Depth Prediction for Monocular Direct Visual Odometry

Author Name
line 1 (of Affiliation): dept. name of organization
line 2: name of organization, acronyms acceptable
line 3: City, Country
line 4: Email: name@xyz.com

Author Name
line 1 (of Affiliation): dept. name of organization
line 2: name of organization, acronyms acceptable
line 3: City, Country
line 4: Email: name@xyz.com

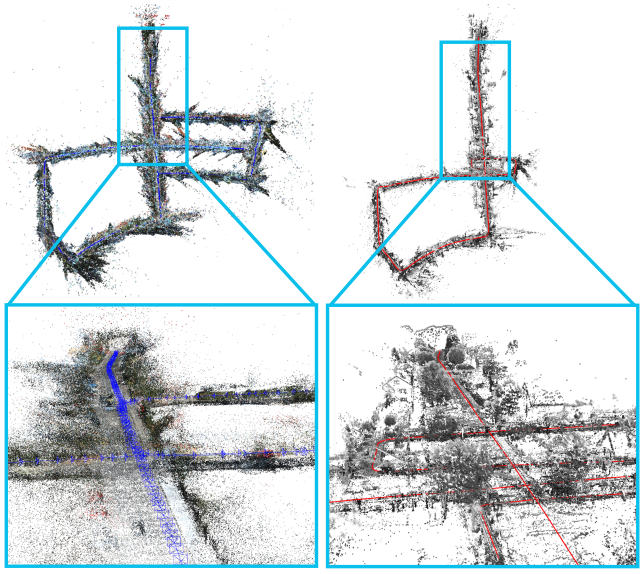
***Abstract*—Depth prediction from monocular images with deep CNNs is a topic of increasing interest to the community. Advances have lead to models capable of predicting disparity maps with consistent scale, which are an acceptable prior for gradient-based direct methods. With this in consideration, we exploit depth prediction as a candidate prior for the course initialization, tracking, and marginalization steps of the direct visual odometry system, enabling the second-order optimizer to converge faster into a precise global minimum. In addition, the given depth prior supports large baseline stereo scenarios, maintaining robust pose estimations against challenging motion states like in-place rotation. Inspired by LDSO [1], we employ indexed image features and a covisibility graph with online loop closure to refine our pose graph. The experiments on KITTI demonstrate that our proposed method achieves state-of-the-art performance compared to previous traditional direct visual odometry counterparts.**

***Keywords*-depth prediction; visual odometry; deep learning; visual SLAM;**

I. Introduction

Direct methods [2][3][4] have received more attention in recent years after the popularization of global shutter and high resolution cameras. Direct image alignment, such as optical flow and it's variants, optimize a photometric error to estimate camera ego-motion and are well known for their speed and precision. Feature extraction from images are not required for such methods, and thus, they are able work in feature deficient environments (even on pure gradient images). However, the inclusion of original images in the photometric loss function greatly increases the degree of non-convexity, which is fatal to gradient-based direct methods. Moreover, photometric consistency is a very strong assumption that is not guaranteed in practice.

Many models have been proposed to increase the efficiency and robustness of direct methods. Illumination-robust cost [5] and rigid body mutual information [6] have been proposed to compensate for drastic illumination changes between frames. The photometric calibration model [7] has been employed as a service to estimate camera exposure time, gamma reaction, and vignette to adapt for auto-exposure cameras. Further, the adoption of reprojection residual patterns have been shown to smoothen the photometric error manifold, quickening the rate of convergence for gradient-based optimization.



(a) Ours (b) DSO [4]

Figure 1: Example mapping results on KITTI seq 05. Direct visual odometry with the depth prior information compared to DSO. Our method successfully maintains constant scale and precision when estimating camera poses for in-place rotations. Note the apparent scale-drift of DSO.

During initialization, LSD-SLAM [3] and DSO [4] perform an exhaustive search on a set of empirical predefined initial poses. This bulky initialization process deteriorates the real-time performance of these systems. In addition, without prior depth or pose information the optimizer typically initializes the inverse depth and pose with identity matrices, constructing one large arrow shaped Hessian matrix from them before solving their delta updates jointly. As the pixel depth and camera pose are initially coupled, one degree of freedom is left in the nullspace [4], causing the optimizer to converge to an arbitrary scale at each initialization. The lack of global constraints (e.g. loop closure) in the tracking thread (local bundle adjustment) allows for estimation error to accumulate, which ultimately drifts the scale of both pose and depth estimates [8].

