ACEFusion - Accelerated and Energy-Efficient Semantic 3D Reconstruction of Dynamic Scenes

ACEFusion - Accelerated and Energy-Efficient Semantic 3D Reconstruction of Dynamic Scenes

Mihai Bujanca, Barry Lennox, and Mikel Luján

***Abstract*— ACEFusion is the first 3D reconstruction system able to capture the geometry and semantics of dynamic scenes using an RGB-D camera in real-time on a robotic computing platform. Harnessing the hardware accelerators of an Nvidia Jetson AGX Xavier, the system uses heterogeneous computing to achieve 30 FPS under a 30W power budget. Using a data-parallel design, we perform most image computation on the dedicated hardware accelerators, freeing the general purpose cores and GPU to process 3D geometry. To further increase efficiency, we employ a hybrid geometry representation with octrees for static-semantic reconstruction and surfels for dynamic reconstruction. ACEFusion achieves competitive results on standard benchmarks while efficiently performing a more complex overall task than existing SLAM techniques. Figure. 1 shows the output of our system on a dynamic sequence.**

I. Introduction

Humans and animals construct *cognitive maps*, complex and meaningful internal representations of the world, to achieve their goals. Developing computer systems which allow robots to construct such representations is not only essential for autonomy, but also in enabling interaction between humans and AI systems.

Simultaneous Localization and Mapping (SLAM) captures for robotics the problem of building such maps of the environment. Given a mobile robot placed in an unknown environment at an unknown location, a SLAM system aims to simultaneously create a map of the environment and localize the robot within the map. Advancements in sensors, machine learning, and computing capabilities have led SLAM beyond its initial scope, with mapping capabilities extending to capturing the 3D geometry of static and dynamic environments as well as their semantic aspects. The notion of *Spatial AI* [1] was recently coined to refer to the broader problem of building rich cognitive maps by extending SLAM beyond a position tracking problem.

ACEFusion builds significantly more expressive maps than most Spatial AI systems, describing the geometry and semantics of static and dynamic scene elements. The accuracy of pose estimation and 3D geometry reconstruction is evaluated using datasets available in the SLAMBench framework [2], [3], [4]. To assess robustness [5], we compare the effect of dynamic objects on ACEFusion against other state-of-the-art systems. Two central aspects that distinguish ACEFusion from prior work, and allowing to achieve up to 30FPS on a 30W power budget are the choice of representation and the parallelization scheme.

Whereas other systems typically use a single representation, we adopt a surfel-based representation for dynamic objects, while the static background is modelled using octrees. The insight behind this choice is that surfel-based representations are fit for updating a small, moving object at every frame, while octrees representing a large and static scene can efficiently update the visible part of the scene at every frame and adapt to different levels of granularity as necessary. While prior art requires powerful GPUs to achieve real-time performance, ACEFusion distributes the computational load across the Deep Learning Accelerators (DLA), Video Image Compositor (VIC), general purpose processing cores, and GPU available on the Nvidia Jetson Xavier AGX board. ACEFusion runs image processing and pose estimation in parallel with 3D reconstruction. Expensive image processing tasks are offloaded onto accelerators, allowing most of the cores and GPU processing to be dedicated to static and dynamic reconstruction, respectively.

University of Manchester, Manchester, UK

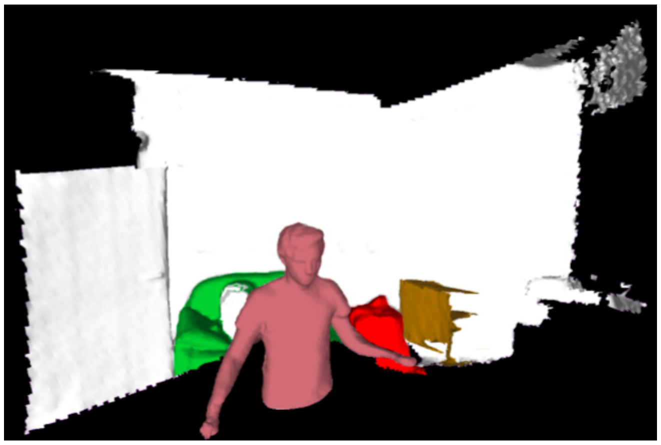


Fig. 1: Output of the system: 3D reconstruction including humans and semantically labelled objects - *e.g.* couch (green), table (brown).

Summing up our contributions, this paper presents ACEFusion: an expressive, robust, accurate, and efficient Spatial AI system with the following features:

1) A hybrid representation using octrees for static-semantic reconstruction on the CPU and surfels for dynamic reconstruction on the GPU.
2) An image processing pipeline running in parallel with the reconstruction tasks on dedicated accelerators, with results used to enhance downstream tasks.
3) The system runs in real time on an Nvidia Jetson AGX Xavier.
4) Matching or improving upon state-of-the-art results on pose estimation in dynamic environments.

	Static	Dynamic	Semantic	Representation	Low-power	Computing hardware	Optical flow
KinectFusion [6]	x			Voxel		GPU	
ElasticFusion [7]	x			Surfel		GPU	
InfiniTAM [8]	x			Voxel	x	CPU or GPU	
SuperEight [9]	x			Octree		CPU	
FlashFusion [10]	x			Octree		CPU	
DynamicFusion [11]		x		Voxel		GPU	
SurfelWarp [12]		x		Surfel		GPU	
SemanticFusion [13]	x		x	Surfel		GPU	
Kimera [14]	x		x	Mesh		CPU	
FlowFusion [15]	x			Voxel		GPU	x
SplitFusion [16]	x	x		Voxel		GPU	
FullFusion [17]	x	x	x	Voxel		GPU	
ACEFusion (ours)	x	x	x	Octree + Surfels	x	CPU + GPU + DLA + VIC	x

TABLE I: Comparison of state-of-the-art. Only ACEFusion is designed to run on a low-power platform suitable for robotics.

