Parallel, Real-Time Monocular Visual Odometry

Shiyu Song
University of California, San Diego
shs012@ucsd.edu

Manmohan Chandraker
NEC Labs America
manu@nec-labs.com

Clark C. Guest
University of California, San Diego
cguest@ucsd.edu

***Abstract*—We present a real-time, accurate, large-scale monocular visual odometry system for real-world autonomous outdoor driving applications. The key contributions of our work are a series of architectural innovations that address the challenge of robust multithreading even for scenes with large motions and rapidly changing imagery. Our design is extensible for three or more parallel CPU threads. The system uses 3D-2D correspondences for robust pose estimation across all threads, followed by local bundle adjustment in the primary thread. In contrast to prior work, epipolar search operates in parallel in other threads to generate new 3D points at every frame. This significantly boosts robustness and accuracy, since only extensively validated 3D points with long tracks are inserted at keyframes. Fast-moving vehicles also necessitate immediate global bundle adjustment, which is triggered by our novel keyframe design in parallel with pose estimation in a thread-safe architecture. To handle inevitable tracking failures, a recovery method is provided. Scale drift is corrected only occasionally, using a novel mechanism that detects (rather than assumes) local planarity of the road by combining information from triangulated 3D points and the inter-image planar homography. Our system is optimized to output pose within 50 ms in the worst case, while average case operation is over 30 fps. Evaluations are presented on the challenging KITTI dataset for autonomous driving, where we achieve better rotation and translation accuracy than other state-of-the-art systems.**

I. Introduction

Visual odometry for real-world autonomous outdoor driving is a problem that had gained immense traction in recent years. We present a real-time, monocular visual odometry system that relies on several innovations in multithreaded structure-from-motion (SFM) architecture to achieve excellent performance in terms of both timing and accuracy.

While stereo SLAM systems routinely achieve high accuracy and real-time performance, the challenge remains daunting for monocular ones. Yet, monocular systems are attractive for the automobile industry since they are cheaper and the calibration effort is lower. Costs of consumer cameras have steadily declined in recent years, but cameras for practical visual odometry in automobiles are expensive since they are produced in lesser volume, must support high frame rates and be robust to extreme temperatures, weather and jitters.

The challenges of monocular visual odometry for autonomous driving are both fundamental and practical. For instance, it has been observed empirically and theoretically that forward motion with epipoles within the image is a "high error" situation for SFM [15]. Vehicle speeds in outdoor environments can be high, so even with cameras that capture imagery at high frame rates, large motions may occur between consecutive frames. This places severe demands on an autonomous driving visual odometry system, necessitating extensive validation and refinement mechanisms that conventional systems do not require. Our system makes judicious use of a novel multihtreaded design to ensure that motion estimates (and the underlying structure variables) become available only after extensive validation with long-range constraints and thorough bundle adjustments, but without delay.

The timing requirements for visual odometry in autonomous driving are equally stringent – a pose must be output at every frame in a fixed amount of time. Thus, our system is optimized for worst-case timing scenarios, rather than the average-case optimization for most traditional systems. For instance, traditional systems may produce a spike in timings when keyframes are added, or loop closure is performed [2]. In particular, our multithreaded system produces pose outputs in at most $50\,ms$, regardless of whether a keyframe is added or scale correction performed. The average frame rate of our system is much higher, at above $30\,fps$.

Visual odometry is an inherently sequential operation. This is especially true for outdoors autonomous driving, as opposed to indoor or desktop applications where the possibility of repeatedly viewing the same scene structures is high. For our application, with rapidly changing points in the visible field of view, bundle adjustment must be per-frame, while highly accurate new 3D points must be added to the system with no luxury of off-cycles or revisited regions of the map to perform delayed refinements (unlike [8]). Thus, designing a multithreaded system requires achieving a delicate balance between accuracy and latency. A high redundancy epipolar search mechanism, novel keyframe architectures that allow updating trackable 3D points with reliable long tracks and thread safe modules that allow online global bundle adjustment are some of the innovations that allow our system to meet the competing demands of speed, accuracy and robustness.

A key emphasis of this work is to illustrate that besides the obvious speed advantages, well-designed multithreading can also greatly contribute to the accuracy and robustness of the system. This is best illustrated by our epipolar search and keyframe insertion architectures – Sections III and IV explore this theme in greater detail. But the accuracy of our system is also borne out by our experiments in Section VI that show the system to be resilient in retaining highly accurate scale over long periods. To handle inevitable scale drift over long sequences, Section V presents a mechanism for scale correction using local planarity of the ground plane, but it does not require

the assumption to be true or enforced at every frame. Instead, it automatically determines the validity of the assumption and the need for scale correction, ensuring that the system only uses reliable scale information.

Our system architecture is intricately designed to meet the challenge of accurate and efficient monocular autonomous driving. In Section II-A, we discuss how our design is fundamentally different from prior works, better suited to the application and easily extensible.

II. Related Work

A. Comparison to Other Monocular Architectures

PTAM: An elegant two-thread architecture separating the tracking and mapping aspects of monocular visual SLAM has been proposed by Klein and Murray [8]. However, it is designed for small workspace environments and relies extensively on repeatedly observing a small set of 3D points ("loopy browsing motions"). This does not scale well to large outdoor environments or driving situations where scene points rapidly disappear from the field of view. The latter is an important restriction that motivates our improved architecture.

