A Methodology for Principled Approximation in Visual SLAM

Yan Pei
University of Texas at Austin
Austin, USA
ypei@cs.utexas.edu

Swarnendu Biswas
Indian Institute of Technology Kanpur
Kanpur, India
swarnendu@cse.iitk.ac.in

Donald S. Fussell
University of Texas at Austin
Austin, USA
fussell@cs.utexas.edu

Keshav Pingali
University of Texas at Austin
Austin, USA
pingali@cs.utexas.edu

ABSTRACT

This paper proposes a methodology for exploiting approximate computing to reduce the time and energy requirements of *Simultaneous Localization and Mapping* (SLAM) algorithms, which are used in important problem domains like robotics and autonomous driving in which autonomous agents navigate through unknown environments. Algorithms for SLAM use sensors to probe the environment, integrate this information into a map of the surroundings (mapping), and determine where the agent is in this map (localization). *Visual* SLAM algorithms use cameras as sensors. They can be used in places where GPS information is not available, but they have high computational requirements, leading to poor performance and high energy usage on embedded platforms.

Existing studies of approximation in SLAM have mostly used offline control, which requires the trajectory be known before the agent starts to move. This is not realistic in most SLAM applications. In this paper, we present a general methodology for applying principled *online* approximation to visual SLAM algorithms. We implemented our proposed methodology in four visual SLAM algorithms (including one visual inertial SLAM algorithm) and evaluated them on several platforms. Our experimental results show that across different algorithms and platforms, our proposed methodology results in savings of up to 77% and 40% in computation time and energy consumption respectively with acceptable quality loss in localization and mapping accuracy over a variety of inputs.

CCS CONCEPTS

• **Computing methodologies** → **Approximate computing**; **Control methods**; *Computer vision problem.*

KEYWORDS

Approximate computing, SLAM, online control, ElasticFusion, KinectFusion, ORB-SLAM2, ICE-BA

PACT '20, October 3–7, 2020, Virtual Event, GA, USA

ACM ISBN 978-1-4503-8075-1/20/10…$15.00
https://doi.org/10.1145/3410463.3414636

ACM Reference Format:
Yan Pei, Swarnendu Biswas, Donald S. Fussell, and Keshav Pingali. 2020. A Methodology for Principled Approximation in Visual SLAM. In *Proceedings of the 2020 International Conference on Parallel Architectures and Compilation Techniques (PACT '20), October 3–7, 2020, Virtual Event, GA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3410463.3414636

1 INTRODUCTION

Approximate computing can be used to reduce the time and energy required to execute certain kinds of programs. Abstractly, these programs have a number of parameters, usually called *knobs*, that can be tuned to control the quality of the approximation and the running time or energy consumption of the program for a given input. For example, in iterative linear solvers, the "number of iterations" is a knob that can be dialed up to increase the quality of the solution at the cost of increased running time and energy [2].

Strategies for introducing approximation into program execution include skipping loop iterations [56], randomly discarding tasks [48], and relaxing synchronization operations in parallel programs [13, 49]. In the programming languages community, there has been work on identifying program patterns that are amenable to approximation [53], and providing language-level support for approximation through program analyses and transformations [8, 14, 15, 43, 54, 55]. These studies have largely focused on studying opportunities for exploiting approximate computing without providing guarantees on output quality.

To control approximation in a principled way, it is necessary to have a *quality estimator* (in the context of finite-element methods, this is called an *error estimator* [35]). An *a priori* estimator provides an estimate of output quality for given input and knob settings without running the program, and it can be used to perform *proactive control* in which knobs are set optimally for a given input before execution. For example, Xin *et al.* use machine learning to build inexpensive *proxy models* for *a priori* estimation of the quality and running time of a program, and they use these models together with features of the input to perform proactive control of approximation in programs such as hierarchical graph partitioners [58].

In contrast, an *a posteriori* quality estimator provides an estimate of output quality for a given input and knob settings *after* the program is executed. *A posteriori* estimators are useful for streaming programs in which the system is presented with a stream of inputs and it can be assumed that the output quality and run-time behavior of the program for successive inputs is stationary. In such programs, the *a posteriori* error estimate obtained after processing

an input can be used to tune knobs for the next input, an approach which is called *reactive online control*. *A posteriori* error estimation is straightforward if the *ground truth*, *i.e.*, the output produced without making any approximation, is available. For example, the Green system periodically executes the exact version of the program in parallel with the approximate version to determine the *a posteriori* error for a given input, and tunes knobs based on this for the next epoch [1]. Online control has been used to reduce the energy requirements of programs by clever task scheduling on heterogeneous processors and to exploit DVFS [45, 46, 61].

In this paper, we consider the problem of principled control of approximation in Simultaneous Localization and Mapping (SLAM) algorithms. SLAM solves the problem of determining the *poses* (position and orientation) of an autonomous agent in an unknown environment while constructing a map of that environment [7, 9, 17, 18, 20, 26, 27, 30, 33, 34, 37, 40, 47, 59, 63, 64]. It is used widely in robotics, autonomous vehicles, and computer vision. Visual SLAM has become popular because of the ubiquity of affordable cameras in the embedded and mobile spaces [21, 52]. However, the algorithms have high computational costs given their reliance on compute-intensive kernels such as bundle adjustment [60], iterative closest point [3], and loop closure detection [23]. These high computational requirements lead to poor performance such as low frame rates and limited battery life, resulting in suboptimal user experience and presenting a practical barrier to deploying SLAM algorithms on embedded and resource-constrained devices.

One solution is approximate computing but introducing approximation into visual SLAM algorithms is difficult for several reasons.

- In problems like n-body methods and graph partitioners, the input to the program is available at the beginning of the computation so it can be inspected to determine how to exploit approximation. In SLAM, the agent must react to an unknown environment while navigating through it.
- Knob settings should be changed dynamically during navigation to obtain the best performance. For example, in scenes in which there are many landmarks, navigation is simple so more approximation can be used; conversely, more accurate computation might be advisable when there are few landmarks.
- Problems for which online control has been used successfully such as DVFS can be viewed abstractly as *unconstrained* optimization problems. In contrast, introducing approximation into SLAM is a *constrained optimization* problem since approximation must not prevent the agent from navigating through the environment.
- Online approximation controllers such as Rumba [29] assume that if the error is unacceptable, the controller can back up and redo the computation with less approximation. Re-computation is not allowed for SLAM because it is used for real-time control such as autonomous vehicle navigation where failure is not an option.
- The exact pose of the agent (ground truth) can never be determined exactly so ground truth cannot be used to come up with *a posteriori* error estimates.

In this paper, we propose a general methodology to introduce principled online approximation in visual SLAM algorithms to trade-off computation time and energy consumption for an acceptable loss in accuracy. Our proposed methodology controls knobs in a visual SLAM algorithm in an online fashion during the motion of the autonomous agent without requiring *a priori* knowledge of the environment and trajectory. We implemented the proposed methodology in four popular visual (or visual inertial[1]) SLAM algorithms — *ElasticFusion* [63, 65], *KinectFusion* [26, 40], *ORB-SLAM2* [37, 38] and *ICE-BA* [34] — which are widely used in this area. We evaluated our methodology on a variety of real-world inputs on two platforms, a heterogeneous ODROID-XU4 big.LITTLE system and an NVIDIA Jetson TX2 board.

The primary contributions of this paper are as follows:

- We propose a general methodology to introduce and control approximation in visual SLAM algorithms. This methodology works across different combinations of SLAM algorithms, programming language implementations, and hardware platforms.
- We designed an online controller that adapts to a variety of real-world input datasets.
- We describe how this methodology can be used for four diverse visual or visual inertial SLAM algorithms: ElasticFusion, KinectFusion, ORB-SLAM2 and ICE-BA.
- Our experiments show that the proposed methodology is able to reduce the computation time and energy consumption by up to 77% and 40% across different visual SLAM algorithms and platforms, while limiting accuracy loss within reasonable bounds.

The system design principles for online control of approximation presented in this paper are general and can be applied to other problems where one has to make trade-offs between different objectives, such as online resource managers in multicore processors, checkpointing for fault-tolerance [10, 11], inter-processor communication [16, 32, 50], and choosing formats for sparse matrices [31, 36]. Using concepts from control theory, like we do in this paper, may make these systems more efficient and robust.

2 SLAM ALGORITHMS AND MOTIVATION

This section describes the visual (or visual inertial) SLAM algorithms used in this paper and their performance on resource-constrained platforms.

2.1 Visual SLAM algorithms

In visual SLAM algorithms, the processing of a frame is divided into two phases: *localization* and *mapping*.

(1) *Localization* (also known as *tracking*): The incoming frame is aligned with the surface of the existing map to determine the new pose (position and orientation) of the agent, assuming that the agent's movement between successive frames is small. Alignment algorithms are usually iterative methods that can be based on pixels (*direct* method) or on features

[1] Besides camera, visual inertial SLAM also uses IMU sensor.

extracted from the frame (*indirect* method). The difference after the alignment provides a measure of the motion between the old pose and the new one.

(2) *Mapping*: In the mapping phase, the map is incrementally updated based on the new pose and the pixels (*dense* method) or features (*sparse* method) in the current frame.

Figure 1 shows ORB-SLAM2 tracking a sequence of frames and building the 3D point cloud.

Figure 1: A screenshot of ORB-SLAM2 [38] tracking camera poses (blue tetrahedrons) while simultaneously building the 3D point cloud. The top picture shows the point cloud and the inset picture shows the actual scene.

We pick four representative open-source visual SLAM algorithms that use different tracking and mapping methods, described briefly below. Table 1 shows the important knobs for these algorithms.

