Predicting Visual Pose Estimation Failure on a Vehicle for Off-Road Driving

Ran Cheng[1], Travis Manderson[1], Ioannis Rekleitis[2] and Gregory Dudek[1]

***Abstract*— We develop and evaluate an algorithm to predict how alternative future paths will impact visual odometry with the ultimate goal of selectively tuning navigation behaviours to improve state estimation performance. The key to our approach is learning the relationship between distant regions visible in the image and actual terrains we eventually encounter when we arrive at those places. Our approach involves matching current experience and visual odometry performance with previous views of the same regions when they were still distant. We use the Direct Sparse Odometry algorithm for short-term visual odometry and a deep convolutional neural network based on ResNeXT to encode the relationship between current estimation error and past observations.**

We evaluate our approach on training, test, and validation data collected using a $\frac{1}{5}$ scale self-driving car developed for these experiments. Our ability to predict which regions would cause odometry to fail is roughly 80 *per cent*. The vehicle uses RTK-GPS for ground truth and an on-board GPU (Jetson TX2) for real-time computing.

I. Introduction

We consider the use of vision-based odometry [1] and localization for vehicle navigation in natural or unstructured environments where the availability of complete maps cannot be presupposed. In particular, we focus on visual odometry in the context of off-road driving. We focus on the detection of locations where visual odometry will likely perform poorly so that, where possible, we can adjust the behaviour of our vehicle to avoid these regions.

Robust navigation in unknown and unstructured environments is a difficult task due to the varying conditions that a vehicle can encounter and must account for. These environments present mobility challenges for vehicles due to extreme variations in terrain surface material, geometry and obstacles, and computer vision challenges due to varying textures, illuminations, occlusions, and general scene depth. Specifically, the performance of vision-based localization techniques can be unpredictable when deployed in such environments. In this work, we seek to predict in advance when the performance of Visual Odometry (VO) will degrade so that remedial action can be taken.

Algorithms for VO invariably rely on the scene exhibiting adequate richness of stable texture to estimate the relative transformation between image frames, regardless of whether the technique is gradient or feature based. The choice of features in a given frame to track has been studied for several decades [2], nearly always with emphasis on almost static scenes. Moreover, for many years, the focus was on feature-matching techniques such as the SIFT [3], SURF [4], BRIEF [5], and ORB [6] features that are fast to compute and robust to illumination and orientation changes. Most recently, the SuperPoint [7] is a feature detector and descriptor using a Deep Convolutional Neural Network (DCNN) trained using self-supervision that results in state-of-the-art feature matching. However, an additional issue arising in real-world navigation that is largely ignored in the classic vision literature is the relationship between physical phenomena, including scene morphology and the reliability of observable visual features and textures. For example, otherwise reliable visual textures that might be seen in images of grass or leafy trees can be very unreliable outdoors when there is gusty wind making those leaves or grass move.

Fig. 1. Off-road vehicle developed and used in this work. For computers, contains an NVidia Jetson TX2, Intel i7 NUC and low-level management unit. For sensors, there are redundant IMU's, two forward-facing cameras, lidar, and an RTK GPS (on top of the tall mast).

Vision-based localization has been considered by many authors [8], [9], [10], [11], [12], [13], [14], including variations to improve robustness by tightly integrating Inertial Measurement Unit (IMU) measurements [15], [16]. Despite these impressive advances, there exist situations where unfavorable local environments make it difficult or impossible to visually estimate the translation between consecutive images [17]. While some of these VO algorithms produce a confidence estimate that can be used to instantaneously suggest when they are under-performing, our interest is in developing a *long-range forecast* of eventual problem areas *before* the visual data is sufficiently clear and proximal to actually run the entire visual odometry pipeline. Notably, these problem areas can, in practice, be problematic not only due to visual properties of the scene, but also possibly due to terrain variations that compromise the accuracy of the plan model

[1]Mobile Robotics Laboratory, School of Computer Science, McGill University, Montreal, Canada

[2]Computer Science and Engineering Department, University of South Carolina, Columbia, SC. USA

used as part of the estimator (for example, over sand or gravel). Likewise, a human driver may seek to avoid a glossy area in the distance because it might contain slippery terrain, and a snow skier might avoid a region with blowing snow that impairs visibility.

In this work, we test our results on a scale-model autonomous car driving off-road (see Fig. 1). Although our model vehicle can reach speeds of 80km/h and exhibits full autonomy, by using a small-sized vehicle, we minimize the logistic overhead and risk of deployment while conducting experiments.