Deep learning based methods have demonstrated remark-

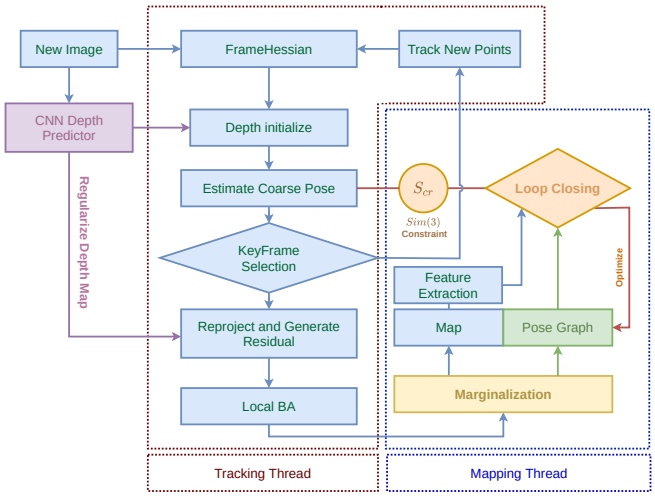


Figure 2: CNN-DVO pipeline.

able progress on maintaining consistent scale for both depth prediction and camera pose estimation. It is noteworthy to mention that the majority of deep learning models are actually predicting inverse depths (disparity maps). With proper scaling, we can seamlessly feed this inverse depth prior into the depth filter module in the direct visual odometry system. This addition can aid in restraining the effect of scale-drift for successive tracking and mapping tasks.

In this work, we present a real-time monocular visual odometry system with an auxiliary deep depth predictor. Since applying the Schur complement for marginalization is equivalent to solving the marginal distribution over camera pose given a depth prior, the joint optimization can thus be broken down into two stages: **a)** solve the pose from coarse depth prior; **b)** correct depth from the pose given by **a**. Unlike Zhao et, al. [9] and Loo et, al. [10] which only use the depth for direct image alignment, we also incorporate the depth priors into the tracking and marginalization backend. We argue that the use of a deep network predicted pose to initialize direct image alignment is unnecessary, since the decoupled pose can be recovered from a coarse depth prior efficiently. In the ablation study of this paper, the pose network initializer is shown to provide only minor improvements to our final results. Inspired by LDSO [1], we divide the direct visual odometry problem into two main parts: coarse tracking and map refinement. Instead of performing global photometric bundle adjustment, we select keyframes based on marginalization results, and form a pose graph with the BoW database (Bag of Words) of ORB features extracted from each keyframe. These corner features are uniformly selected from the image space to ensure that the mapped points can be reproduced in a later frame for proper loop closure. In summary, our contributions can be listed as the following:

- We present a unified monocular direct visual odometry framework that incorporates deep CNN predicted depths as a prior for initialization, local bundle adjustment optimization, and loop closure (Fig. 2), mitigating the effects of scale-drift for pose estimation and map building.
- An enhanced point selection strategy (Fig. 3) that more evenly samples tracking points in the image space to reduce the clustering of points near high gradient regions and promote robustness of pose estimation to extreme motion states such as in-place rotations.
- We demonstrate that the introduction of depth priors achieves high accuracy during large scale tracking sequences in a public dataset compared to state-of-the-art direct methods.

II. Related Works

A substantial amount of error in the scale of estimations can be attributed to a lack of physical constraints available to VO systems. This deficiency is often remedied by integrating more sensors into the system - for instance, an IMU which provides constraints in the form of inertial measurements. The stereo version of DSO [11] and inertial measurement [12] have substantially improved the scale of estimations, however, the cost of adding a new sensor and the multiple signal alignment makes their deployment infeasible for mobile robots with limited hardware.

Attempts to solve visual odometry with the help deep learning methods have received more and more attention. SfMLearner [13] started the unsupervised architecture based on back-warping amongst consecutive image frames. DVSO [14] then developed the same idea on left-right warping and generated virtual stereo pairs to estimate monocular depth and camera poses. Yet the lack of normalization in their depth predictions increased the likelihood of divergence. DDVO [15] addressed this issue in their model by normalizing the output of the depth CNN before feeding it into the loss function. Furthermore, they tried to integrate the direct image alignment backend and the depth estimation pipeline to jointly optimize their pose and depth networks which were coupled with a shared encoder. Monodepth2 [16] further investigated the occlusion effect and adopted a coarse-to-fine trick by predicting depth and pose in different scale spaces. All the above methods propose an independent pose network to produce the consecutive frame warping transformation matrix; generalizability of the pose network remains an issue regardless of the training procedure, which is not the case with traditional visual odometry methods. Consequently, DDSO [9] and CNN-SVO [10] turned-in to utilize the depth prediction results from a deep CNN to help initialize the visual odometry system, and then use traditional geometric methods to estimate camera poses. Note that while the initialization plays an important part in VO systems, scale-drifts will still carry on during the tracking thread. Both DDSO and CNN-SVO assume that the predicted depths are accurate across the whole image, but

in actuality, depths predicted at further distances tend to be noisy. This uncertainty can be problematic, as the relatively long life span of distant map points more heavily contribute to pose estimation in a local sliding window. In this work, we carefully model uncertainty and propose an uncertainty-aware inverse depth variance propagation and fusion.

III. Method

We start with the complete pipeline (Fig. 2) of our CNN-DVO in Sec. III-A, and briefly illustrate our depth estimation model in Sec. III-B for initialization, tracking and loop closure in Sec. III-C, Sec. III-D and Sec. III-E, respectively.