II. Related Work

What should an expressive representation of an environment contain? Broadly speaking, *form* and *meaning*. However, from a computational point of view, a finer distinction has to be made, one that accounts for the complexity of the processes involved in building cognitive maps. We briefly review prior work on acquiring such representations of the 3D environment in the context of robotics.

3D static geometry – The introduction of low-cost RGB-D cameras, such as the Microsoft Kinect and greater processing power of high-end GPUs, made the first real-time 3D reconstruction methods possible. Since KinectFusion [6], systems have improved in accuracy, efficiency, and robustness, enabling applications such as augmented reality or geometry-aware path planning [9]. For static geometry, two main approaches have been investigated: surfel based [7], [18], and volumetric [6], [19] (better for large scenes).

Prior art for efficient 3D reconstruction has used voxel hashing [20], [8], or octrees [21], [22] to reduce the computational and memory cost. The *static reconstruction module* of ACEFusion uses an octree-based representation, building upon the volumetric fusion methods of *supereight* [9].

Semantically-labelled geometry – Representing geometry may be sufficient for tasks such as path planning in a static setting, but seeing the world in terms of undifferentiated geometry can be severely limiting. Complex behaviour requires understanding the meaning of geometry; for instance, grasping a door handle or an object on a table [23], [24]. ScanComplete [25] uses semantic priors to fill in missing information in large-scale scenes. Advancements in deep learning, in particular for object detection, have facilitated the development of robotic systems with scene understanding capabilities. Garg *et al.* [26] presents a recent large-scale review.

Performing online semantic segmentation along with 3D reconstruction has been actively studied in the past few years, and approaches such as SemanticFusion [13] and CNN-SLAM [27] can reconstruct the geometry and semantics of static scenes in real time. SceneCode [28] recently introduced a code-based learned joint representation of scene semantics and geometry to perform monocular dense semantic reconstruction. Octree structures are suited for jointly representing semantics and geometry, and have been used for semantic mapping of indoor [29] and outdoor [30] scenes.

Dynamic geometry – In a static scene, the change between consecutive frames can be approximated by a single linear transformation corresponding to the camera motion. When objects in the scene move (in particular non-rigidly), observations are determined by multiple independent causal factors, introducing non-linearity in the problem of explaining changes in input. As non-linear problems are significantly more computationally-expensive than linear ones, various strategies to reduce the computational load have been proposed. Some treat moving objects as outliers and attempt to find reliable static landmarks [31], [32] or mask out [33], [34], [35], [15] dynamic elements.

Non-rigidly moving bodies are either agents or objects actively deformed by agents (*e.g.* clothing, tools). Since agents can produce changes when interacting with the environment, requiring robotic systems to adapt, foregoing the modelling of non-rigid objects would ignore some of the most important elements of a scene. ACEFusion performs real-time non-rigid reconstruction by improving upon SurfelWarp [12].

Static and dynamic geometry – A few recent works combine ideas from the approaches discussed above to capture scenes as completely as possible. The most similar systems to our work are SplitFusion [16] and FullFusion [17]. SplitFusion uses an instance segmentation neural network to split the input into rigid and non-rigid frames, then reconstructs the geometry. FullFusion uses a scene decomposition method based on semantics and geometry and additionally integrates semantic information into the scene representation.

Hardware-accelerated AI – A new class of embedded accelerators dedicated to deep learning inference and computer vision tasks has become available to the robotics community. The Nvidia Jetson series features Deep Learning Accelerators and Programmable Vision Accelerators. Recent work demonstrated the use of such accelerators can significantly speed up object detection [36], while FPGA accelerators [37] show great potential for energy-efficient SLAM. While it is common for 3D reconstruction algorithms to employ GPU acceleration, dedicated System-on-Chip with hardware accelerators targeted towards efficient image processing have not been used in parallel with GPU and multi-cores in the context of 3D reconstruction.

