Deep Visual Gravity Vector Detection for Unmanned Aircraft Attitude Estimation

Gary J. Ellingson
Department of Mechanical Engineering, Brigham Young University, gellings13@gmail.com

David Wingate
Brigham Young University - Provo

Tim McLain
Brigham Young University - Provo, mclain@byu.edu

Original Publication Citation

2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 2017, Vancouver, British Columbia.

Deep Visual Gravity Vector Detection for Unmanned Aircraft Attitude Estimation

Gary Ellingson[1], David Wingate[2], and Tim McLain[3]

Abstract— **This paper demonstrates a feasible method for using a deep neural network as a sensor to estimate the attitude of a flying vehicle using only flight video. A dataset of still images and associated gravity vectors was collected and used to perform supervised learning. The network builds on a previously trained network and was trained to be able to approximate the attitude of the camera with an average error of about 8 degrees. Flight test video was recorded and processed with a relatively simple visual odometry method. The aircraft attitude is then estimated with the visual odometry as the state propagation and network providing the attitude measurement in an extended Kalman filter. Results show that the proposed method of having the neural network provide a gravity vector attitude measurement from the flight imagery reduces the standard deviation of the attitude error by approximately 12 times compared to a baseline approach.**

I. Introduction

Flying vehicles of all types (rockets, fixed-wing aircraft, and rotarcraft) must overcome the force of gravity to remain airborne. This fundamental task requires an accurate knowledge of the vehicle's attitude relative to gravity. While three-axis acceleration and rotation sensors (accelerometers and gyros) are often used for attitude estimation, they have a number of limitations. These sensors have inaccuracies in the form of noise and drift and are often sensitive to changes in temperature. Also, accelerometers measure both the portion of the specific force counteracting gravity and the inertial acceleration of the vehicle. The use of these sensors requires complex estimation frameworks to obtain the attitude of the vehicle and are often combined with other sensors such as cameras [1] to improve their accuracy.

People have an innate ability to sense their attitude from visual clues. They can easily tell the orientation at which a picture was taken simply by looking at the picture. This ability is demonstrated by how people react to situations where visual clues do not match their actual orientation and what they see does not match their internal senses. A common example of this is the disorientation and imbalance a person can experience when wearing a virtual reality headset.

First-person-view radio-controlled drone pilots are also able to perceive the drone's position, velocity, and pose given only a monocular video. Skilled pilots can perform quick, accurate, and aggressive maneuvers. This implies that all necessary state information for these maneuvers is contained in the video, although accessing and quantifying the sensory information for robotics applications remains difficult in practice.

Deep learning is one of the most compelling advances in machine learning in recent memory. It has swept over both industry and academia, crushing benchmarks and generating impressive progress across fields as diverse as speech recognition [2]–[5], parsing of natural scenes [6], [7], machine translation [8]–[10], robotics [11]–[14], machine vision [15]–[18], and even the game of Go [19]. Convolutional neural networks have been applied to image processing for tasks such as detection and classification of objects in the images [20]. By making the network deeper (i.e. more layers) performance of these networks can be improved. Image processing is a good application for deep learning because it can be hard to mathematically model many visual tasks and large datasets exist for training the deep neural networks (DNNs). DNNs are able to learn (through optimization) a function approximation for performing the visual tasks. Integrating DNNs into robotics applications, however, remains an area of ongoing research.

There is an extensive history of unmanned aircraft using vision [21], including the limited use of DNNs for visual perception. To our knowledge there has been no use of DNNs for state estimation purposes. Visual odometry (VO) is commonly used on autonomous vehicles for measuring the movement of the vehicle using visual information. VO has a long history [22] that includes a large variety of methods and techniques [23]. Often methods include tracking features in the images and using the movement of these features across frames to construct a transformation from one frame to the next.

This paper demonstrates the use of deep learning to perform visual gravity vector detection. To do this, we create a DNN where the input to the network is a monocular image and the output is the orientation of the camera relative to gravity. This allows the camera to be a direct attitude sensor, which simplifies the attitude estimation for a flying vehicle. The paper demonstrates the possibility of real-time use of DNNs for attitude estimation using flight video from a small multirotor aircraft.

II. Related Works

Several visual attitude estimation approaches have already been proposed. Nearly all approaches use an edge detection

*This work has been supported by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award No. IIP-1161036.