II. Related Work

Vision based state estimation has been extensively examined with impressive progress in the last decade [18], [19], [20], [17] and with major improvements in accuracy when inertial data are fused with vision in [21], [22], [23], [24], [25]. Despite that, the local scene structure remains a determinant of performance and failures cannot be precluded when the scene is not conducive to feature identification.

In early work, Jenkin *et al.* performed path planning to optimize the trade-off between trajectory length and a simple model of the observability of engineered visual features [26], [27]. Similarly, Roy and Thrun performed navigation to balance observation-based uncertainty and path length when using LIDAR data [28]. In our own prior work, we have considered modulating the gaze direction of an air vehicle during flight to improve the quality of pose estimation [29] by rotating the vehicle in different directions. In that work, unlike the current one, we employed local binary patterns (LBPs) and developed a classifier to associate LBP values with anticipated SLAM output, but did not make long-term or long-range inferences regarding future outputs. Similar to our previous work, Rodrigues [30] used artificial potential fields that used features in the image frame and the destination goal to guide an aerial vehicle towards this goal while maintaining good visual features for tracking.

Using vision for outdoor navigation dates back several decades. The CMU Navlab was one of the early autonomous vehicles that used adaptive vision-based methods (based on color classification) for navigation (in that case for obstacle avoidance) [31]. Many other notable systems also used monocular or stereo features for motion tracking [32], [33], [34], [35]. Likewise, various authors have considered how to learn the relationship between scene changes, object identity, ego-motion, and appearance, but very rarely - if ever - explicitly consider how eventual failures in the vision pipeline might have been predicted and avoided by past images [36], [37]. One of the few threads of research to consider this task, at least implicitly, was in the context of early work on active vision where goals beyond minimizing a loss function are broadly taken into account [38], [39].

LeCun [40] has considered the challenges of off-road navigation by combining wheel encoders and visual orientation estimation as a low-cost solution to off-road navigation. At the time, this approach was comparable to purely visual odometry solutions. More recently, Gurau has taken an experience-based approach to predicting the performance of perception systems by evaluating appearance similarities between current and past observations [41].

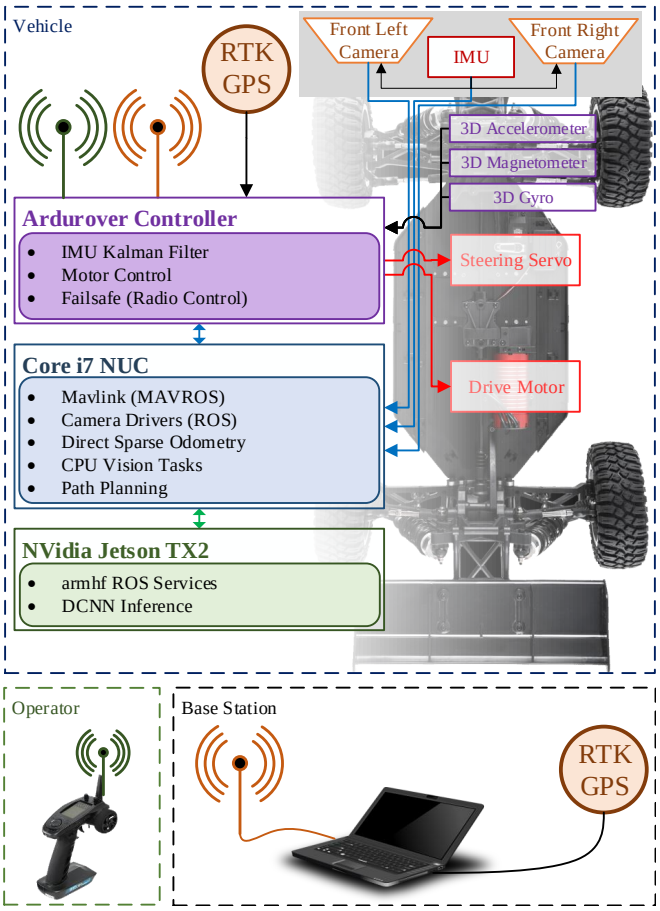

Fig. 2. Robotic platform shown with three main computers, cameras, attitude and position sensors and motor connections. A based station is used to monitor the vehicle remotely as well as send RTCM RTK GPS correction signals to the robot. An operator remotely controls the robot when it's not in fully autonomous mode, or overide the vehicle and in case of emergency.

