On-Line Large Scale Semantic Fusion

Tommaso Cavallari(✉) and Luigi Di Stefano

Department of Computer Science and Engineering,
University of Bologna, Bologna, Italy
{tommaso.cavallari,luigi.distefano}@unibo.it

Abstract. Recent research towards 3D reconstruction has delivered reliable and fast pipelines to obtain accurate volumetric maps of large environments. Alongside, we witness dramatic improvements in the field of semantic segmentation of images due to deployment of deep learning architectures. In this paper, we pursue bridging the semantic gap of purely geometric representations by leveraging on a SLAM pipeline and a deep neural network so to endow surface patches with category labels. In particular, we present the first system that, based on the input stream provided by a commodity RGB-D sensor, can deliver interactively and automatically a map of a large scale environment featuring both geometric as well as semantic information. We also show how the significant computational cost inherent to deployment of a state-of-the-art deep network for semantic labeling does not hinder interactivity thanks to suitable scheduling of the workload on an off-the-shelf PC platform equipped with two GPUs.

Keywords: SLAM · Deep learning · Semantic segmentation · Large scale reconstruction · Semantic fusion

1 Introduction

Most previous work on recovery the world from images has been concerned with 3D geometry only, the advent of commodity RGB-D sensors having made this task remarkably affordable and effective. On the other hand, Deep Learning is emerging as the state of the art approach to infer complex semantics from images. In this paper we bring together geometric reconstruction by RGB-D sensing and semantic perception by Deep Learning to create a novel Semantic SLAM pipeline. With the proposed system, the user can explore the environment interactively by a hand-held RGB-D sensor. As in most previous work, this allows to attain a dense, detailed 3D reconstruction of the scene; peculiarly to our system, though, the resulting map is also endowed *online* and *fully automatically* with semantic labels determining the likelihood of each surface patch to depict objects of specific categories. To achieve this objective, we build

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-49409-8_10) contains supplementary material, which is available to authorized users.

upon a deep convolutional network for semantic image segmentation [11] and a real-time reconstruction approach suited to map large-scale environments [15]. Driving factor behind the development of this pipeline is the need for a system whereby an untrained user may reliably scan and acquire semantically annotated 3D reconstructions of large indoor environments. As highlighted in Sect. 2, previous work [18] would allow generation of similarly annotated 3D maps while requiring proper interaction with a trained user. Conversely, to minimize the effort by the user, we integrate seamlessly dense mapping and semantic labeling into a single pipeline that can output detailed reconstructions of large scale environments wherein each voxel stores a complete probability mass function over a set of semantic categories of interest. Hopefully, our accomplishment may foster research on topics such as indoor scene understanding, object discovery and/or recognition, human/robot interaction and navigation.

The paper is organised as follows: next section discusses previous work related to the proposed system, which will then be described in Sect. 3. Quantitative and qualitative results are provided in Sect. 4, while in Sect. 5 we will draw concluding remarks.

2 Related Work

One of the first breakthroughs in the field of real time 3D reconstruction is KinectFusion by Newcombe *et al.* [14]. Their system shows how the processing power of modern GPUs and the availability of affordable RGB-D sensors can be harnessed to accurately reconstruct the workspace in real time. KinectFusion, though, is bound to map small scale environments due to its reliance on a dense voxel grid as mapping data structure. Several subsequent works tackled this shortcoming, at first by moving the active reconstruction volume alongside with camera movements and downloading from GPU to CPU memory the map previously observed by the sensor [16,20]. More sophisticated data structures aimed at storing only those pieces of information required by the mapping task were then introduced, so to enlarge significantly the mappable volume and speed up the computation, either via hierarchical, octree-based methods [22], hash-based data-structures [9,15] or combinations of both techniques [10].

Thanks to the focus on deep learning in the last years, several semantic segmentation techniques were proposed that could process entire images in fractions of a second, providing pixel-wise category labels or probability mass functions over a set of such categories. Gupta *et al.* [8], process pairs of RGB and Depth images with multiple deep neural networks followed by an SVM classifier, providing a threefold output: bounding boxes for object detection, per-pixel confidences to segment such instances and a full-image semantic segmentation output. Long *et al.* [11] show how Fully Convolutional Networks can provide accurate per-pixel, per-category scores on entire images in a deterministic amount of time. Eigen and Fergus [6] demonstrate how a single deep network architecture can successfully be employed for three different tasks: predicting depth and normals from RGB images as well as performing semantic segmentation to infer, again, per-pixel category probabilities. Zheng *et al.* [23], then, join the strengths of Conditional Random fields and Convolutional Neural Networks within a unique framework trained end-to-end to obtain semantic segmentation.

Recently, works concerning semantic labeling of reconstructed environments have started appearing: Valentin *et al.* [18] show a system, based on the InfiniTAM 3D reconstruction pipeline [9], that, employing multi-modal user interaction, can learn to classify user selected categories via random forests trained on-line. Miksik *et al.* [13] deploy a setup based on a head-mounted stereo camera together with the VoxelHashing 3D reconstruction pipeline [15]; by tracking the target of a portable laser pointer through the acquired frames, the user is able to mark areas of the scene as pertaining to a certain object category. Such labels are then fed to a densely connected Conditional Random Field that learns how to classify voxels online in the reconstructed scene. Differently from these recent works, the pipeline proposed in this paper does not require any user interaction to perform the labeling and train the classifier, and, therefore, an untrained user can proficiently reconstruct large scale environments just by moving around a hand-held RGB-D sensor.

