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

Carlos Mateo, Juan Antonio Corrales Ramon, Youcef Mezouar

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

Carlos M. MATEO AGULLÓ[1, 2*], Juan Antonio Corrales Ramón[1, 2], Youcef Mezouar[1, 2]

[1]Sigma Clermont, France, [2]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

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 perfomance computing

Abstract

Word count: 154

This paper presents a 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 stat-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.

frontiers

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

ABSTRACT

This paper presents a novel approach to implement hierarchical, dense and dynamic reconstruction of 3D objects based on the VDB (Variational Dynamic B+ Trees) data structure for robotic applications. The scene reconstruction is done by the integration of depth-images using the Truncated Signed Distance Field (TSDF). The proposed reconstruction method is based on dynamic trees in order to provide similar reconstruction results than current state-of-the-art methods (i.e. complete volumes, hashing voxels and hierarchical volumes) in terms of execution time, but with a direct multi-level representation that remains real-time. In fact, this representation provides two major advantages: it is a 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 of this approach in a robot arm platform.

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

1 INTRODUCTION

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

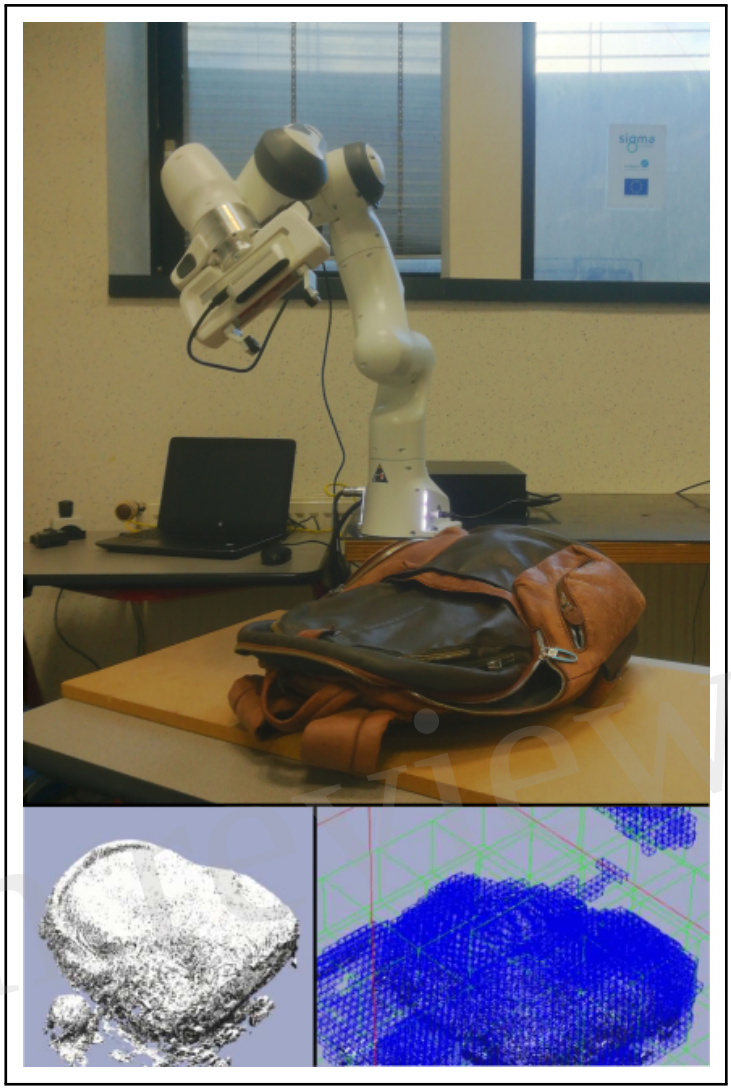

Figure 1. Robot platform used to evaluate our approach. Top: Franka Panda robot equipped with a sensor D435 during the reconstruction of a backpack. Bottom-left: voxelized reconstruction. Bottom-right: topology of the reconstruction.

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

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

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

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

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

2 TERMINOLOGY OF VDB TREES

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

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

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

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

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

Although theoretically this scheme of reconstruction is unbounded in the 3D index space $\mathbf{x} \equiv (x,\ y,\ z)$, this is naturally limited to bit-precision and memory constrains. The data encoded in each node consist 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 map the node with its correspondent brick B in atlas space A; two flags α and β which provide information about its activation and visibility; and a list pointing to its children nodes N in next level. The data value contained inside each voxel $\{\overline{d}, \overline{w}\} \in \mathsf{B}$ represents the truncated signed distance field TSDF and the weight. These values are computed by the integration of consecutive depth images D. Since the proposed method is formulated for robotic manipulation, every new D is transformed into the robot base frame by ${}^{b}\mathrm{M}_{c}$.

3 PROPOSED METHOD

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

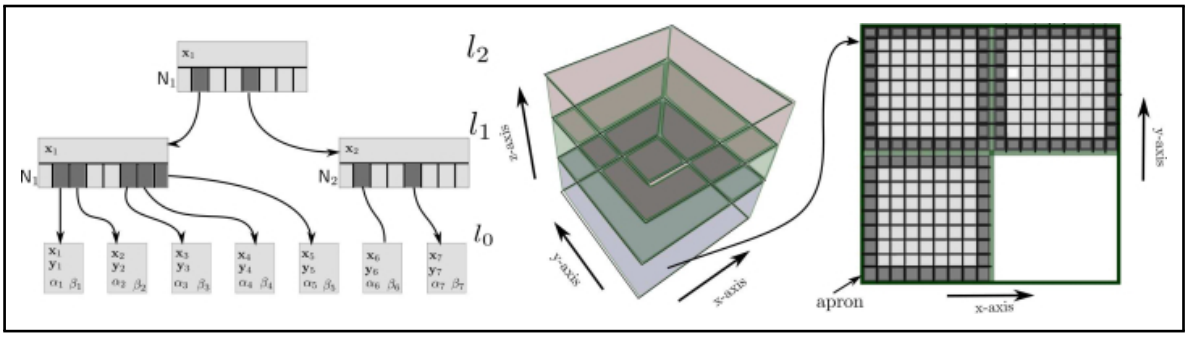

Figure 2. Representation of Variational Dynamics B+ trees adapted to GPU architecture. Left image represents a tree which defines the implicit topology of VDB (for simplification in 2D space), with the following configuration: 2^2, 2^2, 2^2. Therefore, each node of the internal l_1 and root l_2 levels 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.