As an example, consider the epipolar search mechanism to add new points to the map. PTAM uses the existing distribution of points to restrict the epipolar search range, but such constraints are not possible in our system since the distribution changes dramatically per-frame. Moreover, PTAM performs epipolar search on demand and data association refinement is used to validate the point in other frames during free time on the mapping thread when exploring already seen regions. Again, our system does not have the leeway to revisit regions of the map, so all our refinement must be online. These considerations lead us to move epipolar search to a separate thread of its own, but it presents a unique opportunity for enhancing robustness. As described in Section III-B, we perform epipolar search per-frame, which allows only those points to be added to the map that have been validated multiple times and effectively tracked over a desired number of frames. This innovation is crucial in ensuring high accuracy for a system like ours that primarily relies on rapidly moving 3D points for pose computations.

Several other aspects differentiate our system from PTAM's parallel architecture, such as global bundle adjustment, which may take several seconds for PTAM while looping over already seen regions. The computational demands on our system are much higher at every frame due to changing imagery, so this is neither possible nor desired. Instead, a small global bundle adjustment over the previous few keyframes suffices. The need to do this in parallel with pose computations requires novel keyframe architectures to accommodate timing constraints, as well as to maintain thread safety, as discussed in Section IV.

2D-2D Matching (Libviso): A system for autonomous driving that is a counterpoint to ours is proposed by Geiger et al. in [6]. It relies on matching and computing relative pose between every consecutive pair of frames through a fundamental matrix estimation and uses continuous scale correction against a locally planar ground. There are several drawbacks of such an approach: it is known that two-view estimation leads to high translational errors in the case of narrow baseline forward motion [15] and there is higher drift since distant constraints from long tracks are not used.

In constrast, our approach is carefully designed to introduce long-range constraints while maintaining efficiency and relies on 3D-2D pose estimation rather than narrow baseline fundamental matrix estimation. Further, our system is accurate enough to hold the correct scale for lengthy intervals while ground plane estimation on low texture roads is noisy, so using that information for scale correction at every frame is likely to cause higher errors. Instead, we implement an error-tolerant scale correction mechanism in Section V-B that needs to be invoked only occasionally (once every 100 frames in our experiments, see Figure 8). Thus, it is worth emphasizing that we do not *enforce* local planarity as a hard assumption, rather only seek to *use* it when true.

B. Other Related Work

Binocular stereo is one of the success stories of visual SLAM, providing real-time localization in environments as diverse as indoors [2], outdoors [14], or even extraterrestrial terrains [16]. Parallel implementations for visual stereo SLAM that harness the power of GPUs have been demonstrated to achieve frame rates exceeding 30 fps in indoor environments [2]. Several approaches have also been proposed that use or combine information from alternate acquisition modalities such as omnidirectional [23], ultrasound [1] or depth sensors [4].

A few purely vision-based monocular systems have achieved good localization accuracy, but mainly for smaller indoor environments [3], [8], [9] (in the case of [9], impressive agility is further attained by incorporating edgelets in the map). Systems using only monocular input are rarer for large-scale autonomous navigation, due to the well-documented problem of scale drift. Loop closure is a popular technique used for scale correction in SLAM [25], which is exploited by the large-scale monocular system of [22]. However, in our application, one cannot rely on the vehicle encountering loops and it is expensive to maintain such global information of the trajectory. Moreover, loop closure is suitable for map building, however, autonomous driving requires accurate online localization.

Several works have attempted to alleviate the difficulty of monocular visual odometry by incorporating prior information about the environment. For instance, Scaramuzza et al. use nonholonomic constraints for wheeled robots, along with a planar motion model to restrict the solution space [20] . Royer et al. present an outdoor system where the robot is guided along a path by a human to enable a learning step that builds a 3D map of the environment for future navigation [18]. A method specialized for monocular robot localization in circular pipes that exploits the simpler geometry of the environment is presented by Hansen et al. in [7]. Compared to the above, our system returns full 6 degrees of freedom pose information at every frame without any assumptions on the camera motion and only polls to detect local planarity on the road.

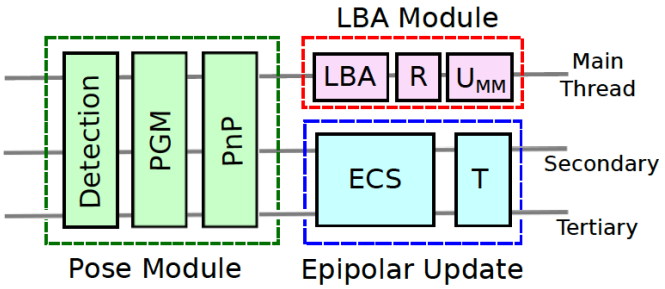


Fig. 1. System architecture for every steady state frame. The acronyms above represent PGM: Pose-guided matching, LBA: local bundle adjustment, R: re-finding, U: Update motion model, ECS: Epipolar constrained search, T: triangulation. The modules are depicted in their multithreading arrangement, in correct synchronization order but not to scale.

III. Steady State Architecture

To initialize, the system extracts FAST corners [17] with ORB descriptors [19] and matches between consecutive frames using locality sensitive hashing (LSH). With sufficient baseline (around 5 frames), a set of 3D points is initialized by relative pose estimation [13], triangulation and bundle adjustment. Each frame during initialization is processed within 10 ms.

At steady state, the system has access to a stable set of at least N_s 3D points that have undergone extensive bundle adjustment in prior frames (we choose $N_s = 100$). The preceding poses have also undergone multiple non-linear refinements, so can be considered highly accurate. The system architecture at every frame in steady state operation is illustrated in Figure 1.