A. Complete Pipeline

We begin by describing our strategy for selecting the initial tracking points in the system. Point selection is a fundamental component of any DVO system and holds a strong bearing on initialization and tracking robustness. Gradient-based direct methods favor the selection of points in high gradient regions, such as corners and edges, as they contribute more to the photometric reprojection error, and thus, can boost the efficiency of bundle adjustment. However, the potential clustering of points near these corner or edge-like features may expose the system to high estimation error when most high gradient features are grouped in the same sub-section of a given image (Fig. (7) of [11]). To reduce estimation error under such circumstances, we propose a new point selection method which takes a more even spread of points into consideration while also sampling near high gradient features (as shown in Fig. 3).

We adopt the keyframe based sliding window approach to estimate the local camera poses and refine the sparse depth maps. Within the local sliding window, the $SE(3)$ keyframe pose and inverse depth are parameterized and composed into a minimization problem as formulated in [1], Eq. (1). A central idea of our paper is the proposed use of a deep CNN predicted depth in the current camera image to estimate camera pose in the next frame. We utilize the pre-trained Monodepth2 [16] as our depth predictor which helps to refine the pose estimation as shown in Fig. 2.

By continuously propagating depth into each tracking frame and refining it in the local bundle adjustment, we can easily lock down the inverse depth variance upper bound. We perform Gauss-Newton optimization for each selected point to refine the optimal reprojection locations in the new frame and recover the new inverse depth in the epipolar line [17]. Given the inverse depth prior, the search interval is limited by $d \pm 2\sigma_d$ where d is the inverse depth, σ_d is the standard deviation which is initially set as linear combination of the inverse depth prior [10] and observation variance [17].

$$\sigma_d^2 = \lambda(\frac{d'}{6})^2 + (1-\lambda)\sigma_{d,obs}^2 \tag{1}$$

Here λ is the weight (empirically set to 0.54), d' is the predicted inverse depth from the deep CNN, and $\sigma_{d,obs}^2$ is

Figure 3: Top image is the original RGB input, middle image demonstrates our point selection result (colored by distance), bottom image shows the DSO point selection result. Unlike DSO which selects points depending on the local image gradient, we increase the sample rate in low gradient regions to maintain an even spread of tracking points in the image space. We also drop all points selected in the upper section of the image where the predicted depths are noisy.

defined in Eq. (10) [17]. The empirical setting provides adequate room for noisy depth predictions to converge.

We incorporate the depth prior into our photometric equation as a depth residual, which penalizes the deviations in inverse depth between keyframes, and properly scales the estimated transformations between them. The residual in turn is defined as:

$$r_{id} = (I_j[p'(T_i, T_j, d', c)] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i(p) - b_i) \tag{2}$$

Where T_i, T_j are camera poses, d': the depth prior, c: camera intrinsic, a_i, a_j, b_i, b_j: affine brightness coefficients. I_i, I_j are two keyframes, p' in I_j are projected points from p in I_i given the transformation above. The windowed optimization part is similar to DSO; please refer to Section 2.3 in [4] for implementation details.

In loop closure, we extract the BRIEF descriptors on each keyframe and maintain a DBoW [18] databse for loop detection query, then initialize the transformations by applying RANSAC PnP and optimize the $Sim(3)$ transformation through 3D-2D geometric constraints. We only update the optimized pose in the front-end thread (visualizer) to avoid corrupting the local sliding window by modifying the backend pose graph, and to keep the tracking thread running efficiently.

A notable feature of our method is the ability to recover the scale of absolute pose constraints between the loop

candidate and the current frame. During global pose optimization, we fix the pose estimations of all reference frames and apply the aggregated correction transformation updates only in the front-end visualizer. Since the pose Hessian prior is carried on in each frame in the local sliding window, modifying the pose in the backend will corrupt the local windowed bundle adjustment, and thus, for thread safety and tracking robustness, the global map optimization only occurs upon manual triggering or when tracking concludes.

B. Deep CNN depth estimation

Monocular depth estimation has been well explored by the community and many existing methods show impressive results in public datasets such as KITTI and CityScapes. We use pretrained Monodepth2 [16] checkpoints which have been fine-tuned for KITTI raw stereo data and perform the real-time depth inference on GPU (20 ms). The predicted depth priors are disparity maps, which are analogous to inverse depth scaled by focal length and baseline. In practice, the disparity maps were scaled by 0.1 to ensure numerical stability.

We observed relatively high estimation errors from the deep CNN at mid-to-long distances as depicted in Fig 4. This is a result of predicting disparity maps (equivalent to inverse depth map), where long range pixel-wise depths are encoded as small decimal valued inverse-depths (disparities) which contribute very little to the final loss. This source of high estimation error is prevalent in many methods that predict disparity maps [16][13][15], and is not the case in supervised learning approaches that use ground truth depth labels.

