10.2514/6.2015-2899

AIAA AVIATION Forum
22-26 June 2015, Dallas, TX
15th AIAA Aviation Technology, Integration, and Operations Conference

Reinforcement Learning with Autonomous Small Unmanned Aerial Vehicles in Cluttered Environments

"After all these years among humans, you still haven't learned to smile." - Star Trek®: The Original Series, Ep. 39

Loc Tran*, Charles Cross†, Gilbert Montague‡, Mark Motter§, James Neilan¶,
Garry Qualls‖, Paul Rothhaar**, Anna Trujillo††, and B Danette Allen‡‡
NASA Langley Research Center, Hampton, Virginia 23681

We present ongoing work in the Autonomy Incubator at NASA Langley Research Center (LaRC) exploring the efficacy of a data set aggregation approach to reinforcement learning for small unmanned aerial vehicle (sUAV) flight in dense and cluttered environments with reactive obstacle avoidance. The goal is to learn an autonomous flight model using training experiences from a human piloting a sUAV around static obstacles. The training approach uses video data from a forward-facing camera that records the human pilot's flight. Various computer vision based features are extracted from the video relating to edge and gradient information. The recorded human-controlled inputs are used to train an autonomous control model that correlates the extracted feature vector to a yaw command. As part of the reinforcement learning approach, the autonomous control model is iteratively updated with feedback from a human agent who corrects undesired model output. This data driven approach to autonomous obstacle avoidance is explored for simulated forest environments furthering autonomous flight under the tree canopy research. This enables flight in previously inaccessible environments which are of interest to NASA researchers in Earth and Atmospheric sciences.

Nomenclature

sUAV	small Unmanned Aerial Vehicle
GPS	Global Positioning System
DARPA	Defense Advanced Research Projects Agency
RGB-D	Red Green Blue - Depth
NASA	National Aeronautics and Space Administration
LaRC	Langley Research Center
SLAM	Simultaneous Localization and Mapping
CAT	Computed Axial Tomography
RC	Radio controlled
ALVINN	Autonomous Land Vehicle in a Neural Network
ROS	Robot Operating System

I. Introduction

The size, stability, and maneuverability of small unmanned aerial vehicles (sUAV) allows them to fly in cluttered and dense environments enabling new missions in NASA Earth and Atmospheric sciences, as well as for the public good in operations such as search and rescue. The unique properties of sUAVs enable them to perform flights and missions that are not otherwise possible with traditional aerial vehicles.

Due to the low cost of entry, research and development around sUAVs have become increasingly prevalent. The industrial sector is investing in potential commercial applications such as autonomous package delivery and overhead power line inspection. Universities have used sUAVs for research projects such as environmental atmospheric data

*Computer Engineer, MS492.
†Software Engineer, Northrop Grumman, Hampton, VA 23861.
‡NASA Student Intern, Baldwin Wallace University, Berea, OH 44017.
§Research Engineer, MS488, AIAA Member.
¶Computer Engineer, MS492.
‖Senior Research Engineer, MS130, AIAA Member.
**Aeronautics Engineer, MS308.
††Senior Research Engineer, MS492, AIAA Member.
‡‡NASA Langley Autonomy Incubator Lead, MS492, AIAA Senior Member.

collection[1], agricultural applications[2], and surveillance[3]. The defense sector is also interested in sUAV applications. For example, DARPA's fast lightweight autonomy (FLA) program is focused on high-speed navigation in cluttered environments[4].

Navigation in global positioning system (GPS) denied and degraded environments is currently an active research topic. GPS signals are not reliable indoors and may also be degraded under heavily forested environments. Malicious agents may also spoof or flood GPS channels. Therefore, relying solely on GPS information for navigation is not a feasible solution for safe, robust, reliable flight in any environment. Thus, it is important to develop navigation techniques that are not dependent on GPS. Further, a problem that GPS cannot solve is obstacle avoidance in unknown cluttered environments. A sense and avoid algorithm is critical to mission success when obstacles are present.

The Autonomy Incubator at NASA Langley Research Center (LaRC) is developing technologies and capabilities focused on autonomous missions using sUAVs[5–10]. In this report, we implement and assess the performance of a recently proposed method for obstacle avoidance in cluttered environments using a monocular forward-facing camera[11].