A. Pose Module

Around 2000 FAST corners with Shi-Tomasi filtering [21] are extracted from a typical outdoors image and ORB descriptors are computed. Using the pose of the previous frame, the pose of the current frame is predicted, assuming constant velocity. Note that we explicitly compute the camera pose at each frame using correspondences, the motion model is only used as guidance to expedite matching. The existing set of stable 3D points are projected into the image using the predicted pose and the ORB descriptor for each is compared to those within a square of side $2r_s$ pixels (we choose $r_s = 15$). Given these $2D - 3D$ point correspondences, we compute the actual camera pose using perspective n-point (PnP) pose estimation in a robust RANSAC framework. The particular implementation we use is EPnP with a model size of 4 points [11]. The RANSAC pose with the largest consensus set is refined using a Levenberg-Marquardt nonlinear optimization.

Our system can easily handle other choices for matching, in particular, we have achieved similar results using normalized cross-correlation (NCC) instead of ORB. But assocaiting a descriptor like ORB with a 3D point can have ancillary benefits, as we will observe in the following sections.

Feature and descriptor extraction, pose-guided matching and pose estimation are all easily parallelizable across multiple threads, using a shared memory multiprocessing platform like OpenMP. Across three threads, the timings for various components of the pose module are summarized in Table I.

FAST corner detection + Shi-Tomasi	1 ms
ORB descriptor extraction	5 ms
Pose-guided matching	1 ms
PnP (RANSAC, 500 iterations)	15 ms
Nonlinear pose refinement	1 ms

TABLE I
Timings for various stages of the pose module.

B. Epipolar Update Module

If the application scenario involves scenes where the set of 3D points being viewed remains unchanged, then the pose module by itself would be sufficient to maintain the camera pose over extended periods. However, in outdoor applications like autonomous navigation, 3D scene points rapidly move out of view within a few frames. Thus, the stable set of points used for pose computation must be continually updated, which is the task entrusted to our epipolar search module.

As depicted in Figure 1, the epipolar search module is parallelized across two threads and follows pose estimation at each frame. The mechanism for epipolar search is illustrated in Figure 2. Let the most recent prior keyframe be frame 0. After pose computation at frame n, for every feature f_0 in the keyframe at location (x_0, y_0), we consider a square of side $2r_e$ centered at (x_0, y_0) in frame n. We consider the intersection region of this square with a rectilinear band p pixels wide, centered around the epipolar line corresponding to f_0 in frame n. The ORB descriptors for all FAST corners that lie within this intersection region are compared to the descriptor for f_0. The closest match, f_n, is found in terms of Hamming distance. This epipolar matching procedure is also repeated by computing the closest match to f_n in frame $n-1$, call it f_{n-1}. A match is accepted only if f_{n-1} also matches f_0. Note that only two sets of matches with respect to frames $(0, n)$ and $(n-1, n)$ must be computed at the frame n, since the matches between $(0, n-1)$ have already been computed at frame $n-1$.

The parameter r_e is automatically determined by the size of the motion, we use $r_e = \min\{1200\|\boldsymbol{\omega}\|^2, 10\}$, where $\boldsymbol{\omega}$ is the differential rotation between frames $n-1$ and n. Since pose estimates are highly accurate due to continuous refinement by bundle adjustment (Section III-C), epipolar lines are deemed accurate and we choose a stringent value of $p = 3$ to impose the epipolar constraint. The Hamming distance computation for 256-bit ORB descriptors in a region of interest is performed as a block, with a fast SSE implementation. To rapidly search for features that lie within the above region of interest, the detected features in an image are stored in a lookup table data structure. The key into the table is the column index of the feature and within each bucket, features are stored in sorted row order. Across two threads, this allows circular matching for a triplet of images, with up to 500 features in each, in $10-15$ ms. As opposed to brute-force searching, the lookup table results in speedups by up to a factor of 10, especially in our autonomous driving application where the images traditionally have wide aspect ratios (to cover greater field of view while limiting uninformative regions like sky).

The features that are circularly matched in frame n are

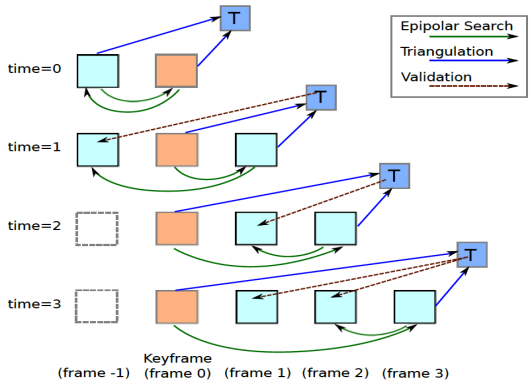


Fig. 2. Mechanism of epipolar constrained search, triangulation and validation by reprojection to existing poses. For current frame n, only 3D points that are validated against all frames 1 to $n-1$ are retained. Only persistent 3D points that survive for greater than L frames may be collected by the next keyframe.

triangulated with respect to the most recent keyframe (frame 0). This two-view triangulation requires approximately 2 ms per frame. The reconstructed 3D point is back-projected in all the frames $1, \cdots, n-1$ and is retained only if a matching ORB descriptor is found within a very tight square of side $2r_b$ pixels (we choose $r_b = 3$). This acts as a replacement for a more accurate, but expensive, multiview triangulation and is satisfactory since epipolar search produces a large number of 3D points, but only the most reliable ones may be used for pose estimation. However, these 3D points are not added to the stable point cloud yet. For that they must first undergo a local bundle adjustment and be collected by the main thread at a keyframe, which are aspects explained in the following sections. Also, note in Figure 2, the exception for the keyframe and frame 1, where validation is not possible. However, this is not an issue since only long tracks are eventually collected by the next keyframe, as explained in Section IV.

