Teach me to Tango:
Fidelity Estimation of Visual Odometry Systems for Robust Navigation

Stefan Jorgensen
stefantj@stanford.edu

Niveta Iyer
nivetai@stanford.edu

Chelsea Sidrane
csidrane@stanford.edu*

Abstract

Robust navigation in uncertain, cluttered environments is one of the major unsolved technical challenges in robotics. Many recent advances in navigation have come from applying empirically justified techniques (e.g. neural networks or complex vision systems) which work well but rarely provide robustness guarantees. The goal of this project is to use a neural network to accurately predict and track the error rate of the pose estimate of Google Tango on a quadcopter in the Autonomous Systems Laboratory. We compare our approach to a baseline method that predicts error rate using classical machine learning techniques on features extracted by standard image processing algorithms. We also analyze the performance of three different network architectures: Deep RNN, CNN-RNN and Deep CNN, and demonstrate that a deep CNN works best for our problem with a normalized RMSE with respect to nominal of 0.983 and a correlation of 0.504.

1. Introduction

Robotic navigation increasingly relies on complex vision systems such as the Google Tango which are presented to practitioners as a black box. Generating robust motion plans for use with these systems requires knowledge of the reliability of such systems, which the current state of the art does not provide. The objective of this project is to design and test a network which predicts the error rate of such vision systems for use by mid-level path planners which account for safety (e.g. [12]). The inputs available to our network are the IMU data (accelerations) and images from the Tango. The ground truth from a VICON motion capture system is used to calculate the pose error rate, which serve as the labels for our inputs. Several networks are trained to predict the error rate for a new image.

*Stefan Jorgensen is with the Department of Electrical Engineering, Stanford University, Stanford, California 94305. His work is supported by NSF grant DGE-114747 Niveta Iyer and Chelsea Sidrane are with the Department of Aeronautics & Astronautics, Stanford University, Stanford, California 94035

1.1. Related Work

1.1.1 Motivating Work

Our work is directly motivated by Ichter et al. 's work [12]. This work implements a robust motion planning algorithm that attempts to minimize path cost, as well as adhere to a constraint on perception quality. The approach used involves 1) building a graph through the state space corresponding to possible trajectories, 2) computing a heuristic for perception quality, 3) performing multi-objective search subject to a constraint on the heuristic value, and 4) verifying the motion plan in a Monte Carlo fashion. The perception heuristic used in this paper involved counting the number of hand-chosen features in view and checking whether this meets a certain threshold. We propose to learn a model mapping image sequences and IMU data to estimates of perception quality in order to create a more precise indicator of perception quality and therefore path safety.

1.1.2 Correcting odometry errors

Correction of odometry errors using neural networks has been a recent subject of interest for researchers in robotics. The authors of [22] discuss calibration of such errors using a two layer feed-forward neural network with Bayesian regularization. The inputs to the network were Generalized Linear Model (GLM) estimates and the targets were the actual positions of the mobile robot. They compare their Artificial Neural Network approach with traditional localization modules and demonstrate a significant improvement in performance. The authors of [16] discuss similar objectives, but compare a Radial Basis Function neural network trained to predict the position of a robot to Kalman filter estimates of the same.

1.1.3 Localization

On the subject of localization, [3] describes a modular MultiLayer Perceptron algorithm for robot pose estimation in a general indoor environment using SONAR sensor measurements as training data. The work in [5] de-

scribes W-LAN based localization system that is based on a discriminant-adaptive neural network (DANN) architecture. Useful client position information is extracted among access points (APs) and fed into the discriminative components (DCs) of the network, which ranks different positions by information quantity. The DCs are then used to recursively update the weights of the network and a pose estimate is learnt. [8] explores global self-localization as an optical character recognition problem of an indoor room. They experiment with Kohonen and Region Feature Neural Networks (RFNN), which are known to perform association tasks well. All of works cited in 1.1.2 and 1.1.3 demonstrate better performance with neural networks than with classical filtering systems, thus motivating the use of neural networks in odometry and localization systems, which are two problems very closely related to our problem.

