Monocular Visual Odometry: Sparse Joint Optimisation or Dense Alternation?

Lukas Platinsky[1], Andrew J. Davison[1], and Stefan Leutenegger[1]

***Abstract*— Real-time monocular SLAM is increasingly mature and entering commercial products. However, there is a divide between two techniques providing similar performance. Despite the rise of 'dense' and 'semi-dense' methods which use large proportions of the pixels in a video stream to estimate motion and structure via alternating estimation, they have not eradicated feature-based methods which use a significantly smaller amount of image information from keypoints and retain a more rigorous joint estimation framework. Dense methods provide more complete scene information, but in this paper we focus on how the amount of information and different optimisation methods affect the accuracy of local motion estimation (monocular visual odometry). We propose a new method for fairly comparing the accuracy of SLAM frontends in a common setting. We suggest computational cost models for an overall comparison which indicates that there is relative parity between the approaches at the settings allowed by current serial processors.**

I. Introduction

Monocular visual odometry (VO) is the process of estimating the incremental motion of a camera purely from the image sequence it captures. It is now well understood that VO can form the 'front-end' of a full SLAM system capable of consistent long-term motion estimation and scene mapping. VO, similarly to the larger problem of full SLAM, is a joint estimation problem: to estimate either camera motion or the shape of the scene it sees, we must estimate both. The representations used by monocular VO systems have progressed and changed significantly in the past 20 years. For a long time, almost all monocular VO systems or SLAM front-ends were feature-based, extracting tens to hundreds of salient points[1] and then performing either joint filtering (e.g. [1], [2], [3]) or bundle adjustment ([4], [5], [6], [7]) to minimise the geometric reprojection error.

The value of 'every pixel' methods, which do not first extract features but rather use all pixels in the images, has been long known in visual SLAM [8]. We can also draw parallels with feature-free methods using other sensors such as laser scan matching [9]. However, recently we have seen a strong shift towards semi-dense and fully dense VO algorithms [10], [11] which perform real-time estimation of the locations of many thousands of points and use photometric alignment against all of them to track camera motion. They use alternating tracking and mapping rather than joint filtering or optimisation so that they remain computationally tractable while building much more complete scene representations.

Currently, there are high quality algorithms and open source implementations of both the sparse and semi-dense approaches, and there is an unclear choice between these two quite different paradigms in applications. Following an approach inspired by Strasdat *et al.*'s 'Why Filter?' comparison of filtering and optimisation methods in sparse SLAM [12], [13], we have developed a framework for rigorous comparison of several aspects of dense and sparse monocular SLAM front-ends. We focus on experimentally answering a specific question: which approach offers the most accurate motion estimation as a function of computational cost? In this paper we start by looking into whether the weight of data in semi-dense VO surpasses the more principled joint optimisation of trajectory and map in the sparse approach. Our novel software framework allows us to abstract away the implementation details of systems and concentrate on the fundamental algorithmic differences. We evaluate the systems against high quality synthetic image data with ground-truth trajectory. See Figure 1 for an example of dense and sparse output from our system. Our framework uses optimisation uniformly based on Ceres Solver [14] with a very careful choice of measurement and evaluation methods.

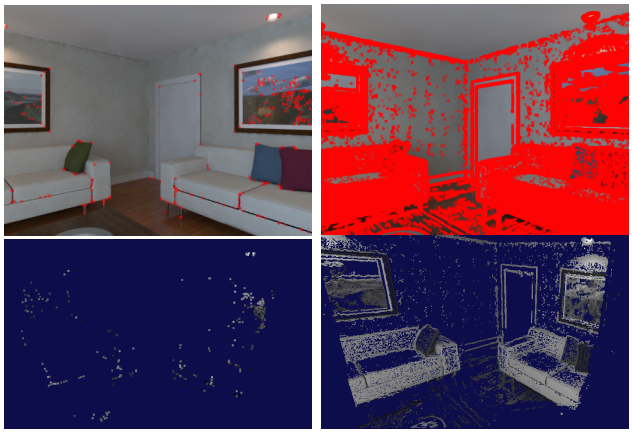


Fig. 1: Differences in the sparse and dense methods. Points extracted from the image (shown in red) for the sparse method and the sparse map are on the left and the points used from the image for the semi-dense method and the semi-dense map are on the right. The greater amount of information about the scene that dense methods provide is apparent. The question is whether they also provide better accuracy for VO systems.

II. Background

Around five years ago, rising commodity processing power, particularly from GPUs, made it apparent that live

[1]Imperial College London

[1]usually keypoints, sometimes edges

dense reconstruction was becoming possible by bringing multi-view stereo depth map estimation into monocular SLAM [15], [16]. These systems were immediately of interest for the increased level of information they provided about the scene, but were not explicitly a move forward from a SLAM point of view because they still relied on feature-based SLAM [5] for camera tracking and map consistency.