The work by Cavallari and Di Stefano [5] shows integration of the semantic labels output by the Fully Convolutional Networks [11] into a dense reconstruction obtained by the original KinectFusion [14]. While the approach described in [5] is similar to that proposed in this paper, their pipeline cannot map accurately large workspaces due to reliance on a dense voxel grid. Moreover, the structure of their pipeline makes it impossible to achieve interactive frame rates with current hardware. Hence, the system presented in this paper is the first ever to permit on-line fully automatic semantic reconstruction of large environments.

Finally, unlike all the above mentioned works addressing volumetric semantic reconstruction, the pipeline proposed in this paper yields at each voxel the full probability mass function across categories rather than estimating the most likely label only. Such a richer output enables not only generation of semantically labeled maps but also assessment of the likeliness of each and every category across the whole scene surface.

3 Description of the Method

The proposed pipeline is composed of two subsystems, each tailored to a specific task, controlled by a main engine handling all input/output operations and dispatching work to both. The two subsystems are:

Labeling Subsystem: tasked with semantically labeling the RGB images gathered from the sensor.

SLAM Subsystem: dealing with camera tracking, map building and on-demand rendering of the reconstructed 3D scene from arbitrary viewpoints.

In the next paragraphs we will provide a detailed description of the above subsystems and then show how the main engine ties them together to attain the overall system.

3.1 Labeling Subsystem

This subsystem represents the interface of our pipeline to an image-based semantic labeling algorithm: given an input RGB image and, optionally, a depth map, this block provides per pixel confidences for a set of categories of interest, thus providing us with a full probability mass function across categories for each pixel of the input image. More specifically, given input images of size $H \times W$ and a set of N categories of interest, C, the output is a "volume" of confidences, L, of size $N \times H \times W$ and wherein each element $L_{i,j,k}$ represents the confidence that the semantic labeling algorithm assigns to category i at pixel (j, k). Should a single label for a pixel become necessary, a simple argmax operation over the N confidences would provide the required output. In our system, though, we exploit the availability of multiple confidences at each image location to reconstruct a multi-label 3D map of the environment wherein each voxel is endowed with information about the likeliness of each category of interest.

The interface just described is sufficiently generic that any labeling algorithm may in principle be incorporated within our pipeline. For instance, algorithms returning rectangular or polygonal ROIs with associated labels can have their output post-processed to paint each ROI in the volume "slice" associated to the correct category. Overlapping ROIs of the same category may also be handled, e.g. by applying a max operator to the confidence stored in each pixel whereas overlapping regions of different categories can be drawn on the corresponding slices and a final per-pixel normalization can then turn the confidence values for each pixel in a proper probability mass function. Additionally, multiple labeling algorithms may be deployed, the only requirement being to run a normalization step independently on each pixel volume "column". Yet, to minimize the postprocessing necessary to obtain the labeled volume, those inherently more amenable to our pipeline are semantic labeling algorithms providing directly per-pixel confidences across categories, nowadays the most effective and efficient proposals in this space relying on Deep Learning [6,8,11,23]. As such, we found Deep Learning particularly conducive to on-line, fully-automatic semantic mapping.

Among deep networks for semantic labeling, our pipeline deploys the Fully Convolutional Network by Long et al. [11], as we found experimentally that, in our settings, this architecture can provide quite clearly the best trade-off between classification accuracy and speed. Given an input RGB image, the pre-trained networks[1] can yield per-pixel confidences for a large number of categories (20, 40 or 60, depending on the specific model) dealing with both indoor and outdoor objects. As the use case of our system concerns mapping indoor environments by a commodity RGB-D sensor, we reduce the number of categories of interest by dropping some unnecessary classes and applying per-pixel softmax normalization on the remaining raw scores to convert the output into a probability mass function.

[1] https://github.com/shelhamer/fcn.berkeleyvision.org.

3.2 SLAM Subsystem

The generation of a semantic map of the observed environment is a task left to the SLAM Subsystem. A typical Simultaneous Localization and Mapping pipeline consists of two main components: the first localizes the camera within the environment by tracking its movements over time (*localization* task); the second relies on the estimated camera pose to integrate the data provided by the sensor into the current representation of the scene (*mapping* task). Typically a third, optional, component is tasked with visualization of the reconstructed scene to provide feedback to the user.

In the system presented in this paper we add a fourth component to perform what we call the *semantic fusion* task, i.e. integration within the reconstructed scene of the semantic information provided by the Labeling Subsystem. This might also be seen as part of the standard *mapping* task but, as we will illustrate in Subsect. 3.3, we split the standard SLAM mapping operation (integrating data from the RGB-D sensor) and the semantic mapping operation (integrating the information provided by the labeler) in order to decouple them and allow for deferred integration of the per-pixel category probabilities into the scene representation. Indeed, this approach is mandatory to enable on-line operation of the overall semantic reconstruction pipeline.