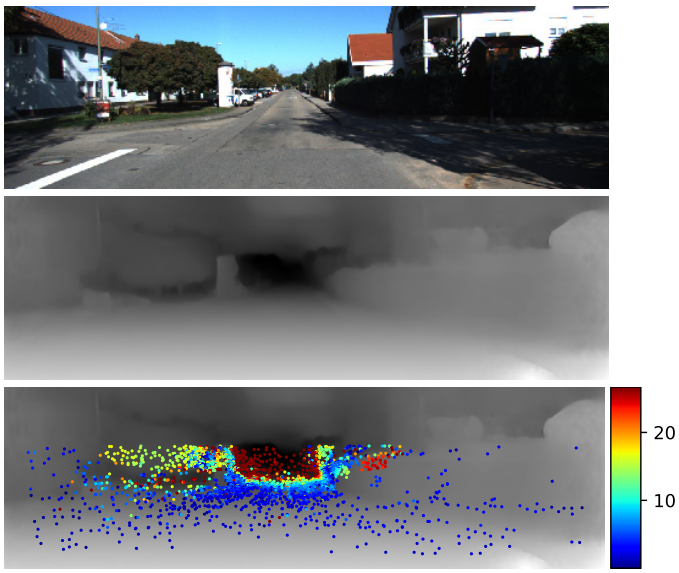


Figure 4: Top image is the original RGB input, middle image is the dense depth predicted from Monodepth2 and is bilinear upsampled by 2x, bottom image is the depth error between predicted dense depth map and successfully tracked map points. The depth error is in meters.

Furthermore, the low disparity of the sky usually leads to noisy depth estimates from the deep CNN in those regions. This noise may corrupt the depth prior, and hence, we remove all tracked points in the upper 30% of the image, as illustrated in Fig. 4. We use the finely predicted parts of images to initialize the depth filter and to track map points - this process is thoroughly explained in the follow sections.

C. Depth estimation for initialization

We employ a depth filter to initialize our inverse depth estimation for each selected immature point. We express the optimal inverse depth as a function of our depth prior and geometrical projection inputs:

$$d^* = d(I_0, I_1, d', \xi, \pi) \tag{3}$$

Here d^* is the optimal inverse depth, d' is the inverse depth prior defined in Eq. (1), ξ is the relative transformation matrix and π is the projection function. The error variance $\sigma^2_{d^*}$ is thus given by:

$$\sigma^2_{d^*} = \alpha^2 \left(c_d + c_d \frac{J_d \Sigma J_d + J'_d \Sigma J'_d}{J_d \Sigma J_d} + \sigma^2_d \right) \tag{4}$$

Where c_d is the normalization constant (empirically set to 0.2), J_d is the Jacobian of d ($J_d = [dx, dy]^T$, as shown in Fig. 5), J'_d is the conjugate Jacobian of d: $J'_d = [dx, -dy]^T$, and Σ is the 2×2 input-error covariance ($\Sigma = [I_x, I_y]^T [I_x . I_y]$), α is defined in Eq. (11) in [17], σ_d is defined in Eq. (1). We initialize our given inverse depth prior to $\mathcal{N}(d', \sigma^2_d)$, and perform a Gauss-Newton optimization to shrink the inverse depth upper bound for each candidate map point:

$$r_{id} = \sum_{i \in \mathcal{P}_i} ||I_1[\pi(p_i, d, \xi)] - e^a I_0(p_i) + b||_\gamma \tag{5}$$

Here r_{id} is the inverse depth residual, γ is huber norm, $\mathcal{P}_i$ is the residual pattern, π, d and ξ are defined in Eq. (3), a, b are affine coefficients defined in Eq. (2). Likewise, the Jacobian of inverse depth can be derived as implemented in [4]:

$$J_{id} = I_x d_x + I_y d_y \tag{6}$$

Where I_x, I_y are image gradients at a given pixel location. d_x and d_y are the search region dimensions along the epipolar line, as shown in Fig. 5.

Since the parameterized inverse depth is a scalar, the Hessian of the inverse depth and error term are also scalars: $H_{id} = J^2_{id}$, $b_{id} = -J_{id} r_{id}$, respectively. The optimal pixel coordinate p will be updated through $p* = p - \frac{b}{H}$, and the inverse depth is consequently recovered from p according to the projection equation in [3]. Note that the optimizer will converge into the optimal pixel coordinate with respect to the inverse depth, thus, the inverse depth can be recovered from the projection equation as illustrated in [19].