To our knowledge, this paper provides the first results where medium to long-term predictions regarding the performance of a state-of-the-art visual odometry system are computed as learned conditional functions of long-term navigation actions taken in natural outdoor environments, thus permitting perceptually-optimized behavior.

III. System Overview

Our platform is a one-fifth scale off-road buggy built from a Losi Xl-E RC electric vehicle as the base platform (as outlined in Fig. 2). It contains a PixRacer flight controller (generally intended to run on-board an unmanned aerial vehicle) running the ArduRover firmware. This controller maintains a pose estimate using an extended Kalman filter that fuses sensor information from redundant triple-axis accelerometers, magnetometers, and gyros, as well as an Real-time kinematic (RTK) Global Positioning System (GPS). Note that the GPS is used here only for *ground-truth* during training and evaluation, while, the rover is intended to operate without this costly sensor normally and/or in

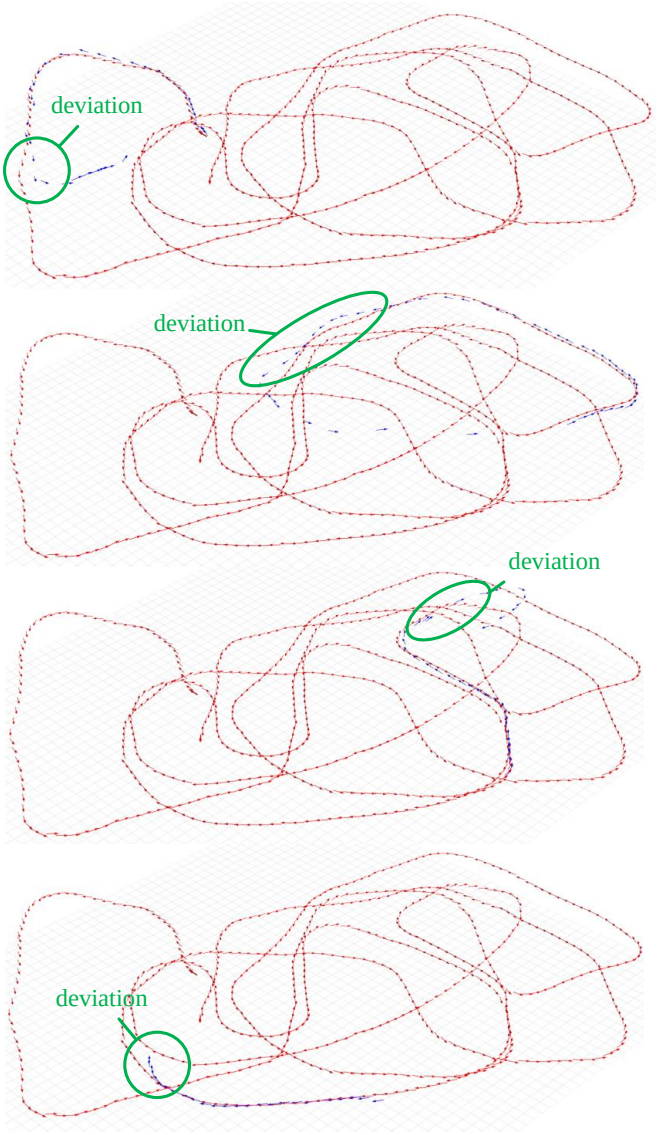

Fig. 3. Example trajectory of the vehicle. The red arrows are the ground-truth poses provided by the RTK GPS while the blue arrows are VO pose estimates computed on segments of the full trajectory. VO was run until the estimate deviated by more than five meters at which point it was stopped and restarted. Examples of these deviations are outlined in green.

GPS-denied environments. Without GPS, the PixRacer still maintains an orientation estimate from the IMU sensors. The PixRacer is also responsible for receiving user commands while operating in manual mode (also used for failsafe in a runaway situation) and for sending steering and throttle signals to the steering servo motor and drive motor respectively.