The SLAM subsystem adopted in our system is built on top of the VoxelHashing reconstruction pipeline by Nießner et al. [15] that, unlike KinectFusion [14], permits mapping of large workspaces by storing the map as a hash-based data structure instead of a dense voxel grid. For a detailed description of VoxelHashing we refer the interested reader to the original paper. In the following, we highlight the main modifications required to store the semantic information peculiar to our approach.

Map Generation and Storage. VoxelHashing employs a hash-based data structure to efficiently index a heap of voxel data blocks. Each voxel block represents the map of a limited region of space. By storing into the GPU memory only such blocks conveying informations useful to the mapping task, and employing an efficient swapping technique to move unneeded blocks from GPU to CPU memory and vice-versa, the extent of the mappable environment can, in principle, be of arbitrary size. Each voxel in the map is endowed with three tokens of information:

TSDF Value: The truncated signed distance from the voxel to the closest surface; being a floating point value, it can be stored as a half precision number so to occupy 2 bytes of memory;

Weight: The confidence in the stored TSDF value; it is used in operations such as fusion of new depth measurements and raycasting of the map; typically is akin to a counter of the number of times the specific voxel has been observed, though other weighting strategies have been proposed [3, 14]; again, a half-precision floating point number;

RGB Data: Colour of the surface patch associated with the voxel; typically encoded as a 4-tuple of unsigned chars to optimize memory alignment.

The memory occupancy for a single voxel amounts thus to 8 bytes. In our pipeline, we augment the standard voxel data structure by a histogram storing a probability mass function over a set of N categories. Each bin represents the probability that an item of a specific category is located in the surface area associated with the voxel. Each histogram bin should therefore be able to encode a floating point value in the interval [0..1]. To reduce memory occupancy, we encode such values into bytes by scaling the floating point number to the interval [0..255]. The final size of the voxel data structure thus increases by N bytes. The set of categories is application dependent and in our tests we employ 8 categories, thus having each voxel occupying 16 bytes, thereby doubling the memory footprint with respect to the standard data structure. Doubling the per-voxel memory occupancy would be worrying if we were using a dense data structure as deployed by KinectFusion. Conversely, thanks to the reduced memory pressure allowed by VoxelHashing, we can easily accommodate such informations onto the GPU memory and, if necessary, move it back and forth with the system RAM via swapping operations. When a new voxel is allocated by VoxelHashing, its histogram is set to the uniform probability, thus having each bin initialized to the value $255/N$, so to express maximum uncertainty on the type of object located within its boundaries.

Rendering. Visualization of the reconstructed scene is typically performed via raycasting. First, a synthetic range image is extracted: given a camera pose of interest, a ray is marched for each pixel of the output image from the camera center until a positive to negative zero-crossing of the TSDF function is encountered, this signaling the presence of a surface. Clearly, marching a ray from the camera centre is expensive since the hash table has to be queried for every step, therefore several optimizations are described in the VoxelHashing paper [15]. The InfiniTAM pipeline [9] also details more enhancements to the raycasting operation that can speed up sensibly the computation.

The raycasted range map can then be used to extract a coloured representation of the environment by trilinearly interpolating the RGB values of the 8 voxels closest to each zero crossing point. Point normals can also be computed by estimating the TSDF gradient in the location corresponding to the range map point.

Semantic labels for each rendered point can then be extracted. In order to determine the label of a single pixel, we apply an argmax operation over the N histogram bins associated to each raycasted 3D point and store the resulting label in an output category map and the associated confidence in an output probability map. While we could trilinearly interpolate between bins associated to the histograms of 8 neighboring voxels in order to obtain an interpolated histogram to subject the argmax operation, in practice we consider only the voxel whose center is closest to the candidate 3D point, on account that, typically, object categories are "large scale" scene attributes and the interpolation of neighboring voxel probabilities would not provide much additional information while notably slowing down the processing speed of the pipeline. The left picture in Fig. 1 provides an exemplar image obtained by raycasting into the current camera view the most likely label in each voxel provided by the argmax operation.

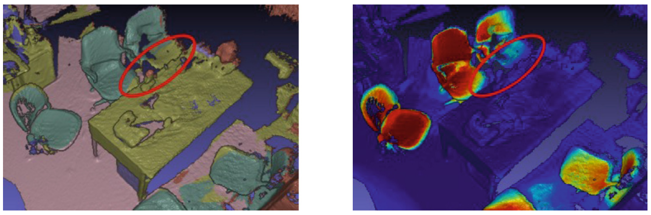

Fig. 1. Left: semantic labels assigned to the reconstruction of an office environment. Right: heat map showing the spatial distribution of "chair" objects. It can be seen how, even in presence of mislabeled areas (highlighted by the red ellipse), the "chair" confidence is not null and thus may help to segment out chairs. (Color figure online)

Our category representation scheme allows also to render the likeliness of a category in each voxel seen from the current camera view. Indeed, we can provide visualizations detailing the spatial distribution of a certain category of interest by selecting the histogram bin associated to that category in every voxel defined by the raycasted range image. For example, as shown in the right picture of Fig. 1, we might wish to render the "chairness" of the reconstructed scene. It is worth observing that, although some surface patches belonging to chairs are mislabeled in the left image of Fig. 1, the right image provides evidence that these indeed may possibly belong to chairs, this information may likely help performing an higher lever task such as segmenting out all the chairs present in the scene.

