Proceedings of the International Conference on Machine Vision and Machine Learning
Prague, Czech Republic, August 14-15, 2014
Paper No. 64

Sky and Ground Detection Using Convolutional Neural Networks

Rytis Verbickas, Anthony Whitehead
Carleton University
1125 Colonel By Dr, Ottawa, Canada
rverbick@connect.carleton.ca; anthony.whitehead@carleton.ca

Abstract - A critical first step in navigation of unmanned aerial vehicles is the detection of the horizon line. This information can be used for adjusting flight parameters, attitude estimation as well as obstacle detection and avoidance. In this paper, a fast and robust technique for precise detection of the horizon is presented. Our approach is to apply convolutional neural networks to the task, training them to detect the sky and ground regions as well as the horizon line in flight videos. Thorough experiments using large datasets illustrate the significance and accuracy of this technique for various types of terrain as well as seasonal conditions.

Keywords**:** Neural, Convolutional, Horizon, UAV, Segmentation.

1. Introduction

As the applications of automated and unmanned systems have grown in recent years, the need for precise and robust guidance and navigation systems has increased as well. Automated computer vision systems aim to take over the role of the eyes and brain of an operator for controlling an automated vehicle. Unmanned aerial vehicles (UAVs) are a particular example of such automated vehicles with numerous applications ranging from traffic surveillance, monitoring of forests in fire situations, transportation, remote sensing with military applications and low altitude magnetic sensing (Staznicky et. al. 2010, Barnard 2008, Park et. al. 2005).

UAV navigation, due to its sensitive nature and high cost in case of collision or failure, has been subject to extensive research (Jian and Xiao-min, 2011). In order to simplify the problem at hand, image segmentation can be significantly beneficial for UAV obstacle detection and navigation systems as it allows the subsequent processing to be limited to subsections of the image. Further analysis can focus on the specific region of interest allowing for an increase in speed and possibly accuracy.

In this paper, the problem of sky and ground segmentation for aerial imagery is studied. A novel method is presented based on using convolutional neural networks (CNNs) to classify whether image pixels are sky or ground**.** This approach does not require estimating the horizon as a continuous or piece-wise line and produces the exact boundary between the sky and the ground regions. This paper is organized as follows: Sky and ground segmentation as well as horizon detection literature is presented in Section 2. In Section 3, we introduce CNNs for segmentation while constructing a large ground truth dataset, which we make publicly available, for training CNNs and perform a thorough evaluation of competing machine learning techniques for the task. Section 4 describes our experimental results and is followed by Section 5 where we conclude with a discussion of results and a statement of future work.

2. Related Work

A number of approaches have been developed in the literature for performing horizon detection using cameras mounted to varying forms of aircraft such as UAVs and micro air vehicles (MAVs). In the case of methods which focus on detecting the horizon boundary, we find a wide range of approaches which either make the assumption that the horizon boundary can be represented as a straight line as seen in (Dusha et. al. 2007, Zafarifar and Weda 2008, McGee et. al. 2005, Bao et. al. 2005, Ettinger et. al. 2002) or specify the exact division between the sky and ground as seen in (Shen et. al. 2013, and Boroujeni et.

al. 2012). In the case where the horizon is flat the former methods are sufficient, however when faced with highly varying terrain (for example at low altitudes) the latter methods are more appropriate. Generally ground is referred to as anything which is below the sky region of the image including water, hills, trees and artificial objects.

For methods which assume the horizon can be represented as a line, we predominantly see the Hough Transform (Duda and Hart, 1972) as a feature extraction technique, specifically in (Dusha et. al. 2007, Zafarifar and Weda 2008, McGee et. al. 2005, Shen et. al. 2013) which is used in conjunction with a separate processing step such as edge detection Zafarifar and Weda 2008, McGee et. al. 2005, Bao et. al. 2005, Shen et. al. 2013). The straight line horizon boundary assumption is valid for a specific range of altitudes. When altitude is too high the curve of the earth will produce a curved horizon line. When the altitude is too low, especially in the case of a terrain following aircraft such as we require, obstacles and hills can also produce a horizon which is not a straight line. The approaches mentioned so far rely on hand crafted processing algorithms in order to narrow down the search region to the horizon boundary which is brittle and requires careful hand tuning.

