



**HAL**  
open science

# Hierarchical, Dense and Dynamic 3D Reconstruction Based on VDB Data Structure for Robotic Manipulation Tasks

Carlos Mateo, Juan Antonio Corrales Ramon, Youcef Mezouar

► **To cite this version:**

Carlos Mateo, Juan Antonio Corrales Ramon, Youcef Mezouar. Hierarchical, Dense and Dynamic 3D Reconstruction Based on VDB Data Structure for Robotic Manipulation Tasks. *Frontiers in Robotics and AI*, 2021, 7, 10.3389/frobt.2020.600387 . hal-03187321

**HAL Id: hal-03187321**

**<https://uca.hal.science/hal-03187321>**

Submitted on 31 Mar 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Hierarchical, Dense and Dynamic 3D Reconstruction based on VDB data structure for Robotic Manipulation Tasks

Carlos M. MATEO AGULLO<sup>1, 2\*</sup>, Juan Antonio Corrales Ramón<sup>1, 2</sup>, Youcef Mezouar<sup>1, 2</sup>

<sup>1</sup>Sigma Clermont, France, <sup>2</sup>UMR6602 Institut Pascal (IP), France

*Submitted to Journal:*  
Frontiers in Robotics and AI

*Specialty Section:*  
Robotic Control Systems

*Article type:*  
Original Research Article

*Manuscript ID:*  
600387

*Received on:*  
29 Aug 2020

*Revised on:*  
06 Dec 2020

*Journal website link:*  
[www.frontiersin.org](http://www.frontiersin.org)

In review

---

### *Conflict of interest statement*

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest

### *Author contribution statement*

All authors contribute equally

### *Keywords*

Robot Manipulation, 3D visual perception, Dense reconstruction, robot vision, high performance computing

### *Abstract*

Word count: 154

This paper presents a reviewed approach to implement hierarchical, dense and dynamic reconstruction method based on VDB (variational dynamics B+ Trees) data structure for robot tasks.

Scene reconstruction is done by the integration of depth-images using the Truncated Signed Distance Field (TSDF).

Nowadays, dense reconstruction domain is ruled by three major space representations, complete volumes, hashing voxels and hierarchical volumes.

Here, we propose design the reconstruction method based on dynamic trees can provide similar reconstruction result than current state-of-art methods, but with a direct multi-level representation at expenses of just a slightly higher computational cost, being still real-time.

Additionally, this representation provide two major advantages against the other, hierarchical and unbounded space representation.

The proposed method is optimally implemented to be used on a GPU architecture, exploiting the parallelism skills of this hardware. A series of experiments will be presented to prove the performance, qualitatively, of this approach in a robot arm platform.

### *Contribution to the field*

Visual sensing is an indispensable part in most of the robotic manipulation tasks in the literature. This is mainly due to the fact that most of the robotic manipulation strategies are based on the assumption that the surface of the object is continuously tracked. But nowadays, many solutions just use image-base approaches where instead of using surface tracker they are contour-base strategies. But here, we propose to use 3D visual information, thus, we point to use 3D objects surface as sensing feedback in the control schemes, that is, we propose to include 3D reconstruction method as part of robotic manipulation strategies. Concretely, this article proposes the use of a data structure typically used in data theory and computer graphics to achieve reconstruction with different levels of detail. In the same way, this strategy generates in an isolated way the topology of the objects from their texture (colors, curvatures, etc.).

### *Ethics statements*

#### *Studies involving animal subjects*

Generated Statement: No animal studies are presented in this manuscript.

#### *Studies involving human subjects*

Generated Statement: No human studies are presented in this manuscript.

#### *Inclusion of identifiable human data*

Generated Statement: No potentially identifiable human images or data is presented in this study.

### *Data availability statement*

Generated Statement: The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

# Hierarchical, Dense & Dynamic 3D Reconstruction based on VDB data structure for Robotic Manipulation Tasks

Carlos M. Mateo and Juan A. Corrales and Youcef Mezouar

*Université Clermont Auvergne, SIGMA Clermont, Institut Pascal BP 10448, F-63000 Clermont-Ferrand, France.*

Correspondence\*:

Carlos M. Mateo

cmateoagul@sigma-clermont.fr

## 2 ABSTRACT

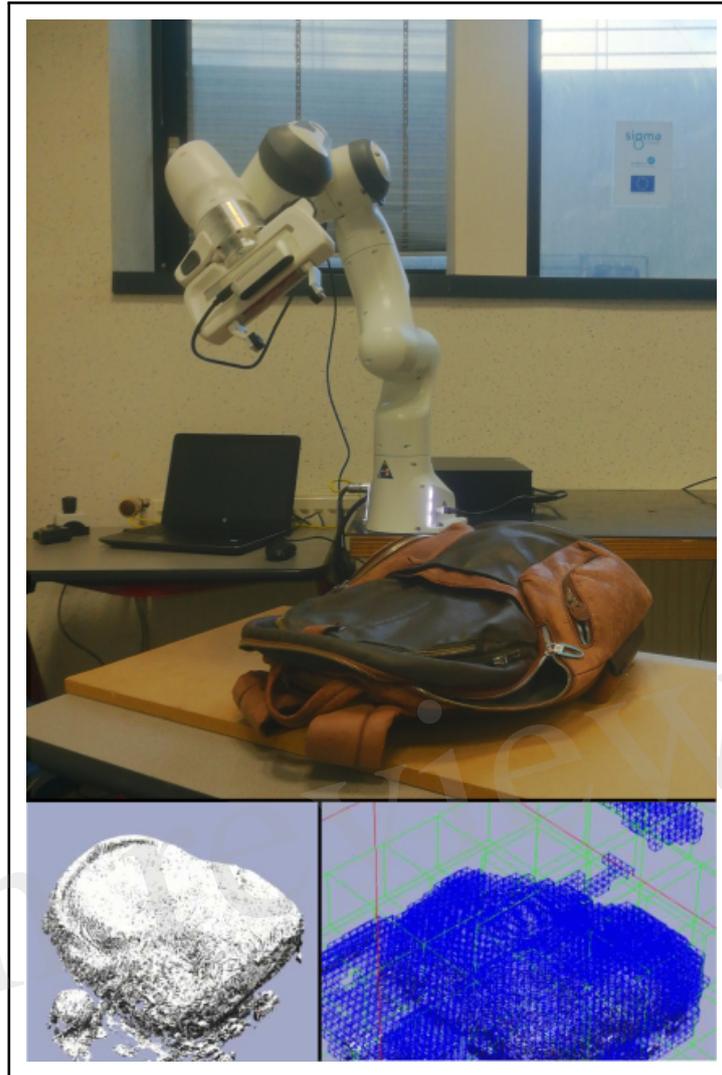
3

4 This paper presents a novel approach to implement hierarchical, dense and dynamic  
5 reconstruction of 3D objects based on the VDB (Variational Dynamic B+ Trees) data structure  
6 for robotic applications. The scene reconstruction is done by the integration of depth-images  
7 using the Truncated Signed Distance Field (TSDF). The proposed reconstruction method is  
8 based on dynamic trees in order to provide similar reconstruction results than current state-  
9 of-the-art methods (i.e. complete volumes, hashing voxels and hierarchical volumes) in terms  
10 of execution time, but with a direct multi-level representation that remains real-time. In fact,  
11 this representation provides two major advantages: it is a hierarchical and unbounded space  
12 representation. The proposed method is optimally implemented to be used on a GPU architecture,  
13 exploiting the parallelism skills of this hardware. A series of experiments will be presented to  
14 prove the performance of this approach in a robot arm platform.

15 **Keywords:** robot manipulation, 3d visual perception, dense reconstruction, robot vision, high performance computing

## 1 INTRODUCTION

16 Industrial robotic research has been extremely prolific in the last decades, with special interest in  
17 applications such as welding, painting and pick-and-place of objects. However, the performance of most of  
18 them relays on the precise visual perception of the workplace so that the robot can react on real-time to  
19 changes on it. An interesting tool for implementing this perception capability is 3D dense reconstruction.  
20 Although 3D dense reconstruction is an well-established field in computer vision and graphics, most of the  
21 new proposed methods are not adapted to the constrains imposed by complex industrial robotic tasks. For  
22 instance, when robots need to manipulate deformable objects, current reconstruction methods fail since  
23 they are based on the assumption of the presence of rigid objects in static scenarios (Zeng et al. (2013),  
24 Whelan et al. (2016) and Puri et al. (2017)). Another well-known problem is the drifting in textureless  
25 scenarios during the camera pose estimation, what implies erroneous reconstructions. Thus, most of the



**Figure 1.** Robot platform used to evaluate our approach. Top: Franka Panda robot equipped with a sensor D435 during the reconstruction of a backpack. Bottom-left: voxelized reconstruction. Bottom-right: topology of the reconstruction.

26 proposed industrial methods decide to use high-precision and expensive visual sensing setups (Son et al.  
27 (2015), Rohrbach et al. (2016) and Zhang et al. (2017)), reducing its applicability in all types of industries.  
28 Therefore, we propose to use a new generation consumer depth camera (such as the Intel RealSense D435)  
29 installed on the robot, so that they can output live half-HD depth maps at high frequency rates with a low  
30 price for implementing a precise reconstruction of the objects to be manipulated (Fig. 1).