Once the range, normals, RGB, category and score maps are extracted, shading can be applied to obtain pleasant visualizations. While the described renderings may convey useful informations to the user, all but those dealing with the range and normal maps are optional in our pipeline and thus can be disabled to increase processing speed. The renderings of the range and normal maps, instead, are pivotal in the camera localization step that will be described next.

Camera Localization. Camera localization is an essential step of the SLAM pipeline: to integrate RGB-D frames coming from the sensor into the global map of the environment, one has to know the pose from which the camera captured such informations. Typically, thanks to the high processing rate of KinectFusion/VoxelHashing, this task can be simplified into the tracking of the camera movement from one frame to the following.

Several approaches to camera tracking task have been proposed in the literature related to KinectFusion, either relying on purely geometric clues, such as the projective ICP approach of the original paper [14] and the direct alignment methods described in [3,4], or aimed at deploying colour information to maximize the photo-consistency between pairs of consecutive frames [17,19] or between the current frame and the colour information stored into voxels [2].

In our system we employ the projective ICP approach described by Newcombe et al. [14] to estimate sensor pose. The method relies on the raycasted depth and normals map as seen from the previous camera pose and the current depth map. An iterative process performs a projective association between points in the current and in the raycasted range maps [1], computes an energy term at each pixel based on the point to plane metric [21], and finally minimizes the sum of all pixel energies by linearising such function in a neighborhood of the previous pose and computing an increment using the Lie algebra representation.

Semantic Fusion. The Semantic Fusion component integrates into the voxel-based map of the environment the per-pixel semantic labels extracted from an input RGB-D frame by the Labeling subsystem. As mentioned earlier, this task is kept disjoint from the canonical "fusion" operation performed by KinectFusion/VoxelHashing to allow the labeler to work asynchronously with respect to the SLAM process, thus not hindering the real-time nature of the latter due to the former being significantly slower.

Once a frame has been labeled, its associated pose, T_l, which was estimated by the camera localisation component, is retrieved and can be used to perform the actual fusion step. Likewise fusion of the RGB-D image, the process is applied to those voxels that fall into the camera frustum and are "close enough" to the surface described by the depth frame associated with the previously extracted labels (cached at the beginning of the labeling); purposely, we employ the same truncation distance as used by the depth integration step.

More precisely, as a first step, the location of each mapped voxel block in the world coordinate frame is transformed in the appropriate camera reference frame by the inverse transformation described by the camera pose. The transformed block center is then projected onto the image plane and, if the resulting coordinates lie inside the image, the block is marked as potentially visible and thus to be updated. Thanks to the GPU, this first step can be efficiently carried out in parallel by associating a thread to each voxel block. A *scan-and-compact* operation is then performed to gather the indexes of all the blocks to be updated in a single buffer, which in turn is used to launch an update thread for each voxel residing in such blocks.

Each voxel center is then projected onto the depth frame by applying the T_l^{-1} transformation and the depth camera intrinsics; its associated depth is then sampled: if the 3D point determined by such depth is sufficiently close to the voxel itself, then the label probabilities vector is subject to the update operation that will be described next.

To integrate the pixel category probabilities provided by the labeling algorithm into the probability histogram stored in each voxel, we perform an operation akin to the running average adopted for the depth integration step followed by a renormalization step to ensure attainment of a valid probability mass function. Denotes as $L \in \mathbb{R}^N$ the pixel p.m.f. and $H \in \mathbb{R}^N$ the corresponding voxel p.m.f., firstly we compute a weight w_p associated to the pixel (employing the same strategy used during the depth integration phase, i.e. we assign a unitary weight to the new labels; see also the description of the Weight field in Sect. 3.2); we then sample the weight stored in the voxel w_v. We compute the updated probability histogram H' as follows:

$$H'_i = \frac{H_i w_v + L_i w_p}{w_v + w_p} \qquad \text{with } i \in [1..N] \tag{1}$$

We then normalize the histogram to obtain a valid probability mass function, H'', that is stored back into the voxel:

$$H''_i = \frac{H'_i}{\sum_i H'_i} \qquad \text{with } i \in [1..N] \tag{2}$$

We do not update the associated voxel weight, leaving that task to the depth fusion component. While weights w_v depend on the number of times a specific voxel has been observed and that number typically differs from the number of times a voxel has been semantically labeled, we found no significant difference between using an ad-hoc semantic weight (that would need to be stored alongside the p.m.f.) and just piggy-backing on the already present TSDF weight. Hence, we exploit the w_v values to give an appropriate strength to the past probability values and prevent a single measurement from significantly changing the stored probabilities. Also, as typically the frame rate of the SLAM subsystem is constant and the time required by the labeling algorithm is also deterministic, the relationship between the depth weight (a counter of the number of integrated frames) and a "semantic weight" would be linear.

3.3 Main Engine

The Main Engine of our system interacts with the RGB-D sensor, dispatches the work to the SLAM and Labeling subsystems, and provides the user with feedback on the on-going operation by displaying rendered images.