In contrast to the approach which focuses on finding only the horizon boundary, methods which classify all pixels as belonging to either sky or ground can also be found McGee et. al. 2005, Thurrowgood et. al. 2009, Shinzato et. al. 2012, Fefilatyev et al. 2006, Minwalla et. al. 2011, de Croon et. al. 2011, Todorovic et. al. 2003) and typically use machine learning. These approaches don't make any simplifying assumptions on flight altitude or horizon shape. The result is a "sky-mask" which allows one to selectively pick the sky or ground regions from the original input image. An example is shown in Fig. 1 with the corresponding input image. In (de Croon et. al. 2011), the authors considered sky segmentation for obstacle detection, using a decision tree with a number of input features; these were a number of statistics such as mean, variance and entropy computed over image patches centered at all pixels. The work in (de Croon et. al. 2011), particularly the features which were used, were based in part on the work of (Fefilatyev et al. 2006) who compared the relative performance of support vector machines (SVMs), decision trees (DTs) and Naïve Bayes classifiers for sky-mask detection. Unfortunately this evaluation was small with only 2 groups of 10 images with the ground portion of all images consisting of water, making it difficult to infer generalizations of the performance of the tested classifiers.

Fig. 1. Example RGB input image and its corresponding sky-mask.

The method proposed in this paper uses a CNN to perform the sky-mask detection. Our method makes no assumptions on flight altitude, which means it does not expect the horizon to be modelled as a straight or even piece-wise line along the image. It also avoids the tediousness of a handcrafted algorithm in order to extract edge or other features and to recognize the horizon using the Hough Transform. The CNN instead learns the feature extraction filters necessary to perform the sky and ground extraction. We perform all evaluations on 15 ground truth datasets containing a total of 13,687 images, constructed specifically for this purpose. To our knowledge the application of CNNs to the detection of sky and ground regions in an image has not been addressed as well as the construction of a large ground truth dataset specifically for the task for sky and ground detection.

3. Machine Learning for Sky-mask Extraction

The approach taken by horizon detection algorithms can broadly be broken down into 2 categories; sky-mask and horizon-mask. The first is a horizon mask or horizon line which focuses specifically on the separating region between sky and ground, either through a horizon line or a mask of the horizon region. The second, which we focus on in this paper, generates a mask of image pixels which are either sky or ground allowing each region to be detected separately. In this paper we refer to a sky-mask image as a binary image where white pixels classify sky and black pixels classify ground, with ground including lakes, trees and any other non-sky image regions.

3.1 Dataset Generation

A ground truth dataset of images and their corresponding sky-masks was constructed using video data gathered from a number of flights from 3 different locations (shown in

Fig. 2). Lighting conditions varied from early afternoon to late evening, including summer and winter scenery as well as varying terrain (lakes, trees, hills, etc.). Due to the low speeds and slow manoeuvring of the aircraft in the video sequences, all videos were subsampled at 2 frames/sec; most flights were limited in range. The first 6 sequences (ARN) were conducted in the summer time at altitudes up to 300 feet, while the following 8 (OR) were also conducted in the summer up to a similar altitude and containing a large amount of ground features including water. Finally, the last sequence (SGL) was conducted during the winter time at low altitude over hilly terrain. The images in this sequence are interlaced frames, which caused frame combining to occur when the aircraft encountered strong wind gusts (Fig. 3).

