Adaptive Continuous Visual Odometry from RGB-D Images

Tzu-Yuan Lin, William Clark, Ryan M. Eustice, Jessy W. Grizzle
Anthony Bloch, and Maani Ghaffari

***Abstract*— In this paper, we extend the recently developed continuous visual odometry framework for RGB-D cameras to an adaptive framework via online hyperparameter learning. We focus on the case of isotropic kernels with a scalar as the length-scale. In practice and as expected, the length-scale has remarkable impacts on the performance of the original framework. Previously it was handled using a fixed set of conditions within the solver to reduce the length-scale as the algorithm reaches a local minimum. We automate this process by a greedy gradient descent step at each iteration to find the next-best length-scale. Furthermore, to handle failure cases in the gradient descent step where the gradient is not well-behaved, such as the absence of structure or texture in the scene, we use a search interval for the length-scale and guide it gradually toward the smaller values. This latter strategy reverts the adaptive framework to the original setup. The experimental evaluations using publicly available RGB-D benchmarks show the proposed adaptive continuous visual odometry outperforms the original framework and the current state-of-the-art. We also make the software for the developed algorithm publicly available.**

I. INTRODUCTION

Visual odometry using depth cameras is the problem of finding a rigid-body transformation between two colored point clouds. This problem arises frequently in robotics and computer vision and is an integral part of many autonomous systems [1]–[5]. Direct visual odometry methods minimize the photometric error using image intensity values measured by the camera [6]–[8]. The explicit representation between color information and 2D/3D geometry (image or Euclidean space coordinates) is not directly available; hence, the current direct methods use numerical differentiation for computing the gradient and are limited to fixed image size and resolution, given the camera model and measurements for re-projection of the 3D points. In this setup, a coarse-to-fine image pyramid [9, Section 3.5] is constructed to solve the same problem several times with initialization provided by solving the previous coarser step.

Alternatively, the Continuous Visual Odometry (CVO) is a continuous and direct formulation and solution for the RGB-D visual odometry problem [10]. Due to the continuous representation of CVO, it neither requires the association between two measurement sets nor the same number of measurements within each set. In addition, there is no need for constructing a coarse-to-fine image pyramid in the continuous sensor registration framework developed in [10]. In this framework, the joint appearance and geometric embedding is modeled by representing the processes (RGB-D images) in a Reproducing Kernel Hilbert Space (RKHS) [11], [12].

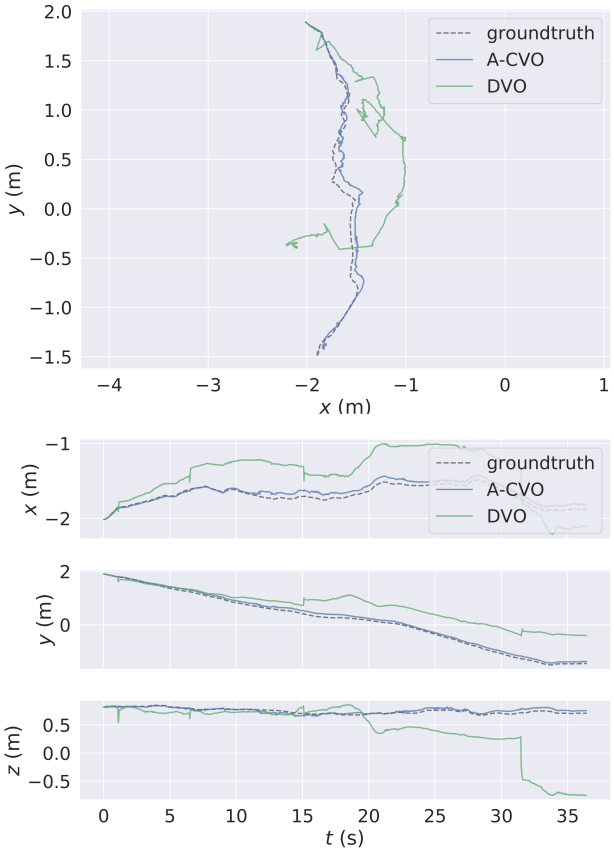

Fig. 1: The top figure shows trajectories of the proposed adaptive continuous visual odometry (A-CVO), dense visual odometry (DVO) [13], and ground truth for `fr3/structure_notexture_near` sequence of the RGB-D benchmark in [14]. The bottom figures show the x, y, and z trajectories vs. time. In the absence of texture, A-CVO performs well and almost follows the ground truth trajectory.

Robust visual tracking has become a core aspect of state-of-the-art robotic perception and navigation in both structured and unstructured indoor and outdoor [13], [15]–[20]. Hence, this work contributes to the foundations of robotic perception and autonomous systems via a continuous sensor registration framework enhanced by an adaptive hyperparameter learning strategy. In particular, this work has the following contributions:

1) We extend the continuous visual odometry framework

*This work was partially supported by the Toyota Research Institute (TRI), partly under award number N021515. Funding for J. Grizzle was in part provided by TRI and in part by NSF Award No. 1808051. Funding for W. Clark and A. Bloch was provided by NSF DMS 1613819 and AFSOR FA9550-18-10028.

