DOI: https://doi.org/10.1016/j.cag.2017.08.003

Robust Enhancement of Depth Images from Depth Sensors

ABM Tariqul Islam[a], Christian Scheel[a], Renato Pajarola[b], Oliver Staadt[a,*],

[a] *Visual Computing Lab, University of Rostock, Germany.*
[b] *Visualization and MultiMedia Lab, University of Zürich, Switzerland.*

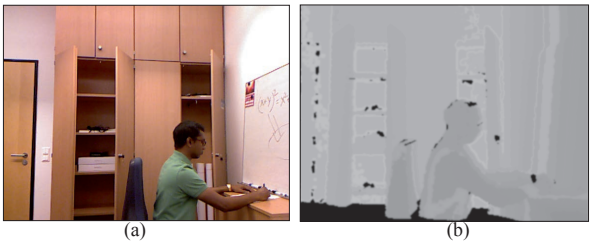
Abstract

In recent years, depth cameras (such as Microsoft Kinect and ToF cameras) have gained much popularity in computer graphics, visual computing and virtual reality communities due to their low price and easy availability. While depth cameras (e.g. Microsoft Kinect) provide RGB images along with real-time depth information at high frame rate, the depth images often suffer from several artifacts due to inaccurate depth measurement. These artifacts highly degrade the visual quality of the depth frames. Most of these artifacts originate from two main sources – the missing/invalid depth values and fluctuating valid depth values on the generated contents. In this paper, we propose a new depth image enhancement method, for the contents of depth cameras, which addresses these two main sources of artifacts. We introduce a robust 1D Least Median of Squares (1D LMedS) approach to estimate the depth values of those pixels which have missing/invalid depth values. We use a sequence of frames to look for invalid depth values (considered as outliers), and finally, replace those values with stable and more plausible depth values. By doing so, our approach improves the unstable nature of valid depth values in captured scenes that is perceived as flickering. We use self-recorded and reference datasets along with reference methods to evaluate the performance of our proposed 1D LMedS. Experimental results show improvements both for static and moving parts of a scene.

Keywords: Robust filtering, RGB-D sensor, Depth image, 3D imaging, Depth image enhancement, Virtual reality.

1. Introduction

Nowadays, researchers often use depth cameras (e.g. Microsoft Kinect, ToF cameras) to generate 3D contents. Due to their low cost and easy availability, several areas in computer graphics (e.g. visual computing, human-computer interaction, virtual reality, remote collaboration) adopted these cameras as the primary acquisition device. The sensors of these cameras capture a scene with acceptable resolution and speed, but the captured depth data exhibit considerable amount of artifacts (e.g., flickering artifact and random black spots as shown in Figure 1). Most of these artifacts originate from two primary sources–invalid depth values and fluctuating valid depth values on the generated contents. Due to these two prime sources of artifacts, we often see highly degraded 3D contents whose quality is often too low for many applications such as telepresence, object tracking in a scene, forensic analysis of crime scenes etc. In case of telepresence, a low quality 3D scene capturing limits the sensation of a natural presence of remote users on the local site [1]. Similarly, for tracking applications, the noise on an object's surface greatly deteriorates the tracking performance. Likewise, in forensic analysis of a crime scene, where every detail is crucial, a degraded scene might hamper the recovery of correct information [2].



(a) (b)

Figure 1: **RGB-D frame from a Kinect sensor** – (a) color image, (b) depth image. The artifacts (e.g. the black spots) in (b) occurs due to the invalid depth values reported by Kinect RGB-D sensor.

Accordingly, researchers conducted a number of

*Corresponding author
Email addresses: tariqul.islam@uni-rostock.de (ABM Tariqul Islam), CScheel@gmx.de (Christian Scheel), pajarola@ifi.uzh.ch (Renato Pajarola), oliver.staadt@uni-rostock.de (Oliver Staadt)

works to enhance the depth frames from the depth cameras by addressing the aforementioned sources of artifacts. In those works, they propose to use different filters in combination with motion compensation and color image data (see e.g., [1, 3–17]). In this paper, we present a robust method of enhancing depth frames which is a continuation and extension of our previous works [18, 19]. Here we have extended our previous works by including a more detailed technical description of our proposed method, several additional datasets and reference works to evaluate our work and, quantitive and qualitative comparison analysis on real-world and artificial depth data. Moreover, we clarify the concept of general Least Median of Squares and the concept of how our method improves each pixels of RGB-D frames. Furthermore, we demonstrate how our method can be implemented to improve the 3D representation quality of a telepresence communication.

Our approach. Contrary to prior works, we propose the use of Least Median of Squares (LMedS)[20] and extend it to enhance both static and dynamic scenes. By adapting LMedS to be used as a 1D (one dimensional) LMedS, we go through a sequence of frames, detect the invalid and unstable depth values for each pixel, identify the outliers among them and finally, replace them with stable valid depth values. Our proposed 1D LMedS is computationally less expensive than higher dimensional LMedS and the application of 1D LMedS on the depth frames is efficient. Experimental results show improvements in filling the invalid depth values and stabilizing the valid values for static and dynamic parts of a scene.

Contribution. Our contribution towards enhancing the depth frames of depth cameras is as follows:

- A computationally inexpensive and efficient way of enhancing the depth images for real-time applications.
- Introducing 1D LMedS which fits right into the problem domain of depth image enhancement.
- Formulate the estimator, standard deviation and weight function from original LMedS to fit into 1D LMedS.
- Develop a fast data traversal procedure for 1D LMedS.
- Correctly identify the invalid depth values and fluctuating valid depth values, and stabilize flickering artifacts for both the static and dynamic scenes.
- Evaluate our method with local and reference datasets and with reference methods.

2. Related work

Over the past few years, we have seen a number of works on enhancing the depth images captured by depth cameras. Most of these works use filters, like median filters [1, 4], Kalman filters [5], guided image filters [6, 7, 21, 22], bilateral filters [8, 10–15], and sometimes a combination of median and bilateral filters [12, 23]. Beside the type of applied filter, the approaches can basically be categorized by the application of motion compensation, the covered depth sensor type, the inclusion of color image data, and realtime support, for example, by hardware acceleration. Most of these approaches utilize spatial or range filter kernels, while only very few of them consider the temporal noise component.

For instance, Maimone and Fuchs use a GPU-adapted median filter for enhancing the depth images [1]. Their method successfully fills the black spots on the surface of the objects, but it occasionally reports wrong interpolated values. Moreover, it also exhibits temporally inconsistent depth frames due to not considering temporal information. Matyunin et al. use motion information along with a median filter which generates good results for some image areas, but suffers mainly from noisy object boundaries [4]. Besides, it suffers from large computational overhead. Lai et al. propose to use a recursive median filter on the depth images from Kinect sensors. However, they mainly focus on improving object recognition accuracy rather than depth map accuracy [24].

Amamra and Aouf use a GPU-adaptive Kalman filter to enhance the depth images in real-time [5]. However, this method suffers when object's surface is sharply reflected. Wasza et al. propose another GPU-based approach where a combination of normalized convolution and guided filter is used to improve the depth map in real-time, but it suffers from blurry object borders [25]. Some other works (e.g. [10, 11]) use bilateral filters for depth image enhancement. Chen et al. use a joint bilateral filter where color image is used as guidance image [11]. Their method enhances the depth images except for the image areas without color information. Yang et al. also enhance depth images by using bilateral filters [12]; but it is not suitable for dynamic scenes because it does not consider temporal information.

Hui and Ngan use a combination of 2D median and bilateral filters where the bilateral filter removes the artifacts introduced by the median filter [23]. Maimone et al. also use a combination of median and bilateral filters to enhance the depth images used in telepresence [26]. Although, their method enhances the surface of objects, it exhibits temporally inconsistent depth frames.

Telea uses an image inpainting method [27] which

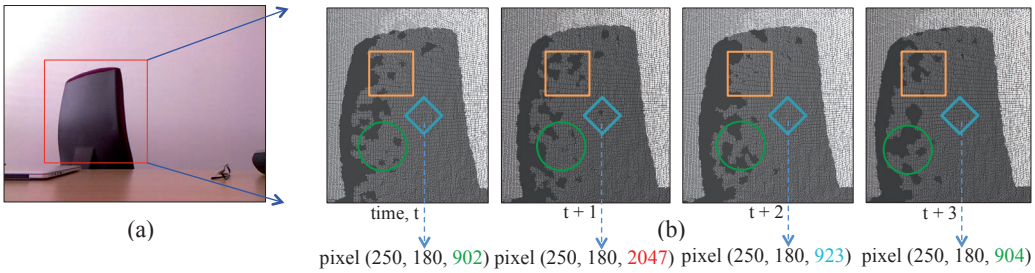


Figure 2: **RGB-D frames from Kinect with static objects** – (a) color image, (b) four consecutive depth frames of square marked area of (a). The depth values of one pixel inside the diamond marked area in (b) are depicted below the frames. We observe three types of depth value distortions: 2047 is an invalid value (reported occasionally), 923 is an outlier and the others are regular depth values fluctuating because of Gaussian noise.