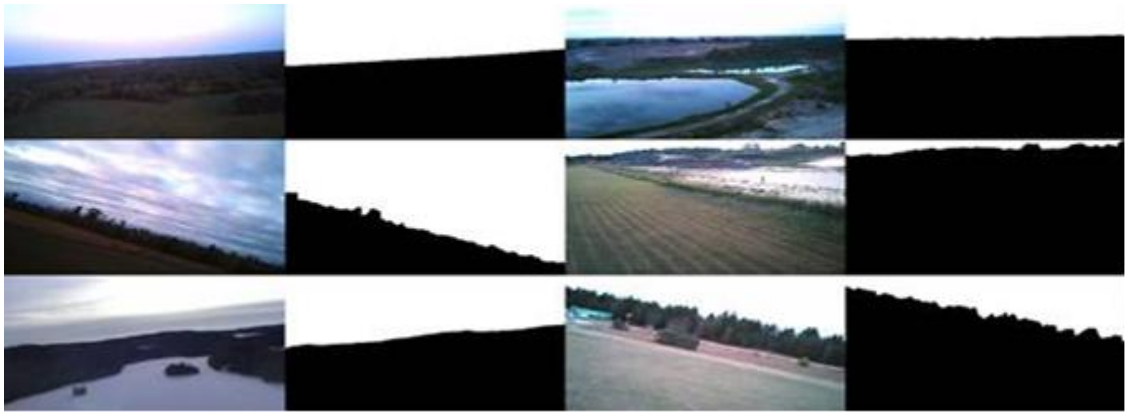

Fig. 2. Example Input and Sky-mask image pairs (Starting top left, top to bottom: Arn datasets 1, 3. SGL dataset 1. OR datasets 1, 6 & 7).

Fig. 3. Frame combining in the SGL dataset introduced due to rapid motion.

A summary of the total number of frames in each sequence is outlined in Table 1. Ground truth data for these datasets was generated using a combination of an existing horizon detection method not based on machine learning (Boroujeni et. al. 2012). The low accuracy of this method for certain datasets was supplemented by considerable (in some sequences over 50% of images) hand labelling of ground truth pixels (sky or ground). The input and ground truth images are 628x426 for the ARN and OR datasets and are 698x426 for the SGL.

Table 1. Tabulation of ground truth datasets
(Rows correspond to a flight location; columns are the flight number for the location).

	1	2	3	4	5	6	7	8
ARN	567	454	235	796	1207	698	/	/
OR	363	460	376	530	4254	1427	678	344
SGL	1427	/	/	/	/	/	/	/

3.2 Convolutional Neural Networks Applied to Sky-mask Detection

CNNs are a form of multi-layer neural network particularly adept at visual pattern recognition (Neubauer 1998, LeCun et. al. 1998). The main difference compared to regular multi-layer neural networks is that CNN's employ weight sharing, where neurons are arranged in groups at a network layer (feature maps) and all neurons in a feature map apply the same set of weights (a kernel) to respective regions (receptive fields) in the previous layer; this is a characteristic operation of a convolutional layer in the network. Further to this, alternating network layers also reduce the spatial resolution of the previous layer and are known as pooling layers (similar to image pyramids). By arranging the network in an alternating convolution and pooling pattern, the networks obtain a degree of invariance to image feature translations and rotations. The output of a CNN is typically a fully connected classification layer containing one output neuron for each output class. We instead focus on using a convolutional layer as the output layer of the network.

This configuration has the advantage of having a significant weight savings over a fully connected output layer. For example the network in Fig. 4 has 20 feature maps, each 33x33, at layer S2 followed by a 29x29 output layer. In the fully connected case, the number of weights (including biases) at the output layer alone would be 18,317,821. When changed to a single 29x29 convolutional feature map (with a 5x5 kernel) the weight number is reduced to 501. This savings is particularly attractive in more memory restricted architectures such as an FPGA.Since CNN's are a modified variant of multi-layer neural networks, they can be trained with gradient descent using the well-known back-propagation algorithm, with a small modification due to weight sharing. In our work we use the common squared error measure for the network loss function and also follow the work in (LeCun et. al. 1998) in computing a diagonal Gauss-Newton approximation to the Hessian matrix in order to compute the optimal learning rates for gradient descent. The interested reader is referred to (LeCun et. al. 1998) for details, including the information on the type of activation function which was used at all layers (hyperbolic tangent). Finally, we consider the input image as having one or more input channels in some color space, while output images are binary.