[1]Gary Ellingson is a PhD candidate in the Department of Mechanical Engineering, Brigham Young University gary.ellingson@byu.edu

[2]David Wingate is an associate professor in the Department of Computer Science, Brigham Young University, wingated@cs.byu.edu

[3]Tim McLain is a professor in the Department of Mechanical Engineering, Brigham Young University mclain@byu.edu

algorithm and use the edges to calculate attitude information. Sensing the horizon using an FPGA was used to measure the pitch and roll angles of the aircraft [24]. Regularities in urban environments provide edges that point to vanishing points. These edges were used directly in a Kalman filter update in [25] to limit the drift of inertial sensors. A similar approach was used on a ground robot in indoor environments [26]. By assuming all edges are orthogonal in three directions (the so-called Manhatan World assumption) a vanishing-point filter was developed that estimates the Euler angles of the camera attitude [27]. All of these approaches use strict assumptions and, in some cases, the papers reported failures when assumptions were not true.

Deep learning has also been used on robots and autonomous aircraft. Using a DNN classifier, localization relative to a satellite image and landing zone detection for a parafoil aircraft has been demonstrated [28]. This work also showed that the output of a DNN can be used as a sensor for an update in a Bayesian filter. Obstacle avoidance by a ground robot was performed in [29] by training on human inputs. Deep learning has further been used for aircraft perception and mapping from image and hyperspectral data [30]. DNNs were used in [31] to predict the success of a robotic gripper using camera images.

III. Building the DNN

For a DNN to accurately detect the attitude at which an image was taken, the DNN must be constructed and trained with care. The following sections describe the collection of training data, the architecture of the network, and the results of training.

A. Training Data

Prior to this work, there did not exist a large dataset of images collected at various attitudes that also included precise measurement of the attitude of the camera. For this reason a dataset was constructed for training the DNN.

First an Android application (app) was written for collecting the data. The app simultaneously takes a cropped picture from the phone's camera and records data from the phone's internal accelerometers. To reduce accelerometer noise 50 measurements were quickly taken and averaged to produce the attitude measurement of the camera. The app used OpenCV vision processing libraries [32] for image capture and processing. A screenshot of the app can be seen in Fig. 2.

No camera calibration was performed for the training dataset. The principle point of the image was assumed to be the image center and the camera frame was a standard east-down-north frame. The transformation from accelerometer frame to the image frame was assumed to be axis aligned according the Android API. This means the image x, y, z axes were assumed to be aligned with the accelerometer x, $-y$, $-z$ axes respectively.

1) Data Collection: The data was collected on a LG Nexus 5 smartphone around the Brigham Young University

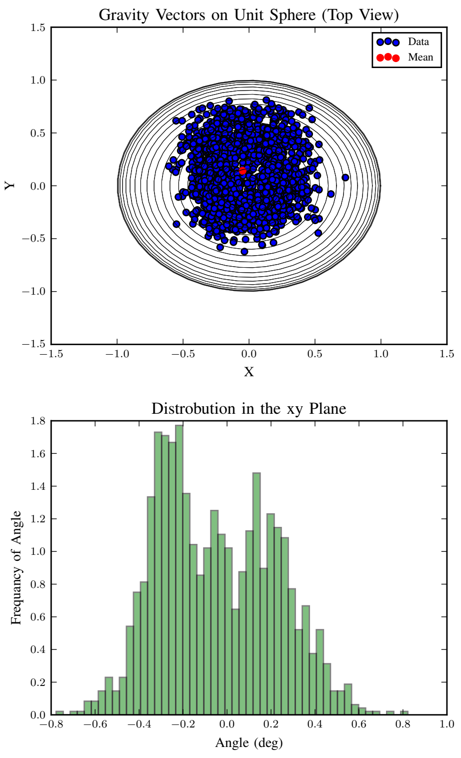

Fig. 1. *Top: Normalized gravity vectors plotted on the unit sphere in the camera frame. Showing the orientations of the collected images. Bottom: Distribution of the gravity vector angles about the optical axis. Used to resample angles for dataset augmentation.*

(BYU) campus. Attempts were made to minimize acceleration while the pictures were captured so that the accelerometers would measure mainly components of the gravity vector. The dataset includes images of the inside and outside of campus buildings taken at various attitudes that were similar to what a multirotor might experience in a non-aggressive flight (within about 40 degrees of vertical). However, one limitation to the data is that it was collected by a standing person and not a flying vehicle, meaning that virtually all the images are 4 to 6 feet from the ground rather then at all possible altitudes. In total 1500 images were taken. Example images are shown in Fig. 2.