A. Background

The difference between autonomy and automation is a heavily debated subject. In this paper, we consider autonomous UAVs to be able to sense information about its current state or environment and perform tasks based upon this information without the direct control of a human. While the UAV is self-governing in the autonomous state, humans are not necessarily out of the loop. Humans can play a role in initiating the UAV to enter autonomous mode and may remotely supervise the UAV as it executes its own tasks.

Reinforcement learning is an area in machine learning that focuses on creating software that performs tasks in an environment based upon maximizing a reward function. In the past decade, reinforcement learning has been a very active topic in small unmanned aerial vehicles. A popular example is Ng et al's[12] work implementing an autonomous inverted hover with an RC (radio controlled) helicopter. The same research group extended the work so that the RC helicopter was able to perform an entire aerobatic airshow autonomously[13]. The classical example of autonomous driving is ALVINN (Autonomous Land Vehicle in a Neural Network) which used machine learning to correspond steering control to camera images for autonomous highway driving[14]. In this paper, we implemented a recent publication addressing sUAV flight in a forested environment using a reinforcement learning approach[11]. The approach learns control commands to fly a sUAV through cluttered environments using features extracted from a forward-facing camera with examples given by a human expert.

II. Environment Setup

Simulated experiments were conducted using a combination of the Robot Operating System[15] (ROS) and Gazebo®[16]. ROS is an open source framework that is popularly used for writing robotics software. The collection of libraries and tools in ROS allow for fast development. Gazebo® is a visualization and simulation tool used in conjunction with ROS. We used a simulated sUAV model that was developed in ROS[17]. Using Gazebo®, we created a simulated environment where 1000 tree trunks are placed randomly in a 200×100 meter space as shown in Figure 1.

The vehicle we used for our indoor testing was a Parrot AR.Drone® 2.0. This sUAV provides an integrated package for rapid prototyping. For future work, we will be using a custom sUAV. An open source library, cvdrone[18], was used to control the AR.Drone®. A sparse indoor "forest" was created with small artificial trees.

In the algorithm, the sUAV is presented with training information from an expert navigating a course and learns reactive controls based upon features extracted from the forward-facing camera. The algorithm is described in more detail in Section III. In each experiment, the sUAV is set at a fixed altitude and moved forward at a constant speed. The pilot controls the sUAV with a yaw command to navigate safely through the environment. User control of the sUAV was obtained through a joystick. To limit the controls to just a yaw control, buttons on the joystick were mapped to takeoff, land, and start/stop constant forward movement. The same controller was applied to the to both the simulated and indoor environment. The experiments were conducted using a desktop computer with an Intel® Core™ i7-990X processor.

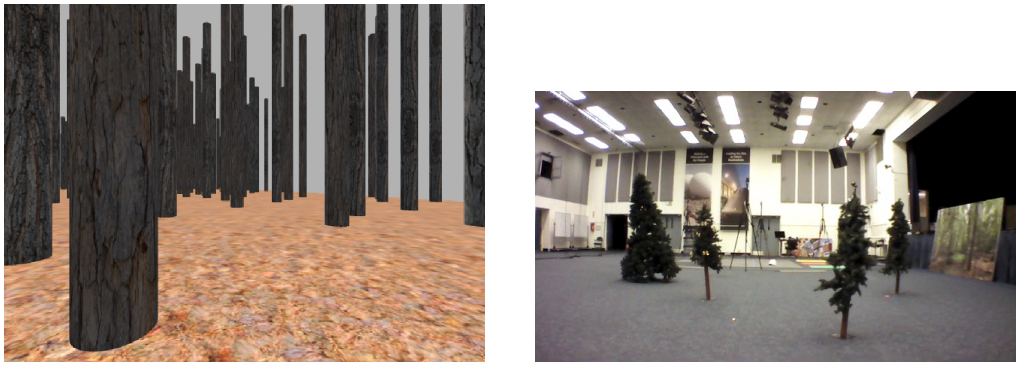

Figure 1. Testing Environments. Left shows simulation in Gazebo. Right shows indoor environment

III. Method

This work is based on a data aggregation method described by Ross et al[11]. As a point of departure, we closely modeled our implementation to the original algorithm but we have made adjustments based on our findings as described later in this document. In this algorithm, we collect features from a forward-facing camera of a sUAV while it is piloted through a forested environment by a human expert. A supervised regression method is applied to correspond the features collected to the expert's commands. Feedback is given by the expert to correct failure points. The feedback is then aggregated with the existing training information and a new model is created. In this section, we will discuss the feature extraction methods, model generation with kernel ridge regression, and aggregation of feedback. A flow chart of the algorithm is shown in Figure 2.