T.Y. Lin, W. Clark, R. Eustice, J. Grizzle, A. Bloch, and M. Ghaffari are with the University of Michigan, Ann Arbor, MI 48109, USA. {tzuyuan, wiclark, eustice, grizzle, abloch, maanigj}@umich.edu.

for RGB-D cameras to an adaptive framework via online hyperparameter learning. We also perform a sensitivity analysis of the problem and propose a systematic way to choose the sparsification threshold discussed in [10].
2) We generalize the appearance (color) information inner product in [10] to a kernelized form that improves the performance. With this improvement alone, the experimental evaluations show that the original continuous visual odometry is intrinsically robust and its performance is similar to that of the state-of-the-art robust dense (and direct) RGB-D visual odometry method [13].
3) We evaluate the proposed algorithm using the publicly available RGB-D benchmark in [14] and make the software for the developed algorithm publicly available[1].

The remainder of this paper is organized as follows. The problem setup is given in §II. The adaptive continuous visual odometry framework is discussed in §III. The sensitivity analysis of the problem is provided in §IV. The experimental results are presented in §V. Finally, §VI concludes the paper and discusses future research directions.

II. Problem Setup

Consider two (finite) collections of points, $X = \{x_i\}$, $Z = \{z_j\} \subset \mathbb{R}^3$. We want to determine which element $h \in \mathrm{SE}(3)$, where $R \in \mathrm{SO}(3)$ and $T \in \mathbb{R}^3$, aligns the two point clouds X and $hZ = \{hz_j\}$ the "best." To assist with this, we will assume that each point contains information described by a point in an inner product space, $(\mathcal{I}, \langle\cdot,\cdot\rangle_{\mathcal{I}})$. To this end, we will introduce two labeling functions, $\ell_X : X \to \mathcal{I}$ and $\ell_Z : Z \to \mathcal{I}$.

In order to measure their alignment, we will be turning the clouds, X and Z, into functions $f_X, f_Z : \mathbb{R}^3 \to \mathcal{I}$ that live in some reproducing kernel Hilbert space, $(\mathcal{H}, \langle\cdot,\cdot\rangle_{\mathcal{H}})$. The action, $\mathrm{SE}(3) \curvearrowright \mathbb{R}^3$ induces an action $\mathrm{SE}(3) \curvearrowright C^\infty(\mathbb{R}^3)$ by $h.f(x) := f(h^{-1}x)$. Inspired by this observation, we will set $h.f_Z := f_{h^{-1}Z}$.

Problem 1. *The problem of aligning the point clouds can now be rephrased as maximizing the scalar products of f_X and $h.f_Z$, i.e., we want to solve*

$$\underset{h\in\mathrm{SE}(3)}{\arg\max}\ F(h), \quad F(h) := \langle f_X, h.f_Z\rangle_{\mathcal{H}}. \tag{1}$$

A. Constructing the functions

We follow the same steps in [10] with an additional step in which we use the kernel trick to kernelize the information inner product. For the kernel of our RKHS, $\mathcal{H}$, we first choose the squared exponential kernel $k : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$:

$$k(x,z) = \sigma^2 \exp\left(\frac{-\|x-z\|_3^2}{2\ell^2}\right), \tag{2}$$

for some fixed real parameters (hyperparameters) σ and ℓ, and $\|\cdot\|_3$ is the standard Euclidean norm on $\mathbb{R}^3$. This allows us to turn the point clouds to functions via

$$f_X(\cdot) := \sum_{x_i\in X} \ell_X(x_i)k(\cdot,x_i),$$
$$f_Z(\cdot) := \sum_{z_j\in Z} \ell_Z(z_j)k(\cdot,z_j). \tag{3}$$

We can now define the inner product of f_X and f_Z by

$$\langle f_X, f_Z\rangle_{\mathcal{H}} := \sum_{x_i\in X, z_j\in Z} \langle \ell_X(x_i), \ell_Z(z_j)\rangle_{\mathcal{I}} \cdot k(x_i,z_j). \tag{4}$$

We use the well-known kernel trick in machine learning [21]–[23] to substitute the inner products in (4) with the appearance (color) kernel. The kernel trick can be applied to carry out computations implicitly in the high dimensional space, which leads to computational savings when the dimensionality of the feature space is large compared to the number of data points [22]. After applying the kernel trick to (4), we get

$$\begin{aligned}\langle f_X, f_Z\rangle_{\mathcal{H}} &= \sum_{x_i\in X, z_j\in Z} k_c(\ell_X(x_i), \ell_Z(z_j)) \cdot k(x_i,z_j) \\ &:= \sum_{x_i\in X, z_j\in Z} c_{ij} \cdot k(x_i,z_j),\end{aligned} \tag{5}$$

where we choose k_c to be also the squared exponential kernel with fixed real hyperparameters σ_c and ℓ_c that are set independently.

III. Adaptive Continuous Visual Odometry via Online Hyperparameter Learning

The length-scale of the kernel, ℓ, is an important hyperparameter that affects the performance and convergence of the algorithm significantly. In the original framework in [10], ℓ was set using a fixed set of conditions within the solver to reduce the length-scale as the algorithm reached a local minimum. Intuitively, large values of ℓ encourage higher correlations between points that are far apart from each other; and small values of ℓ encourage the algorithm to focus on only points that are very close to each other with respect to the distance metric of the kernel (here we use the Euclidean distance). This latter case results in faster convergence and it can be thought of as refinement steps where the target and source clouds are already almost aligned.

