Unsupervised Deep Visual-Inertial Odometry with Online Error Correction for RGB-D Imagery

E. Jared Shamwell, *Member, IEEE*, Kyle Lindgren, *Student Member, IEEE*, Sarah Leung, and William D. Nothwang

Abstract—While numerous deep approaches to the problem of vision-aided localization have been recently proposed, systems operating in the real world will undoubtedly experience novel sensory states previously unseen even under the most prodigious training regimens. We address the localization problem with online error correction (OEC) modules that are trained to correct a vision-aided localization network's mistakes. We demonstrate the generalizability of the OEC modules and describe our unsupervised deep neural network approach to the fusion of RGB-D imagery with inertial measurements for absolute trajectory estimation. Our network, dubbed the Visual-Inertial-Odometry Learner (VIOLearner), learns to perform visual-inertial odometry (VIO) without inertial measurement unit (IMU) intrinsic parameters or the extrinsic calibration between an IMU and camera. The network learns to integrate IMU measurements and generate hypothesis trajectories which are then corrected online according to the Jacobians of scaled image projection errors with respect to spatial grids of pixel coordinates. We evaluate our network against state-of-the-art (SoA) VIO, visual odometry (VO), and visual simultaneous localization and mapping (VSLAM) approaches on the KITTI Odometry dataset as well as a micro aerial vehicle (MAV) dataset that we collected in the AirSim simulation environment. We demonstrate better than SoA translational localization performance against comparable SoA approaches on our evaluation sequences.

Index Terms—Visual-inertial odometry, unsupervised deep learning, refinement, localization, neural networks

1 Introduction

Despite remarkable successes in computer vision related tasks such as image classification, deep approaches are still generally outperformed by comparable shallow approaches in the domains of vision-aided localization and navigation (e.g., visual odometry (VO), visual-inertial odometry (VIO), and visual simultaneous localization and mapping (VSLAM)). While unsupervised approaches have helped to overcome the supervised dataset problems, robustness still plagues deep approaches to vision-aided localization. In part, this is due to the very low error tolerances of localization problems-a single mistake while estimating the individual poses that constitute a trajectory can cause catastrophic failure and render the entire localization solution unusable.

It is impractical to train feedforward networks to handle every foreseeable situation that might occur in the real world. Instead, we propose to address the localization problem by baking in online error correction (OEC) modules that can be trained to detect and correct mistakes in intermediate network outputs and ultimately produce lower error estimates. A high-level overview of our architecture and its OEC modules can be seen in Fig. 1.

Here we describe Visual-Inertial-Odometry Learner (VIOLearner) which is an unsupervised visual-inertial odometry network that learns to correct its own mistakes at runtime using novel trainable OEC modules. The OEC modules correct feedforward errors from intermediate network outputs to prevent catastrophic failures and trajectory divergences.

While our approach is capable of operating with only monocular imagery and inertial measurement unit (IMU) measurements when RGB-D imagery is unavailable (and thus will generate unscaled trajectories; see Section 7.7.1 for more detail), we have chosen to include depth in our input domain as a means of achieving absolute scale recovery: while absolute depth can be generated using an onboard sensor, the same cannot be said for the change in pose between image pairs.

We demonstrate superior performance compared to similar state-of-the-art (SoA) approaches. VIOLearner achieves SoA performance on KITTI [1] for learning-based approaches. To our knowledge, we are only the second deep visual-inertial approach (the first being VINet of [2]). Additionally, we show how the OEC modules provide increased VIO robustness against spatial misalignments between a camera and IMU that plague SoA traditional approaches to VIO [3], [4], [5].

In addition to results presented in [6], here we thoroughly evaluate the benefit of the OEC modules. In particular, we separate the effects of the OEC modules from the pose estimates extracted from IMU measurements by leveraging the highly controlled nature of robotic simulation environments. We evaluate our approach on simulated data from a quadcopter in the Unreal Engine backed AirSim micro aerial

- *E.J. Shamwell, S. Leung, and W.D. Nothwang are with the US Army Research Laboratory, Adelphi, MD 20783. E-mail: {earl.j.shamwell.civ, sarah.leung.ctr, william.d.nothwang.civ}@mail.mil.*
- *K. Lindgren is with the US Army Research Laboratory, Adelphi, MD 20783, and also with the BioRobotics Lab, University of Washington, Seattle, WA 98195. E-mail: kyle.m.lindgren.ctr@mail.mil.*

Manuscript received 11 Oct. 2018; revised 4 Jan. 2019; accepted 19 Mar. 2019. Date of publication 15 Apr. 2019; date of current version 2 Sept. 2020. (Corresponding author: E. Jared Shamwell.)
Recommended for acceptance by M. Bennamoun and Y. Guo.
Digital Object Identifier no. 10.1109/TPAMI.2019.2909895

vehicle (MAV) simulation package [7]. As described later in Section 5.4.2, we are able to vary IMU-camera extrinsics which allows us to empirically separate the effects of the different modules in VIOLearner.

The main contribution of the approach here described is the unsupervised learning of trajectory estimates with absolute scale recovery from RGB-D + inertial measurements with

- Built-in OEC modules;
- Real-time operation; and
- Loosely temporally synchronized camera and IMU.

In this paper, we provide an exhaustive evaluation showing the benefit of the online error corrector by validating VIOLearner against a second dataset, performing depth ablation studies to demonstrate our OEC modules and multi-hypothesis approach, and analyzing VIOLearner for robustness to spatial misalignments of a camera and an IMU.

The remainder of the paper is organized as follows: Section 2 provides background information and motivations; Section 3 presents related work; Section 4 outlines our deep network approach; Section 5 describes our architecture and experimental approach; Section 6 describes our evaluation procedures; Section 7 presents and discusses our experimental results; and Section 8 offers concluding thoughts and directions for future work.

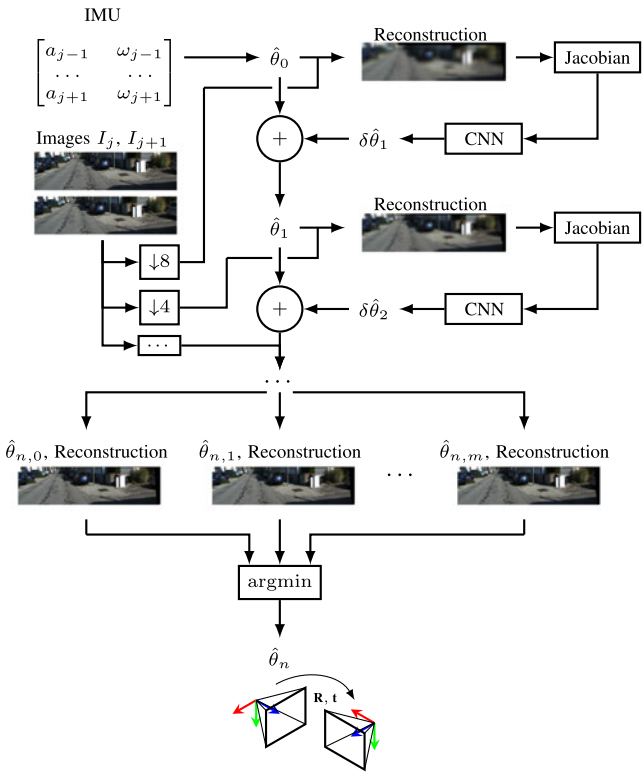


Fig. 1. Generalized overview of our VIOLearner network (see Fig. 2 for a more detailed/accurate representation of the architecture). Note the hierarchical architecture where θ, the true change in pose between images, is estimated at multiple scales and then corrected via convolutional processing of the Jacobian of euclidean projection error at that scale with respect to a grid of source coordinates. The final Level n computes m hypothesis reconstructions (and associated estimates $\hat{\theta}_{n,m}$) and the $\hat{\theta}_{n,m}$ with the lowest paired euclidean projection error between the reconstruction and true target image I_{j+1} is output as the final network estimate $\hat{\theta}_n$.

2 Background

2.1 Vision-Aided Localization

Originally coined to characterize honey bee flight [8], the term "visual odometry" describes the integration of apparent image motions for dead-reckoning-based navigation (literally, as an odometer for vision) [9]. While work in what came to be known as visual odometry (VO) began in the 1980s for the Mars Rover project [10], [11], the term was not popularized in the engineering context until around 2004 [12].

While VO and VSLAM (described below in Section 3.1) can be performed by a monocular camera system, only scaled rotation can be recovered. Monocular VO approaches recover translation up to an unknown scale [13], [14], [15], [16] as depth is unobservable from monocular cameras and thus there is ambiguity in the scale of translational estimates.

Successful large-scale monocular systems for autonomous navigation are uncommon, primarily due to scale drift [17]. To recover scale, depth or some other fixed reference is needed (see below in Sections 2.1.1 and 2.1.2). As described below, RGB-D data is an ideal source of information for vision-aided localization and is becoming increasingly available on robotic platforms.

2.1.1 Visual-Inertial

Approaches in VIO combine visual estimates of motion with those measured by multi-axis accelerometers and gyroscopes in IMUs. As IMUs measure only linear accelerations and angular velocities, inertial approaches to localization are prone to exponential drift over time due to the double integration of accelerations into pose estimates. Combining inertial estimates of pose change with visual estimates allows for the 'squashing' of inertial estimates of drift. For VIO, integrating raw IMU measurements can provide noisy, short-term estimates of scale.

While IMU measurements can help resolve scale ambiguity, gravity must be separated from the measured linear accelerations which means that the orientation of the IMU in the world frame must be correctly estimated [18]. Accumulated map drift introduces orientation error that leads to scale error from which systems can not always recover.

2.1.2 Stereo and RGB-D Imagery

For stereo imagery a metric baseline distance between cameras is known and with this information, translational scale and depth can be recovered, potentially alleviating visual-inertial approaches of the full burden of scale estimation. However, a downside of using stereo cameras to infer depth is that stereo cameras have effective ranges that are inversely proportional to the baseline between the stereo camera pair: small camera baselines produce high-error depth estimates for scene elements further from the cameras while large camera baselines fail to capture depth close to the cameras and introduce calibration difficulties which can lead to poor performance [19] (not to mention that maximum baselines are system/vehicle dependent-a 1 m stereo camera baseline on an 0.35 m diagonal MAV would be quite impractical, for example).

RGB-D data is simply RGB imagery with a fourth channel that contains the corresponding metric depth at each pixel location. As mentioned earlier, this depth can be measured by stereo cameras, an RGB-D sensor (e.g., Microsoft Kinect, Asus

Xtion, Orbec Astra, or new indoor/outdoor Intel RealSense cameras), or LIDARs (e.g., a Velodyne VLP-16 or PUCK Lite) directly onboard a vehicle. Camera RGB images along with measurements from scanning LIDARs can be converted to RGB-D imagery assuming extrinsic and intrinsic sensor calibrations are known (see Section 5.3.1 for more detail).