However, they were soon followed by DTAM [11], a monocular SLAM system where feature based tracking was replaced by whole image alignment against a dense reconstruction of every pixel in the scene. This work coincided with release of the first commodity depth camera, and great general increased interest in dense, feature-free SLAM approaches. In particular, KinectFusion [17] showed that the 30Hz depth maps measured by Kinect could be fused incrementally into a consistent volumetric signed distance function model, with camera tracking achieved by immediate aligment of new depth maps against the fused model. Point based fusion [18] achieved a similar thing using surfels. There is now a great overlap between visual SLAM methods that use either raw RGB or depth camera data as input, and Newcombe [19] in particular unified these approaches.

Curcially, these new dense systems rather than using joint estimation of camera motion and scene shape demonstrated that very accurate local SLAM can be achieved by *alternation* of tracking and map improvement. Joint optimisation became computationally infeasible for this amount of data, but the arguably inferior computational method using the alternation still provides results competitive with the sparse jointly-optimised methods. When a new frame arrives, the new position of the camera is tracked by aligning it with the current reconstruction which is temporarily assumed to be correct. Then, it is used to improve the reconstruction by assuming the the new tracked camera position is correct.

DTAM and the Kinect-based dense systems required GPU processing, but Engel *et al.* [10] took the important step of showing that the same essential methods could be made less computationally intense if thresholding was used to reduce the number of pixels under consideration. DTAM's main computational expense is the estimation of fully dense depth maps, requiring variational optimisation of a regularised cost function to fill in less textured scene areas with good depth estimates. Engel *et al.*'s monocular semi-dense visual odometry [10] discards pixels with low image gradient, only performs reconstruction of pixels with some appreciable edge magnitude, and tracks the camera against the semi-dense reconstructions this leads to. Still, though, the essential approach of dense SLAM is retained: the number of reconstructed points is high, and tracking and map improvement are carried out in alternation. It is this alternation (and lack of joint estimation such as bundle adjustment) which rather cheaply allows the reconstruction of a number of semi-dense points which is much larger than feature-based approaches. The method of [10] and its later extension into a full largescale SLAM system LSD-SLAM [20] run in real-time on a CPU.

LSD-SLAM is now widely used, but at the same time well-engineered new sparse SLAM systems such as ORB-SLAM [6] and SVO [4] seem to offer similar performance in local VO and SLAM. The main contribution of this paper is providing a way to examine these two different paradigms and give some explanations of why the less precise alternating optimisation method provides results that are competitive with the carefully selected jointly-optimised keypoints. It is the case that from the point of view of an experienced SLAM theorist, the alternation of tracking and map improvement is probabilistically inferior to the joint estimation performed by sparse approaches. Alternating tracking and mapping is equivalent to assuming independence rather than correlation between the estimates of the reconstructed scene elements, an approach which was proven to be inaccurate and discarded early on in SLAM research (e.g. [21], [22]). Our hypothesis is that the vastly increased *amount of data* which dense and semi-dense approaches make use of can make up for (and ultimately outweigh) the inaccuracy introduced by their use of more approximate estimation methods.

III. A New Framework for Fair Comparison

We take inspiration for the current paper from the 'Why Filter?' work of Strasdat *et al.*[12], [13]. This work provided an analysis and comparison of two different approaches to sparse feature-based SLAM. It compared joint filtering and bundle adjustment approaches and suggested that in most cases bundle adjustment optimisation proves to be a more efficient method for the task of precise pose estimation. If that analysis could be characterised as 'MonoSLAM vs. PTAM', then our current work could be considered the next step: 'PTAM vs. LSD-SLAM'.

We use several aspects of the approach of Strasdat *et al.*. The different algorithms to be compared are used to estimate visual odometry on a synthetic dataset with known ground-truth camera motion. We aim to present results for the two methods we are comparing as a graph showing the minimum VO error which can be obtained for a certain computational cost, and to determine whether one method is better than the other for all possible levels of computation or whether there are regimes where different methods dominate. Like [12], [13] our framework is not real-time; it is a tool for evaluating accuracy against ground truth. Computation speed modelling is performed separately.

The definition we adopt of the difference between a 'dense' VO algorithm as opposed to a 'sparse' one has the following key factors:

1) *Choice of scene elements:* A sparse method uses feature detection and reconstructs disjoint scene elements, usually numbered at maximum in the hundreds. A dense method chooses to estimate the locations of a large number of elements which are close enough together that they will tend to join up. It may try to reconstruct a scene element for every pixel and thus be fully dense, or use some weak thresholding and be semi-dense.
2) *Estimation strategy:* A sparse method employs joint estimation of the camera motion and feature locations,

via either joint probabilistic filtering or sliding window optimisation. A dense method performs alternating estimation of the scene structure and per-frame camera motion, decoupling the correlation between scene elements.