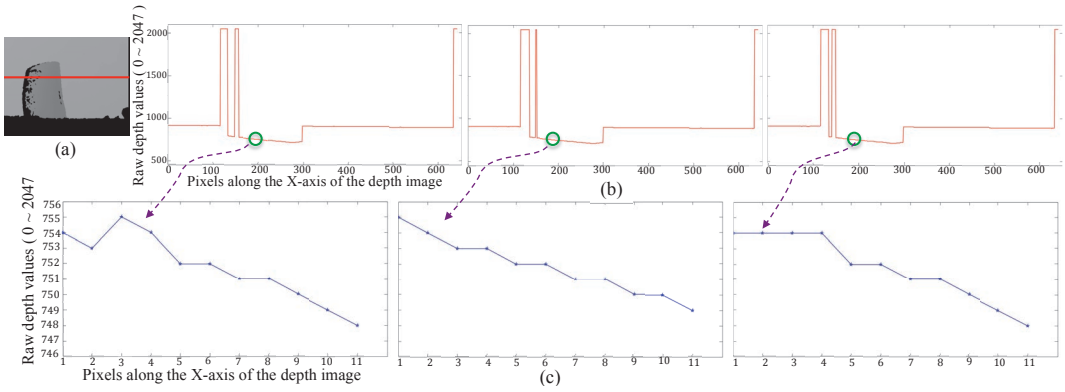


Figure 3: **Invalid and unstable valid depth values from Kinect sensor on a sequence of frames** – (a) depth image with the red marked scan line, (b) depth values of the scan line for 3 consecutive frames of (a), (c) depth values of the green circled marked area on the plots of (b).

considers both the depth and color information to fill-in the missing spots [14]. However, it considers only the spatial relationship among the pixels, resulting in wrong interpolated values and noisy object boundaries. Camplani et al. use an iterative joint-bilateral filter which combines depth and color data by analyzing an edge-uncertainty map and pre-detected foreground regions [8, 28]. It performs well for static scenes, but is not suitable for dynamic scenes. Other works (e.g. [17, 29]) also focus on depth map processing, but these are based on stereo [29] or are specific to time-of-flight (ToF) sensors [17].

There are also other works which utilize color image data to enhance the quality of the corresponding depth images (e.g. [9, 30–32]). While most of these methods use the Markov Random Field model or the auto regression model to enhance the depth images, they differ mainly in the design of regularization terms in the objective function. The resulting depth images are quite good and contain almost no artifacts. Although, the energy minimization approaches, which these works use, are computationally expensive [30], recent computational development would enable these methods to be used for realtime applications. There are also some works (e.g. [33]) which use color-depth image pairs and interpolation based approach to refine and enhance the corresponding depth images. The authors in [33] propose a combination of de-noise and interpolation method which performs quite well for stationary objects of a scene; it nicely refines the areas with artifacts in depth images.

We have also seen works (e.g. [17, 23, 34, 35]) which use the flow of information in consecutive color images. In [17], Kim et al. use a combination of spatial and temporal depth enhancement process that considers motion flow between consecutive color images to infer information about object motion in the corresponding depth images. However, they basically ignore this data in the dynamic parts of the depth images as they use it only to detect stationary parts. Based on this, they apply a bilateral filter to improve the quality which works well for static parts. Hui et. al. enhance depth images captured from a moving RGB-D system [23]. They also estimate the optical flow of consecutive color images. How-

ever, instead of building a temporal filter on top of the obtained data, their method estimates additional depth cues from the flow which are then combined with the original depth images. Their method is intended for mobile setups and not applicable to stationary cameras as considered in our case. Another work [35] by Avetisyan et. al. also use optical flow between successive frames and they are able to get decent quality depth frames. Although their method works in real-time, it suffers when missing data or noise stays persistently in one region of the depth image sequence. Besides, their method occasionally yields invalid motion vectors which deteriorate the estimated history of depth values; as a result, ghosting artifacts are shown for fast moving objects.

Even though each of the mentioned works is designed to solve different issues related to the depth data from depth cameras and they perform well in their respective applications areas, most of them lack in considering the temporal inconsistency and often, they suffer from large computational overhead. In this paper, we address the temporal inconsistency issue of depth cameras.

3. Problem description

Depth cameras (such as Kinects and ToF cameras) exhibit artifacts in the form of missing data, invalid depth values and valid fluctuating depth values. Here we use Kinect data to illustrate the artifacts, because major part of the depth image enhancement works utilize Kinects as acquisition device. Figure 1 shows the artifacts found on Kinect's depth frames. These artifacts occur because the structured light sensor of a Kinect produces faulty depth measurements. A Kinect captures both the color and depth data of a scene. The depth of an object is measured by projecting light patterns from an infrared light source to the object space [36]. An infrared camera receives the reflected light pattern from the surface of the object and compares it against a reference pattern. An estimated disparity image of the objects inside a scene and corresponding depth calculations are obtained by the differences between the captured patterns and the reference pattern [8, 37]. The faulty depth measurements of a Kinect might originate due to multiple reflections, strong lighting conditions, scattering object surface etc. [8, 36, 37].

Moreover, the depth measurements are also affected by the instability over time, which results in flickering spots [8]. Flickering occurs even on a scene with static objects (see Figure 2); here, we can observe the unstable nature of valid depth values for certain pixels. Moreover, Figure 3 (b) shows that a Kinect reports both invalid values (the spikes on the plots) and valid values

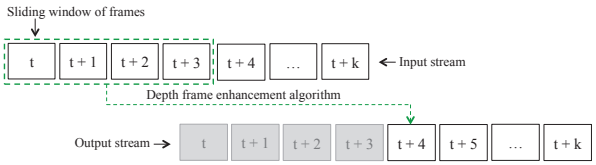


Figure 4: **Illustration of sliding window of frames in 1D LMedS** – number of frames inside the slide window is 4 and total number of frames is k.

for the same pixels in a sequence of frames, although the objects are static. Besides, Figure 3 (c) shows that the valid depth values, inside the circled marked areas of Figure 3 (b), also differ for the same pixels in the sequence of frames. We take this unstable nature into consideration and apply 1D LMedS to replace the unstable and invalid values with valid stable depth values.

4. Least Median of Squares

The Least Median of Squares (LMedS) [20] is a statistical technique for robust regression of a p-dimensional sample set (x_i, y_i), which minimizes the median of the squared residuals ${r_i}^2$ over a set of estimates θ in $\mathbb{R}^p$. Here, r_i and θ are expressed as in Equation 1 and Equation 2:

$$r_i(n) = y_i - (x_{i1}\hat{\theta}_1 + \cdots + x_{i(p-1)}\hat{\theta}_{p-1} + x_{ip}\hat{\theta}_p) \quad (1)$$

$$\theta = (\theta_1, \theta_2, \cdots, \theta_p)^t \quad (2)$$

Basically, a vector n represents a hyperplane that, in a certain way, describes the linear dependency of y_i from the $p-1$ variables in x_i. LMedS is known to be robust to false matches and outliers while fitting equations to the observed dataset. It estimates the data points by solving the nonlinear minimization problem which is $r_i(n)$. The outliers from the dataset can be filtered out by using a robust standard deviation $\hat{\sigma}$ which uses the minimum median of $r_i(n)$. But, the LMedS estimator becomes computationally expensive when the dimension p of sample set (x_i, y_i) gets bigger. In our approach, we propose to use a one dimensional LMedS which is a good fit for the nature of our problem domain and its implementation in such problem domain is also efficient.