3 Related Work

3.1 Traditional Methods

In VO and VSLAM, only data from camera sensors is used and tracked across frames to determine the change in the camera pose. Simultaneous localization and mapping (SLAM) approaches typically consist of a front end, in which features are detected in the image, and a back end, in which features are tracked across frames and matched to keyframes to estimate camera pose, with some approaches performing loop closure as well. ORB-SLAM2 [20] is a VSLAM system for monocular, stereo, and RGB-D cameras. It used bundle adjustment and a sparse map for accurate, real-time performance on CPUs. ORB-SLAM2 performed loop closure to correct for the accumulated error in its pose estimation. ORB-SLAM2 has shown SoA performance on a variety of VO benchmarks.

In VIO and visual-inertial SLAM, the fusion of imagery and IMU measurements are typically accomplished by filter-based approaches or nonlinear optimization approaches. The multi-state constraint Kalman filter (MSCKF) [21] has become a standard for filtering-based approaches to VIO. While its complexity is linear in the number of features being used for egomotion estimation and it was generally more robust compared to optimization-based approaches (as has been recently reported in [22]), MSCKF approaches are typically less accurate in comparison. ROVIO [5] is another filtering-based VIO algorithm for monocular cameras, which used a robust and efficient robocentric approach in which 3D landmark positions were estimated relative to the current camera pose. It used an Extended Kalman Filter (EKF) to fuse the sensor data, utilizing the intensity errors in the update step. However, because ROVIO is a monocular approach, accurate scale is not recovered. OKVIS [3] is a optimization-based keyframe visual-inertial SLAM approach for monocular and stereo cameras. OKVIS relied on keyframes, which consisted of an image and estimated camera pose, a batch nonlinear optimization on saved keyframes, and a local map of landmarks to estimate camera egomotion. However, it did not include loop closure, unlike the SLAM algorithms with SoA performance.

There are also several approaches which enhance VIO with depth sensing or laser scanning. Two methods of depth-enhanced VIO are built upon the MSCKF [21], [23] algorithm mentioned above. One method is the MSCKF-3D [24] algorithm, which used a monocular camera with a depth sensor, or RGB-D camera system. The algorithm performed online time offset correction between camera and IMU, which is critical for its estimation process, and used a Gaussian mixture model for depth uncertainty. Pang et al. [25] also demonstrated a depth-enhanced VIO approach based on MSCKF, with 3D landmark positions augmented with sparse depth information kept in a registered depth map. Both approaches showed improved accuracy over VIO-only approaches. [26] also demonstrated dense RGB-D-Inertial SLAM using a hand-held RGB-D camera in small, indoor environments. Finally, Zhang and Singh [27] proposed a method for leveraging data from a 3D laser. The approach utilized a multi-layer pipeline to solve for coarse to fine motion by using VIO as a subsystem and matching scans to a local map. It demonstrated high position accuracy and was robust to individual sensor failures.

3.2 Learning-Based Methods

Recently, there have been several successful unsupervised approaches to depth estimation that are trained using reconstructive loss from image warping similar to our own network. Garg et al. [28], Godard et al. [29], Zhan et al. [30], and Smolyanskiy et al. [31] used such methods with stereo image pairs with known camera baselines and reconstructive loss for training. Thus, while technically unsupervised, the known baseline effectively provides a known transform between two images. Our network approaches the same problem from the opposite direction: we assume known depth and estimate an unknown pose difference.

Pillai and Leonard [32] demonstrated visual egomotion learning by mapping optical flow vectors to egomotion estimates via a mixture density network (MDN). Their approach not only required optical flow to already be externally generated (which can be very computationally expensive), but also was trained in a supervised manner and thus required the ground truth pose differences for each exemplar in the training set.

SFMLearner [33] demonstrated the unsupervised learning of unscaled egomotion and depth from RGB imagery. They input a consecutive sequence of images and output a change in pose between the middle image of the sequence and every other image in the sequence, and the estimated depth of the middle image. However, their approach was unable to recover the scale for the depth estimates or, most crucially, the scale of the changes in pose. Thus, their network's trajectory estimates needed to be scaled by parameters estimated from the ground truth trajectories and in the real world, this information will of course not be available. SFMLearner also required a sequence of images to compute a trajectory. Their best results were on an input sequence of five images whereas our network only requires a source-target image pairing. The recent work of [34] extended SFMLearner with a semi-differentiable iterative closest point (ICP) module.

UnDeepVO [35] is another unsupervised approach to depth and egomotion estimation. It differs from [33] in that it was able to generate properly scaled trajectory estimates. However, unlike [33] and similar to [28], [29], it used stereo image pairs for training where the baseline between images is known and thus, UnDeepVO can only be trained on datasets where stereo image pairs are available. Additionally, the network architecture of UnDeepVO cannot be extended to include motion estimates derived from inertial measurements because the spatial transformation between paired images from stereo cameras are unobservable by an IMU (stereo images are recorded simultaneously).

VINet [2] was the first end-to-end trainable visual-inertial deep network. While VINet showed robustness to temporal and spatial misalignments of an IMU and camera, it still required extrinsic calibration parameters between camera and IMU. This is in contrast to our VIOLearner which

requires no IMU intrinsics or IMU-camera extrinsics. In addition, VINet was trained in a supervised manner and thus required the ground truth pose differences for each exemplar in the training set which are not always readily available.

Finally, LS-Net [36] was a recent approach to learning-based monocular multi-view stereo and egomotion estimation. While VIOLearner [6], [37], which the present work extends, was the first approach to use a learned optimizer to minimize photometric loss for egomotion estimation, [36] similarly leveraged Jacobians and optimized both for egomotion as well as depth. However, their approach is supervised and required ground truth. Additionally, their approach, is not end-to-end trainable in practice (the initialization network must be pre-trained before the optimization network). DeepTAM [38] is another recent approach to learning-based VSLAM that also learns to refine depth predictions and operates at multiple scales, similar to our approach.

3.3 Existing Datasets

While numerous datasets aimed at vision-aided odometry exist [39], [40], [41], [42], [43], [44], few heterogeneous datasets of sizes suitable for deep learning applications on RGB-D data are available. The KITTI dataset [1] included RGB and depth data captured over 39.2 km and was ground truthed with accuracies within 10cm via an OXTS RT 3003 GPS solution. However, KITTI only used software synchronization between sensors which causes issues for some VIO approaches due to the inaccurate time synchronization between imagery and IMU data.

We have elected to use the KITTI odometry dataset to evaluate our approach and details can be found in Sections 5.3 and 5.3.1. As described and explained in Section 5.4, we also elected to generate our own RGB-D datasets using the AirSim MAV simulation environment [45] to provide an aerial vehicle benchmarking dataset that includes depth, in contract to the popular EuRoC MAV [43] and Zurich Urban MAV [41] datasets.

4 Approach

VIOLearner is an unsupervised VIO deep network that estimates the scaled egomotion of a moving camera between some time t_j at which source image I_j is captured and time t_{j+1} when target image I_{j+1} is captured. VIOLearner receives an input RGB-D source image, a target RGB image, IMU data from t_{j-1} to t_{j+1}, and a camera calibration matrix K with the camera's intrinsic parameters. With access to K, VIOLearner can generate camera pose changes in the camera frame using a view-synthesis approach where the basis for its training is in the euclidean loss between a target image and a reconstructed target image generated using pixels in a source image sampled at locations determined by a learned 3D affine transformation (via a spatial transformer of [46]).

4.1 Multi-Scale Projections and Online Error Correction

Similar to [33], VIOLearner performs multi-scale projections. We scale projections by factors of 8, 4, and 2. However, in our network, multi-scale projections not only help to overcome gradient locality during training (see [33], [47] for a broader discussion of this well-known issue), but also aid in online error correction at runtime.

Generally, at each level, the network computes the Jacobians of the reprojection error at that level with respect to the grid of coordinates. Convolutions are then performed on this Jacobian (sized HxWx2) and a $\delta\hat{\theta}$ is computed. This $\delta\hat{\theta}$ is added to the previously generated affine matrix $\hat{\theta}$. This is repeated for each level in the hierarchy. VIOLearner uses a total of 5 levels and additional detail on the computations performed at each level is provided in the following sections.

Generally, a Jacobian of a minimum size HxWx6 would be needed for traditional least-squares optimization. While our HxWx2 Jacobian only directly represents 2D information and does not include error with respect to depth, the convolutional processing of these Jacobians enables it to glean 3D information from this 2D representation and update the 6-DOF camera pose. In practice, we found that using this compressed HxWx2 2D Jacobian was an effective 6-DOF pose optimizer.

4.2 Level 0

VIOLearner first takes raw IMU measurements and learns to compute an estimated 3D affine transformation $\hat{\theta}_0$ that will transform a source image I_j into a target image I_{j+1} (the top left of Fig. 2a). The network downsamples ($\downarrow$ in Fig. 2) the source image I_j (and associated depth and adjusted camera matrix K for the current downsampling factor) by a factor of 8 and applies the 3D affine transformation to a normalized grid of source coordinates $[X_{src}, Y_{src}]$ to generate a transformed grid of target coordinates $[X_{tgt}, Y_{tgt}]$.