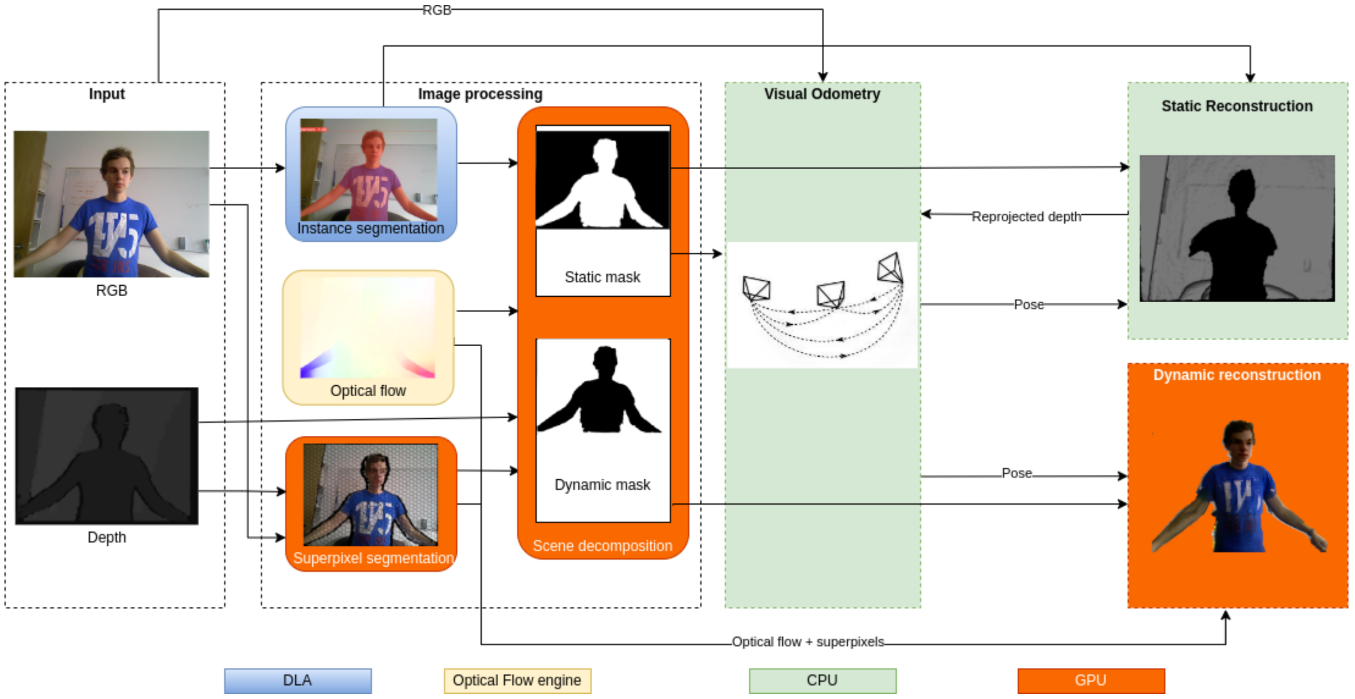


Fig. 2: Overview of our pipeline. The Image processing module pre-processes the input in parallel with the odometry and reconstruction nodes, preparing a new frame for processing while the reconstruction of the previous frame is being performed.

III. Method

A. Overview

A key contribution of the proposed method is the distribution of tasks across the available hardware units. We aim to finish preprocessing new inputs (in the Image Processing module), while the more demanding reconstruction tasks finish processing the previous frames. In addition, we seek to adopt the most useful principles from prior work while minimizing superfluous computation.

Our pipeline (Fig. 2) is described by four modules:

1) **Image processing**: extracts useful information about the input (optical flow, superpixels, instance segmentation) and decomposes the scene into static and dynamic parts.
2) **Pose estimation**: finds the transformation between the camera and world frame.
3) **Static reconstruction**: reconstructs a semantic 3D model of the scene using a volumetric representation.
4) **Dynamic reconstruction**: reconstructs 3D models of dynamic objects using a point-based representation.

B. Low-power hardware platform

Our method is optimised to use the hardware on Nvidia Jetson platforms for real-time, energy-efficient operation. The development and testing platform is an Nvidia Jetson Xavier AGX module running under 64-bit Ubuntu 18.04 OS. The module features an 8-core ARMv8.2 64-bit running at 2.25GHz, 16 GB of RAM, a 512-core Nvidia Volta GPU, 2 Deep Learning Accelerators (DLA), double Vision Accelerator, a Video Image Compositor (VIC) unit and an Optical Flow Accelerator. Due to limited on-board storage, the module is equipped with a Samsung 970 EVO Plus 1TB SSD installed via the high-speed M.2 Key-M connector through which all data is loaded.

C. Hardware-accelerated image processing

The image processing module extracts information that can be used to improve the accuracy and runtime of downstream tasks from the inputs and previous frames. The most important output computed by the module is a mask separating the static and dynamic elements of the input. All operations are performed in image space, taking advantage, when possible, of dedicated hardware. This creates two advantages. On one hand, the processing module is accelerated compared to executing the same operations on general-purpose hardware (processor/GPU). On the other hand, this frees up the processors and GPU to execute other tasks in parallel. The input to the pipeline is a pair of registered RGB-D frames $F = \{\mathcal{C} : \Omega \mapsto \mathbb{N}^3, \mathcal{D} : \Omega \mapsto \mathbb{N}\}$ and a previously stored keyframe $F^{kf} = \{\mathcal{C}^{kf}, \mathcal{D}^{kf}\}$ with $\Omega \subset \mathbb{N}^2$ the pixel space.

Semantic segmentation – We propose to use DNN-based models to compute a semantic segmentation of the inputs $\mathcal{I} : \Omega \mapsto P(\mathcal{L} = l)$ with each pixel encoding a probability distribution over the set of labels $\mathcal{L}$. Recognized classes are assigned *a priori* to static (e.g. *table*) or dynamic (e.g. *person*) categories and used in segmenting the frame into static and dynamic regions.

Feature matching and optical flow – Sparse features are used to obtain a camera pose estimate and subsequently for frame warping and visual odometry. First, ORB features are extracted and classified as static or dynamic using $\mathcal{MI}$. The static features are matched against a previously-created local map of static features associated with the current keyframe and the affine homography 2x2 matrix H relating F^{kf} to F in the image plane is computed using RANSAC. The keyframe F^{kf} is warped into the current frame using the onboard VIC such that the resulting F_{warp} is approximately aligned with F. Any significant differences between F^{warp} and F are independent of camera motion and thus can be exploited to segment dynamic objects. These differences can be quantified using the dense optical flow $\mathcal{OF}$ between $\mathcal{C}_{warp}$ and $\mathcal{C}$, and the depth difference $\mathcal{D}_{Diff} = \|\mathcal{D}_{warp} - \mathcal{D}\|$. The reprojection error can be classified as static/dynamic based on the reprojected distance similar to DynaSLAM [38].

$$\begin{gathered} \Delta_z = z_{proj} - z \\ \Delta_z > \theta_z \end{gathered} \tag{1}$$

Superpixel RGB-D clustering – Superpixels are continuous and non-overlapping clusters of pixels tiling an image, each defined as $\mathfrak{s} = \{u_{0..m} | u_i \in \Omega\}$. We use an efficient GPU implementation provided by NVidia to segment each new input F into superpixels which maximize depth, orientation, and color consistency. Henceforth, operations are performed on the resulting set of superpixels S rather than on individual pixels, improving accuracy and efficiency of further computation.