31 Real-time dense reconstruction presents important challenges when non-delay performance and fine-  
32 quality results are required. In particular, the incremental integration of overlapping depth maps into dense  
33 volumetric grids is not affordable for sequential methods. Thus, this problem has been addressed by many  
34 works employing different types of data structures accelerated by General Purpose Graphic Processor Units  
35 (GPGPU). The most successful methods in the context of hierarchical volumetric grid surface representation  
36 are based on Octree data structures, such as the work proposed by Hornung et al. (2013) for robotic collision  
37 avoidance tasks. Nevertheless, the main problem with this space representation is its low branching-rate  
38 that makes trees considerable deep at low-quality reconstructions. Other approaches more popular in

39 computer graphics are based on N-trees (Chen et al. (2013)) or B-trees (Johnson and Sasha (2002)). A less  
40 known data structure in computer graphics, but quite popular in data science are the B+ trees. These trees  
41 split the topology representation from the stored data (Museth (2013)). The works presented by Hoetzlein  
42 (2016) and Wu et al. (2018) are not mere implementations of the VDB (Variational Dynamic B+ trees) data  
43 structure for graphics hardware but they include a major change: data consistence is kept by using an apron  
44 voxels wrap with the neighbor voxels in contrast to use a neighbor index list. In fact, the use of bitmask is  
45 not necessary any more for discovering child nodes.

46 Implicit volumetric approaches in active sensing have demonstrated fine-quality results, starting with  
47 the method by Curless and Levoy (1996), which presents, for the first time, the use of a truncated signed  
48 distance field (TSDF). TSDF can also be used at real-time rates (Izadi et al. (2011) and Newcombe et al.  
49 (2011)) but a well-known problem of these methods is the lack of memory management. This led to use  
50 this approach just in reduced spaces with modest resolution. To overcome this problem, moving volume  
51 variants have been developed (Roth and Marselette (2012)). However, the problem is shifted to streaming  
52 out-of-core the data while the sensor moves. A more attractive approach is presented by Nießner et al.  
53 (2013), which uses a Hash table to compact the volume grid. However, careful consideration reveals several  
54 performance issues according to Museth (2013). Finally, Chen et al. (2013) presents hierarchical data  
55 structures that subdivide space more effectively, but they cannot be parallelized efficiently due to their  
56 additional computational complexity.

57 A real-time dense and dynamic 3D reconstruction method implementation, typically used in data science  
58 and computer graphics, is proposed to be used in robotics tasks in order to provide fine-quality results  
59 in a hierarchical topology. This new approach has the benefits of dense volumetric grid methods and  
60 the multi-level topology representation of hierarchical data structures, but it does not require a memory  
61 constrained voxel grid. This method is based on VDB trees that compress space and allow a real-time  
62 integration of new depth images. Additionally, this data structure isolates the implicit surface topology  
63 from the data which is stored densely in cells (called bricks). Although this kind of high performance  
64 hierarchical technique has been proposed for a variety of image rendering, simulations, collision detection  
65 tasks (Yang et al. (2017)) and semantic segmentation (Dai et al. (2018) and Hou et al. (2019)); a new  
66 extension based on the continuous update of the underlying data is proposed for surface reconstruction in  
67 robotics manipulation tasks (Fig. 1). All parts of the proposed pipeline (sensor pose acquisition, depth map  
68 integration and surface rendering) are performed on GPU hardware and they are validated by interactive  
69 robotic reconstructions of several scenes.

## 2 TERMINOLOGY OF VDB TREES

70 The proposed method is based on the VDB tree structure to represent a reconstructed scenario in a  
71 volumetric grid. VDB exploits spatial coherency of time-varying data to separately encode data values and  
72 grid topology (Fig. 2). There is no topology restrictions on the sparsity of the volumetric grid and it has a  
73 fast random access pattern  $O(1)$ . In fact, VDB models a virtually infinite 3D index space that allows for  
74 cache-coherent and fast data access into sparse volumes of high resolution. The VDB data structure is  
75 fundamentally hierarchical, facilitating adaptive grid sampling.

76 VDB dynamically arranges nodes in a hierarchical data structure, normally a tree (being the grid topology,  
77 Fig 2 left), where bricks are leaf nodes at the same fixed depth of an acyclic and connected graph with  
78 large but variable branching factors. This makes the tree being height-balanced but shallow and wide. This  
79 reduces the tree depth and the number of operations to traverse it from the root node to brick level. B+

80 tree is the type used by VDB, which has a variable number of children per node and it can be seen like a  
81 traditional B-tree where each leaf contain data keys (index).

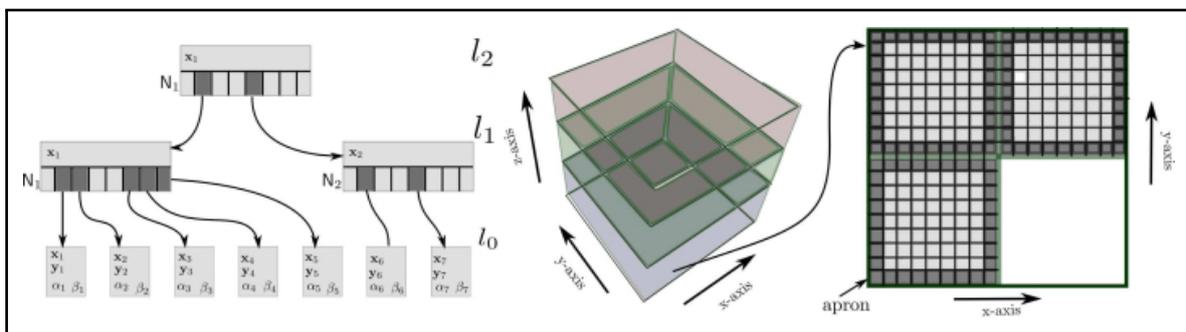
82 The proposed implementation of VDB in Museth (2013) uses a direct access bit mask to guarantee a fast  
83 and compact direct access to a binary representation of the local topology of a particular node. In contrast,  
84 we use the approach presented by Wu et al. (2018), where a pre-reserved and unsorted memory scheme bit  
85 masking is not necessary. This unmasked node access provides a better computational performance since  
86 the resorting of the node list is avoided.

87 As mentioned before, data values (or voxels) are stored separately from the topology (Fig. 2, center and  
88 right). The proposed storage scheme presented by Hoetzlein (2016) is used in order to stack the voxels in a  
89 3D heap (atlas), packing them inside bricks. The atlas is allocated in a GPU 3D texture to efficiently access  
90 to the data. The atlas is resized in z-axis if there is no more empty space in the current atlas. Each brick in  
91 the atlas keeps an apron of the nearest neighbor voxels wrapping them. Therefore, vicinity consistence in  
92 the data layout is kept.

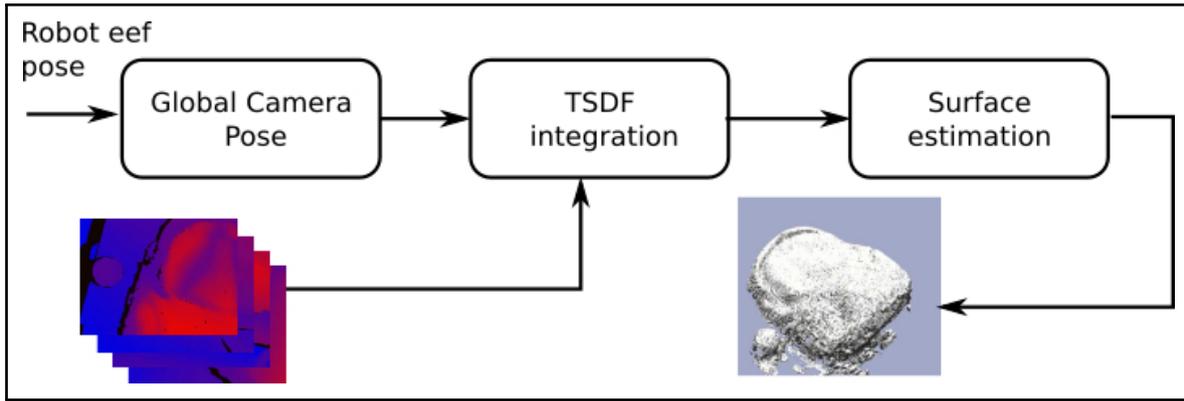
93 Although theoretically this scheme of reconstruction is unbounded in the 3D index space  $\mathbf{x} \equiv (x, y, z)$ ,  
94 this is naturally limited to bit-precision and memory constrains. The data encoded in each node consist  
95 of (Fig. 2): an index  $\mathbf{x}$  to address the node in a discrete pose inside the volumetric grid  $V$ ; an index  $\mathbf{y}$  to  
96 map the node with its correspondent brick  $B$  in atlas space  $A$ ; two flags  $\alpha$  and  $\beta$  which provide information  
97 about its activation and visibility; and a list pointing to its children nodes  $N$  in next level. The data value  
98 contained inside each voxel  $\{\bar{d}, \bar{w}\} \in B$  represents the truncated signed distance field TSDF and the weight.  
99 These values are computed by the integration of consecutive depth images  $D$ . Since the proposed method is  
100 formulated for robotic manipulation, every new  $D$  is transformed into the robot base frame by  ${}^bM_c$ .