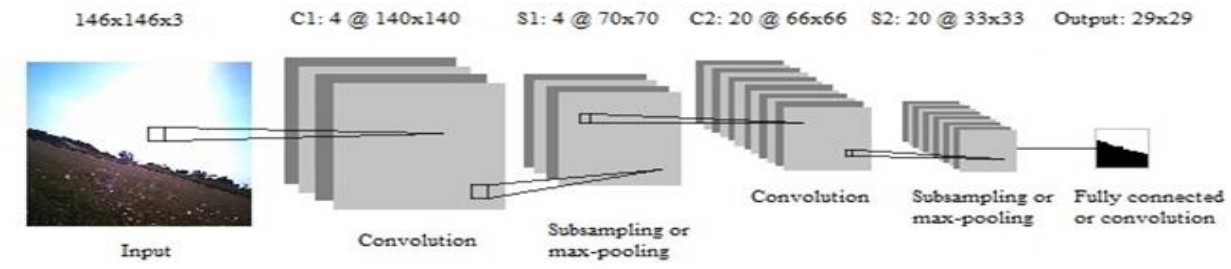

Fig. 4. Convolutional Neural Network Layers.

Both input images and target output images are remapped so that pixels have a range of [-1, 1] with the following mapping (where x_{in} is the pixel value in the image and x_{out} is the pixel value used in the network):

$$x_{out} = \frac{2}{255} x_{in} - 1 \quad (1)$$

3.3 Competing Approaches

The work in (Fefilatyev et al. 2006) outlines a number of metrics which are computed over a window around a pixel in order to generate a feature vector representation of the pixel for training. These metrics include mean, variance and entropy among others (21 in total which includes the original red, blue and green components of the pixel). The authors in (Fefilatyev et al. 2006) also evaluate a number of classifiers, among them being SVMs and DTs; an example we follow.

However the small image set size used by the authors limits their evaluation and in order to make training with SVM's and DT's on the entire dataset feasible, both input and ground truth images needed to be down-sampled. A resolution of 54x54 was chosen with window sizes of 3x3 and 5x5 for collecting the feature vector for each pixel. We report results for 3x3 windows due to the lack of noticeable performance difference compared to 5x5 windows. The MATLAB 2012b packages for training SVMs and DTs was used with default settings with the exception of the split criterion for decision trees which was set to deviance and a MinLeaf value of 10,000.

4. Experiments

4.1 Evaluation

We use the squared error as our metric for driving network training. To perform network evaluation after training, including testing, we use a confusion matrix. Under this approach, any sky region in a target image is labelled the positive class while ground regions become the negative class. This allows for network classification accuracy for a single image to be computed from the diagonal entries of the confusion matrix, while false positive and false negative rates can indicate where the network is struggling. The average accuracy over a training or test set can also be computed simply summing respective terms in multiple matrices together.

A framework was implemented which allows the construction of arbitrary CNN's as well as their training and evaluation process to run on a CUDA GPU. CUDA is a parallel computing architecture (Web-1) particularly suited for this application and allowed a large numbers of experiments to be run. SVM's and DT's were trained on the same machine but using the SVM and DT packages in MATLAB. All tests were run on an Intel i7-2600 machine having 12GB of RAM and an NVidia GTX 460 video card with 1GB of memory and compute capability 2.x. Note that although a graphics card was used for evaluations, so that a large number of experiments could be run, CNN's can be easily deployed on low-cost, low-power platforms such as FPGA's which is our eventual goal.