5. Proposed strategy

To enhance the depth frames by filling the invalid depth values d_{in_vld} and unstable valid depth values d_{u_vld} with stable valid depth values d_{s_vld}, we introduce a 1D LMedS approach which is robust to outliers. We apply

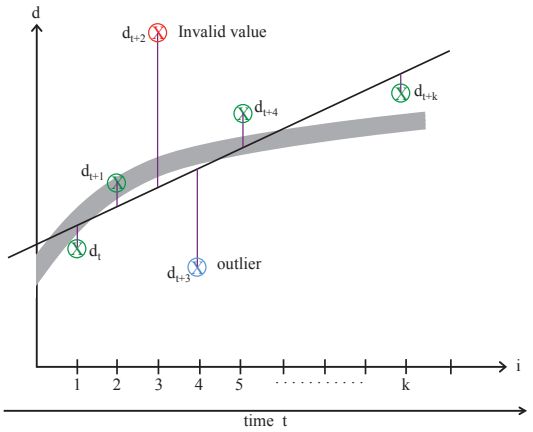


Figure 5: **Illustration of 1D LMedS on one depth pixel in a sliding window.** The thick gray line shows the real distance of a scene object over time. The checked circles represent depth values d_{t+i-1} obtained in k consecutive frames. The thin black line approximates the gray line by calculating the estimator m i.e. by minimizing the median of the absolute distances to the depth values, visualized by vertical purple lines. The red marked *invalid value* at $i = 3$ and the turquoise marked *outlier* at $i = 4$ are detectable by our 1D LMedS.

the 1D LMedS on a per pixel basis over a sequence of frames inside a temporal sliding window. Let us consider, for example, that we have k consecutive depth frames and we take n frames in a sliding window (see Figure 4). Due to Kinect's nature of depth value generation, we get different depth values d_i for any single pixel for the n previous consecutive frames inside the sliding window. We apply the 1D LMedS on this set of depth values which vary along the temporal domain.

The name 1D LMedS (one dimensional LMedS) comes from the concept that the depth values d_i of a certain pixel (x,y) changes only in one dimension that is along the temporal domain i.e. over the consecutive frames. Our goal is to locate the invalid and unstable depth values (which we consider as outliers) from the set of depth values of each pixel and replace them with a stable valid depth value. An illustration of 1D LMedS on k consecutive frames is depicted in Figure 5. Our depth enhancement strategy is based on three principle steps. For each pixel inside the sliding window (i.e. for the n consecutive frames):

- detect if there are invalid depth values d_{in_vld}.
- detect if the valid depth values d_{vld} are fluctuating from one frame to another.
- if the d_{vld} are fluctuating, find the outliers among the d_{vld}, remove them and process the inliers to get a final stable depth value d_{s_vld} for that pixel. Also replace the d_{in_vld} with the d_{s_vld}.

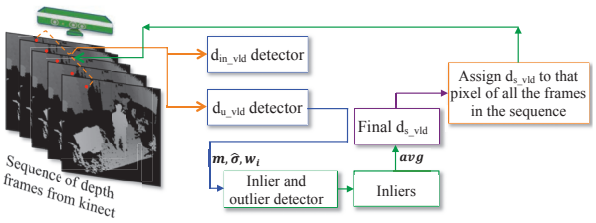


Figure 6: **Block diagram of the processing steps of 1D LMedS** – the red dots in the frame sequence represent one pixel of the frames.

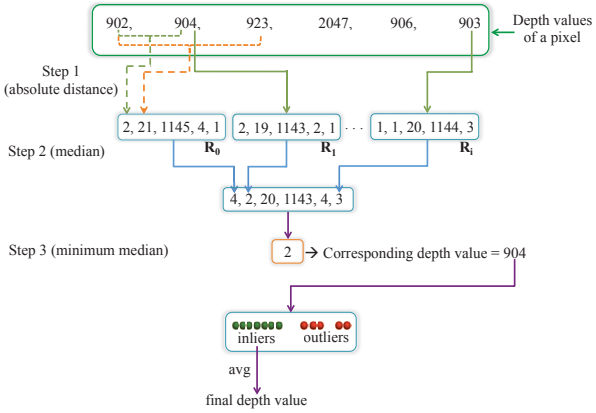


Figure 7: **Illustration of 1D LMedS for one pixel in the n consecutive frames** – the values on the top green box are the raw depth values obtained for one pixel over the frames inside the sliding window.

The processing steps of 1D LMedS are depicted in Figure 6. For detecting d_{in_vld}, we look for the pixels which have invalid depth values; the depth value of pixels with invalid depth can be found in [8, 37]. To detect the unstable depth values d_{u_vld} from the set of valid depth values d_{vld} for the n frames, we use the standard deviation $\hat{\sigma}$ of the depth values for the n frames and then compare with certain constraints (stated in Equation 4) to remove them as outliers. A detailed description of these processes is explained in the following paragraphs.

To remove the outliers and fill the invalid depth values for a certain pixel, we calculate the estimator m which is the minimum of the medians of absolute residuals R_i for each depth value d_i of the set of depth values for n consecutive frames. In 1D LMedS, we use the absolute difference for getting the residuals R_i among the depth values d_i; in [20], R_i^2 is used to avoid negative values.

The whole calculation process is illustrated in Figure 7. Here, in step 1, for a set of depth values d_i, we calculate absolute difference for each depth value to the rest of the depth values for that pixel. So, we obtain a set of residuals for each depth values of d_i. In Figure 7, the

first set of absolute distances i.e. $R_0 = \{2, 21, 1145, 4, 1\}$ are calculated for the first depth value 902, R_1 for the second depth value 904 and so on. In step 2, we calculate the medians from each set of residuals R_i and then, in step 3, we calculate the 1D LMedS estimator m which is the minimum of the medians. Equation 3 shows the formula to calculate m.

$$\begin{aligned} \underset{(i,j:1,\cdots\cdots,n)}{R_i(d_i)} &= \{|d_i - d_j|, \quad if \quad i \neq j\} \\ m &= \min_{(i:1,\cdots\cdots,n)} \text{median} \quad R_i(d_i) \end{aligned} \tag{3}$$

After that, we retrieve the corresponding depth value whose index matches the index of m. Here, in step 3, we can see that the value of m is 2; which originates from the second set of residuals R_1. As R_1 originates from the second depth value 904, we use the value of m and the retrieved depth value (904) for the rest of the process of 1D LMedS. We calculate the standard deviation $\hat{\sigma}$ and weight w_i, which decide if a certain depth value is an inlier or outlier, according to the formula stated in Equation 4. After we obtain inliers and outliers, we finally obtain the stable valid depth value by taking the average of the inliers.

$$\begin{aligned} \hat{\sigma} &= 1.4826\left[1 + \frac{5}{n-1}\right]m \\ w_i &= \begin{cases} 1, & if \quad r \leq 2.5\hat{\sigma} \\ 0, & otherwise. \end{cases} \end{aligned} \tag{4}$$

The constant 1.4826 of Equation 4 is a coefficient to achieve the optimal efficiency in the presence of Gaussian noise, $5/(n-1)$ is used to compensate the effect of using a small set of data and the constant 2.5 is used with $\hat{\sigma}$ because in a Gaussian situation there will be very few residuals larger than $2.5\hat{\sigma}$; for a detailed explanation of the coefficients, please look at pages 17, 29, 44, 131, 132 and 202 in [38]. However, this process needs quite some time to traverse for all the depth values of each pixel for these n frames. Because, in Figure 7, we can see that to obtain the absolute distances R_is, we need to check each depth value to the rest of the depth values to get the R_is which takes quite some time. Therefore, to reduce the calculation of R_is, we adapt a fast data traversal procedure to fit 1D LMedS.

Fast data traversal procedure for 1D LMedS. The fast data traversal procedure is stated in Algorithm 1. We apply this procedure for each of the depth values d_i of every pixel of the n frames. Algorithm 1 shows the pseudocode for fast traversal for one pixel over the n frames. Here, the fast traversal occurs due to the faster

Algorithm 1 Fast traversal for the depth values of one pixel

```
1: procedure
2:     Define   MAX 10000
3:     int   idx = 0,   n   ▷ n = no. of frames in the window
4:     Float  diff,  depthVal[n],  preFinalDepthVal
   ▷  depthVal[n] contains the unsorted depth values
   of a pixel
5:     depthVal_S[n]=sorted depth values of depthVal[n]
     ▷ values of depthVal_S[n] are in ascending order
6:     min ← Float  MAX
7:     for int i = 0 to (n-1)/2  do
8:         diff = depthVal_S[i + n/2] − depthVal_S[i]
9:         if diff < min then
10:            min = diff
11:            idx ← i
12:        end if
13:    end for
14:    preFinalDepthVal = (depthVal_S[idx + n/2] + depthVal_S[idx]) / 2
15: end procedure
```

computation of the minimum of medians m (stated as idx in Algorithm 1). Here, before beginning the calculation of obtaining m and corresponding depth value, we sort the depth values d_i (see line 5 of Algorithm 1) and split the traversal path to half of the window size. The validity and efficiency of using half of the sorted sample data (i.e. splitting the sorted depth values into half) rather than checking each data to the rest of the other data one by one (which we previously did in Figure 7) is stated in [20]. We adapt this time-efficient median calculation process into 1DLMedS to reduce the overall residual (i.e. difference between the depth values) calculation time for each of the sorted depth values inside the window. After we obtain the minimum difference (see line 10 of Algorithm 1), we find the index `idx` of the minimum difference (see line 11) and retrieve the depth value which has this minimum index `idx` (see line 14). Based on this depth value and index `idx`, we calculate the $\hat{\sigma}$, and find the inliers and outliers according to w_i of Equation 4. Then we take the average of the depth values of the inliers as the final depth value.