Now the question to answer is how can we tune ℓ automatically and online at each iteration so that the overall registration performance is maximized? In this section, we provide a solution that is based on a greedy gradient descent search. As we will see, this approach is highly appealing due to its simplicity and the gain in performance. We first revisit Problem 1. The maximization of the inner product is a reduced form of the original cost $J(h) := \|f_X - h.f_Z\|_{\mathcal{H}}^2$ and the fact that $\mathrm{SE}(3) \curvearrowright C^\infty(\mathbb{R}^3)$ is an isometry. That is

$$\begin{aligned}\|f_X - h.f_Z\|_{\mathcal{H}}^2 &= \|f_X\|_{\mathcal{H}}^2 + \|f_Z\|_{\mathcal{H}}^2 - 2\langle f_X, h.f_Z\rangle_{\mathcal{H}} \\ &= \sum_{x_i,x_j\in X} \alpha_{ij}\cdot k(x_i,x_j) + \sum_{z_i,z_j\in Z} \beta_{ij}\cdot k(z_i,z_j) \\ &\quad - 2\sum_{x_i\in X, z_j\in Z} c_{ij}\cdot k(x_i,h^{-1}z_j),\end{aligned} \tag{6}$$

[1]Software is available for download at https://github.com/MaaniGhaffari/cvo-rgbd

where coefficients α_{ij} and β_{ij} are defined for each function's inner product with itself similar to (5).

Computing the gradient of (6) with respect to ℓ is straightforward and is given by

$$\begin{aligned}\frac{\partial J(\ell)}{\partial \ell} &= \frac{\partial J}{\partial k} \cdot \frac{\partial k}{\partial \ell} \\ &= \frac{1}{\ell^3}\Bigg[\sum_{x_i, x_j \in X} \alpha_{ij} \cdot p_{ij} \cdot k(x_i, x_j) \\ &\quad + \sum_{z_i, z_j \in Z} \beta_{ij} \cdot q_{ij} \cdot k(z_i, z_j) \\ &\quad - 2 \sum_{x_i \in X, z_j \in Z} c_{ij} \cdot r_{ij} \cdot k(x_i, h^{-1} z_j) \Bigg], \end{aligned} \tag{7}$$

where we defined $p_{ij} := \|x_i - x_j\|_3^2$, $q_{ij} := \|z_i - z_j\|_3^2$, and $r_{ij} := \|x_i - z_j\|_3^2$. Then using the following update (integration) rule we find the length-scale for the next iteration,

$$\ell_{k+1} = \ell_k - \gamma_\ell \cdot \frac{\partial J(\ell_k)}{\partial \ell_k}, \tag{8}$$

where γ_ℓ is the step size (learning rate).

This strategy alone can lead to failure or extremely poor performance based on our observations. The reason is that CVO uses semi-dense data and in the absence of structure or texture in the environment the gradient can be weak or not well-behaved. To address this problem, we can simply define a search interval for the length-scale as $\ell \in [\ell_{min}, \ell_{max}]$. This additional step not only keeps ℓ in a feasible region but also allows the algorithm to detect when tracking is difficult and issue a warning message. To improve the convergence, when $\ell \geq \ell_{max}$, we reduce both ℓ and ℓ_{max} by a reduction factor, $0 < \lambda_\ell < 1$, and continue as before.

IV. Sensitivity Analysis

Understanding how k in equation (2) depends on ℓ is a surprisingly delicate problem which, surprisingly, offers a systematic way to choose the kernel sparsification threshold (see Table I). Consider the following normalization of k:

$$g(s) = \exp\left(-\frac{1}{2s^2}\right), \quad s = \frac{\ell}{\|x - y\|}, \tag{9}$$

so $\sigma^2 g(s) = k(x, y)$. Suppose we want to find an approximation of g as s gets small; this is equivalent to understanding $k(x, y)$ as $\|x - y\| >> \ell$. Performing a Taylor expansion of $g(s)$ about $s = 0$ results in the zero function. A simple enough calculation shows that

$$\frac{d^k}{ds^k} g(s) = p_k(s) g(s), \tag{10}$$

where p_k is some rational function. Due to the fact that g is exponential, we have that

$$\lim_{s \to 0^+} Q(s) g(s) = 0, \tag{11}$$

for *any* rational function Q. This shows that the Taylor series of g about $s = 0$ is trivially zero. (The underlying reason for the Taylor series being zero while the function is not is because g is not *analytic* at $s = 0$. In fact, if we view g as a complex function there is an *essential singularity* at $s = 0$, see §5.6 in [24].)

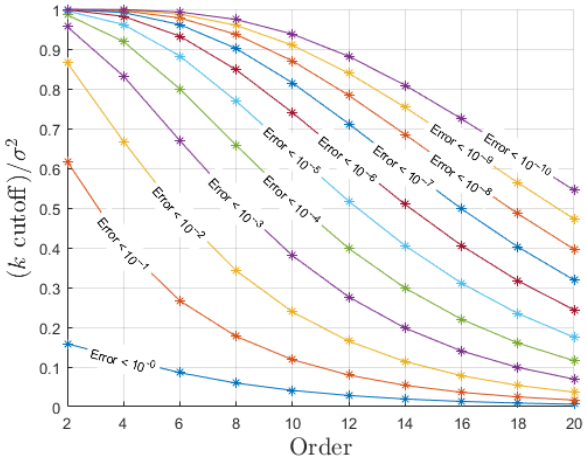

Fig. 2: This shows the cutoff values for k based on the order of the approximation and the required error tolerances. For example, if we use a 6^{th}-order expansion and we require an error of less than 10^{-3}, then all points where $k(x, y) < 0.6694\sigma^2$ need to be ignored.