### 3 PROPOSED METHOD

101 As previously stated, the developed method (Fig. 3) is devoted to resolve the reconstruction of dense and  
102 dynamic scenarios for robot manipulation tasks. Therefore, a constant and accurate camera pose information  
103 retrieval is assumed by the robot direct kinematic solver. This fact makes the method independent of camera  
104 pose estimation strategies like in Zeng et al. (2013), Puri et al. (2017), Newcombe et al. (2011), Nießner  
105 et al. (2013) and Nguyen et al. (2012). The main reason not to use camera pose estimation is to avoid



**Figure 2.** Representation of Variational Dynamics B+ trees adapted to GPU architecture. Left image represents a tree which defines the implicit topology of VDB (for simplification in 2D space), with the following configuration:  $2^2, 2^2, 2^2$ . Therefore, each node of the internal  $l_1$  and root  $l_2$  levels has a child list  $N$  of size 16. Nodes in the leaf level  $l_0$  have an index pointing to the atlas space  $\mathbf{y}$  in addition to volumetric index  $\mathbf{x}$ . Atlas is represented in the center of the image as heap. Right image shows one slice of the atlas space. Apron voxels are used to keep vicinity consistence. Only  $pool_0$  is shown in this figure.



**Figure 3.** Overview of the proposed 3D dense reconstruction pipeline.

106 drifting problems in textureless scenes due to bad error minimization in the Iterative Closest Point (ICP)  
 107 algorithm (Besl and McKay (1992) and Zhang (1994)).

108 Therefore, the current global camera pose is obtained by transforming the local camera pose  ${}^eM_c$  with  
 109 respect to the current robot end-effector pose:  ${}^bM_e = {}^bM_e \times {}^eM_c$ . The local pose  ${}^eM_c$  is estimated using  
 110 virtual visual servoing (VVS), as in Marchand and Chaumette (2002).

---

#### Algorithm 1: Topology Manipulation

---

**Result:** Topology and nodes set:  $\mathcal{T} \wedge \mathcal{N}$

```

1 initialization  $\mathcal{T} \leftarrow \{0\}$ ;
2 initialization  $\mathcal{N} \leftarrow \{0\}$ ;
3 initialization  $\hat{\mathcal{N}} \leftarrow \{0\}$ ;
4 while sensor is ON do
5    $\mathcal{D}_t \leftarrow \text{read\_depth\_image}$ ;
6    $\mathcal{N}_t \leftarrow \text{extract\_normals}$ ;
7   foreach  $k \leftarrow 1, |\mathcal{D}_t|$  do
8      $p \leftarrow \text{DDA}({}^eM_c, \mathcal{D}_t(k))$ ;
9     for  $l \leftarrow 1, L$  do
10       $idx_l \leftarrow \text{gen\_index}(p, l)$ ;
11      if  $\nexists idx_l$  then
12         $\hat{\mathcal{N}}(l, k) \leftarrow idx_l$ ;
13      end
14    end
15  end
16   $\text{radix\_sort}(\hat{\mathcal{N}})$ ;
17   $\mathcal{N} \leftarrow \text{reject\_duplicates}(\hat{\mathcal{N}})$ ;
18  for  $l \leftarrow L, 2$  do
19     $\mathcal{T}(l, l-1) \leftarrow \text{parenty}(\mathcal{N}(l), \mathcal{N}(l-1))$ ;
20  end
21  push  $\mathcal{T} \wedge \mathcal{N}$ ;
22 end

```

---

#### 111 Update of Topology and Atlas Spaces

112 The volumetric grid topology is updated before the integration of each new depth image. Thereby, new  
 113 nodes are added to those space quanta which fall inside the footprint of a depth sample  $z \in D$  of the

114 truncated region. The  $z$  are processed in parallel, activating new nodes in the topology and allocating bricks,  
 115 in atlas space, within the truncation region around the observed surface. Similarly to Nießner et al. (2013),  
 116 the truncation region is adapted based on the variance of the depth measurements in order to compensate  
 117 large uncertainties.

118 To update the topology, an indexes list of new nodes is created by ray-tracing scanning of  $V$  at all tree  
 119 levels. Note (algorithm 1) that the the topology  $\mathcal{T}$  and nodes set  $\mathcal{N}$  is an empty structure at the initialization.  
 120 This scan is also used to update the visibility of those nodes which are already active ( $\alpha = 1$ ) in the  
 121 topology. Secondly, those nodes belonging to the indexes list created by the ray-tracing are allocated.  
 122 Thirdly, every new node is linked with its parent in top-down direction.

123 A common chosen method to implement the ray-tracer is the Digital Differential Analyzer algorithm  
 124 (DDA, by Amanatides and Woo (1987)) because it interpolates values over an interval between start and  
 125 end points. This work defines this interval (i.e the ray bounding region) according to the root node range,  
 126 in contrast to Nießner et al. (2013) where rays were bounded to the truncation region. This strategy is used  
 127 to update all visibility information in the current frustum region (Fig. 4). The gradient value  $\nabla x$  used to  
 128 traverse each ray at level  $l$  is equal to the resolution at level  $l - 1$ . This is exemplified in the algorithm 1,  
 129 from line 8 to line 15. This foreach instruction is executed in a parallel fashion to compute the nodes which  
 130 holds the depth values  $D_t(k)$  measured by the sensor  ${}^eM_c$  at time  $t$ . Note, that the this nodes are computed  
 131 for each level  $l$  used to represent the tree.

### 132 Depth Image Integration

133 Depth images are integrated inside of the current volumetric grid: within the bricks whose node position  
 134 fall inside the camera view frustum and are not occluded (Fig. 4). Camera view frustum is defined by the  
 135 near and far clipping distances. This option keeps a constant computational cost of TSDF integration. Thus,  
 136 the method performance depends just on the size of the view range and not the density of the reconstruction.  
 137 In contrast to other works like Nießner et al. (2013), a brick selection strategy is not required since  $\alpha$  and  
 138  $\beta$  variables are directly consulted. Thus, all atlas bricks whose node position is inside camera range and  
 139 visible ( $\beta = 1$ ) are evaluated to implicitly update the volumetric grid.