Masking – The final step of the image processing pipeline concerns computing binary masks to separate the static and dynamic parts of the input. The masks can be used to filter inputs to other tasks by element-wise multiplication, e.g. $\mathcal{C}_{stat} = \mathcal{C} \odot \mathcal{M}_{stat}$. The above operations (clustering, optical flow, and semantic segmentation) are independent and computed in parallel. The mask of dynamic objects is obtained in two steps: firstly, an initial dynamic mask $\mathcal{M}_{init} : \Omega \mapsto \{0, 1\}$ is obtained using the previously computed $\mathcal{M}_{\mathcal{I}}$ by classifying a superpixel $\mathfrak{s}$ as dynamic if the dominant label of the superpixel is a dynamic class or the mean optical flow magnitude exceeds the threshold μ_{OF}^{high}.

$$\begin{gathered} \delta(x, y) = 1 \Leftrightarrow x = y \\ MaxLabel(\mathcal{I}, \mathfrak{s}, L) = \underset{l \in L}{\arg\max} \sum_{u \in \mathfrak{s}} \delta(\mathcal{I}(u), l) \\ \mathcal{M}_{\mathcal{I}} = \{MaxLabel(\mathcal{I}, \mathfrak{s}, L) \in L_{dyn} | \mathfrak{s} \in S, u \in \mathfrak{s}\} \\ \mathcal{M}_{\mathcal{OF}} = \bigcup_{\mathfrak{s} \in S} \|\bar{\mathfrak{s}}_{\mathcal{OF}}\|_2^2 > \mu_{OF}^{high} \\ \mathcal{M}_{init} = \mathcal{M}_{\mathcal{OF}} \cup \mathcal{M}_{\mathcal{I}} \end{gathered} \tag{2}$$

The initial mask is then extended using a flood-fill method (see Algorithm 1) relying on depth to select adjacent superpixels and relying on residuals computed from per-superpixel statistics. These include the average depth difference, flow magnitude, as well as distance to the closest dynamic keypoint.

Algorithm 1: Generating $\mathcal{M}_{dyn}$ using flood fill.

```
Input  : Initial mask $\mathcal{M}_{init}$, Superpixel stats $S_{stat}$
Output: Binary mask $\mathcal{M}_{dyn}$
Let queue $\mathcal{Q}$;
Let $\mathcal{N}(\mathfrak{s})$ the set of neighbors of superpixel $\mathfrak{s}$;
Let $\mathcal{M}_{dyn} = 0$ a binary-valued image;

Add all superpixels in $\mathcal{M}_{init}$ to $\mathcal{Q}$;
foreach $\mathfrak{s} \in \mathcal{Q}$ do
    foreach $\mathbf{n} \in \mathcal{N}(\mathfrak{s})$ do
        if $\mathbf{n} \in \mathcal{M}_{init}$ or $\mathbf{n} \in \mathcal{M}_{dyn}$ or $\mathbf{n} \in \mathcal{Q}$ then
            continue;
        end
        if $||\mathcal{D}(\mathbf{p}) - \mathcal{D}(\mathbf{n})|| < \theta_D \cdot D(\mathbf{p})$ then
            Add $\mathbf{n}$ to $\mathcal{Q}$;
            if $R(\mathbf{n}) > \theta_R$ then
                $\mathcal{M}_{dyn}(\mathbf{n}) = 1$;
            end
        end
    end
    Remove $\mathfrak{s}$ from $\mathcal{Q}$;
end
return $\mathcal{M}_{init} \cup \mathcal{M}dyn$
```

$$\begin{gathered} R(\mathfrak{s}) = \lambda_{OF} R_{OF} + \lambda_{depth} R_{depth} + \lambda_{feat} R_{feat} \\ R_{OF} = \|\bar{\mathfrak{s}}_{\mathcal{OF}}\|_2^2 \\ R_{depth} = \bar{\mathfrak{s}}_{\mathcal{D}_{Diff}} \\ R_{feat} = min(dist(\mathfrak{s}, M_{dyn})) \end{gathered} \tag{3}$$

The static mask $\mathcal{M}_{stat} = \neg \mathcal{M}_{dyn}$ is obtained by inverting the dynamic mask $\mathcal{M}_{dyn}$. Finally, the refined masks are used to reclassify the extracted features and improve masking and pose estimation in subsequent frames.

Additional optimizations – We can exploit the hardware of the Nvidia Jetson AGX platform to speed up and parallelize computation. The DLA is an open-source hardware design that accelerates convolutional neural network operations. Using the TensorRT SDK which provides inference optimisation functions including mixed-precision support and kernel specialization, we employ the lightweight YOLACT Edge [39] semantic segmentation method trained on the MS COCO dataset. To ensure that any dynamic objects are filtered, we use two thresholds: static objects require a high prediction confidence ($> 70\%$), while for dynamic objects we use a 30% threshold. This facilitates noise reduction in the semantic mapping of the static scene, while ensuring that any dynamic objects are segmented out. Optical flow is computed using the onboard Optical Flow Engine. To compensate for the reduced quality compared to CNN-based methods, we use the last 3 keyframes rather than a single keyframe, and take the maximum flow for each superpixel.

D. *Pose estimation*

The pose estimation module estimates the camera pose $\mathbf{T} = \{t \in \mathbb{R}^3, R \in \mathbb{SO}(3)\} \in \mathbb{SE}(3)$, defined as a rigid transformation between the camera coordinate frame to the world coordinate frame. The input to the module is a set of matched keypoints classified as static by the image processing module using the initial mask M_{init} and the geometric threshold from eq. 1. While using the final mask $\mathcal{M}_{dyn}$ We use Levenberg–Marquardt least-squares optimisation to minimize the tracking loss E_{track}.

$$\begin{gathered} R,t = \underset{R,t}{\operatorname{argmin}} E_{track} \\ E_{track} = \lambda_{sparse}E_{sparse} + \lambda_{dense}E_{dense} \end{gathered} \tag{4}$$

To maximise efficiency, only the sparse term is used between keyframes. $\mathbf{T}$ is computed by minimizing the reprojection error between matched keypoints $M = \{(x, x^{kf})_i\}$ between the keyframe and the frame. Balancing the accuracy-efficiency tradeoff, both the sparse and the dense terms are using when computing the pose of each keyframe. E_{dense} is the frame-to-model ICP registration between the projected static depth frame $\mathbf{V} = \pi^{-1}(\mathcal{D}_{stat})$ and the reprojection of the static reconstruction into the previous keyframe, similar to the tracking stage of *supereight* [9].