In this way, 1D LMedS enhances a certain depth frame t_i by assigning stable valid depth values for each pixel of t_i. Then the sliding window moves one frame further to enhance the next frame t_{i+1}. It is worth to mention that, from the $(n+1)$th frame ($(t+4)$th frame on the output stream in Figure 4) we would observe the enhanced depth frame. For the $(n+2)$th frame ($(t+5)$th frame in Figure 4) and later ones, we just have to wait for one extra frame's calculation.

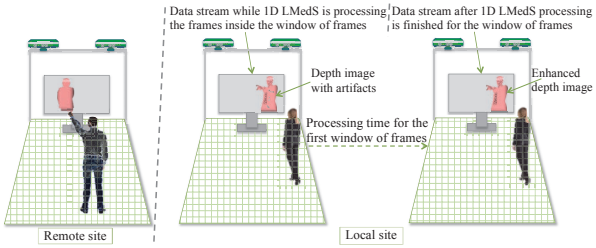


Figure 8: **Illustration of using 1D LMedS in a telepresence scenario** – the depth image on the right screen of the local site is enhanced by 1D LMedS while the left depth image shows artifacts (since the processing of the frames inside the sliding window is in progress).

6. Example use case: telepresence

1D LMedS can be applied to enhance the depth frames in many application areas such as in 3D reconstruction of a scene, tracking or forensic analysis. Telepresence, an emerging application area which often needs 3D scene reconstruction, can benefit from the 1D LMedS depth enhancement method. In case of telepresence, our 1D LMedS can be used to enhance the stream of frames received from the remote site of the communication. There are typically two sides of communication in a traditional telepresence system, which are: local site and remote site. Users staying at each site can communicate and interact with each other via network channel; their surrounding environment is captured and transmitted to the other end of the communication site. Figure 8 illustrates a typical telepresence scenario along with how 1D LMedS can be applied to such scenario.

Since, in case of 1D LMedS, we use a sliding window of frames to enhance the t_ith frame, we will see the enhanced t_ith frame after processing the frames inside the sliding window. While the frames inside the sliding window are being processed, the users will observe the frames with artifacts, but as soon as the processing is finished and the sliding window moves to the next window of frames, they will start observing the enhanced stream. As stated in the last paragraph of Section 5, for the later sliding window, only one frame's worth calculation time is needed since the 1D LMedS calculation for others frames have already been done while processing the frames inside the previous sliding window. Similarly, 1D LMedS can be applied to other applications (e.g. in tracking) which involves a sequence of frames to be processed before carrying out a certain task.

7. Results and discussion

In this section, we perform a set of experiments to evaluate the overall performance of proposed 1D LMedS approach of enhancing a depth image sequence. Here, we use several datasets among which some are captured locally with Kinect v1 and three others (one is from Camplani et. al. [8] and two are from Middlebury RGB-D database [39]) are used as reference datasets. Here we use depth data from Kinect v1 sensor because, most of the reference works have been pursued by using Kinect v1 sensor. The local datasets and the dataset from [8] have the resolution of 480×640 while Middlebury datasets have 555×690 resolution. We use the Middlebury datasets for the quantitative evaluation and other datasets for the qualitative evaluation.

To compare the results, we use five state-of-the-art methods which are the joint bilateral filtering approach [8, 28] from Camplani et. al., the depth-color fusion method [9] from Garcia et. al., depth smoothing method [37] from Nguyen et. al. and the trilateral approach [31] from Wang et.al. To achieve a fair comparison scenario during the experiments, we tuned the parameters of these methods to achieve the best results as stated in those respective methods. Our CPU implementation of 1D LMedS is performed on a Macbook Pro with an Intel Core i7 2.3 GHz processor and 8GB RAM. In Section 7.1 and Section 7.2, we present the qualitative and the quantitative performance comparison respectively.

7.1. Performance on real-world depth sequences

We tested 1D LMedS on various datasets which are captured locally and on a dataset from [8]. Since we did not find any reference sequence of depth frames which was captured with a camera that was kept static while recording, we used local datasets and image sequences from [8] to evaluate the performance of 1D LMedS.

Figure 9 shows the performance of 1D LMedS on three different datasets. Here, the images on the top three rows are captured locally and the images of 4th row are taken from [8]. Here, the images in the 2nd row and the images in the 3rd, 4th and 5th column of 3rd and 4th rows of Figure 9 show the performance of 1D LMedS with different number of frames in the sliding window. As expected, with the number of frames increased in the sliding window, the chance of getting better values for the pixels are also increased and hence, we observe that more and more artifacts are removed with bigger sliding windows. In Figure 10 we show the comparison results among two reference depth enhancement methods ([9] and [37]) and our 1D LMedS on local datasets and the dataset of [8]. From this figure, we can see that 1D LMedS performs quite well in removing the artifacts compared to the reference methods.

We applied 1D LMedS to both static and moving parts of the captured scene. 1D LMedS worked quite

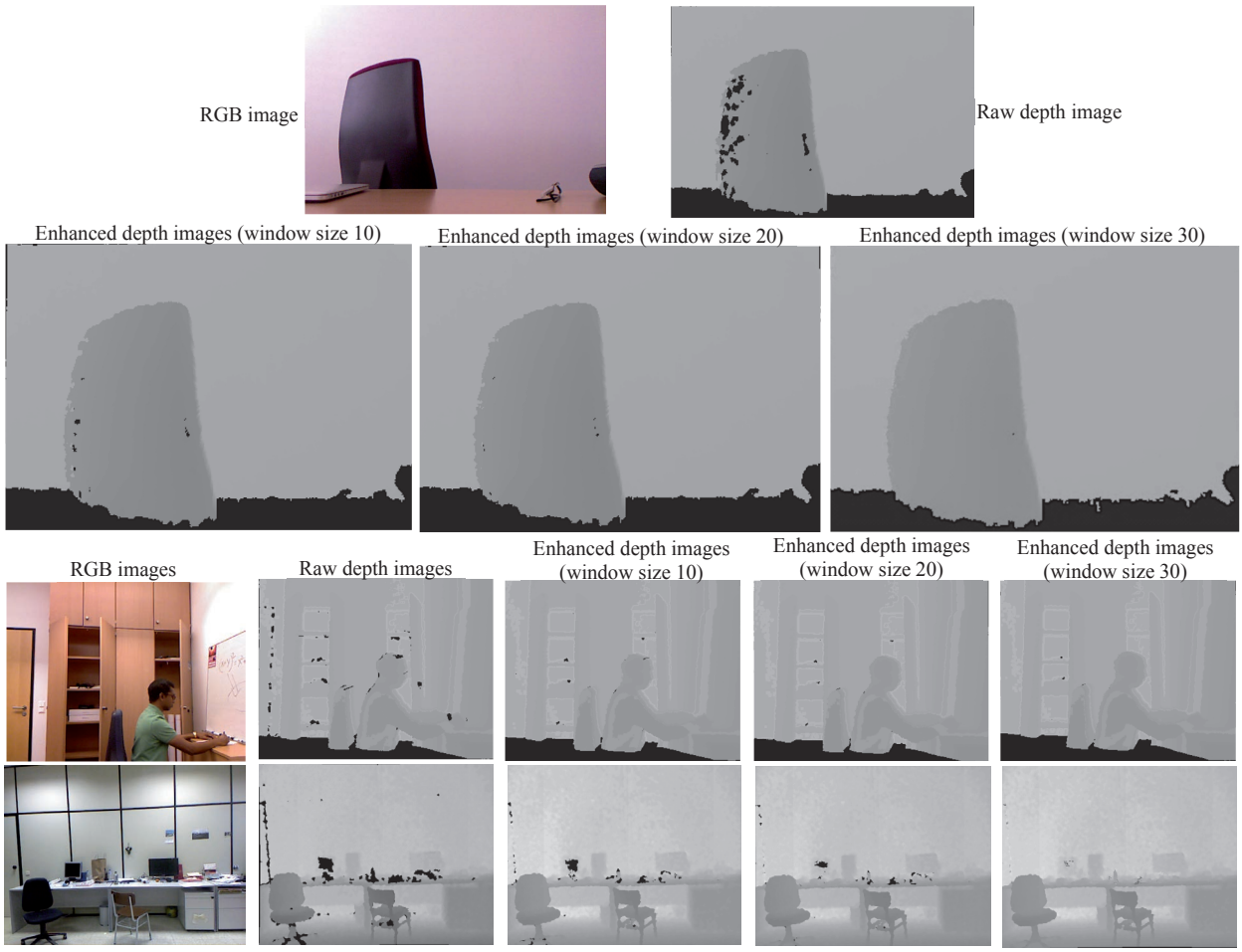


Figure 9: **1D LMedS performance with varying window size on different datasets** – images on the 2nd row and on the 3rd, 4th and 5th columns of 3rd and 4th rows are the enhanced depth images by 1D LMedS with window size 10, 20 and 30 respectively. The test image on the 4th row is the 20th frame which is taken from dataset [8] and the images on the other three rows are captured locally on various days.

