and Development Centre, Department of Computer Science, Faculty of Applied Sciences, University of Sri Jayewardenepura, Sri Lanka, for providing facilities for this study.

VI. DATA AVAILABILITY

https://dx.doi.org/10.21227/dtcs-1r18

REFERENCES

[1] P. O. Candace. (Jul. 2021). *An Overview of Cloth 3D Fashion Design Software Irender, Irender Cloud Rendering Service*. Accessed: Jan. 15, 2022. [Online]. Available: https://irendering.net/an-overview-of-clo-3d-fashion-design-software/

[2] M. Merras, A. Saaidi, N. El Akkad, and K. Satori, "Multi-view 3D reconstruction and modeling of the unknown 3D scenes using genetic algorithms," *Soft Comput.*, vol. 22, no. 19, pp. 6271–6289, 2018.

[3] M. R. Minar, T. T. Tuan, H. Ahn, P. Rosin, and Y.-K. Lai, "3D reconstruction of clothes using a human body model and its application to image-based virtual try-on," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPR)*, Jun. 2020, pp. 1–4.

[4] A. Lobay and D. A. Forsyth, "Shape from texture without boundaries," *Int. J. Comput. Vis.*, vol. 67, no. 1, pp. 71–91, 2006.

[5] A. M. E. Loh and R. Hartley, "Shape from non-homogeneous, non-stationary, anisotropic, perspective texture," in *Proc. BMVC*, 2005, pp. 69–78.

[6] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the art on 3D reconstruction with RGB-D cameras," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 625–652, 2018.

[7] J. Li, W. Gao, Y. Wu, Y. Liu, and Y. Shen, "High-quality indoor scene 3D reconstruction with RGB-D cameras: A brief review," *Comput. Vis. Media*, vol. 8, pp. 369–393, Mar. 2022.

[8] A. Haleem, P. Gupta, S. Bahl, M. Javaid, and L. Kumar, "3D scanning of a carburetor body using COMET 3D scanner supported by COLIN 3D software: Issues and solutions," *Mater. Today, Proc.*, vol. 39, pp. 331–337, Jan. 2021.

[9] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling Kinect sensor noise for improved 3D reconstruction and tracking," in *Proc. 2nd Int. Conf. 3D Imag., Modeling, Process., Vis. Transmiss.*, Oct. 2012, pp. 524–530.

[10] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D model construction for turn-table sequences," in *Proc. Eur. Workshop 3D Struct. Multiple Images Large-Scale Environ.* Berlin, Germany: Springer, 1998, pp. 155–170.

[11] A. L. Popescu and C. Raluca, "Real-time 3D reconstruction using a Kinect sensor," *Comput. Sci. Inf. Technol.*, vol. 2, no. 2, pp. 95–99, 2014.

[12] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, nos. 1–3, pp. 7–42, Apr. 2002.

[13] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proc. IEEE CVPR*, vol. 1, Jun. 2006, pp. 519–528.

[14] Q. Mi and T. Gao, "3D reconstruction based on the depth image: A review," in *Proc. Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.* Cham, Switzerland: Springer, 2022, pp. 172–183.

[15] S. El Hazzat, M. Merras, N. El Akkad, A. Saaidi, and K. Satori, "Enhancement of sparse 3D reconstruction using a modified match propagation based on particle swarm optimization," *Multimedia Tools Appl.*, vol. 78, no. 11, pp. 14251–14276, 2019.

[16] Y. M. Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Miscusik, and S. Thrun, "Multi-view image and ToF sensor fusion for dense 3D reconstruction," in *Proc. IEEE 12th Int. Conf. Comput. Vis. Workshops (ICCV)*, Oct. 2009, pp. 1542–1549.

[17] T.-N. Nguyen, H.-H. Huynh, and J. Meunier, "3D reconstruction with time-of-flight depth camera and multiple mirrors," *IEEE Access*, vol. 6, pp. 38106–38114, 2018.

[18] B. Hu, C. Brown, and R. Nelson, "Multiple-view 3D reconstruction using a mirror," Univ. Rochester, Rochester, NY, USA, Tech. Rep. 863, 2005. Accessed: Feb. 2, 2022. [Online]. Available: https://urresearch.rochester.edu/fileDownloadForInstitutionalItem.action?itemFileId=1791&itemId=1440

[19] P. Xie, Y. Ye, L. Shu, and Z. Song, "A full-body 3D reconstruction using planar mirrors," in *Proc. IEEE Int. Conf. Real-Time Comput. Robot. (RCAR)*, Jul. 2021, pp. 543–547.

[20] J. Tan, W. Lin, A. X. Chang, and M. Savva, "Mirror3D: Depth refinement for mirror surfaces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15990–15999.

[21] E. Auvinet, J. Meunier, and F. Multon, "Multiple depth cameras calibration and body volume reconstruction for gait analysis," in *Proc. 11th Int. Conf. Inf. Sci., Signal Process. Appl. (ISSPA)*, Jul. 2012, pp. 478–483.

[22] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, "Scanning 3D full human bodies using Kinects," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 4, pp. 643–650, Apr. 2012.

[23] T. N. Linh and H. Hiroshi, "Global iterative closet point using nested annealing for initialization," *Proc. Comput. Sci.*, vol. 60, pp. 381–390, Jan. 2015.