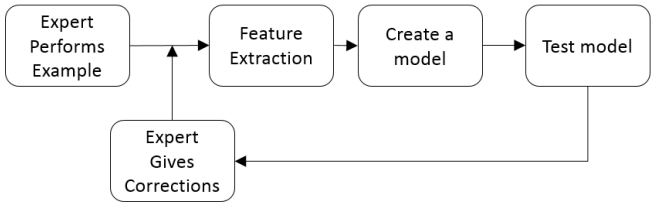

Figure 2. Flowchart of Data Aggregation[11] algorithm

A. Feature Extraction

As a starting point, we implemented three feature extraction methods in a similar manner as described by Ross et al[11]. The features are extracted from a set of sliding windows thus breaking down an image spatially into small regions of interest. This approach provides a coarse localization on the features calculated. In this study, we divided 320×240 images from the forward-facing camera into 7×7 sliding windows with a 50% overlap. The sliding window approach allows features that are generally calculated globally in the image to be localized to fixed regions. The number of sliding windows will offer a trade off between computational complexity and performance. An example of sliding window selection is shown in Figure 3 where the region enclosed by the blue rectangle represents a selected region for a window.

1. Structure Tensor Statistics

Structure tensor statistics provide gradient and corner information about a neighborhood around each pixel in an image. The popular computer vision library, OpenCV[19], uses structure tensor statistics in its goodFeaturesToTrack function. The method is used to find strong corners that are useful in other computer vision algorithms such as optical flow and image registration.

Eigenvectors and eigenvalues are calculated for each pixel neighborhood. Since we are working with 2-D images, there will be two eigenvector/value pairs. Given that the first eigenvalue is larger than the second, the first eigenvector will point in the direction of greatest variation in the selected window and the first eigenvalue can be viewed as a metric for the amount of variation. By definition, the second eigenvector is orthogonal to the first eigenvector and the second eigenvalue similarly provides a value related to the amount of variation in its corresponding eigenvector's direction. A strong response in both eigenvalues may signify a corner at the pixel's location.

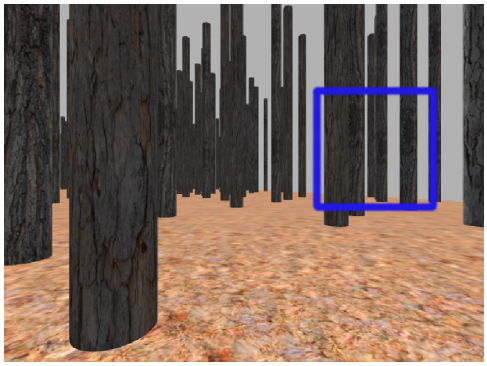

Figure 3. Example sliding window region enclosed by blue rectangle

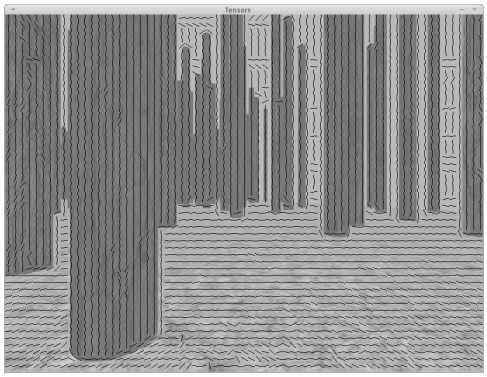

Figure 4. Example gradient information from structure tensor features.

We compute the angle of the first eigenvector to determine the direction of the gradients. A gradient angle is calculated for each pixel in the sliding window. The angles are then accumulated in 15 bins over the range of angle values. Instead of incrementing a counter for each angle, the corresponding eigenvalues are added into the bin. As an example of this binning process, all pixels that have a gradient angle of 0-12 degrees are grouped together. The summation of the eigenvalues for these pixels will be calculated for this bin. The sum of eigenvalues correlates to the total response in an angle's direction. Since we have 15 bins, we end up with 15 tensor features for each sliding window. We chose 15 bins to be consistent with the implementation from Ross et al[11], however, this number should be based upon the size of the neighborhood. For example, a 3×3 neighborhood will have a smaller possible range of angles than the 7×7 neighborhood used in our implementation.