C. Local Bundle Adjustment Module

To refine camera poses and 3D points incorporating information from multiple frames, we implement a sliding window local bundle adjustment. The key data structure is the local bundle cache, which is composed of a frame cache and a match cache. The frame cache stores feature locations, descriptors and camera poses from the most recent N frames. It also stores images for those N frames, for display and debugging purposes. In our system, $N = 10$. The match cache is a list of tables, one element corresponding to each frame. The key into the table is the identity of a 3D point visible in the frame and the stored entries are the identities of the corresponding 2D features in various frames.

The local bundle adjustment module also has other functions. After bundle adjustment, we give the system a chance to re-find lost 3D points using the optimized pose. Since the system spends considerable effort in maintaining a high-quality set of 3D points for pose compuation, it is worthwhile to incur a small overhead to recover any temporarily lost ones (due to image artifacts like blur, specularities or shadows). In fact, a stable 3D point is permanently discarded only when its projection

Module	Operation	Timing
Epipolar Update	Constrained search	$10-15$ ms
	Triangulation	$1-3$ ms
Local Bundle	Windowed bundle adjustment	$10-20$ ms
	Re-find 3D points	1 ms
	Update motion model	0 ms

TABLE II
EPIPOLAR UPDATE AND LOCAL BUNDLE TIMINGS IN STEADY STATE (PARALLEL MODULES)

using the current pose falls outside the image boundaries. Since the bundle adjusted pose is highly accurate, we can perform re-finding by matching ORB descriptors on FAST corners within a very tight square of side $2r_f$ pixels (we choose $r_f = 10$). This ensures re-finding is rapidly achieved within 1 ms.

We use the publicly available SBA package for bundle adjustment [12]. In parallel, the motion model for predicting the pose of the next frame is also updated in this module. The timings for the parallel epipolar update and local bundle adjustment modules are summarized in Table II.

IV. KEYFRAME ARCHITECTURE

The system cannot maintain steady state indefinitely, since 3D points are gradually lost due to tracking failures or when they move out of the field of view. The latter is an important consideration in "forward moving" systems for autonomous driving (as opposed to "browsing" systems such as PTAM), so the role of keyframes is very important in keeping the system alive. The purpose of a keyframe is threefold:

- Collect 3D points with long tracks from the epipolar thread, refine them with local bundle adjustment and add to the set of stable points in the main thread.
- Trigger global bundle adjustment based on previous few keyframes that refines 3D points and keyframe poses.
- Provide the frame where newly added 3D points "reside".

The modules that define operations at a keyframe are illustrated in Figure 3. The pose module remains unchanged from the steady state. It is followed by a collection stage, where 3D points triangulated at each frame in the epipolar thread are gathered by the main thread. Only persistent 3D points that stem from features matched over at least L frames are collected (our circular matching for epipolar search ensures this is easily achieved by seeking 3D points only from at least L frames after the previous keyframe). Note that this mechanism imposes two necessary conditions for a point to be considered for inclusion into the stable set – it must be visible in at least two keyframes and must be tracked over at least L frames. While stringent, these conditions inherently enhance the chances that only reliable 3D points are added into the main thread. In our system, $L = 3$ regardless of environment.

The collected 3D points must reside on a keyframe for all subsequent operations, so a re-finding operation is performed by projecting them using the estimated pose for the frame and finding the best ORB match in a circular region of radius 10 pixels. Now the existing stable 3D points, the collected 3D points from the epipolar thread, their projections in all the frames within the local bundle cache and the corresponding cameras undergo local bundle adjustment. Note that the bundle

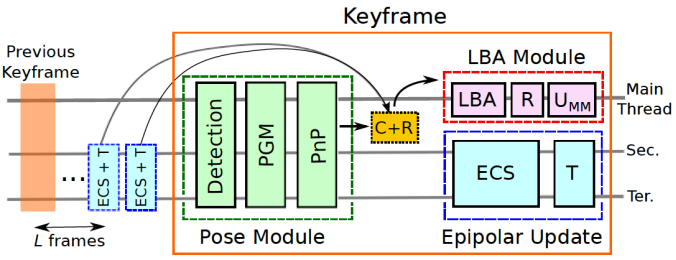


Fig. 3. System architecture for keyframes. C+R stands for a collection and refinding module. It collates persistent 3D points tracked over at least L frames in the epipolar thread and re-finds them in the current frame using the output of the pose module. The LBA is now different from that for steady state, since its cache has been updated with 3D points and their corresponding 2D locations in all the relevant frames on the epipolar thread.

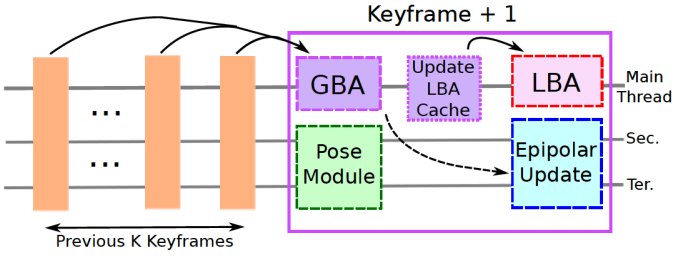


Fig. 4. System architecture for frame following a keyframe. GBA stands for global bundle adjustment. Note that GBA usually finishes within the time consumed by the pose module. The cache update module reconciles the 3D points modified by both PnP and GBA, before it is used by LBA.

adjustment at keyframes differs from steady state operation, but adding long tracks into the bundle adjustment at keyframes is a reason we can avoid more expensive multiview triangulation at each frame in the epipolar thread. The refined 3D points are now ready to be added to the stable set.