One of the key novelties of our proposal is its ability to perform SLAM and semantic mapping fully automatically and on-line. This means that while the user moves around the RGB-D sensor she/he would see on the screen a semantic reconstruction of the workspace created incrementally at interactive frame-rate. In other words, while in KinectFusion/VoxelHashing the user would perceive incremental reconstruction of the geometry of the scene interactively, our system is aimed at providing, just as interactively, both geometry and semantics in the form of surfaces tagged with category labels. Processing speed is therefore of paramount importance to the pipeline as a whole. However, while the SLAM Subsystem can comfortably keep-up with the 30 Hz RGB-D stream delivered by the sensor, state-of-the-art deep networks for semantic labeling require hundreds of milliseconds or even seconds to process a single frame.

This state of affairs mandates the two subsystems to be decoupled so to execute their code in parallel and prioritize SLAM to provide interactive feedback

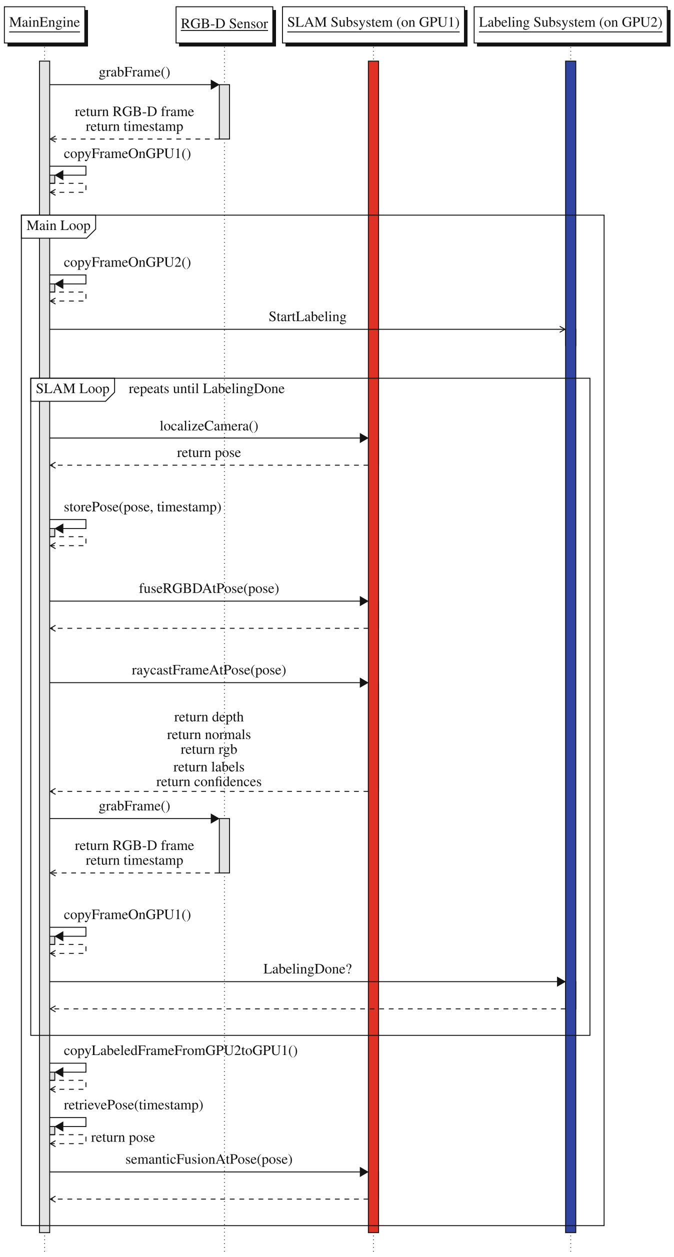

Fig. 2. Sequence diagram depicting the execution flow of the proposed system. The SLAM and Labeling subsystems are deployed on two different GPUs, here colour coded in red and blue. The main engine moves the data between the host memory and the two GPU memories as needed. (Color figure online)

to the user. The Labeling subsystem is thus run on the remaining CPU and GPU time and, by exploiting the deferred Semantic Fusion algorithm described in Sect. 3.2, its output is integrated into the voxel map as soon as it becomes available. To obtain an even higher throughput, we deploy the two subsystems onto two different GPUs, the Main Engine performing the appropriate copies or movements of the data to and from the different boards.

While all frames captured by the RGB-D sensor are used to perform the SLAM task (necessary to obtain an accurate map and a reliable camera localisation), only a minority of those are provided to the Labeling Subsystem. Choice of such candidate frames is left to the Main Engine and performed in a greedy fashion by ignoring all frames grabbed while the Labeling Subsystem is busy labeling a frame and marking for labeling the first new frame received after the labeler has finished its work on a previous frame. We elected not to use a queue-based system to privilege the labeling of several areas of the environment instead of filling the queue with similar frames acquired by nearby viewpoints and have the labeler unavailable to process newly explored areas of the environment.