- ElasticFusion is a direct dense visual SLAM algorithm that incrementally captures dense globally-consistent surfel-based maps of room-scale scenes using an RGB-D camera [63]. ElasticFusion uses frame-to-model camera tracking and windowed surfel-based fusion coupled with frequent model refinement through non-rigid surface deformations. During the tracking phase, ElasticFusion uses both photometric (RGB) tracking and iterative closest point (ICP) by default. The importance of RGB tracking and ICP is weighted by the knob icp_rgb.
- KinectFusion is a direct dense SLAM system [26, 40] that uses ICP for tracking and volumetric representations for mapping. Knob csr determines the frame resolution at which KinectFusion processes input frames and knob icp controls how fast the tracking alignment converges. A lot of work in this area builds on KinectFusion [27, 62–64].
- ORB-SLAM2 is an indirect sparse visual SLAM algorithm [37, 38], and can be used in diverse environments ranging from room-sized scenes to several city blocks. For each input frame, ORB-SLAM2 extracts orb features, which are fast to compute and are used for both tracking and mapping [51].
- ICE-BA represents incremental, consistent and efficient bundle adjustment [34], which is a state-of-the-art backend of visual inertial SLAM. The frontend of the visual inertial SLAM algorithm can be adopted from ORB-SLAM2. We use ICE-BA to refer to a whole visual inertial SLAM algorithm in this paper.

Algorithm	Knob	Description
ElasticFusion	confi	surfel confidence threshold
	cut	cutoff distance for depth processing
	icp_rgb	relative ICP/RGB tracking weight
	reloc	enable relocalization
	fo	use one level pyramid fast odometry
	nso	disable SO(3) pre-alignment
	lc	enable loop closure
KinectFusion	csr	input frame resolution
	tr	rate at which tracking is performed
	ir	rate at which mapping is performed
	icp	threshold for the ICP algorithm
	vr	volume resolution to build map
	mu	truncation distance used in volume
	pd0-pd2	maximum number of iterations allowed on each image pyramid level
ORB-SLAM2	mf	frame maximum number of features
	sl	number of levels in image pyramid
	sf	scale between levels in pyramid
	ift	1st threshold of FAST algorithm
	sft	2nd threshold of FAST algorithm
	lc	enable loop closure
ICE-BA	feat_cap	frame maximum number of features
	feat_ratio	scale number of new features

Table 1: List of selected knobs in different SLAM algorithms that potentially can be tuned for approximation.

Knobs can be tuned to affect the performance and accuracy of SLAM algorithms. Each knob is set to a default value, chosen by the implementers as a reasonable value for that knob. We use the term *DEFAULT* to refer to a configuration of a SLAM algorithm that uses these default values.

2.2 Baseline visual SLAM performance

Since desktops are not natural deployment targets for visual SLAM algorithms, we tested the performance of ElasticFusion, KinectFusion, ORB-SLAM2, and ICE-BA on two embedded platforms, an ODROID XU4 board and an NVIDIA Jetson TX2 board equipped with low-power ARM cores (details in Section 6.1). Each SLAM algorithm is tested over its own reference dataset. ElasticFusion and ORB-SLAM2 use the TUM dataset [57], KinectFusion uses the standard ICL-NUIM living room dataset [22], and ICE-BA uses the EuRoC dataset [12].

To get a sense of how the algorithms perform on these platforms, we used two configurations: DEFAULT and APPROX (all knobs are set to their lowest fidelity value). On the Jetson TX2, ElasticFusion in DEFAULT knob configuration took 326 ms per frame on the average, but only 147 ms per frame in the APPROX knob configuration. For KinectFusion, the corresponding times are 313 ms and 68 ms on the Odroid XU4, and 61 ms and 24 ms on the Jetson TX2. These results show that there is a lot of scope for exploiting approximation to reduce running time.

Unfortunately, we found that the APPROX knob settings often compromised the ability of the agent to navigate successfully through the environment because it introduced a large tracking

error. The metric used in the literature to quantify tracking error is called *Trajectory Root Mean Square Error* (*TE*) in the SLAM literature [57]. The TE measures the root mean square of the difference between the actual and computed poses of the agent at each frame. The increase of TE because of approximation as a proportion of the longest dimension in the frame is required to be within a given bound for the solution to be acceptable. Formally,

$$Q_{loss} = \frac{max(TE - TE_D, 0)}{L} < 1\% \quad (1)$$

where TE_D is the TE when using the default setting and L is the longest dimension of the input's scene. For example, a TE increase of 5 cm is usually considered acceptable in room-sized scenes (~5m×5m×5m). The TE constraint means that approximation cannot be done in a naïve way such as by using the APPROX configuration for all the knobs. For example, knob mf controls the maximum number of features extracted from a frame in ORB-SLAM2 and thus affects the computation time. Computation time per frame is almost halved for input fr1_desk if knob mf is set to its lowest fidelity, but this introduces a TE of 87 cm (Q_{loss} = 17%) in a room-sized scene, which is unacceptable.

The goal of this paper is to develop a generic methodology to introduce approximation in visual SLAM algorithms in a *principled* fashion so that the computation time and energy consumption can be improved without unacceptable tracking quality loss.

3 RELATED WORK ON APPROXIMATION IN SLAM

We summarize the most relevant studies below.

Application-agnostic control. The systems community has proposed many application-agnostic control systems for trading off power or energy for performance and program accuracy [19, 24, 25, 29]. These controllers do not use domain knowledge and usually involve offline training on inputs before the actual execution. However, inputs are not known ahead of time for SLAM. In our experience, they do not work well for controlling visual SLAM [44].

Control of SLAM algorithms. Most prior work on approximating SLAM assume that the *entire video of frames is known before the agent starts to move* [4, 42, 52]. Design space exploration [4] for a given video input is performed by executing actual trials and the results are used to select good knob settings that are then used for the entire input. This is an example of *offline control* since knob settings are determined once and for all before the computation begins. Subsequent studies along this line highlighted opportunities for exploiting approximation in SLAM algorithms [5, 39, 42, 52]. In most applications of SLAM however, video inputs are not known ahead of time. Prior work [44] shows that permitting knobs to be controlled adaptively during the navigation of the agent reduces time and energy requirements more than fixing knobs during execution. This work explored the use of a SLAM-specific controller for the KinectFusion algorithm on one embedded platform for one dataset, but does not give a general methodology for introducing approximation in visual SLAM algorithms. Another recent work focuses on dynamically controlling DVFS on embedded platforms [28], but embedded platforms usually do not have the base computational power to permit turning down the voltage/frequency for visual SLAM algorithms without loss of tracking. The performance advantages of reduced-precision arithmetic in SLAM have been explored in other work [41, 42]. Exploiting reduced-precision is orthogonal to our concerns in this paper.

4 METHODOLOGY FOR DESIGNING SLAM CONTROLLERS

This section presents a general methodology for principled approximation in visual SLAM systems. There are four problems that must be addressed.

- Offline control is not an option for SLAM because entire inputs are not available ahead of time, so we use online control. Online control of SLAM requires an *error estimator* that can be used to adjust knobs whenever a new frame is received. TE is not useful for this because it requires ground truth. Section 4.1 describes an error estimator of our design that performs well for all the SLAM algorithms studied in this paper.
- The next problem is to choose an *online control strategy*. Section 4.2 describes a modified *PID (proportional-integral-derivative) controller* that is suitable for application to various visual SLAM systems.
- In most SLAM systems, the number of knobs is very large, and it is not feasible to control all of them simultaneously. Section 4.3 describes how we choose a subset of knobs that are tuned by the controller.
- PID controllers have the advantage that they are domain-independent, but our experiments show that exploiting SLAM-specific information to improve on the basic PID strategy is essential to achieve good performance. Section 4.4 describes two SLAM-specific optimizations that are selectively applied to SLAM systems.

4.1 Error estimator for controlling visual SLAM

Online control requires an error estimator that can be used by the controller to tune knobs when a new frame is received. As discussed earlier, TE is not useful for this because it needs ground truth.

One possibility is to define an *Instantaneous* Trajectory Error (ITE): intuitively, this is the difference between where the agent thinks it is and where it actually is (i.e., ground truth). The controller can integrate the ITE over time and by comparing this with the TE target, it can determine how much to approximate. However, ground truth is not available during SLAM execution. What is needed therefore is a *proxy* for ITE.

Based on our experiments, *we recommend using pose distance as the error estimator*, where pose distance is defined as the distance between the estimated poses of successive frames. Since visual SLAM algorithms assume that the agent's movement between successive frames is small, a large pose distance between successive frames indicates that either the tracking error is large or the scene is difficult to track (for example, the camera is moving fast). Compared to the error estimator *velocity* suggested by [44], pose distance permits faster reaction time. This is crucial for visual SLAM algorithms because the tracking quality of every frame has an impact on latter frames. Figure 2 shows the relationship between pose distance and ITE for the input lr0 in KinectFusion using its default setting (ITE is

calculated using ground truth). We see that there is a good correlation between ITE and pose distance, especially when pose distance has spikes. This correlation is widely observed in different input trajectories (correlation > 0.4 across the ICL-NUIM dataset). This means that a large pose distance between frames is a good indicator of large tracking error. Less approximation should be introduced when a large pose distance is observed.

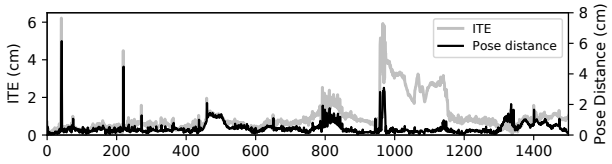


Figure 2: Correlation between the ITE and pose distance of input lr0 in KinectFusion.

We studied several other error estimators, but they were uncorrelated to ITE (correlation < 0.1) and some are costly to compute.

4.2 Online Control Strategy

We use a proportional-integral-derivative (PID) controller as the basic controller for visual SLAM algorithms. This controller is optimized using domain-specific information in Section 4.4.