$$\begin{gathered} E_{sparse} = \sum_{(x,x^{kf})\in M} \rho\left(\left\|x - \pi\left(Rx^{kf} + t\right)\right\|_2^2\right) \\ E_{dense} = \sum_{\mathbf{u}\in\Omega} \left\|\left(\hat{\mathbf{V}}^{kf}(\hat{\mathbf{u}}) - \pi\left(R\mathbf{V}(\mathbf{u}) + t\right)\right)^T \cdot \hat{\mathbf{N}}^{kf}(\hat{\mathbf{u}})\right\|_2^2 \\ \hat{\mathbf{u}} = \pi\left(\mathbf{T}_{k-1}^{-1}\mathbf{T}_k\pi^{-1}(\mathbf{u})\right) \end{gathered} \tag{5}$$

E. CPU-based static-semantic reconstruction

Given RGB-D static pair, the estimated pose, and the semantic segmentation, we aim to fuse the data over all timesteps $k \in \{0..N\}$ into a semantically-labelled 3D representation of the static scene. We build upon *supereight*'s multi-resolution occupancy mapping which leverages octrees [40], [9]. The *supereight* pipeline contains 3 processing stages: *tracking*, *integration*, and *rendering*. We use tracking and rendering only on keyframes for use in pose estimation as described in the previous section. The integration stage is where the sensor data is fused into the dense static model using an adaptive-resolution octree. We modify the octree to include a probability distribution over the set of semantic labels $P(L_{\mathbf{x}} = l_i), l_i \in \mathcal{L}$ on each leaf node $\mathbf{x}$ initialized with the uniform probability distribution. The integration stage includes three processes: data fusion (single layer update), upward propagation, and downward propagation. Data fusion allocates new nodes and if necessary, updates the values via downward propagation.

Downward propagation is an expensive step in the pipeline, used when the camera moves closer to the surface and allows for mapping at a finer resolution. Since camera movement is generally smooth, performing this step at every frame is redundant. We store the frame with the smallest distance to the camera in a cache, along with its pose and only perform the down propagation step on the cached frame. Since the cached frame allows us the best resolution for fusion, the labels are also updated in this step via bayesian recursive update:

$$P\left(L_{\mathbf{x}} \mid F_{0,\dots,k}\right) = P\left(L_{\mathbf{x}} \mid F_{0,\dots,k-1}\right) P\left(\mathcal{I}\left(\hat{\mathbf{x}}\right) = l_i \mid F_k\right)$$

Where $\mathcal{I}(\hat{\mathbf{x}})$ is the pixel value of the node $\mathbf{x}$ reprojected into the semantic frame.

F. GPU-based dynamic reconstruction

Problem definition – Given the camera pose T_{WC}, dynamic RGB-D frame $\{\mathcal{C}_{dyn}, \mathcal{D}_{dyn}\}$, Optical Flow $\mathcal{OF}_{dyn}$, and superpixels S at every time step, the goal is to fuse the data into a non-rigidly deforming 3D model $\mathcal{P}_{ref}$. In contrast with volumetric methods [11], [41], [16] which need to convert between implicit and explicit representations to register incoming data with the 3D model, we build upon the efficient point-based representation of SurfelWarp [12], which can be directly registered with new input.

The reconstruction pipeline can be summarized as follows: given a new input, a warp function $\mathcal{W}$ is estimated to non-rigidly deform the reference model into the live frame. Once the two are aligned, the warped reference model is projected into the camera frame and corresponding pixels are found. Finally, data fusion is performed to update the reference model with the new data. This paper will focus on the optimisations we have developed in warp field estimation and filtering the input to the pipeline. The details of alignment and data fusion are described in SurfelWarp [12].

Warp field estimation – The reference and live models $\mathcal{P}_{ref}, \mathcal{P}_{live}$ consist of arrays of point and normal pairs (surfels) $\mathcal{P}_{(\cdot)}\{v_i, n_i\}$. Additionally, the warp function $\mathcal{W}(p)$ approximates the transformation for each pair $\mathcal{P}_{ref}(i)$ into the live frame necessary for integrating the new information into the model. The warp function has an associated embedded graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ defined by nodes $\mathcal{N} = \{[t \in \mathbb{R}^3, w \in \mathbb{R}^+, \mathbf{T} \in \mathbb{SE}(3)]\}$ with position *t*, weight *w* controlling the influence of a node and transformation $\mathbf{T}$, as well as edges $\mathcal{E} = [\mathcal{N}_i, \mathcal{N}_j]$. The node graph is initialized by subsampling surfel positions at the first frame and extended with new nodes when needed in order to cover the full extent of the dynamic model.

Energy terms – We solve the underlying non-rigid iterative closest point (ICP) using a GPU-accelerated preconditioned conjugate gradient descent Gauss-Newton solver to estimate the warp function parameters to that minimize the error between $\mathcal{P}_{ref}$ and $\mathcal{P}_{live}$.

$$\begin{gathered} \mathcal{W}_t = \arg\min E_{tot}(\mathcal{W}_{t-1}, \mathcal{P}_{ref}, \mathcal{P}_{live}) \\ E_{tot} = \lambda_{depth}E_{depth} + \lambda_{ARAP}E_{ARAP} + \lambda_{corr}E_{corr} \end{gathered} \tag{6}$$

The energy function contains the following terms:

A point-to-plane depth energy term. A surfel correspondence map C between $\mathcal{P}_{ref}$ and $\mathcal{P}_{live}$ is computed according to SurfelWarp, pairing the warped model surfels $\hat{p}_{ref}$ to the live model ones p_{live}. The loss is given by the distance between the vertices, with the orientation n_{live}.