VIOLearner then performs bilinear sampling to produce a reconstruction image Ir by sampling from the source image I_j at coordinates $(x, y) \in [X_{tgt}, Y_{tgt}]$

$$Ir(x,y) = \sum_{w,h}^{W,H} I_j(w,h)\,\max(0, 1-|x-w|)\,\max(0, 1-|y-h|). \tag{1}$$

As in [46], by only evaluating at the sampling pixels, we can therefore compute sub-gradients that allow error back-propagation to the affine parameters $\hat{\theta}_0$ by computing error with respect to the coordinate locations (x, y). For each pixel k, these Jacobians are computed as

$$\frac{\partial Ir_k}{\partial x_k} = \sum_{w,h}^{W,H} I_j(w,h)\max(0, 1-|y_k-h|)\begin{cases} 0, |w-x_k| \geq 1 \\ 1, w \geq x_k \\ -1, w < x_k \end{cases} \tag{2}$$

$$\frac{\partial Ir_k}{\partial y_k} = \sum_{w,h}^{W,H} I_j(w,h)\max(0, 1-|x_k-w|)\begin{cases} 0, |h-y_k| \geq 1 \\ 1, h >= y_k. \\ -1, h < y_k. \end{cases} \tag{3}$$

Starting with Level 0, the euclidean loss is taken between the downsampled reconstructed target image and the actual target image. For Level 0, this error is computed as

$$E_0 = \|Ir^0 - I^0_{j+1}\|^2. \tag{4}$$

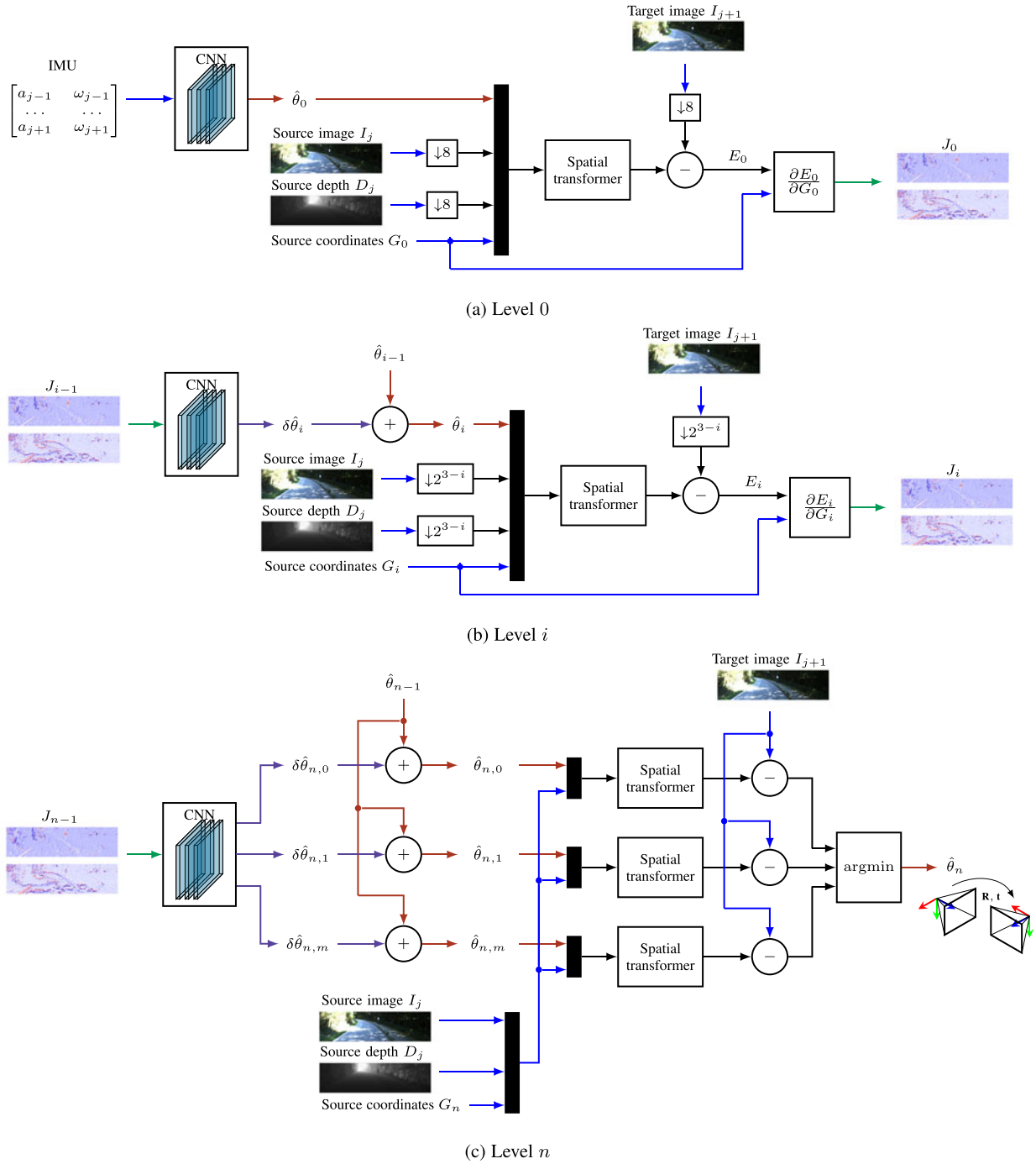


Fig. 2. VIOLearner: (a) IMU data is fed into a series of convolutional neural network (CNN) layers which output a 3D affine matrix estimate of change in camera pose $\hat{\theta}_0$ between a source image I_j and target image I_{j+1}. The transform $\hat{\theta}_0$ is applied to a downsampled source image via a spatial transformer module and the euclidean error between the spatial transformation and the downsampled target image is computed as E_0. The Jacobian $\frac{\partial E_0}{\partial G_0}$ of the error image E_0 is taken with respect to the source coordinates G_0, and fed to the next level. (b) The Jacobian $\frac{\partial E_{i-1}}{\partial G_{i-1}}$ from the previous level is fed through CNN layers to produce an additive refinement $\partial\hat{\theta}_i$ that is summed with the previous transform estimate $\hat{\theta}_{i-1}$ to produce a new transform estimate $\hat{\theta}_i$. The Jacobian $\frac{\partial E_i}{\partial G_i}$ is propagated forward. (c) In the final level, the previous Jacobian $\frac{\partial E_{n-1}}{\partial G_{n-1}}$ is processed through CNN layers for m hypothesis refinements.

The final computation performed by Level 0 is of the Jacobian of the euclidean loss of Equation (4) with respect to the source coordinates G_0 from Equations (2) and (3). The resulting Jacobian matrix J_0 has the same dimensionality as the grid of source coordinates G_0 ($\frac{H}{8}$x$\frac{W}{8}$x2) and is depicted in Fig. 2a. In traditional approaches, the gradient and error equations above are only used during training. However, VIOLearner is novel in that it also computes and employs these gradients during each inference step of the network. During both training and inference, the Jacobian J_0 is computed and passed to the next level for processing.

4.3 Levels i to n-1

For the intermediate levels i through $n-1$, the previous level's Jacobian J_{i-1} is input and processed through layers of convolutions to generate a $\partial\hat{\theta}_i$. This $\partial\hat{\theta}_i$ represents a

computed correction to be applied to the previous level's 3D affine transform $\hat{\theta}_{i-1}$. $\partial\hat{\theta}_i$ is summed with $\hat{\theta}_{i-1}$ to generate $\hat{\theta}_i$ which is then applied to generate a reconstruction that is downsampled by a factor 2^{3-i}. Error is again computed as above in Equation (4) and the Jacobian is similarly found as it was in Level 0 and input to the next Level $i+1$.

4.4 Level n and Multi-Hypothesis Pathways

The final level of VIOLearner employs multi-hypothesis pathways similar to [48] where several possible hypotheses for the reconstructions of a target image (and the associated transformations $\hat{\theta}_m$, $m \in M$ which generated those reconstructions) are computed in parallel. The lowest error hypothesis reconstruction is chosen during each network run and the corresponding affine matrix $\hat{\theta}_{m*}$ which generated the winning reconstruction is output as the final network estimate of camera pose change between images I_j and I_{j+1}.

This multi-hypothesis approach allows the network to generate several different pathways and effectively sample from an unknown noise distribution. For example, as IMUs only measure linear accelerations, they fail to accurately convey motion during periods of constant velocity. Thus, a window of integrated IMU measurements are contaminated with noise related to the velocity at the beginning of the window. With a multi-hypothesis approach, the network has a mechanism to model uncertainty in the initial velocity (see Section 7.4 for a discussion).

Error for this last multi-hypothesis level is computed according to a winner-take-all (WTA) euclidean loss rule (see [48] for more detail and justifications)

$$Ir^{n,*} \longleftarrow \underset{k}{\operatorname{argmin}} \|Ir^{n,k} - I^n_{j+1}\|^2 \tag{5}$$

$$E_n = \|Ir^{n,*} - I^n_{j+1}\|^2, \tag{6}$$

where $Ir^{n,*}$ is the lowest error hypothesis reconstruction. Loss is then only computed for this one hypothesis pathway and error is backpropagated only to parameters in that one pathway. Thus, only parameters that contributed to the winning hypothesis are updated and the remaining parameters are left untouched.

The final loss $\mathcal{L}$ by which the network is trained is then simply the sum of the euclidean loss terms for each level plus a weighted L1 penalty over the bias terms which we empirically found to better facilitate training and gradient backpropagation

$$\mathcal{L} = \sum_{i=0}^{n} E_i + \lambda|bias|. \tag{7}$$

5 Methods

5.1 Network Architecture

5.1.1 IMU Processing

The initial level of VIOLearner uses two parallel pathways of 7 convolutional layers for the IMU angular velocity and linear accelerations, respectively (see Fig. 1 for more detail on the architecture). Each pathway begins with 2 convolutional layers each of 64 single-stride 3 x 5 filters on the $batch$ x 20 x 3 IMU angular velocity or linear accelerations followed by 2 convolutional layers of 128 filters each of stride 2 with the same 3 x 5 kernel. Next, 3 convolutional layers of 256 filters are applied with strides of 2, 1, and 1, and kernels of size 3 x 5, 3 x 3, and 3 x 1. The final convolutional layer in the angular velocity and linear acceleration pathways were flattened into $batch$ x 1 x 3 tensors using a convolutional layer with three filters of kernel size 1 and stride 1 before being concatenated together into a tensor $pose\_imu$.

5.1.2 3D Affine Transformations

The first three components in *pose_imu* correspond to rotations $[\alpha_0, \beta_0, \gamma_0]$ representing rotations about the x, y, and z axes respectively. Rotation matrices are computed as

$$R^0_x(\alpha_0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_0 & -\sin\alpha_0 \\ 0 & \sin\alpha_0 & \cos\alpha_0 \end{bmatrix} \tag{8}$$

$$R^0_y(\beta_0) = \begin{bmatrix} \cos\beta_0 & 0 & \sin\beta_0 \\ 0 & 1 & 0 \\ -\sin\beta_0 & 0 & \cos\beta_0 \end{bmatrix} \tag{9}$$

$$R^0_z(\gamma_0) = \begin{bmatrix} \cos\gamma_0 & -\sin\gamma_0 & 0 \\ \sin\gamma_0 & \cos\gamma_0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{10}$$

and a 3D rotation matrix is generated as

$$R^0 = R^0_z R^0_y R^0_x. \tag{11}$$

The last three elements in $pose\_imu$ directly correspond to a translation vector $T^0 = [x_0, y_0, z_0]^\top$. Together with the 3D rotation matrix R^0, we finally form a 4 x 4 homogeneous transformation matrix $\hat{\theta}_0$ as

$$\hat{\theta}_0 = \begin{bmatrix} R^0 & T^0 \\ 0,0,0 & 1 \end{bmatrix}. \tag{12}$$

5.1.3 Online Error Correction and Pose Refinement

For each $\frac{H}{s}$x$\frac{W}{s}$x2 Jacobian matrix $\frac{\partial E_i}{\partial G_i}$ at all scales $s \in S$, three convolutional layers of 128 filters are applied with kernel sizes 7 x 7, 5 x 5, and 3 x 3 and strides of 1, 2, and 2, respectively. Then, an additional 2 to 5 convolutional layers are applied depending on the downsampling factor of the current level with the number of additional layers increasing as the downsampling factor decreases (the final level using 5 additional convolutional layers). For each level, the final layer generates a pose estimate $pose\_refinement_i$ using a single-strided convolution with a kernel size of 1.