The simplest PID controllers are just proportional (P) controllers: they compare instantaneous estimates of some quantity of interest with a reference value for that quantity (called the *set-point*), and make adjustments to knobs proportionate to this difference. Cruise controllers in cars and thermostats are simple examples. More sophisticated controllers integrate (sum) this difference over time, and use the accumulated value as well for control; these are PI controllers. Finally, PID controllers can also use the derivative of the quantity of interest in their control strategy (this is useful to reduce overshooting the set-point through over-aggressive control).

In our study, we adopt a slightly modified PID controller. The P portion of our PID controller determines the knob setting by comparing the agent's current pose distance with two dynamic pose distance set-points, D_{high} and D_{low}:

$$D_{high} = \alpha * D_{avg} + (1 - \alpha) * D_{max} \quad (2)$$

$$D_{low} = (1 - \alpha) * D_{avg} + \alpha * D_{min} \quad (3)$$

where D_{avg}, D_{max}, and D_{min} are the average, maximum and minimum pose distances from the beginning to the current pose during one execution. The value α determines how those values are combined. As shown in Figure 3, when pose distance is smaller than D_{low}, the knob that provides the most approximation (APPROX) will be applied because the agent is not moving fast. On the other hand, when pose distance is larger than D_{high}, the knob setting of DEFAULT will be applied. Approximation is set to different knob values (Section 4.3) proportionally when pose distance is between D_{low} and D_{high}: the closer to D_{low}, the more approximate knob setting is applied. Dynamic set-points can adapt more quickly to changing trajectories than fixed set-points [44]. Intuitively, if a trajectory is fast and DEFAULT *is able to* track it, D_{low} and D_{high} should be set higher to avoid overly conservative control.

The knob setting determined by the P term is further refined by the I and D terms. The I part in our proposed controller is a

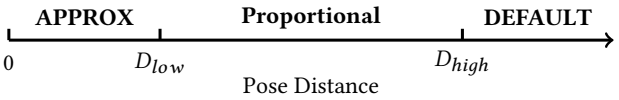


Figure 3: Modified PID controller

sliding window of the recent history of the difference between pose distance and the pose distance set-points. The knob settings computed by the P part are adjusted by computing the average difference over the sliding window. The D part checks whether the pose distance in the sliding window is constantly increasing or decreasing. If the pose distance is increasing, a more accurate knob setting should be applied accordingly to deal with potential difficult scene.

A PID controller determines the exact knob value, which implies that the knob setting may frequently oscillate between different values. One alternative is to let knob settings change incrementally toward the desired value over many time steps. This alternative has a longer reaction time but may have better stability. It can be useful when changing knobs is costly.

4.3 Knob Importance Analysis

Given the PID controller and error estimator, we must decide which knobs should be controlled. The SLAM algorithms in our study have a large number of knobs as Table 1 shows, and some knobs have a wide range of possible values. Controlling all the knobs is neither necessary nor advantageous. We classify knobs into three categories: *infeasible*, *ineffective*, and *effective* knobs. Knobs which are suitable for tuning and which have a meaningful impact on computation time, energy consumption, and accuracy are classified as effective.

Identifying infeasible knobs. Knobs are infeasible if their values cannot or should not be changed dynamically. Changing infeasible knobs may lead to incorrect results or is expensive.

Visual SLAM systems assume that the camera's motion between successive frames is small. Modifying some knobs may violate this fundamental assumption. For example, knobs tr and ir of KinectFusion affect the rate at which tracking and mapping is performed. Lower tr and ir increase the pose distance between successive frames substantially, thereby increasing the difficulty for the tracking module. In addition, knobs that are expensive to tune need to be excluded. Knob vr in KinectFusion determines the scale of the global map. Changing vr involves rescaling the whole 3D map to the new resolution, which is costly to do repeatedly. For the same reason, mu is also classified as infeasible. Furthermore, knob reloc in ElasticFusion determines whether relocalization should be performed, but the information required to tune these knobs is not easy to get.

Ranking knobs by impact. To identify effective knobs, we rank the remaining knobs based on their impact on the output. The ranking is done by *orthogonal line* search, where we change only one knob at a time and keep all other knobs fixed at their default values. We first shrink the range of feasible values of a knob by discretization. For example, the default value of knob mf in ORB-SLAM2 is 1000 and its minimum value is 510[2], so we discretize

[2]In ORB-SLAM2, mf value less than 510 may cause segmentation fault.

the range of mf to [1000, 900, 800, 700, 600, 510]. The inputs used are the standard living room trajectories from ICL-NUIM RGB-D dataset. For each knob, we measure its impact on the computation time, energy usage, and the TE.

Algorithm	Knob	Knob range
ORB-SLAM2	mf	[1000, 900, 800, 700, 600, 510]
ElasticFusion	fo	[false, true]
	icp_rgb	[10, 100]
	lc	[true, false]
KinectFusion	csr	[1, 2, 4, 8]
	icp	[1e-05, 1e-04, 1e-03, 1]
	pd0	[10, 8, 6, 4]
ICE-BA	feat_cap	[1000, 30, 20, 10, 5]
	feat_ratio	[3, 2, 1, 0.5, 0.25]

Table 2: List of effective knobs for each SLAM algorithm.

Table 2 lists the effective knobs for each SLAM algorithm. For a given knob, the settings are listed in an increasing order of approximation. The default setting uses knob values with the least amount of approximation. For KinectFusion, ORB-SLAM2, and ICE-BA, all knobs are tuned simultaneously while ElasticFusion's knobs are adjusted in the order of fo, icp_rgb, and lc to avoid *bang-bang* control. Some knobs should *not* be disabled for a long period such as knob lc, which controls loop closure, so knob lc is forced to be enabled at least every four frames in our control system.

4.4 Improving the controller using domain knowledge

The PID control strategy is domain-agnostic and easy to implement, but our experiments showed that it sometimes performed poorly, such as when the scene has few features for accurate alignment of successive frames. We recommend adding two SLAM-specific optimizations to the controller to ameliorate these problems.

Pose extrapolation using an inertia model. For some difficult video inputs, a single noisy frame may cause the tracking module to produce a pose estimate that is very far from the pose estimate at the previous frame. Since the basic assumption in all SLAM algorithms is that the frame rate is sufficient to ensure that the agent's actual pose does not change drastically between successive frames, this is a sign that the pose estimate produced for the new frame is probably incorrect. If left uncorrected, this potentially incorrect pose will be propagated to future time steps and the agent may lose tracking altogether. The PID controller cannot react to such incorrect estimates since these are inputs to the PID controller.

To avoid this problem, we use a simple inertia model to generate a different estimate for the pose, as illustrated in Figure 4. The difference between two poses can be viewed as a *rigid body transformation*. Let P_{t-1} represent the pose and T_{t-1} represent the transformation matrix at time $t-1$ (*i.e.*, $P_{t-1} = T_{t-1} * P_{t-2}$).

At time t, the pose extrapolation module calculates the pose distance between P_t and P_{t-1}. If this distance is suspicious (*i.e.*, larger than some predefined threshold), the P_t estimate from the tracking module is discarded and P_t is recomputed by applying T_{t-1} to P_{t-1}. Note that pose extrapolation is different from the *relocalization* technique in ElasticFusion and ORB-SLAM2. Relocalization deals with tracking loss or scene changes while pose extrapolation tries to correct the pose whenever needed based on an inertia model. Pose extrapolation is effective in maintaining the accuracy for pure visual SLAM systems such as ElasticFusion, KinectFusion, and ORB-SLAM2.

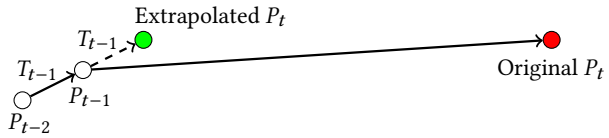


Figure 4: Pose extrapolation is performed when a pose estimate is substantially distant from the previous pose.

Structure Detection. The final domain-specific optimization we use is to exploit additional information in the incoming frames to determine how to control approximation for visual SLAM algorithms that use the direct method, such as ElasticFusion and KinectFusion. The direct method processes frames at the pixel level for tracking, and the alignment is performed using iterative methods. For example, KinectFusion uses only ICP, while ElasticFusion uses ICP together with RGB tracking. If successive frames are feature-rich (e.g., a room with many chairs and tables), then alignment is relatively easy and the controller can get away with more approximation; conversely, if successive frames are feature-poor (e.g., the scene is mostly empty), then tracking is more difficult and approximation should be reduced. Note that the PID controller and the pose extrapolation module cannot react to this in time because pose distance is calculated *after* the frame is processed; in ElasticFusion, RGB tracking should be turned on *before* ElasticFusion starts to process the frame if the frame is likely to cause problems.

To estimate whether a frame is feature-rich, we use a *structure detection* module to calculate the standard deviations of pixel depth values of a frame. Intuitively, a feature-rich frame will have a higher standard deviation than a feature-poor one. In our implementation, we divide the frame into four rectangles. In each rectangle, a uniform sampling of a fixed number (40×30) of points is performed to calculate the standard deviation regardless of frame resolution. If the standard deviations are small, we classify the frame as feature-poor and apply less approximation to process it.

5 IMPLEMENTATION

This section describes how we implemented the control methodology for the four SLAM algorithms. The processing of each frame in visual SLAM algorithms can be abstracted into three main steps: *reading*, *tracking*, and *mapping*. One advantage of our methodology is that it can be implemented as plugins into these three steps without modifying the code.