By this definition, systems like DTAM [11] and LSD-SLAM [20] are dense; and MonoSLAM [23], PTAM [5], ORB-SLAM [6] and SVO [4] are sparse. It is not our aim here to compare and benchmark specific algorithms and systems. We could have taken, for example, the front-ends of ORB-SLAM [6] and LSD-SLAM [20] (both excellent systems and available open source) and run them head to head on our synthetic test sequences, comparing their accuracy and computation times and plotting the results. However, the details of how those systems are implemented would leave doubt about which differences were down to fundamental algorithmic choices. Also, it is hard within software designed for real-time performance to have full control over tunable settings.

Instead we have implemented a completely new and custom VO framework which can be set to emulate the main aspects of dense and sparse, jointly-optimised and alternating approaches with the bulk of estimation performed in a common way. Note that we do not specify the measurement strategy for sparse methods, and methods which either detect point features on every frame and seek correspondence with map as in [5], [6] or use direct patch alignment as in [23], [4] all fit our definition of sparse. In our experiments we in fact use the latter approach because this unifies measurement and optimisation between dense and sparse approaches.

In the coming sections, we first carefully describe our framework and test sequence, then we proceed with analysing the behaviour of the alternating and joint optimisation methods with different amounts of data used. We conclude by proposing a computational cost model and evaluating the accuracy of the sparse jointly-optimised system and the semi-dense alternating system against a common computational cost domain.

IV. Detailed Methodology

In the implementation of the framework, we compare a simulated joint optimisation and alternating optimisation systems. Our sliding window joint optimisation system is inspired by [4], as we minimise the photometric error on patches, but we perform the minimisation directly in the optimisation rather than optimising geometric error between observations.

The alternating optimisation system is heavily inspired by [10]. Both our optimisation systems share the same patch initialisation and similar variance estimation. The only difference is in the tracking and mapping optimisation part. One of the advantages of having this highly homogeneous yet modular framework is that we can easily test different configurations of the methods and determine where the differences in accuracy between the systems come from.

While 'Why Filter?' [13] was able to abstract away the the actual image measurement process and work on synthesized feature measurements directly, this would be impossible to do in our current work because semi-dense and feature-based methods extract different information from images. The question of correspondence between features and pixels in both methods cannot be ignored, as it is one of the important differentiators between the methods. Finding correspondences between a few hundred highly distinctive corner patches versus pixels (or patches) with gradients possibly in just one direction is completely different. Rather than simulate this sort of correspondence we decided to use more realistic data. Therefore, we turn to photorealistic synthetically generated data and direct photometric error minimisation in our experiments.

A. Notation

The coordinate system W is fixed at the origin of the modelled scene. Each point in the map is expressed as a tuple $(\mathbf{u}_i, \rho_i)$, where $\mathbf{u}_i = [x_i, y_i]^T$ is a 2-D vector with x_i and y_i being the pixel locations in the image in which the point was created, and ρ_i, the estimated inverse depth of the point. By depth we mean in this paper the length of the ray from the centre of the camera to the 3D location of the point. Each point is also assigned a variance σ_i^2 as we (similarly to LSD-SLAM[20]) model the probability distribution of the point's inverse depth as a Gaussian distribution $\mathcal{N}(\rho_i, \sigma_i^2)$. In order to calculate the 3D position of the point in the Euclidean coordinates of the origin frame, we need to backproject the point using the function $\pi'(\mathbf{u}, \rho) : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$. We also define a function $\pi(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^2$ which projects a 3D point $\mathbf{p}$ onto image pixel coordinates.

In the following equations the function $w_{jk}(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^3$ denotes the warping transformation of a 3D point $\mathbf{p}$ in the coordinate frame C_j of the frame j into the coordinate frame C_k of the frame k. Hence, $w_{jk}(\mathbf{p}) = \mathbf{T}_{WC_k}\mathbf{T}_{WC_j}^{-1}\mathbf{p}$, where $\mathbf{T}_{WC_j} \in SE3$ is the 6-DOF transformation from world coordinates into the camera coordinates in frame j. In the rest of the paper we assume $\mathcal{P}$ to be the set of 3D points that were selected for the optimisation and $\mathbf{p}_i = \pi'(\mathbf{u}_i, \rho_i)$ is a 3D point in its origin frame coordinates. $I_j(\mathbf{u}) : \mathbb{R}^2 \to \mathbb{R}$ is the image intensity function at pixel $\mathbf{u}$.

1) Map Representation: The map for both systems consists of a set of points that each corresponds to a patch in the image they were initialised from. Each point is associated with the calculated transformation $\mathbf{T}_{WC_j}$, where j is the index of the 'origin image' from which the point was initialised. Each point carries a set of brightness values contained in the patch of size SxS centred around the points. We choose $S = 5$ for all experiments.