The outputs $pose\_refinement_i$ are split similarly to $pose\_imu$ into rotations $[\partial\alpha_i, \partial\beta_i, \partial\gamma_i]$ and translation $T^i = [\partial x_i, \partial y_i, \partial z_i]^\top$. The new rotations and translations for level i are then computed as

$$[\alpha_i, \beta_i, \gamma_i] = [\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}] + [\partial\alpha_i, \partial\beta_i, \partial\gamma_i] \tag{13}$$

$$[x_i, y_i, z_i]^\top = [x_{i-1}, y_{i-1}, z_{i-1}]^\top + [\partial x_i, \partial y_i, \partial z_i]^\top. \tag{14}$$

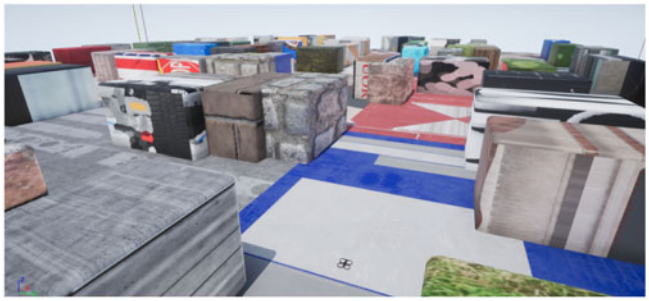


Fig. 3. The virtual *Blocks* environment exhibits visually unique imagery through use of an array of object materials and textures.

This repeats for each level until the final level where $\hat{\theta}_n$ is output as the final estimate of the change in camera pose.

5.1.4 Multi-Hypothesis Generation

We experimented with using between $1-8$ hypothesis pathways in the final level of VIOLearner as described above in Section 4.4.

5.2 Training Procedures

VIOLearner was trained for 100,000 iterations using a batch size of 16. As the network was trained, we calculated error on the validation set at intervals of 1000 iterations. We used the Adam solver with momentum1=0.9, momentum2=0.99, gamma=0.5, learning rate=$2e{-}4$, and an exponential learning rate policy. Additional parameters for the VIOLearner network architecture are shown in Table 1. The network was trained using single-point precision (FP32) on a desktop computer with a 3.00 GHz Intel i7-6950X processor and Nvidia Titan X GPUs.

5.3 KITTI Odometry Dataset

We evaluate VIOLearner on the KITTI Odometry dataset [1] and used sequences $00-08$ for training excluding sequence 03 because the corresponding raw file $2011\_09\_26\_drive\_0067$ was not online at the time of publication. Sequences 09 and 10 were withheld for the test set as was done in [33]. Additionally, 5 percent of KITTI sequences $00-08$ was withheld as a validation set. This left a total of 18,422 training images, 2,791 testing images, and 969 validation images.

In all experiments, we randomly selected an image for the source and used the successive image for the target. Corresponding 100 Hz IMU data was collected from the KITTI raw datasets and for each source image, the preceding 100 ms and the following 100 ms of IMU data was combined yielding a length 20 x 6 vector (100 ms prior to the source image and the approximately 100 ms between source and target image). We chose to include IMU data in this way so that the network could learn how to implicitly estimate a temporal offset between camera and IMU data as well as glean an estimate of the initial velocity V_0 at the time of source image capture by looking to previous data.

5.3.1 RGB-D Generation

KITTI includes pointclouds captured using a Velodyne LIDAR as well as calibration extrinsics between LIDAR and cameras. However, depth is not available for the full resolution of KITTI images so we cropped each image in KITTI from 376 x 1241 to 224 x 1241 (and first resized each image to 376 x 1241 if the resolution was different as is the case for certain sequences) and then scaled the cropped images to size 128 x 480.

To generate RGB-D imagery, we interpolated the sparse Velodyne point clouds and projected them onto the camera imaging plane using the provided KITTI extrinsic and camera calibration matrices.

5.4 AirSim MAV Simulation Environment

An experimental environment (Fig. 3) was built within Unreal Engine, a game development platform which enables realistic, high-fidelity visualizations and physics simulations. Drone flight and control leverage the Microsoft AirSim plugin [45] for simulated flight dynamics as well as a suite of sensors including inertial (GPS, barometer, gyroscope, accelerometer) and imaging (RGB, depth, segmentation). For this work, 128x480 resolution RGB-D imagery is captured at 10 Hz with IMU measurements taken at 100 Hz.

The environment utilized simulates a maze-like block arena, denoted *Blocks*. A total of 20 trajectories were

TABLE 1
Network Architectures of VIOLearner

Sub-network	Layer	1	2	3	4	5	6	7	8	9
IMU (Angular Velocity and Linear Acceleration)	No. Filters	64	64	128	128	256	256	256		
	Kernel Size	3 x 5	3 x 5	3 x 5	3 x 5	3 x 5	3 x 3	3 x 1		
	Stride	1	1	2	2	2	1	1		
OEC Module (↓8)	No. Filters	128	128	128	64	32	6			
	Kernel Size	7 x 7	5 x 5	3 x 3	3 x 3	1 x 6	1 x 1			
	Stride	1	2	2	2	1	1			
OEC Module (↓4)	No. Filters	128	128	128	64	32	32	6		
	Kernel Size	7 x 7	5 x 5	3 x 3	3 x 3	3 x 3	1 x 6	1x1		
	Stride	1	2	2	2	2	1	1		
OEC Module (↓2)	No. Filters	128	128	128	64	64	32	32	6	
	Kernel Size	7 x 7	5 x 5	3 x 3	3 x 3	3 x 3	3 x 3	1 x 6	1 x 1	
	Stride	1	2	2	2	2	2	1	1	
OEC Module (↓1)	No. Filters	128	128	128	256	256	128	64	32	6
	Kernel Size	7 x 7	5 x 5	3 x 3	3 x 3	3 x 3	3 x 3	3 x 3	1 x 6	1 x 1
	Stride	1	2	2	2	2	2	2	1	1

captured within the *Blocks* environment, with each trajectory containing 2 minutes of flight time. The dataset was split with 70 percent of trajectories in the training set, 15 percent in the validation set, and 15 percent in the test set.

5.4.1 Trajectory Characteristics

Manual flight control through the environments provided turn and velocity inputs for each trajectory, with AirSim managing drone dynamics such that turns and changes in velocity apply realistic effects to roll, pitch, and yaw motion. Velocity was kept between 2 m/s and 4 m/s while vertical positioning of the aerial vehicle remained fixed throughout all flights.

5.4.2 Generating IMU-Camera Configurations

Post-processing of AirSim generated ground truth vehicle kinematics facilitates analysis of network performance on trajectories that differ in IMU sensor configurations. Additionally, vehicle linear accelerations and angular velocities for any IMU position and orientation configuration can be calculated without the need for flight playback.

Linear acceleration observed at any desired point on the rigid body vehicle can be calculated using the linear acceleration a, angular velocity ω, and angular acceleration α vectors of a known point on the vehicle, in addition to the transformation matrix θ, containing the rotation matrix R and translation vector T between the two points [49]. Linear acceleration of the desired point is calculated by $\hat{a} = a_T + a_R$, where the tangential component a_T is the linear acceleration of the known point, and the radial component a_R is found from the derivative of linear velocity with respect to time.

$$v = \omega \times T$$

$$a_R = \frac{dv}{dt} = \frac{dw}{dt} \times T + w \times \frac{dT}{dt}, \tag{15}$$

where

$$\frac{dw}{dt} = \alpha, \ \frac{dT}{dt} = v,$$

giving

$$a_R = \alpha \times T + w \times v. \tag{16}$$

Angular velocity, the other IMU measurement of interest here, is shared among all points on a rigid body, so $\hat{\omega} = \omega$.

Noise is added to both generated measurements to increase similarity to real world sensor performance. The noise model injects two types of sensor errors: white noise n, and sensor bias b. A noised linear acceleration measurement, for instance, is generated by

$$\tilde{a}(t) = a(t) + b(t) + n(t), \tag{17}$$

with both sensor error types sampled at each time t.

Lastly, generated linear accelerations and angular velocities undergo rotations according to R to account for changes in orientation between the known and desired IMU poses.

6 Evaluation

We compare our approach to a collection of recent VO, VIO, and VSLAM approaches described earlier in Section 3:

- OKVIS (Stereo) [3]
- SFMLearner [33]
- Mahjourian et al. [34] (results reproduced from [34])
- Zhan et al. [30] (results reproduced from [30])
- VINet [2] (results reproduced from [2])
- UnDeepVO [35] (results reproduced from [35])
- VISO2 (Mono) [50] (results reproduced from [35])
- ORB-SLAM (Mono) [15] (results reproduced from [34])
- ORB-SLAM2 (Stereo) [20]
- EKF+VISO2 (results reproduced from [2])

As was affirmed in the recent work of Delmerico et al. [22], optimization based approaches to VIO (e.g., OKVIS [3]) typically outperform stochastic filtering based approaches (e.g., MSCKF [21]). We have thus chosen OKVIS [3] as the primary comparator to SoA VIO approaches.

We include stereo versions of competing algorithms where available to provide scale information to more accurately compare to our RGB-D approach. In the literature, there is no readily comparable approach with the exact input and output domains as our network (namely RGB-D + inertial inputs and scaled output odometry; see Section 3 for approaches with our input domain that do not publicly provide their code or evaluation results on KITTI 09 and KITTI 10). To overcome this limitation, we also include approaches that use far more data to compute camera transforms than ours (i.e., SFMLearner, Mahjourian et al., Zhan et al., and VINet optimize over multiple consecutive monocular images or stereo image-pairs; OKVIS, ORB-SLAM, and ORB-SLAM2 perform bundle adjustment; and ORB-SLAM2 performs full bundle adjustment and loop closure).

We perform 6-DOF least-squares Umeyama alignment [51] for trajectory alignment on SFMLearner, OKVIS, and ORB-SLAM2. For SFMLearner, we follow [33] to estimate scale from ground truth for each estimate. It should be noted that OKVIS and ORB-SLAM2 were evaluated at images scaled down to size 128 x 480 to match the image resolution used by VIOLearner.