well with static parts of the scene (see Figure 11(a – e)) while stabilizing the depth values over consecutive frames (see Figure 11(d – e)). We can also perceive the performance of 1D LMedS by comparing Figure 11(e) to Figure 2(b). Here, we used 25 frames in the sliding window. For the moving parts of the scene, 1D LMedS also performed good in filling the missing depth values (see Figure 12 and Figure 13). It is worth to mention that, for a scene with moving objects, we use a smaller sliding window size than a scene with static objects while the implementation of 1D LMedS remains the same in both cases. If we use similar window sizes in dynamic scenes as in static scenes, we would observe ghosting artifacts in the resulting images (see the gray shadow near the edge of the chair, marked with purple rectangle and zoomed for clear visualization, in Figure 13). Another issue to mention here is, when a pixel does not have a valid depth value throughout the whole sliding window, that pixel continues to remain as a black spot. It is also true for a large area with missing depth value (see the orange ellipse marked area in Figure 13). Increasing window size in the hope for finding valid depth values for such pixels might work but that would result in lower computation speed and increased overall latency [40]. In Figure 12, we show the original and enhanced images after pixel aligning the original and enhanced depth images to their corresponding color images; the images are rendered in OpenGL. For the results of Figure 12 and 13, we used 30 and 35 frames, respectively, in the sliding window.

With a bit more careful observation of the enhanced depth images, we found out that our 1D LMedS preserves the sharpness of the edges of different objects better than most of these mentioned methods, see the red

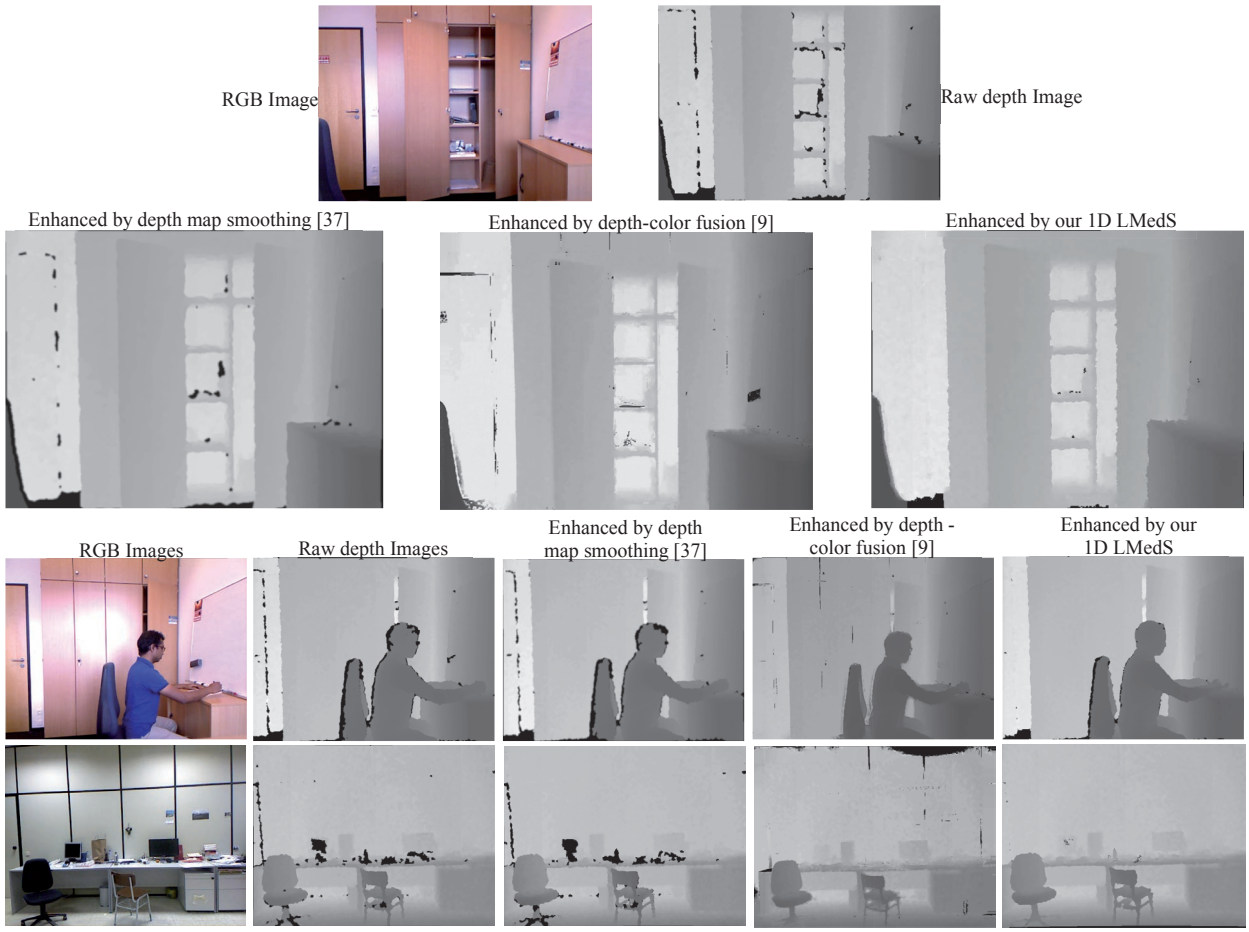


Figure 10: **Comparison of results on datasets captured locally and from [8] with our 1D LMedS, depth smoothing method [37] and fusion based enhancement[9]** – the images on 2nd row and on the 3rd, 4th and 5th columns of 3rd and 4th rows are the enhanced depth images by method [37], by method [9] and by our 1D LMedS respectively. The test image on the 4th row is the 20th frame of the dataset [8] and the images on the other three rows are captured locally.

marked area and the zoomed parts in Figure 14. Here, we take the method of [31] to compare the enhancement results. On the surface of the objects, the method from [31] produces quite nice results, however it suffers along the edge of the objects. Moreover, we also notice that the method in [31] highly depends on a very accurate registration between the color and the depth image pairs. If the color image and depth image pair is not perfectly aligned, this method shows blurry object edges. This is due to the fact that the color of an object in the color image can be considered to be belonging to a different object in the corresponding depth image pair. Here, for example on the first and third row in Figure 14, the edge of the nose and mouth of the person, and the edge of the right side of the chair are mistakenly considered to be belonging to both the background and the foreground. This results in blurry edges which have depth values that are mixed from depth values of different objects. Besides this, when the background and the foreground objects have same color (see the diamond marked areas on the color and depth images on the first row of Figure 14; here the hair of the person and the background have the same color), it also produces artifacts near the edges of the objects (see the yellow dotted region on the zoomed part of Figure 14) in the depth image. Here too, the algorithm mixes the depth values from both objects. In 1D LMedS, we do not face these issues since we only need the depth sequence for the enhancement.

Moreover, 1D LMedS takes far less time to enhance a single frame than other methods. For example, to enhance one frame of Dataset 1, 1D LMedS takes only 0.14 second (see the value in the 4th column of 2nd row in Table 1), whereas, the method in [31] takes more than 20 minutes (1200 seconds) due to its time consum-

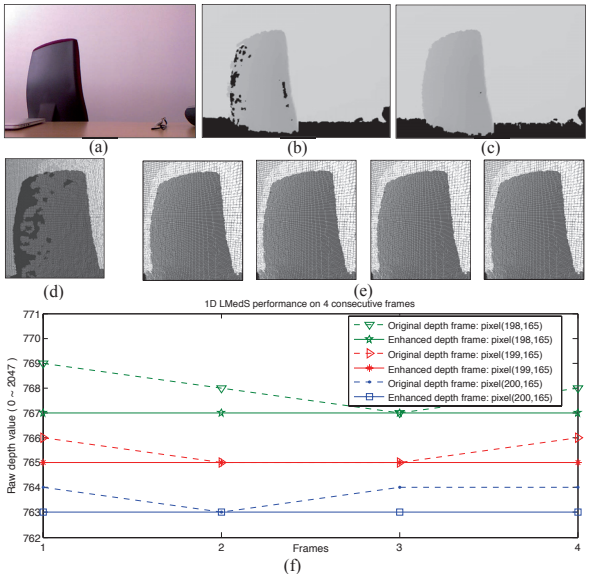


Figure 11: **Results with static object** – (a) color image, (b,d) raw depth image, (c) enhanced depth image by 1D LMedS, (e) 4 consecutive depth frames (zoomed for clear visibility) with stable depth values, (f) the depth values (before and after applying 1D LMedS) of 3 pixels on the 4 frames of (b) show that the enhanced frames have stable depth values for all of these 3 pixels over the consecutive frames.