The modules that define operations at the frame immediately after a keyframe are illustrated in Figure 4. The pose module re-finds the (new) set of stable 3D points. The pose module is now split across only two threads, in order to accommodate a global bundle adjustment in the main thread. This bundle adjustment involves the previous K keyframes and their associated 3D points, in order to introduce long-range constraints to better optimize the newly added set of 3D points. For our system, choosing $K = 5$ allows the global bundle adjustment to finish within 15 ms. There are two reasons a more expensive bundle adjustment involving a much larger set of previous keyframes (or even the whole map) is not necessary to refine 3D points with long-range constraints. First, the imagery in autonomous driving applications is fast moving and does not involve repetitions, so introducing more keyframes into the global bundle adjustment yields at best marginal benefits. Second, our goal is instantaneous pose output rather than map-building, so even keyframes are not afforded the luxury of delayed output. This is in contrast to parallel systems like [2] where keyframes may produce a noticeable spike in the per-frame timings.

Following global bundle adjustment, the 3D coordinates of all the points are updated. Note that overlapping sets of 3D points are used by both global bundle adjustment and pose modules in parallel, however, both may also cause this set to change (PnP

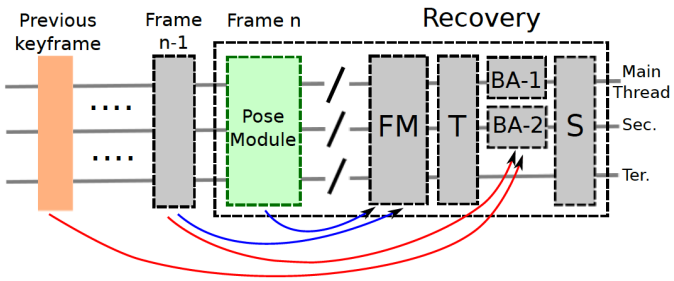


Fig. 5. System architecture for a recovery frame. FM stands for Feature matching, BA-1 and BA-2 are bundle adjustments and S denotes scale recovery.

may reject 3D points that are outliers, while bundle adjustment may move the position of 3D points). To ensure thread safety, an update module is included that reconciles changes in the 3D point cloud from both the prior parallel modules. The local bundle adjustment module, which simply reads in 3D point identities, receives this updated set for optimization based on the N frames in the local bundle cache. In parallel with local bundle adjustment, the epipolar search also makes use of the updated keyframe pose. Note that while the keyframe pose has seen a global bundle adjustment, the pose of the subsequent frame has not. This does not cause any inconsistency in practice since poses tend to be much more stable than points – a camera is constrained by hundreds of points, but a point is visible only in a few cameras. Thereafter, the system resumes steady-state operation until the next keyframe, unless a recovery or firewall condition is triggered. The following sections explain those concepts in detail.

V. Error-Correcting Mechanisms

A. Recovery

On rare occasions, the system might encounter a frame where pose-guided matching fails to find any features (due to imaging artifacts or a sudden large motion). In such a situation, a recovery mode is triggered, as illustrated in Figure 5. Let the frame where system recovery initiates be n and let k be the immediately preceding keyframe. During recovery, the frames $(n, n-1)$ are matched by comparing ORB descriptors over the entire image using fast LSH and accepting only bidirectional matches. Relative pose is computed using the 5-point algorithm [13] in a robust RANSAC framework and inlier matches are triangulated. However, scale information is lost in the process. So, we also consider 3D points observed between frames $(n-1, k)$. Both the sets of 3D points are moved to the coordinate system of frame $n-1$ and a 1-point RANSAC is performed. The hypothesis for the RANSAC is the ratio of the norms of the sampled 3D point in the two sets. The corrected scale factor between frames $(n, n-1)$ is assigned as the average ratio in the largest consensus set. To ensure that 3D points used for scale recovery are as accurate as possible, two instances of bundle adjustments are run in parallel – one between frames $(n, n-1)$ and another between frames $(n-1, k)$.

The system is designed to keep repeating the recovery mechanism until a stable set of 3D points is found. In our experiments, we always recover a stable set of 3D points after

only one frame of recovery. For sequences in the KITTI dataset, recovery is required on an average once in 1500 frames.

B. Firewall

It is well-known that scale drift is a significant problem in monocular visual odometry. As discussed previously, using global knowledge of the trajectory for scale correction, such as loop closure, is not an option for practical autonomous driving applications. Instead, we use the fact that the camera is mounted a fixed height above the road plane. Unlike prior methods, we expect our system to be accurate enough to be able to hold scale for extended periods, so while we compute scale against the ground at every frame, the scale correction mechanism is triggered sparingly. The system automatically determines when scale correction may be required - in practice, once in approximately 100 frames. Not requiring per-frame scale correction also ensures that the system is able to tide over regions where the road surface may not be planar.

Our scale detection is implemented in a separate thread. At every frame, we consider a small region of interest closest to the car (mid-section of the lower half of the image). Features within this region that can be matched to the previous frame are triangulated. We are constrained to using this narrow baseline since matching is harder on low-textured roads and the features in this region of interest rapidly move out of the field of view. A 3-point RANSAC is used to find the best-fitting plane to these triangulated points and the distance of the plane to the camera is computed as h_1. If the known height of the camera mounting is h_0, the scale correction factor is $s_1 = \dfrac{h_0}{h_1}$.