$$\begin{aligned} \hat{p}_{ref} &= \mathbf{T}^{-1} * \mathcal{W}(p_{ref}) \\ \mathbf{C} &= \Big\{ \mathbf{v}_i, \mathbf{n}_i | \mathbf{v}_i = \{\hat{v}_{ref}, v_{live}\}, \mathbf{n}_i = \{\hat{n}_{ref}, n_{live}\} \Big\} \\ E_{depth}(\mathcal{W}) &= \sum_{\{\mathbf{v}_i, \mathbf{n}_i\} \in \mathbf{C}} \left\| (n_{live}^T (\hat{v}_{ref} - v_{live}) \right\|_2^2 \end{aligned} \tag{7}$$

We adopt the sparse correspondence energy term from [42] using Global Patch Collider (GPC) [43] to match sparse feature patches. The reasoning behind using this term is that the sparse features are likely to reflect the general movement of the dynamic model, and as such it leads to faster convergence of E_{tot}. Similarly to the depth loss we use the L2 norm, in this case applied to the set of matched features $\mathbf{q}$ with vertices in the live model.

$$\begin{aligned} \mathbf{M} &= \{\mathbf{q}, v_{live}\} \\ \hat{\mathbf{q}} &= \mathbf{T}^{-1} * \mathcal{W}(\mathbf{q}) \\ E_{corr}(\mathcal{W}) &= \sum_{m \in \mathbf{M}} \rho \left(\|\hat{\mathbf{q}} - v_{live}\|_2^2 \right) \end{aligned} \tag{8}$$

Finally, an as-rigid-as-possible (ARAP) regularization term over all pair-wise connected nodes in the graph associated with the warp field. This has the effect of smoothing the optimisation problem by "pulling" the graph nodes together and prevents occlusions from breaking the model.

$$E_{ARAP}(\mathcal{W}) = \sum_{i \in \mathcal{G}} \sum_{j \in \mathcal{E}(i)} \|\mathbf{T}_j t_j - \mathbf{T}_i t_j\|_2^2 \tag{9}$$

Additional optimizations – Even under an efficient formulation, non-rigid reconstruction is the most computationally-expensive process in our pipeline. Thus we introduce several high-level optimisations which minimally impact reconstruction quality. Frame skipping is a simple heuristic used widely to improve the speed of reconstruction algorithms, however a naive approach using a fixed hyperparameter for frame skipping may cause reconstruction to fail, as scene motion cannot be known a priori. Instead, use the average flow of each superpixel to filter out those below the threshold μ_{OF}^{dyn}, thus requiring the warp field estimation to compute less parameters for every frame. The filtering step is skipped for keyframes as well as every $N_{full} = 5$ frames in order to prevent error accumulation.

IV. Experiments

All software and dependencies are compiled with gcc-7 and CUDA 10.2, using the highest level of optimisation, including platform-specific optimisations. The static reconstruction module uses OpenMP to distribute the load across 6 of the available 8 cores. One extra core is dedicated to scheduling, while the final core is left for operating system processes. We use the `jetson-stats` tool[1] to disable DVFS and lock all hardware clocks to the maximum frequency afforded for a given power budget (30W). In the following experiments we use $\lambda_{sparse} = 0.1, \lambda_{dense} = 0.9$ to estimate the pose for keyframes (see eq. 4).

[1]https://github.com/rbonghi/jetson_stats

	Ours	RE	SF	DS
sitting_static	0.009	0.011	0.014	**0.007**
sitting_xyz	0.021	0.026	0.039	**0.015**
sitting_halfsphere	0.035	0.038	0.041	**0.028**
walking_static	0.011	0.014	0.015	**0.007**
walking_xyz	0.025	0.074	0.093	**0.017**
walking_halfsphere	0.035	0.048	0.681	**0.026**

TABLE II: ATE-RMSE(m) on the TUM RGB-D dataset dynamic sequences.
Re = ReFusion, SF = StaticFusion, DS = DynaSLAM

	Ours	RE	SF	DS
balloon	**0.028**	0.175	0.233	0.030
balloon2	0.030	0.254	0.293	**0.029**
balloon_tracking	**0.045**	0.302	0.221	0.049
balloon_tracking2	**0.033**	0.322	0.366	0.035
crowd	**0.016**	0.204	3.586	**0.016**
crowd2	**0.027**	0.155	0.215	0.031
crowd3	**0.023**	0.137	0.168	0.038
kidnapping_box	0.030	0.148	0.336	**0.029**
kidnapping_box2	**0.030**	0.161	0.263	0.035
moving_no_box	**0.070**	0.071	0.141	0.232
moving_no_box2	**0.029**	0.179	0.364	0.039
moving_o_box	0.343	0.343	0.331	**0.044**
moving_o_box2	0.443	0.528	0.309	**0.263**
person_tracking	0.070	0.289	0.484	**0.061**
person_tracking2	**0.071**	0.463	0.626	0.078
placing_no_box	**0.088**	0.106	0.125	0.575
placing_no_box2	**0.020**	0.141	0.177	0.021
placing_no_box3	**0.051**	0.174	0.256	0.058
placing_o_box	0.324	0.571	0.330	**0.255**
removing_no_box	0.020	0.041	0.136	**0.016**
removing_no_box2	0.025	0.111	0.129	**0.021**
removing_o_box	0.314	**0.222**	0.334	0.291
synchronous	**0.014**	0.441	0.446	0.015
synchronous2	0.010	0.022	0.027	**0.009**

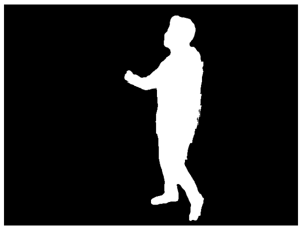
TABLE III: ATE-RMSE(m) on the Bonn Dynamic dataset

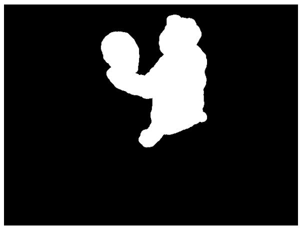
A. *Pose estimation*

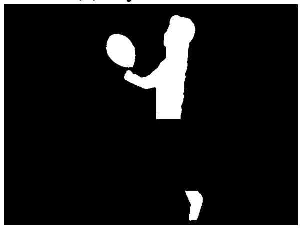
We use the dynamic sequences of the TUM RGB-D dataset [44] to evaluate the accuracy of our pose estimation against previous SLAM systems designed for dynamic environments:

- StaticFusion, a surfel-based approach which uses pose registration residuals between the reprojected 3D model and the most recent frame to segment dynamic elements. Runs below 15 frames per second (FPS).
- ReFusion, a voxel-based approach using a similar registration residual based segmentation. Runs at approx. 0.5 FPS.
- DynaSLAM, a sparse SLAM system which uses a joint geometric and semantic approach to dynamic object segmentation. Runs below 10 FPS.