Rather than expanding about $s = 0$ (where points are far apart), we can expand about $s = \infty$ (where points are close together). This results in the following expansion:

$$g(s) \sim 1 - \frac{1}{2}s^{-2} + \frac{1}{8}s^{-4} - \frac{1}{48}s^{-6} + \dots. \tag{12}$$

While this approximation is accurate when s is large, as s approaches zero this approximation falls apart. The exact function, g, approaches zero as $s \to 0$ but the approximation has a pole at zero regardless of order. This motivates a minimum cutoff for s such that (12) has a well controlled error. By applying this cutoff to the original function k, we can obtain a kernel sparsification threshold that guarantees error bounds in the approximation (12). A plot of these values is shown in Fig. 2.

V. Experimental Results

We now present experimental evaluations of the proposed method Adaptive CVO (A-CVO). We compare A-CVO with the original CVO [10] and the state-of-the-art direct (and dense) RGB-D visual odometry (DVO) [16]. Since the original DVO source code requires outdated ROS dependency [25], we reproduced DVO results using the version provided by Matthieu Pizenberg [26], which only removes the dependency for ROS while maintains the DVO core source code unchanged. We also include the DVO results of Kerl et al. [16] for reference. We refer to the reproduced DVO results as *DVO* and the results directly taken from [16] as *Kerl et al. [16]*.

A. Experimental Setup

To improve the computational efficiency, we adopted a similar approach to Direct Sparse Visual Odometry (DSO) by Engel et al. [17] to create a semi-dense point cloud

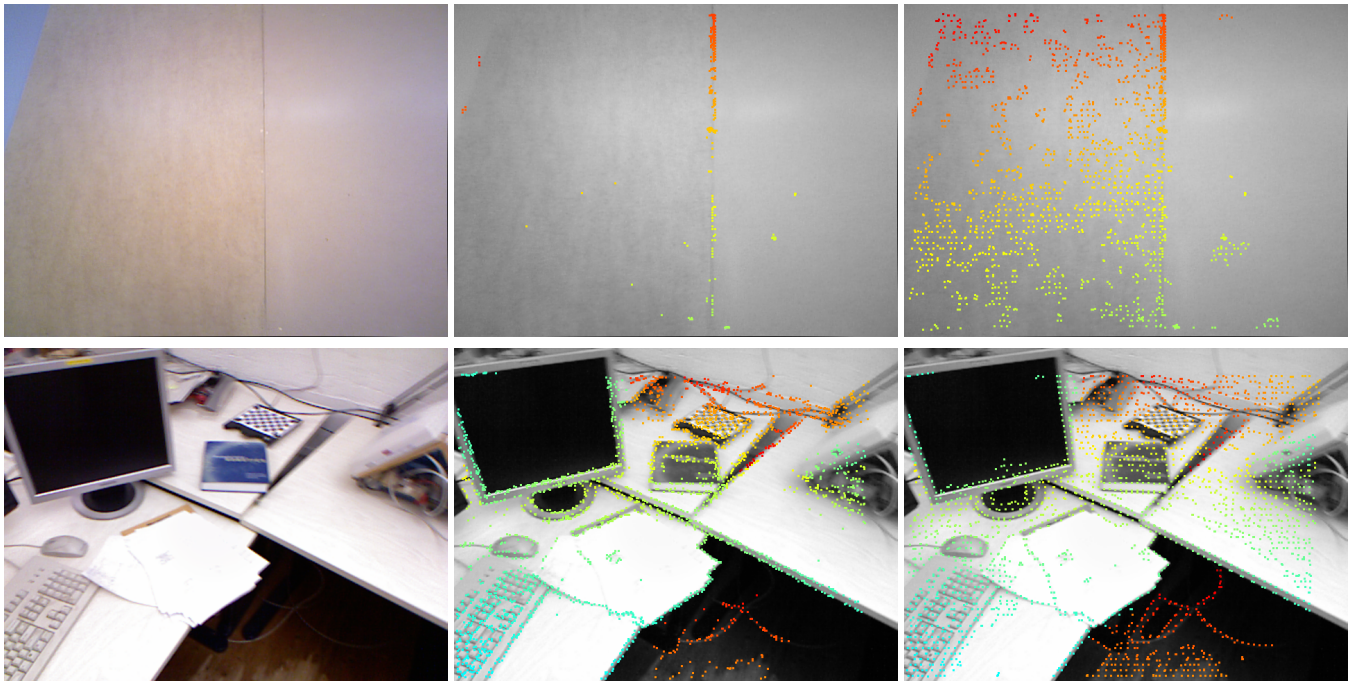

Fig. 3: The visualization of the point selection mechanism adopted from DSO [18]. The top row shows image `1341840842.006342.png` in `fr3/nostructure_notexture_far` sequence. The bottom row shows image `1305031453.359684.png` in `fr1/desk` sequence. The leftmost images are the original image recorded by a Microsoft Kinect. The images in the middle show the points selected by the DSO point selection algorithm. The top right image shows the points selected when the number of points selected by DSO is insufficient. Extra points are selected by downsampling the highlighted edge computed using the Canny edge detector [27]. The bottom right image shows the points selected solely by downsampling of the Canny edge detector output. In this case, since the DSO point selector already picked up enough points, the points from the Canny detector are not used. The points highlighted in the images are made 9 times bigger (i.e. 9 pixels are drawn) to make the visualization more clear.