However, since the 3D points are triangulated from a small baseline, we cannot rely on s_1 always being accurate. So, an alternative mechanism is also considered that does not require triangulation. We compute the planar homography between the set of road points in the two frames, using a 4-point RANSAC where hypotheses are generated by linear estimation of algebraic error. This is followed by Levenberg-Marquardt based nonlinear optimization of the optimal symmetric reprojection error over the largest consensus set:

$$\min_{\mathbf{H}} \sum_i \|\mathbf{x}_i{}' - \mathbf{H}\mathbf{x}_i\|^2 \; + \; \|\mathbf{x}_i - \mathbf{H}^{-1}\mathbf{x}_i{}'\|^2, \qquad (1)$$

where $\mathbf{H}$ is the planar homography that maps homogeneous inlier point set $\mathbf{x}$ in the previous frame to the corresponding set $\mathbf{x}'$ in the current frame. The form of the homography $\mathbf{H}$ is:

$$\mathbf{H} = \mathbf{R} + \frac{1}{h_2}\mathbf{t}\mathbf{n}^\top, \qquad (2)$$

where $(\mathbf{R}, \mathbf{t})$ is the relative pose, $\mathbf{n}$ is the unit normal of the proposed ground plane and h_2 is the distance of the plane from the camera. The height h_2 may be recovered from $\mathbf{H}$ using a singular value decomposition of $\mathbf{H}^\top\mathbf{H}$, as discussed in [24]. Again, the scale factor may be computed as the ratio $s_2 = \dfrac{h_0}{h_2}$.

Having computed s_1 and s_2 at current frame k, we compare their values and consider them in mutual agreement when they differ by less than 5%. When in mutual agreement, it is likely that the points in consideration actually belong to a planar ground and a potential scale factor, $s_k = \text{mean}(s_1, s_2)$, is available. If $|s_k - 1| > 0.1$, the system polls for scale correction. First, it seeks to confirm the computed scale s_k by verifying that the previous mutually agreed scale factor was within 5% of s_k. If not, it waits for the next mutually agreed scale factor to confirm s_k. Once confirmed, the system inserts a firewall whereby the 3D points and camera poses in the bundle cache are scale corrected. The frames preceding the bundle cache are now fixed and play no further part in future estimation (in effect, their data structures are emptied). The ground detection and firewall estimation require about 30 ms.

Seq	VISO2-S (Stereo)		VISO2-M (Mono)		Ours (Monocular)	
	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)
00	0.0183	1.71	0.0369	12.62	0.0142	7.14
02	0.0111	1.58	0.0194	3.71	0.0097	4.34
03	0.0127	1.81	0.0288	9.05	0.0093	2.90
04	0.0097	1.21	0.0163	7.58	0.0064	2.45
05	0.0160	1.35	0.0575	12.74	0.0107	8.13
06	0.0107	1.11	0.0275	3.71	0.0108	7.56
07	0.0256	1.72	0.1235	25.77	0.0234	9.92
08	0.0155	2.15	0.0369	16.88	0.0122	7.29
09	0.0126	1.57	0.0227	3.94	0.0096	5.14
10	0.0119	1.16	0.0596	29.36	0.0121	4.99
Avg	0.0149	1.65	0.0381	11.53	0.0119	6.42

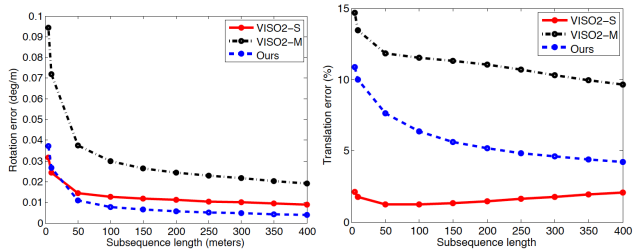
TABLE III
COMPARISON OF ROTATION AND TRANSLATION ERRORS

VI. EXPERIMENTS

We present experimental results on the state-of-art KITTI dataset [5], which is collected in real-world driving situations and provides ground truth for evaluation. The sequences vary in length from a few hundred meters to several kilometers, covering urban, residential, countryside and highway roads. Vehicle speeds range from 0 to 60 kmph at a relatively low frame rate of 10 Hz, which results in large inter-frame motions. Several instances of standing still and frequent presence of other vehicles, bicycles and pedestrians add to the complexity.

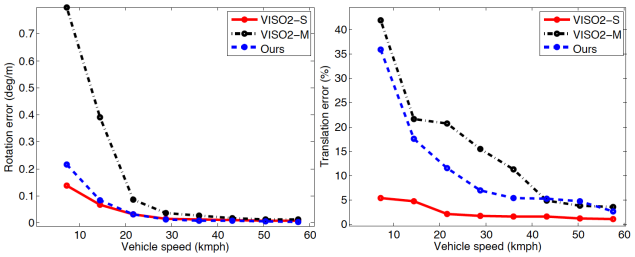
The evaluation metrics are provided by Geiger et al. [5], based on an extension of those proposed by Kümmerle et al. in [10]. Rotation and translation errors are reported as averages of all relative transformations at a fixed distance, as well as functions of various subsequence lengths and vehicle speeds. For timings, our experiments are performed on a laptop with Intel Core i7 2.40 GHz processor with 8GB DDR3 RAM and 6M cache. The main modules occupy three parallel threads as depicted in Sections III-V, while ground detection for scale correction occupies a thread of its own.