The PixRacer is connected to an Intel i7 NUC mini PC running Ubuntu and the Robot Operating System (ROS) software package. Pose estimates (from the PixRacer, GPS, and IMU) are continually sent to the mini PC at 10 Hz. When operating in autonomous mode, the mini PC sends steering and throttle commands relayed through the PixRacer to the steering and drive motors. Communication is based on the MAVLINK protocol and MAVROS. The mini PC connects via USB 3.0 to our forward-facing stereo-camera-IMU sensor containing two uEye UI-3251-LE cameras and a VectorNav, VN100 IMU on a rigid frame. Each camera is hardware synchronized to the IMU and uses 4 mm fixed-focal length lenses that are pre-calibrated. In our experiments, we use the mini PC to run VO algorithms and high-level navigation tasks.

An NVidia Jetson TX2 (which also runs Ubuntu and the ROS software package) communicates with the mini PC using gigabit ethernet to receive the robot camera image. This compute module is used to run the DCNNs and sends the results to the mini PC to be used by high-level planning.

During field experiments, a base station is used to primarily send RTK RTCM correction signals to the robot but is also used to monitor the condition of the robot in terms of the GPS position variance, communication link quality, and battery life.

In general, this set-up is small enough be transported by a single user, but allows for long duration run times (approximately five hours with two 4-cell 10Ah LiPo batteries) and large-scale experiments. In this work, most of our driving takes place at very low speeds for safety and permitting reasons, but the vehicle is capable of traveling up to 80 km/h. Based on the scaled-down size of the vehicle (compared to a regular automobile) and the low speeds used, the length of the experimental trials reported below have features akin to much longer trials on a standard-sized vehicle.

IV. Data Collection

Our data was generated by manually driving the vehicle over a diverse set of trajectories in an outdoor environment with varying amounts of grass and dirt and surrounded by trees, bushes, and buildings (as exemplified in Figure 5). In total, we collected approximately six hours and 20 km of driving data in an area of 5,000 m^2. We used an RTK GPS, which provides 30 cm relative accuracy for position *ground-truth* and redundant triple-axis magnetometers, accelerometers, and gyros mounted at the front and back of the vehicle for orientation *ground-truth*.

Along with the *ground-truth* pose of the vehicle, we recorded the steering and throttle commands and the images at 15 Hz. We fixed the exposure to the same value for each of the left and right cameras. Each image is 800 x 600 pixels and has a FOV of 123° x 108°.

A. Post-process visual odometry pose estimation

Following the collection of image sequences, we ran a batch process to compute the pose estimate of our trajectories using stereo Direct Sparse Odometry (DSO). We ran DSO as long as the relative error remained below five meters. The error e at time t is the Euclidean distance between the between the VO position (p_v) and the *ground-truth* position estimate (p_g):

$$e(t) = \sqrt{\|p_g(t) - p_v(t)\|} \quad (1)$$

Since our objective is to collect instances where scene properties induced failure, we needed to collect disparate examples of both effective and failing odometry, but once

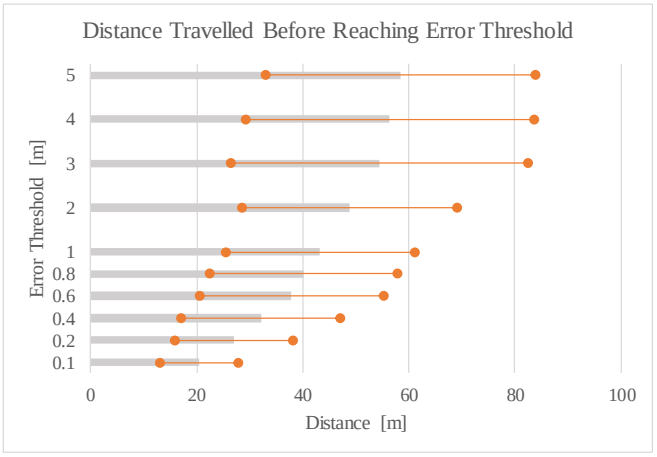

Fig. 4. Average distance the vehicle traveled before the error reached a specific threshold. It is expected that there will always be some error due to the variance in the GPS position and general VO noise, however once the error reaches one meter, it often continued to deviate rapidly and it can be assumed that the tracking would continue to deviate.

the estimator diverged from the *ground truth*, then continued estimation was not useful. Thus, once the error exceeded five meters error, we stopped and restarted DSO, initializing from ground true values at that point in the trajectory. This process resulted in approximately 350 sub-trajectories. An example of this process is depicted in Fig. 3, where four VO estimated sub-trajectories are overlaid on the *ground-truth* trajectory. For each sub-trajectory, it can be seen that the estimated pose deviates after some distance. Figure 4 shows the average distance the vehicle traveled before the error reached a specified threshold. It can be seen that once the error reached one meter, it generally continued to deviate quickly.