1.1.4 Filtering approaches

Filtering-based approaches such as those described in [7] have been used for decades to estimate pose and statistical uncertainty using inertial data such as acceleration. Incorporating vision into such systems, as described in [2], where information from accelerometers and gyroscopes is combined with monocular camera images to derive the pose and velocity of an Unmanned Aerial Vehicle, is a more recent development. [2] uses a non-linear complimentary filter for estimation, but endorses the concept of concatenating image features and IMU data. There has also been a significant body of work focused on identifying and matching hand-defined features from frame to frame in sequences of images [7] in order to feed optical flow estimates into filtering algorithms. A drawback of these methods is that they can require careful calibration and are therefore not robust.

1.1.5 CNNS for human body pose

CNNs are widely used for human pose estimation, as described in [21] and [20]. In [21], they use a seven layer convolutional Deep Neural Network (DNN) to predict a pose vector by minimizing the L2 distance between true and predicted pose vectors. What is interesting is that they train a cascade of pose regressors, i.e. a linear regression layer on top of the last layer of each convolutional block of the network, to capture finer details of the human joint movement. An alternate method is described in [15], where regression and detection tasks are trained simultaneously, and their performances are comparable. In [20], they introduce a unifed network consisting of a ConvNet Part-Detector and a Markof Random Field (MRF) inspired spatial model to detect human body pose and demonstrate higher accuracy than existing architectures. The work in [1] explores combining a pre-trained CNN (as a fixed feature extractor) and continuity filters to recognize geographical places from images. In particular, they first create a confusion matrix from the CNN features to depict place match hypotheses, and apply a spatial continuity check and a sequential filter that captures the motion between two images to compute the strongest place match hypothesis. The idea of capturing information between two images motivates the use of optical flow techniques in our project.

1.1.6 CNNS for optical flow

In the area of using neural networks for optical flow extraction, which we believe to be highly relevant to pose estimation, Fischer et al. [6] demonstrate that it is possible to beat model-based optical flow approaches and blended matching and variational approaches using a CNN trained end-to-end on both 1) several stacked image frames, and 2) two successive frames convolved together. Their network architecture provides a useful reference for designing our network, and provides confidence that end-to-end approaches can be successful for these types of estimation problems. In [14], an interesting motion extraction method is presented where they detect "synchronicity" across stereo frames, unlike traditional optical flow extraction methods.

1.1.7 CNN + RNN for pose

Finally, Clark, Wang et al [4] train a recurrent, convolutional neural net end-to-end for visual-inertial odometry. This approach concatenates IMU information from a recurrent network setup, and images processed using convolutional filters and a correlational map between successive frames in order to achieve performance on-par with an EKF that incorporates optical flow information. This model learns to predict pose – which is very similar to our goal of predicting pose error rate. For this reason, we use similar architechtures for our project.

2. Problem Statement and Approach

The network seeks to learn a transform from an input sequence of N images and inertial data to a sequence of N error rates for the pose estimate:

$$E : \{(\mathbb{R}^{640\times480}, \mathbb{R}^9)_{1:N}\} \rightarrow \{(\mathbb{R}^1)_{1:N}\} \qquad (1)$$

This particular combination of inputs and outputs is motivated by the Tango itself. The Tango performs localization using inertial data, image data, and a map of the room. Therefore, we hypothesized that error rates could be learned from this input information as well.

Additionally, we were inspired by the success of [4]. This work describes a successful approach to sequence-to-sequence visual-inertial odometry learning. The goal is to predict the pose of a robot based on a sequence of images

and inertial measurements. The network architecture proposed in [4] is shown in Figure 1. A CNN is used to process the image sequence, and an RNN is used to process the pose data. The two are fused by concatenating the feature vectors and passing it through an LSTM layer. A convolutional and recurrent architecture which works on raw images also makes intuitive sense for our problem, as it should be able to capture the fine features, time dependent nature of the error rate changes, and the dynamics of the Tango estimator.

The deep convolutional architecture used in [4] follows the trend of success of deep convolutional networks in image classification competitions such as Imagenet. ResNet, a winner of the 2015 ILSVRC challenge, has 152 layers [10]. For this reason, we also explore using deep convolutional nets for our problem. Although our problem involves regression and not classification, we believe that the powerful generalization ability of a deep network could provide good predictions.

2.1. Experimental Setup

Our data comes from the experimental setup developed by Benoit Landry, Brian Ichter, Ed Schmerling, and Prof. Pavone of the Autonomous Systems Laboratory (described in detail in their paper, [12]). The testbed is equipped with a VICON system, which uses eight infrared cameras to estimate the pose of special reflective markers. Several markers were placed on a quadcopter and its position relative to the starting point recorded using the Robot Operating System (ROS). The quadcopter was also equipped with a Google Tango. The Tango does not give access to any quality metrics such as covariance estimates. The position estimate and image data from the Tango was also recorded using ROS. The quadcopter was put through several trajectories around the testbed. White sheets were hung in the testbed room in order to purposefully provide feature-poor localization areas.