Figure 5 shows the flow chart for our implementation. Box b0 reads an input frame, denoted as F_t. A *bootstrap phase* is used to permit SLAM to run without approximation and to initialize an accurate map (Box b1). In this phase, the algorithm's execution follows its original procedure, where the pose (P_t) and the map (M_t) at time t are calculated by the tracking and the mapping module

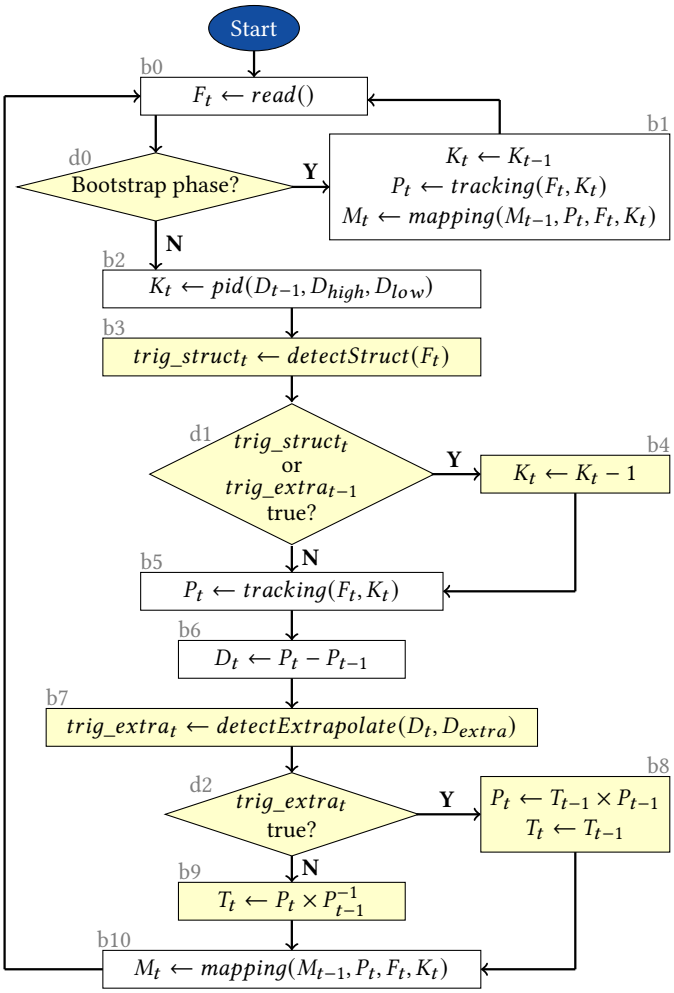


Figure 5: Diagram of the proposed methodology

with knobs (K_t) set to DEFAULT respectively. The length of the bootstrap phase is set equal to the length of the sliding window in the PID controller's I part.

Control of the SLAM algorithm starts after the bootstrap phase. The PID controller determines the knob setting K_t for processing the current frame (Box b2). The input to the PID controller is the pose distance between the last two frames (D_{t-1}) and two dynamic pose distance set-points (D_{low} and D_{high}).

- If no optimization is applied (ignore highlighted boxes for now), the tracking algorithm will use the new knob setting and calculate the pose (P_t) for the current frame (Box b5). The pose distance between the current frame and the previous frame (D_t) is calculated in Box b6, and this value is fed into the PID controller for the next frame. The mapping module takes P_t as its input (Box b10), and outputs the map M_t after processing of this frame.
- The logic of pose extrapolation and structure detection are shown in the highlighted boxes. The need for applying pose extrapolation for the current frame is checked in Box b7 by comparing D_t with a predefined threshold according to different datasets (3.5 cm for ICL-NUIM dataset and 4.5 cm for TUM datset). If pose extrapolation is required, then P_t is recomputed using a transformation matrix T_{t-1} calculated from P_{t-1} to P_{t-2} (Box b8). The mapping module (Box b10) takes the recomputed pose as its input. If no pose extrapolation is needed, a new transformation matrix (T_t) is still generated in case pose extrapolation is needed for the next frame (Box b9).

  Box b3 shows the module of structure detection. If the current frame is feature-poor, knobs are tuned to reduce approximation by one level according to Table 2 (Diamond d1 and Box b4). We denote the operation of reducing approximation by "-1" for simplicity. We treat each knob of one algorithm equally so all the knobs are adjusted at the same time except ElasticFusion. For ElasticFusion, we tune the knobs hierarchically, following the order of fo, icp_rgb, and lc, otherwise the knob space would be too small. Note that if extrapolation is performed for the previous frame, knob settings are also tuned to increase accuracy because extrapolation indicates a difficult scene (Box b7).

In our study, the PID controller is implemented in all four algorithms. Pose extrapolation is added to ElasticFusion, KinectFusion, and ORB-SLAM2 because ICE-BA is already using inertial information. Structure detection is added only to ElasticFusion and KinectFusion because ORB-SLAM2 and ICE-BA are already extracting features from frames.

6 EXPERIMENTAL RESULTS

6.1 Experiment Setup

We have implemented our controller in four visual SLAM algorithms, ElasticFusion, KinectFusion, ORB-SLAM2, and ICE-BA, and evaluated them on different embedded platforms.

Platforms. The first embedded platform is a heterogeneous ODROID XU4 big.LITTLE system, equipped with four Cortex-A15 cores with 2 GHz max frequency, four Cortex-A7 cores with 1.4 GHz max frequency, and a Mali-T628 MP6 GPU that supports OpenCL 1.2. The power dissipation on this board is measured by an external SmartPower2 device[3]. OpenCL version of KinectFusion and C++ version of ORB-SLAM2 and ICE-BA are tested on this platform.

The second platform is an NVIDIA Jetson TX2 development kit that has four Cortex-A57 cores and two Denver2 cores running at 2 GHz. The board is equipped with 256-core Pascal GPU running at 1.3 GHz with CUDA version 10.0. The power dissipation is measured through its internal i2C interface. The idle power of both platforms is subtracted from the actual measurements. CUDA version of ElasticFusion and KinectFusion, C++ version of ORB-SLAM2 and ICE-BA are evaluated on this board.

Datasets. Each SLAM algorithm comes with its own reference datasets. For ORB-SLAM2 and ElasticFusion, we use 14 inputs from the TUM RGB-D SLAM dataset [57]. The dataset includes ground truth. Most of these inputs can be successfully tracked by the default setting of both algorithms. Both ORB-SLAM2 and ElasticFusion are run in RGB-D mode. We use the ICL-NUIM [22] living room dataset for KinectFusion. The dataset contains four synthetic living room scenes, with ground truth. We also extended the ICL-NUIM dataset

[3]https://wiki.odroid.com/accessory/power_supply_battery/smartpower2

Algorithm	Language	Datasets	Computation Time (ms)			Energy Consumption (mJ)			Trajectory Error (cm)		
			DEF	CTRL	Saving	DEF	CTRL	Saving	DEF	CTRL	Max Q_{loss}
KinectFusion	OpenCL	ICL-NUIM Extended	313	73	76.7%	2293	1385	39.6%	1.2	2.1	0.74%
ORB-SLAM2	C++	TUM	386	284	26.4%	1302	984	24.4%	2.3	2.8	0.48%
ICE-BA	C++	EuRoC	68	58	14.7%	277	235	15.2%	10.6	11.1	0.55%

Table 3: Overall performance comparison between DEFAULT and CONTROLLED on the ODROID XU4 platform.

Algorithm	Language	Datasets	Computation Time (ms)			Energy Consumption (mJ)			Trajectory Error (cm)		
			DEF	CTRL	Saving	DEF	CTRL	Saving	DEF	CTRL	Max Q_{loss}
ElasticFusion	CUDA	TUM	327	193	41.0%	912	562	38.4%	3.5	4.4	0.76%
KinectFusion	CUDA	ICL-NUIM Extended	61	26	57.4%	858	550	35.9%	1.0	2.7	0.84%
ORB-SLAM2	C++	TUM	257	191	25.7%	1100	838	23.8%	2.5	2.7	0.96%
ICE-BA	C++	EuRoC	48	39	18.8%	410	404	1.5%	11.7	11.4	0.25%

Table 4: Overall performance comparison between DEFAULT and CONTROLLED on the Jetson TX2 platform.

with 14 real-world scenes from the literature [44] for diversity. The largest dimension for the above two datasets is around ~5 meters. ICE-BA is evaluated over the EuRoC MAV dataset [12] (ground truth is available). The first 5 scenes' dimension is around 15 meters and the latter 6's is 8 meters. The first column of Table 5, 8, 7 and 9 list the inputs for each visual SLAM algorithm used in our study.

6.2 Performance on Embedded Platforms

We perform a quantitative comparison of the trajectory root mean square error (TE), computation time, and the energy consumption per frame between two configurations:

- DEFAULT (DEF): knobs are fixed to their default value,
- CONTROLLED (CTRL): knobs are adaptively-controlled by the proposed methodology during execution.

The overall computation time and energy consumption on each platform is shown in Table 3 and Table 4 respectively. Each number is the geomean average of the performance of all the corresponding inputs of each visual SLAM algorithm. The detailed analysis of each algorithm is described below.

ElasticFusion. Our proposed methodology improves ElasticFusion's computation time (or fps) and energy consumption by 41% and 38% respectively on the Jetson TX2 board. Detailed numbers are shown in Table 5. Performance numbers for trajectories that cannot even be tracked by DEFAULT are marked as "-". In terms of accuracy, DEFAULT has a TE of 3.5 cm while the TE for CONTROLLED is 4.4 cm. The maximum quality loss Q_{loss} is 0.76% with fr2_dwp, which is considered as acceptable in a room-sized scene.

KinectFusion. The improvement in computation time and energy usage per frame for KinectFusion is 77% and 40% on ODROID XU4, 57% and 36% on Jetson TX2 board. Due to lack of space, KinectFusion's, ORB-SLAM2's, and ICE-BA's performance data are included in the Appendix A. Detailed performance of KinectFusion are shown in Table 7. On ODROID XU4 board, the frame rate increase from 3.2 to 13.7 only with a maximum Q_{loss} of 0.74% (lab0). On Jetson TX2 board, frame rate (38.5) exceeds the real-time requirement with maximum Q_{loss} of 0.84% (mr0, lab0). Note that due to architecture differences, the tracking accuracies are different on each platform.