Compared to the other feature-based algorithms, we chose stereo DSO as our VO implementation due to computational efficiency. However, the stereo matching approach is based on photometric Gaussian-Newton optimization and produces imperfect scale estimation when it comes to outdoor illumination conditions and exposures. Further, if we were to initialize DSO in the GPS-world frame based on the compass alone, even small errors in the magnetometer would result in drift between the VO pose estimate and the *ground-truth*. Therefore, to initialize the DSO accurately in the GPS-world frame and to overcome poor scale initialization, we calibrate the orientation and scale for each run after a few meters of translation. We also avoid high-speed, in-place rotations in our trajectories that can cause the scale and tracking to diverge due to sudden non-overlapping image frames.

B. Trajectory Calibration

For each sub-trajectory we perform calibration between the VO pose estimates and the *ground-truth* after a few meters of translation using the methods described in [42] by minimizing the least squares error between a set of *ground-truth* and VO poses:

Fig. 5. Example reprojection of the vehicle's position in a previous frame. The image on the left is a past image, and the yellow outline is an estimate of the vehicles location at the current time, shown in the right.

$$\underset{R,t,c}{argmin}\left(\frac{1}{n}\sum_{i=0}^{n}\|p_g(i) - cR(p_v(i)+t)\|^2\right) \tag{2}$$

where R, t and c are rotation, translation scale between the two trajectories respectively and $p_g(i)$ and $p_v(i)$ are the set of poses from when DSO is initialized till time n when a few meters of translation is reached.

C. Reprojection Patch Generation

Since our goal is to predict when tracking will diverge, we use the *ground-truth* to back-project the position of the vehicle into a set of past images and extract a 100 x 600 pixel image patch at this location. The projection of the vehicle's position at time t into a previous image position at time t' is defined as:

$$u_{t,t'}, v_{t,t'} = g(T_t^{t'}\mathbf{p_g(t)}) \tag{3}$$

Where g is the function that projects a point from the world into the cameras image plane and $u_{t,t'}$ and $v_{t,t'}$ are the horizontal and vertical pixel locations of an image at time t' of the vehicle's position at time t.

To generate our set of image patches, the vehicle's pose (from the *ground-truth*) is sampled every 10 cm of translation. For each of these samples, the vehicle's pose is back-projected into previous images at 30 cm intervals up to 10 meters in the past. For each patch that is extracted, we keep a record of the image location where it was extracted (which relates to a possible future position in the world) and the error of the robot's pose estimate at time t. An example of this reprojection is shown in Fig. 5.

V. Approach

Learning a driving controller in a traditional reinforcement-learning framework based on exploration of the state space would be prohibitively sample-inefficient and dangerous for the vehicle. Instead, we mirror the approach used in large-scale self-driving car systems and learn from human training data. We make use of the fact that our vehicle can be easily controlled remotely by an expert driver to collect state-transition data (in terms of *ground-truth* pose information) and input images that can be used to build a predictive model. Our approach to training our model is broken into the following steps:

1) Collect ground-truth pose data and image frames.

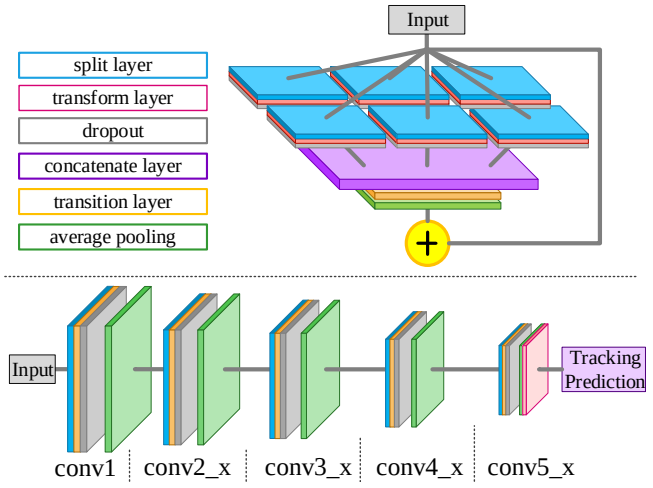

Fig. 6. Overview of the deep convolutional network. The top subfigure shows a close-up view of one of the convolutional stages. The final stage in the network is a global average pooling layer which results in the tracking prediction.