2) Data Analysis and Augmentation: The data was analyzed to prior to training to show the distribution of the data collected. Fig. 1 shows all the gravity vectors normalized and plotted on the unit sphere as well as the distribution of the data around the optical (accelerometer $-z$) axis.

Simple statistics were also calculated for the dataset to construct a baseline comparison for the DNN. Two randomly selected gravity vectors sampled from the dataset have on average 31.5 degrees of angular separation. Further, the average angular deviation from the mean gravity vector is 22.0 degrees. This means that if the neural network is able to achieve less than 22.0 degrees of error between the measured and estimated gravity vectors then it is doing better than just learning the distribution of dataset. Therefore 22.0 degrees will be the baseline comparison for the DNN results and the

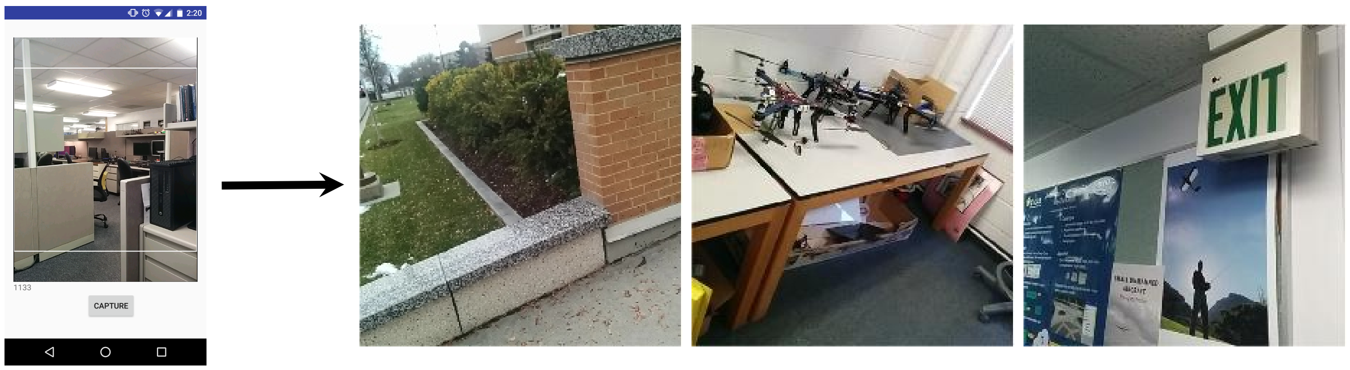

Fig. 2. *Data collection Android application and examples of the collected images (associated gravity vectors were also collected). The reader can note that the orientations of the images, while unusual, are easily perceived.*

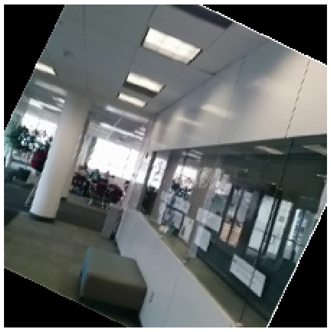

Fig. 3. *Image rotated around the image center. Used to augment the training dataset.*

mean gravity vector a baseline measurement comparison on the flight test results.

It is important to note that a dataset of only 1500 images is small for training a deep conventional neural network and is critical to the DNN design. To augment the data, each image was rotated around the image center to a new orientation within the original distribution. Images were padded with zeros and pixels were resampled with bilinear interpolation, as shown in Fig. 3. Gravity vectors were also rotated around the $-z$ axis by the same angle to match the orientation of the resampled image. Initial experiments showed that data augmentation greatly reduced the DNNs overfitting on the dataset and generalized well to unrotated images.

The dataset was also split into training and testing data. Because of the relatively small size of the original dataset and because the dataset augmentation made every training batch unique, only 100 images were randomly selected and removed for testing, leaving 1400 for training. Both test and training sets were rotated as described above before being input into the DNN.