B. Optimisation

1) Photometric Cost Function: The usual approach for sparse systems is to establish feature correspondences between frames and then minimise the geometric reprojection error. Some approaches (e.g. [6]) use only feature detectors to establish these correspondences across frames. Many other approaches (e.g. [5]) use further photometric error minimisation to find the sub-pixel location of the features.

SVO [4] uses only photometric error to find correspondences. We decided to use a single optimisation performed directly on photometric error. As the optimisation is always initialised close to the minimum, we get the best accuracy in joint optimisation by going back to the original data. In terms of our main aim of comparing against dense alternation methods this brings all systems onto the same playing field of optimisation of photometric error, and means that our results are not affected by the details of feature extraction and matching. The photometric error of reprojecting patch i originating in frame j into frame k is:

$$E_p(\mathbf{p},j,k) = \sum_{u,v\in[-S/2..S/2]} (I_j(\pi(\mathbf{p}) + [u,v]^T - I_k(\pi(w_{jk}(\mathbf{p})) + [u,v]^T)^2 \ . \quad (1)$$

We do not use any warping, because the sliding window is small enough not to introduce large shifts in viewpoint. For estimating the intensities and gradients between the pixel centres, we use bilinear interpolation throughout the paper.

2) Sparse vs Semi-Dense Optimisation Problem: In the previous section we have defined the photometric error for a single patch in equation 1. Now, given a set of patches $\mathcal{P}$, we define a photometric error function for that set to be:

$$E(\mathcal{P},j,k) = \sum_{\mathbf{p}_i\in\mathcal{P}} (E_\mathrm{p}(\mathbf{p}_i,j,k)) \ . \quad (2)$$

To identify the set of points $\mathcal{P}_s$ for the sparse methods, we use the BRISK feature detector [24] with the extension described in [25]. This helps us achieve a more homogeneous distribution of the features across the image. To extract set of patches $\mathcal{P}_d$ for the semi-dense method, we select pixels with intensity gradients greater than a pre-determined threshold. We vary the threshold parameters for both methods to obtain different number of points selected.

3) Joint vs Alternating Optimisation: We adapt the equation 2 for the purposes of optimisation:

$$E_m(\mathcal{P},\mathcal{D},j,k) = \sum_{\mathbf{p}_i\in\mathcal{P}} (C(E(\pi'(\mathbf{u}_i,\rho_i'),j,k))) \ . \quad (3)$$

Here, $\mathcal{D}$ is the set of inverse depths from which we pick ρ_i' that we are optimising and $C(e)$ is a Cauchy loss used to add robustness during the optimisation.

The 'joint optimisation' side of our comparison uses sliding window bundle adjustment. This optimisation needs to be anchored in a reference frame to prevent the system being ill-defined. Because we are in monocular case, we also have scale ambiguity. To tackle this, we keep the transformations of the first two frames of the sliding window ($T_{WC_{m-M}}$, $T_{WC_{m-M+1}}$) fixed. We minimise the photometric patch error defined in equation 3. The optimisation for the current frame m over the sliding window of camera transforms $\mathcal{W} = [\mathbf{T}_{WC_{m-M+2}}..\mathbf{T}_{WC_m}]$ and the set of inverse depth estimates $\mathcal{D}$ for the points $\mathcal{P}$ can be written as:

$$\mathcal{D}',\mathcal{W}' = \underset{\rho_i'\in\mathcal{D},\mathbf{T}_{WC_k}\in\mathcal{W}}{\arg\min} \sum_{l\in[m-M..m]} E_m(\mathcal{P},\mathcal{D},j,l) + \lambda \sum_{\mathbf{p}_i\in\mathcal{P}} (\frac{1}{\sigma_i^2}(\|\rho_i' - \rho_i\|)) \ . \quad (4)$$

Here ρ_i is the inverse depth from the optimisation done at the previous frame. The second term is included to provide a prior on the point distribution around its latest estimated position. We minimise the cost function using the Schur algorithm. For our test sequences, we set $M = 20$ as the window size. To apply the depth update, we simply set $\mathcal{D}'$ as the new current estimate for the inverse depths and we update the variances similarly to variance update in [20].