2) Perform VO pose estimation using the image frames until the error between the estimate and *ground-truth* reaches a predetermined threshold, after which the process is restarted:
   - Start DSO and allow it to run for several seconds after which the local DSO frame-of-reference is aligned to the GPS-world frame-of-reference.
   - Allow DSO to run until the pose estimate reaches an error of five meters from the ground-truth.
   - Append the DSO pose estimate and the error at each frame to the pre-collected data.
3) Sample the sequence of image frames (along with the the pose estimation error at that frame) and back-project the location of the vehicle into past images using the *ground-truth* pose and inverse camera model.
4) Extract image patches at the back-projected image location and use the corresponding pose estimate error to train a DCNN.

VI. Model Overview

To associate images with odometry performance, we trained a DCNN constructed using a variant of the architecture family known as ResNeXT [43]. This architecture uses aspects of both the Resnet [44] and Inception [45] network designs to achieve good performance while being sample efficient so that less training data is required than other architectures. This requirement of less training data is accompanied by the use of fewer hyper-parameters for tuning compared to alternative state-of-the-art architectures such as Inception [45]. Equally important in the context of robotics, is that fact that it uses a "split-transform-merge pipeline" with narrower convolutional layers that makes both learning and prediction more efficient and creates less intensive computational demands. As a result, this approach is more suitable for embedded real-time applications, such as our vehicle equipped with only a single TX2 GPU module.

Fig. 7. Prediction accuracy of the training and validation sets after eight hours on a Titan Xp.

The design of ResNeXT uses the stack block design of ResNet, but rather than sequential convolutional blocks, ResNeXT separates each block into multiple paths, as shown in the top portion of Fig. 6. This local sub-division of the network allows ResNeXT converge faster with fewer layers and smaller layer size than competing approaches.

Our network is trained to predict the likelihood of good or poor VO tracking if the vehicle were to drive in the direction the image patch that was extracted. We consider "good" tracking to occur when the error ($e(t)$) is under one meter and assign such patches the corresponding label. Our dataset contains an unbalanced ratio of *good tracking* labels to poor *tracking labels* due to the fact that when DSO diverges from *ground-truth*, we purposefully stop DSO.

Traditionally, over-sampling techniques have been used to overcome this problem, but in our case, many of the 'poor tracking' cases are consecutive image sequences that highly correlated and will be over-fitted by the DCNN. Since the tracking precision is usually good when the motion and illumination varies smoothly, VO methods, especially DSO, often maintain good tracking precision so we can get only a limited number of independent negative samples characteristic of when VO gets lost. Meanwhile, our dataset was also small relative to the number of parameters (300,000 labeled samples for 7 million parameters). As a result, we employ concrete dropout [46] after every convolution layer to keep the network from over-fitting. We also added random noise to the instances of the smaller class while oversampling it as suggested by Yap *et al.* [47].

VII. Experimental Evaluation

We trained our ResNeXT DCNN model using a dataset of roughly 600,000 images which was split into 300,000 images used for training, 150,000 images for validation

and 150,000 images used as a test set. The network was initialized using the pre-trained SE-ResNeXT-50 (8×14d) weights [48] (known to do well on generic vision tasks and associated with the best performing entry in the ImageNet challenge). The associated hyper-parameters that we used are shown in Table I.

Fig. 7 shows the result after 140,000 iterations which took eight hours on an NVidia Titan Xp. The Training accuracy converged after 120,000 iterations to 91% and the final validation error was 79%. On our test set the model prediction accuracy was 86%.

Our data collection was performed in a relatively open area since we wanted to provide the best environment for our low-cost RTK GPS and it was also easily accessible to perform experiments. However, this resulted in less than ideal variations in texture making the distinction between good and poor regions difficult to distinguish between. With this in mind, we are pleased with the prediction accuracy of 86% and expect that an environment with higher variations in texture and terrain will result in an even higher prediction accuracy.

We suspect that the difference in prediction accuracy between the test set and validation set can be attributed to ambiguity in the rate at which DSO diverges for similar looking scenes. We consider tracking to be poor once the error reaches one meter, however the scene leading up and to this error varies and some will lead to quicker divergence than others.

Figure 8 shows two accurately predicted examples from the model. The *poor tracking* prediction shows an image that is mainly grass and trees which alone is challenging for DSO while the lower image contains some buildings which presents some higher contrast textures.