B. Technical Approach

The ideal approach for this problem would be to train a full custom DNN for approximating the attitude from an image. However, state of the art DNNs require datasets with millions of training instances [33] and are trained with hundreds of thousands of training steps each with large mini-batches [34]. Without the ability to use extremely large training datasets and with relatively limited computational resources we will require a simpler approach. Thus, the results presented here represent a proof of concept only and not a state-of-the-art implementation.

1) DNN Architecture: The first design choice was to reuse a previously trained DNN for the base of the new DNN architecture, a common approach [35]. Even though the DNN was trained on a different task, because it was trained on similar data it has already learned both the lower-level texture information and the high-level semantic information useful to the new task. In practice, it also provides the depth required for the new task without the computation burden of training the entire network. Our DNN architecture reuses a 16-layer DNN created by the Visual Geometry Group at the University of Oxford [36] that was previously trained with data from the ImageNet database [33] to perform image classification. We will refer to the network itself as VGG.

The next choice was to decide at what level to pull activations from VGG. While the top of the network must be replaced and trained on the new task, it was initially unclear where on the network to take the activations for input into the replacement. It is generally accepted that the lowest levels of a DNN learn about texture information and the highest levels learn about semantic information. Both texture and semantic information, however, could be useful in determining the attitude of the camera. For example, texture could be used to detect the edges of objects that point toward vanishing points on the horizon, and semantic information could be used to detect whether the image contains sky or grass, which are both clues for determining orientation.

A small topology search was performed to determine which level of the VGG network to use. This was done by training on several different levels and comparing performance. One convolutional layer and two fully connected layers were added to activations from VGG layers conv1.1, conv2.1, conv3.1, conv4.1, and conv5.1 and then individually trained. Fig. 4 shows the final design with activations used

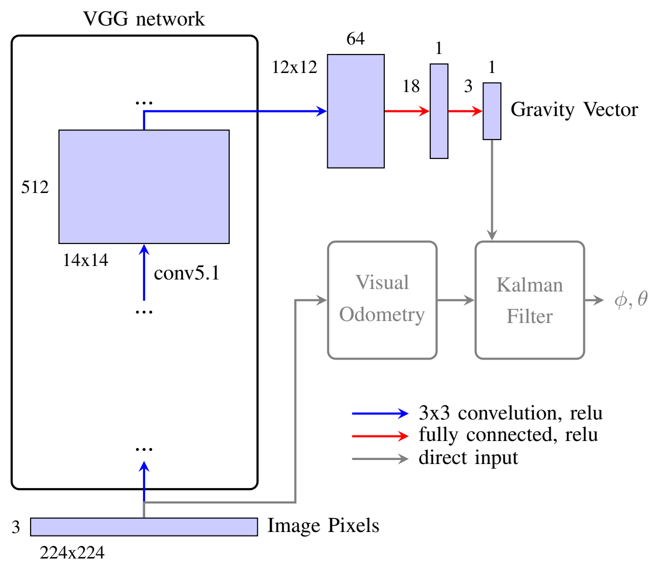

Fig. 4. *Computation graph showing the DNN architecture. The original VGG DNN goes up the left side of the image. The addition of added network uses activations from level conv5.1 of VGG. Gray sections indicate the flight testing attitude estimation scheme (see Section IV).*

from the conv5.1 layer of VGG that achieved the best accuracy.

The VGG activations are input into a bank of 64 3×3×512 convolutions. The result of the convolution is input into two fully connected layers of sizes 9216×18 and 18×3. The final output is the estimated gravity vector.

2) Loss Function: The loss function comes from treating the output of the network as a log probability. The loss L is expressed by the function

$$L(y_i, \mu_i) = (y_{ix} - \mu_{ix})^2 + (y_{iy} - \mu_{iy})^2 + (y_{iz} - \mu_{iz})^2$$

where μ_i is the output vector from the DNN for image i and y is the associated true gravity vector recorded by the accelerometers.

C. Training Results

The network was trained using TensorFlow [37] with a batch size of 100 images for approximately 250 epochs.

After training, the DNN produced gravity vectors that are on average within 8 degrees of the measured attitude. Fig. 5 shows the accuracy as the network was trained. Note that the training data does not dip below the test data, which means that the network was not over fitting the training data. Because the error is far below the baseline, it is clear that the DNN did not learn the distribution of the dataset, but has actually learned to perceive the gravity vector from the image.