4.2 Experimental Results

For our evaluation we investigate a number of parameter settings for training CNNs including training duration, network depth and feature map count. We begin by investigating 3 layer networks having a single convolutional and max-pooling layer and a final convolutional output layer with 1 feature map. The inputs to this network are 246x246, with 7x7 kernels at the first convolutional layer, 2x2 at the max-pooling layer and 5x5 kernels at the output convolutional layer. The resulting output from the network is 116x116. Interestingly, subsampling layers didn't work when attempting to train the network to generate sky-mask images which is why max-pooling layers were used, although it is not immediately clear why. For our experiments we use a random half of the training set at the beginning of each epoch to compute the diagonal Hessian approximation; the approach is described in (LeCun et. al. 1998). Learning rate selection was found to have little effect on overall training performance regardless of the number of

layers or feature maps in the network; a rate of $0.00001(0.9)^e$ was used for all experiments where e is the epoch index. Increasing the rate by an order of magnitude or down by 3 orders of magnitude had little to no effect on convergence which appears to be an effect of not having a fully connected output layer.

Table 2. Accuracy results training a 3 layer CNN with 246x246 inputs and 116x116 outputs, 5 epochs of training and 8 feature maps at the first convolutional layer (left-most column is the training dataset).

	Arn 1	Arn 3	OR 1	OR 6	OR 7	SGL
Arn1	91.77	93.73	91.35	96.41	92.66	95.02
Arn3	89.65	93.94	91.53	95.99	91.20	94.16
OR 1	90.56	94.78	91.80	96.32	91.59	94.86
OR 6	91.98	94.87	92.58	97.32	93.45	95.26
OR 7	92.97	93.51	92.09	96.59	94.70	93.90
SGL	84.85	88.70	80.05	88.74	88.24	96.71

Tests looked at the effects of early stopping and varying the number of feature maps. In the case of early stopping (5 epochs instead of 20) little affect was noticed between a 3 layer network with 8 or 16 feature maps. The results are summarized in for 5 epoch networks (for a subset of all datasets). Note that the summary statistics are reported are over all datasets while tables report performance for representative datasets due to space considerations. The average difference in accuracy (accuracy of a network with 16 feature maps minus accuracy with 8 feature maps) across all datasets was only 0.16% indicating a very small advantage to longer training sessions and also that over-fitting does not appear to be a problem. If we take into account only the accuracy difference when training on a dataset at one location and testing on a dataset at another location we obtain an average difference of 0.08%, confirming the lack of over-fitting occurring with longer training sessions. This could be explained by the small amount of shared weights in the network producing a highly varying error surface in addition to camera noise, vibration and rapid illumination changes.

In the case of training for 5 or 20 epochs with 16 feature maps at the first convolutional layer we see a more pronounced average difference in accuracy of 0.42% but in favour of 5 epoch networks. We also see that the larger difference of 0.39% is in testing on a dataset from a different location. When looking at the difference between networks with 8 or 16 feature maps, and taking networks resulting from 5 epochs of training, we see an average difference of only 0.13% in favour of 16 feature maps. This difference is smaller for 20 epoch networks where the average difference is only 0.06% (in favour of 16 feature maps) again demonstrating that more feature maps is only very marginally better.

Turning our attention to evaluating the performance difference between different classifiers we use a network with 246x246 inputs again however we add 2 extra layers (1 convolution with 12 feature maps and 1 max pooling layer) in addition to the first 2 layers (which both have 8 feature maps). The network has a convolutional output layer which produces a network output layer with 54x54 outputs; the same as the image sizes we train our SVM and DT networks with. The only quirk is due to the fact that a window around a pixel is required to generate a feature vector, to compare between CNNs and the other classifiers we must take into account the reduced image size. Thus for 3x3 sampling windows this size becomes 52x52 and 50x50 for 5x5 windows. The respective training results for each type of network are summarized in Table 3, Table 4 and Table 5. We observe that CNNs, on average difference, outperform SVMs by 1.27% and DTs by 2.25%. When considering testing on a dataset at a different location we see a larger average difference of 4.86% in favour of CNNs indicating that SVMs do outperform CNNs when testing on datasets at the same location. In the case of DTs, CNNs outperform with a difference of 0.25% indicating they outperform DTs in both cases.

In general we see that performance is quite close across different datasets, however we do observe that datasets where the lighting conditions are good (middle of a sunny day), classification performance is generally good regardless of whether a network was trained on a dataset with low or high levels of illumination.