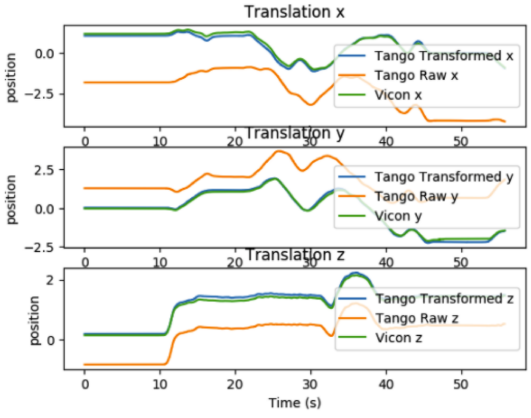

Figure 2. Example dataset. X,Y & Z position estimated by VICON system (green) and by Google Tango - unaligned (orange) and aligned (blue).

2.2. Data Pre-Processing

Our dataset consists of 15,998 images and 67,964 inertial and position messages. Figure 2.1 shows a sample of position estimates. The reference frame of the Tango data was aligned to the reference frame of the VICON system by performing a least squares fit, as described by [17], over the entire sequence of data.

The aligned position data from the Vicon and the Tango was used to compute an error signal, with each time stamp having an x,y, and z error. The norm of the error was computed for each time stamp, and a finite-difference was taken in order to produce the "label" for each image: the rate of change of the magnitude of the error.

We experimented with several pre-processing schemes for the image data. These pre-processing steps included:

- Subtracting the mean and normalizing
- Downsampling from 640 x 480 to 64x48
- Computing dense optical flow
- Cropping the image to 64x48, centered
- centering IMU streams

3. Experiments/Results/Discussion

3.1. Metrics

Given N test images, we denote y_i as the true label for image i, $\hat{y}_i$ as the predicted label, and x_i as the ith input data which is either an image, IMU, or concatenation of the two, depending on the setting. We also denote the mean label as μ and the mean prediction as $\hat{\mu}$, and standard deviations of the label and prediction as σ and $\hat{\sigma}$, respectively.

We use several metrics to evaluate the quality of a prediction. The root-mean squared error,

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{2}$$

measures the power of the error signal and is a common metric for comparing two signals. The relative RMSE,

$$RMSE_{rel}(y, \hat{y}) = \frac{RMSE(y, \hat{y})}{RMSE(y, \vec{0})} \tag{3}$$

gives the ratio of the root-mean squared error relative to a zero-prediction. This can be interpreted as the inverse of the signal-to-noise ratio. One weakness of the RMSE is that it does not necessarily reflect how similar the *shape* of two signals is, and so we also compute the correlation:

$$\frac{1}{\sigma\hat{\sigma}}\sum_{n=1}^{N}|(y_n - \mu)(\hat{y}_n - \hat{\mu})| \tag{4}$$

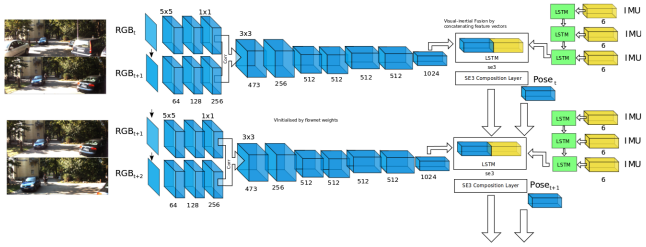

Figure 1. Network architecture from [4].