One limitation to the data is that the results can only be as accurate as the accelerometer readings during data collection. While averaging several measurement may have helped to reduce the noise of the sensor, there was no way to account

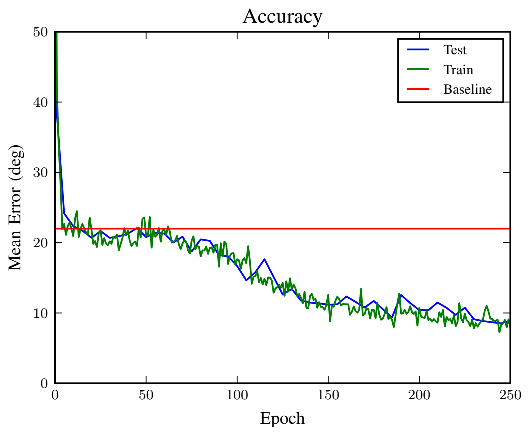

Fig. 5. *Mean error (deg) for both training and testing data as the DNN is trained. Training accuracy does not dip below the testing accuracy, indicating the DNN did not overfit on the training data.*

for the bias drift on the sensors. Further analysis showed that accelerometer drift can cause at least 2.5 degrees of error and normal hand movement of a camera phone may have caused up to 4.3 degrees of error. This means that better accuracy of the DNN may be achievable if the attitude of the images in the dataset were known with more accuracy.

IV. Experimental Flight Tests

The purpose of the flight tests is to show the feasibility of using the DNN as a sensor in an attitude estimation filter. A successful test would result in a significant reduction in the error of the aircraft attitude estimates. The experimental setup, including video processing and measurement filtering, as well as flight test results are described below.

A. Approach

Obtaining the true attitude is important for evaluating the accuracy of the attitude estimates. Motion capture (Vicon) is commonly used as ground truth and provides an accurate basis for comparison. It was considered for this experiment, but using motion capture limits the flight environments to the location of the motion capture system. Since a motion capture room did not provide a diverse, visually-rich environment necessary to evaluate the effectiveness of the DNN measurements, we require a different approach.

Alternatively, the aircraft's attitude with respect to the gravity vector is also known at the beginning and end of the flight when the aircraft is sitting on a level surface. At these times, the true attitude of the aircraft is given by pitch and roll angles of zero. Therefore, the attitude estimates can be initialized with zeros and the estimated attitude at the end of the flight used to evaluate the accumulated error in the attitude estimates, and thus, effectiveness of the approach.

Our experiment used 33 flights of an Inductrix FPV multirotor aircraft by Horizon Hobby (see Fig. 6). Analog video was transmitted from the aircraft and recorded for analysis in post processing. For each flight the aircraft began on a level surface, flew manually for approximately 60 seconds

Fig. 6. *Inductrix FPV aircraft used in test flights.*

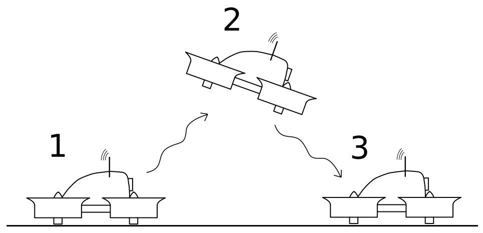

Fig. 7. *Test flight processes where the vehicle* 1 *began on a level surface,* 2 *flew for approximately 60 seconds, and* 3 *landed on a level surface.*

and landed on a level surface (see Fig. 7). Each flight was flown without aggressive maneuvers in indoor environments and at similar altitude to the training data.

1) Visual Odometry: Visual odometry was implemented using OpenCV and used to process the recorded video to obtain the attitude of the aircraft (pitch and roll angles) throughout the flight. For all flights the initial angles were assumed to be zero.

The VO method used implemented a pyramidal KLT tracker [38]–[40] for finding feature correspondences. OpenCV's built-in functions were used to find the fundamental matrix between two frames using an eight-point algorithm [41] and RANSAC [42] to reject outliers. The fundamental matrix was then decomposed into a transformation between camera poses. The transformations were compounded from frame to frame for video at 15 Hz to produce the position and orientation of the vehicle throughout the flight. The visual odometry implemented here does not represent a state-of-the-art solution but similar approaches are commonly used to obtain the motion of a camera. The visual odometry required that the multirotor's camera be calibrated to obtain the camera's intrinsic parameters. The camera was assumed to be body-centered and axis-aligned with the vehicle.