---

#### Algorithm 2: Truncated signed distance field integration

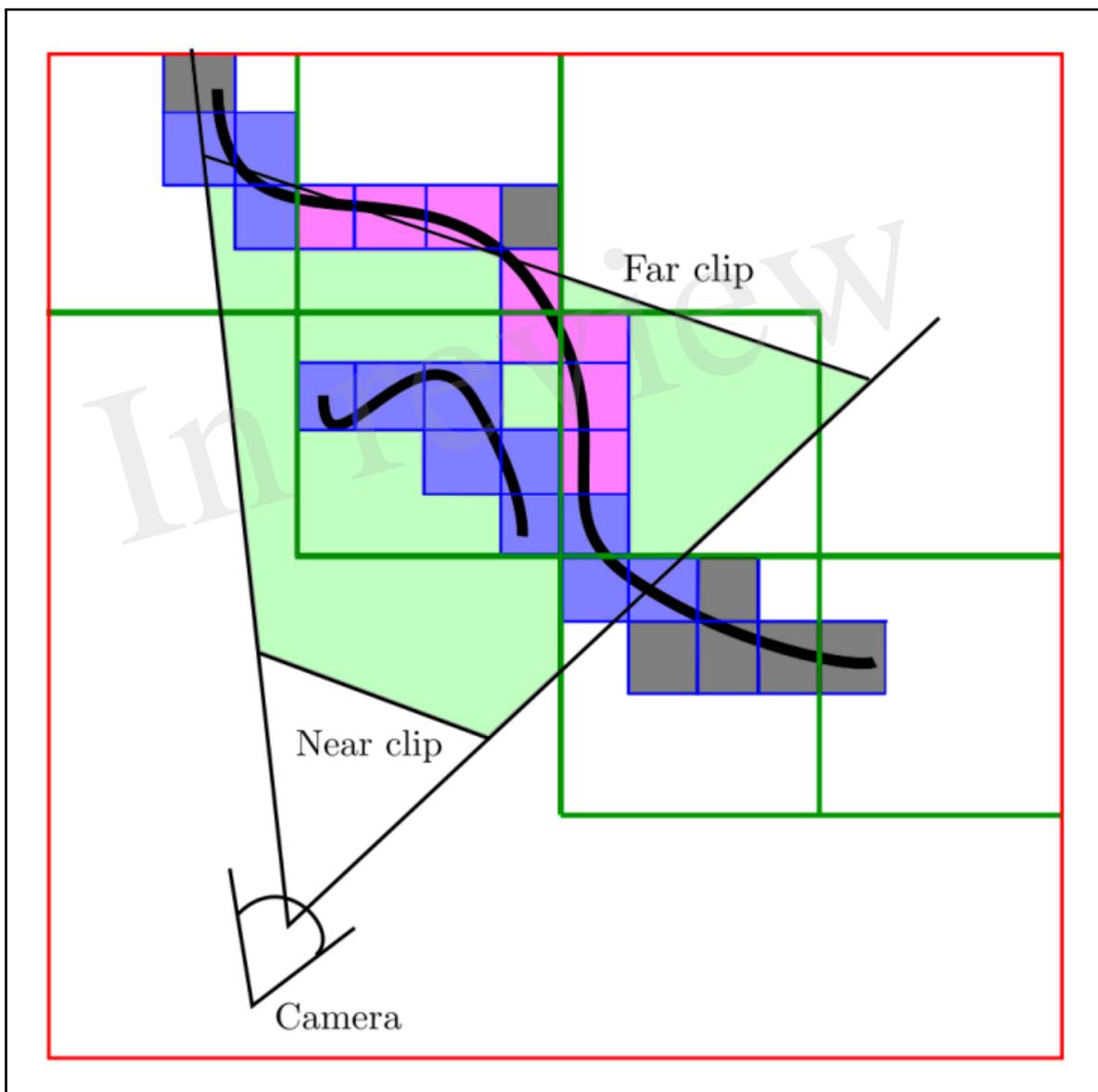
---

**Result:** TSDF:  $\bar{d} \wedge \bar{w}$   
 1 initialization  $\bar{d} \leftarrow \{0\}$  ;  
 2 initialization  $\bar{w} \leftarrow \{0\}$  ;  
 3 **while** sensor is ON **do**  
 4      $\mathcal{D}_t \leftarrow \text{read\_depth\_image}$ ;  
 5      $\mathcal{N}_t \leftarrow \text{extract\_normals}$ ;  
 6     **foreach**  $k \leftarrow 1, |\mathcal{D}_t|$  **do**  
 7          $x \leftarrow \text{get\_closest\_point\_in\_T}({}^eM_c, D_t(k))$ ;  
 8          $d \leftarrow \text{project\_z}({}^eM_c, x)$ ;  
 9          $w \leftarrow n_t \oplus n_{t+1}$ ;  
 10         apply equations (1)  $\wedge$  (2);  
 11     **end**  
 12 **end**

---

140 Integrating a new depth image involves the update of the bricks by re-computating the associated TSDFs  
 141 and weights (Curless and Levoy (1996)). The calculation of the TSDF is presented in Fig. 5 for the  
 142 special case of 1D. The sensor is positioned at the origin looking down the  $z$ -axis direction and takes two

143 measurements ( $z_1$  and  $z_2$ ) in two different time stamps. The signed distance field ( $d_1(\mathbf{x})$  and  $d_2(\mathbf{x})$ ) may  
 144 extend indefinitely in either direction, but the weight functions ( $w_1(\mathbf{x})$  and  $w_2(\mathbf{x})$ ) bound them behind the  
 145 range points. Concretely, the weight function  $w$  shown in (line 9) of algorithm 2, represents the similarity  
 146 function based on angular differences between current normal and the integrated one. Thus  $\oplus$  is defined as  
 147 the dot product of  $n_t \cdot n_{t+1}$ . This implies, that integration of new depth measurements are weight according  
 148 with the embedded shape. The weighted combination of the two profiles (Eq. 2) is illustrated in Fig. 5 in  
 149 purple. The integral combination rules are as follows:

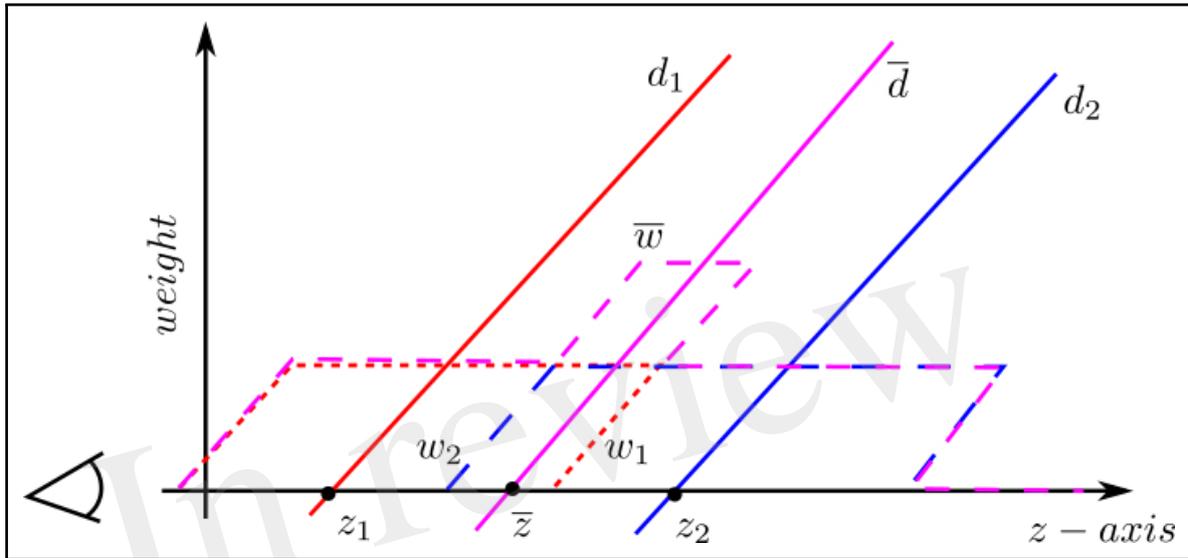


**Figure 4.** A 2D representation of how nodes are labeled according to the DDA algorithm. In this representation, a hierarchy with  $2^2$  nodes for levels  $l_1$  and  $l_2$  is defined. Leaf nodes are not explored. Camera frustum is defined by near and far clips. Nodes at  $l_0$  with gray color are outside of the view frustum, purple nodes are active but not visible and the blue ones are visible and active nodes. Neither gray nor purple nodes will be integrated.

$$\bar{d}_{t+1}(\mathbf{x}) = \frac{\bar{w}_t(\mathbf{x})\bar{d}_t(\mathbf{x}) + w_{t+1}(\mathbf{x})d_{t+1}(\mathbf{x})}{\bar{w}_t(\mathbf{x}) + w_{t+1}(\mathbf{x})}, \quad (1)$$

$$\bar{w}_{t+1}(\mathbf{x}) = \bar{w}_t(\mathbf{x}) + w_{t+1}(\mathbf{x}), \quad (2)$$

150 where,  $d_t(\mathbf{x})$  and  $w_t(\mathbf{x})$  are the signed distance and weight functions from the  $t$ -th range image.  $\bar{d}_t(\mathbf{x})$   
 151 and  $\bar{w}_t(\mathbf{x})$  are the cumulative signed distance and weight functions after integrating the  $t$ -th range image.



**Figure 5.** Computation of the TSDF (Truncated Signed Distance Field) in one-dimensional space. This figure shows two different measures  $z_1$  and  $z_2$  of the same surface spot at different times, along  $z$ -axis in the camera frame. Solid lines show distance fields  $d_1$  and  $d_2$  and dash lines represent weights  $w_1$  and  $w_2$ . Purple lines represent integral distance  $\bar{d}$  and weight  $\bar{w}$ . The surface position  $\bar{z}$  is obtained from this integral distance.

152 Note the importance of updating all bricks that fall into the current frustum, irrespective of whether they  
 153 reside in the current truncation region. This is done to prevent the integration of bricks which have been  
 154 added due to surface changes or outliers in the depth map and are no longer observed.

### 155 Node Rejection and Surface Generation

156 This step removes voxel blocks allocated due to noisy outliers and moved surfaces. Node rejection  
 157 operates on the updated atlas layout to mark a node as rejected and topology layout to remove the nodes.  
 158 For each brick, a summarization step is performed to obtain both the minimum absolute  $\bar{d}$  value and  
 159 the maximum  $\bar{w}$ . If the maximum  $\bar{w}$  of a brick is zero or the minimum  $\bar{d}$  is bigger than a threshold, the  
 160 associated brick is flagged for deletion. In a second pass, in parallel, all flagged leaves are deleted from  
 161 the topology. When all deletion operations are successfully done, all nodes in the rest of tree levels  $l \neq 0$   
 162 are unlinked following a down-top pattern. Once both layouts (topology and atlas) have been updated, all  
 163 nodes are set as non-visible.

164 Most previous works on dense volumetric reconstruction (such as Nguyen et al. (2012)) extract the  
 165 implicit iso-surface before rendering the underlying surface. In contrast, the proposed method generates

	Normal est.	Integration	Surface generation	Topology update
Shoe	4.3	15.8	19.5	48.1
Tape	4.1	14.9	18.1	46.0
Alum piece	4.9	13.5	15.4	49.8
Backpack	5.2	18.4	21.6	54.3
Non-Static Obj.	5.7	19.1	23.7	62.8

**Table 1.** Algorithm's profile during experimentation. The table shows the average (per frame) time in milliseconds taken by the most relevant stages during the reconstruction.