Figure 4 shows a sparse example of the angle information in a simulated environment. The lines in this figure are pointing in the direction of the second eigenvector which are orthogonal to the texture gradient. Due to the difference in texture, the tree trunks impose a different response compared to the ground and background.

2. *Radon*

The Radon transform computes line integrals projected over a set of angles. In image processing, the 2-D image is projected onto 1-D lines at a varying set of angles. The Radon transform is used most prominently in medical imaging particularly in CAT (computed axial tomography) scans. X-ray scans are taken at varying angles around a patient and a reconstruction is created with an inverse Radon transform to create a topological map. In the case of our implementation, it is useful in detecting strong lines in an image. To compute a Radon projection R for a specific angle θ, the following function is calculated:

$$R_\theta(x') = \int f(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)dy'$$

where $f(x, y)$ is the 2-D image. Since the images are discrete, we use a discrete version of the Radon transform:

$$R_\theta(x') = \sum_{y'=0}^{n-1} g(x', y')$$

where $g(x', y') = f(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$.

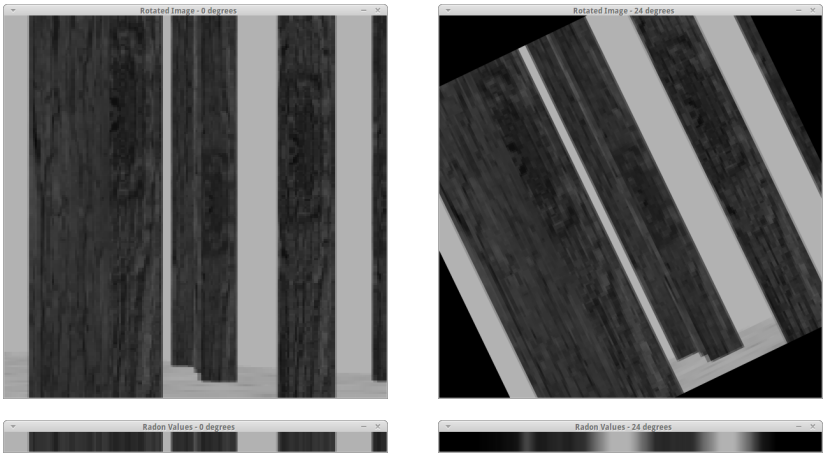

Figure 5. Example Radon transformation. Left and right show a rotation of 0 and 24 degrees respectively. Top row shows the rotation applied to the sliding window selection from Figure 3. Bottom shows the values calculated from the Radon transform.

In this implementation, 15 angles between 0 and 180 degrees are used and the computed transforms are collected in 15 bins over x. For each angle, the largest two bins are saved as features for a total of 30 Radon features per window similar to the implementation by Ross et al[11].

In Figure 5, we show two Radon feature calculations for the window shown in Figure 3. The two top images show the image region rotated by 0 and 24 degrees for the left and right images respectively. The sum of each column of pixels is calculated in the Radon transformation creating a $1 \times n$ vector where n is the width of the window in pixels. The bottom row in Figure 5 shows a scaled representation of the column summation. The Radon transform thus calculates these column summations for a set of angles specified by the implementer. As can be seen by the bottom left image, the tree trunks and open areas are shown as dark and light regions respectively. In the binning for this feature, the columns of each summation vector is split into 15 sections. Our simulated environment consists only of vertical tree trunks so it is evident in the bottom left figure where open regions and tree trunks are present. A more realistic scene may include branches or trunks at varying angles that may be better detected with the Radon transform at angles with a rotation orthogonal to the branch's orientation. The original implementation uses the values of the largest two bins as features. But assuming that branches are generally darker than the rest of the scene, it may be more informative to save bins with a lower intensity rather than the bins with the largest values. We plan on exploring this modification in our future work.

Note that rotations may cause cause data to be missing in some areas of the image window as seen in the black regions of Figure 5 (top right). Currently, the missing data is ignored but we plan on filling these missing regions with pixels outside of the window of interest in our future work.