Input
640x480 Images
640x480 Opt. Flow
9DoF IMU
IMU + 64x48 Cropped Opt. Flow
64x48 Downsampled Opt. Flow
Architecture
SqueezeNet
ResNet
Single CNN + Deep LSTM
Single CNN + LSTM
SqueezeNet +LSTM
Output
Error Rate Magnitude
Discretized Error Rate Magnitude

Figure 3. List of inputs, architectures, and outputs used

In the discrete prediction setting, where we discretize the error rates into a finite number of bins, we also consider classification accuracy:

$$\frac{1}{N}\sum_{n=1}^{N}\mathbf{1}\{y_n = \hat{y}_n\} \tag{5}$$

3.2. Traditional Baseline

3.2.1 Features

MATLAB's Computer Vision System Toolbox was used to extract features from the images. Three main types of features were extracted:

- Local Binary Patterns: LBPs capture local textural information. A set of 8 neighbors was selected from a circularly symmetric pattern of radius 1 around each pixel to compute their correlations.
- BRISK Features: Since corners play a major role in position estimation and hence the error rate, BRISK, or Binary Robust Invariant Scalable Keypoints were extracted using the BRISK algorithm in MATLAB. An example of the feature is presented in 4.
- Harris Corner Detectors: This uses the Harris Stephens algorithm [9] to detect corners in an image. An example of the feature is presented in 5.

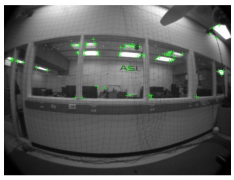

Figure 4. Example Harris Corner Detections

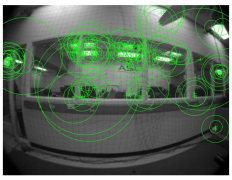

Figure 5. Example BRISK Features

As a baseline, the non-linear regression module from Statics and Machine Learning Toolbox in MATLAB was used as a preliminary model for prediction. The metric used for accuracy measurement was the Root Mean Squared Error (RMSE), which is defined by equation 2. The first step was to experiment with different non-linear regression models to conclude with the best model for testing. The various models used were: Linear Regression (model contains an intercept and linear terms for each predictor), Quadratic Regression (model contains an intercept, linear terms, and squared terms), Non-Linear Regression (SVM regression for high dimensional data, commonly used for images), Stepwise Linear Regression, SVM for Regression and Gaussian Process Regression (GPR).

A comparison of the different methods is given in Fig 7. Clearly, SVM for regression outperforms all other models, hence this was used to perform predictions and set the final error on the test set. An interesting observation was that the error shot up for the Gaussian regression model, which suggests that a Gaussian distribution may not represent the data very well.

The predicted measurements were overlaid against the ground truth to visualize the performance of the model. The label used was the average error rate in x, y, and z directions. Fig. 6 shows that the SVM predictor does a decent job in retaining the overall shape of the ground truth. The RMSE relative to nominal over the test data was 6.356, meaning the classifier does not perform well according to this metric. The correlation obtained was 0.0279, hence the model does not qualitatively capture the error signal properties as well.

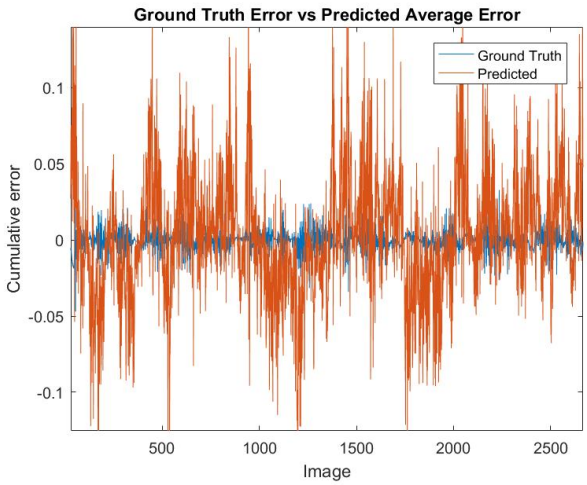

Figure 6. Comparison between true and predicted error rate

Thus, we turn to CNNs and RNNs to accurately learn and predict these error rates.