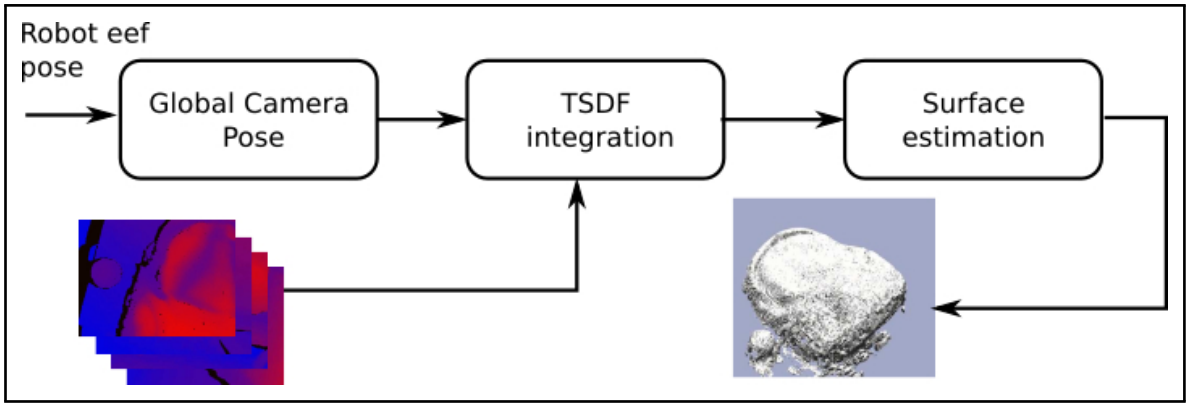

Figure 3. Overview of the proposed 3D dense reconstruction pipeline.

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

Therefore, the current global camera pose is obtained by transforming the local camera pose ${}^{e}\mathrm{M}_{c}$ with respect to the current robot end-effector pose: ${}^{b}\mathrm{M}_{e} = {}^{b}\mathrm{M}_{e} \times {}^{e}\mathrm{M}_{c}$. The local pose ${}^{e}\mathrm{M}_{c}$ is estimated using virtual visual servoing (VVS), as in Marchand and Chaumette (2002).

Algorithm 1: Topology Manipulation

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

initialization $\mathcal{T} \leftarrow \{0\}$;
initialization $\mathcal{N} \leftarrow \{0\}$;
initialization $\hat{\mathcal{N}} \leftarrow \{0\}$;
while *sensor is ON* **do**
  $\mathcal{D}_t \leftarrow read\_depth\_image$;
  $\mathcal{N}_t \leftarrow extract\_normals$;
  foreach $k \leftarrow 1, |\mathcal{D}_t|$ **do**
    $p \leftarrow DDA({}^{e}\mathrm{M}_{c}, D_t(k))$;
    for $l \leftarrow 1, L$ **do**
      $idx_l \leftarrow gen\_index(p, l)$;
      if $\nexists\, idx_l$ **then**
        $\hat{\mathcal{N}}(l, k) \leftarrow idx_l$;
      end
    end
  end
  $radix\_sort(\hat{\mathcal{N}})$;
  $\mathcal{N} \leftarrow reject\_duplicates(\hat{\mathcal{N}})$;
  for $l \leftarrow L, 2$ **do**
    $\mathcal{T}(l, l-1) \leftarrow parenty(\mathcal{N}(l), \mathcal{N}(l-1))$;
  end
  push $\mathcal{T} \wedge \mathcal{N}$;
end