3. Laws' Masks

Laws' Texture Masks are image processing filters that are used to detect various pixel intensity related features[20]. Laws' Masks are computed by applying a small 2-D convolutional kernel to an image that are used to determine various textures. The 2-D kernels are generated from transpose multiplication of a set of one dimensional convolution kernels. The 1-D kernels used in this study are:
L3 = [1 2 1], E3 = [-1 0 1], and S3 = [-1 2 -1]

Here, L3 would detect a pixel level spike, E3 would detect an edge in the direction of the filter, and S3 would detect a spot. An example computation of a 2-D convolution matrix is L3E3 = L3 $\times$ E3^T. All nine combinations of the 1-D kernels are used; L3L3, L3E3, L3S3, E3L3, E3E3, E3S3, S3L3, S3E3 and S3S3. The 2-D filters would detect variations of levels, edges, and spots in both the vertical and horizontal directions. These masks are applied to each of the sliding windows to produce nine features per window.

4. Kernel Ridge Regression

Classical linear regression minimizes a least squares problem:

$$C(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T x_i)^2$$

In our case, the features extracted from forward-facing video are arranged in a vector x_i while the desired output y_i is the yaw command ranging from $[-1, +1]$. Here, we correspond -1 to a maximum left turn and $+1$ similarly to a

right turn. The learning method would find the values for the weights $\mathbf{w}$. Thus, the regression algorithm is essentially minimizing an error/cost calculation, C. Ridge regression takes the classical model further by adding a regularization penalty to the cost term.

$$C(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T x_i)^2 + \lambda \|f\|$$

Where f is a penalty function. Here, λ is a parameter that controls the tradeoff between minimizing the costs and regularizing the solution. This is often used to prevent over-fitting models. Kernel ridge regression introduces a nonlinear component to this approach using the "kernel trick".

$$f(x) = y'(K + \lambda I)^{-1}k$$

where, $k = (k(x_1, x), \ldots, k(x_n, x))^T$ is the vector of inner products between the set of data y_i and the point x and $K_{ij} = \langle x_i, x_j \rangle$.

Kernel ridge regression maps the inputs into a feature space defined by the kernel. The learning algorithm then finds linear patterns within the new feature space. The kernel used in this experiment is the radial basis function.

B. Integrating Expert Feedback

After an initial model is created, it is applied to the same environment to assess the performance of the model. It is often the case that the model would not be able to perform similarly to the expert input. Imagine a case where an expert pilot training the system flies perfectly and does not get within one meter of a tree during the initial training phase. Small variations in initial conditions, flight variations, or modeling inaccuracies may eventually bring the sUAV to an image view that is dissimilar to the training set of images. In this example, this could be a view of a tree that is less than one meter away. Since the current training does not account for this situation, the model will not respond appropriately and is likely to fail.

In the DAgger (Data Aggregation) algorithm[11], the failure events are used to further train the model. In each failure, the expert pilot would supply a corrected input response. In our experiments, a yaw command was provided while other flight controls were kept constant. The features for that particular image view along with the corrected yaw response is "aggregated" to the training data set. A new model is created with kernel ridge regression using the new information.

In the implementation, integrating feedback presented a few challenges. One approach is to playback the experiments in real-time and allow the expert pilot to give corrected commands via joystick. This proves a challenge for the expert since there is a delay associated with the reaction time of the pilot. Also, the expert's feedback commands are not reflected in the video playback. The expert then tends to give an excessive command to compensate. For example, if the expert sees a moment where the sUAV should turn left but the video feed does not replicate the command, the expert will tend to turn more towards the left. Aggregating these imprecise corrections results in a model with erratic responses. One method to alleviate this is to play back the image sequence at a slower pace. This method was used in the original paper's implementation. In the approach we used, the expert is supplied a frame-by-frame instead of a real-time playback because it allowed for a greater control over the feedback response.

To visualize the feedback, we created a user interface as shown in Figure 6 similar to the implementation by Ross et al[11]. The purple bar on the bottom of the figures show the model's computed response for the yaw commands. The white bar above this shows the commands given by the human expert. In the left image, the model computes a left turn when an obstacle is directly in front of the camera view. In this case, we do not correct the yaw command. In the right image on the other hand, the model computes a left turn but an obstacle is on the left of side of the image. The left turn may have been induced by the tree on the right of the image. This gives an example of when a model produces a response that needs to be updated. In this case, the human expert gives a corrected yaw command as shown in Figure 6 (right).