Figure 2 shows a sequence diagram detailing the execution flow of the system. Once an RGB-D frame is grabbed by the sensor, its data is copied to both GPUs and the labeling thread is activated. At the same time, the camera pose from which the environment was observed is estimated by the SLAM Subsystem and stored in a pose database together with the frame timestamp, used to retrieve such pose at a later stage; subsequently, depth and colour information are fused into the hash-based TSDF structure. A raycasting operation is then performed to obtain the range and normals maps required by the ICP algorithm to localize the camera at the next iteration; if semantic visualization is desired, then colour, label and confidence maps are rendered as well.

The main engine then verifies if the labeling algorithm has terminated its computation; if not, another iteration of the SLAM pipeline is executed. Conversely, the labeling output (i.e. the $N \times H \times W$ volume storing per-pixel category probabilities described in Subsect. 3.1) is transferred from the labeler GPU to the SLAM GPU and the viewpoint from which the frame had been observed is retrieved from the pose database. The semantic labels are then fused via the algorithm described in Sect. 3.2. The process is repeated until the user wishes to terminate; at that point the entire map of the environment can be saved as a mesh via application of the marching cubes algorithm [12]. The mesh can be coloured using either the RGB values stored in the hash-based map or a colour mapped representation of category or confidence values.

4 Results

The system presented in this paper pursues interactive and fully automatic semantic mapping of large workspaces. In this section we will show quantitative and qualitative results provided by the system. Firstly, we present an evaluation of the computational requirements of the entire system, detailing the overall impact of the two main subsystems. Afterwards, we show qualitative results depicting the kind of semantic reconstructions that can be achieved by running the system. In the supplementary material we provide a video demonstrating interactive semantic mapping. From the video, one may perceive the effect of the deferred semantic fusion and how this approach does not hinder incremental interactive reconstruction while adding semantic information into the map over time.

4.1 Performance Evaluation

Our system relies on computation modules deployed on both the CPU and two graphics processors. Our testing setup consists in a PC with a Intel Core i7 4960X CPU and two GeForce Titan Black graphics cards (each with 6 GB of dedicated memory). The SLAM Subsystem is deployed on one card while the Labeling Subsytem based on the Fully Convolutional Network [11] is deployed on the other (the amount of GPU memory required by the neural network is ∼5.5 GB).

Table 1 shows the average time spent in the main components of the pipeline. It is evident how the most computationally intensive component of the proposed system is that concerned with semantic labeling of input frames and how its decoupling and deployment on a separate GPU is necessary to maintain an interactive frame-rate (∼17 Hz) that allows users to seamlessly deploy the system to semantically reconstruct a location. For comparison, in the last column we

Table 1. Processing time broken down by component. *Total time per frame does not include the exact time spent in executing the SemanticFusion step as this is performed only after the Labeling Subsystem terminates processing an input RGB-D image and therefore its execution time is amortized over a larger number of frames. The total time per frame is thus the average time spent to process a frame, yielding a frame-rate of 17.48 fps for the multi GPU system and of 5.4 fps for the single GPU setup.

Algorithm section	Times (ms.)		
	Multiple GPU		Single GPU
	GPU 1	GPU 2	
Frame Grabbing + Preprocessing	10.27	–	20.21
Camera Localisation	4.30	–	24.90
Depth + RGB Fusion	8.79	–	16.96
ICP Raycast (Depth + Normals)	5.16	–	13.35
RGB + Labels + Confidence Raycast	6.49	–	23.79
Shading + GUI update	11.32	–	71.64
Other processing	7.70	–	10.26
SemanticFusion*	9.59	–	10.44
Frame Labeling	–	284.09	438.91
Total time per frame*	57.20	284.09	186.55

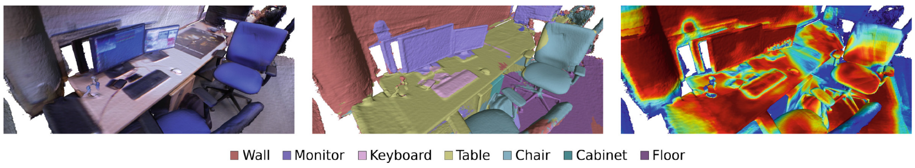

Fig. 3. A semantically reconstructed office environment. The left image shows the coloured mesh generated by our system. The center image shows the most likely category label for each voxel, while the associated confidence values are displayed as a heat map on the right.

show the processing times that can be obtained by running the proposed pipeline on a single GPU accelerator. For this test, we employed a workstation with an Intel Core i7 4930K CPU and a Tesla K40, with 12 GB of memory. The availability of a single GPU, even with twice the memory as those deployed in the previous test, severely hinders the overall system speed due to the computation being bound by the number of available GPU cores rather than by memory availability, this bringing evidence towards the idea of deploying two separate GPUs to realize our system.

4.2 Qualitative Results

Our system enables the user to attain interactively a semantic reconstruction of the environment by employing a hand-held commodity RGB-D sensor such as the Kinect. In this section we show exemplar results obtained in different environments.

Several office sequences depicting a variety of indoor objects such as "monitors", "chairs", "tables", etc... were acquired and processed by our system. Figure 3 shows a view from one of such sequences where chairs, monitors, and the keyboard are labeled quite correctly. Wall and floor regions are also mostly correct while the "table" category shows a slightly lower segmentation accuracy, its labels bleeding into the "cabinet" located below. The Fully Convolutional Network model used to obtain the depicted results is "pascalcontext-fcn8s".