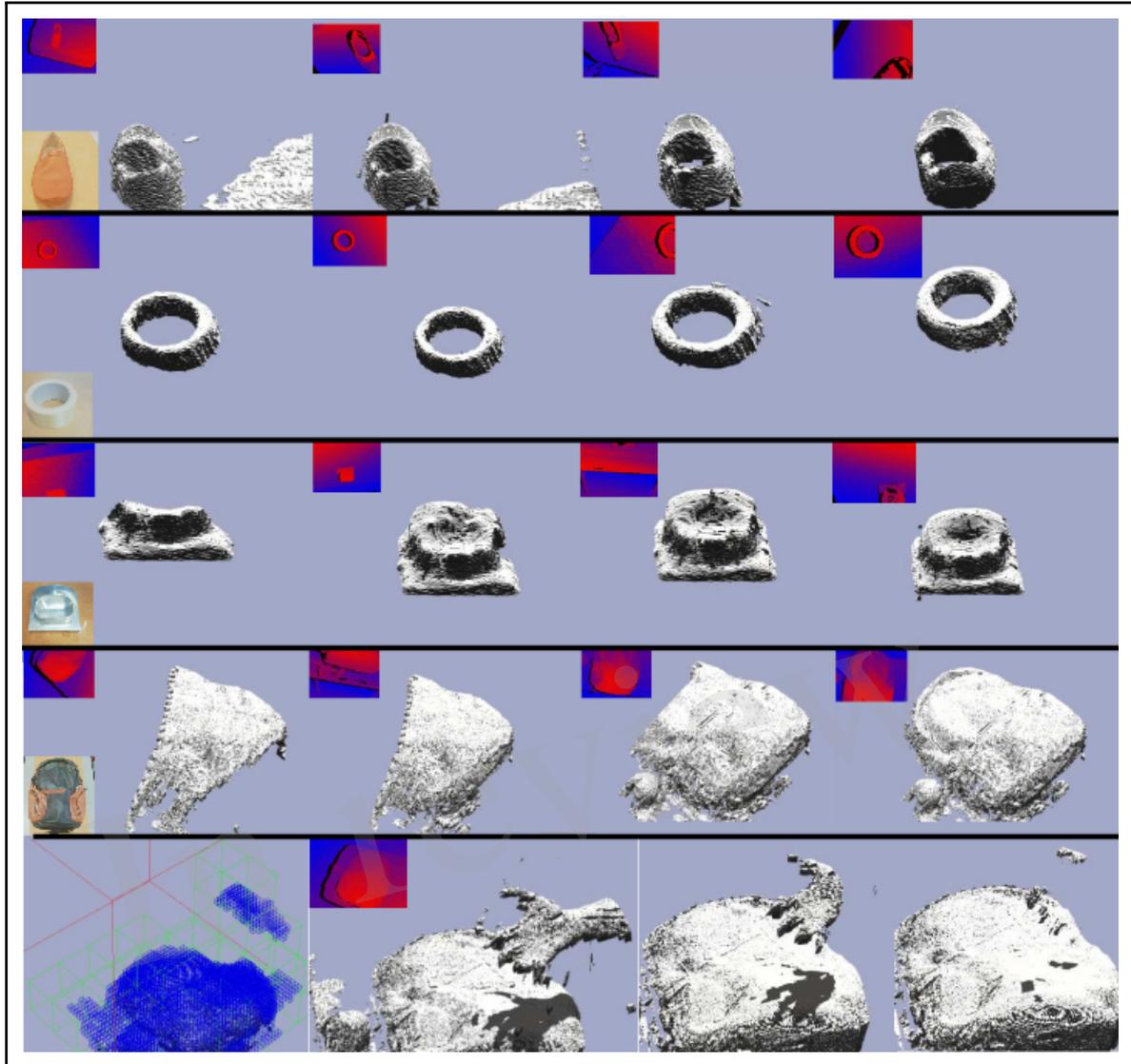
166 the render image of the reconstructed surface directly from the volumetric grid, like in Chen et al. (2013).  
 167 In order to compute the normal surface, needed for shading, the gradient of the TSDF at the zero-crossing  
 168 is estimated by using first order finite differences and trilinear interpolation. The vast majority of samples  
 169 lie in the same leaf grid due to the use of a shallow tree with relatively large branching factors.

## 4 RESULTS

170 All the experiments are executed in a laptop PC equipped with an Intel Core i7-6820HQ CPU at 2.70GHz,  
 171 32GB of RAM and an embedded Quadro M2000M GPU. The robot platform is composed of a Robot  
 172 Franka Panda equipped with a RGBD camera Intel RealSense D435. Four sequences are captured with this  
 173 set up in order to evaluate the proposed method: a shoe, an adhesive tape, a small aluminum piece and a  
 174 backpack (Fig. 6). All the experiments are performed on top of a table situated in  $z = 0$  with respect to  
 175 robot base. Shoe experiment is the middle size one,  $0.3 \times 0.12 \times 0.8$ m, of brown leather. Tap experiment  
 176 is the thin hoop of size  $0.1 \times 0.1 \times 0.08$ m. Small aluminum piece experiment is used to show how this  
 177 method can deal with noisy information (measurements corrupted because the material of the object), the  
 178 size of the object is  $0.07 \times 0.07 \times 0.06$ m. Backpack experiment is composed by two objects an apple and a  
 179 backpack of  $0.47 \times 0.33 \times 0.18$ m. The fourth experiment is extended by adding a non static object (e.g. a  
 180 human hand) in the scene. The topology configuration for all experiments is the same and it includes for  
 181 each axis direction:  $2^3$  nodes at root level;  $2^3$  nodes at internal level; and  $2^4$  nodes at leaves level. The voxel  
 182 resolution is set to  $1mm^3$ . The camera pose in all sequences follow the same trajectory (Fig. 7), performed  
 183 by the robot.

184 Fig. 6 presents the reconstruction evolution of all four scenes used in the evaluation. Note that for the  
 185 surface visualization, the rendering voxels strategy is shown because this representation fits better in  
 186 reconstructions aimed to measure tasks. Otherwise, the visualization would be misleading. To enrich the  
 187 voxel representation, internal voxels are also visualized. Since the scene is static with regard to the robot  
 188 base, the reconstruction is done just in those measures with positive z values.

189 It is remarkable that the final results in all four scenes have finished without drifting problems. Concretely,  
 190 in experiment one (shoe), the model evolves from a rough-quality to a fine-quality. The second experiment  
 191 (tape) is similar than the first one but with an additional difficulty: it is a tiny object, with just a  $60mm^2$   
 192 diameter and a  $3mm$  height. The third experiment (small aluminium piece) is again a tiny object but it is  
 193 made by aluminum, a reflective material. Last experiment is split in two rows because it presents a more  
 194 complex scenario. The sequence has two main parts: firstly, the backpack is reconstructed (fourth row) in a  
 195 first robot trajectory execution (i.e. pass), but later a non-static object (e.g. a human hand) appears in the  
 196 scenario (fifth row). Even with this occlusion, the previous reconstruction is not affected and it continues to  
 197 be done successfully in the next robot pass. The shadowing method is used to illustrate the occlusion. When  
 198 the camera is situated for the next pass, the hand goes away from the scene. While the camera does not pass

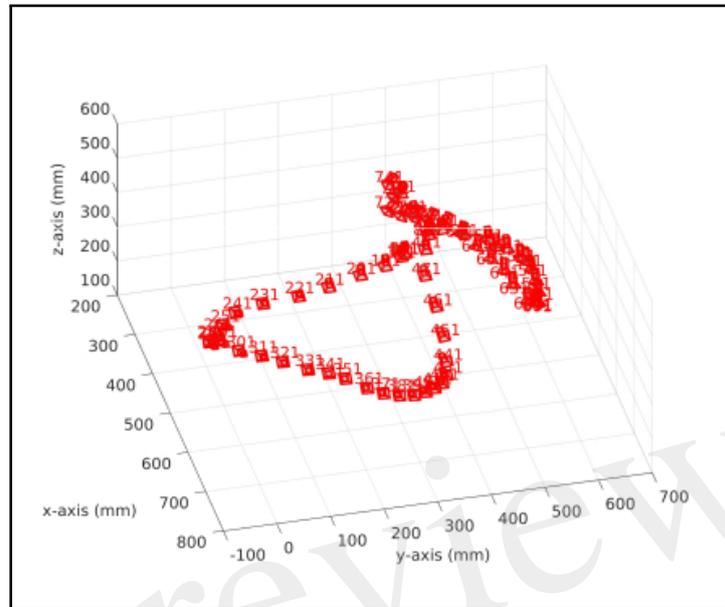


**Figure 6.** Snapshots of reconstructions experiments for 4 different objects: a shoe (1st row), a tape (2nd row), a small aluminium piece (3rd row) and a backpack (4th and 5th rows).

199 over the region where the hand was, the hand voxels stay. When the camera records once again that scene  
 200 region, the hand voxels vanish without affecting the backpack reconstruction. More precisely, after the  
 201 hand is removed from the scene for the first time, the leaf nodes used to code it inside the volumetric grid  
 202 stay a while. The observed voxels vanish before the nodes because TSDF values inside the voxels become  
 203 positive, breaking zero-crossing condition. Afterwards, all TSDF reach maximum distance, marking bricks  
 204 to be rejected.

205 Unlike other reconstruction methods, this work presents a study of the feasibility of a reconstruction  
 206 method based on the VDB data structure in robotic tasks (especially manipulation). Because the constrains  
 207 in this kind of task are mainly knowing the topology of the objects and real time response, in the following  
 208 we carry out the following study of computational times. The table 1 presents the time taken by four of  
 209 the most relevant parts of this method: Normal estimation; depth integration; surface generation; and  
 210 topology update. Time are the average taken for processing a frame. It is interesting to observe that  
 211 the normal estimation, integration and surface generation is quite constant. This is mainly because this

212 steps are processed in parallel, while normal estimation is computed in image space, the integration and  
213 surface generation is computed in atlas space. As drawback this method keeps updating the topology in a  
214 non-parallel fashion, fortunately this time tends to decrease linearly as the surface is captured, once the  
215 surface is captured the time consumption is negligible, no matter if the motion of the object. This indicates  
216 that most of the time is expended when the topology needs to branch.