Figure 12: **Results-1 with moving object** – (a) image with raw depth map, (b) image with enhanced depth map by 1D LMedS. Here, the depth images are pixel aligned to their corresponding color images.

ing energy minimization computation. Due to this high computation, similar color image based methods such as [9, 30–32] are not suitable for real-time applications.

Table 1 shows the performance of proposed 1D LMedS in terms of frames per second (*fps*) on the four datasets we used in our experiments. In Table 1, we refer *Dataset 1* to the images on the 1st and 2nd rows of Figure 9, *Dataset 2* to the images on the 3rd row of Figure 9, *Dataset 3* to the images on the 4th row of Figure 9 and *Dataset 4* to the images on the 2nd row of Figure 10. We show here the performance by varying the number of frames in the sliding window. As an example, for the *Dataset 1* with 25 frames in the sliding window (*ws* = 25), we achieved a performance of 8 *fps* with our CPU implementation of 1D LMedS.

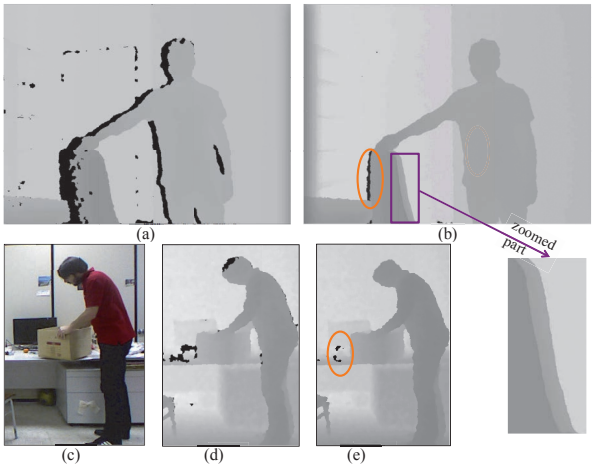


Figure 13: **Results-2 with moving object** – (a,d) raw depth images, (b,e) enhanced depth images by 1D LMedS, (c) color image (part of frame 244 from [8]). The black area marked with orange ellipses in (b) and (e) occurs because the depth sensor cannot visualize that area and hence, there is no depth information for that area. The ghosting artifact (a trail of gray shadow) marked with purple rectangle and zoomed in (b) occurs due to two reasons: the object is moving and we take 35 frames in the sliding window.

Table 1: Performance, in frames per second (*fps*), of 1D LMedS by varying the sliding window size (*ws*) on four local datasets.

Datasets	*fps* (*ws:10*)	*fps* (*ws:20*)	*fps* (*ws:25*)	*fps* (*ws:30*)
Dataset 1	12	10	8	7
Dataset 2	13	11	8	7
Dataset 3	12	9	7	6
Dataset 4	12	10	7	6

In Figure 15 and Figure 16, we depict a comparison result between the method [37] and 1D LMedS on a sequence of 100 frames on two local datasets. The depth values plotted on both figures are for the pixels marked by the yellow dot on the depth images. In both cases, we observe that our proposed 1D LMedS outperforms the method [37] in removing the invalid depth values while stabilizing the valid depth values.

7.2. Performance on artificially degraded Kinect depth sequences

Here, we perform yet another comparison experiment for an extensive evaluation of the performance of our 1D LMedS. This time we take reference datasets (the 'Art' and the 'Book') which is publicly available in the Middlebury depth datasets [39]. Since we perform a quantitative experiment here, we use standard metrics such as PSNR (Peak Signal-to-Noise Ratio) and SSIM

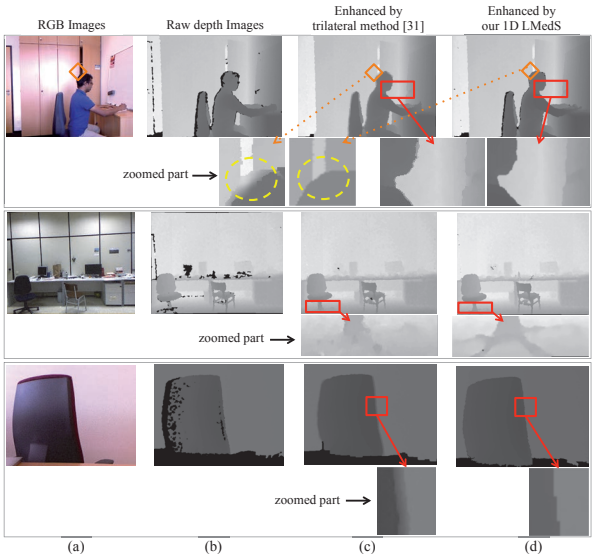


Figure 14: **Detailed comparison of results based on the marked areas of the three datasets with our 1D LMedS and trilateral method [31]** – a) color images, (b) depth images, (c, d): enhanced depth images by trilateral method [31], by our 1D LMedS method respectively. The test image on the 2nd row is the 20th frame of the dataset [8] and the images on the other 2 rows are captured locally.

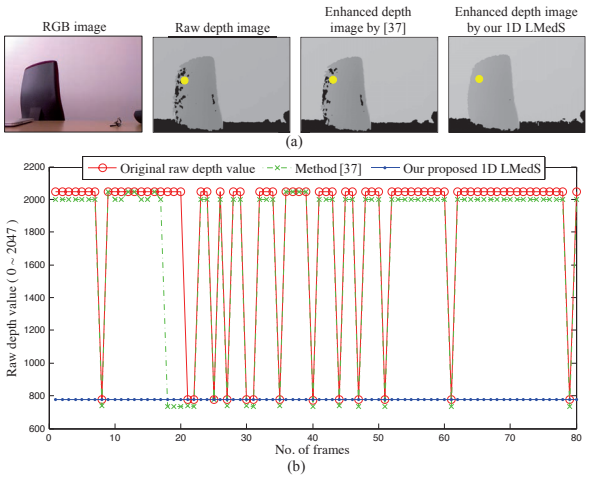


Figure 15: **Comparison of results-1 with method [37] and proposed 1D LMedS on 100 frames of a local dataset** – (a) from left to right – color image, raw depth image, enhanced depth images by [20] and by 1D LMedS, (b) depth enhancement results (for the pixel marked by yellow circle) by method [37] and 1D LMedS. 1D LMedS removes more invalid depth values than method [37] and it also stabilizes the valid depth values which is shown by the blue filled-circle marked line in (b). In (a), the enhanced frame is the 27th frame of the 100 frames and in (b), we show the depth values for 80 enhanced frames (sliding window contains the first 20 frames).

(Structural SIMilarity) index for the reference datasets. Now, since ground truth depth images are required to compute the PSNR and SSIM metrics and, since Middlebury datasets provide the ground truth depth images, we follow the procedure stated in [32] to generate artificially Kinect-like degraded depth images. In [32], the authors create the structural missing phenomenon of Kinect's depth images along the areas with depth discontinuities and the random missing phenomenon on the areas which are flat.

Since we need a sequence of frames for 1D LMedS and since the Middlebury datasets do not provide sequence of depth images, we take multiple copies of 'Art' and 'Book' depth images and then apply the artificial degradation process from [32] randomly on each of these images. Hence, we get a sequence of frames which contain Kinect-like degradation on different places for each of the frames. Then, we apply 1D LMedS on these sequence of frames and get the final enhanced depth image. We can consider this sequence of frames as a part of a static scene where artifacts occurs over the temporal domain, which we have also seen in our previous experiment in Figure 11. For the other methods [28, 31] to which we compare the performance of 1D LMedS, we use the color and depth images from the 'Art' and 'Book' datasets, and the artificial Kinect-like degraded image from [32]. For these methods [28, 31], the depth recovery process is performed on this artificially degraded depth image.

Table 2: Quantitative comparison results of different methods on standard datasets with artificial Kinect-like degradations. Here, for our 1D LMedS, sliding window size (*ws*) is set to 20.