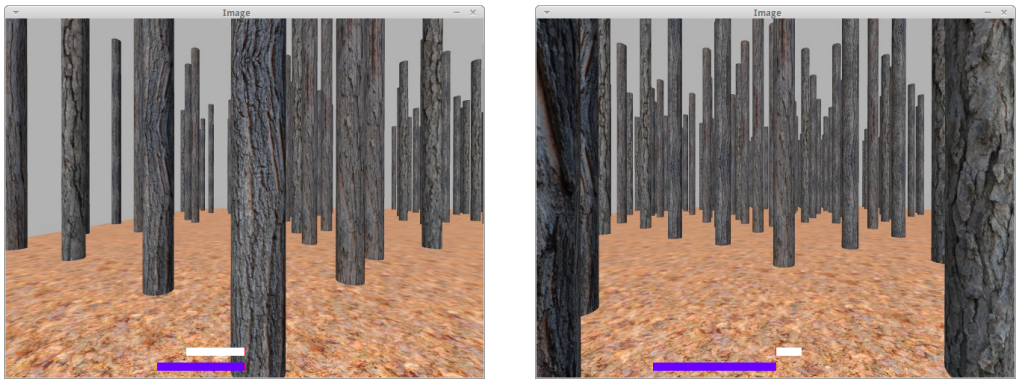

Figure 6. Example frames from expert feedback correction. Top bar shows expert input while bottom bar shows the model's computed command

IV. Results

Under the simulated environment, the sUAV is able to navigate to the end of the simulated forest without colliding with a tree after 5 iterations of training. Initial testing in the indoor environment demonstrated that the system was able to navigate through a sparse randomly placed arrangement of artificial trees. At this time, testing is ongoing in a denser scene.

The computational time is a limiting factor for the proposed algorithm in our current configuration. While calculating the results from the model is very fast, the feature extraction on the image is computationally intensive. In terms of time for a 320×240 image split into 7×7 windows, the average computational time for the Laws Masks features, Radon features, and the structure tensor features were 14ms, 60ms, and 15ms respectively. Altogether, the algorithm runs at about 12 fps. While this frame rate was adequate to achieve autonomous obstacle avoidance in our experiments, a faster frame rate will be required to achieve the speed and agility we need to accomplish our targeted missions. Since the Radon features take about four times as long to compute than the other features, it may be useful in the future work to reduce the number of Radon features calculated and increase the number of sliding windows.

During the indoor testing, there is a noticeable latency for the images from the sUAV's camera to be relayed to the base station. The front camera video is compressed using H.264 video encoding and sent via wifi to the base station. The sending and decoding of the video creates the latency. The lag associated with this was measured to be approximately 0.635 seconds and is enough to create difficulty for the human pilot to avoid obstacles.

The field of view of the AR.Drone® 2.0 front-facing camera is 92 degrees. In Fig 7, the left image shows the camera feed from the forward-facing camera of the AR.Drone®. At this point the AR.Drone® is approximately one foot away from the tree. The right image shows a third person perspective of the sUAV at the same point. The limited field of view poses a challenge when avoiding obstacles using vision. One example of this happening is if the sUAV is directed to go left to avoid one obstacle only to turn into another one that is not in its field of view.

Figure 7. Field of view example. Left shows first person view from sUAV. Right shows third person view.

V. Conclusion and Future Work

We reviewed a data driven reinforcement learning approach to reactive obstacle avoidance for small unmanned aerial vehicles. We found that given enough training information, a sUAV could autonomously avoid obstacles that were present in the training phase. Some of the challenges that arise come from a narrow field of view, computational costs, and latency. While vision based feature extraction methods are effective, it is difficult to quickly handle the large amount of information contained in video streams. Since our goal is ultimately to move to on-board processing,

we plan to make adjustments to the feature selection methods in our future work. This will include exploring faster feature extraction methods or optimizing the current methods. Also, the work will be ported to a different vehicle that is tailored to the needs of the Autonomy Incubator and will have a wider field of view front camera, heavier payload capacity, and ability for faster more agile flight. The improved capabilities will allow additional sensors to be incorporated thus providing useful information. For example, an active sensor such as a line scanning lidar would give fast depth information at low computational costs. With these improvements, we hope to achieve safe, robust, and reliable autonomous sUAV flight in cluttered and dense environments which will enable new missions in Earth and Atmospheric sciences.

Acknowledgments

This work is in support of NASA Langley's Autonomy Incubator and NASAs Convergent Aeronautics Solutions (CAS) Vertical Lift Hybrid Autonomy (VLHA) project.