Update of Topology and Atlas Spaces

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

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

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

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

Depth Image Integration

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

Algorithm 2: Truncated signed distance field integration

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

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

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

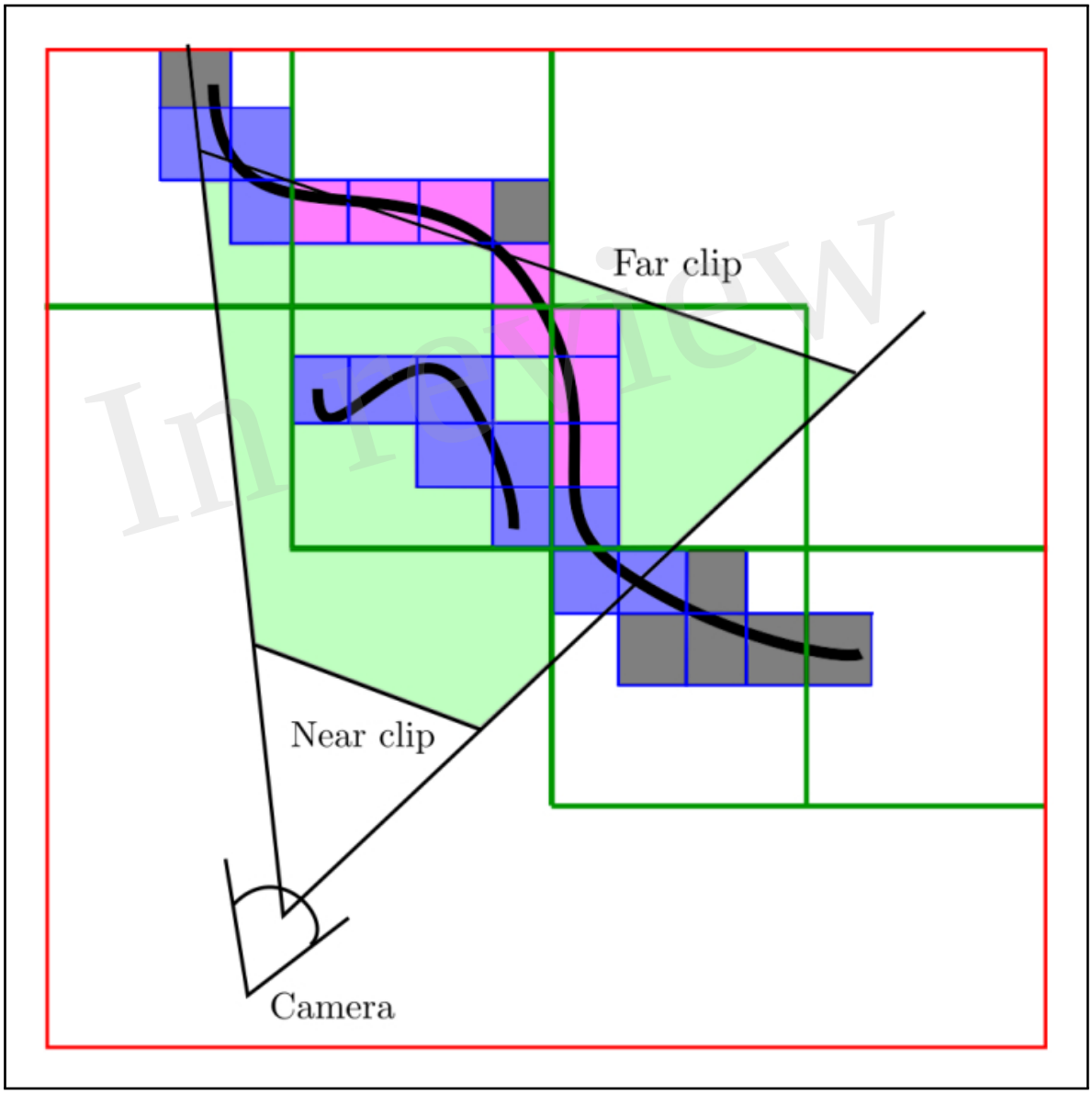

Figure 4. A 2D representation of how nodes are labeled according to the DDA algorithm. In this representation, a hierarchy with 2^2 nodes for levels l_1 and l_2 is defined. Leaf nodes are not explored. Camera frustum is defined by near and far clips. Nodes at l_0 with gray color are outside of the view 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.

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

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

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

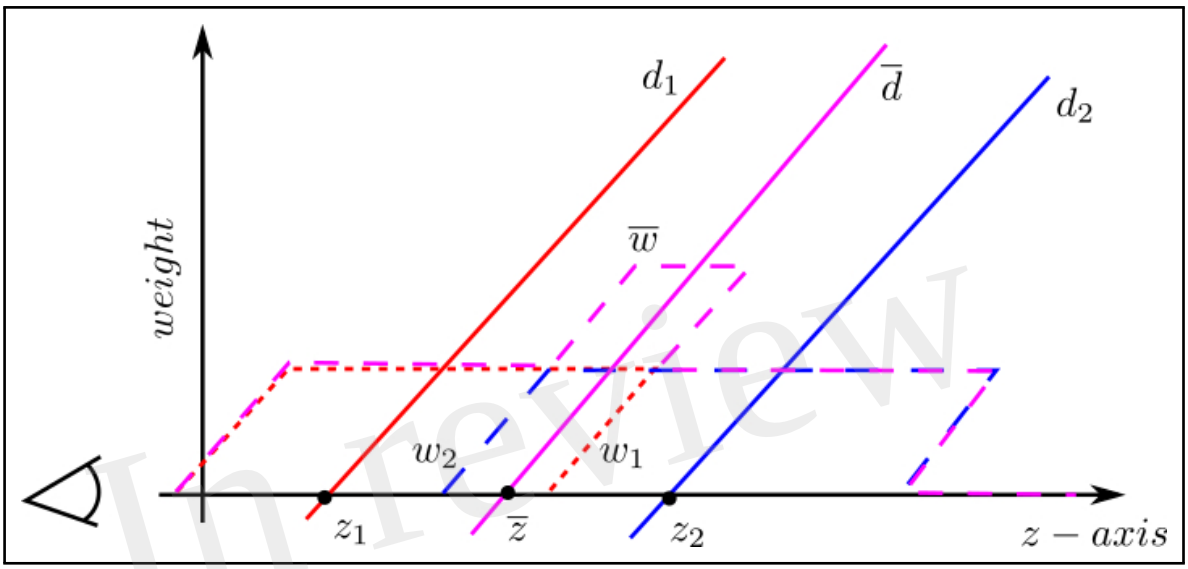

Figure 5. Computation of the TSDF (Truncated Signed Distance Field) in one-dimensional space. This figure shows two different measures z_1 and z_2 of the same surface spot at different times, along z-axis in the camera frame. Solid lines show distance fields d_1 and d_2 and dash lines represent weights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

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

Node Rejection and Surface Generation

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

Most previous works on dense volumetric reconstruction (such as Nguyen et al. (2012)) extract the 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.

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

4 RESULTS

All the experiments are executed in a laptop PC equipped with an Intel Core i7-6820HQ CPU at 2.70GHz, 32GB of RAM and an embedded Quadro M2000M GPU. The robot platform is composed of a Robot Franka Panda equipped with a RGBD camera Intel RealSense D435. Four sequences are captured with this set up in order to evaluate the proposed method: a shoe, an adhesive tape, a small aluminum piece and a backpack (Fig. 6). All the experiments are performed on top of a table situated in $z = 0$ with respect to robot base. Shoe experiment is the middle size one, $0.3 \times 0.12 \times 0.8$m, of brown leather. Tap experiment 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 method can deal with noisy information (measurements corrupted because the material of the object), the 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 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 human hand) in the scene. The topology configuration for all experiments is the same and it includes for each axis direction: 2^3 nodes at root level; 2^3 nodes at internal level; and 2^4 nodes at leaves level. The voxel resolution is set to $1mm^3$. The camera pose in all sequences follow the same trajectory (Fig. 7), performed by the robot.