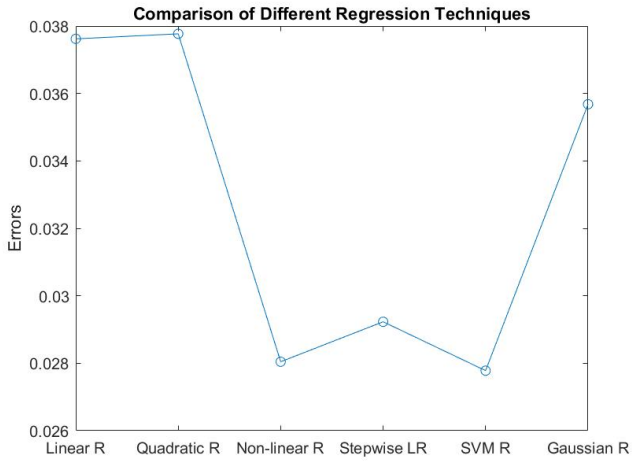

Figure 7. Comparison between different regression techniques

3.3. Recurrent Models

As a second baseline, we use a recursive network with sequences of IMU data as input, sequences of error rate as output, and with RMSE cost. To capture the image data we added an additional input dimension for the Frobenius norm of the optical flow. The best recursive model we trained gave a relative RMSE of 1.05 and a correlation 0.002, which means that it essentially performs as well as predicting zero at every time step. This is not terribly surprising because it does not utilize much image data, whereas the Google Tango relies primarily on image input.

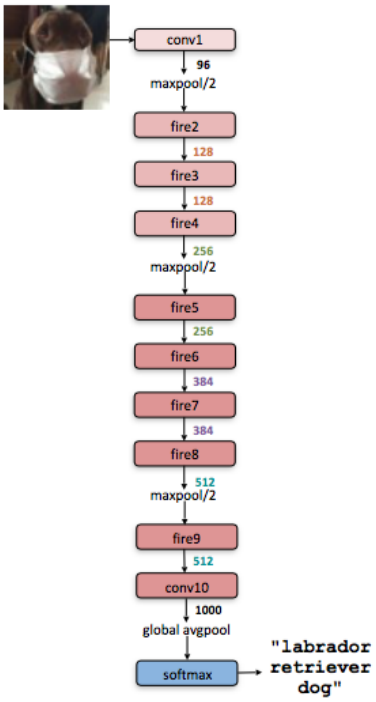

Figure 8. Squeezenet Architechture [11]

3.4. Convolutional Models

Next we considered purely convolutional models which do not take IMU or position data as inputs, but only images or dense flows computed using OpenCV. We did limited training using the full 640x480 images (or flows), as these images quickly filled the available GPU memory and took excessive training time.

We pose the problem as a classification problem by discretizing the error rates into levels that were selected in order to balance the number of training datapoints in each level. We trained our networks using a softmax classifier with entropic loss with anywhere from 3 to 128 levels of discretization. The architectures we tried were a single layer network, SqueezeNet [11] [18] , and ResNet [10] [19].

The SqueezeNet architecture (see Fig. 8) with 16 levels is the only network which we were able to train to give non-trivial results. The Squeezenet architechture features Fire modules which involves "squeeze" layers (1x1 convolutions) and "expand" layers. This structure was designed to maximize accuracy and minimize the number of parameters needed, as a 1x1 convolutional filter has many fewer parameters than a 3x3 filter.

We used a learning rate of 0.0002 with a decay of 0.98 applied every 300 iterations and batch size 1024. Low resolution (64x48) flows were fed into the network. After 30,000 training iterations, the network achieved a training accuracy of 0.511, and validation accuracy of 0.09 (which is

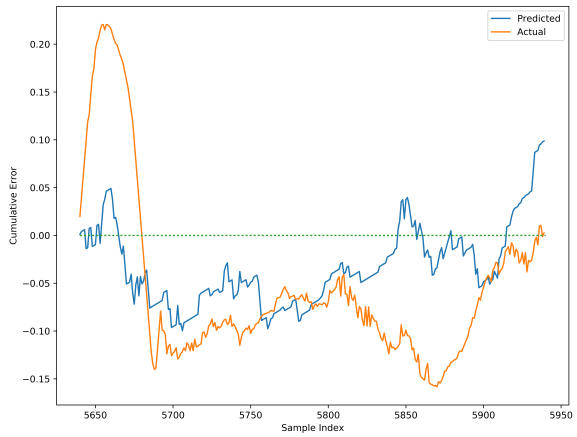

Figure 9. Comparison of predicted cumulative error (blue) versus actual error (yellow) over a 10s window of validation data.