TABLE I: Parameters used for evaluation using TUM RGB-D Benchmark, similar values are chosen for all experiments. The kernel characteristic length-scale is chosen to be adaptive as the algorithm converges; intuitively, we prefer a large neighborhood of correlation for each point, but as the algorithm reaches the convergence reducing the local correlation neighborhood allows for faster convergence and better refinement.

Parameters	Symbol	Value
Transformation convergence threshold	ϵ_1	1e−5
Gradient norm convergence threshold	ϵ_2	5e−5
Minimum step length		0.2
Kernel sparsification threshold		8.315e−3
Spatial kernel initial length-scale	ℓ_{init}	0.1
Spatial kernel signal variance	σ	0.1
Spatial kernel minimum length-scale (A-CVO)	ℓ_{min}	0.039
Spatial kernel maximum length-scale (A-CVO)	ℓ_{max}	0.15
Color kernel length-scale	ℓ_c	0.1
Color kernel signal variance	σ_c	1
ℓ integration step size (A-CVO)	γ_ℓ	0.3
ℓ reduction factor (A-CVO)	λ_ℓ	0.7

(around 3000 points) for each scan. To prevent insufficient points being selected in environments that lack rich visual information, we also used a Canny edge detector [28] from OpenCV [27]. When the points selected by the DSO point selector are less than one-third of the desired number of points, more points will be selected by downsampling the pixels highlighted by the Canny detector. While generating the point cloud, RGB values are first transformed into HSV colormap and normalized. The normalized HSV values are then combined with the normalized intensity gradients and utilized as the labels of the selected points in the color space. For all experiments, we used the same set of parameters, which are listed in Table I.

All experiments are performed on a Dell XPS15 9750 laptop with Intel i7-8750H CPU (6 cores with 2.20 GHz each) and 32GB RAM. The source code is implemented in C++ and compiled with the Intel Compiler. The kernel computations are parallelized using the Intel Threading Building Blocks (TBB) [29]. Using compiler auto-vectorization and the parallelization, the average time for frame-to-frame registration is 0.5 sec. The frame-to-frame registration time for the original CVO is 0.2 sec (5 Hz).

B. TUM RGB-D Benchmark

We performed experiments on two parts of RGB-D SLAM dataset and benchmark by the Technical University of Munich [14]. This dataset was collected indoors with a Microsoft Kinect using a motion capture system as a proxy for ground truth trajectory. For all tracking experiments, the entire images were used sequentially without any skipping, i.e., at full frame rate. We evaluated A-CVO, CVO, and DVO on the training and validation sets for all the fr1 sequences and the structure versus texture sequences. RGB-D benchmark tools [14] were then used to evaluate the Relative Pose Error (RPE) of all 3 methods, and evo [30] was utilized to visualize the trajectory.

Table II shows the Root-Mean-Squared Error (RMSE) of the RPE for `fr1` sequences. The Trans. columns show the RMSE of the translational drift in m/sec and the Rot. columns show the RMSE of the rotational drift in

TABLE II: The RMSE of Relative Pose Error (RPE) for `fr1` sequences. The trans. columns show the RMSE of the translational drift in $\mathrm{m/sec}$ and the rot. columns show the RMSE of the rotational error in $\mathrm{deg/sec}$. The Average* shows the average result excluding `fr1/floor` sequence since Kerl et al. [16] reported a failure on that sequence. The rotational errors for Kerl et al. were left empty for they were not reported in the original paper [16]. There's no corresponding validation datasets for `fr1/teddy` and `fr1/floor`. The results show that A-CVO out-performs the other two methods on both training and validation sets of `fr1`.

	Training								Validation							
	CVO [10]		A-CVO		Kerl et al. [16]		DVO		CVO [10]		A-CVO		Kerl et al. [16]		DVO	
Sequence	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.
fr1/desk	0.0486	2.4860	0.0375	**2.1456**	**0.0360**	n/a	0.0387	2.3589	0.0401	2.0148	0.0431	**1.8831**	**0.0350**	n/a	0.0371	2.0645
fr1/desk2	0.0535	3.0383	**0.0489**	**2.5857**	0.0490	n/a	0.0583	3.6529	0.0225	1.7691	0.0224	**1.6584**	**0.0200**	n/a	0.0208	1.7416
fr1/room	0.0560	2.4566	0.0529	**2.2750**	0.0580	n/a	**0.0518**	2.8686	**0.0446**	**3.9183**	0.0465	3.9669	0.0760	n/a	0.2699	7.4144
fr1/360	**0.0991**	3.0025	0.0993	**3.0125**	0.1190	n/a	0.1602	4.4407	0.1420	3.0746	0.0995	**2.2177**	**0.0970**	n/a	0.2811	7.0876
fr1/teddy	0.0671	4.8089	**0.0553**	**2.2342**	0.0600	n/a	0.0948	2.5495	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
fr1/floor	0.0825	2.3745	0.0899	2.2904	fail	n/a	**0.0635**	**2.2805**	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
fr1/xyz	0.0240	1.1703	**0.0236**	**1.1682**	0.0260	n/a	0.0327	1.8751	0.0154	1.3872	**0.0150**	**1.2561**	0.0470	n/a	0.0453	3.0061
fr1/rpy	0.0457	3.3073	0.0425	3.0497	0.0400	n/a	**0.0336**	**2.6701**	0.1138	3.6423	**0.0799**	**2.4335**	0.1030	n/a	0.3607	7.9991
fr1/plant	0.0316	1.9973	0.0347	1.8580	0.0360	n/a	**0.0272**	**1.5523**	0.0630	4.9185	**0.0591**	4.1925	0.0630	n/a	0.0660	**2.5865**
Average*	0.0532	2.7834	**0.0493**	**2.2911**	0.0530	n/a	0.0622	2.7460	-	-	-	-	-	n/a	-	-
Average all	0.0561	2.7380	**0.0534**	**2.2910**	n/a	n/a	0.0623	2.6943	0.0631	2.9607	**0.0522**	**2.5155**	0.0630	n/a	0.1544	4.5571