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

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

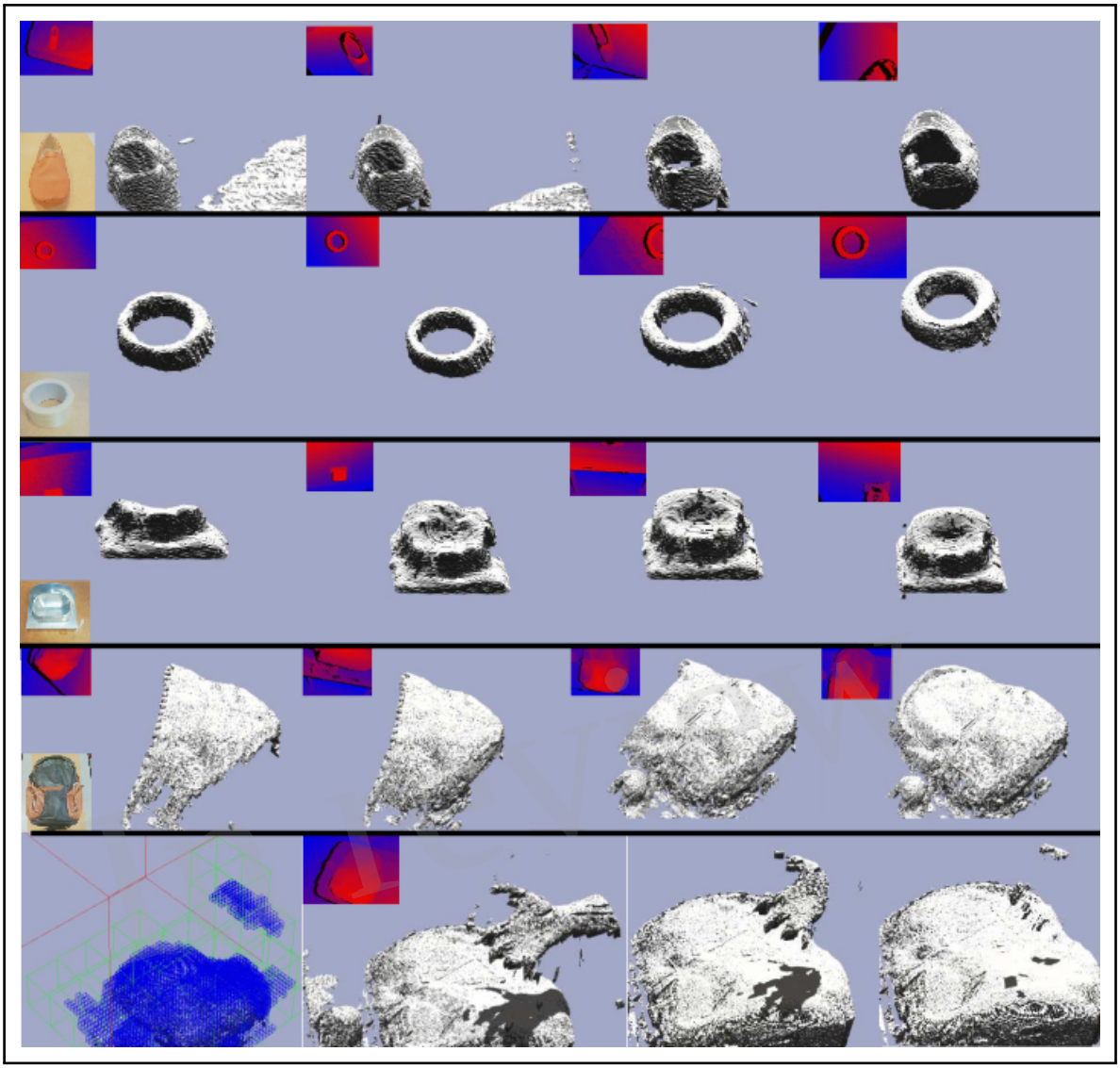

Figure 6. Snapshots of reconstructions experiments for 4 different objects: a shoe (1st row), a tape (2nd row), a small aluminium piece (3rd row) and a backpack (4th and 5th rows).

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

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

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

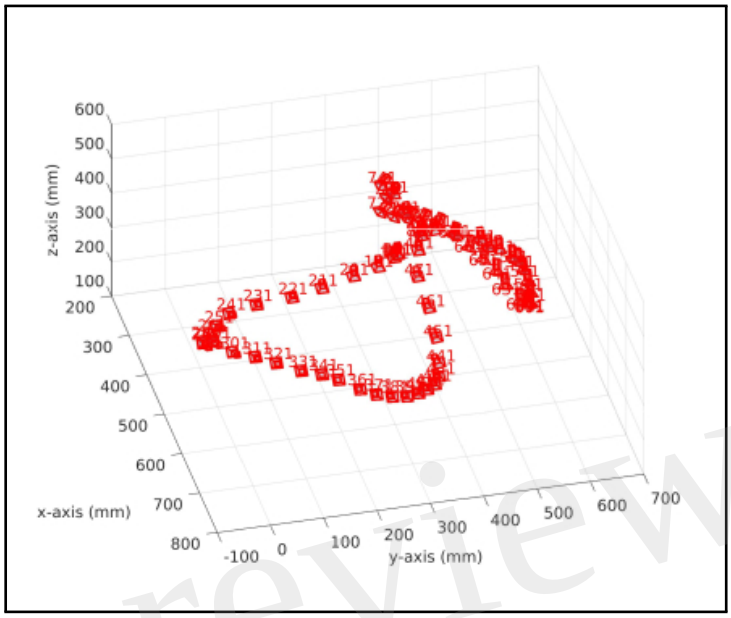

Figure 7. Trajectory of the camera used during the experiments. We sample pose in order to take 1 each 10 poses.

5 CONCLUSIONS

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

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.

FUNDING

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n^{o} 869855 (Project SoftManBot) and by Regional Founds for Auvergne (France).