It should be noted that we trained separate networks for KITTI and AirSim. VIOLearner implicitly learns to estimate camera-IMU extrinsics and IMU instrinsics directly from the data which means that VIOLearner does not currently translate from one dataset (with a given camera-IMU configuration) to another (with a different camera-IMU configuration).

6.1 Ablations

6.1.1 RGB vs. RGB-D

We perform ablation studies where instead of providing RGB-D images, we provide only RGB imagery and use the monocular depth generation sub-network from SFMLearner to learn to estimate monocular depth (see Section 7.7.1 for more information). These different networks are referred to as VIOLearner RGB-D, and VIOLearner RGB.

6.1.2 Visual vs. Visual-Inertial

Additionally, we perform ablation studies where a version of our network is not provided IMU data to estimate the initial warp estimate and instead uses a vision-only CNN similar to [33] to estimate this initial warp (see Section 7.7.2 for

TABLE 2
Comparisons to VO Approaches on KITTI Sequences 00, 02, 05, 07, and 08

Seq	*VIOLearner (RGB-D)*		*VIOLearner (RGB)*		*VIOLearner (no IMU)*		*UnDeepVO*		*SFMLearner*		*VISO2-M*		*ORB-SLAM2-S*	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
0	2.79	1.09	**1.5**	**0.61**	2.47	1.05	4.14	1.92	65.27	6.23	18.24	2.69	2.96	1.21
2	2.4	0.88	**1.2**	**0.43**	3.33	1.33	5.58	2.44	57.59	4.09	4.37	1.18	4.78	1.61
5	2.3	1.11	**0.97**	**0.51**	2.53	1.28	3.4	1.5	16.76	4.06	19.22	3.54	1.72	0.64
7	2.44	1.54	**0.84**	**0.66**	1.92	1.68	3.15	2.48	17.52	5.38	23.61	4.11	1.61	0.93
8	2.93	1.32	**1.56**	**0.61**	2.78	1.35	4.08	1.79	24.02	3.05	24.18	2.47	1.84	0.77
Mean	2.57	1.19	**1.22**	**0.56**	2.61	1.34	4.07	2.03	36.23	4.56	17.92	2.8	2.58	1.03

Results reproduced from [35]. $t_{rel}(\%)$ is the average translational error percentage on lengths 100 m - 800 m and $r_{rel}(^\circ)$ is the rotational error $(^\circ/100m)$ on lengths 100 m - 800 m. [35] only reported results on KITTI Odometry sequences 00, 02, 05, 07, and 08 so in this table so we only report identical results for VIOLearner.

more information). This version of the network is referred to as VIOLearner (no IMU) and results are only included to provide additional perspective on the vision-only performance of our architecture (and specifically the OEC modules) compared to other vision-only approaches.

6.2 Online Error Correction

The OEC modules were designed to perform error correction at runtime, and thus, be able to correct mistakes stemming from either uncertain/degraded sensory information, or from novel (unseen) sensory information. But as is the case with any deep network, overfitting is always a concern: how do we know that the OEC modules are actually learning to perform a dynamic function and not simply overfitting to the domain? To evaluate the OEC modules, we leveraged AirSim simulations where we could systematically vary the camera-IMU extrinsics. We train networks on data with a given camera-IMU extrinsic calibration and then manipulated the extrinsic calibration for data on which the network was tested (as described in Section. 5.4.2). By varying these extrinsics, we are able to generate camera-IMU misalignments which result in increased error in the Level 0 prediction of $\hat{\theta}_0$ (the first estimate of pose change generated by the network that comes entirely from IMU data).

6.3 Evaluation Metrics

We evaluate our trajectories primarily using the standard KITTI relative error metric (reproduced below from [1])

$$E_{rot}(F) = \frac{1}{F} \sum_{(i,j)\in F} \angle[(\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)] \tag{18}$$

$$E_{trans}(F) = \frac{1}{F} \sum_{(i,j)\in F} \|(\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)\|_2, \tag{19}$$

where F is a set of frames (i, j), $\hat{p} \in SE(3)$ and $p \in SE(3)$ are estimated and true camera poses, respectively, $\ominus$ is the inverse compositional operator, and $\angle[\bullet]$ is the rotation angle.

For KITTI, we use lengths of 100, 200, 300, 400, 500, 600, 700, and 800 m. For AirSim, which has far shorter trajectories, we evaluate on lengths of 25, 50, 75, and 100 m.

Additionally, we compute the root mean squared error (RMSE) for trajectory estimates on three or five frame snippets as has been done recently in [33] and [34].

7 Results and Discussion

7.1 Visual Odometry

7.1.1 KITTI

VIOLearner RGB outperforms the VO approaches listed above as seen in Table 2. It should be noted that the results in Table 2 for VIOLearner, UnDeepVO, and SFMLearner are for networks that were tested on data on which they were also trained which is in accordance with the results presented in [35]. We were thus unable to properly evaluate UnDeepVO against VIOLearner RGB or VIOLearner RGB-D on test data that was not also used for training as such results were not provided for UnDeepVO nor is their model available online at the time of writing. Regardless, VIOLearner RGB-D outperforms UnDeepVO with superior translation and rotation error for all sequences. Surprisingly, VIOLearner RGB outperforms VIOLearner RGB-D on the sequences in Table 2. This may be due to known motion artefacts that contaminate 3D LIDAR data which was used to generate the depth channel of inputs for VIOLearner RGB-D which does not affect depth learned by VIOLearner RGB.

We did however also evaluate VIOLearner more conventionally by training on sequences $00-08$ and testing on sequences 09 and 10 as was the case for [30], [33]. These results are shown in Table 4 (see Fig. 4 for VIOLearner RGB-D, VIOLearner RGB, SFMLearner, ORB-SLAM2, and OKVIS trajectories on KITTI sequences 09 and 10). VIOLearner RGB-D significantly outperforms SFMLearner on both KITTI sequences 09 and 10. Similarly, VIOLearner RGB-D outperforms the deep VO approach of Zhan et al. [30] and the traditional VISO2 on both sequences. As expected, VIOLearner RGB-D outperforms VIOLearner RGB.

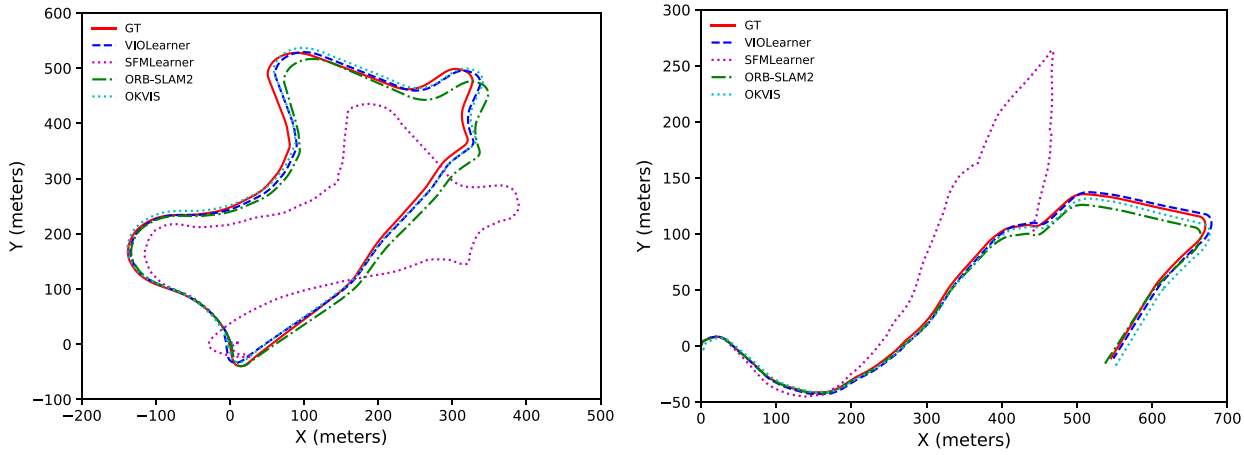
7.1.2 AirSim

Results on the AirSim sequences are shown in Table 5 and plots of the AirSim trajectories are shown in Fig. 5. VIOLearner RGB-D outperforms SFMLearner, the only VO approach with which we compare on AirSim, on all sequences (see Sections 7.2 and 7.3 below for a discussion of VIO and VSLAM results, respectively).

7.2 Visual Inertial Odometry

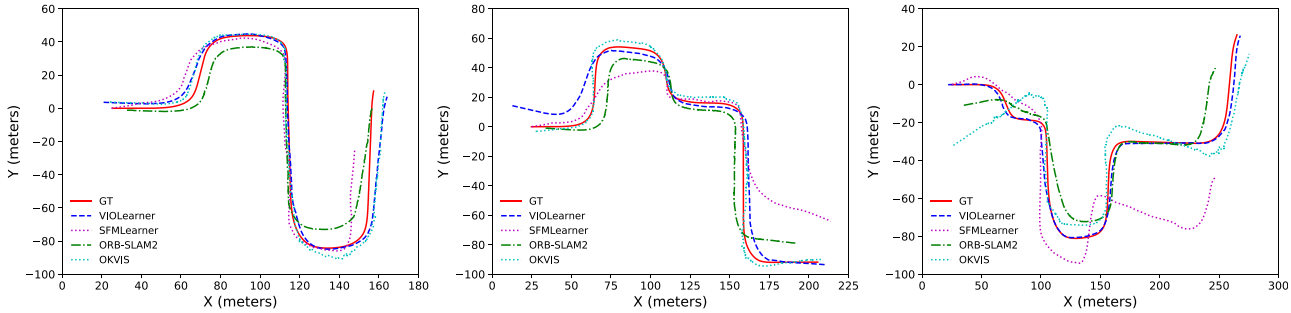
7.2.1 KITTI

The authors of VINet [2] provide box plots of their method's error compared to several SoA approaches for 100 m -



(a) KITTI Seq. 09 (b) KITTI Seq. 10

Fig. 4. Trajectories for KITTI sequences 09 and 10. Ground truth (GT) and results from VIOLearner, SFMLearner, ORB-SLAM2, and OKVIS are shown.



(a) AirSim Seq. 1 (b) AirSim Seq. 2 (c) AirSim Seq. 3

Fig. 5. Trajectories for AirSim sequences 0, 1, and 2. Ground truth (GT) and results from VIOLearner, SFMLearner, ORB-SLAM2, and OKVIS are shown.