In future work, we plan to use the output of this predictive model in a navigation planner that can make high-level decisions to avoid regions resulting in poor VO estimates and maintain a high confidence of its localization state.

VIII. Conclusions

In this paper, we describe a method for learning to forecast which regions of a given scene are going to have high chance of failure for a given vision-based state estimator, so they can be avoided by the planning module, while the estimator is running concurrently. Our approach uses DSO for visual odometry, and DCNN for modeling the predictions of possibly problematic parts of the scene, but it is also applicable and relevant to other vision-based odometry and state estimation methods. We demonstrate our system on a small-scale self-driving vehicle that traverses about 20km.

In ongoing work, we are applying this method to data collected using an underwater swimming robot and partial results (without the full estimation pipeline shown in this paper) have been achieved for both collision-free navigation [49], [50] and accurate localization [51], [52]. The marine applications are complicated by ocean current (among other factors) and are generally more challenging environments in which to navigate; however, our preliminary results are very promising. Unfortunately, space does not permit their full exposition here. The complete pipeline described in this paper is currently being adapted to a legged swimming vehicle [53], [49]; see Fig. 9.

In the context of our self-driving car, we are also integrating this estimator into the long term planner. While there is little doubt that avoiding regions of perceptual unreliability is generally important, precisely quantifying the benefit depends on task-dependent factors.

Learning Rate	Batch Size	l2 wight	decay factor	Label Smoothing
0.0001	256	0.0001	0.001	0.1

TABLE I

Convolutional neural network hyper parameters.

Fig. 8. Two accurate example predictions from the model (shown horizontally to save space). The top image was predicted to result in poor tracking during a sequence where the VO estimate diverged, while the lower image predicted good tracking while the subsequent error was less than one meter.

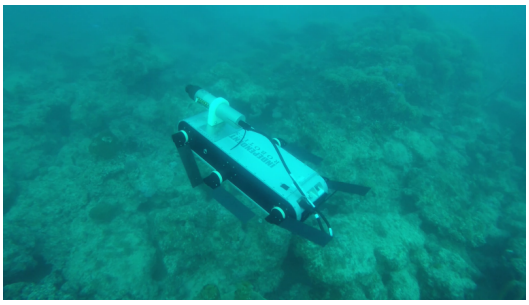

Fig. 9. Vision-based autonomous underwater vehicle exploring a coral reef.

References

[1] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. I652 – I–659.

[2] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 1994, pp. 593–600.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 778–792. [Online]. Available: http://dl.acm.org/citation.cfm?id=1888089.1888148

[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[7] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *arXiv:1712.07629*, Dec 2017.

[8] S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery, "Augmenting inertial navigation with image-based motion estimation," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 4. IEEE, 2002, pp. 4326–4333.

[9] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1403–1410.

[10] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[11] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10, Nov. 2007.

[12] C. Forster, M. Pizzoli, and D. Scaramuzza, in *Proc. IEEE Intl. Conf. on Robotics ...*, 2014.

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[14] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Mar. 2018.

[15] D. G. Kottas, J. A. Hesch, S. L. Bowman, and S. I. Roumeliotis, "On the consistency of vision-aided inertial navigation," in *Experimental Robotics*. Springer, 2013, pp. 303–317.

[16] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2015, pp. 5303–5310.

[17] A. Quattrini Li, A. Coskun, S. M. Doherty, S. Ghasemlou, A. S. Jagtap, M. Modasshir, S. Rahman, A. Singh, M. Xanthidis, J. M. O'Kane, and I. Rekleitis, "Experimental comparison of open source vision based state estimation algorithms," in *International Symposium of Experimental Robotics (ISER)*, Tokyo, Japan, Mar. 2016.

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

[19] G. Chahine and C. Pradalier, "Survey of monocular SLAM algorithms in natural environments," 2018.

[20] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," 2018.

[21] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," 2007, pp. 3565–3572.

[22] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," vol. 30, no. 4, pp. 407–430, 2011.

[23] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," vol. 30, no. 1, pp. 56–79, 2011.

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

[25] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," vol. 2, no. 2, pp. 796–803, 2017.

[26] M. Jenkin, E. Milios, P. Jasiobedzki, N. Bains, and K. Tran, "Global navigation for ark," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, vol. 3, July 1993, pp. 2165–2171 vol.3.