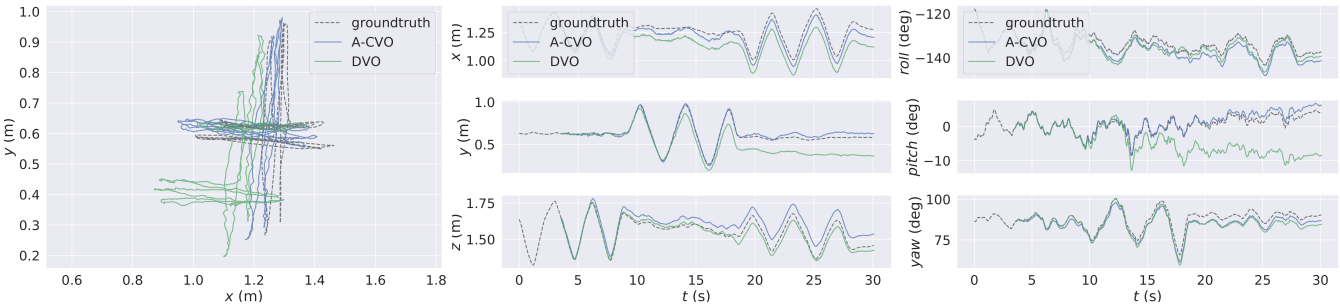

Fig. 4: The trajectory comparison of A-CVO, DVO, and groundtruth for `fr1/xyz` sequence. The left figure shows the trajectory in the xy plane. The middle figure shows the x, y, z trajectory with respect to time. (Note that it's not the error plot.) The figure in the right shows the angles of roll, pitch, and yaw with respect to time. This sequence contains a repetitive motion and can show the repeatability of a method. As shown in the figures, A-CVO can follow the groundtruth trajectory while DVO drifts in the y direction.

$\mathrm{deg/sec}$. The Average* shows the average result by excluding `fr1/floor` sequence since Kerl et. al reported failure on that sequence [16]. The rotational errors were not reported in the original paper [16]. There are no corresponding validation sequences for `fr1/teddy` and `fr1/floor`. A-CVO improves the performance over CVO and outperforms DVO on both translational and rotational metrics. On the training sequences, A-CVO reduces the average translational error of CVO by 7.2%, and on the validation sequences, the improvement reaches to 17.2%. A-CVO has a 6.9% lower translational error than Kerl et al. on the training set (excluding the failure case). On the validation set, A-CVO has 17.1% improved performance compared with Kerl et al. which shows A-CVO can generalize across different scenarios better. It is worth noting that CVO is intrinsically robust and its performance is similar to that of the state-of-the-art robust dense (and direct) RGB-D visual odometry method [13]. Next experiment will further reveal that CVO has the advantage of performing well in extreme environments that lack rich structure or texture.

C. Experiments using Structure vs. Texture Sequences

Table III shows the RMSE of RPE for the structure vs. texture sequences. This dataset contains image sequences in structure/nostructure and texture/notexture environments. As elaborated in [10], by treating point clouds as points in the function space (RKHS), CVO and A-CVO are inherently robust to the lack of features in the environment. A-CVO and CVO show the best performance on cases that either structure or texture is not rich in the environment. This reinforces the claim in [10] that CVO is robust to such scenes.

However, by online hyperparameter learning, A-CVO allows the parameters to be adaptively varying with the environment without the need for manual tuning, which improves the performance over the original CVO. We also note that DVO has the best performance on the case where the environment contains rich texture and structure information. This can be because of two reasons: 1) CVO and A-CVO adopted a semi-dense point cloud construction from DSO [18], while DVO uses the entire dense image without subsampling. Although the semi-dense tracking approach of Engel et al. [17], [18] is computationally attractive and we advocate it, the semi-dense point cloud construction process used in this work is a heuristic process and might not necessarily capture the relevant information in each frame optimally; 2) DVO uses a motion prior as regularizer whereas CVO and A-CVO solely depend on the camera information with no regularizer. We conjecture this latter is the reason DVO, relative to the training set, does not perform well on validation sequences. The motion prior is a useful assumption when it is true! It can help to tune the method better on the training sets but if the assumption gets violated can lead to

TABLE III: The RMSE of Relative Pose Error (RPE) for the structure v.s texture sequence. The Trans. columns show the RMSE of the translational drift in m/sec and the Rot. columns show the RMSE of the rotational error in deg/sec. The Average* shows the average value excluding `fr3/nostructue_notexture_near` and `fr3/nostructure_notexture_far`. The ✓means the sequence has structure/texture and × means the sequence does not have structure/texture. The results show while DVO performs better in structure and texture case, A-CVO has significantly better accuracy in the environments that lack structure and texture.