TABLE 3
Comparisons to VIO Approaches on KITTI Odometry Sequence 10

Length	*VIOLearner (RGB-D)*			*VIOLearner (RGB)*			*VIOLearner (no IMU)*			*VINet*			*EKF+VISO2*		
	Med.	1st Quar.	3rd Quar.	Med.	1st Quar.	3rd Quar.	Med.	1st Quar.	3rd Quar.	Med.	1st Quar.	3rd Quar.	Med.	1st Quar.	3rd Quar.
100	1.75	0.85	2.37	1.42	1.01	2.01	2.4	1.52	2.92	$\approx$ **0**	$\approx$ 0	$\approx$ 2.18	$\approx$ 2.7	$\approx$ 0.54	$\approx$ 9.2
200	3.1	2.23	4.28	3.37	2.27	5.71	4.48	3.09	6.05	$\approx$ **2.5**	$\approx$ 1.01	$\approx$ 5.43	$\approx$ 11.9	$\approx$ 4.89	$\approx$ 32.6
300	**4.92**	2.96	6.74	5.7	3.24	8.31	7.1	5.35	9.63	$\approx$ 6.0	$\approx$ 3.26	$\approx$ 17.9	$\approx$ 26.6	$\approx$ 9.23	$\approx$ 58.1
400	**6.78**	4.91	8.9	8.83	5.99	10.86	11.52	9.28	12.78	$\approx$ 10.3	$\approx$ 5.43	$\approx$ 39.6	$\approx$ 40.7	$\approx$ 13.0	$\approx$ 83.6
500	**7.49**	6.58	9.82	10.34	6.67	12.92	15.44	13.81	17.09	$\approx$ 16.8	$\approx$ 8.6	$\approx$ 70.1	$\approx$ 57.0	$\approx$ 19.5	$\approx$ 98.9

Results reproduced from box plots in [2] and medians, first quartiles, and third quartiles were estimated from figures. [2] only reported errors on distances of 100, 200, 300, 400, and 500 m from KITTI Odometry sequence 10 so we only report identical results for VIOLearner in this table. Full results for VIOLearner on sequence 10 can be found in Table 4.

500 m on sequences in the KITTI Odometry dataset. We have extracted the median, first quartile, and third quartile from their results plots to the best of our ability and included them in Table 3. For longer trajectories (300, 400, and 500 m), VIOLearner RGB-D outperforms VINet on KITTI sequence 10. It should also again be noted that while VINet requires camera-IMU extrinsic calibrations, our network is able to implicitly learn this transform from the data itself. The authors of [2] reported OKVIS failing to run on the KITTI Odometry dataset and instead used a custom EKF with VISO2 as a comparison to traditional SoA VIO approaches. VIOLearner RGB-D outperforms their EKF with VISO2.

We were able to successfully run KITTI Odometry on sequences 09 and 10 and have included the results in Table 4. VIOLearner RGB-D outperforms OKVIS on KITTI sequences 09 and 10. OKVIS requires tight synchronization between IMU measurements and images which KITTI does not provide, which is most likely the reason for its much lower performance on KITTI despite its use of saved keyframes and a local map. This also highlights a strength of VIOLearner RGB-D in that it is able to compensate for

TABLE 4
Comparisons to VO and VIO Approaches on KITTI Sequences 09 and 10

	VIOLearner (RGB-D)		*VIOLearner (RGB)*		*VIOLearner (no IMU)*		*SFMLearner*		*Zhan et. al*		*OKVIS-S*		*ORB-SLAM2-S*	
Seq														
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
9	**1.82**	1.08	2.27	1.52	2.71	1.4	21.63	3.57	11.92	3.6	6.35	2.43	2.21	**0.96**
10	**1.74**	1.38	2.74	1.35	2.93	1.88	20.54	10.93	12.62	3.43	5.86	2.44	1.9	**1.1**
Mean	**1.78**	1.23	2.53	1.31	2.82	1.64	21.09	7.25	12.27	3.52	6.11	2.44	2.06	**1.03**

$t_{rel}(\%)$ is the average translational error percentage on lengths 100 m - 800 m and $r_{rel}(^\circ)$ is the rotational error $(^\circ/100\ \mathrm{m})$ on lengths 100 m - 800 m calculated using the standard KITTI benchmark [1]. All results are on imagery of size 128 x 480.

TABLE 5
AirSim Trajectory Estimation from VIOLearner RGB-D, VIOLearner RGB, SFMLearner, OKVIS, and ORB-SLAM2

	VIOLearner (RGB-D)		VIOLearner (RGB)		VIOLearner (no IMU)		SFMLearner		OKVIS-S		ORB-SLAM2-S	
Seq												
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
1	**3.19**	6.61	7.1	18.0	4.89	11.12	21.97	17.61	4.87	5.9	12.4	**1.93**
2	**6.6**	13.45	10.13	23.0	6.84	12.01	20.39	14.03	7.08	6.89	13.25	**2.3**
3	**5.04**	10.67	9.52	19.0	8.01	14.38	17.6	19.06	10.59	14.31	14.64	**5.5**
Mean	**4.94**	10.24	8.92	20.0	6.58	12.5	19.99	16.9	7.51	9.03	13.43	**3.24**

VIOLearner RGB-D uses RGB-D imagery and produces scaled trajectories. VIOLearner RGB is augmented with SFMLearner's monocular depth estimation pipeline learns to generate depth estimates from monocular imagery and produces unscaled trajectory estimates. Results for VIOLearner RGB were scaled using the same method of [33] to compare to ground truth.

loosely temporally synchronized sensors without explicitly estimating their temporal offsets.

7.2.2 AirSim

On AirSim sequences, which have perfect temporal synchronization, VIOLearner RGB-D outperforms OKVIS Stereo on all sequence in translation error. However, OKVIS outperforms VIOLearner RGB-D in mean rotation error and rotation error on two of the three individual sequences.

7.3 Visual Simultaneous Localization and Mapping

Additionally, we have included benchmark results from ORB-SLAM2 Stereo. ORB-SLAM2 performs SLAM, unlike our pure odometry-based solution, and is included as a example of SoA localization to provide additional perspective on our results.

7.3.1 KITTI

VIOLearner RGB-D outperforms ORB-SLAM2 Stereo on mean translation error and translation error for KITTI sequences 09 and 10. This was surprising as ORB-SLAM2 uses bundle adjustment, loop closure, maintains a SLAM map, and generally uses far more data for each pose estimate compared to our VIOLearner architecture. However, ORB-SLAM2 Stereo outperforms VIOLearner RGB-D in mean rotation error and rotation errors for both KITTI sequences 09 and 10.

7.3.2 AirSim

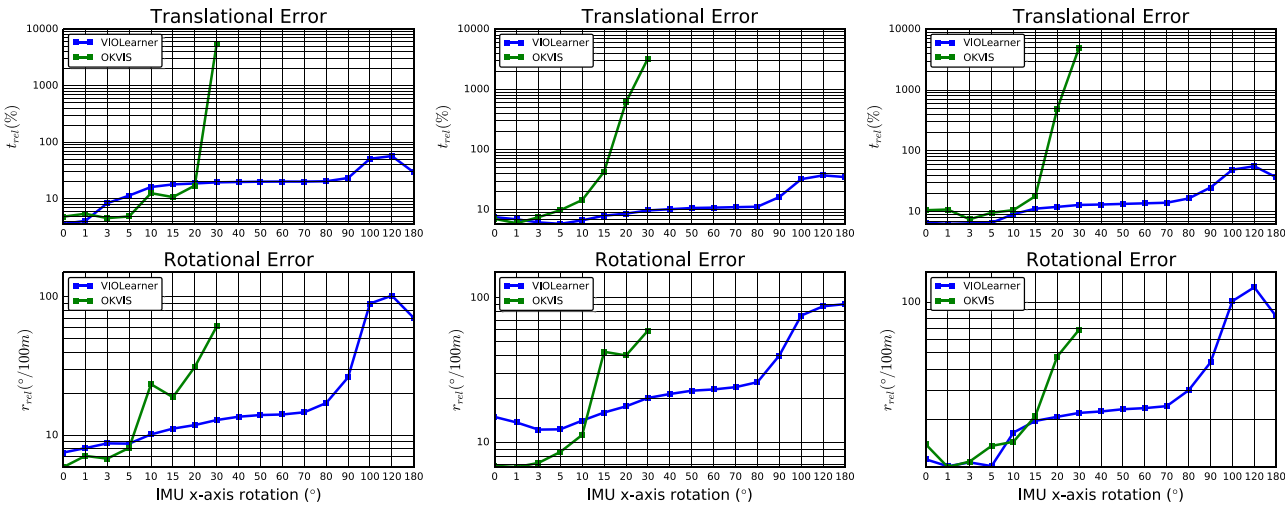
Similarly for the AirSim sequences, VIOLearner RGB-D and VIOLearner RGB outperforms ORB-SLAM2 Stereo on mean translation error as well as translation error on each individual AirSim sequence. Again, however, ORB-SLAM2 produced a lower mean rotational error and lower rotational error on each individual AirSim sequence.

7.4 Multi-Hypothesis Error Reduction

For the $\hat{\theta}$s generated by each of the four hypothesis pathways in the final level for KITTI sequence 09, the average variance of pose error between the four hypotheses was $2.7\mathrm{e}{-5}$ in the x-dimension, $3\mathrm{e}{-6}$ in the y-dimension, $4.69\mathrm{e}{-4}$ in the z-dimension, and $4.06\mathrm{e}{-4}$ was the euclidean error between the computed translation and the true translation. The z-dimension shows 1-2 orders of magnitude more variance compared to the x- and y- dimensions and is the main contributor to hypothesis variance. For the camera frame in KITTI, the z-direction corresponds to the forward direction which is the predominant direction of motion in KITTI and also where we would expect to see the largest influence from uncertainty in initial velocities. These results are consistent with the network learning to model this uncertainty in initial velocity as intended.

7.5 Online Error Correction

Results in Fig. 7 show the pose root mean squared error (RMSE) between the actual θ and the $\hat{\theta}$ generated at each level and suggest that our online error correction mechanism is able to reduce pose error. It should however be noted that $\hat{\theta}$ for Levels 0 to 2 are computed using downsampled images and thus their Jacobian inputs both have access to less information and use one less convolutional layer each. The extent to which this affects the plots in Fig. 7 is as of yet not fully clear. However, Level 3 operates on full-size inputs and there



(a) AirSim Seq. 1 (b) AirSim Seq. 2 (c) AirSim Seq. 3

Fig. 6. Results on VIOLearner and OKVIS trajectory estimation on the *Blocks* environment given induced IMU orientation offset in the x-dimension. Measurement errors are shown for each sequence with translational error percentage (top row) on lengths 25 m - 100 m and rotational error in degrees per 100 m (bottom row) on lengths 25 m - 100 m. In contrast to VIOLearner, after 20-30° (depending on the sequence), OKVIS exhibits catastropic failure in translation estimation.