We compare the performance of our system to state-of-the-art stereo and monocular systems associated with the dataset, namely VISO2-S and VISO2-M [6]. Rotation and translation errors relative to ground truth, averaged over all subsequence lengths and vehicle speeds, are presented in Table III. Note that the rotation performance of our system is excellent, resulting in error rates lower than even stereo. The translation performance of our system is also significantlly better than VISO2-M, resulting in error rates under 10% for all sequences except



(a) Rotation error across various distances

(b) Translation error across various distances



(c) Rotation error across various speeds

(d) Translation error across various speeds

Fig. 6. Comparison of our system against state-of-art stereo and monocular systems. Solid red line corresponds to stereo output from VISO2-S, dotted black to monocular output from VISO2-M and dashed blue to monocular output from our system. Averages across all ten sequences are plotted, using the evaluation of [5].

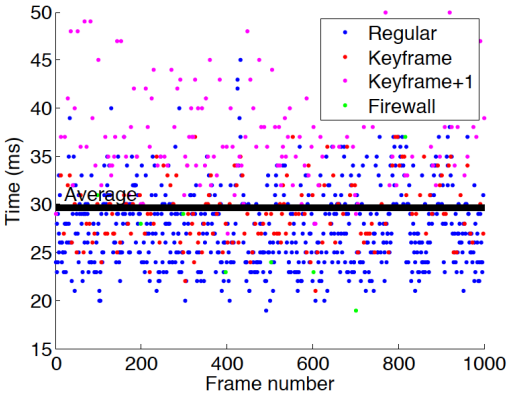
one.[1] This demonstrates the effectiveness of our multithreaded architecture, which adds only thoroughly validated long-tracked 3D points through a parallel epipolar search, as well as allows for repeated local and global bundle adjustments without delay.

Note that images in the dataset are recorded at only 10 Hz, so it inherently favors systems such as VISO2-M [6] based on per-frame 2D-2D matching rather than those tracking 3D-2D correspondences. Nevertheless, our system is robust enough to still produce results that outperform other state-of-the-art. In Figure 6, we compare the rotation and translation errors for the above systems across various subsequence lengths from 5 to 400 meters and speeds from 5 to 60 kmph. The plotted values are averages across all ten sequences in Table III.

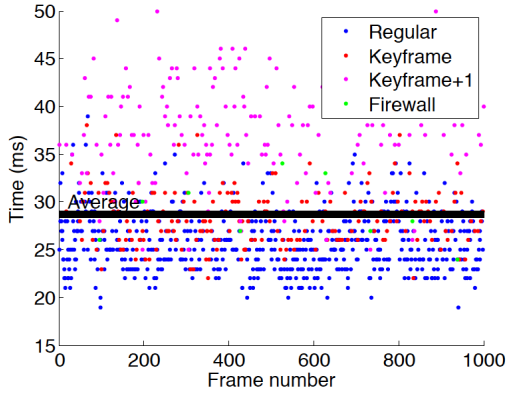
Figure 7 shows the recovered visual odometry path from our monocular system, overlaid with the ground truth, for a few example sequences. This also visualizes the excellent rotation handling of our system, which manages to accurately recover the pose even through several tight rotations in many sequences. The translational error is also small and in most cases, occurs due to difficult imaging conditions at a few isolated frames.

To illustrate our assertion that the system returns real-time pose at an average of 30 fps and a worst-case timing of 50 ms per frame, Figure 8 provides the timing graphs of the system on two sequences. In particular, note that the insertion of keyframes, triggering bundle adjustments or error-correcting

[1]Sequence 01 has been omitted in the above since it involves a long section on the highway at speeds 90 kmph recorded at only 10 Hz, causing all three systems to break down. Translation errors for the three systems are 8% and 33% and 34%, respectively. This is an artifact of the low frame rate of the acquisition, rather than an inherent limitation of the systems.



(a) Sequence 05



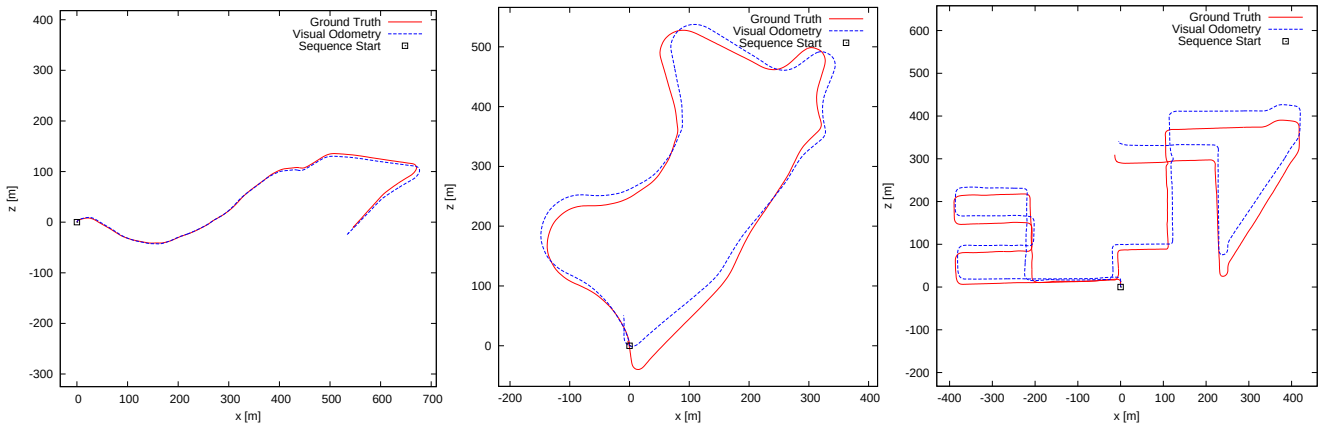
(b) Sequence 08

Fig. 8. The runtimes of our system for various types of frames. Blue denotes steady state frame, red denotes a keyframe, magenta the frame after a keyframe and green denotes a scale correction followed by firewall insertion. The black line is the average time per frame, which correspondes to 33.7 fps for sequence 05 and 34.9 fps for sequence 08.

mechanisms do not result in significant spikes in our timings, which is in contrast to several contemporary real-time systems.