ACKNOWLEDGMENT

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

REFERENCES

Amanatides, J. and Woo, A. (1987). A Fast Voxel Traversal Algorithm for Ray Tracing. In *Eurographics*. vol. 87, 3–10. doi:https://doi.org/10.2312/egtp.19871000

[Dataset] Besl, P. and McKay, N. (1992). A Method for Registration of 3-D Shapes. doi:10.1109/34.121791

Chen, J., Bautembach, D., and Izadi, S. (2013). Scalable real-time volumetric surface reconstruction. *ACM Trans. Graph.* 32, 1. doi:10.1145/2461912.2461940

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '96* (New York, New York, USA: ACM Press), 303–312. doi:10.1145/237170.237269

Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., and Nießner, M. (2018). ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans. In *IEEE Conf. Comput. Vis. Pattern Recognit.* 4578–4587. doi:10.1109/CVPR.2018.00481

Hoetzlein, R. K. (2016). GVDB: Raytracing Sparse Voxel Database Structures on the GPU. *High Perform. Graph.* doi:10.2312/hpg.20161197

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robots* 34, 189–206. doi:10.1007/s10514-012-9321-0

Hou, J., Dai, A., and Nießner, M. (2019). 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4421–4430

Izadi, S., Kim, Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., et al. (2011). KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. *Proc. 24th Annu. ACM User Interface Softw. Technol. Symp. - UIST '11* , 559–568doi:10.1145/2047196.2047270

Johnson, T. and Sasha, D. (2002). The performance of current B-tree algorithms. *ACM Trans. Database Syst.* 18, 51–101. doi:10.1145/151284.151286

Marchand, É. and Chaumette, F. (2002). Virtual visual servoing: A framework for real-time augmented reality. *Comput. Graph. Forum* 21, 289–297. doi:10.1111/1467-8659.t01-1-00588

Museth, K. (2013). VDB: high-resolution sparse volumes with dynamic topography. *ACM Trans. Graph.* 32, 1–22. doi:10.1145/2487228.2487235

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., et al. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE Int. Symp. Mix. Augment. Reality, ISMAR 2011* (IEEE), 127–136. doi:10.1109/ISMAR.2011.6092378

Nguyen, C. V., Izadi, S., and Lovell, D. (2012). Modeling kinect sensor noise for improved 3D reconstruction and tracking. *Proc. - 2nd Jt. 3DIM/3DPVT Conf. 3D Imaging, Model. Process. Vis. Transm. 3DIMPVT 2012* , 524–530doi:10.1109/3DIMPVT.2012.84

Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. Graph.* 32, 1–11. doi:10.1145/2508363.2508374

Puri, P., Jia, D., and Kaess, M. (2017). GravityFusion: Real-time dense mapping without pose graph using deformation and orientation. *IEEE Int. Conf. Intell. Robot. Syst.* 2017-Septe, 6506–6513. doi:10.1109/IROS.2017.8206559

Rohrbach, A., Rohrbach, M., Hu, R., Darrell, T., and Schiele, B. (2016). Computer Vision – ECCV 2016. *1511.03745V1* 9905, 1–10. doi:10.1007/978-3-319-46448-0

Roth, H. and Marsette, V. (2012). Moving Volume KinectFusion. *Proc. Br. Mach. Vis. Conf.* , 112.1—-112.11doi:http://dx.doi.org/10.5244/C.26.112

Son, H., Kim, C., and Kim, C. (2015). 3D reconstruction of as-built industrial instrumentation models from laser-scan data and a 3D CAD database based on prior knowledge. *Autom. Constr.* 49, 193–200. doi:10.1016/j.autcon.2014.08.007

Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). ElasticFusion: Real-time dense SLAM and light source estimation. In *Int. J. Rob. Res.* 14, 1697–1716. doi:10.1177/0278364916669237

Wu, K., Truong, N., Yuksel, C., and Hoetzlein, R. (2018). Fast fluid simulations with sparse volumes on the GPU. *Comput. Graph. Forum* 37, 157–167. doi:10.1111/cgf.13350

Yang, Z., Gao, F., and Shen, S. (2017). Real-time monocular dense mapping on aerial robots using visual-inertial fusion. *Proc. - IEEE Int. Conf. Robot. Autom.* , 4552–4559doi:10.1109/ICRA.2017.7989529

Zeng, M., Zhao, F., Zheng, J., and Liu, X. (2013). Octree-based fusion for realtime 3D reconstruction. *Graph. Models* 75, 126–136. doi:10.1016/j.gmod.2012.09.002

Zhang, T., Liu, J., Liu, S., Tang, C., and Jin, P. (2017). A 3D reconstruction method for pipeline inspection based on multi-vision. *Meas. J. Int. Meas. Confed.* 98, 35–48. doi:10.1016/j.measurement.2016.11.004

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vis.* 13, 119–152. doi:10.1007/BF01427149