Using the same neural network as in Fig. 3, we have run our system also on sequences belonging to the Stanford 3D Scene Dataset [24,25]. In Figs. 4 and 5 we show the resulting reconstructions. It can be observed how the system can label correctly large objects, such as sofas and tables. Moreover, voxels pertaining to smaller objects, such as the stacked books in Fig. 4, are mostly correctly labeled alike. The two lamps in Fig. 4 (top row) are inevitably mislabeled because "lamp" does not belong to the set of categories handled by the neural network. As concerns the potted plant in Fig. 5, it is worth pointing out that labels tend to propagate into the wall due to the thin and partially reflective nature of the leaves which causes depth estimation by the RGB-D sensor to fail and prevent accurate 3D reconstruction of the object.

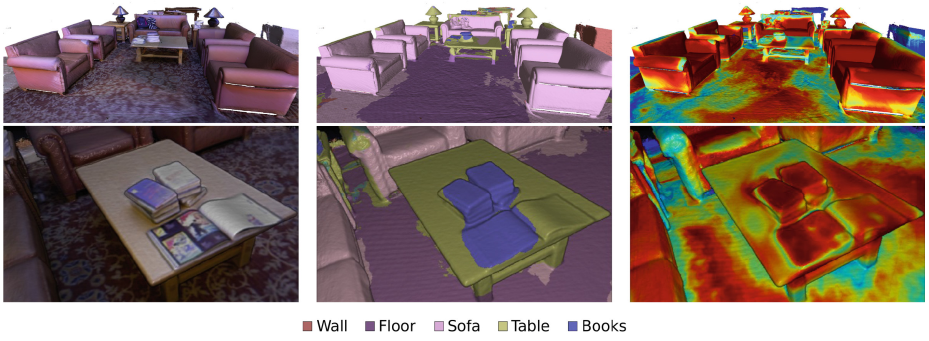

Fig. 4. *ReadingRoom* sequence from the Stanford 3D Scene Dataset. Top row: raycasted views acquired during the reconstruction. Bottom row: details from the final reconstruction. Columns as in Fig. 3.

Throughout our experiments, we observed that the boundaries between different objects are reasonably accurate for smaller items (*e.g.* books, keyboards, chairs, monitors,...) while the contours dealing with larger objects, such as tables, sofas and structural elements (*i.e.* walls, floors and ceilings) tend to be less accurately localized and "bleed" onto neighboring voxels. It is noteworthy, though, that the confidence associated to such incorrect boundary zones is typically significantly lower than that estimated within the internal portions of objects. This effect is especially evident if the object is observed from a significant distance and can be traced back to the 2D nature of the semantic labeling process: labeled pixels are reprojected onto the voxel-based map using the current depth image and thus a single (potentially mis-)labeled pixel affects a larger area of the reconstruction the farther away it is from the camera.

Overall, the experimental findings suggest that our system can label correctly both the main large-size scene structures such as floor, walls, tables, chairs as well as several smaller objects like monitors, books, keyboards. Moreover, the confidence maps turn out quite reliable, due to high confidence labels unlikely turning out wrong and mislabeled areas featuring low scores. Therefore, our semantic reconstructions and associated confidence maps may provide valuable cues to facilitate high-level reasoning pursuing indoor scene understanding.

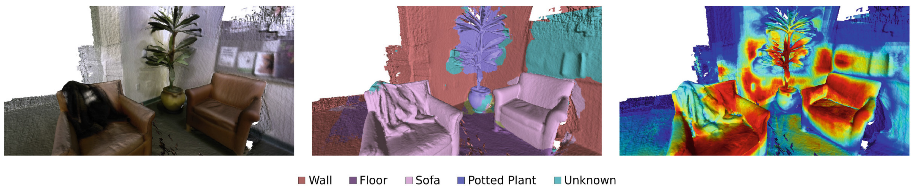

Fig. 5. Details of the reconstruction of the *Lounge* sequence from the Stanford 3D Scene Dataset. Columns as in Fig. 3.

5 Final Remarks

We have presented the first interactive system allowing an user to perform 3D reconstruction of a large scale environment while semantically labeling the observed surfaces seamlessly. To this purpose, we split the proposed pipeline in two subsystems, each relying on state-of-the-art approaches. The Labeling subsystem pursues per-pixel semantic segmentation of RGB images by a recently proposed deep neural network [11]. Indeed, our architecture is agnostic to the actual labeler and, thus, holds the potential to accommodate the advances in the field likely to be provided by the ever-increasing research efforts on deep learning architectures for semantic segmentation and object detection. The SLAM subsystem relies on the VoxelHashing approach [15] to handle reconstruction of large scenes. Nießner's pipeline, though, has been modified to achieve storage, deferred integration and visualization of semantic information. To provide the user with a fluid interactive experience we deploy the proposed system on a off-the-shelf Personal Computer endowed with two GPUs and suitably schedule the work-load on such platforms.