Table 3. Accuracy results training a 5 layer CNN classifier on 246x246 inputs and 54x54 network outputs, 5 epochs of training, 8 feature maps at the first convolutional layer and 12 at the second.

	ARN 1	ARN 3	OR 1	OR 6	OR 7	SGL
ARN 1	93.89	93.50	91.98	96.36	93.76	94.90
ARN 3	84.91	90.77	88.74	93.28	89.57	75.52
OR 1	91.06	92.51	93.56	95.13	95.78	92.37
OR 6	91.90	94.06	92.29	97.16	95.10	95.33
OR 7	92.68	90.91	91.27	96.86	97.07	94.25
SGL	82.80	86.76	74.96	82.78	87.35	97.26

Table 4. Accuracy results training a decision tree classifier on 54x54 images using 3x3 sample windows.

	Arn 1	Arn 3	OR 1	OR 6	OR 7	SGL
Arn 1	99.62	77.88	83.01	94.57	89.74	94.64
Arn 3	90.04	92.30	89.40	95.04	90.21	91.01
OR 1	88.82	92.03	89.74	93.28	92.17	96.47
OR 6	90.93	83.64	84.61	96.72	90.90	98.04
OR 7	89.13	84.27	86.91	94.71	93.82	97.61
SGL	84.32	83.78	80.55	88.67	89.02	99.79

Table 5. Accuracy results training a SVM classifier on 54x54 images using 3x3 sample windows.

	Arn 1	Arn 3	OR 1	OR 6	OR 7	SGL
Arn 1	98.48	81.31	86.81	95.50	92.28	97.29
Arn 3	91.34	98.43	92.65	95.86	89.90	98.34
OR 1	92.54	89.29	97.56	92.20	86.50	95.88
OR 6	93.06	84.77	83.85	98.89	91.81	97.87
OR 7	90.02	86.01	89.58	94.37	98.14	95.81
SGL	83.32	84.78	82.32	81.22	91.04	98.98

For example dataset OR 6 has excellent levels of illumination with networks from all other datasets performing well. Where all classifiers struggle is distinguishing lakes within the ground region and the sky. If we look specifically at the network error rate we see that the false positive rate is typically twice the false negative rate as a result. We see that, for CNNs, training on the SGL dataset does have an impact for certain datasets, in particular OR1 (and also on OR 2, 3 and 4). This can be explained by the relatively low lighting levels in this dataset and it seems more importantly that the entire dataset is over snowy terrain. However we don't observe the reverse when training on low illumination images and testing on the SGL dataset. In the case of dataset Arn 3, we do notice on average a drop-off in performance for CNN's trained on this dataset. This may be caused by Arn 3 being the smallest dataset, having been flown in the lowest lighting conditions and over the most limited flight path. We also note an oddity with SVM training on datasets OR4 and OR8 and testing on the SGL dataset where it was observed that the testing accuracy dropped to 35.86% and 36.20% respectively. Similar oddities were observed using 5x5 windows as well but when training on the ARN 3 dataset and testing on the SGL dataset. This oddity occurred every more frequently when training linear kernel SVM's.

An important performance test is to check whether the filters the network has learned are robust to handle unusual aircraft attitude. The typical bank angles observed in the ground truth image is up to a maximum of approximately 60 degrees of bank angle. We therefore take every image in our ground truth dataset, flip it upside down and apply them as new test sets. When we do this we notice that performance

drops on average only by 0.51% and no more than 3% for any network. This result is encouraging for unusual aircraft attitudes as it indicates that aircraft orientation has little effect on classification performance and that the learned filters are quite robust. Indeed generating additional data through rotations and translations of the original images generally produced no performance improvement.