[27] J. Sattar, E. Bourque, P. Giguere, and G. Dudek, "Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction," in *Fourth Canadian Conference on Computer and Robot Vision (CRV '07)*, May 2007, pp. 165–174.

[28] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *Advances in Neural Information Processing Systems*, 2000, pp. 1043–1049.

[29] T. Manderson, A. Holliday, and G. Dudek, "Gaze selection for enhanced visual odometry during navigation," in *Proceedings of the Conference on Computer and Robot Vision (CRV)*, 2018.

[30] R. T. Rodrigues, M. Basiri, A. P. Aguiar, and P. Miraldo, "Low-level active visual navigation: Increasing robustness of vision-based localization using potential fields," *IEEE Robotics and Automation Letters*, vol. 3, pp. 2079–2086, 2018.

[31] C. Thorpe and T. Kanade, "Vision and navigation for the CMU navlab," in *Proceedings of the Symposium on Photogrammetric and Optical Engineering (SPIE)*, vol. 0727, 1987, pp. 0727 – 0727 – 6. [Online]. Available: https://doi.org/10.1117/12.937805

[32] E. D. Dickmanns and V. Graefe, "Applications of dynamic monocular machine vision," *Machine vision and Applications*, vol. 1, no. 4, pp. 241–261, 1988.

[33] M. Betke, E. Haritaoglu, and L. S. Davis, "Real-time multiple vehicle detection and tracking from a moving vehicle," *Machine vision and applications*, vol. 12, no. 2, pp. 69–83, 2000.

[34] G. P. Stein, O. Mano, and A. Shashua, "A robust method for computing vehicle ego-motion," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 362–368.

[35] N. Franceschini, J.-M. Pichon, and C. Blanes, "From insect vision to robot vision," *Phil. Trans. R. Soc. Lond. B*, vol. 337, no. 1281, pp. 283–294, 1992.

[36] D. Jugessur and G. Dudek, "Local appearance for robust object recognition," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 1, June 2000, pp. 834–839 vol.1.

[37] I. Rekleitis, G. Dudek, and E. Milios, "Experiments in free-space triangulation using cooperative localization," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2, Oct 2003, pp. 1777–1782 vol.2.

[38] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International journal of computer vision*, vol. 1, no. 4, pp. 333–356, 1988.

[39] J. Aloimonos, "Purposive and qualitative active vision," in *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, vol. 1. IEEE, 1990, pp. 346–360.

[40] M. Grimes and Y. LeCun, "Efficient off-road localization using visually corrected odometry," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 2649–2654.

[41] C. Gurau, D. Rao, C. H. Tong, and I. Posner, "Learn from experience: Probabilistic prediction of perception performance to avoid failure," *The International Journal of Robotics Research*, 2017.

[42] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, April 1991.

[43] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5987–5995.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[46] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3581–3590. [Online]. Available: http://papers.nips.cc/paper/6949-concrete-dropout.pdf

[47] B. W. Yap, K. A. Rani, H. A. A. Rahman, S. Fong, Z. Khairudin, and N. N. Abdullah, "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets," in *DaEng*, 2013.

[48] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," 2018.

[49] T. Manderson and G. Dudek, "Gpu-assisted learning on an autonomous marine robot for vision based navigation and image understanding," in *MTS/IEEE OCEANS*, Charleston, SC, USA, Oct. 2018.

[50] T. Manderson, R. Cheng, D. Meger, and G. Dudek, "Navigation in the service of enhanced pose estimation," in *International Symposium on Experimental Robotics (ISER)*, 2018.

[51] F. Shkurti, I. Rekleitis, M. Scaccia, and G. Dudek, "State estimation of an underwater robot using visual and inertial information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, US, Sep. 2011, pp. 5054–5060.

[52] S. Rahman, A. Quattrini Li, and I. Rekleitis, "Sonar Visual Inertial SLAM of Underwater Structures," in *IEEE International Conference*

on Robotics and Automation, Brisbane, Australia, May 2018, pp. 5190–5196.

[53] G. Dudek, M. Jenkin, C. Prahacs, A. Hogue, J. Sattar, P. Giguere, A. German, H. Liu, S. Saunderson, A. Ripsman, S. Simhon, L. A. Torres-Mendez, E. Milios, P. Zhang, and I. Rekleitis, "A visually guided swimming robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton AB, Canada, Aug. 2005, pp. 1749–1754.