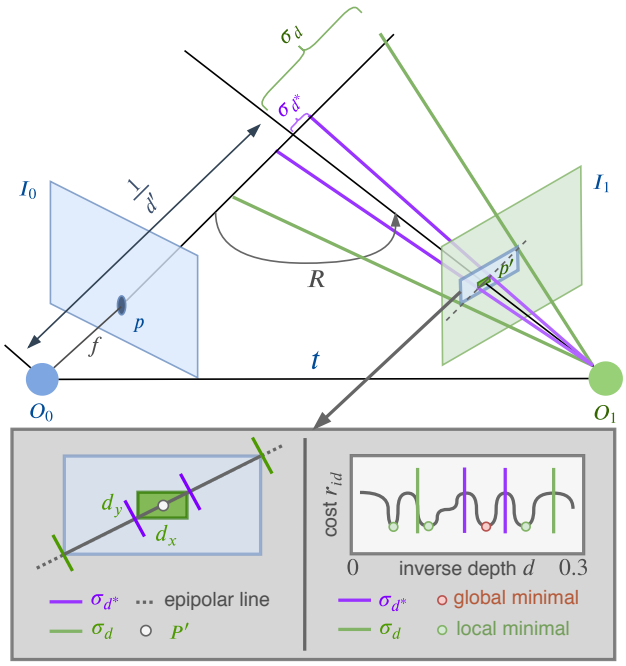


Figure 5: The uncertainty propagation of inverse depth. f is the focal length, I_* are the images, d' is the inverse depth prior, σ_d^2 is the inverse depth variance, $\sigma_{d^*}^2$ is the optimal inverse depth variance. d_x and d_y are search region dimensions on the epipolar line. The depth prior narrows the search region and makes our method tenable to large baseline stereo problems, depicted in Fig. (3) of [17] (i.e. strong rotation motions exemplified in Fig. (7) of [11])

Not only does the depth prior shrink the range of the search in the epipolar line, but it also initializes the starting point in the neighbourhood of the local minimum which increases the convergence rate of the second order optimizer. Further, the observation variance fused from the depth prior can then be propagated to new frames to infer the new inverse depth and camera pose.

D. Depth propagation for tracking

Ideally, the predicted depth from the deep CNN should be precise everywhere. However, we observe that the error near high gradient areas such as object boundaries are often quite large, despite the introduction of an edge-aware term in the commonly used smoothness loss. This extra noise is modelled in the inverse depth variance of new tracking frames, once the camera position of the frame has been estimated. The new inverse depth d_1' is defined as:

$$d_1'(d_1, d') = \mathcal{N}\left(\frac{\sigma_{d_1}^2 d' + \sigma_{d'}^2 d_1}{\sigma_{d_1}^2 + \sigma_{d'}^2}, \frac{\sigma_{d_1}^2 \sigma_{d'}^2}{\sigma_{d_1}^2 + \sigma_{d'}^2}\right) \tag{7}$$

where d_1 and σ_{d_1} are defined in Eq. (14) and Eq. (15) from [17]. Note that this is equivalent to the update step in a Kalman filter where the noisy observation $\mathcal{N}(d', \sigma_{d'}^2)$ is fused with the geometrical propagation prior $\mathcal{N}(d_1, \sigma_{d_1}^2)$.

E. Depth prediction for marginalization and loop closing

We build our the loop closure and pose optimization backend from LDSO [1] and feed the extra depth prior measurements. Since we already have the converged inverse depth in active map points in the library and have the close estimation of depth prior by deep CNN, the constraints are performed only on 3D-3D corespondence, the $\mathbf{Sim}(3)$ transformation can be cast into $\mathbf{SE}(3)$ by scaling the translation part by the depth prior. Hence, the cost function is simplified to:

$$\mathbf{E}_{loop} = \sum_{q_i \in Q} ||\Pi(\mathbf{S_{cr}}\Pi^{-1}(p_i, d_{p_i})) - \Pi^{-1}(q_i)||_2 \tag{8}$$

Where p_i is the reconstructed feature point in the loop candidate, d_* is the inverse depth of active map points in the corresponding frame, Q are the matched features in the current keyframe, Π and Π^{-1} are the projection and backward projection functions, and S_{cr} is the target $Sim(3)$ constraint to be estimated from the cost function above. However, storing each keyframe and their tracked map points in the global mapping database is a heavy burden on the system in terms of time and memory. To avoid this, we follow the keyframe selection trick in S-PTAM [20]: with a current pose estimated in the sliding window, a frame is selected to be a keyframe if the tracking feature points consist of less than 90% of those from the previous keyframe in the pose graph.

We apply marginalization for three tasks: (a) to solve camera poses, (b) eliminate outlier map points, and (c) remove redundant keyframes. Note that the Jacobians of geometrical parameters $((T_i, T_j, d, c))$ with respect to two frames' poses are linearly related. The big Hessian matrix can be decomposed into two parts: one pose Hessian which is very small, one point Hessian vector which is a large vector containing all the inverse depth Jacobians in each entry. The pose Hessian is aggregated through each point Hessian as follows:

$$\hat{H_\xi} = \sum_{i \in \mathcal{P}_g} \{H_\xi - H_{\xi,p_i} H_{p_i}^{-1} H_{p_i,\xi}\} \tag{9}$$

$$\hat{b_\xi} = \sum_{i \in \mathcal{P}_g} \{b_\xi - H_{\xi,p_i} H_{p_i}^{-1} b_{p_i}\} \tag{10}$$