Our 'alternating optimisation' implementation uses alternating tracking and mapping steps instead of joint optimisation. In the tracking step we optimise for the pose of the latest camera frame as follows:

$$\mathbf{T}'_{\mathbf{WC_m}} = \underset{\mathbf{T}_{WC_m}}{\arg\min} \sum_{\mathbf{p}_i\in\mathcal{P}} (\frac{1}{\sigma_i^2} C(E_p(\mathbf{p}_i,j,m))) \ . \quad (5)$$

We use the solver to minimise the sum of patch errors robustified with a Cauchy loss function. Only the currently optimised pose is updated, and all other parameters are kept fixed. The map update step, applied after tracking, is:

$$\mathcal{D}' = \underset{\rho_i'\in\mathcal{D}}{\arg\min} E_m(\mathcal{P},\mathcal{D},j,k) \ . \quad (6)$$

We again optimise only the inverse depths of the points. We keep the x and y coordinates fixed. Otherwise, we would have a greater number of unknowns than observations resulting in an ill-defined system. This is also similar to the way the optimisation is done in LSD SLAM [20]. Note that the alternating method assumes pixel independence, so we can just perform an exhaustive search over the epipolar lines, as this is substantially faster than performing the optimisation via Ceres in this case. After the optimal inverse depth is found, we apply an individual Kalman Filter update to each of the points, just as in LSD-SLAM. To assign variance to the measurements, we use the method proposed in [10].

All reconstruction and tracking optimisations are executed using the Ceres Solver [14]. To keep the semi-dense system close to reality, the mapping optimisation is done by exhaustively searching the epipolar line and finding the point with the minimal patch error E_p within the 2σ inverse depth interval centered around the current inverse depth of the point. The same is done for the initialisation of new patches for joint optimisation.

C. Maintaining the Map

New points are added to the map from each keyframe after its pose has been tracked. Keyframes are created every 20 frames[2]. We first extract a set of potentially new pixels. We then reproject all the points in the map onto the current

[2]Because the motion in the sequences is regular, we can use this simplification instead of a more common overlap-based heuristic.

frame, and select the pixels that do not have any point reprojected onto them. These new points are first added at an arbitrary depth of 2.0m and their depth is refined in the next frame by performing exhaustive epipolar search.

When a new keyframe is created, all points that are are not visible in that frame get deleted from the map. To propagate the previous measurements of a point during reprojection to a new keyframe we initialise the point at the position of the new interest point $x_{\text{new}}, y_{\text{new}}$. The inverse depth of a reprojected point from frame j to frame m is set as $\rho_{\text{new}} = \frac{1}{\|w_{jm}(\mathbf{p}_{\text{old}})\|}$, and $\sigma^2_{\text{new}} = v_r \sigma^2_{\text{old}}$, where $v_r = 1.1$. The purpose of increasing the variance is to accommodate the imperfection in the reprojection. We similarly penalise points with high photometric error at their estimated location during each mapping update.[3]

D. Experiments

We run our experiments on 4 different systems:

1) Sparse jointly optimised system using points $\mathcal{P}_s$ and equation 4.
2) Sparse aletrnating system using points $\mathcal{P}_s$ and equations 5 and 6.
3) Semi-dense jointly optimised system using points $\mathcal{P}_d$ and equation 4.
4) Semi-dense aletrnating system using points $\mathcal{P}_d$ and equations 5 and 6.

1) Scene and Sequence: The sequence we use is a realistically modelled scene from the ICL-NUIM dataset [26]. A sequence of frames along a looped circular trajectory is rendered at VGA resolution using high-precision ray-tracing as in [27] and [26]. Example images are shown in Figure 1.

2) Modelling Image Noise: We add realistically modelled image intensity noise to the synthetic images. This is based on a model described in [27]. We assume a linear approximation for irradiance and apply the noise model to the intensity values. The parameters we use for the noise generation were chosen after visual inspection of the images to simulate realistic camera noise for an indoor setting.

3) Algorithm Initialisation: The systems are initialised by keeping poses fixed and running just the mapping part for the first 20 frames of the sequence while providing the poses of the camera coming from the VO part of LSD-SLAM[20]. After this initial short sequence of frames, the tested systems perform tracking and mapping on their own. We use the next 100 frames for error measurement.

We provide initial estimates of the camera poses for all subsequent frames. In order to measure the accuracy of the optimisations rather than the accuracy of different coarse-to-fine approaches, we only perform optimisation at the finest level. To provide an initialisation that is close enough the ground truth for optimal convergence, we advance the system's pose estimate $\mathbf{T}_{\mathbf{WC_{m-1}}}$ by the difference calculated by LSD-SLAM VO[20] $\tilde{\mathbf{T}}_{\mathbf{C_{m-1}C_m}}$ between the last and new frame m: $\hat{\mathbf{T}}_{\mathbf{WC_m}} = \tilde{\mathbf{T}}_{\mathbf{C_{m-1}C_m}} \mathbf{T}_{\mathbf{WC_{m-1}}}$. This brings realistic noise into the initial pose estimates.

[3]We empirically found these settings to give most precision across the tested systems.