References

[1]de Boisblanc, I., Dodbele, N., Kussmann, L., Mukherji, R., Chestnut, D., Phelps, S., Lewin, G., and de Wekker, S., "Designing a hexacopter for the collection of atmospheric flow data," *Systems and Information Engineering Design Symposium (SIEDS), 2014*, April 2014, pp. 147–152.

[2]Tokekar, P., Vander Hook, J., Mulla, D., and Isler, V., "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 5321–5326.

[3]Semsch, E., Jakob, M., Pavlicek, D., and Pechoucek, M., "Autonomous UAV Surveillance in Complex Urban Environments," *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on*, Vol. 2, Sept 2009, pp. 82–85.

[4]"Zoom in, Zoom out: Speedy, Agile UAVs Envisioned for Troops in Urban Missions," http://www.darpa.mil/NewsEvents/Releases/2014/12/22.aspx, Dec. 2014, [Online; posted 22-December-2014].

[5]Allen, B. D., Cross, C., Motter, M., Neilan, J., Qualls, G., Rothhaar, P., and Trujillo, A., "Towards Safe, Robust Autonomous Operations - Who's "got the bridge?"," Tech. rep., AIAA 2015 Special Invited Session, 2015.

[6]Neilan, J., Cross, C., Motter, M., Qualls, G., Rothhaar, P., Trujillo, A., and Allen, B. D., "Using Ensembles of Classifiers for intuitive decision making; Progress towards a better Cmdr. Data. - "What it needs in order to evolve... is a human quality. Our capacity to leap beyond logic."," Tech. rep., AIAA 2015 Special Invited Session, 2015.

[7]Rothhaar, P., Cross, C., Neilan, J., Qualls, G., Trujillo, A., and Allen, B. D., "Safe flight zone avionics architecture - Return to thine own solar system immediately."," Tech. rep., AIAA 2015 Special Invited Session, 2015.

[8]Cross, C., Neilan, J., Qualls, G., Rothhaar, P., Trujillo, A., and Allen, B. D., "An Open, Distributed Software Architecture for UxS Operations - "It's difficult to work in groups when you're omnipotent," So don't be," Tech. rep., AIAA 2015 Special Invited Session, 2015.

[9]Trujillo, A., Cross, C., Motter, M., Neilan, J., Qualls, G., Rothhaar, P., and Allen, B. D., "Collaborating with Autonomous Agents - I'm a Doctor, Jim. Not an Engineer!," Tech. rep., AIAA 2015 Special Invited Session, 2015.

[10]Qualls, G., Cross, C., Motter, M., Neilan, J., Rothhaar, P., Trujillo, A., and Allen, B. D., "Test and Evaluation in Complex Environments - Operating in Strange New Worlds and Measuring Success," Tech. rep., AIAA 2015 Special Invited Session, 2015.

[11]Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., and Hebert, M., "Learning Monocular Reactive UAV Control in Cluttered Natural Environments," *CoRR*, Vol. abs/1211.1690, 2012.

[12]Ng, A., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E., "Autonomous Inverted Helicopter Flight via Reinforcement Learning," *Experimental Robotics IX*, edited by J. Ang, MarceloH. and O. Khatib, Vol. 21 of *Springer Tracts in Advanced Robotics*, Springer Berlin Heidelberg, 2006, pp. 363–372.

[13]Abbeel, P., Coates, A., and Ng, A. Y., "Autonomous Helicopter Aerobatics through Apprenticeship Learning," *The International Journal of Robotics Research*, 2010.

[14]Pomerleau, D., "ALVINN: An Autonomous Land Vehicle In a Neural Network," *Advances in Neural Information Processing Systems 1*, edited by D. Touretzky, Morgan Kaufmann, 1989.

[15]"ROS," http://www.ros.org/, Accessed: 2015-05-01.

[16]"Gazebo," http://gazebosim.org/, Accessed: 2015-05-01.

[17]Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., and von Stryk, O., "Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo," *Simulation, Modeling, and Programming for Autonomous Robots*, edited by I. Noda, N. Ando, D. Brugali, and J. Kuffner, Vol. 7628 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2012, pp. 400–411.

[18]"cvdrone," https://github.com/puku0x/cvdrone, Accessed: 2015-05-01.

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

[20]Laws, K. I., *Textured image segmentation*, Ph.D. thesis, University of Southern California, 1980.