After the linear aggregation, the pose update δ can be solved from:

$$\delta = \hat{H_\xi}^{-1} \hat{b_\xi} \tag{11}$$

and then used to update the current state: $\xi' = \delta + \xi$. Note that the plus operation here is defined in Lie manifold. In practice, $H_{\xi,p_i} = J_\xi J_{p_i}$ is stored as a vector of Jacobians where each Jacobian entry corresponds to a point, and

Sequence		Absolute Pose Error (APE)					Relative Pose Error (RPE)				
	Method	Max	Mean	Min	Rmse	Std	Max	Mean	Min	Rmse	Std
seq 00	DSO [4]	239.06	101.12	4.05	118.28	61.34	5.08	0.61	0.00	0.72	0.39
	SDSO [11]	26.47	7.34	0.57	9.34	5.78	1.75	0.10	0.00	0.19	0.16
	LDSO [1]	28.53	11.95	2.49	13.45	6.17	**0.87**	0.09	0.00	**0.11**	**0.06**
	CNN-SVO [10]	64.41	17.66	4.07	20.11	9.63	2.26	0.09	0.00	0.15	0.12
	Ours	**14.66**	**3.57**	**0.10**	**4.55**	**2.83**	4.26	**0.08**	**0.00**	0.23	0.22
seq 02	DSO	308.07	100.73	6.76	125.05	74.10	7.12	0.50	0.00	0.61	0.35
	SDSO	**11.62**	**5.22**	**0.32**	**5.81**	**2.56**	0.88	0.07	0.00	0.11	0.09
	LDSO	121.45	27.96	3.47	34.92	20.91	3.64	0.09	0.00	0.15	0.12
	CNN-SVO	84.03	45.80	21.85	48.25	15.17	7.39	0.58	0.01	0.74	0.47
	Ours	23.11	10.05	4.01	10.98	4.41	**0.50**	**0.05**	**0.00**	**0.06**	**0.04**
seq 05	DSO	174.91	55.78	10.95	65.11	33.58	3.25	0.46	0.00	0.57	0.34
	SDSO	26.61	7.46	**0.17**	8.97	4.98	2.87	0.10	0.00	0.21	0.19
	LDSO	13.53	3.80	1.04	4.25	1.90	0.58	**0.05**	**0.00**	**0.07**	**0.04**
	CNN-SVO	33.59	10.77	0.69	12.22	5.77	2.39	0.09	0.00	0.20	0.18
	Ours	**7.06**	**2.81**	1.00	**3.09**	**1.27**	**0.48**	0.06	0.00	0.08	0.05
seq 06	DSO	146.81	76.08	3.41	83.63	34.72	5.81	0.48	0.01	0.64	0.43
	SDSO	9.50	**4.29**	**0.60**	4.89	2.35	1.64	**0.08**	**0.00**	**0.14**	0.12
	LDSO	24.99	11.20	0.71	13.27	7.11	**0.58**	0.12	0.00	0.15	**0.09**
	CNN-SVO	220.98	71.30	22.59	81.73	39.95	6.18	0.89	0.09	1.15	0.73
	Ours	**7.58**	4.57	0.92	**4.82**	**1.52**	1.05	0.11	0.01	0.15	0.10
seq 07	DSO	64.96	19.33	0.36	23.43	13.23	0.98	0.25	0.01	0.32	0.19
	SDSO	4.16	2.18	0.39	2.35	0.89	0.95	0.08	0.00	0.13	0.10
	LDSO	47.98	17.58	4.52	20.10	9.74	2.43	0.20	0.01	0.27	0.19
	CNN-SVO	11.35	3.31	0.37	4.05	2.33	1.04	0.07	0.00	0.10	0.08
	Ours	**0.66**	**0.42**	**0.13**	**0.44**	**0.13**	**0.53**	**0.04**	**0.00**	**0.06**	**0.04**
seq 09	DSO	164.56	57.58	3.11	68.21	36.57	11.24	0.32	0.00	0.48	0.36
	SDSO	**10.85**	**3.93**	**0.72**	**4.50**	**2.18**	1.27	0.09	0.00	0.15	0.12
	LDSO	165.46	64.18	4.45	75.80	40.34	1.25	0.31	0.01	0.37	0.20
	CNN-SVO	27.59	13.23	3.88	14.69	6.38	4.49	0.28	0.01	0.52	0.44
	Ours	13.85	4.26	1.49	4.90	2.42	**0.36**	**0.05**	**0.00**	**0.06**	**0.04**

Table I: Quantitative results of trajectory APE and RPE evaluated on KITTI dataset. Note that DSO and LDSO are scaled and $\mathbf{Sim}(3)$ aligned with ground truth trajectories. Evaluation results on seq 01, 03, 04, 10 are not listed since the trajectories are too short and lack rotations. Above results are averaged over 5 experiments of each method.