Figure 1.JPEG

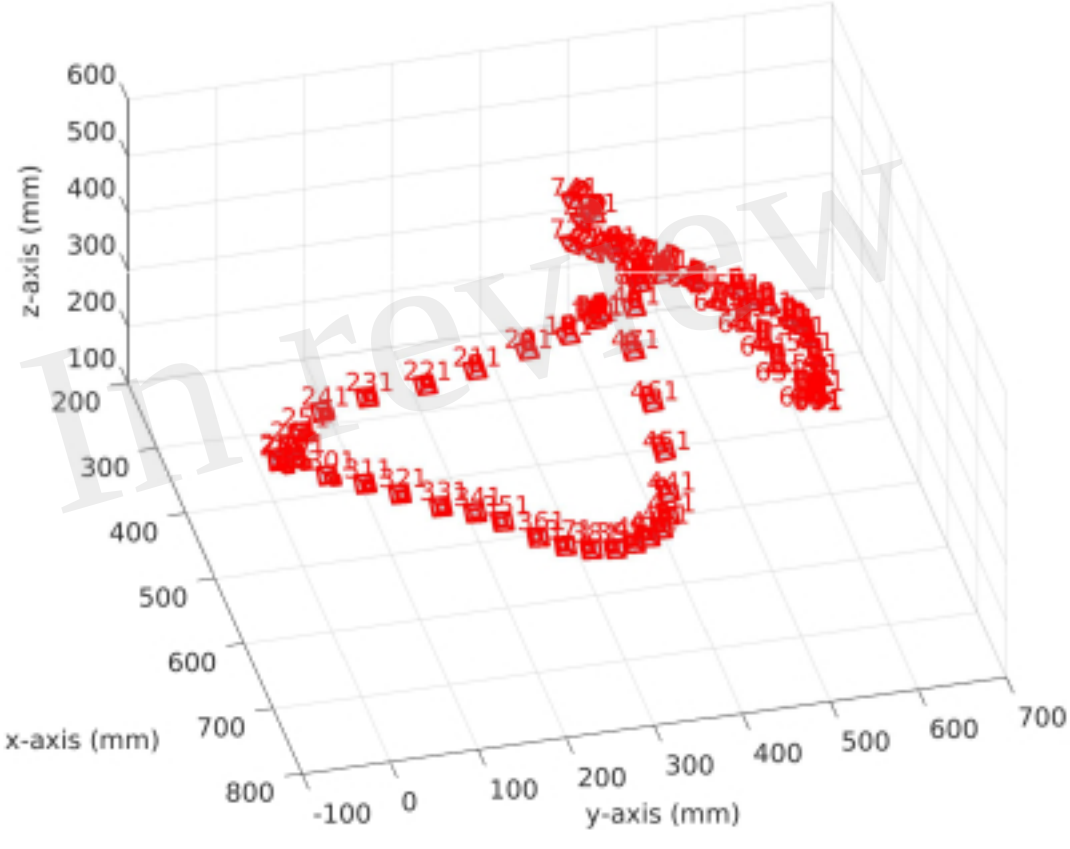

Figure 2.JPEG

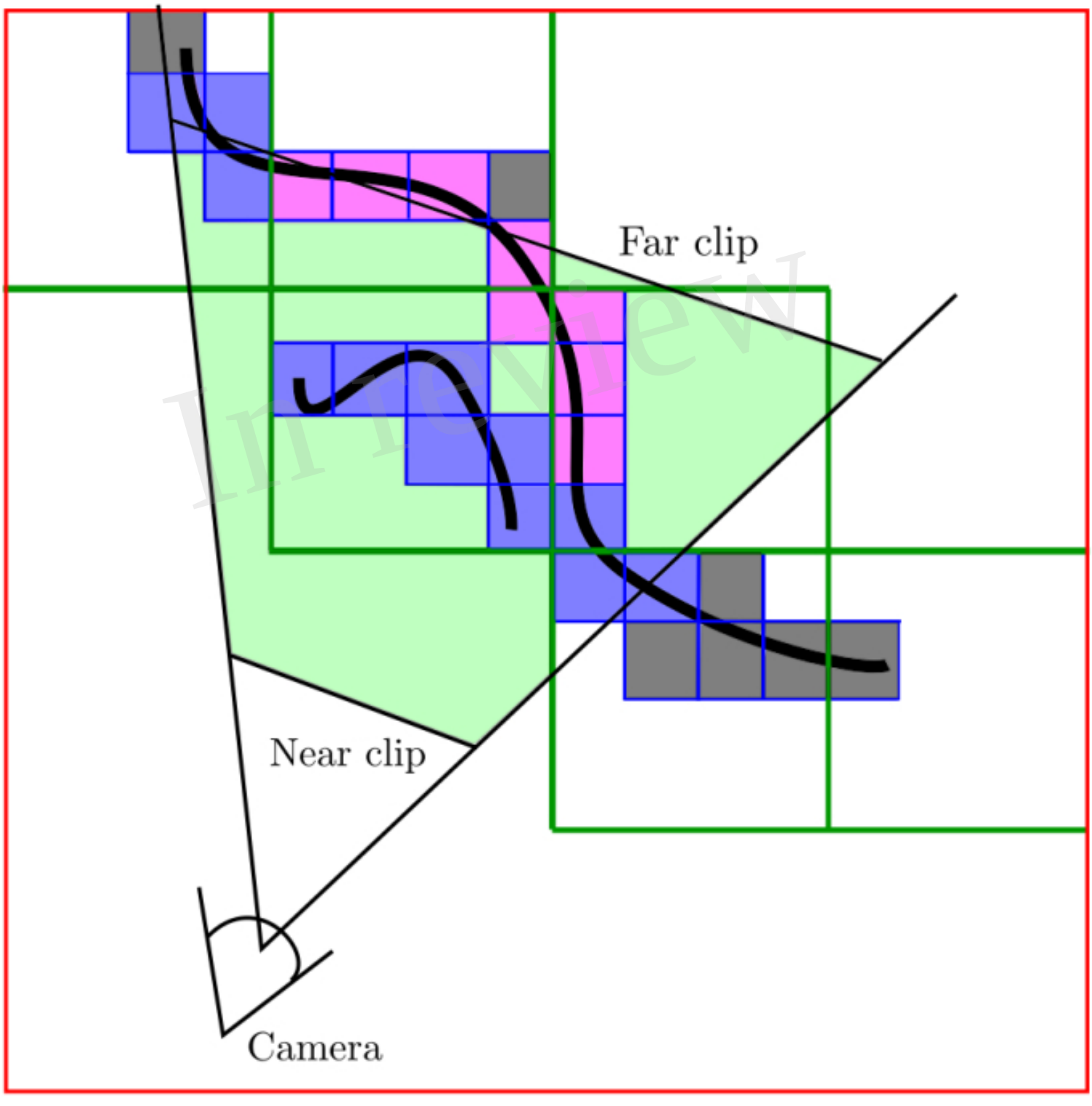