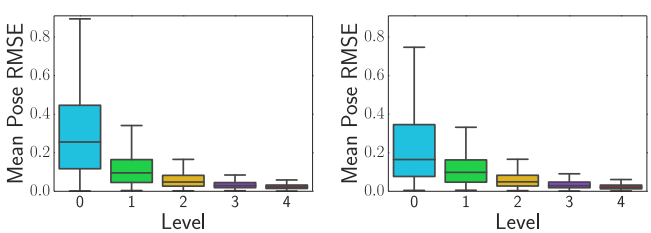
is still a reduction in error between Level 3 and Level 4. While this reduction in the final layer can be partially attributed to the multi-hypothesis mechanism in Level 4, the mean RMSE for Level 3 is 0.042 while the mean RMSE from each individual hypothesis pathway is 0.0375, 0.0555, 0.0262, 0.0303, 0.0332 (0.0261 when the lowest error hypothesis is chosen) for KITTI 09 and 0.029, 0.058, 0.034, and 0.037 (0.027 when the lowest error hypothesis is chosen) for KITTI 10. The lower mean RMSE from individual Level 4 hypotheses (with the exception of the second hypothesis pathway) suggests that the observed error reduction is indeed an effect of our online error correction mechanism rather than simply an artefact of image resolution.

7.5.1 Spatial Misalignments

The AirSim simulated spatial misalignments between different camera and IMU configurations (as described in Section 5.4.2) provide a method for evaluating the effectiveness of VIOLearner's OEC modules.

As seen in Fig. 6, orientation offsets within a realistic range of less than 10 degrees shows low amounts of error and great applicability to real world implementation. Further, offsets within a range of less than 80 degrees display a modestly sloped plateau that suggests successful online error correction. Error measures appear to drastically increase around the 90 degree mark, and this is plausibly expected since rotations of this magnitude result in complete dimension shift, and, unsurprisingly, the network appears unable to compensate.

Contrastingly, OKVIS showed surprising robustness to rotation errors under 20 degrees but is unable to handle orientation offsets greater than 30 degrees.



(a) KITTI 09 (b) KITTI 10

Fig. 7. Pose root mean squared error (RMSE) from poses computed by VIOLearner compared to ground truth after each level for KITTI sequences 09 and 10. We see a substantial decrease in error after repeated applications of our error correction module.

7.6 Generalizability

To evaluate the ability of the VIOLearner architecture and its OEC modules to generalize, we performed additional experiments with regard to generalizability to motion and image perturbations/domain shifts.

7.6.1 Motion

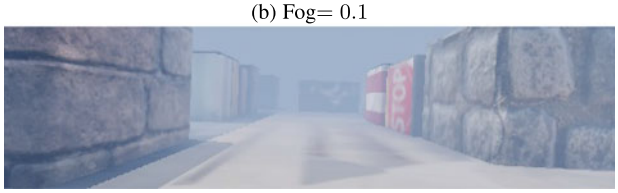
While the AirSim datasets exhibited greater motion variability than the KITTI datasets (in particular rotational variability), the training trajectories and evaluation trajectories presented thus far were predominantly forward moving with slow, car-like turns (see Fig. 5). To evaluate the ability of VIOLearner to generate to previously unseen trajectories very different from those that the network was trained on, we collected two additional evaluation trajectories. In the first trajectory, referred to as *figure-eight*, the simulated quadcopter flew in a figure-eight pattern in AirSim in the same Blocks environment. In the second trajectory, referred to as *rotation-only*, the simulated quadcopter yawed in-place without translating. Results for these two trajectories are shown in Table 9. VIOLearner still outperformed OKVIS-S and ORB-SLAM2-S on the *figure-eight* trajectory in translation error but was outperformed by OKVIS-S on the *rotation-only* trajectory.

7.6.2 Imagery

As mentioned earlier, VIOLearner implicitly learns the extrinsic parameters of the camera-IMU system as well as IMU intrinsics. As such, we were unable to evaluate a

(a) No Fog

(b) Fog= 0.1



(c) Fog= 0.3

Fig. 8. Sample images from the three fog conditions.

VIOLearner network trained in one domain (e.g., KITTI) on a different domain (e.g., AirSim). To evaluate the ability of VIOLearner to generalize to previously unseen imagery and handle domain shifts, we performed additional experiments where we re-collected the three AirSim evaluation trajectories from Table 5 using identical motion profiles but varying the weather-conditions (see Fig. 8). Results for these two trajectories are shown in Table 8. In the lightest fog condition, VIOLearner outperformed OKVIS-S and ORB-SLAM-S in translation error. For the medium fog condition, VIOLearner was outperformed on two of the three trajectories. For the highest fog condition, ORB-SLAM2-S failed to run on any of the trajectories while VIOLearner outperformed OKVIS-S in translation error.

7.7 Ablation Studies

7.7.1 Depth Ablation

As mentioned earlier, VIOLearner does not precisely compare to many of the other deep approaches to vision-aided odometry in that it is provided with RGB-D imagery in order to produce trajectories that are correctly scaled. To confirm that our results are not due solely to our inclusion of depth data in the form of a depth image, we performed ablation studies where rather than providing RGB-D imagery, we used RGB imagery and implemented the monocular depth generation architecture of [33] to train VIOLearner to not only learn odometry, but also to learn unscaled depth.

TABLE 6
Absolute Trajectory Error (ATE) on KITTI 09 and KITTI 10 Averaged over Three or Five Multi-Frame Snippets (Reproduced from [34])

Method	Seq. 09	Seq. 10	Mean
ORB-SLAM (full)	0.014	0.012	0.013
ORB-SLAM (short)	0.064	0.064	0.063
Mean Odom.	0.032	0.028	0.030
SFMLearner (5-Frame)	0.021	0.032	0.027
Mahjourian et al. with ICP (3-Frame)	0.013	0.012	0.0125
VIOLearner RGB (5-Frame)	**0.012**	**0.012**	**0.012**

Note that while SFMLearner [33] and Mahjourian et al. [34] use three or five frame batches of data as training/testing input, VIOLearner is provided with far less visual data as it only uses a single source and target image pair and IMU data between the pair.

Tables 4 and 5 show results from VIOLearner RGB-D (which was provided RGB-D data and IMU data), and VIOLearner RGB (which was instead only provided monocular RGB imagery and IMU data) on KITTI and AirSim, respectively. For KITTI, VIOLearner RGB outperforms approaches SFMLearner [33] and the network of Zhan et al. [30].

In addition to evaluating with relative error over the entire trajectory, we also evaluated VIOLearner RGB using RMSE over five-frame snippets as was done in [33] and [34] for their similar monocular approaches. As shown in Table 6, on KITTI trajectories 09 and 10, VIOLearner RGB surpasses RMSE performance of both SFMLearner and Mahjourian et al.

7.7.2 IMU Ablation

Similar to the previous section, we performed additional ablation studies where rather than using IMU data to generate the initial warp estimate, we use a vision-only CNN similar to the pose network from SFMLearner [33] (this network is referred to as VIOLearner (no IMU)). Results for this network configuration are shown in Tables 2, 3, and 4 for KITTI and Table 5 for AirSim. Compared to other vision-only approaches, for KITTI, VIOLearner (no IMU) outperforms approaches SFMLearner [33] and the network of Zhan et al. [30] and yet is outperformed by ORB-SLAM-2 which uses full bundle adjustment. For AirSim, VIOLearner (no IMU) outperforms approaches SFMLearner [33] and ORB-SLAM2-S in translation error.

7.7.3 Hypothesis Ablations

To validate and evaluate the benefits of our multi-hypothesis approach, we performed experiments on KITTI and AirSim using between $1-8$ hypothesis pathways.

TABLE 7
Results from Networks Trained with 1-8 Hypothesis Pathways on KITTI and AirSim

#	*KITTI*						*AirSim*							
	09		*10*		*Mean*		*1*		*2*		*3*		*Mean*	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
1	2.32	1.15	2.99	1.5	2.66	1.33	3.81	7.5	7.08	14.01	5.63	11.22	5.51	10.91
2	1.63	**0.74**	2.37	**1.15**	2.0	**0.95**	3.85	7.61	6.09	11.71	5.29	9.99	5.08	9.77
4	1.82	1.08	**1.74**	1.38	**1.78**	1.23	**3.19**	**6.61**	6.6	13.45	5.04	10.67	5.82	10.24
8	**1.35**	0.74	2.39	1.24	1.87	0.99	3.33	6.68	**5.74**	**11.26**	**4.53**	**8.3**	**4.53**	**8.75**

TABLE 8
Robustness Evaluation of VIOLearner to Shifts in the Visual Domain

Seq	VIOLearner (RGB-D)		OKVIS-S		ORB-SLAM2-S	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
1 (Light Fog)	**6.45**	11.11	7.1	7.69	12.14	**2.03**
2 (Light Fog)	**11.07**	21.45	11.38	11.35	13.47	**1.44**
3 (Light Fog)	**9.38**	19.04	16.2	18.6	15.03	**6.59**
1 (Med. Fog)	8.86	16.42	**6.78**	**5.79**	X	X
2 (Med. Fog)	13.82	26.9	18.78	26.95	**13.43**	**2.8**
3 (Med. Fog)	**12.2**	23.65	14.17	15.24	13.02	**4.77**
1 (Heavy Fog)	**15.65**	28.82	15.7	**16.91**	X	X
2 (Heavy Fog)	**20.01**	**39.96**	21.16	47.34	X	X
3 (Heavy Fog)	**18.11**	33.65	33.95	**21.22**	X	X

Trajectory estimation from VIOLearner RGB-D, OKVIS, and ORB-SLAM2 on AirSim sequences 1, 2, and 3 with light fog ($foglevel = 0.05$), medium fog ($foglevel = 0.1$), and heavy fog ($foglevel = 0.3$). An X indicates that the algorithm failed to successfully complete the sequence.

As shown in Table 7, for both KITTI and AirSim, VIOLearner shows increased performance in two hypothesis configurations compared to one hypothesis configurations. Similarly, four or eight hypothesis configurations outperform two hypothesis configurations. With the exception of AirSim sequence 2, all four hypothesis configurations were superior to two hypothesis configurations in translation error. However, eight hypothesis networks did not always outperform the four hypothesis networks. This suggests a possible diminishing return on adding additional hypotheses that eventually introduces a performance decrease (see Section 8.1 for further discussion).