is aggregated to solve $\hat{H}_\xi$, where J_ξ is the Jacobian of camera pose. According to Eq. (6), the Jacobian of points (parameterized as inverse depth) is defined as:

$$J_{p_i} = f_x(t_1 - ut_3)I_x\frac{d'_T(p_i)}{d'_R(p_i)} + f_y(t_2 - vt_3)I_y\frac{d'_T(p_i)}{d'_R(p_i)} \quad (12)$$

Where $fx(t_1 - ut_3)$, $f_y(t_2 - vt_3)$ are proportional to d_x, d_y defined in Eq. (6), u, v are pixel coordinates in the target frame and t is the relative translation between the reference frame and target frame, and d'_R and d'_T are inverse depth priors from the reference frame and target frame. Note that these depth priors are used to initialize the Jacobian of points following the First Estimation Jacobian trick [21], as we assume the depth Jacobian is smooth within the local tangent space of current parameter state.

The inverse depth in turn will be updated for each point in the host frame based on the optimized camera pose ξ':

$$\delta_{p_i} = H_{p_i}^{-1}(b_{p_i} - H_{\xi,p_i}^T\xi') \quad (13)$$

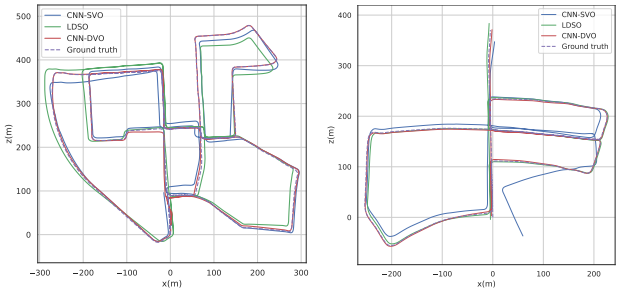
For each point p_i, the inverse depth $d_{new}[p_i]$ is thus updated by $d_{new}[p_i] = d'_R[p_i] + \delta_{p_i}$. Here, $d'_R[p_i]$ is the inverse depth of point p_i in the reference frame, and $d'_T = d'_1$ is defined in Eq. (7). The removal of marginal points and frames are expressed in Eq. (17) to (19) in [4].

IV. Experiment

In this section, we demonstrate our experiment results evaluated on the odometry dataset of the KITTI Vision Benchmark [22]. Our depth estimation and dense local bundle adjustment are performed on a Nvidia RTX 2080 max-Q GPU. Feature extraction and the loop closure (implemented with G^2O [23]) are all done by CPU (Intel Core i7-8750H) running at 2.6 GHz with 16 GB RAM. The input image resolution is 1241 $\times$ 376, all images are pre-rectified. Depth inference takes additional time on CPU (125 $\pm$ 50 ms), but can run faster on GPU (20 $\pm$ 10 ms), which makes our GPU-based implementation run in real-time on a laptop computer.

Ablation study: Using the KITTI Odometry Benchmark, we show the effect of each design decision of our system. Table II contains the results of our approach without incorporating depth priors into the pipeline, without integrating loop closure for pose optimization, and while using a predicted pose prior for initialization.

Starting with our principle contribution - our experiments show that the inclusion of depth priors helps to progressively reduces scale-drift over long sequences by restricting the reprojection of tracking points in the local bundle adjustment near the global minimum. The remaining accumulated estimation errors can then be further corrected with pose



(a) Seq 00 (b) Seq 05

Figure 6: Sample trajectory comparison in KITTI dataset with our CNN-DVO method. Trajectories for each method are averaged over 5 runs.

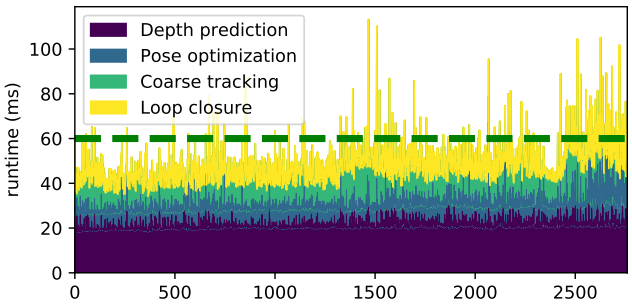


Figure 7: Processing time needed to map sequence 05 from the KITTI Vision Benchmark. See Fig. (1) for the final mapping result.

optimization through loop closure. We find that initializing our system with a predicted pose prior provides a relatively small accuracy boost on select sequences, but on average, does not improve the performance of the system.

Table II: Ablation study of major components - ATE (m) on all KITTI training sequences. Experiments are averaged over 7 runs of each sequence.