4) Quantifying Error: Quantifying the performance of VO or any other type of incremental motion estimation must be carefully performed, and we strongly believe that the commonly reported measures such as RMSE on a whole estimated absolute trajectory or 'percentage drift' are not meaningful (the latter because the ratio of drift to distance travelled will always depend on the distance travelled). Instead we use a relative measure; the average difference between estimated and ground truth motion between every pair of consecutive frames:

$$\mathbf{E_t} = \sum_{k \in [1..F]} \|\mathbf{t}_{C_{k-1}C_k}\| / F \ , \tag{7}$$

where $F = 100$ is the number of frames used for measurements and $\mathbf{t}^r_{C_{k-1}C_k}$ is the translation part of a relative transform between the last two frames $\mathbf{T}_{\mathbf{C_{k-1}C_k}} = \mathbf{T}_{WC_k} \mathbf{T}^{-1}_{WC_{k-1}}$. We use a similar measure to quantify orientation error:

$$\mathbf{E_r} = \sum_{k \in [1..F]} \|\mathbf{r}_{C_{k-1}C_k}\| / F \ , \tag{8}$$

where $\mathbf{r}_{C_{k-1}C_k}$ is the rotation component of $\mathbf{T}_{\mathbf{C_{k-1}C_k}}$ expressed in angle-axis notation, so that the norm of $\mathbf{r}_{C_{k-1}C_k}$ is the rotation angle in radians. As our results show, these errors are normally highly correlated.

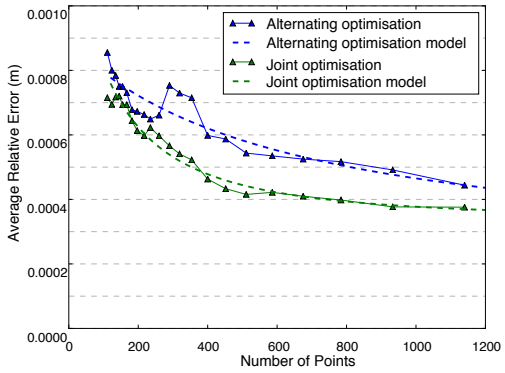
V. Experimental Results

We first present results on camera pose accuracy for the dense and sparse methods respectively in terms of the number of points used by each method. We will then combine these results with computational cost models in the next section.

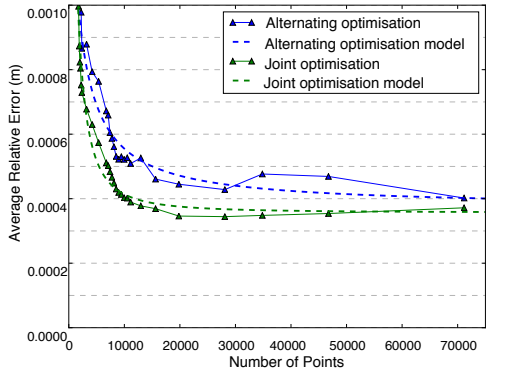
A. Accuracy vs. Number of Points

We vary the number of points over a wide range for each method to obtain an average relative error measure for different map densities. The results can be seen in Figures 2a and 2b. We are mostly interested in the 'sparse jointly-optimised' and 'semi-dense alternating' parts of the graphs. These correspond to the systems that are used in real applications. The 'sparse alternating' and the 'jointly-optimised semi-dense' systems are infeasible in practice either because of inadequate results, or large computation requirements, respectively, and are used only to analyse the relative behaviour of joint and alternating optimisation.

As expected, we measure reduced average errors as the number of points used increases. Ultimately, both jointly-optimised systems reach very similar levels of accuracy slightly below 0.4mm average relative error per frame, though with the use of very different numbers of points — around 500 points at 0.38mm accuracy in the sparse case, and around 10000 points at 0.35mm accuracy in the semi-dense case. We observe similar behaviour for the alternating case with a slightly higher errors of 0.45mm and 0.4mm respectively. These figures are reassuringly similar to the operating points chosen by real sparse and semi-dense systems.



(a)



(b)

Fig. 2: Average relative error against number of points. The raw results from our experiments (solid lines) are shown together with fitted curves (dotted lines). We observe that the joint optimisation method decreases rapidly in both cases, but as more data are used in the optimisations, the alternating method slowly catches on. In (a) we observe a curve similar to [13] for the joint optimisation. The alternating optimisation error decreases at a slower rate, but we can see that the difference gets smaller with more data. In (b) the error rapidly decreases for both methods till around 15,000–20,000 points. We again observe a slightly steeper decrease for jointly-optimised minimisation compared to alternating optimisation, but this difference decreases with more data.

The similar level of accuracy of the two main methods is an important find, as this confirms our hypothesis that the greater weight of data used in the semi-dense method makes up for the simplification made by using alternating rather than joint optimisation. This can be further backed up by the fact that the differences between the errors of the jointly-optimised and alternating methods slightly decreases with adding more data in both figures.