**Figure 7.** Trajectory of the camera used during the experiments. We sample pose in order to take 1 each 10 poses.

## 5 CONCLUSIONS

217 A novel dense and dynamic 3D reconstruction method has been implemented based on a hierarchical  
218 database structure (GPU oriented) for integrating depth images by truncated signed distance field theory.  
219 A qualitative validation of the reconstruction of 4 different scenes with different properties (materials,  
220 size, occlusions...) is performed to show the performance of this method. Current results show that this  
221 method provide stable reconstruction in most of the situations. But, the method present a fast recovering of  
222 reconstructions in fail situation. Future directions in our research explore the use of this method to simulate  
223 material dynamic in situ, taken advance of the GPU optimized VDB data structure. This will allow keeping  
224 tracking non-rigid surfaces while the are being manipulated. Moreover, we will work in the design of active  
225 perception using as source data the volumetric grid, instead of use directly depth images or point clouds.

## CONFLICT OF INTEREST STATEMENT

226 The authors declare that the research was conducted in the absence of any commercial or financial  
227 relationships that could be construed as a potential conflict of interest.

## FUNDING

228 This project has received funding from the European Union's Horizon 2020 research and innovation  
229 programme under grant agreement n° 869855 (Project SoftManBot) and by Regional Funds for Auvergne  
230 (France).

## ACKNOWLEDGMENT

231 The authors would like to specially thank Emmanuel Duc (Manager of the Additive Manufacturing Chair  
232 at SIGMA Clermont) and Grigore Gogu (IP Institute Pascal) for their support and guidance.

## REFERENCES

- 233 Amanatides, J. and Woo, A. (1987). A Fast Voxel Traversal Algorithm for Ray Tracing. In *Eurographics*.  
234 vol. 87, 3–10. doi:<https://doi.org/10.2312/egtp.19871000>
- 235 [Dataset] Besl, P. and McKay, N. (1992). A Method for Registration of 3-D Shapes. doi:[10.1109/34.121791](https://doi.org/10.1109/34.121791)
- 236 Chen, J., Bautembach, D., and Izadi, S. (2013). Scalable real-time volumetric surface reconstruction. *ACM*  
237 *Trans. Graph.* 32, 1. doi:[10.1145/2461912.2461940](https://doi.org/10.1145/2461912.2461940)
- 238 Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images.  
239 In *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '96* (New York, New York,  
240 USA: ACM Press), 303–312. doi:[10.1145/237170.237269](https://doi.org/10.1145/237170.237269)
- 241 Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., and Nießner, M. (2018). ScanComplete: Large-Scale  
242 Scene Completion and Semantic Segmentation for 3D Scans. In *IEEE Conf. Comput. Vis. Pattern*  
243 *Recognit.* 4578–4587. doi:[10.1109/CVPR.2018.00481](https://doi.org/10.1109/CVPR.2018.00481)
- 244 Hoetzlein, R. K. (2016). GVDB: Raytracing Sparse Voxel Database Structures on the GPU. *High Perform.*  
245 *Graph.* doi:[10.2312/hpg.20161197](https://doi.org/10.2312/hpg.20161197)
- 246 Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient  
247 probabilistic 3D mapping framework based on octrees. *Auton. Robots* 34, 189–206. doi:[10.1007/](https://doi.org/10.1007/s10514-012-9321-0)  
248 [s10514-012-9321-0](https://doi.org/10.1007/s10514-012-9321-0)
- 249 Hou, J., Dai, A., and Nießner, M. (2019). 3d-sis: 3d semantic instance segmentation of rgb-d scans. In  
250 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4421–4430
- 251 Izadi, S., Kim, Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., et al. (2011). KinectFusion: real-time  
252 3D reconstruction and interaction using a moving depth camera. *Proc. 24th Annu. ACM User Interface*  
253 *Softw. Technol. Symp. - UIST '11*, 559–568doi:[10.1145/2047196.2047270](https://doi.org/10.1145/2047196.2047270)
- 254 Johnson, T. and Sasha, D. (2002). The performance of current B-tree algorithms. *ACM Trans. Database*  
255 *Syst.* 18, 51–101. doi:[10.1145/151284.151286](https://doi.org/10.1145/151284.151286)
- 256 Marchand, É. and Chaumette, F. (2002). Virtual visual servoing: A framework for real-time augmented  
257 reality. *Comput. Graph. Forum* 21, 289–297. doi:[10.1111/1467-8659.t01-1-00588](https://doi.org/10.1111/1467-8659.t01-1-00588)
- 258 Museth, K. (2013). VDB: high-resolution sparse volumes with dynamic topography. *ACM Trans. Graph.*  
259 32, 1–22. doi:[10.1145/2487228.2487235](https://doi.org/10.1145/2487228.2487235)
- 260 Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., et al. (2011).  
261 KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE Int. Symp. Mix.*  
262 *Augment. Reality, ISMAR 2011 (IEEE)*, 127–136. doi:[10.1109/ISMAR.2011.6092378](https://doi.org/10.1109/ISMAR.2011.6092378)
- 263 Nguyen, C. V., Izadi, S., and Lovell, D. (2012). Modeling kinect sensor noise for improved 3D  
264 reconstruction and tracking. *Proc. - 2nd Jt. 3DIM/3DPVT Conf. 3D Imaging, Model. Process. Vis.*  
265 *Transm. 3DIMPVT 2012*, 524–530doi:[10.1109/3DIMPVT.2012.84](https://doi.org/10.1109/3DIMPVT.2012.84)
- 266 Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3D reconstruction at scale  
267 using voxel hashing. *ACM Trans. Graph.* 32, 1–11. doi:[10.1145/2508363.2508374](https://doi.org/10.1145/2508363.2508374)
- 268 Puri, P., Jia, D., and Kaess, M. (2017). GravityFusion: Real-time dense mapping without pose graph  
269 using deformation and orientation. *IEEE Int. Conf. Intell. Robot. Syst.* 2017-Septe, 6506–6513.  
270 doi:[10.1109/IROS.2017.8206559](https://doi.org/10.1109/IROS.2017.8206559)
- 271 Rohrbach, A., Rohrbach, M., Hu, R., Darrell, T., and Schiele, B. (2016). Computer Vision – ECCV 2016.  
272 *1511.03745V1* 9905, 1–10. doi:[10.1007/978-3-319-46448-0](https://doi.org/10.1007/978-3-319-46448-0)

- 273 Roth, H. and Marsette, V. (2012). Moving Volume KinectFusion. *Proc. Br. Mach. Vis. Conf.* , 112.1—  
274 112.11doi:http://dx.doi.org/10.5244/C.26.112
- 275 Son, H., Kim, C., and Kim, C. (2015). 3D reconstruction of as-built industrial instrumentation models  
276 from laser-scan data and a 3D CAD database based on prior knowledge. *Autom. Constr.* 49, 193–200.  
277 doi:10.1016/j.autcon.2014.08.007
- 278 Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). ElasticFusion:  
279 Real-time dense SLAM and light source estimation. In *Int. J. Rob. Res.* 14, 1697–1716. doi:10.1177/  
280 02783649166669237
- 281 Wu, K., Truong, N., Yuksel, C., and Hoetzlein, R. (2018). Fast fluid simulations with sparse volumes on  
282 the GPU. *Comput. Graph. Forum* 37, 157–167. doi:10.1111/cgf.13350
- 283 Yang, Z., Gao, F., and Shen, S. (2017). Real-time monocular dense mapping on aerial robots using  
284 visual-inertial fusion. *Proc. - IEEE Int. Conf. Robot. Autom.* , 4552–4559doi:10.1109/ICRA.2017.  
285 7989529
- 286 Zeng, M., Zhao, F., Zheng, J., and Liu, X. (2013). Octree-based fusion for realtime 3D reconstruction.  
287 *Graph. Models* 75, 126–136. doi:10.1016/j.gmod.2012.09.002
- 288 Zhang, T., Liu, J., Liu, S., Tang, C., and Jin, P. (2017). A 3D reconstruction method for pipeline inspection  
289 based on multi-vision. *Meas. J. Int. Meas. Confed.* 98, 35–48. doi:10.1016/j.measurement.2016.11.004
- 290 Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput.*  
291 *Vis.* 13, 119–152. doi:10.1007/BF01427149