The result of the visual odometry is a homogenous transformation matrix from the camera's initial position to the current position. The pitch and roll angles of the aircraft were obtained by decomposing rotation portion of the matrix into Euler angles. The rotation sequence included a yaw about the camera y axis, a pitch (θ) about the camera x axis, and then a roll (ϕ) about the camera z axis.

While the performance of visual odometry methods varies, all VO methods, without other sensing, are subject to drift as errors accumulate. The top left plot of Fig. 8 shows the pitch-angle performance of our method. Without any attitude measurement, the attitude quickly drifts from the truth and the attitude at landing varies widely from the true value of zero pitch.

2) Extended Kalman Filter: An extended Kalman filter (EKF) was then implement to incorporate the visual gravity vector detection as a measurement update. The EKF matrix equations are similar the attitude estimator in [43] but are simplified because the measurement only includes the gravity vector and not the body accelerations. The filter states are roll and pitch angles of the aircraft or

$$x = \begin{bmatrix} \phi \\ \theta \end{bmatrix}$$

. The nonlinear propagation step ($\dot{x} = f(x,u)$) incorporated the rotation obtained from the visual odometry and the state transition matrix for propagating the state covariance in the filter is

$$\frac{\partial f}{\partial x} = \begin{bmatrix} q\cos(\phi)\tan(\theta) - r\sin(\phi)\tan(\theta) & \frac{q\sin(\phi)+r\cos(\phi)}{\cos(\theta)^2} \\ -q\sin(\phi) - r\cos(\phi) & 0 \end{bmatrix}$$

where p, q, and r are the instantaneous rotation rates about the camera z, x, and y axis respectively. In the EKF update step, there is only limited theoretical development for obtaining the process noise Q term, because the process is nonlinear, and in practice is often hand tuned for performance.

The update step used the DNN gravity vector to improve the estimate. Accounting for the orientation of the accelerometers according to the Android API the measurement model is

$$h(x,u) = \begin{bmatrix} -\cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \\ -\sin(\theta) \end{bmatrix}$$

and the measurement matrix used for calculating the Kalman gain is

$$\frac{\partial h}{\partial x} = \begin{bmatrix} -\cos(\theta)\cos(\phi) & \sin(\theta)\sin(\phi) \\ -\cos(\theta)\sin(\phi) & -\sin(\theta)\cos(\phi) \\ 0 & -\cos(\theta) \end{bmatrix}$$

.

The measurement noise term R was obtained by calculating the variance of DNN errors after training. After the measurement update, the transformation from the initial camera frame to the current camera frame was reconstructed from the Euler angles for further propagation. The DNN measurements were calculated on every second image used for visual odometry (7.5 Hz) throughout the flights.

B. Results

For comparison, a test was performed using the DNN baseline as the measurement update. In this test the mean gravity vector from the training dataset was used for every Kalman filter measurement with an appropriately large measurement noise covariance. This test demonstrates the result of allowing the filter to know which direction was most likely upward. The top center plot of Fig. 8 shows the filtered result