7.8 Runtimes

VIOLearner runs in real-time and is faster than ORB-SLAM2-S and OKVIS-S. VIOLearner has a mean runtime of 27 ms on AirSim evaluation sequences 1, 2, 3 and KITTI 09 and 10 (as to be expected of a deep approach of a fixed size, there was no statistically significant difference between runtimes on the various datasets for VIOLearner). ORB-SLAM2-S had a mean runtime of 67 ms per image on AirSim evaluation trajectories 1, 2, and 3 and OKVIS-S has a mean runtime of 122 ms. On KITTI, ORB-SLAM2-S had a mean runtime of 89 ms on sequences 09 and 10 while OKVIS-S had a mean runtime of 143 ms.[1]

8 Conclusion and Future Work

Despite using only single source-target image pairs, VIOLearner outperforms recent deep VO approaches in translation error as well as traditional VIO and VSLAM approaches that use sequences of images, key-frame based bundle adjustment, and full bundle adjustment and loop closure, respectively. This is enabled by novel OEC modules embedded in our end-to-end trainable deep visual-inertial architecture which allow the network to correct intermediate output errors.

1. Both KITTI and AirSim imagery were captured at 10 Hz so real-time performance in this case is 100 ms per image or less.

TABLE 9
Robustness Evaluation of VIOLearner to Novel Trajectories/ Motions Not Seen During Training

Seq	VIOLearner (RGB-D)		OKVIS-S		ORB-SLAM2-S	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
Figure-Eight	**4.7**	11.88	5.2	5.14	13.38	**1.84**
Rotation-Only	9.08	21.57	**5.18**	4.51	13.5	**2.66**

Trajectory estimation from VIOLearner RGB-D, OKVIS, and ORB-SLAM2 on a figure-eight trajectory and a rotation-only trajectory in AirSim.

In this work, we have presented our VIOLearner architecture and demonstrated superior performance against SoA VO, VIO, and even VSLAM approaches that use far more data. VIOLearner's novel multi-step trajectory estimation via convolutional processing of Jacobians at multiple spatial scales yields a trajectory estimator that learns how to correct errors online. We show how including RGB-D data produces higher accuracy trajectory estimates with absolute scale. Even when RGB-D data is not provided, VIOLearner with RGB data outperforms similar deep monocular approaches.

The main contributions of VIOLearner are its unsupervised learning of scaled trajectory, online error correction based on the use of intermediate gradients, and ability to combine uncalibrated, loosely temporally synchronized, multimodal data from different reference frames into improved estimates of odometry.

8.1 Multi-Hypothesis Outputs

In AirSim, the eight hypothesis networks generally outperformed the four hypothesis networks (as seen in Table 7) but on KITTI, the opposite was true. Previous results from [48] suggest a positive correlation between noise in the input data and benefits of additional hypotheses which may also translate to variance in the input data. As the MAV AirSim trajectories typically exhibited more varied motion compared to KITTI trajectories, this may explain why the eight-hypothesis networks seem to have been better utilized on AirSim compared to KITTI. Future work will be needed to address and clarify the efficacy of additional hypotheses.

8.2 Rotational Error

The disparity between translational performance and rotational performance may be due to VIOLearner's Euler rotation formulation (as opposed to a quaternion formulation), or may be due to VIOLearner only receiving a single source and target image and being unable to perform any type of bundle adjustment. This question will need to be investigated in future work, both by extending VIOLearner to use a non-Euler orientation formulation and to perform some type of bundle adjustment (as is done for traditional approaches such as [3], [5], [20]) or surrogate bundle adjustment (as is done for other deep approaches [2], [30], [34], [52]).

Acknowledgments

Kyle Lindgren and Sarah Leung contributed equally to this work.

REFERENCES

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[2] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "VINet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 3995–4001.

[3] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, 2015.

[4] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 965–972, Apr. 2018.

[5] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *The Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1053–1072, 2017.

[6] E. J. Shamwell, S. Leung, and W. D. Nothwang, "Vision-aided absolute trajectory estimation using an unsupervised deep network with online error correction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 2524–2531.

[7] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Proc. Field Serv. Robot.*, 2018, pp. 621–635.

[8] M. Srinivasan, M. Lehrer, W. Kirchner, and S. Zhang, "Range perception through apparent image speed in freely flying honeybees," *Visual Neurosci.*, vol. 6, no. 5, pp. 519–535, 1991.

[9] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part II: Matching, robustness, optimization, and applications," *IEEE Robot. Autom. Mag.*, vol. 19, no. 2, pp. 78–90, Jun. 2012.

[10] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars Exploration Rovers," *J. Field Robot.*, vol. 24, no. 3, pp. 169–186, 2007.

[11] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Stanford Univ., Stanford, CA, Tech. Rep. CMU-RI-TR3, 1980.

[12] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2004, vol. 1, p. I.

[13] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 15–22.

[14] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, 2007, pp. 225–234.

[15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[16] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2320–2327.

[17] S. Song, M. Chandraker, and C. C. Guest, "High accuracy monocular SFM and scale correction for autonomous driving," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 730–743, Apr. 2016.

[18] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments," *J. Field Robot.*, vol. 28, no. 6, pp. 854–874, 2011.

[19] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 4973–4980.

[20] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[21] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 3565–3572.

[22] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," *Memory*, vol. 10, 2018, Art. no. 20.

[23] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The Int. J. Robot. Res.*, vol. 32, no. 6, pp. 690–711, 2013.

[24] M. N. Galfond, "Visual-inertial odometry with depth sensing using a multi-state constraint Kalman filter," Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 2014.

[25] F. Pang, Z. Chen, L. Pu, and T. Wang, "Depth enhanced visual-inertial odometry based on multi-state constraint Kalman filter," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 1761–1767.

[26] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense RGB-D-inertial SLAM with map deformations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 6741–6748.

[27] J. Zhang and S. Singh, "Enabling aggressive motion estimation at low-drift and accurate mapping in real-time," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5051–5058.

[28] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 740–756.

[29] C. Godard, O. Mac Aodha , and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, vol. 2, Art. no. 7.

[30] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 340–349.

[31] N. Smolyanskiy, A. Kamenev, and S. Birchfield, "On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 1007–1015.

[32] S. Pillai and J. J. Leonard, "Towards visual ego-motion learning in robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5533–5540.

[33] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, vol. 2, Art. no. 7.

[34] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5667–5675.

[35] R. Li, S. Wang, Z. Long, and D. Gu, "UnDeepVO: Monocular visual odometry through unsupervised deep learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7286–7291.

[36] R. Clark, M. Bloesch, J. Czarnowski, S. Leutenegger, and A. J. Davison, "Learning to solve nonlinear least squares for monocular stereo," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 284–299.

[37] E. J. Shamwell, S. Leung, and W. D. Nothwang, "Vision-aided absolute trajectory estimation using an unsupervised deep network with online error correction," *arXiv: 1803.05850*, 2018.

[38] H. Zhou, B. Ummenhofer, and T. Brox, "DeepTam: Deep tracking and mapping," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 851–868.

[39] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, and J. González-Jiménez, "The Málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario," *The Int. J. Robot. Res.*, vol. 33, no. 2, pp. 207–214, 2014.

[40] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan north campus long-term vision and lidar dataset," *The Int. J. Robot. Res.*, vol. 35, no. 9, pp. 1023–1035, 2016.

[41] A. L. Majdik, C. Till, and D. Scaramuzza, "The Zurich urban micro aerial vehicle dataset," *The Int. J. Robot. Res.*, vol. 36, no. 3, pp. 269–273, 2017.

[42] B. Pfrommer, N. Sanket, K. Daniilidis, and J. Cleveland, "PennCOSYVIO: A challenging visual inertial odometry benchmark," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3847–3854.

[43] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.

[44] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The TUM VI benchmark for evaluating visual-inertial odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1680–1687.

[45] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field Service Robot.*, Springer, pp. 621–635, 2018.

[46] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.

[47] P. Anandan, J. Bergen, K. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," in *Motion Analysis and Image Sequence Processing*. Berlin, Germany: Springer, 1993, pp. 1–22.

[48] E. J. Shamwell, W. D. Nothwang, and D. Perlis, "An embodied multi-sensor fusion approach to visual motion estimation using unsupervised deep networks," *Sensors (Basel, Switzerland)*, vol. 18, no. 5, 2018, Art. no. E1427.
[49] R. Resnick and D. Halliday, *Physics*. Hoboken, NJ, USA: Wiley, 1966, no. pt. 1. [Online]. Available: https://books.google.com/books?id=JncgAQAAIAAJ
[50] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *Proc. IEEE Intell. Veh. Symp.*, 2011, pp. 963–968.
[51] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, Apr. 1991.
[52] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," in *Proc. Eur. Conf. Comput. Vis.*, 2016, vol. 1, pp. 286–301.

E. Jared Shamwell received the BA degree in economics and philosophy from Columbia University, New York, in 2009, and the PhD degree in neuroscience from the University of Maryland, College Park, in 2017, where he focused on neuro-inspired approaches to autonomous navigation and machine learning. He is currently a research scientist with GTS stationed at the US Army Research Laboratory where he primarily works on machine learning for autonomy and position, navigation, and timing (PNT). He is a member of the IEEE.

Kyle Lindgren received the BS and MS degrees in electrical engineering from the University of Washington, Seattle, in 2015 and 2017, respectively. He is working toward the PhD degree at the University of Washington. With a concentration in embedded systems as an undergraduate, he gained further embedded systems experience in the aerospace industry through multiple internships with Honeywell Aerospace in Redmond, WA. His research interests are in robotics, computer vision, and machine learning. His Master's thesis was focused on adding motion compensation to the control of surgical robots for beating heart surgery utilizing stereo endoscopy and computer vision algorithms to create and track deformable models of the surface. He is a Journeyman fellow with ORAU stationed at the US Army Research Laboratory. He is a student member of the IEEE.

Sarah Leung received the MS degree in robotics from the University of Pennsylvania, Philadelphia, in 2014, with a background in mechanical and aerospace engineering from Cornell University, Ithaca, NY. She was previously an engineer at Boeing, working on CH-47 structural design in Philadelphia, PA, and on robotic systems for automated fiber placement in Ladson, SC. She is currently a research scientist with GTS stationed at the US Army Research Laboratory, Adelphi, MD, focusing on autonomous navigation and sensor fusion.

William D. Nothwang is currently the team leader for the electronics for Sense and Control Team within the Sensors and Electron Devices Directorate at the Army Research Laboratory, where they conduct research into distributed state estimation for the dismounted soldier and micro air vehicles; specifically applied to Position, Navigation, and Timing and human physiological state monitoring. He has participated in the National Academy of Engineering's Frontiers of Engineering and the NAE's Japan-America FOE. He has been a co-author on two Army R&D Awards; 3 ARL R&D Awards; 6 Directors Research Initiatives, and 6 Directors Strategic Initiatives. In 2012, he contributed to the team that won the Directors Choice Award for Innovation in Research, and his lab has been awarded one of the "Coolest Labs" by Popular Science.