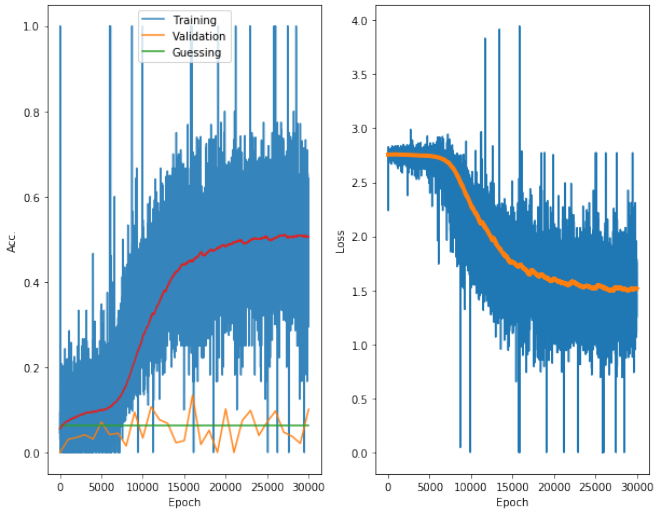

Figure 10. Left: Accuracy versus training iteration for training (blue, mean shown in read), nominal (green) and validation (yellow). Right: Typical learning curve for SqueezeNet on our problem. Note the sharp decrease in loss after 8000 training iterations. This long training time makes it difficult to work with large input images.

47% better than random guessing). The normalized RMSE with respect to nominal is 0.983 and correlation is 0.504. Figure 9 shows the predicted sequence versus the ground truth for the first 10 seconds of validation data (i.e. this example was not cherry picked).

3.5. Hybrid Models

We also considered hybrid CNN and RNN approaches inspired by the VINet architecture. We tried a network with a single convolutional layer and three LSTM layers, as well as the activations of SqueezeNet fed into an LSTM. We found that the hybrid networks were difficult to train, even when we loaded pre-trained weights to the squeezenet portion of the network. The best results had low correlation.

3.5.1 Single CNN + Multilayer LSTM

This model used a single 2D convolutional layer to perform feature extraction on the image inputs. 640x480 flows center cropped to 64x48 were fed into the model. A dense layer was used to extract features from the IMU data. The two feature vectors were concatenated together and passed to a three-layer LSTM, each layer with a hidden state size of 512 neurons, and which processed 10 timesteps of data at time. The LSTM outputs were passed into a dense layer to produce scores for 100 discretized bins. The model was implemented in Keras [13].

The model was trained using the raw error rates, and then a locally weighted average was applied to the predictions and the true labels in order to calculate the signal statistics.

The resulting predictions have very low correlation of 0.029, but the RMSE was 0.0042 with a RMSE over nominal of 1.66. While the validation accuracy for the discretized data was 0.012, the within-one-neighboring-bin accuracy was 0.038. Our problem is a regression problem, not truly a classification problem, meaning that this within-one accuracy metric may better reflect the performance of the model.

3.5.2 SqueezeNet + LSTM

Inspired by VINet, we built a network which passes optical flows through SqueezeNet, then passes the concatenated vector of SqueezeNet activations and IMU data to an LSTM. When the network weights were trained from initialization, we were not able to find a set of hyperparameters that yielded better results than the nominal zero predictions (the best network had correlation 0.005 and relative RMSE 1.001). Since the SqueezeNet portion of the network took a much smaller learning rate to train (0.0002) versus the RNN portion (0.01) in our earlier experiments, we tried pre-trained weights for the squeezenet portion of the network but without improvements.

4. Conclusions and Future work

The model that yielded the best reqults was SqueezeNet. With this model, we obtained statistically significant predictions with correlation 0.504. Qualitatively, the signals output by the network are often the right ”shape” over small time horizons, as illustrated in Figure 9 from 5700 to 5800. The network also predicts approximately correct disturbance locations, as seen in figure 11. However what the network does not do well is predict the correct magnitude

at the correct time - for the sample shown in Figure 11, the predicted error bursts tend to be after the ground truth error bursts. Quantitatively, our best network does significantly better than the baselines, as shown by the high correlation between the predicted and actual error rates.

Our best network appears to overfit, since it classifies with 50% accuracy over the training set but only 9% accuracy on the validation set. Misclassification does not necessarily mean that the prediction is bad, since adjacent bins have relatively similar prediction values, but this discrepancy is significantly higher than it should be. We did not use dropout or other regularization techniques because they slow convergence time and our training time was already excessive.

The most difficult issues that we dealt with were 1) selecting a learning rate which led to good performance, and 2) batching the data in a way that let the network learn the appropriate time dependent relationship between the input and outputs.

Finding a good learning rate is difficult because the network takes a long time to start learning, as shown in Figure 10, where the loss does not decrease until 8000 training iterations. This is further exasperated by the high variance in the loss curve, which makes automated approaches to identify good learning rates difficult.