Computation time improvements from approximation are pronounced because KinectFusion exposes a knob csr which controls

	DEFAULT			CONTROLLED		
	RMSE (cm)	Time (ms)	E (mJ)	RMSE (cm)	Time (ms)	E (mJ)
fr1_desk	2.6	308	812	3.1	217	597
fr1_desk2	4.3	313	770	4.6	224	557
fr1_floor	-	-	-	-	-	-
fr1_plant	4.6	327	827	6.7	205	530
fr1_room	19.8	327	795	13.9	181	461
fr1_xyz	1.2	327	798	1.8	196	480
fr2_desk	9.8	335	1084	11.4	178	603
fr2_dwp	6.8	338	1106	10.6	187	643
fr2_xyz	2.0	354	971	2.4	171	549
fr3_long	2.4	328	1098	3.4	198	666
fr3_nt_near	3.6	353	902	3.9	246	633
fr3_sn_far	3.1	311	901	4.4	184	556
fr3_st_near	1.0	314	975	2.4	181	582
fr3_st_far	2.6	320	908	2.7	167	497
geomean	3.5	326	912	4.4	193	562

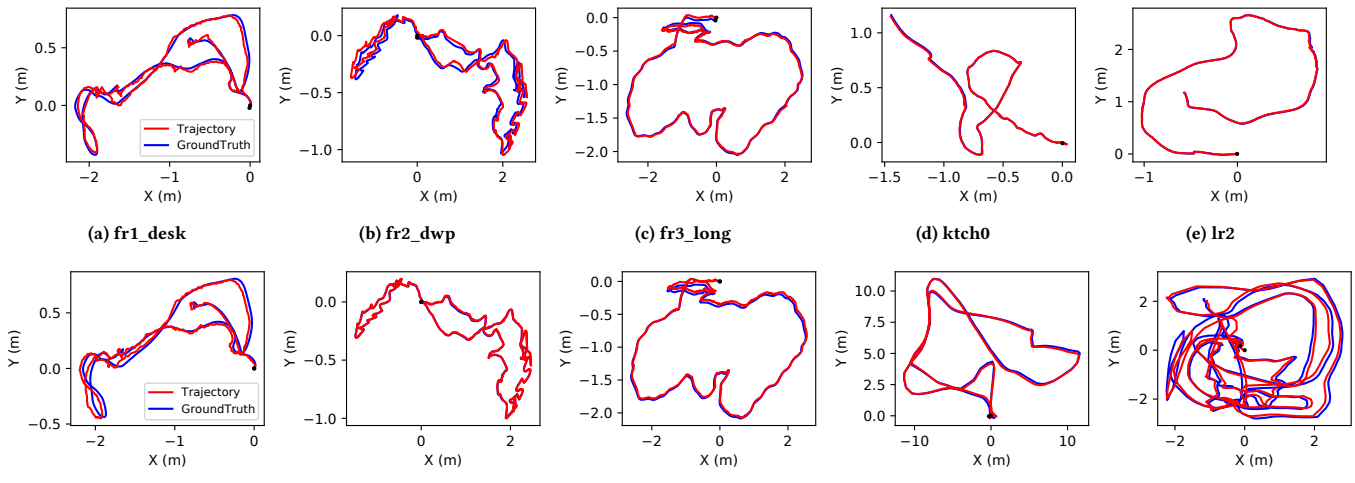
Table 5: ElasticFusion on the Jetson TX2 platform.

Controller	RMSE (cm)	Time (ms)	E (mJ)
SLAMBooster	2.9	83	1437
Proposed Methodology	2.1	73	1385

Table 6: Performance comparison between proposed methodology with SLAMBooster on ODROID XU4.

the frame resolution, and has significant impact on both computation time and energy usage. Our methodology outperforms SLAMBooster [44] with less TE (Table 6) for two reasons: the use of dynamic set-points permits better adaptation to differences in input scenes, and pose distance reduces the reaction time of the controller compared to using velocity.

ORB-SLAM2. Our experiment shows that on both platforms, controlled ORB-SLAM2 is able to reduce the computation time and energy consumption by 26% and 24% respectively. Table 8 lists detailed performance numbers for each input on each platform of ORB-SLAM2. We exclude fr3_sn_far because even DEFAULT yields a TE over 100 cm. The maximum Q_{loss} is 0.48% (fr1_plant) on the



(a) fr1_desk (b) fr2_dwp (c) fr3_long (d) ktch0 (e) lr2

(f) fr1_desk (g) fr2_dwp (h) fr3_long (i) MH_05_difficult (j) V2_03_difficult

Figure 6: Sample computed trajectories by ElasticFusion (a-c), KinectFusion (d,e), ORB-SLAM2 (f-h) and ICE-BA (i, j).

ODROID XU4 and 0.96% (fr1_room) on the Jetson TX2. Our proposed methodology meets the quality loss bound.

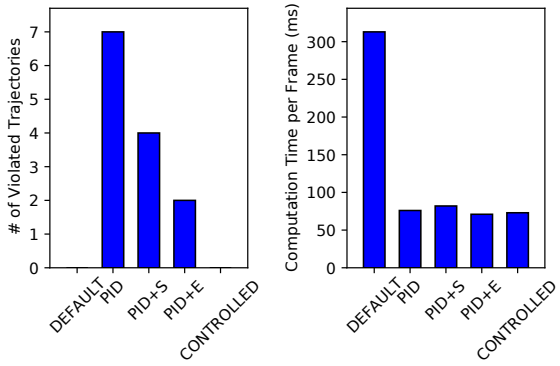
ICE-BA. As shown in Table 9 (see in Appendix A), computation time and energy usage for ICE-BA are reduced by 15% and 15% on ODROID XU4, 18% and 1.5% on Jetson TX2 while keeping the maximnum Q_{loss} (V2_02) within quality bound. ICE-BA's frame rate increases by 2.6 and 4.8 on two boards respectively. Interestingly, there is no significant energy saving on the Jetson board. The potential reason is explained in Section 6.8. We believe ICE-BA can be further improved if more knobs are exposed in the implementation for tuning.

Figure 6 shows some examples of computed trajectories tracked by the four visual SLAM algorithms compared to the ground truth. Using our methodology permits visual SLAM algorithms to maintain tracking effectively with approximation for inputs ranging from room-size scenes to hall-size scenes.

Discussion. Although introducing approximation should generally increase TE, we found that TE can sometimes decrease after approximation. This is because visual SLAM algorithms are nonlinear, and approximation can affect the output in unexpected ways. Given the nonlinearity of the underlying system and the fact that the decisions made by the controller may be different for different inputs, it is difficult to even formulate the notion of optimal controllers for this problem, let alone design them, but our methodology gives a way to design controllers that are easy to implement and perform well in practice.

6.3 Importance of Domain-specific Control Optimizations

In Section 4.4, we introduced SLAM-specific control optimizations such as pose extrapolation and structure detection in the PID controller for better tracking accuracy. In Figure 7, we show the importance of these optimizations when applied to KinectFusion. The configurations in order are 1) DEFAULT, 2) PID ONLY, 3) PID + S:



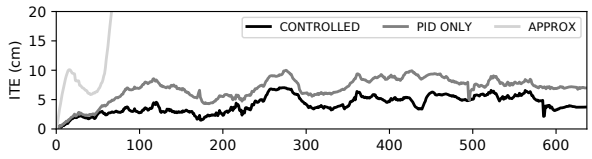
(a) Number of failed inputs (b) Computation time

Figure 7: Importance of incorporating domain specific optimizations to the PID controller.

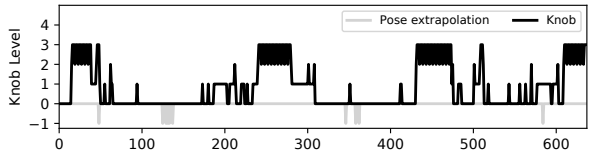
PID with structure detection, 4) PID + E: PID with pose extrapolation, and 5) CONTROLLED: PID with both structure detection and pose extrapolation.

Figures 7a and 7b show the number of inputs that violoate TE requirement and the computation time of different configurations respectively. Compared to DEFAULT, PID by itself improves computation time significantly but suffers large degradation in TE. In particular, 7 of 17 inputs have a quality loss of more than 1%. Structure detection optimization improves the TE at the cost of ~8% overhead to the computation time, but even then 4 out of 17 inputs suffer from large tracking error. Pose extrapolation alone can reduce the number of violated inputs to 2 (kctn1, off2) but these two inputs can be saved by structure detection. Combining pose extrapolation and structure detection helps bring the TE down and the average TE is only worse by 0.9 cm (Q_{loss} = 0.74%) over the default setting.

SLAM-specific optimizations are important for the other visual SLAM algorithms as well. For example, the input fr3_nt_near lacks structure information. Unlike ORB-SLAM2 or ICE-BA using orb



(a) Instantaneous Trajectory Error



(b) Knob level and pose extrapolation trigger activity

Figure 8: ITE of input fr1_desk2 with corresponding knob and pose extrapolation activity with ElasticFusion.

features, ElasticFusion is not able to process this input properly with only the PID controller. The controller may turn off RGB tracking (set knob icp_rgb to 100), while only RGB information is available in this input. This input can be successfully tracked after structure detection is applied because this optimization helps keep RGB tracking enabled in scenes that lack structural information.

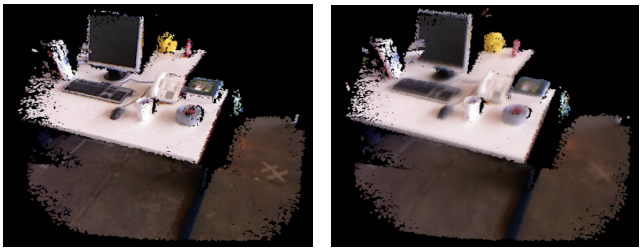
A more dramatic illustration of the importance of these optimizations is given in Figure 8a, which shows the ITE over time of the input fr1_desk2 with ElasticFusion for three different ways of controlling knobs: APPROX (the most approximate configuration, light gray line), PID ONLY (gray line), and CONTROLLED (black line). The ITEs are computed after the execution by comparing the output trajectory with the available ground truth.