Seq	Ours	Ours $w\backslash o$ DP*	Ours $w\backslash o$ LC*	Ours $w\backslash$ PP*	CNN-SVO
00	**4.55**	118.28	13.45	9.34	130.03
01	9.79	9.71	10.48	**6.79**	202.36
02	10.98	125.05	34.92	**5.81**	48.24
03	**1.05**	2.33	2.99	1.25	3.26
04	0.52	0.98	1.12	**0.38**	2.10
05	**2.23**	65.11	4.25	8.97	20.39
06	**3.04**	83.63	13.27	4.89	12.50
07	**0.44**	23.42	2.10	2.35	4.05
08	**4.51**	116.85	6.48	6.49	10.65
09	4.90	68.21	5.80	**4.50**	14.69
10	1.49	13.68	7.68	**1.20**	8.74
mean	**3.96**	57.02	9.14	**5.00**	41.55

$w\backslash o$ DP*: without depth prior ($\mathbf{Sim}(3)$ scale aligned).
$w\backslash o$ LC*: without loop closure for pose optimization.
$w\backslash$ PP*: with pose prior for initialization.

Comparison with state-of-the-art: As presented in Table I, we evaluate our CNN-DVO method along with state-of-the-art direct visual odometry methods like DSO, LDSO, CNN-SVO and stereo DSO for comparison on all odometry sequences from the KITTI dataset (sample trajectories in Fig. 6). Since DSO and LDSO are initialized in arbitrary scale, we perform $\mathbf{Sim}(3)$ alignment with ground truth for evaluation consistency. In some experiments, we demonstrate better performance than Stereo DSO (sequences containing a lot of strong rotation motion). Since our point selection strategy more evenly samples points across the global image, the tracking thread is less affected by the frequent removal of outliers. Also, the fast depth initialization of new tracking points constrains the scale of the pose Hessians marginalization and regularizes the pose delta update close to the parameter tangent space.

Runtime evaluation: Since we estimate the depth in low-resolution, it takes roughly 0.05s for each frame, and a significant amount of time is saved by running the system with fewer tracking points (increased sparsity). The overall runtime is slightly slower than DSO, but the initialization process is faster.

Fig. (7) shows the processing time required for sequence 05 from the KITTI Vision Benchmark. Note that the inference for depth priors are done on our GPU; the average inference time is 23 ms.

Our tracking pipeline operates at 15 FPS, while LDSO runs at 13 FPS, DSO at 28 FPS, and CNN-SVO at 14 FPS. For the front-end pose update with loop closure, we operate at 3 FPS, with LDSO slightly quicker at 4 FPS.

V. Conclusion

In this paper, we present a novel deep learning powered direct visual odometry system. To our knowledge, it is the only direct formulation that fully integrates depth prediction with every major system component (initialization, tracking, marginalization) to jointly optimize for all model parameters including inverse depth, camera pose and affine model in real-time (with GPU-based implementation). Extra constraints given by the depth prior provides strong benefits to our system, as it **a)** fixes the arbitrary initialization scale into constant scale, **b)** it greatly restrains the scale-drift during tracking, and **c)** it recovers the 3D correspondence of the measurements in loop closure while helping to maintain consistent scale. We also propose an improved point selection strategy to sample points near both low and high gradient image regions to avoid clustering, rendering our tracking thread more adaptable to large baseline stereo problems (e.g. in-place rotation). We have demonstrated that our approach achieves state-of-the-art results on the KITTI Odometry Benchmark.

In the future, we aim to extend our pipeline to include self-supervised fine-tuning of the deep CNN depth predictor. This can be achieved by employing a sparse-to-dense network to reconstruct a dense depth map from the newly optimized

sparse depths after local bundle adjustment. To this end, we hope to bridge the current optimization framework with depth refinement to maintain precise pose and depth estimations in new and challenging domains.

ACKNOWLEDGMENT

REFERENCES

[1] X. Gao, R. Wang, N. Demmel, and D. Cremers, "Ldso: Direct sparse odometry with loop closure," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2198–2204.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[3] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[4] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[5] X. Wu and C. Pradalier, "Robust semi-direct monocular visual odometry using edge and illumination-robust cost," *arXiv preprint arXiv:1909.11362*, 2019.

[6] K. S. Shankar and N. Michael, "Robust direct visual odometry using mutual information," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2016, pp. 9–14.

[7] P. Bergmann, R. Wang, and D. Cremers, "Online photometric calibration of auto exposure video for realtime visual odometry and slam," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 627–634, 2017.

[8] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2878–2885, 2018.

[9] C. Zhao, Y. Tang, and Q. Sun, "Deep direct visual odometry," *arXiv preprint arXiv:1912.05101*, 2019.

[10] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, "Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5218–5223.

[11] R. Wang, M. Schworer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3903–3911.

[12] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2510–2517.

[13] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[14] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 817–833.

[15] C. Wang, J. Miguel Buenaposada, R. Zhu, and S. Lucey, "Learning depth from monocular videos using direct methods," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2022–2030.

[16] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3828–3838.

[17] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1449–1456.

[18] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[19] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular slam," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[20] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. J. Berlles, "S-ptam: Stereo parallel tracking and mapping," *Robotics and Autonomous Systems*, vol. 93, pp. 27–42, 2017.

[21] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A first-estimates jacobian ekf for improving slam consistency," in *Experimental Robotics*. Springer, 2009, pp. 373–382.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[23] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.