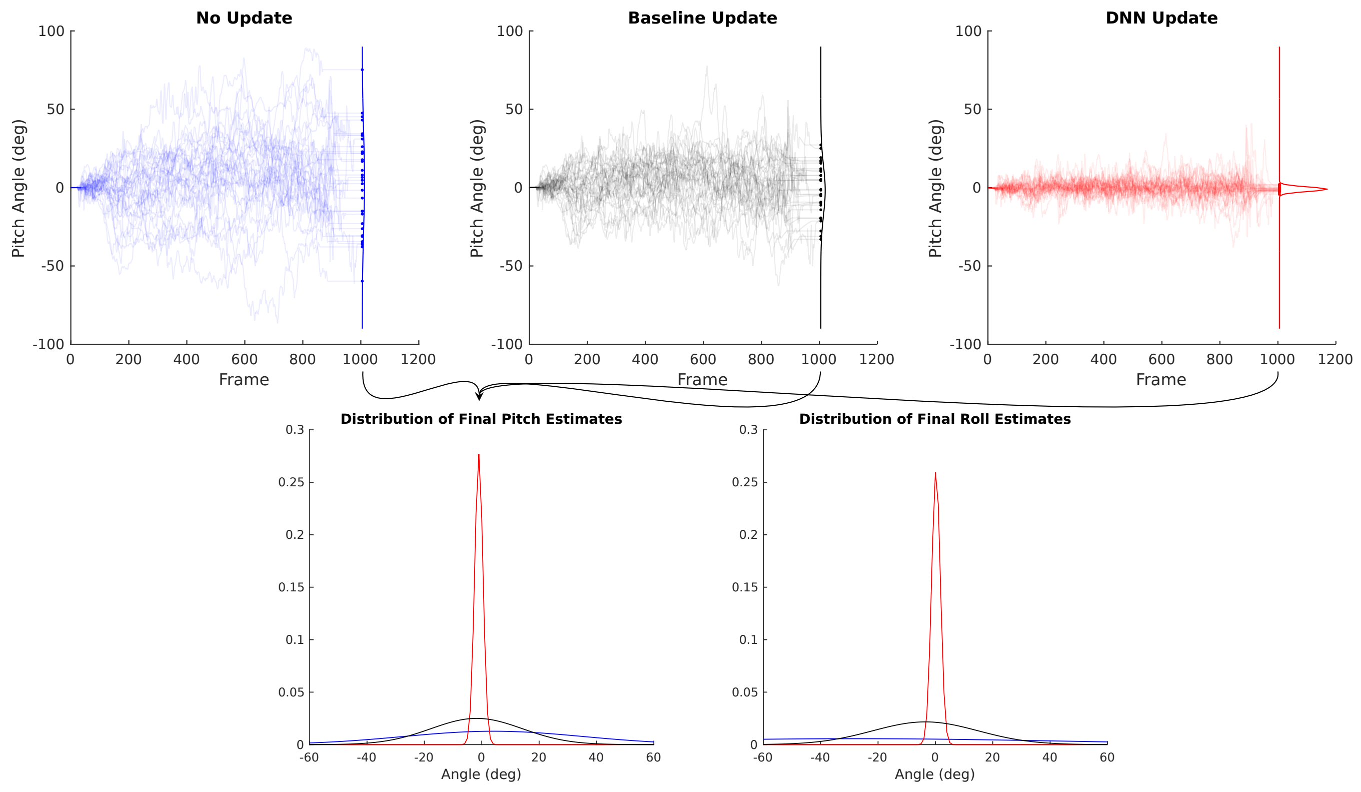

Fig. 8. *Top: Aircraft estimated pitch angles for all 33 flights from visual odometry with no gravity vector update, a baseline update, and a DNN gravity vector update. Bottom: Comparison of the normal distributions approximated from the angles at landing where the true angle is zero. Note the DNN update provided much lower variance comparison to the others. Similar results were observed for the roll angle performance.*

for the pitch and roll angles for all flights. While the pitch and roll angles appear to be bounded they have a large variance both throughout the flight and at landing.

The top right plot in Fig. 8 shows the results of incorporating the DNN's gravity vector detection with the visual odometry for the same test flights. The results show that, as expected, the pitch and roll angles are non-zero during all of the flights (due to flight maneuvering) and return to close to zero when the aircraft landed. This means the DNN is performing as a sensor and has effectively bounded the drift and greatly improved the accuracy of the estimates in the extended Kalman filter. Table I shows the means and standard deviations for the flight attitude angles at landing with no measurement update, the baseline update, and the DNN gravity vector update. The DNN reduces the standard deviation of the langing angles by approximately 12 times from the results using the baseline update.

TABLE I

MEAN AND STANDARD DIVINATION OF ATTITUDE ANGLES AT LANDING.

	No Update		Baseline Update		DNN Update	
	mean	std div	mean	std div	mean	std div
Pitch (deg)	4.0	31.3	-1.8	16.0	**-1.0**	**1.4**
Roll (deg)	-28.1	70.7	-3.5	18.4	**0.2**	**1.5**

V. Conclusion

After collecting a dataset using an Android app, we have designed and trained a DNN to estimate the attitude of the camera from an image. The architecture uses VGG activations from layer conv5.1 and three additional custom layers. Although limitations exist due to the size and quality of the collected dataset, the DNN was able to be trained using TensorFlow and was able to estimate the direction of a gravity vector from visual clues in an image.