Direct comparison of the commonly used 'sparse joint' and 'dense alternating' methods is still not possible, as each method has a substantially different computational cost per point. We address this by introducing a computational cost model in the next section.

B. Accuracy vs. Computational Cost

As we have explained, our software framework is designed for accuracy analysis, and we do not expect its computational speed in different configurations to be representative of the optimised real-time systems that it represents. For this reason we rely on mathematical models to assess the computational performance of the competing algorithms tested, anchored by measurements from real systems.

1) Estimating Computational Cost: The theoretical complexity of the joint optimisation was shown to be linear in the number of points in [12], [13]. The alternating methods also perform a constant number of operations on each point during the tracking and mapping phases. Because we keep the number of frames fixed in the local sliding window, the only variable affecting the computation complexity within each of the methods is the number of points they use.

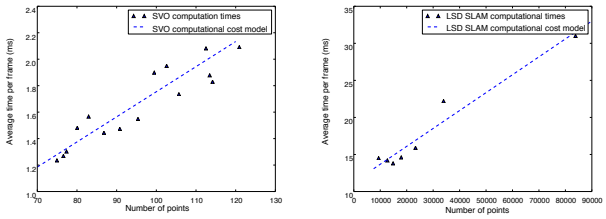
Theoretical computational cost analysis unfortunately cannot capture the subtle differences between the two different methods, as they only differ by a constant factor. They both scale linearly, but the slope is different due to the different amount of computation spent on each point. Tuning the systems or calculating the exact number of operations is also infeasible in practice, as these would vary widely across possible implementations.

Therefore, we model the computational cost as a linear function in the number of points $c(N) = aN + b$. To get values for the constants a and b, we use least squares to compute models that best fit the real timings performed on the SVO and LSD-SLAM open source real-time systems. These measurements and our model fits are shown in Figure 3. For SVO, we obtain $c_{\text{SVO}}(N) = N * 0.0190\text{ms/point} - 0.1428\text{ms}$.[4] For LSD SLAM, our model is $c_{\text{LSD}}(N) = N * 0.0002\text{ms/point} + 11.2653\text{ms}$. This suggests that LSD SLAM spends roughly 100 times less computation power per point than SVO. All timings were done on the same desktop computer with an Intel Core i7 960 CPU running at 3.20GHz. It should be taken into account that while we do believe that these models are representative of the relative performance of sparse and dense approaches, these two real-time systems both have various types of approximation and low-level optimisation and other implementations will have different characteristics. Therefore we should consider the computational cost models as approximate.

C. Comparative Results on Accuracy as a Function of Cost

Now we can finally plot both systems' accuracy against their computational costs. We scale the horizontal axes in Figures 2a and 2b by the models described in the previous section, so that number of points can be interpreted in terms of computational cost. The results can be seen in Figure 4. As we increase the computational power available in the SLAM system, semi-dense alternating methods can reach similar performance as the jointly-optimised sparse methods. (For

[4]The very small negative constant may seem odd, but it is just a minor artefact of the best-fit model and does not affect the results of the comparison.



(a) (b)

Fig. 3: Runtime measurements from open source real-time systems which we use for computational cost modelling, where we plot average processing time per frame as a function of the number of points used. (a) sparse system SVO [4]; (b) semi-dense system LSD-SLAM [20]. The straight line fits shown are the processing cost models which we bring into our comparative results in Section V-C.

completeness, we show the behaviour for rotation estimation in Figure 5.)

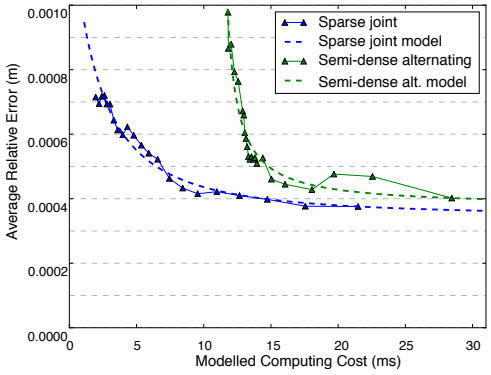
This curve shows relative parity of the methods in the regimes which are normally used in real-time operation. We believe that our results strongly suggest that the new paradigm of semi-dense alternation is well on the way to being the new norm for VO systems. Our results have assumed a single serial processing resource, but dense alternation methods benefit more strongly from parallelisation than sparse methods and the increasing parallelism available in commodity processors will only aid these approaches in the future. Dense methods also allow bringing in prior knowledge about surfaces, which we believe can push the accuracy of the dense and semi-dense methods further.