Batching the data is a challenge due to the inherent correlation in our data gathering process. Ideally training samples would be independent and identically distributed, but in practice they correspond to maneuvers such as 'turning', 'slowing down', and so on, which have significant effects on the error rate. Selecting batches in such a way that the network can learn these maneuvers is something which should be considered in future work.

Future work should also use higher resolution input data. We were unable to train a stable network using high resolution images for the experiments described here, so low resolution flows were used. Additionally, in our experiments cropping high resolution data was not effective. We hypothesize that using another method that first incorporates high resolution features, and then preserves high resolution features from early activation layers, may significantly improve results.

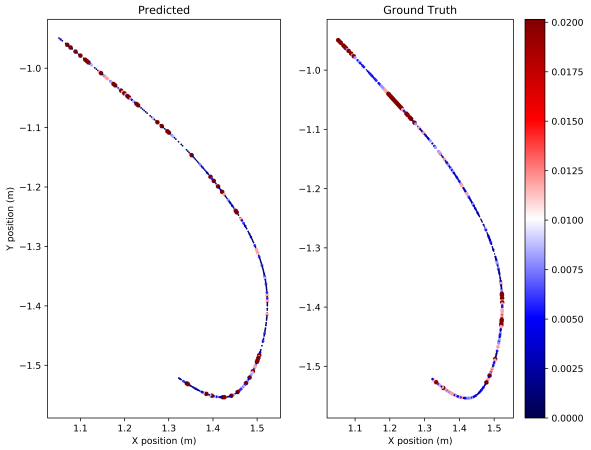

Figure 11. Trace of predicted versus actual error rates over 10 seconds of validation data. The color/size of a datapoint corresponds to the error rate (units of m/s). Note the similarity in frequency and intensity of error 'bursts', although the predictions occur at somewhat different times than the actual error bursts.

References

[1] Z. Chen, O. Lam, A. Jacobson, and M. Milford. Convolutional neural network-based place recognition. *arXiv preprint arXiv:1411.1509*, 2014.

[2] T. Cheviron, T. Hamel, R. Mahony, and G. Baldwin. Robust nonlinear fusion of inertial and visual data for position, velocity and attitude estimation of uav. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2010–2016. IEEE, 2007.

[3] W.-S. Choi and S.-Y. Oh. Range sensor-based robot localization using neural network. In *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*, pages 230–234. IEEE, 2007.

[4] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Conference on Artificial Intelligence*. AAAI, 2016.

[5] S.-H. Fang and T.-N. Lin. Indoor location system based on discriminant-adaptive neural network in ieee 802.11 environments. *IEEE transactions on neural networks*, 19(11):1973–1978, 2008.

[6] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[7] J. Gui, D. Gu, S. Wang, and H. Hu. A review of visual inertial odometry from filtering and optimisation perspectives. *Advanced Robotics*, 29(20):1289–1301, 2015.

[8] R. Gutierrez, T. A. Chase, M. W. White, and J. C. Sutton III. Autonomous mobile robot global self-localization using kohonen and region-feature neural networks. *Journal of Robotic Systems*, 14(4):263–282, 1997.

[9] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[11] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size, 2016.

[12] B. Ichter, B. Landry, E. Schmerling, and M. Pavone. Robust motion planning via perception-aware multiobjective search on GPUs. In *Field and Service Robotics*, Sept. 2017. Submitted.

[13] Keras. Keras.

[14] K. R. Konda and R. Memisevic. Learning visual odometry with a convolutional network. In *VISAPP (1)*, pages 486–490, 2015.

[15] S. Li and A. B. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Asian Conference on Computer Vision*, pages 332–347. Springer, 2014.

[16] I. Petrović, E. Ivanjko, and N. Perić. Neural network based correction of odometry errors in mobile robots. In *FIRA Congress 2002*, 2002.

[17] O. Sorkine-Hornung and M. Rabinovich. Least-squares rigid motion using SVD. Technical report, Department of Computer Science, ETH Zurich, 01 2017.

[18] C. C. Staff. Assignment 3 code: Squeezenet.

[19] TensorFlow. Tensorflow/models/resnet.

[20] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[21] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[22] H. Xu and J. J. Collins. Estimating the odometry error of a mobile robot by neural networks. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 378–385. IEEE, 2009.