Among the shortcomings of our system is, *in primis*, reliance of the camera localization step on a purely geometric tracking approach (i.e. the standard projective ICP used by VoxelHashing and KinectFusion) which, while good enough to accurately estimate camera poses across frames, is not immune from a certain amount of drift that may become evident when the hand-held sensor is brought back to a previously observed area. Hence, we plan to deploy more recent approaches, such as [7], that may enable exploration of large-scale environments with negligible drift. Another issue worthy of further investigation concerns the accuracy of the semantically labeled 3D maps: sometimes labels bleed onto voxels belonging to neighboring objects due to the independence of category histograms computed at neighbouring voxels. Accordingly, the application of pairwise CRFs either at label integration or mesh generation time to ensure spatial consistency of neighboring labels is currently under investigation.

Acknowledgements. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

1. Blais, G., Levine, M.: Registering multiview range data to create 3D computer objects. IEEE Trans. Pattern Anal. Mach. Intell. **17**(8), 820–824 (1995)
2. Bylow, E., Olsson, C.: Robust camera tracking by combining color and depth measurements. In: 2014 22nd International Conference on Pattern Recognition (ICPR) (2014)
3. Bylow, E., Sturm, J., Kerl, C., Kahl, F., Cremers, D.: Real-time camera tracking and 3D reconstruction using signed distance functions. In: Robotics: Science and Systems (RSS) (2013)
4. Canelhas, D.R., Stoyanov, T., Lilienthal, A.J.: SDF tracker: a parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In: IEEE International Conference on Intelligent Robots and Systems, pp. 3671–3676 (2013)

5. Cavallari, T., Stefano, L.: Volume-based semantic labeling with signed distance functions. In: Bräunl, T., McCane, B., Rivera, M., Yu, X. (eds.) PSIVT 2015. LNCS, vol. 9431, pp. 544–556. Springer, Heidelberg (2016). doi:10.1007/978-3-319-29451-3_43
6. Eigen, D., Fergus, R.: Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. ICCV, November 2015
7. Fioraio, N., Taylor, J., Fitzgibbon, A., Di Stefano, L., Izadi, S.: Large-scale and drift-free surface reconstruction using online subvolume registration. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4475–4483. IEEE, June 2015
8. Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from RGB-D images for object detection and segmentation. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 345–360. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10584-0_23
9. Kahler, O., Prisacariu, V.A., Ren, C.Y., Sun, X., Torr, P., Murray, D.: Very high frame rate volumetric integration of depth images on mobile devices. IEEE Trans. Visual. Comput. Graph. **21**(11), 1241–1250 (2015)
10. Kahler, O., Prisacariu, V., Valentin, J., Murray, D.: Hierarchical voxel block hashing for efficient integration of depth images. IEEE Rob. Autom. Lett. **3766**(c), 1 (2015)
11. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
12. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH 1987, vol. 21, issue 4, pp. 163–169 (1987)
13. Miksik, O., Torr, P.H., Vineet, V., Lidegaard, M., Prasaath, R., Nießner, M., Golodetz, S., Hicks, S.L., Pérez, P., Izadi, S.: The semantic paintbrush. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI 2015, pp. 3317–3326. ACM, New York, April 2015
14. Newcombe, R.A., Davison, A.J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., Fitzgibbon, A.: KinectFusion: real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136. IEEE, October 2011
15. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3D reconstruction at scale using voxel hashing. ACM Trans. Graph. **32**(6), 1–11 (2013)
16. Roth, H., Marsette, V.: Moving volume KinectFusion. In: Proceedings of the British Machine Vision Conference, pp. 112.1–112.11 (2012)
17. Steinbrucker, F., Sturm, J., Cremers, D.: Real-time visual odometry from dense RGB-D images. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 719–722. IEEE, November 2011
18. Valentin, J., Vineet, V., Cheng, M.M., Kim, D., Shotton, J., Kohli, P., Niessner, M., Criminisi, A., Izadi, S., Torr, P.: SemanticPaint: interactive 3D labeling and learning at your fingertips. ACM Trans. Graph. (TOG) (2015)
19. Whelan, T., Johannsson, H., Kaess, M., Leonard, J.J., McDonald, J.: Robust real-time visual odometry for dense RGB-D mapping. In: 2013 IEEE International Conference on Robotics and Automation, vol. 1, pp. 5724–5731. IEEE, May 2013
20. Whelan, T., Kaess, M., Fallon, M.: Kintinuous: spatially extended KinectFusion. In: Robotics Science and Systems (Workshop on RGB-D: Advanced Reasoning with Depth Cameras) (2012)

21. Yang, C., Medioni, G.: Object modelling by registration of multiple range images. Image Vis. Comput. **10**, 145–155 (1992). IEEE Computer Society Press
22. Zeng, M., Zhao, F., Zheng, J., Liu, X.: Octree-based fusion for realtime 3D reconstruction. Graph. Models **75**(3), 126–136 (2013)
23. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.S.: Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1529–1537 (2015)
24. Zhou, Q.Y., Koltun, V.: Dense scene reconstruction with points of interest. ACM Trans. Graph. **32**(4), 112:1–112:8 (2013)
25. Zhou, Q.Y., Miller, S., Koltun, V.: Elastic fragments for dense scene reconstruction. In: 2013 IEEE International Conference on Computer Vision, pp. 473–480. IEEE, December 2013