			Training								Validation							
Sequence			CVO [10]		A-CVO		Kerl et al. [16]		DVO		CVO [10]		A-CVO		Kerl et al. [16]		DVO	
structure	texture	dist.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.
×	✓	near	0.0279	1.3470	**0.0267**	**1.3033**	0.0275	n/a	0.0563	1.7560	**0.0310**	1.6367	0.0313	1.6089	n/a	n/a	0.0315	**1.1498**
×	✓	far	**0.0609**	1.2342	0.0613	**1.1985**	0.0730	n/a	0.1612	3.4135	0.1374	2.3929	**0.1158**	**2.0423**	n/a	n/a	0.5351	8.2529
✓	×	near	0.0221	**1.3689**	0.0261	1.5059	**0.0207**	n/a	0.1906	10.6424	0.0465	2.0359	**0.0405**	**1.7665**	n/a	n/a	0.1449	4.9022
✓	×	far	0.0372	1.3061	**0.0323**	**1.1114**	0.0388	n/a	0.1171	2.4044	0.0603	1.8142	**0.0465**	**1.3874**	n/a	n/a	0.1375	2.2728
✓	✓	near	0.0236	1.2972	0.0367	1.6223	0.0407	n/a	**0.0175**	**0.9315**	0.0306	1.8694	0.0394	2.2864	n/a	n/a	**0.0217**	**1.2653**
✓	✓	far	0.0409	1.1640	0.0369	1.0236	0.0390	n/a	**0.0171**	**0.5717**	0.0616	1.4760	0.0446	1.1186	n/a	n/a	**0.0230**	**0.6312**
×	×	near	0.2119	9.7944	**0.1790**	**7.0098**	n/a	n/a	0.3506	13.3127	0.1729	**5.8674**	**0.1568**	6.8221	n/a	n/a	0.1747	6.0443
×	×	far	**0.0799**	**3.0978**	0.1151	3.8035	n/a	n/a	0.1983	6.8419	0.0899	2.6199	**0.0805**	**2.4138**	n/a	n/a	0.2000	6.5192
Average*			**0.0355**	**1.2862**	0.0367	1.2942	0.0400	n/a	0.0933	3.2866	0.0612	1.8708	**0.0530**	**1.7017**	n/a	n/a	0.1490	3.0790
Average all			**0.0631**	2.5762	0.0643	**2.3223**	n/a	n/a	0.1386	4.9843	0.0787	2.4640	**0.0694**	**2.4307**	n/a	n/a	0.1586	3.8797

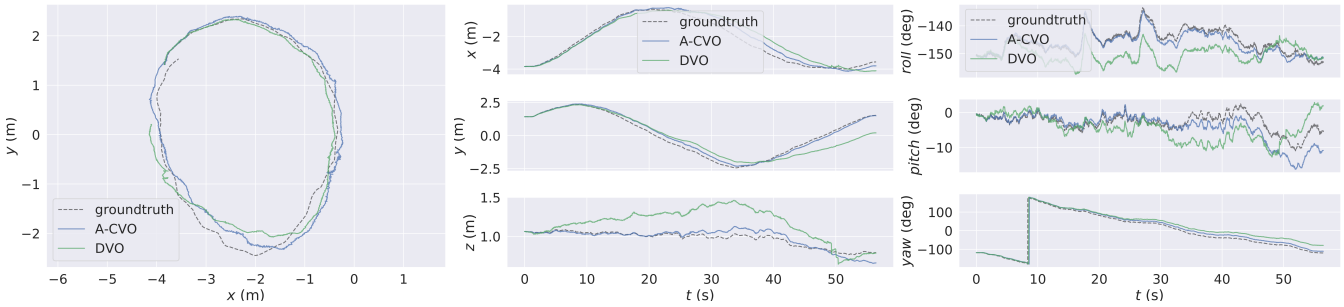

Fig. 5: The comparison of A-CVO, DVO, and groundtruth trajectory for `fr3/nostructure_texture_near` sequence. The left figure shows the trajectory in the xy plane. The middle figure shows the x, y, z trajectory with respect to time. (Note that it's not the error plot.) The figure in the right shows the angles of roll, pitch, and yaw with respect to time. From the plot, we can see A-CVO follows the groundtruth trajectory better than DVO.

poor performance. The addition of an IMU sensor, of course, can improve the performance of all the compared methods and is an interesting future research direction.