Further, we have shown that like people, a DNN can be trained to perceive the orientation of a picture. This can be done to produce an orientation estimate with about 8 degrees of error on average. Given more training data, more accurate attitude measurements, and a completely custom built DNN, we believe that this could be a feasible approach to attitude estimation of a flying vehicle, where the camera provides an inertial attitude measurement for the aircraft state estimation. This work has also shown that a DNN can be used as a sensor in a extended Kalman filter where error in the DNN output is accounted for as measurement noise. This may be a feasible method for applying deep learning to other robotics applications.

References

[1] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual–inertial odometry," *The International Journal of Robotics Re-*

search, vol. 32, no. 6, pp. 690–711, 2013.

[2] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, 2012.

[3] L. Deng, G. E. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 8599–8603, 2013.

[4] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, pp. 1533–1545, Oct. 2014.

[5] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[6] R. Socher, C. Lin, A. Y. Ng, and C. D. Manning, "Parsing natural scenes and natural language with recursive neural networks," in *International Conference on Machine Learning (ICML)*, 2011.

[7] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," *International Conference on Machine Learning (ICML)*, pp. 609–616, 2009.

[8] J. Zhang and C. Zong, "Deep neural networks in machine translation: An overview," *IEEE Intelligent Systems*, vol. 30, no. 5, pp. 16–25, 2015.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3104–3112, 2014.

[10] K. Cho, B. Van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Oct. 2014.

[11] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research (JMLR)*, 2016.

[13] M. Wulfmeier, P. Ondruska, and I. Posner, "Deep inverse reinforcement learning," *arXiv*, vol. 1507.04888, 2015.

[14] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. I. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv*, vol. 1511.03791, 2015.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.

[16] J. Schmidhuber, "Multi-column deep neural networks for image classification," *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3642–3649, 2012.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] T. Zeng, R. Li, R. Mukkamala, J. Ye, and S. Ji, "Deep convolutional neural networks for annotating gene expression patterns in the mouse brain," *BMC Bioinformatics*, vol. 16, p. 147, 2015.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, 2016.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[21] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for UAVs: Current developments and trends," *Journal of Intelligent & Robotic Systems*, pp. 1–28, 2017.

[22] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[23] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part II: Matching, robustness, optimization, and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[24] T. Cornall, G. Egan, and A. Price, "Aircraft attitude estimation from horizon video," *Electronics Letters*, vol. 42, no. 13, pp. 744–745, 2006.

[25] M. Hwangbo and T. Kanade, "Visual-inertial uav attitude estimation using urban scene regularities," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2451–2458, IEEE, 2011.

[26] D. G. Kottas and S. I. Roumeliotis, "Exploiting urban scenes for vision-aided inertial navigation.," in *Robotics: Science and Systems*, 2013.

[27] W. Elloumi, S. Treuillet, and R. Leconge, "Real-time camera orientation estimation based on vanishing point tracking under manhattan world assumption," *Journal of Real-Time Image Processing*, pp. 1–16, 2014.

[28] B. S. Chiel, *GPS-denied multi-agent localization and terrain classification for autonomous paraffin systems*. PhD thesis, Boston University, 2016.

[29] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 2759–2764, IEEE, 2016.

[30] C. Hung, Z. Xu, and S. Sukkarieh, "Feature learning based approach for weed classification using high resolution aerial images from a digital camera mounted on a UAV," *Remote Sensing*, vol. 6, no. 12, pp. 12037–12054, 2014.

[31] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *arXiv preprint arXiv:1603.02199*, 2016.

[32] G. Bradski *Dr. Dobb's Journal of Software Tools*, 2000.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[35] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition.," in *Icml*, vol. 32, pp. 647–655, 2014.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[37] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[38] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[39] C. Tomasi and T. Kanade, "Detection and tracking of point features," 1991.

[40] J. Shi *et al.*, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pp. 593–600, IEEE, 1994.

[41] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," *International journal of computer vision*, vol. 27, no. 2, pp. 161–195, 1998.

[42] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[43] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton University Press, 2012.