The ITE of APPROX increases dramatically after 50 frames, and tracking never recovers after that. The final TE of this configuration is over 130 cm. The PID ONLY is able to track the trajectory and its TE is 7.9 cm. The configuration CONTROLLED performs best and its TE is 4.6 cm. The ITE between PID ONLY and CONTROLLED differs significantly around frame 50, the ITE for PID ONLY increases up to 10 cm while CONTROLLED's stays within 6 cm. The reason for this is explained by Figure 8b, which shows the evolution of knob level (black line) and pose extrapolation (gray line) of the configuration CONTROLLED. Knob level 0 means no approximation, 1 means fo is true, 2 means fo and icp_rgb are true, 3 means all three knobs are set. Knob level oscillates from 2 and 3 frequently because lc (loop closure) should not be disabled for long. At around frame 50, pose extrapolation is activated by the controller because it detects a pose anomaly (-1 means activated), which enables CONTROLLED to continue tracking accurately. In contrast, PID ONLY ingests a pose that is compromised and the error slowly accumulates over the rest of the frames, showing the importance of the SLAM-specific optimizations in our methodology.

6.4 Pose Distance Vs. Velocity

Velocity is computed over several frames while pose distance uses only the two most recent frames. In our controller, pose distance is used as the error estimator instead of velocity due to its faster reaction time. As illustrated in Figure 8a, TE in the initial stages can accumulate and propagate to later stages of the execution. Using pose distance as the error estimator helps the system to be more responsive to deviations in the trajectory.

Our experiments show that, if velocity is used, average TE increases by 0.9 cm, 0.39 cm, 0.03 cm and 0.87 cm for KinectFusion, ElasticFusion, ORB-SLAM2 and ICE-BA respectively. The performance and energy are not changed significantly and their gain mainly comes from other optimizations.



(a) DEFAULT scene (b) CONTROLLED scene

Figure 9: Controller's impact on mapping.

6.5 Impact of α in the PID Controller

The PID controller in our proposed methodology uses the parameter α, introduced in Section 4, to adjust dynamic pose distance set-points: the lower α is, the larger D_{low} and D_{high} can be. Larger pose distance set-points potentially mean less approximation will be introduced during execution because the computed pose distance has a greater chance to fall below D_{low} and D_{high}. The impact of α on ElasticFusion's performance is shown below: computation time and energy usage grow with α. We use 0.5 for all the SLAM algorithms in our paper.

Ratio α	RMSE (cm)	Time (ms)	E (mJ)
0.25	4.1	208	601
0.50	4.4	193	562
0.75	5.1	186	544

6.6 Impact on Mapping

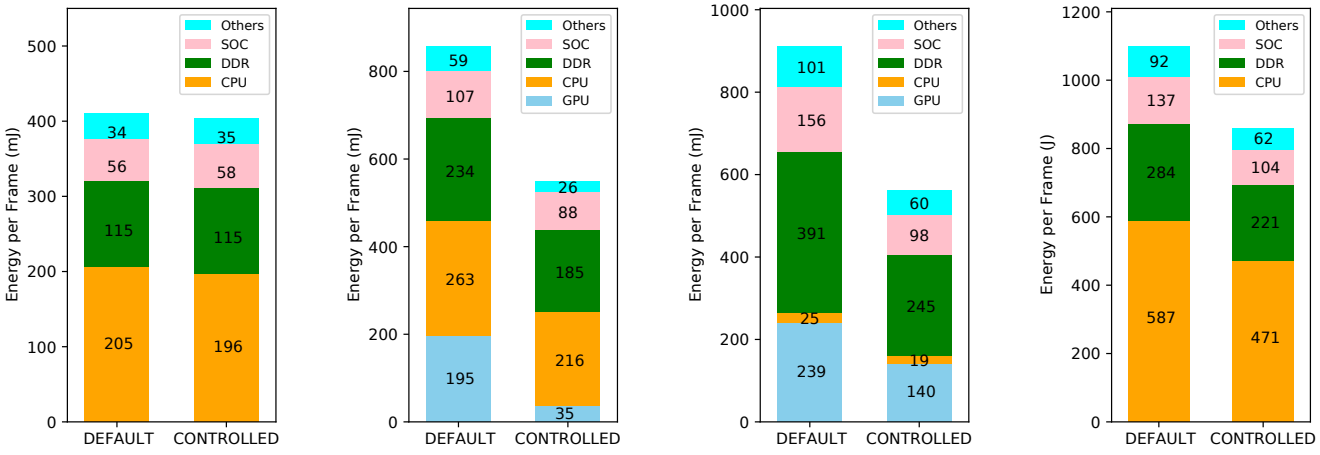
Mapping is an integral component of SLAM, so it is important that approximation should not significantly affect the reconstructed scenes. Mapping needs to be good enough so that the agent can still navigate correctly. For example, we compared the scenes reconstructed by ElasticFusion with the DEFAULT and the CONTROLLED configurations in Figure 9. Scene geometry remains the same despite slight scene resolution degradation. Our proposed methodology does not affect the quality of the mapping significantly.

6.7 Breakdown of Computation Time

The KinectFusion implementation [6][4] provides a detailed breakdown of the time spent in the different stages of computation[5]. The

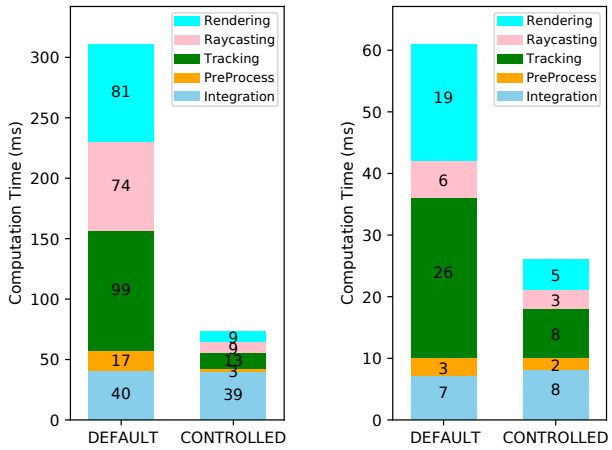
[4]https://github.com/pamela-project/slambench2

[5]Other algorithms do not provide a breakdown of the computation time.



(a) ICE-BA (b) KinectFusion (c) ElasticFusion (d) ORB-SLAM2

Figure 10: Breakdown of the impact of approximation on the energy consumption on Jetson TX2 platform.



(a) Platform: ODROID XU4 (b) Platform: Jetson TX2

Figure 11: Breakdown of the impact of approximation on the computation time of different stages in KinectFusion.

different stages in KinectFusion are preprocessing, tracking, integration, raycasting, and rendering. Figure 11 shows the breakdown of the impact of approximation on the computation time overhead of the different stages on both platforms. The computation time of the integration stage has no improvement on both platforms because we do not control knob vr and mu, which are related to the integration stage. The amount of computation incurred in preprocessing, tracking, raycasting, and rendering are mainly related to the frame resolution controlled by knob csr. Therefore, the saving is the most significant in these four stages. The time reduction of the tracking stage is further helped by knob icp and pd.

6.8 Breakdown of Energy Consumption

The NVIDIA Jetson TX2 board allows measuring power dissipation for different on-board hardware components. The power dissipation is measured through the i2C interface, a serial protocol for low power devices. Energy usage can be divided into five parts: cpu, gpu, ddr, soc, and others.

Figure 10a shows the distribution of the energy usage across the different on-board hardware components for ICE-BA. Computation time is reduced for ICE-BA by the proposed controller but energy usage is not improved substantially because power dissipation goes up in various components to sustain the higher frame rate.

KinectFusion's energy usage distribution is shown in Figure 10b. GPU energy is significantly reduced because the knob csr scales the frame size so that the number of CUDA cores used is largely reduced.

Figures 10c and 10d show the distribution of the energy usage for ElasticFusion and ORB-SLAM2. The energy consumption by the CPU is negligible since ElasticFusion is implemented in CUDA, while ORB-SLAM2 is a C++ application so GPU energy consumption is minimal. The improvement in the energy usage is proportional to the savings in computation time because ElasticFusion and ORB-SLAM2's data throughput is not high. Board components are still running at the same speed even though computation time is reduced.

7 CONCLUSION

In this paper, we proposed a general methodology for introducing principled approximation in visual SLAM algorithms to reduce their computation and energy requirements. We have implemented our proposed methodology in four SLAM algorithms and evaluated them across different platforms. Our results show that our methodology is effective in improving the run-time performance and energy usage of visual SLAM with negligible loss in accuracy.

ACKNOWLEDGMENTS

This research was supported by NSF grants 1618425, 1705092, and 1725322.