In review

Figure 1.JPEG

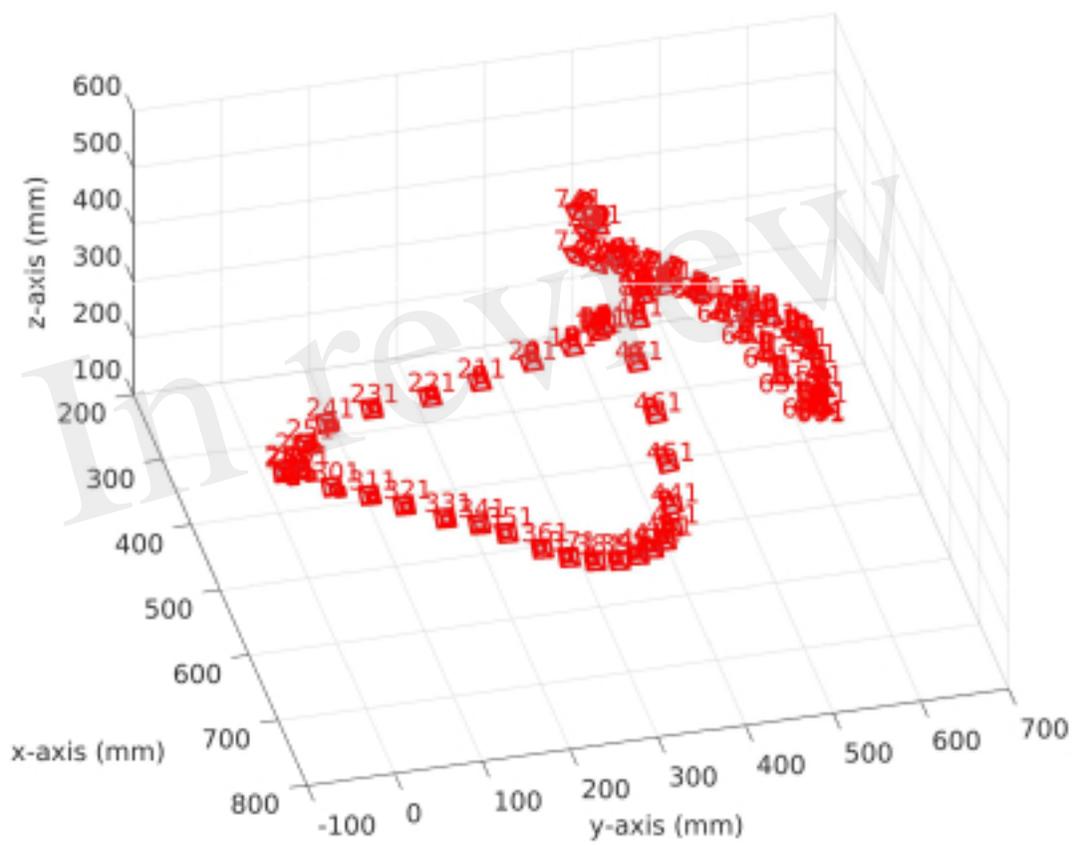


Figure 2.JPEG

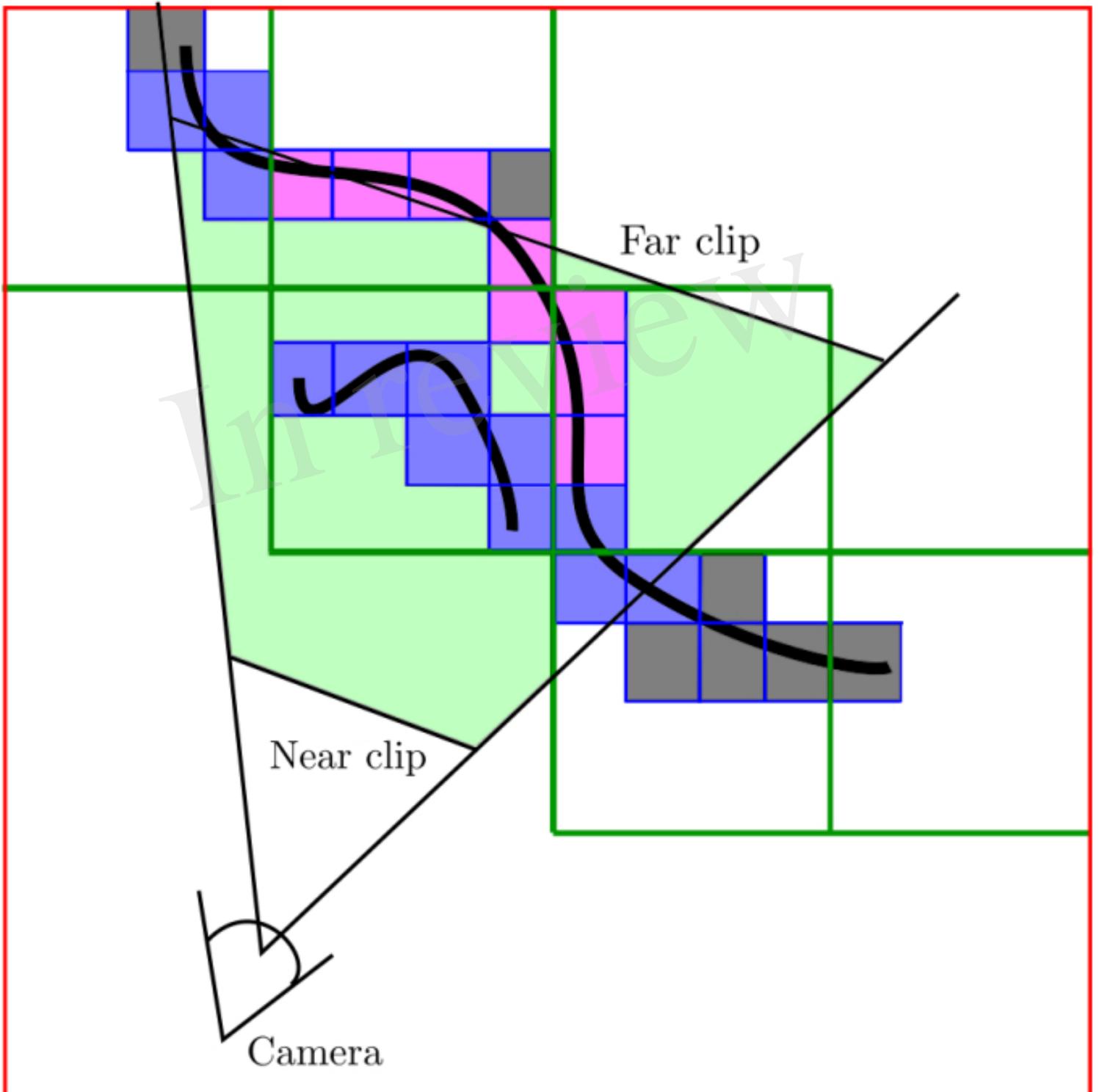


Figure 3.JPEG

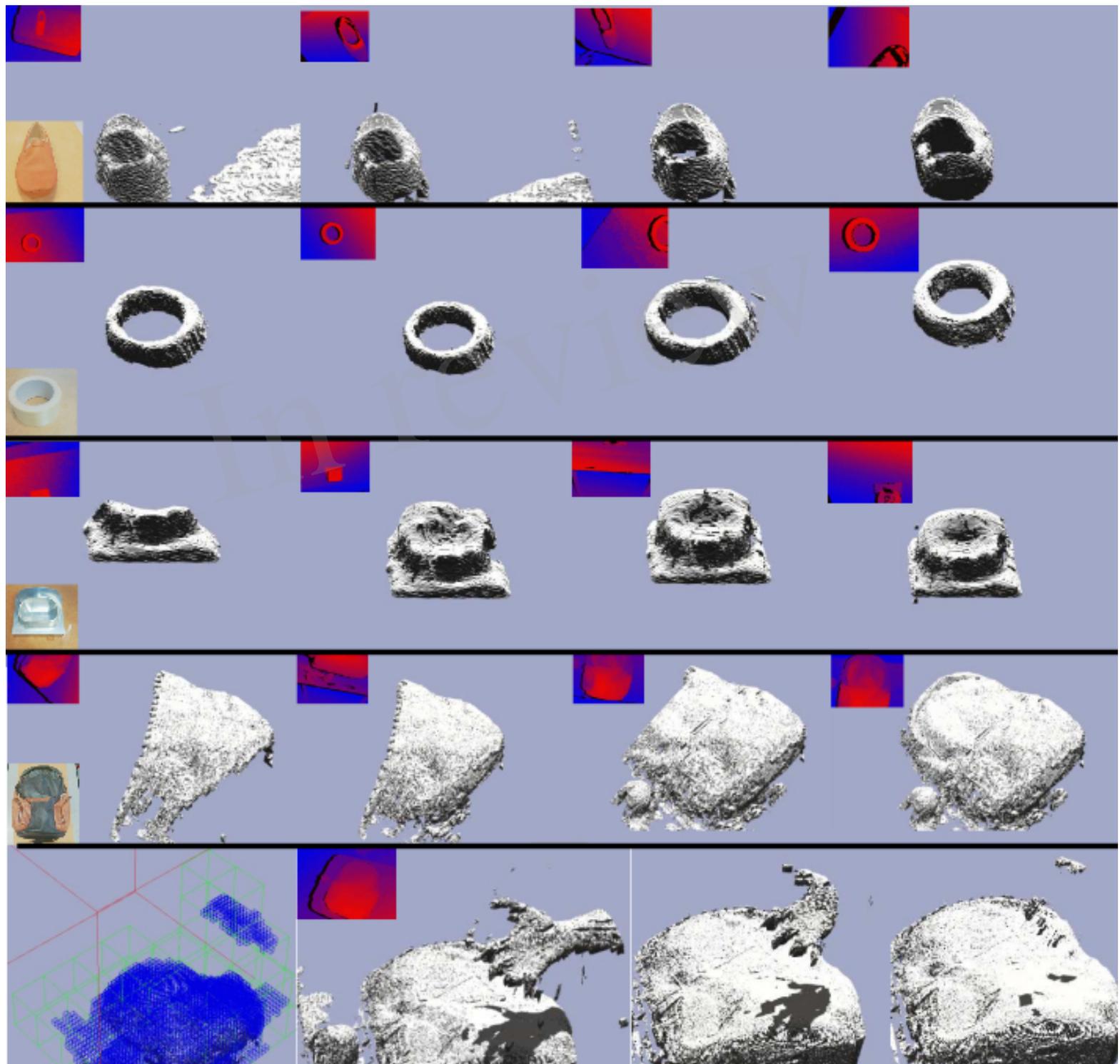