VI. Conclusion

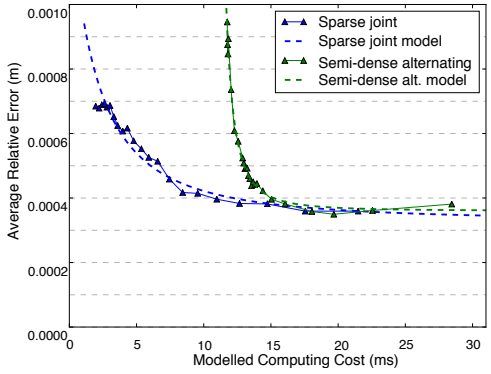
We have introduced a framework for fair comparison between sparse joint optimisation and dense-alternation methods for visual odometry. It keeps implementation details to a minimum and directly compares the two paradigms. From our experiments using a synthetic dataset we can see that both methods are very similar in their peak performance and that the large amount of data that semi-dense methods use makes up for the loss in accuracy coming from the efficient but simplified alternating optimisation.

For future work, we plan to investigate the systems further. Visual-inertial fusion is now the norm in most deployed visual odometry systems, in particular where rapid camera motion is expected. An important extension of this work would be to see how the addition of an IMU would affect the conclusions in this paper.

Another direction of this work is evaluating the advantages that imposing smoothness priors in the case of semi-dense data can bring. We believe that surface or even object priors can push the accuracy and efficiency of dense methods ultimately far beyond even a fully jointly optimised approach.

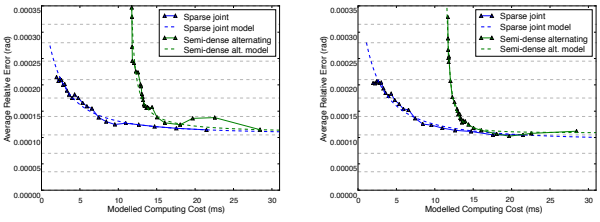


(a)



(b)

Fig. 4: Average relative translation error against modelled computational cost for both dense and sparse approaches. We can see that while sparse joint optimisation gives better results at low computational cost, the semi-dense method catches up when around 15ms per frame of computation time is available. The two plots show (a) raw results and (b) the results when the experiment is repeated with ground truth poses used in the initialisation sequence. More precise initialisation helps the alternating optimisation achieve slightly better accuracy and provide results on par with the jointly-optimised method.



(a) (b)

Fig. 5: Average relative rotation error against modelled computational cost when the algorithms are initialised using (a) LSD-SLAM[20] or (b) ground truth. The shape of these plots are almost identical to those for translation error.

REFERENCES

[1] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics (T-RO)*, vol. 24, no. 5, pp. 932–945, 2008.

[2] A. J. Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003.

[3] E. Eade and T. Drummond, "Monocular SLAM as a Graph of Coalesced Observations," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[5] G. Klein and D. W. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 5, pp. 1147–1163, 2015.

[7] D. Nistér, O. Naroditsky, and J. Bergen, "Visual Odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[8] A. I. Comport, E. Malis, and P. Rives, "Accurate Quadri-focal Tracking for Robust 3D Visual Odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[9] J.-S. Gutmann and K. Konolige, "Incremental Mapping of Large Cyclic Environments," in *International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1999.

[10] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.

[11] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-Time," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[12] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-Time Monocular SLAM: Why Filter?" in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[13] ——, "Visual SLAM: Why filter?" *Image and Vision Computing (IVC)*, vol. 30, no. 2, pp. 65–77, 2012.

[14] S. Agarwal, M. K., and Others, "Ceres solver," http://ceres-solver.org.

[15] R. A. Newcombe and A. J. Davison, "Live Dense Reconstruction with a Single Moving Camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[16] J. Stuehmer, S. Gumhold, and D. Cremers, "Real-Time Dense Geometry from a Handheld Camera," in *Proceedings of the DAGM Symposium on Pattern Recognition*, 2010.

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[18] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion," in *Proc. of Joint 3DIM/3DPVT Conference (3DV)*, 2013.

[19] R. A. Newcombe, "Dense Visual SLAM," Ph.D. dissertation, Imperial College London, 2012.

[20] J. Engel, T. Schoeps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[21] J. A. Castellanos, J. D. Tardós, and G. Schmidt, "Building a global map of the environment of a mobile robot: The importance of correlations," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[22] A. J. Davison, "Mobile Robot Navigation Using Active Vision," Ph.D. dissertation, University of Oxford, 1998.

[23] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29, no. 6, pp. 1052–1067, 2007.

[24] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary robust invariance scalable keypoints," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[25] S. Leutenegger, "Unmanned solar airplanes: Design and algorithms for efficient and robust autonomous operation," Ph.D. dissertation, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 22113, 2014.

[26] A. Handa, T. Whelan, J. B. McDonald, and A. J. Davison, "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. [Online]. Available: http://www.doc.ic.ac.uk/ ahanda/VaFRIC/iclnuim.html

[27] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison, "Real-Time Camera Tracking: When is High Frame-Rate Best?" in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.