REFERENCES

[1] Woongki Baek and Trishul M. Chilimbi. 2010. Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation. In *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '10)*. ACM, New York, NY, USA, 198–209. https://doi.org/10.1145/1806596.1806620

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. 1989. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[3] Paul J Besl and Neil D McKay. 1992. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 2 (1992), 239–256. https://doi.org/10.1109/34.121791

[4] Bruno Bodin, Luigi Nardi, M. Zeeshan Zia, Harry Wagstaff, Govind Sreekar Shenoy, Murali Emani, John Mawer, Christos Kotselidis, Andy Nisbet, Mikel Lujan, Björn Franke, Paul H.J. Kelly, and Michael O'Boyle. 2016. Integrating Algorithmic Parameters into Benchmarking and Design Space Exploration in 3D Scene Understanding. In *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation (PACT '16)*. ACM, New York, NY, USA, 57–69. https://doi.org/10.1145/2967938.2967963

[5] Bruno Bodin, Harry Wagstaff, Sajad Saeedi, Luigi Nardi, Emanuele Vespa, John H Mayer, Andy Nisbet, Mikel Luján, Steve Furber, Andrew J Davison, Paul H.J. Kelly, and Michael O'Boyle. 2018. SLAMBench2: Multi-Objective Head-to-Head Benchmarking for Visual SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[6] Bruno Bodin, Harry Wagstaff, Sajad Saeedi, Luigi Nardi, Emanuele Vespa, John H Mayer, Andy Nisbet, Mikel Luján, Steve Furber, Andrew J Davison, Paul H.J. Kelly, and Michael O'Boyle. 2018. SLAMBench2: Multi-Objective Head-to-Head Benchmarking for Visual SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

[7] K. Boikos and C. S. Bouganis. 2016. Semi-Dense SLAM on an FPGA SoC. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. 1–4. https://doi.org/10.1109/FPL.2016.7577365

[8] James Bornholt, Todd Mytkowicz, and Kathryn S. McKinley. 2014. Uncertain⟨T⟩: A First-order Type for Uncertain Data. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems* (Salt Lake City, Utah, USA) *(ASPLOS '14)*. ACM, New York, NY, USA, 51–66. https://doi.org/10.1145/2541940.2541958

[9] S. Borthwick and H. Durrant-Whyte. 1994. Simultaneous Localisation and Map Building for Autonomous Guided Vehicles. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, Vol. 2. IEEE, 761–768 vol.2. https://doi.org/10.1109/IROS.1994.407552

[10] G. Bronevetsky, R. Fernandes, D. Marques, K. Pingali, and P. Stodghill. 2006. Recent advances in checkpoint/recovery systems. In *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*. 8 pp.–.

[11] Greg Bronevetsky, Daniel Marques, Keshav Pingali, and Paul Stodghill. 2003. C³: A system for automating application-level checkpointing of MPI programs. In *16TH INTERNATIONAL WORKSHOP ON LANGUAGES AND COMPILERS FOR PARALLEL COMPUTERS (LCPC'03*. 357–373.

[12] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. 2016. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research* 35, 10 (2016), 1157–1163.

[13] Simone Campanoni, Glenn Holloway, Gu-Yeon Wei, and David Brooks. 2015. HELIX-UP: Relaxing Program Semantics to Unleash Parallelization. In *Proceedings of the 13th Annual IEEE/ACM International Symposium on Code Generation and Optimization (CGO '15)*. IEEE Computer Society, Washington, DC, USA, 235–245. http://dl.acm.org/citation.cfm?id=2738600.2738630

[14] Michael Carbin, Deokhwan Kim, Sasa Misailovic, and Martin C. Rinard. 2012. Proving Acceptability Properties of Relaxed Nondeterministic Approximate Programs. In *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation* (Beijing, China) *(PLDI '12)*. ACM, New York, NY, USA, 169–180. https://doi.org/10.1145/2254064.2254086

[15] Michael Carbin, Sasa Misailovic, and Martin C. Rinard. 2013. Verifying Quantitative Reliability for Programs That Execute on Unreliable Hardware. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications* (Indianapolis, Indiana, USA) *(OOPSLA '13)*. ACM, New York, NY, USA, 33–52. https://doi.org/10.1145/2509136.2509546

[16] Roshan Dathathri, Gurbinder Gill, Loc Hoang, Hoang-Vu Dang, Alex Brooks, Nikoli Dryden, Marc Snir, and Keshav Pingali. 2018. Gluon: A Communication-optimizing Substrate for Distributed Heterogeneous Graph Analytics. In *PLDI*.

[17] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. 2007. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 29, 6 (April 2007), 1052–1067. https://doi.org/10.1109/TPAMI.2007.1049

[18] Jakob Engel, Thomas Schöps, and Daniel Cremers. 2014. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European conference on computer vision*. Springer, 834–849.

[19] Anne Farrell and Henry Hoffmann. 2016. MEANTIME: Achieving Both Minimal Energy and Timeliness with Approximate Computing. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. USENIX Association, Denver, CO, 421–435. https://www.usenix.org/conference/atc16/technical-sessions/presentation/farrell

[20] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. 2014. SVO: Fast Semi-Direct Monocular Visual Odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 15–22. https://doi.org/10.1109/ICRA.2014.6906584

[21] Udo Frese. 2010. Interview: Is SLAM Solved? *KI-Künstliche Intelligenz* 24, 3 (Sept. 2010), 255–257.

[22] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. 2014. A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 1524–1531. https://doi.org/10.1109/ICRA.2014.6907054

[23] Kin Leong Ho and Paul Newman. 2007. Detecting Loop Closure with Scene Sequences. *International Journal of Computer Vision* 74, 3 (2007), 261–286.

[24] Henry Hoffmann. 2015. JouleGuard: Energy Guarantees for Approximate Applications. In *Proceedings of the 25th Symposium on Operating Systems Principles (SOSP '15)*. ACM, New York, NY, USA, 198–214. https://doi.org/10.1145/2815400.2815403

[25] C. Imes, D. H. K. Kim, M. Maggio, and H. Hoffmann. 2015. POET: A Portable Approach to Minimizing Energy Under Soft Real-time Constraints. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium*. 75–86. https://doi.org/10.1109/RTAS.2015.7108419

[26] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 559–568.

[27] Olaf Kähler, Victor Adrian Prisacariu, Carl Yuheng Ren, Xin Sun, Philip Torr, and David Murray. 2015. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices. *IEEE Transactions on Visualization and Computer Graphics* 21, 11 (Nov. 2015), 1241–1250.

[28] Abdullah Khalufa, Graham Riley, and Mikel Luján. 2019. A Dynamic Adaptation Strategy for Energy-Efficient Keyframe-Based Visual SLAM. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*. The Steering Committee of The World Congress in Computer Science, Computer …, 3–10.

[29] Daya S Khudia, Babak Zamirai, Mehrzad Samadi, and Scott Mahlke. 2015. Rumba: An online quality management system for approximate computing. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 554–566. https://doi.org/10.1145/2749469.2750371

[30] Georg Klein and David Murray. 2007. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 1–10. https://doi.org/10.1109/ISMAR.2007.4538852

[31] Vladimir Kotlyar, Keshav Pingali, and Paul Stodghill. 1997. A Relational Approach to the Compilation of Sparse Matrix Programs. In *Euro-Par '97: Proceedings of the Third International Euro-Par Conference on Parallel Processing*. Springer-Verlag, London, UK, 318–327. http://iss.ices.utexas.edu/Publications/Papers/EUROPAR1997.pdf

[32] Andrew Lenharth, Donald Nguyen, and Keshav Pingali. 2016. Parallel Graph Analytics. *Commun. ACM* 59, 5 (April 2016), 78–87. https://doi.org/10.1145/2901919

[33] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. 2015. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research* 34, 3 (2015), 314–334.

[34] Haomin Liu, Mingyu Chen, Guofeng Zhang, Hujun Bao, and Yingze Bao. 2018. Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1974–1982.

[35] Erdogan Madenci and Ibrahim Guven. 2015. *The Finite Element Method and Applications in Engineering Using ANSYS* (2nd ed.). Springer Publishing Company, Incorporated.

[36] Nikolay Mateev, Keshav Pingali, Paul Stodghill, and Vladimir Kotlyar. 2000. Next-generation generic programming and its application to sparse matrix computations. In *ICS '00: Proceedings of the 14th international conference on Supercomputing* (Santa Fe, New Mexico, United States). ACM, New York, NY, USA, 88–99. https://doi.org/10.1145/335231.335240

[37] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardøs. 2015. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* 31, 5 (Oct. 2015), 1147–1163. https://doi.org/10.1109/TRO.2015.2463671

[38] Raul Mur-Artal and Juan D Tardós. 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics* 33, 5 (2017), 1255–1262.

[39] Luigi Nardi, Bruno Bodin, M. Zeeshan Zia, John Mawer, Andy Nisbet, Paul H. J. Kelly, Andrew J. Davison, Mikel Luján, Michael F. P. O'Boyle, Graham Riley, Nigel Topham, and Steve Furber. 2015. Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM. In *IEEE International Conference*

on Robotics and Automation (ICRA). arXiv:1410.2167.

[40] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *IEEE ISMAR*. IEEE.

[41] Jinwook Oh, Jungwook Choi, Guilherme C Januario, and Kailash Gopalakrishnan. 2016. Energy-Efficient Simultaneous Localization and Mapping via Compounded Approximate Computing. In *Signal Processing Systems (SiPS), 2016 IEEE International Workshop on*. IEEE, 51–56.

[42] Oscar Palomar, Andy Nisbet, John Mawer, Graham Riley, and Mikel Lujan. 2017. Reduced precision applicability and trade-offs for SLAM algorithms. In *Third Workshop on Approximate Computing (WACAS)*.