	Datasets			
	Art		Book	
	PSNR (dB)	SSIM	PSNR (dB)	SSIM
Camplani [28]	29.1845	0.9442	26.5911	0.9416
Wang [31]	30.3104	0.9587	26.8854	0.9492
Our 1D LMedS	32.2324	0.9723	28.3123	0.9624

To compute the PSNR and SSIM metrics, we consider the ground truth depth images from the 'Art' and 'Book' datasets as the reference depth images, and the enhanced depth images as the images to be compared against the reference depth images. Figure 17 shows the color images, artificially degraded depth images and enhanced depth images results by [28], [31] and our 1D LMedS respectively. Table 2 shows the PSNR and SSIM metrics which reflects the quantitative comparison among these methods.

When we take a closer look at the results (see Figure 17) of different depth image enhancement methods, we observe that all of them perform fine to recover the missing depth values on the surface areas which are flat. However, the difference among these methods lies in

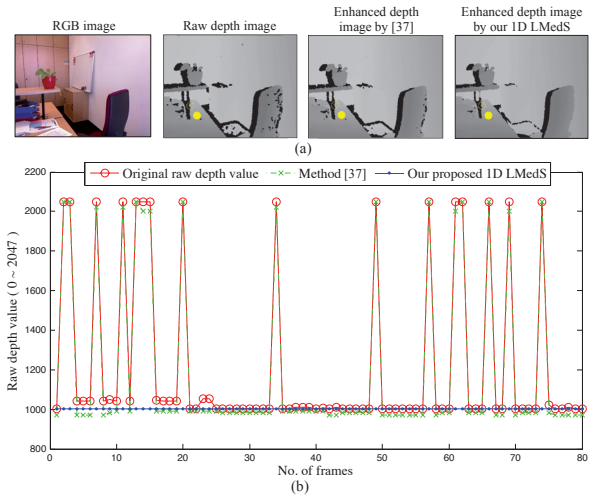


Figure 16: **Comparison of results-2 with method [37] and 1D LMedS on 100 frames of another local dataset** – here also, 1D LMedS removes more invalid depth values than method [37] and stabilizes the valid depth values (see the blue filled-circled line in (b)).

Table 3: Quantitative comparison results of 1D LMedS by varying the sliding window size (*ws*) on standard datasets with artificial Kinect-like degradations.

Window size(*ws*)	Datasets			
	Art		Book	
	PSNR (dB)	SSIM	PSNR (dB)	SSIM
10	31.7458	0.9651	27.8957	0.9573
20	32.2324	0.9723	28.3123	0.9624
25	32.6981	0.9774	28.7923	0.9682
30	33.0148	0.9802	29.1635	0.9710

the enhanced areas near the edges of the objects where the structural discontinuity occurred inside the degraded depth images. Although, all of these methods recover the edges, the results from methods [28] and [31] show blurry object edges, whereas our 1D LMedS preserves the sharpness of the object's edges. The result of this subjective comparison is also reflected in the PSNR and SSIM metrics in Table 2 where 1D LMedS outperforms the other methods. We show the PSNR and SSIM metrics of our 1D LMedS with different sliding window size (*ws*) on the Middlebury datasets with artificial Kinect-like degradations in Table 3.

As an alternative of using the depth values of only the current pixel in the sliding window of frames, we also considered using regional information surrounding the current pixel. We can use a patch of size p×p and take the average of the depth values of the pixels inside this patch and use that value as the current pixel's depth value while processing the frames in the sliding window. Here, the method of 1D LMedS remains the same, except, instead of taking just one depth value from the current pixel, we consider the patch as a vector of consecutive depth values. We test this approach for the pixels with invalid depth values. While testing this idea, we found out that taking a smaller patch (3×3) and smaller sliding window size doesn't necessarily improve the depth enhancement performance; because, there were not enough valid depth values in the patches. However, we got a slightly better result with patch size 7×7; the improvements were mostly in the areas which have large areas with invalid depth values. However, we observe a noticeable decrease in computational speed and increase in latency with such bigger patch size. Therefore, we decide not to include the outcome from this idea in this article and plan to improve on this idea with a better formulation and keep it as future work.

7.3. Limitations

Our approach has some limitations. Firstly, we cannot recover invalid depth values or noise that persist in one region of the depth image sequence with our 1D LMedS method (see the black area marked with orange ellipses in Figure 13(b) and (e). In this case, regional information, such as using a patch of neighboring pixels, might be useful to refine the large areas with artifacts. Secondly, we may see ghosting artifacts if we set the size of the sliding window too large (see the gray-shadowed area marked with a purple rectangle in Figure 13(b)). Ghosting artifacts may also occur for fast-moving objects, where depth values change drastically from one frame to another. We considered using our previous method from [35] base on optical flow motion vectors to further reduce ghosting. That method, however, works well with small temporal windows sizes (the experiments in [35] used a window with five image frames) and increasing the window size might increase ghosting due to accumulating errors in optical flow calculations. Thus, optimal temporal windows sizes for the method described in this paper and the method from [35] are not compatible. For future work, we can envision a meta algorithm that classifies image regions based on optical flow. The method from [35] could then be applied to regions with large optical flow, while the method proposed in this paper could be applied to regions with smaller optical flow. Although, 1D LMedS exhibits fewer edge-bleeding artifacts, it could be further reduced by using an anisotropic diffusion approach along with structural similarity of depth and color frames. Moreover, a GPU implementation of 1D LMedS would allow us to enhance the depth frames in real time.

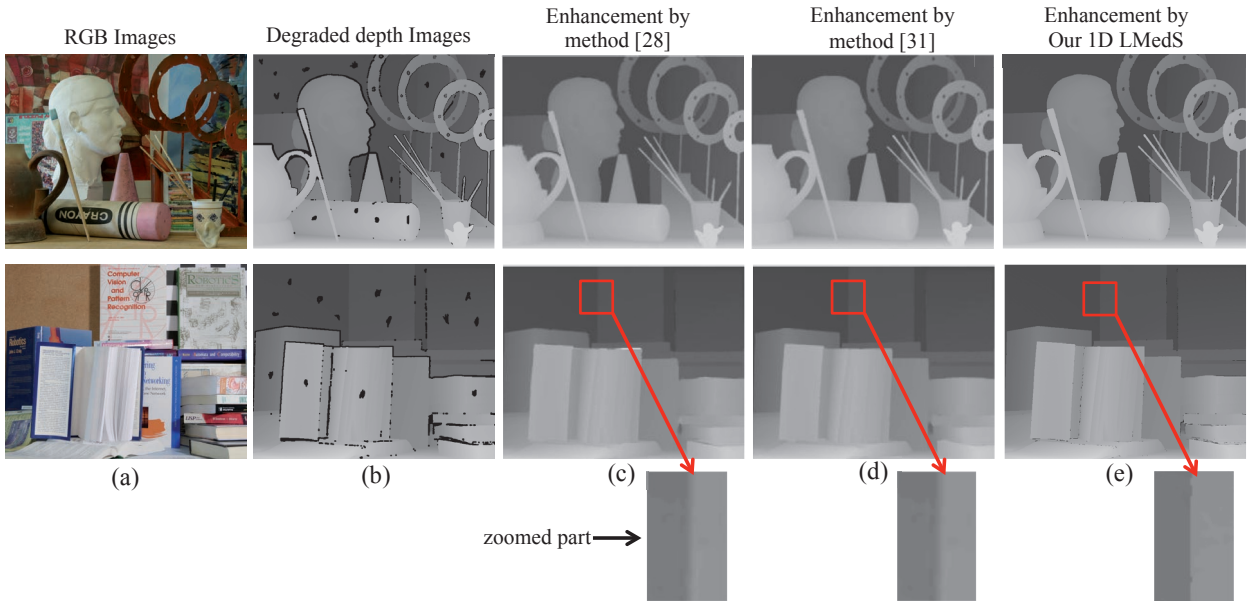


Figure 17: **Subjective comparison results on datasets 'Art' and 'Book' from Middlebury database [39] with artificial Kinect-like degradations** – a) color images, (b) degraded depth images, (c – e): enhanced depth images by method [28], method [31] and our 1D LMedS respectively. The zoomed parts below the 2nd row show detailed comparison, among these methods, on preserving the sharpness of the object's edges.

8. Conclusion

We have introduced a robust 1D LMedS approach which can efficiently enhance the depth images generated by depth cameras. The output of our proposed approach shows good performance in filling the invalid depth values and stabilizing the valid values for static and dynamic parts of a scene. We have used local datasets and reference datasets to show the performance of 1D LMedS. For all the test datasets, our 1D LMedS method exhibits good performance. Moreover, we have also used other reference methods on the test datasets to compare results with 1D LMedS; in this case also, 1D LMedS performs quite good comparing with other methods. The proposed 1D LMedS is computationally less expensive than higher dimensional LMedS and its implementation procedure on depth frames is efficient.