Figure 3.JPEG

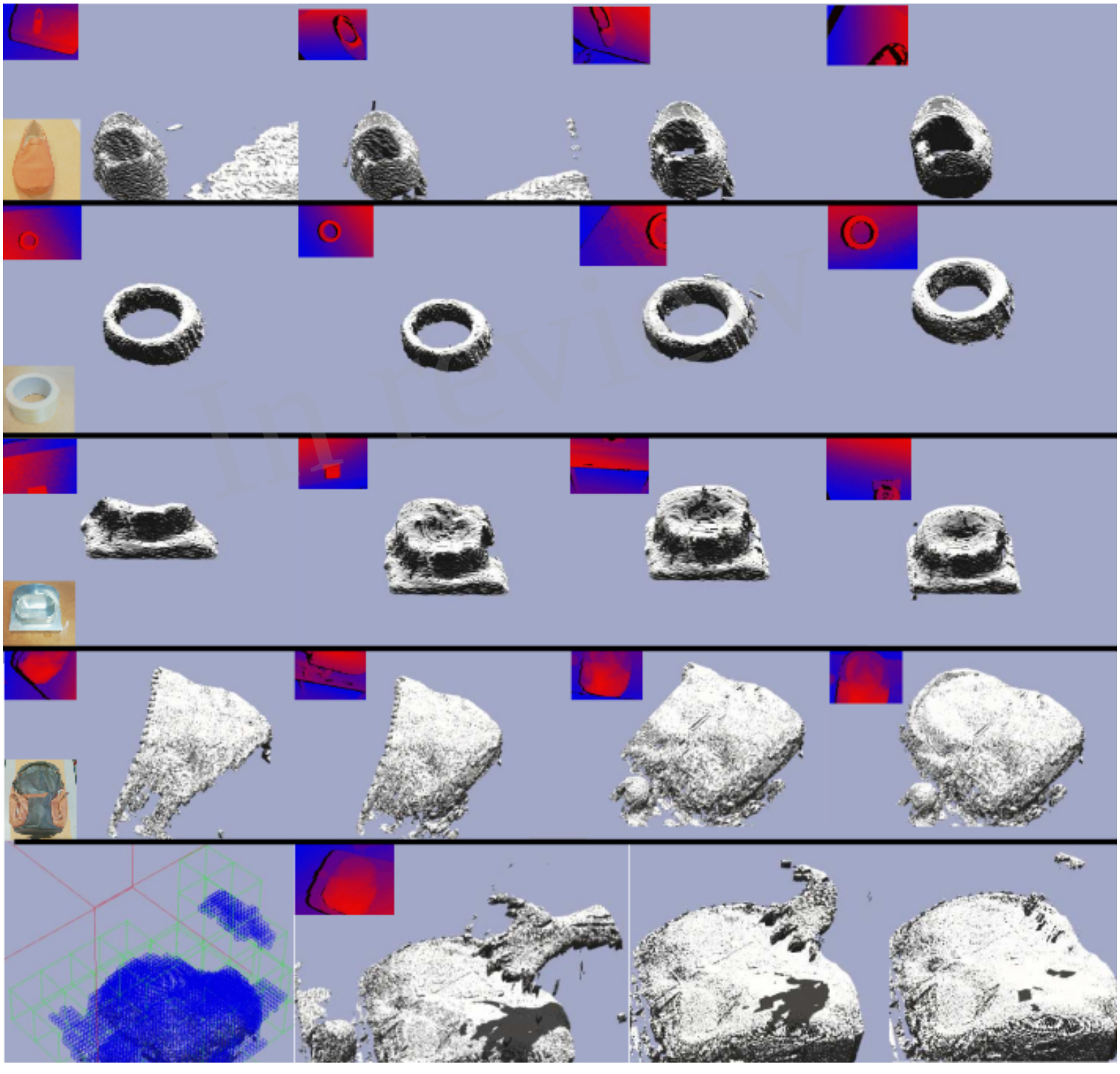

Figure 4.JPEG

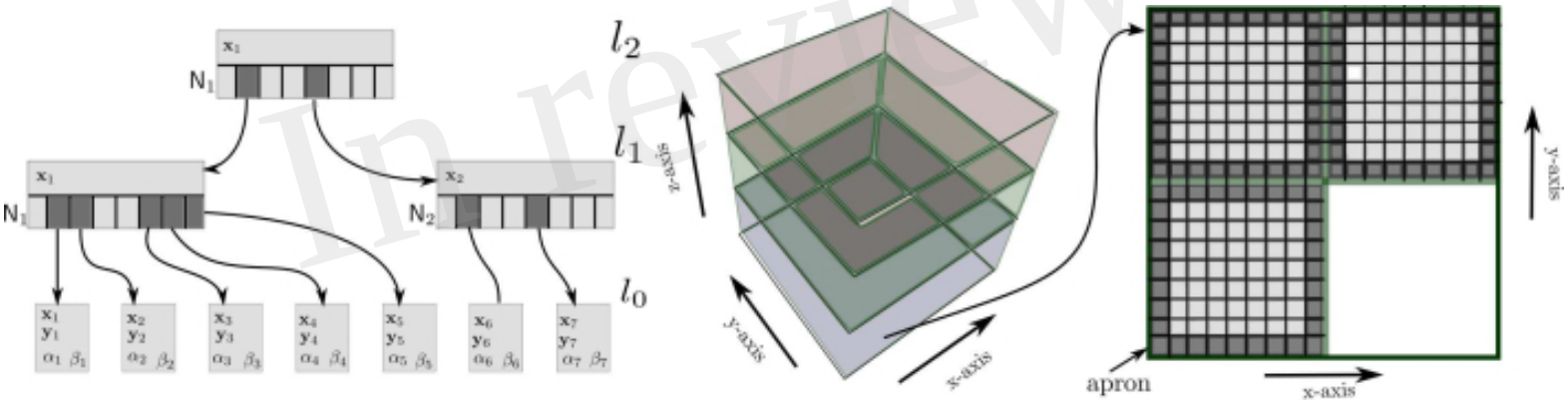

Figure 5.JPEG