Looking at the output of feature maps within intermediate convolutional layers gives insight into how the final sky-mask is produced. In Figure 5, we show each color channel of a YUV image (the input layer) followed by the activations of each feature map at the first convolutional layer. Finally the activations at the second convolutional layer (the output layer) are shown with the target output shown alongside. The filters learned at the first convolutional layer produce feature maps which have already detected large portions of the desired sky-mask. One of the feature maps (3rd from the top) is directly detecting the horizon itself, instead of sky and ground.

In terms of training time, it is difficult to make comparisons since CNNs were trained using a GPU. However if we consider that the time taken to train a 3 layer 246x246 input to 116x116 output CNN on the largest dataset (OR 5) we see a training time of 1.5 hours for 5 epochs including all overhead (such as learning rate computations).

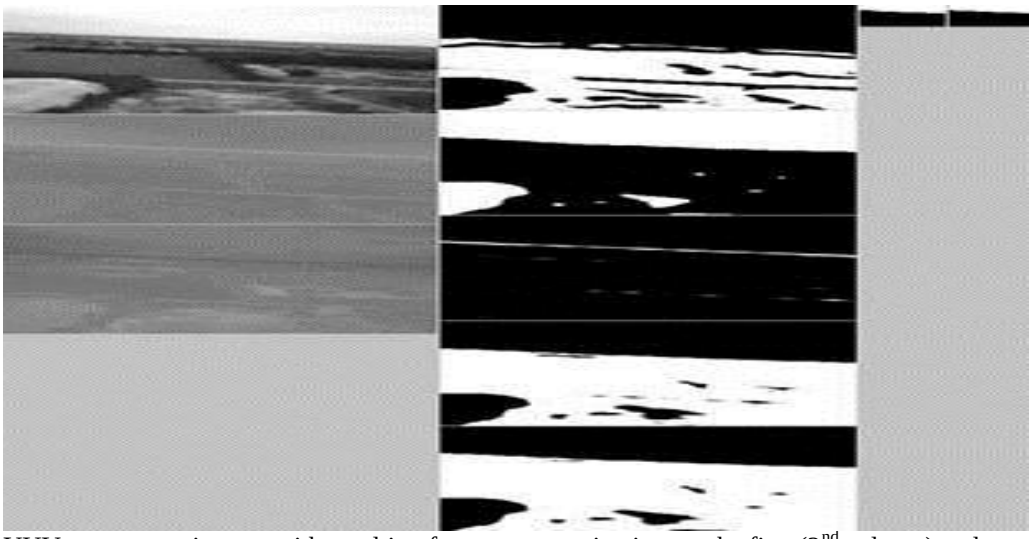

Fig. 5. YUV component images with resulting feature map activations at the first (2nd column) and second (3rd column) convolutional layers as well as the target output (4th column). Not shown are the outputs of the down-sampling layers.

5. Conclusions

This paper presents a novel approach to horizon detection using CNNs. From our extensive experiments we observe that CNNs on average outperform competing classifiers for the task at hand. However an added benefit with CNNs for this task is their simplified training process including resilience to learning rate variation, small required number of feature maps (increasing which shows no performance boost) and a shallow number of layers producing a small number of learnable parameters. This reduces computational requirements for their deployment. Even for larger output sizes (116x116) these benefits allow one to train CNNs on many datasets with potentially large numbers of images with little to no change in performance compared to networks with smaller output sizes; something which is not possible with DTs or SVMs without significant subsampling or another method to reduce the amount of training data. This makes CNNs especially suited for the task of horizon detection. Future work involves applying post-processing to improve performance as well as training a CNN to perform horizon-mask classification (in case only a horizon line is desired instead of a full sky-mask). As mentioned the dataset used in our evaluation will be made publicly available.

References

Barnard J. “The Use of Unmanned Aircraft In Oil, Gas And Mineral E+P Activities." In *Society of Exploration Geophysicists*, page 1132. Las Vegas, Nevada (2008).

Bao G., Xiong S. and Zhou Z. "Vision-based horizon extraction for micro air vehicle flight control." *Instrumentation and Measurement, IEEE Transactions on* 54, no. 3 (2005): 1067-1072.