D. Discussions and Limitations

We have shown that A-CVO and CVO perform well across different indoor scenarios and different structure and texture conditions. The TUM RGB-D benchmark used in this paper was collected using a Microsoft Kinect for Xbox 360 which has a rolling shutter camera and is not designed for robotic applications. We observed that the blurred images due to camera motion are the most challenging frames for registration. The performance can degrade considerably as the extraction of the semi-dense structure cannot capture the structure and texture of the scene accurately. For example, a table edge that is usually a reliable part of an image, when blurred, can result in hallucinating multiple lines. Although more recent cameras used in robotics often use global shutters, the problem is still relevant and should be addressed. Exploring point selection strategies to improve the performance on challenging frames is also an interesting topic as future work.

The current implementation of CVO/A-CVO exploits vectorization and multi-threading which means the provided software gain additional performance benefits automatically as vector registers continue becoming wider. However, robotic applications require real-time software and more work is needed in order to achieve real-time performance using CPUs. An interesting research avenue to obtain real-time performance is a GPU implementation of the CVO/A-CVO.

VI. Conclusion and Future Work

We have developed an adaptive continuous visual odometry method for RGB-D cameras via online hyperparameter learning. The experimental results indicate that the original continuous visual odometry is intrinsically robust and its performance is similar to that of the state-of-the-art robust dense (and direct) RGB-D visual odometry method. Moreover, online learning of the kernel length-scale brings significant performance improvement and enables the method to perform better across different domains even in the absence of structure and texture in the environment.

In the future, we can use the invariant IMU model in [31] to predict the next camera pose and use the predicted pose as the initial guess in the A-CVO algorithm. This alone can increase the performance as the model performs an exact integration within a small time between two images. The integration of A-CVO into multisensor fusion systems [32]–[36] and keyframe-based odometry and SLAM systems [16], [37] are also interesting future research directions.

Acknowledgment

This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. The authors would like to thank Andreas Girgensohn for the helpful discussion on choosing the HSV colormap.

REFERENCES

[1] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," pp. 5724–5731, 2013.

[2] S. A. Scherer and A. Zell, "Efficient onbard RGBD-SLAM for autonomous MAVs," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1062–1068.

[3] R. G. Valenti, I. Dryanovski, C. Jaramillo, D. P. Ström, and J. Xiao, "Autonomous quadrotor flight using onboard rgb-d visual odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 5233–5238.

[4] G. Loianno, J. Thomas, and V. Kumar, "Cooperative localization and mapping of MAVs using RGB-D sensors," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 4021–4028.

[5] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Robotics Research*. Springer, 2017, pp. 235–252.

[6] C. Audras, A. Comport, M. Meilland, and P. Rives, "Real-time dense appearance-based SLAM for RGB-D sensors," in *Australasian Conference on Robotics and Automation*, 2011.

[7] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2011, pp. 2320–2327.

[8] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2011, pp. 719–722.

[9] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[10] M. Ghaffari, W. Clark, A. Bloch, R. M. Eustice, and J. W. Grizzle, "Continuous direct sparse visual odometry from RGB-D images," in *Proceedings of the Robotics: Science and Systems Conference*, Freiburg, Germany, June 2019.

[11] B. Schölkopf, R. Herbrich, and A. Smola, "A generalized representer theorem," in *Computational learning theory*. Springer, 2001, pp. 416–426.

[12] A. Berlinet and C. Thomas-Agnan, *Reproducing kernel Hilbert spaces in probability and statistics*. Kluwer Academic, 2004.

[13] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3748–3754.

[14] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012.

[15] S. Klose, P. Heise, and A. Knoll, "Efficient compositional approaches for real-time robust direct visual odometry from RGB-D data," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1100–1106.

[16] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for RGB-D cameras," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

[17] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proceedings of the European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[18] 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, 2018.

[19] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[20] S. Bryner, G. Gallego, H. Rebecq, and D. Scaramuzza, "Event-based, direct camera tracking from a photometric 3D map using nonlinear optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, 2019.

[21] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[22] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. MIT press, 2006, vol. 1.

[23] K. P. Murphy, *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

[24] E. Saff and A. Snider, *Fundamentals of Complex Analysis with Applications to Engineering and Science*. Prentice Hall, 2003.

[25] C. Kerl, "Dense Visual Odometry (dvo)," https://github.com/tum-vision/dvo, 2013.

[26] M. Pizenberg, "DVO core (without ROS dependency)," https://github.com/mpizenberg/dvo/tree/76f65f0c9b438675997f595471d39863901556a9, 2019.

[27] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[28] J. Canny, "A computational approach to edge detection," in *Readings in Computer Vision*. Elsevier, 1987, pp. 184–203.

[29] Intel Corporation, "Official Threading Building Blocks (TBB) GitHub repository," https://github.com/intel/tbb, 2019.

[30] M. Grupp, "evo: Python package for the evaluation of odometry and SLAM." https://github.com/MichaelGrupp/evo, 2017.

[31] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended Kalman filtering for robot state estimation," *arXiv preprint arXiv:1904.09251*, 2019.

[32] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[33] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 1885–1892.

[34] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[35] R. Hartley, M. G. Jadidi, L. Gan, J.-K. Huang, J. W. Grizzle, and R. M. Eustice, "Hybrid contact preintegration for visual-inertial-contact state estimation within factor graphs," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, October 2018, pp. 3783–3790.

[36] K. Eckenhoff, P. Geneva, and G. Huang, "Closed-form preintegration methods for graph-based visual–inertial navigation," *International Journal of Robotics Research*, vol. 38, no. 5, pp. 563–586, 2019.

[37] 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.