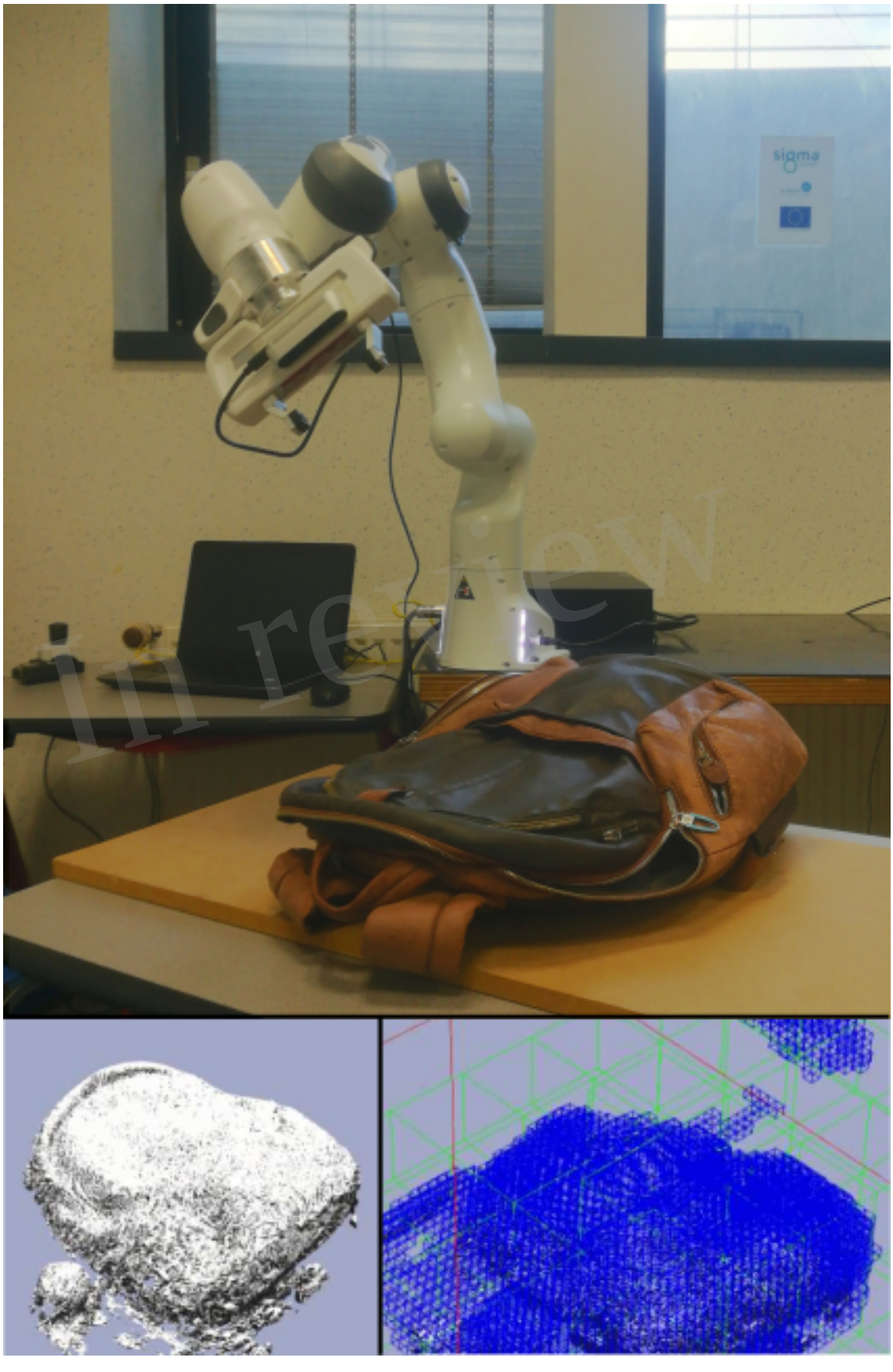

Figure 6.JPEG

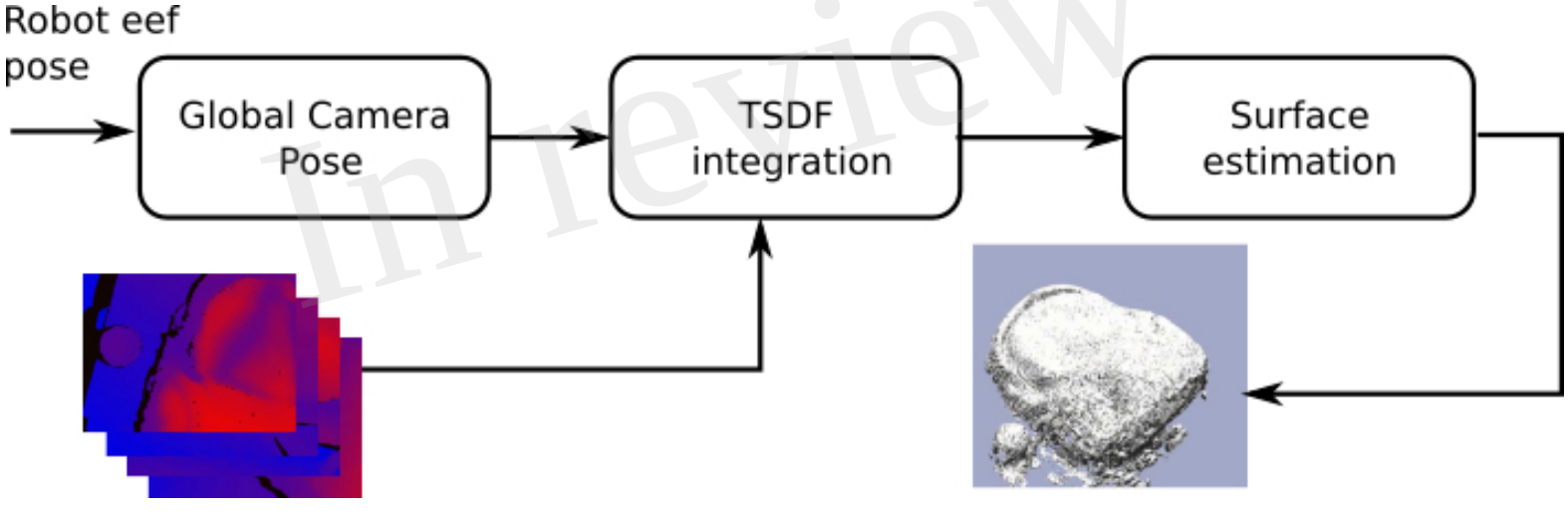

Figure 7.JPEG