Boroujeni N., Etemad A. and Whitehead A. "Robust Horizon Detection using Segmentation for UAV Applications." In *Computer and Robot Vision (CRV), 2012 Ninth Conference on*, pp. 346-352. 2012.

de Croon G., Wagter D., Remes B. and Ruijsink R. "Sky Segmentation Approach to obstacle avoidance." In *Aerospace Conference, 2011 IEEE*, pp. 1-16. 2011.

Duda R. and Hart P. "Use of the Hough transformation to detect lines and curves in pictures." *Communications of the ACM* 15, no. 1 (1972): 11-15.

Dusha D., Boles W. and Walker R.. "Attitude estimation for a fixed-wing aircraft using horizon detection and optical flow." In *Digital Image Computing Techniques and Applications, 9th Biennial Conference of the Australian Pattern Recognition Society on*, pp. 485-492. 2007.

Ettinger S., Nechyba M., Ifju P., and Waszak M. "Vision-guided flight stability and control for micro air vehicles." In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, pp. 2134-2140. 2002.

Fefilatyev S., Smarodzinava V., Hall L. and Goldgof D.. "Horizon detection using machine learning techniques." In *Machine Learning and Applications, 2006. ICMLA'06. 5th International Conference on*, pp. 17-21. 2006.

Jian L. and Xiao-min L., “Vision-based Navigation and Obstacle Detection for UAV,” International Conference on Electronics, Communications and Control, pp. 1771–1774, 2011.

McGee T., Sengupta R. and Hedrick K. "Obstacle detection for small autonomous aircraft using sky segmentation." In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 4679-4684. 2005.

Minwalla C., Watters K., Thomas P., Hornsey R., Ellis K. and Jennings S. "Horizon extraction in an optical collision avoidance sensor." In *Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference on Electric and Computer Engineering*, pp. 210-214.

Neubauer C., “Evaluation of convolutional neural networks for visual recognition,” IEEE Trans. Neural Networks, vol. 9, no. 4, pp. 685-696, July 1998.

Park S., Won D.H., Kang M.S., Kim T.J., Lee H.G. and Kwon S.J. RIC (robust internal-loop compensator) based flight control of a quad-rotor type UAV,” IEEE /RSJ International Conference on Intelligent Robots and Systems, pp. 3542–3547, 2005.

Shen Y., Krusienski D., Li J., Rahman Z., A Hierarchical Horizon Detection Algorithm, In *IEEE Geoscience and Remote Sensing Letters*, Vol. 10, No.1, pp.111-114, Jan. 2013.

Shinzato P., Grassi V., Osorio F., and Wolf D.. "Fast visual road recognition and horizon detection using multiple artificial neural networks." In *Intelligent Vehicles Symposium (IV)*, pp. 1090-1095. 2012.

Straznicky P., Samson C., Ahmadi M., Goubran R., T. Pierce, Whitehead A., and Ferguson S.. “GeoSurv II Unmanned Aircraft System – A solution for geomagnetic airborne surveys”. 3rd Joint CMOS-CGU (Canadian Meteorological and Oceanographic Society; Canadian Geophysical Union) Congress, Ottawa, ON, 31 May – 4 June, 2010.

Thurrowgood S., Soccol D., Moore R., Bland D., and Srinivasan M V. "A vision based system for attitude estimation of UAVs." In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 5725-5730. 2009.

Todorovic S., Nechyba M., and Ifju P.. "Sky/ground modeling for autonomous MAV flight." In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1, pp. 1422-1427. 2003.

LeCun Y., Bottou L., Bengio Y. and Haffner P., “Gradient Based Learning Applied to Document Recognition,” Proceedings of IEEE, 86(11):2278-2324, 1998.

Zafarifar B., and Weda H.. "Horizon detection based on sky-color and edge features." In *Visual Communications and Image Processing 2008*, Vol. 6882, pp. 20-28. 2008.

Web-1: NVIDIA, (2012, April 16). NVIDIA CUDA C Programming Guide (v 4.2) [online], Available: http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDAGuide.pdf