Figure 4.JPEG

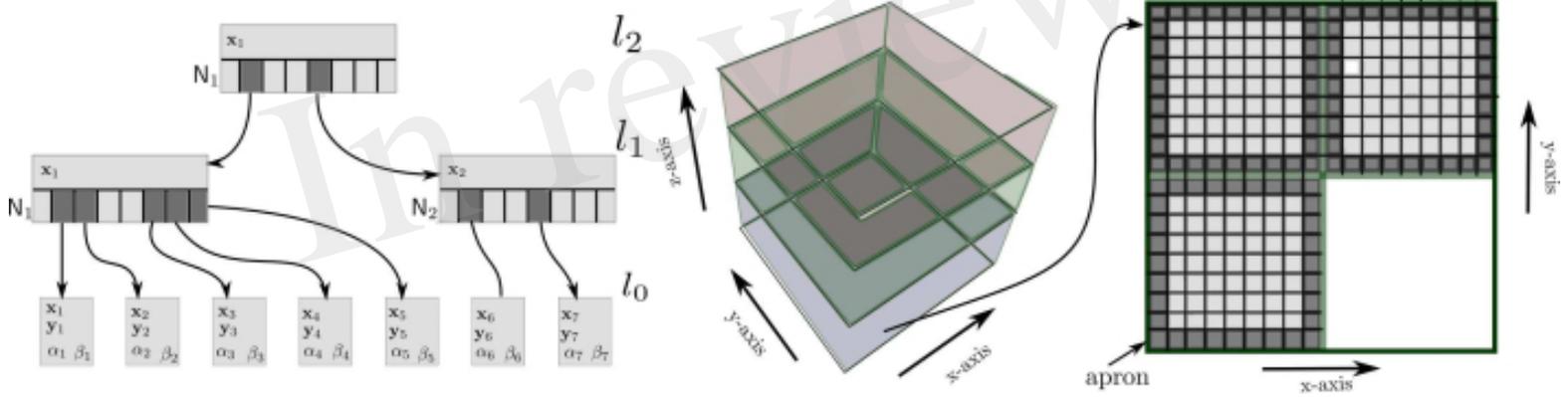


Figure 5.JPEG

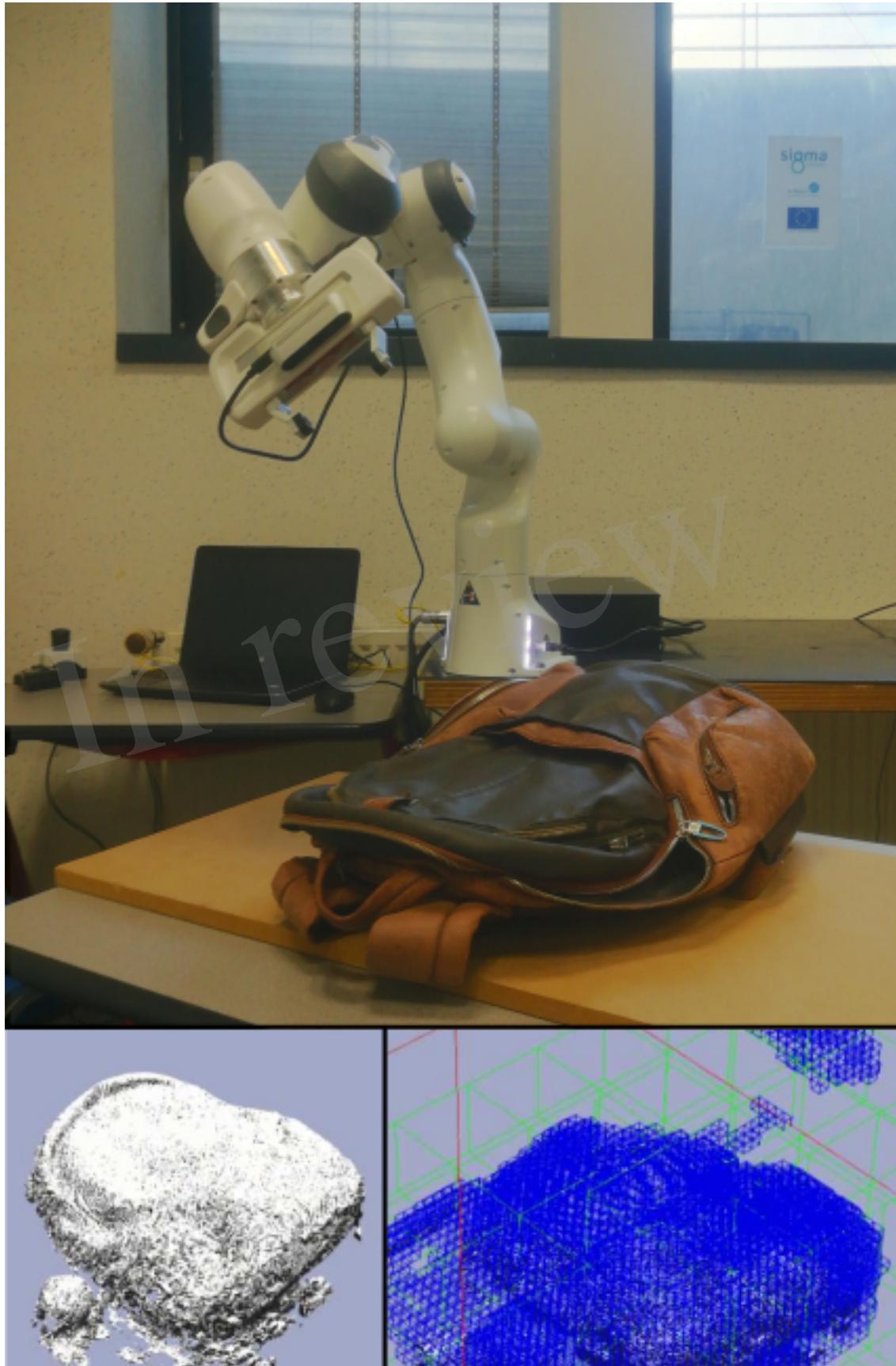


Figure 6.JPEG

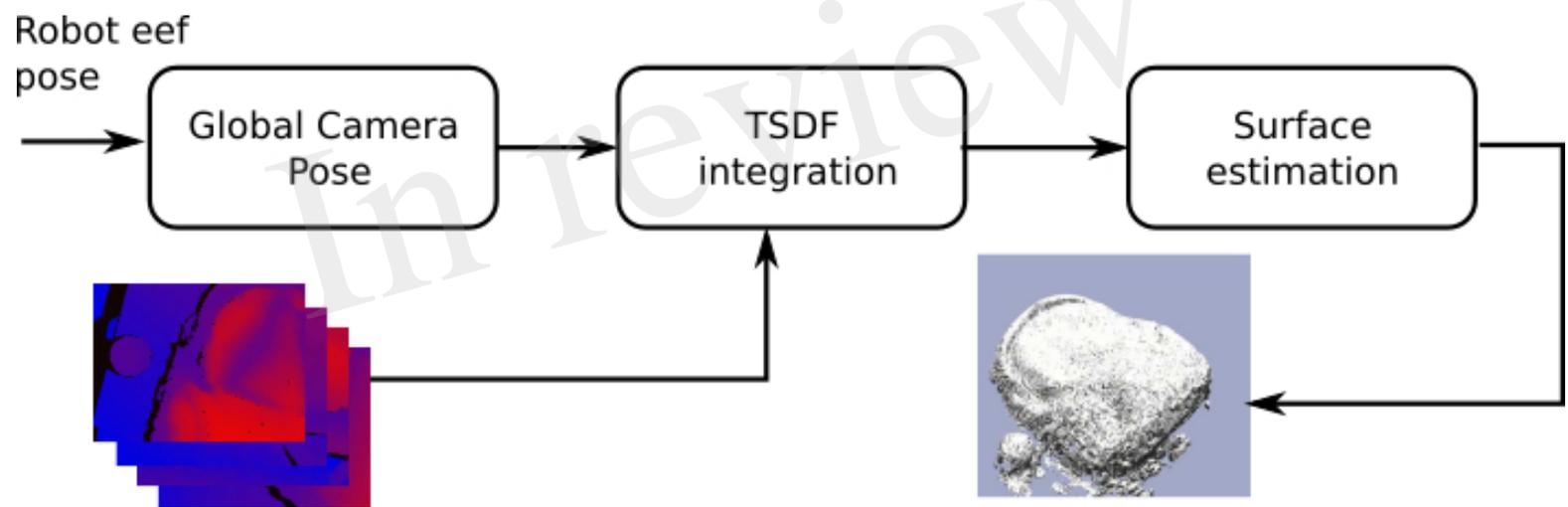


Figure 7.JPEG