The results of the experiments, presented in Tables II and III, show that our performance surpasses dense approaches such as ReFusion and StaticFusion, and often performs better than sparse ones, such as DynaSLAM, while building a dense semantic 3D model. Our system is the only one running in real time on the power constrained device, thanks to the heterogeneous computation afforded by the hardware. We note that the worst performance for our system is on


(a) DynaSLAM


(b) ReFusion


(c) Ours (only optical flow)

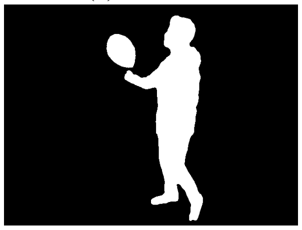

(d) Ours (final)

Fig. 3: Qualitative comparison of masking

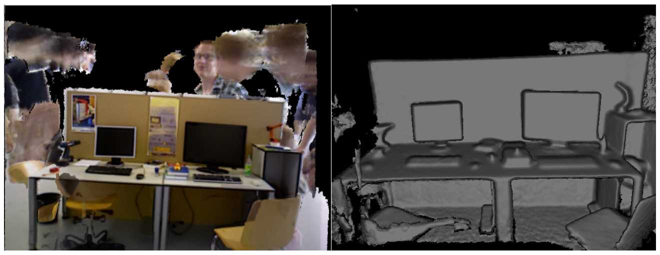
"obstruction" sequences of the Bonn dataset, where a large box covers the frame for extended periods, the deterioration in performance suggesting that our system is prone to "the kidnapping problem".

B. Masking

We compare the quality of masking with that produced by competing methods, such as ReFusion and DynaSLAM in Fig. 3. Note that the final mask produced by our system is sharp and includes both objects recognised as dynamic by the semantic module, as well as other moving objects (in this, case a balloon), thanks to the flow-based segmentation. In contrast, DynaSLAM misses the balloon, as it is neither recognized by their semantic module nor does it have enough features for the multi-view geometry module to classify as dynamic. Meanwhile, ReFusion typically oversegments or undersegments when computing the mask, and always produces dilated masks, exceeding the boundary of the object. This may be acceptable when non-rigidly deforming objects are not being reconstructed, but in our case could negatively impact the dynamic reconstruction.

C. 3D reconstruction

Our final static 3D reconstruction output is similar to ReFusion, however during operation there are differences that may prove important. Detecting objects known *a priori* to be dynamic allows us to segment them out before they move, thus minimizing artefacts not only in the final model, but during operation as well. Figure 4a demonstrates the difference on a sequence from the TUM RGB-D dataset. Both systems are able to produce clean final outputs, removing most of the artefacts produced, however ACEFusion creates fewer artefacts to begin with.


(a) ReFusion (left) vs ACEFusion reconstructions running on the *walking_xyz* sequence of the TUM RGB-D dataset.

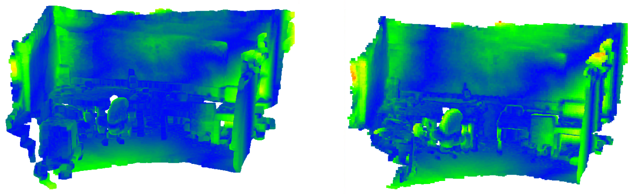

(b) Comparison of the reconstruction error of ReFusion (left) and ACEFusion (right) on the *crowd3* sequence of the Bonn RGB-D dataset.

Fig. 4: Qualitative evaluation of 3D reconstruction

V. Conclusions

We have presented ACEFusion, an efficient and accurate Spatial AI system. We achieve comparable accuracy to sparse SLAM systems, such as DynaSLAM, while running in real time on a low-power platform and solving a more complex problem than competing systems; ACEFusion builds dense semantically-labelled models of static and non-rigidly deforming geometry.

Whereas other SLAM systems typically use a single representation, ACEFusion adopts a surfel-based representation for dynamic objects, while the static background is modelled using octrees. While prior art requires powerful desktop GPUs to achieve real-time performance, ACEFusion is the first to distribute the computational load of Spatial AI system across multiple hardware accelerators: the Deep Learning Accelerators (DLA), Video Image Compositor (VIC), general purpose processing cores, and GPU available on the Nvidia Jetson Xavier AGX board.

The experiments have showed that the performance ACEFusion surpasses dense approaches, such as ReFusion and StaticFusion, and often performs better than sparse ones, such as DynaSLAM, while building a dense semantic 3D model. Our system is the only one running in real time on the power constrained device. We have also compared the quality of masking with that produced by ReFusion and DynaSLAM. The final masks produced by ACEFusion are sharp and includes multiple dynamic moving objects thanks to the flow-based segmentation.

Acknowledgments

This research is supported by the EPSRC, grant RAIN Hub EP/R026084/1. Mikel Luján is supported by an Arm/RAEng Research Chair Award and a Royal Society Wolfson Fellowship.

References

[1] A. J. Davison, "FutureMapping: The computational structure of spatial AI systems," *arXiv preprint arXiv:1803.11288*, 2018.