Since the proposed 1D LMedS performs quite well in removing the invalid depth values and stabilizing the inconsistent valid depth values (considered as outliers), one might apply 1D LMedS to address the so called 'flying pixel' [41] issues found in the content generated by ToF sensors. Such pixel's depth values could also be identified as outliers and be stabilized with 1D LMedS.

Acknowledgements

This work was supported by the EU FP7 Marie Curie ITN "DIVA" under REA Grant Agreement No.290227 and the DFG Research Grant STA 708/5-1 "SMOOTH". We would like to thank Dr. Massimo Camplani (Grupo de Tratamiento de Imágenes, Universidad Politécnica de Madrid) for providing dataset and Dr. Frederic Garcia (Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg) for his valuable comments during the preparation of this article.

References

[1] A. Maimone, H. Fuchs, Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras, in: 10th IEEE ISMAR, 2011, pp. 137–146.

[2] M. Urschler, J. Hller, A. Bornik, T. Paul, M. Giretzlehner, H. Bischof, K. Yen, E. Scheurer, Intuitive presentation of clinical forensic data using anonymous and person-specific 3d reference manikins, Forensic Science International 241 (0) (2014) 155 – 166.

[3] M. Dou, H. Fuchs, Temporally enhanced 3d capture of room-sized dynamic scenes with commodity depth cameras, in: IEEE Virtual Reality (VR), 2014, pp. 39–44.

[4] S. Matyunin, D. Vatolin, Y. Berdnikov, M. Smirnov, Temporal filtering for depth maps generated by kinect depth camera, in: 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011, pp. 1–4.

[5] A. Amamra, N. Aouf, Gpu-based real-time rgbd data filtering, Journal of Real-Time Image Processing (2014) 1–18.

[6] K. He, J. Sun, X. Tang, Guided image filtering, in: 11th European Conference on Computer Vision (ECCV), Springer Berlin Heidelberg, 2010, pp. 1–14.

[7] J. Liu, X. Gong, J. Liu, Guided inpainting and filtering for kinect depth maps, in: 21st International Conference on Pattern Recognition (ICPR), 2012, pp. 2055–2058.

[8] M. Camplani, T. Mantecon, L. Salgado, Depth-color fusion strategy for 3-d scene modeling with kinect, IEEE Transactions on Cybernetics 43 (6) (2013) 1560–1571.
[9] F. Garcia, D. Aouada, T. Solignac, B. Mirbach, B. Ottersten, Real-time depth enhancement by fusion for rgb-d cameras, Computer Vision, IET 7 (5) (2013) 1–11.
[10] S. Beck, A. Kunert, A. Kulik, B. Froehlich, Immersive group-to-group telepresence, IEEE Transactions on Visualization and Computer Graphics 19 (4) (2013) 616–625.
[11] L. Chen, H. Lin, S. Li, Depth image enhancement for kinect using region growing and bilateral filter, in: 21st International Conference on Pattern Recognition, 2012, pp. 3070–3073.
[12] Q. Yang, N. Ahuja, R. Yang, K.-H. Tan, J. Davis, B. Culbertson, J. Apostolopoulos, G. Wang, Fusion of median and bilateral filtering for range image upsampling, IEEE Transactions on Image Processing 22 (12) (2013) 4841–4852.
[13] J. Dolson, J. Baek, C. Plagemann, S. Thrun, Upsampling range data in dynamic environments, in: IEEE Computer Vision and Pattern Recognition (CVPR), 2010.
[14] F. Qi, J. Han, P. Wang, G. Shi, F. Li, Structure guided fusion for depth map inpainting, Pattern Recognition Letter 34 (1) (2013) 70–76.
[15] C. Richardt, C. Stoll, N. A. Dodgson, H.-P. Seidel, C. Theobalt, Coherent spatiotemporal filtering, upsampling and rendering of rgbz videos, Computer Graphics Forum 31 (2) (2012) 247–256.
[16] B.-S. Lin, M.-J. Su, P.-H. Cheng, P.-J. Tseng, S.-J. Chen, Temporal and spatial denoising of depth maps, Sensors 15 (8) (2015) 18506–18525.
[17] S.-Y. Kim, J.-H. Cho, A. Koschan, M. Abidi, Spatial and temporal enhancement of depth images captured by a time-of-flight depth sensor, in: 20th International Conference on Pattern Recognition (ICPR), 2010, pp. 2358–2361.
[18] A. B. M. T. Islam, C. Scheel, R. Pajarola, O. Staadt, Robust enhancement of depth images from kinect sensor, in: 2015 IEEE Virtual Reality (VR), 2015, pp. 197–198.
[19] A. B. M. T. Islam, C. Scheel, R. Pajarola, O. Staadt, Depth image enhancement using 1d least median of squares, in: Computer Graphics International (CGI'15), 2015.
[20] P. J. Rousseeuw, Least median of squares regression, Journal of the American statistical association 79 (388) (1984) 871–880.
[21] S. Bhattacharya, S. Gupta, K. Venkatesh, High accuracy depth filtering for kinect using edge guided inpainting, in: International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, pp. 868–874.
[22] L. Zhang, P. Shen, S. Zhang, J. Song, G. Zhu, Depth enhancement with improved exemplar-based inpainting and joint trilateral guided filtering, in: IEEE International Conference on Image Processing (ICIP), 2016, pp. 4102–4106.
[23] T.-W. Hui, K. N. Ngan, Motion-depth: Rgb-d depth map enhancement with motion and depth in complement, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 3962–3969.
[24] K. Lai, L. Bo, X. Ren, D. Fox, A large-scale hierarchical multi-view rgbd object dataset, in: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 1817–1824.
[25] J. Wasza, S. Bauer, J. Hornegger, Real-time preprocessing for dense 3-d range imaging on the gpu: Defect interpolation, bilateral temporal averaging and guided filtering, in: International Conference on Computer Vision Workshops, 2011, pp. 1221–1227.
[26] A. Maimone, J. Bidwell, K. Peng, H. Fuchs, Enhanced personal autostereoscopic telepresence system using commodity depth cameras, Computers & Graphics 36 (7) (2012) 791 – 807.
[27] A. Telea, An image inpainting technique based on the fast marching method, Graphics Tools 9 (1) (2004) 25–36.
[28] M. Camplani, L. Salgado, Efficient spatio-temporal hole filling strategy for kinect depth maps, in: Proc. SPIE, Vol. 8290, 2012, pp. 82900E–82900E–10.
[29] J. Zhu, L. Wang, R. Yang, J. Davis, Fusion of time-of-flight depth and stereo for high accuracy depth maps, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[30] L. Sheng, K. N. Ngan, Depth enhancement based on hybrid geometric hole filling strategy, in: 2013 IEEE International Conference on Image Processing, 2013, pp. 2173–2176.
[31] Z. Wang, J. Hu, S. Wang, T. Lu, Trilateral constrained sparse representation for kinect depth hole filling, Pattern Recognition Letter 65 (C) (2015) 95–102.
[32] J. Yang, X. Ye, K. Li, C. Hou, Y. Wang, Color-guided depth recovery from rgb-d data using an adaptive autoregressive model, IEEE Transactions on Image Processing 23 (8) (2014) 3443–3458.
[33] S. Milani, G. Calvagno, Correction and interpolation of depth maps from structured light infrared sensors, Signal Processing: Image Communication 41 (2016) 28 – 39.
[34] Y. Yang, Q. Liu, R. Ji, Y. Gao, Dynamic 3d scene depth reconstruction via optical flow field rectification, Public Library of Science (PloS ONE) 7 (11) (2012) e47041.
[35] R. Avetisyan, C. Rosenke, M. Luboschik, O. Staadt, Temporal filtering of depth images using optical flow, in: Proceedings of the 24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'16), 2016.
[36] K. Khoshelham, E. O. Elberink, Accuracy and resolution of kinect depth data for indoor mapping applications, Sensors 12 (2) (2012) 1437–1454.
[37] C. Nguyen, S. Izadi, D. Lovell, Modeling kinect sensor noise for improved 3d reconstruction and tracking, in: 2nd International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012, pp. 524–530.
[38] P. J. Rousseeuw, A. M. Leroy, Robust Regression and Outlier Detection, John Wiley & Sons, Inc., New York, NY, USA, 1987.
[39] MiddleburyDatasets, `http://vision.middlebury.edu/stereo/data/`, accessed: 2017-05-16 (2013).
[40] S. Ohl, M. Willert, O. Staadt, Latency in distributed acquisition and rendering for telepresence systems, IEEE Tran. on Visualization and Computer Graphics 21 (12) (2015) 1442–1448.
[41] A. Kolb, E. Barth, R. Koch, R. Larsen, Time-of-flight sensors in computer graphics, in: Proc. Eurographics (State-of-the-Art Report), 2009.