It can also be observed that keyframes are inserted once in about 5 and 6 frames for sequences 08 and 05, respectively. This is expected since a fast moving vehicle will demand new 3D points from the epipolar update module at frequent intervals. It does not affect the performance of our system since the global bundle adjustment triggered after a keyframe finishes before the next frame's pose computation and runs in parallel to it. In fact, keyframe insertion is an opportunity to introduce long-range constraints in the optimization (so long as the epipolar update module can return long enough tracks). Thus, to ensure speed and accuracy, it is crucial for a multithreaded visual odometry system to not only have a well-designed keyframe architecture, but also to have its various modules like pose estimation, epipolar search and various bundle adjustments operating in optimal conjunction with each other.

Also note that scale correction is performed once approximately 100 frames for both the sequences in Figure 8. This illustrates our observations that the proposed system is accurate enough to hold scale for extended periods and that we use local planarity information when available and appropriate, rather than enforcing it as a hard assumption.



(a) Sequence 10 (1201 frames) (b) Sequence 09 (1591 frames) (c) Sequence 08 (4071 frames)

Fig. 7. Monocular visual odometry localization results of our system compared to ground truth. Note the excellent rotation accuracy of our system and only minor translation errors. The need for scale correction was automatically determined based on ground detection and performed only once in over 100 frames.

VII. Discussions and Future Work

We have presented a novel multithreaded system for large-scale, real-time, monocular visual odometry, targeted towards autonomous driving applications with fast-changing imagery. Our key contribution is a demonstration that judicious multithreaded design can boost both the speed and accuracy for handling challenging road conditions. Our system is optimized to provide pose output in real-time at every frame, without delays for keyframe insertion or global bundle adjustment. This is achieved through a novel per-frame epipolar search mechanism that generates redundantly validated 3D points persistent across long tracks and an efficient keyframe architecture to perform online thread-safe global bundle adjustment in parallel with pose computation. Further, we demonstrate that our system is accurate enough to require only occasional scale correction, for which we present an automatic mechanism that detects planarity of the ground to compute reliable scale factors. Currently, the timing bottleneck for our system is the local bundle adjustment. For future work, we will explore multithreaded bundle adjustment optimized for small-sized problems that arise in autonomous driving applications. We also plan to incorporate real-time detection of pedestrians and cars for better handling crowded scenes.

Acknowledgments

This research was conducted at NEC Laboratories America, Cupertino, during the first author's internship in summer 2012.

References

[1] S. Ahn, J. Choi, N. L. Doh, and W. K. Chung, "A practical approach for EKF-SLAM in an indoor environment: fusing ultrasonic sensors and stereo camera," *Autonomous Robots*, vol. 24, no. 3, pp. 315–335, 2008.

[2] B. Clipp, J. Lim, J.-M. Frahm, and M. Pollefeys, "Parallel, real-time visual SLAM," in *IROS*, 2010, pp. 3961–3968.

[3] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *PAMI*, vol. 29, no. 6, pp. 1052–1067, 2007.

[4] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *ICRA*, 2012.

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *CVPR*, 2012.

[6] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *IEEE Intelligent Vehicles Symposium*, 2011.

[7] P. Hansen, H. S. Alismail, P. Rander, and B. Browning, "Monocular visual odometry for robot localization in LNG pipes," in *ICRA*, 2011.

[8] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *ISMAR*, 2007.

[9] ——, "Improving the agility of keyframe-based SLAM," in *ECCV*, 2008.

[10] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009.

[11] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *IJCV*, vol. 81, no. 2, pp. 155–166, 2009.

[12] M. A. Lourakis and A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," *ACM Trans. Math. Software*, vol. 36, no. 1, pp. 1–30, 2009.

[13] D. Nistér, "An efficient solution to the five-point relative pose problem," *PAMI*, vol. 26, no. 6, pp. 756–777, 2004.

[14] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *CVPR*, 2004, pp. 652–659.

[15] J. Oliensis, "The least-squares error for structure from infinitesimal motion," *IJCV*, vol. 61, no. 3, pp. 259–299, 2005.

[16] C. Olson, L. Matthies, M. Schoppers, and M. Maimone, "Stereo ego-motion improvements for robust rover navigation," in *ICRA*, vol. 2, 2001, pp. 1099–1104.

[17] E. Rosten, R. Porter, and T. Drummond, "FASTER and better: A machine learning approach to corner detection," *PAMI*, vol. 32, pp. 105–119, 2010.

[18] E. Royer, J. Bom, M. Dhome, B. Thuilot, M. Lhuillier, and F. Marmoiton, "Outdoor autonomous navigation using monocular vision," in *IROS*, 2005, pp. 3395–3400.

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *ICCV*, 2011, pp. 2564 –2571.

[20] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac," in *ICRA*, 2009, pp. 488–494.

[21] J. Shi and C. Tomasi, "Good features to track," in *CVPR*, 1994, pp. 593–600.

[22] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," in *Robotics: Science and Systems*, 2010.

[23] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *IROS*, 2008.

[24] J. Weng, T. S. Huang, and N. Ahuja, *Motion and Structure from Image Sequences*. Springer Series in Information Sciences, 1993.

[25] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular slam," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.