[2] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. Kelly, A. J. Davison, M. Luján, M. F. O'Boyle, G. Riley *et al.*, "Introducing slambench, a performance and accuracy benchmarking methodology for slam," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5783–5790.

[3] B. Bodin, H. Wagstaff, S. Saeedi, L. Nardi, E. Vespa, J. H. Mayer, A. Nisbet, M. Luján, S. Furber, A. J. Davison *et al.*, "SLAMBench2: Multi-objective head-to-head benchmarking for visual SLAM," *arXiv preprint arXiv:1808.06820*, 2018.

[4] M. Bujanca, P. Gafton, S. Saeedi, A. Nisbet, B. Bodin, F. O'Boyle Michael, A. J. Davison, G. Riley, B. Lennox, M. Luján, and S. Furber, "SLAMBench 3.0: Systematic automated reproducible evaluation of SLAM systems for robot vision challenges and scene understanding," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6351–6358.

[5] M. Bujanca, X. Shi, M. Spear, P. Zhao, B. Lennox, and M. Luján, "Robust SLAM Systems: Are We There Yet?" in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5320–5327.

[6] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2011, pp. 127–136.

[7] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense slam without a pose graph," *Proc. Robotics: Science and Systems, Rome, Italy*, 2015.

[8] O. Kahler, P. V. Adrian, R. C. Yuheng, X. Sun, P. Torr, and D. Murray, "Very high frame rate volumetric integration of depth images on mobile devices." *IEEE transactions on visualization and computer graphics*, vol. 21, no. 11, pp. 1241–1250, 2015.

[9] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. Kelly, and S. Leutenegger, "Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1144–1151, 2018.

[10] L. Han and L. Fang, "FlashFusion: Real-time Globally Consistent Dense 3D Reconstruction using CPU Computing."

[11] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *CVPR*, 2015.

[12] W. Gao and R. Tedrake, "Surfelwarp: Efficient non-volumetric single view dynamic reconstruction."

[13] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," *arXiv preprint arXiv:1609.05130*, 2016.

[14] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020.

[15] T. Zhang, H. Zhang, Y. Li, Y. Nakamura, and L. Zhang, "Flowfusion: Dynamic dense rgb-d slam based on optical flow," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[16] Y. Li, T. Zhang, Y. Nakamura, and T. Harada, "Splitfusion: Simultaneous tracking and mapping for non-rigid scenes," *arXiv preprint arXiv:2007.02108*, 2020.

[17] M. Bujanca, M. Luján, and B. Lennox, "FullFusion: A Framework for Semantic Reconstruction of Dynamic Scenes," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[18] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: Online surfel-based mesh reconstruction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2494–2507, 2019.

[19] V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. Torr, and D. W. Murray, "Infinitam v3: A framework for large-scale 3d reconstruction with loop closure," *arXiv preprint arXiv:1708.00783*, 2017.

[20] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 169, 2013.

[21] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3d mapping in real-time on a cpu," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2021–2028.

[22] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, 2013.

[23] N. Blodow, L. C. Goron, Z.-C. Marton, D. Pangercic, T. Rühr, M. Tenorth, and M. Beetz, "Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 4263–4270.

[24] H. Dang and P. K. Allen, "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1311–1317.

[25] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner, "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018.

[26] S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T.-J. Chin, I. Reid, S. Gould *et al.*, "Semantics for robotic mapping, perception and interaction: A survey," *arXiv preprint arXiv:2101.00443*, 2021.

[27] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction."

[28] S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison, "Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 776–11 785.

[29] J. Stückler, B. Waldvogel, H. Schulz, and S. Behnke, "Dense real-time mapping of object-class semantics from rgb-d video," *Journal of Real-Time Image Processing*, vol. 10, no. 4, pp. 599–609, 2015.

[30] S. Sengupta and P. Sturgess, "Semantic octree: Unifying recognition, reconstruction and representation via an octree constrained higher order mrf," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1874–1879.

[31] D. Li, X. Shi, Q. Long, S. Liu, W. Yang, F. Wang, Q. Wei, and F. Qiao, "Dxslam: A robust and efficient visual slam system with deep features," *arXiv preprint arXiv:2008.05416*, 2020.

[32] W. Dai, Y. Zhang, P. Li, and Z. Fang, "Rgb-d slam in dynamic environments using points correlations," *arXiv:1811.03217*, 2018.

[33] T. Zhang and Y. Nakamura, "Posefusion: Dense rgb-d slam in dynamic human environments," in *2018 International Symposium on Experimental Robotics*, 2018.

[34] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "Staticfusion: background reconstruction for dense rgb-d slam in dynamic environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[35] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss, "ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals," *arXiv*, 2019.

[36] Y. Qian, H. Zheng, D. He, Z. Zhang, and Z. Zhang, "R-cnn object detection inference with deep learning accelerator," in *2018 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*. IEEE, 2018, pp. 297–302.

[37] R. Liu, J. Yang, Y. Chen, and W. Zhao, "eslam: An energy-efficient accelerator for real-time orb-slam on fpga platform," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019.

[38] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[39] H. Liu, R. A. R. Soto, F. Xiao, and Y. J. Lee, "Yolactedge: Real-time instance segmentation on the edge (jetson agx xavier: 30 fps, rtx 2080 ti: 170 fps)," *arXiv preprint arXiv:2012.12259*, 2020.

[40] E. Vespa, N. Funk, P. H. Kelly, and S. Leutenegger, "Adaptive-resolution octree-based volumetric slam," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 654–662.

[41] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, "Volumedeform: Real-time volumetric non-rigid reconstruction," *arXiv preprint arXiv:1603.08161*, 2016.

[42] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi, "Fusion4d: Real-time performance capture of challenging scenes," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 114:1–114:13, Jul. 2016.

[43] S. Wang, S. Ryan Fanello, C. Rhemann, S. Izadi, and P. Kohli, "The global patch collider," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 127–135.

[44] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.