[24] M. Kowalski, J. Naruniec, and M. Daniluk, "Livescan3D: A fast and inexpensive 3D data acquisition system for multiple Kinect v2 sensors," in *Proc. Int. Conf. 3D Vis.*, Oct. 2015, pp. 318–325.

[25] S. Brandao, J. P. Costeira, and M. Veloso, "Effortless scanning of 3D object models by boundary aligning and stitching," in *Proc. Int. Conf. Comput. Vis. Theory Appl. (VISAPP)*, vol. 1, Jan. 2014, pp. 667–674.

[26] O. Sorkine, "Laplacian mesh processing," in *Eurographics (State of the Art Reports)*. Princeton, NJ, USA: Citeseer, 2005, pp. 53–70.

[27] A. G. Jepsen, P. Winer, and A. Takagi. (2021). *Multi-Camera Configurations—D400 Series Stereo Cameras*. Accessed: Feb. 2, 2022. [Online]. Available: https://dev.intelrealsense.com/docs/multiple-depth-cameras-configuration

[28] S. B. Adikari, N. C. Ganegoda, R. G. N. Meegama, and I. L. Wanniarachchi, "Applicability of a single depth sensor in real-time 3D clothes simulation: Augmented reality virtual dressing room using Kinect sensor," in *Advances in Human-Computer Interaction*. London, U.K.: Hindawi, 2020.

[29] R. Usamentiaga and D. F. Garcia, "Multi-camera calibration for accurate geometric measurements in industrial environments," *Measurement*, vol. 134, pp. 345–358, Feb. 2019.

[30] A. G. Jepsen, J. Sweetser, and T. Khuong. (2021). *Intel Realsense Self-Calibration for D400 Series Depth Cameras*. Accessed: Feb. 1, 2022. [Online]. Available: https://dev.intelrealsense.com/docs/self-calibration-for-depth-cameras

[31] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," 2018, *arXiv:1801.09847*.

[32] P. H. Schönemann, "A generalized solution of the orthogonal Procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[33] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: An open-source mesh processing tool," in *Proc. Eurograph. Italian Chapter Conf.*, Salerno, Italy, 2008, pp. 129–136.

[34] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[35] J. Sweetser and A. G. Jepsen. (2022). *Optical Filters for Intel Realsense Depth Cameras D400*. Accessed: Feb. 10, 2022. [Online]. Available: https://dev.intelrealsense.com/docs/optical-filters-for-intel-realsense-depth-cameras-d400

[36] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. Godisart, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic studio: A massively multiview system for social interaction capture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 190–204, Jan. 2019.

[37] H. Joo, T. Simon, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *Proc. CVPR*, 2017, pp. 1145–1153.

[38] Z. Li, T. Yu, C. Pan, Z. Zheng, and Y. Liu, "Robust 3D self-portraits in seconds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1344–1353.

[39] F. Yu and D. Gallup, "3D reconstruction from accidental motion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3986–3993.

[40] Y. Xue, S. Zhang, M. Zhou, and H. Zhu, "Novel SfM-DLT method for metro tunnel 3D reconstruction and visualization," *Underground Space*, vol. 6, no. 2, 2021, pp. 134–141.

SASADARA B. ADIKARI received the B.Sc. degree in physics from the University of Sri Jayewardenepura, Sri Lanka, in 2014, where he is currently pursuing the Ph.D. degree in concepts and applications of augmented/virtual reality. He is also working as a Lead Software Developer. He has industrial experience of more than six years of experience including various industry version 4.0, such as artificial intelligence, machine learning, the IoT, and reality technologies. His research interest includes application of reality technologies with machine learning.

NALEEN GANEGODA received the B.Sc. degree (Hons.) in mathematics, in 2004, and the Ph.D. degree from the University of Sri Jayewardenepura, in 2011. His postgraduate research is about mathematical modeling of disease transmission. He has been supervise several M.Phil. degrees in applied mathematics. He has collaborates on applied mathematics research with the Mathematics Institute, University of Koblenz, Germany, initially funded by the National Science Foundation of Si Lanka. He is currently a Senior Lecturer with the Department of Mathematics, University of Sri Jayewardenepura. He has been able to publish more than ten indexed journals publications in the last year or so. His teaching interests include numerical methods, optimization, and computational mathematics. He received the Gold Medal for the best performance for that year, in 2004.

RAVINDA MEEGAMA received the B.Sc. degree (Hons.) in computer science from the University of Colombo and the M.Sc. degree in computer science from the Asian Institute of Technology, Thailand. He undertook research in medical imaging leading to a Ph.D. degree from NTU, Singapore, in 2004. He is currently a Senior Professor of computer science with the Department of Computer Science, Faculty of Applied Sciences, University of Sri Jayewardenepura. His current research interests include machine learning approaches for image processing, computer graphics, and mobile computing.

INDIKA L. WANNIARACHCHI received the B.Sc. degree (Hons.) from the Department of Physics, University of Sri Jayewardenepura, in 2005. He received the Ph.D. thesis in computational physics from Wayne State University, MI, USA, in 2013. He joined as a Probationary Lecturer with the Department of Science and Technology, Uva Wellassa University, Sri Lanka, in 2006, and contributed to development of the university in many ways. Since 2015, he has been with the Department of Physics, University of Sri Jayewardenepura, where he is currently a Senior Lecturer. After completing his doctorate, he decided to expand his research interests more on applied physics areas include computer vision and image processing, computational physics, electronics and embedded systems, and electronics structure.