[43] Jongse Park, Hadi Esmaeilzadeh, Xin Zhang, Mayur Naik, and William Harris. 2015. FlexJava: Language Support for Safe and Modular Approximate Programming. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015)*. ACM, New York, NY, USA, 745–757. https://doi.org/10.1145/2786805.2786807

[44] Y. Pei, S. Biswas, D. S. Fussell, and K. Pingali. 2019. SLAMBooster: An Application-Aware Online Controller for Approximation in Dense SLAM. In *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. 296–310. https://doi.org/10.1109/PACT.2019.00031

[45] Raghavendra Pradyumna Pothukuchi, Amin Ansari, Petros Voulgaris, and Josep Torrellas. 2016. Using Multiple Input, Multiple Output Formal Control to Maximize Resource Efficiency in Architectures. In *Proceedings of the 43rd International Symposium on Computer Architecture* (Seoul, Republic of Korea) *(ISCA '16)*. IEEE Press, Piscataway, NJ, USA, 658–670. https://doi.org/10.1109/ISCA.2016.63

[46] Amir M. Rahmani, Bryan Donyanavard, Tiago Müch, Kasra Moazzemi, Axel Jantsch, Onur Mutlu, and Nikil Dutt. 2018. SPECTR: Formal Supervisory Control and Coordination for Many-core Systems Resource Management. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems* (Williamsburg, VA, USA) *(ASPLOS '18)*. ACM, New York, NY, USA, 169–183. https://doi.org/10.1145/3173162.3173199

[47] A. Ratter, C. Sammut, and M. McGill. 2013. GPU Accelerated Graph SLAM and Occupancy Voxel Based ICP for Encoder-Free Mobile Robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 540–547. https://doi.org/10.1109/IROS.2013.6696404

[48] Martin Rinard. 2006. Probabilistic Accuracy Bounds for Fault-Tolerant Computations That Discard Tasks. In *Proceedings of the 20th Annual International Conference on Supercomputing (ICS '06)*. ACM, New York, NY, USA, 324–334. https://doi.org/10.1145/1183401.1183447

[49] Martin C. Rinard. 2007. Using Early Phase Termination To Eliminate Load Imbalances At Barrier Synchronization Points. In *Proceedings of the 22Nd Annual ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications (OOPSLA '07)*. ACM, New York, NY, USA, 369–386. https://doi.org/10.1145/1297027.1297055

[50] Anne Rogers and Keshav Pingali. 1994. Compiling for Distributed Memory Architectures. *IEEE Trans. Parallel Distrib. Syst.* 5, 3 (1994), 281–298. https://doi.org/10.1109/71.277789

[51] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. 2011. ORB: an efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*. IEEE, 2. https://doi.org/10.1109/ICCV.2011.6126544

[52] Sajad Saeedi, Luigi Nardi, Edward Johns, Bruno Bodin, Paul H. J. Kelly, and Andrew J Davison. 2017. Application-oriented Design Space Exploration for SLAM Algorithms. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5716–5723. https://doi.org/10.1109/ICRA.2017.7989673

[53] Mehrzad Samadi, Davoud Anoushe Jamshidi, Janghaeng Lee, and Scott Mahlke. 2014. Paraprox: Pattern-Based Approximation for Data Parallel Applications. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 35–50. https://doi.org/10.1145/2541940.2541948

[54] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. 2011. EnerJ: Approximate Data Types for Safe and General Low-Power Computation. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '11)*. ACM, New York, NY, USA, 164–174. https://doi.org/10.1145/1993498.1993518

[55] Adrian Sampson, Pavel Panchekha, Todd Mytkowicz, Kathryn S. McKinley, Dan Grossman, and Luis Ceze. 2014. Expressing and Verifying Probabilistic Assertions. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Edinburgh, United Kingdom) *(PLDI '14)*. ACM, New York, NY, USA, 112–122. https://doi.org/10.1145/2594291.2594294

[56] Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. 2011. Managing Performance vs. Accuracy Trade-offs With Loop Perforation. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE '11)*. ACM, New York, NY, USA, 124–134. https://doi.org/10.1145/2025113.2025133

[57] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. 2012. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

[58] Xin Sui, Andrew Lenharth, Donald S. Fussell, and Keshav Pingali. 2016. Proactive Control of Approximate Programs. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16)*. ACM, New York, NY, USA, 607–621. https://doi.org/10.1145/2872362.2872402

[59] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. 2017. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6243–6252.

[60] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. 1999. Bundle Adjustment—A Modern Synthesis. In *International Workshop on Vision Algorithms*. Springer, 298–372.

[61] Shu Wang, Chi Li, Henry Hoffmann, Shan Lu, William Sentosa, and Achmad Imam Kistijantoro. 2018. Understanding and Auto-Adjusting Performance-Sensitive Configurations. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems* (Williamsburg, VA, USA) *(ASPLOS '18)*. ACM, New York, NY, USA, 154–168. https://doi.org/10.1145/3173162.3173206

[62] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. 2015. Real-time large scale dense RGB-D SLAM with volumetric fusion. *The International Journal of Robotics Research* 34, 4-5 (2015), 598–626.

[63] Thomas Whelan, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison. 2015. ElasticFusion: Dense SLAM Without A Pose Graph. In *Proceedings of Robotics: Science and Systems*. Rome, Italy. https://doi.org/10.15607/RSS.2015.XI.001

[64] Thomas Whelan, John Mcdonald, Michael Kaess, Maurice Fallon, Hordur Johannsson, and John J. Leonard. 2012. Kintinuous: Spatially Extended KinectFusion. In *3rd RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*.

[65] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. 2016. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research* 35, 14 (2016), 1697–1716.

A APPENDIX

This appendix includes detailed performance number of KinectFusion, ORB-SLAM2, and ICE-BA due to page limit.

		ODROID XU4						Jetson TX2					
		DEFAULT			CONTROLLED			DEFAULT			CONTROLLED		
	# frames	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)
lr0	1510	1.5	330	2404	1.7	71	1447	1.3	62	903	1.5	23	589
lr1	967	1.9	341	2463	1.6	69	1479	0.6	62	922	1.4	25	597
lr2	882	1.1	352	2564	1.2	103	1569	1.2	63	947	1.7	25	596
ktcn0	1550	0.9	277	2114	1.5	64	1354	1.2	58	829	1.0	25	544
ktcn1	800	2.1	273	2182	3.0	65	1365	3.1	59	816	4.7	26	539
lab0	800	0.8	287	2205	4.5	67	1334	0.5	61	830	4.5	27	542
lab1	1250	0.7	282	2090	2.0	66	1361	0.7	58	810	2.4	28	538
lab2	1250	3.4	292	2191	6.2	73	1361	3.1	59	817	6.8	27	535
lab3	1250	1.2	316	2213	3.5	73	1347	0.7	62	846	1.9	28	543
mr0	929	4.0	357	2449	2.5	80	1401	1.4	67	888	5.6	26	549
mr1	1494	1.3	347	2411	4.9	93	1432	6.8	65	872	9.5	26	542
mr2	1450	0.8	320	2312	1.3	73	1345	0.6	61	842	3.7	26	547
off0	750	0.5	294	2165	0.7	74	1334	0.2	59	821	0.8	27	539
off1	1050	1.3	349	2470	0.8	99	1499	1.7	66	887	5.8	28	561
off2	1200	0.4	296	2155	1.2	68	1294	0.5	60	822	1.2	25	520
pd0	1600	1.3	336	2385	3.1	67	1334	0.3	65	902	2.6	28	544
pd1	1500	1.4	304	2269	4.3	64	1325	0.7	59	851	2.9	26	540
geomean		1.2	313	2292	2.1	73	1385	1.0	61	858	2.7	26	550

Table 7: Performance of KinectFusion on the ODROID XU4 and Jetson TX2 platform.

		ODROID XU4						Jetson TX2					
		DEFAULT			CONTROLLED			DEFAULT			CONTROLLED		
	# frames	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)
fr1_desk	573	3.7	536	2032	4.5	365	1410	3.1	394	1741	3.5	284	1254
fr1_desk2	637	3.7	530	2005	5.2	472	1730	4.7	403	1776	4.3	352	1551
fr1_floor	1238	1.9	270	766	2.0	201	555	1.9	161	623	2.1	128	505
fr1_plant	1139	3.3	468	1779	5.7	406	1310	4.9	343	1503	4.2	260	1212
fr1_room	1360	11.8	383	1366	13.8	319	1187	17.1	278	1221	21.9	225	1097
fr1_xyz	792	1.3	411	1451	1.3	341	1316	1.4	281	1239	1.3	237	1041
fr2_desk	2451	2.9	417	1377	2.0	267	969	2.9	267	1090	2.0	191	789
fr2_dwp	3898	1.0	396	1338	1.5	312	1184	0.9	296	1256	1.1	224	964
fr2_xyz	3666	0.5	307	946	0.5	221	765	0.6	171	726	0.5	138	609
fr3_long	2509	2.5	544	1997	4.2	290	1008	2.5	394	1688	5.2	225	971
fr3_nt_near	1646	5.3	261	756	6.6	203	685	4.9	153	651	5.9	116	504
fr3_sn_far	794	-	-	-	-	-	-	-	-	-	-	-	-
fr3_st_near	908	1.5	342	1093	1.3	226	685	1.5	196	842	1.8	120	528
fr3_st_far	1065	1.4	308	977	2.2	214	709	1.8	198	853	1.8	139	615
geomean		2.3	386	1302	2.8	284	984	2.5	257	1100	2.7	191	838

Table 8: Performance of ORB-SLAM2 on the ODROID XU4 and Jetson TX2 platform.

		ODROID XU4						Jetson TX2					
		DEFAULT			CONTROLLED			DEFAULT			CONTROLLED		
	# frames	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)	RMSE (cm)	Time (ms)	E (J)
MH_01	3685	11.8	72	307	12.6	57	227	6.6	51	404	7.7	37	404
MH_02	3043	6.4	72	278	6.4	56	220	11.7	53	408	8.7	36	401
MH_03	2703	7.1	67	286	8.7	60	248	14.3	51	410	8.3	43	407
MH_04	2035	14.2	59	238	18.9	53	206	19.7	44	407	19.7	36	403
MH_05	2276	19.8	60	249	13.4	54	208	15.1	44	408	19.1	33	393
V1_01	2915	5.7	59	244	8.7	55	226	6.2	42	403	5.8	38	406
V1_02	2283	9.6	66	273	7.5	60	244	9.5	46	407	10.7	39	402
V1_03	1713	14.2	73	301	14.3	62	264	15.2	49	421	17.0	43	407
V2_01	2251	9.4	59	228	8.4	54	207	10.5	43	394	8.8	38	394
V2_02	2152	10.9	78	317	15.3	65	277	13.6	52	424	15.6	45	411
V2_03	1925	15.6	88	353	15.2	68	288	13.9	52	424	12.9	44	423
geomean		10.6	68	277	11.1	58	235	11.7	48	410	11.4	39	404

Table 9: Performance of ICE-BA on the ODROID XU4 and Jetson TX2 platform.