UAV Navigation above Roads Using Convolutional Neural Networks

Thomas Ayoul tayoul@stanford.edu
Toby Buckley tobyb@stanford.edu
Felix Crevier fcrevier@stanford.edu

Abstract

In this project a UAV is used to autonomously fly between waypoints in a Pygame simulation. The UAV is constrained to move only above concrete pathways while mapping the surrounding area. A semantic segmentation neural net is fed images from a gimbaled downward-facing camera and outputs a same size array of categories; either road or not-road. The resultant binary image is post-processed using the Hough Transform to probabilistically fit line segments to the road, and the closest line is followed by a low-level flight controller. The best net obtained an overall accuracy of 96%, with 72% true-positives in classifying road pixels.

1. Introduction & Problem Statement

The problem of autonomous navigation to a given way point using a fixed-wing UAV is explored. The purpose of this problem is to avoid buildings and collect all flying traffic in one area without reliance on prior maps. Additionally, throughout its flight, the UAV collects a map of the environment for future use. Because of the problem, the UAV is constrained to fly above existing roadways only. It is assumed that no prior map information is known, so roads are sensed on board the plane. Sensing is performed through a downward facing, body-fixed camera whose images are processed using a convolutional neural net (CNN). The CNN performs semantic segmentation to partition the image into areas of 'road' & 'not-road'. The output is post-processed using the Hough Transform into line segments which best contour the curvature of the road. Finally, a low-level flight controller uses the closest line segment to fly above the road (Fig. 1).

The biggest requirements on our net is real-time forward passes, and high road accuracy. To meet this, a GPU is used for training and forward-passes, and many different nets are trained using DIGITS, a training system webapp, to determine the best architecture. By utilizing a neural net in this problem, we hope to increase the accuracy and robustness of road detection, while allowing safer flight to occur.

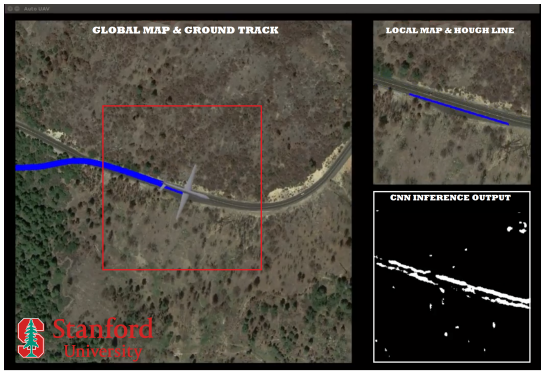

Figure 1: Simulation of the UAV. Pygame is used to run the game.

2. Prior Work

Segmentation, or classification on a pixel basis, is as old as computer vision. The desire to segment images into a small number of categories has applications in medical imagery, aerial photography, file compression, and more. Hard-coded techniques include k-means clustering in the pixel-space, edge detection, thresholding, and more[1].

A more in-depth solutions was explored in 2000, where areas of road can be connected through 'snakes' even when occlusion is present[14]. The paper uses a geometric variational principle method using a curve-energy functional to calculate the 'snakes'.

The classic machine learning technique of support-vector-machines was also used in 2007 by Huang et. al. to extract road centerlines from high-resolution imagery[15]. This approach relies on defining features and taking advantage of roads natural structures (eg. narrow width, long length)

Recently, deep neural nets were utilized for segmentation with great success. The basic idea is to take the classic classifier CNN architecture and scale the output to give pixel by pixel information on what category each fall into. In order to obtain a pixel-to-pixel mapping from input to categories, the output of a classic CNN must be upsampled in smart ways and to the correct size in order to retain the information gained in the downsampling layers.

In 2013, Volodymyr Mnih explored many different networks for aerial image segmentation into roads and not-roads [4]. Mnih studied fully-connected, factored, local untied, and convolutional layers of varying depth. He also researched the effects of the hyperparameters. His results indicate that a three layer depth gives you the most performance for computation, but further layers are subject to diminishing returns. At his best, Mnih received 90% recall accuracy.

More recently in 2015, J. Long et al. improved the start-of-the-art in segmentation by introducing the 'fully convolutional net' (FCN) [3]. High-resolution segmentation is obtained by upsampling the outputs of the CNN at different depths of the architecture and adding together the result. This method allows the global information from deeper in the net to combine with local information found shallower in the net. The paper boasts state-of-the-art performance, with up to 90.3% pixel accuracy on a subset PASCAL VOC 2011[12]. The upsamples are calculated by de-convolving the layer, sweeping a filter over the matrix but with an effective stride of $1/f$, f being an integer. J. Long converted classic nets such as AlexNet, VGG, and Googlenet into FCN nets and demonstrated their ability to fully segment images.

In the same year, V. Badrinarayanan and A. Kendall introduced the "SegNet" architecture[2]. This architecture also utilizes an encoder-decoder structure, where the encoder is topologically identical to classic nets. The decoder portion is similar to FCN's, but instead of de-convolving the layers, it uses an upsample+convolve layer. Because the SegNet approach uses a larger decoder, it boasts improved performance over FCN. SegNet obtains state-of-the-art performance, with 90.4% global testing accuracy on average on the CamVid 11 dataset[11].

3. Methods

We chose to use the SegNet architecture due to its good documentation, tutorials, and public github[5] with examples for training your own segmentation neural net. Our project started with the the SegNet-basic architecture and from there we tweaked the design to explore the changes in behavior it created (Fig. 2).

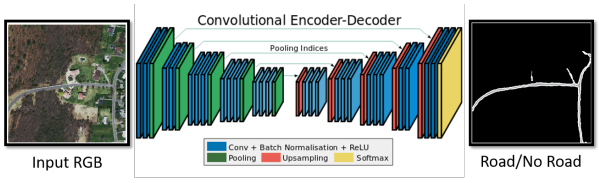

Figure 2: The SegNet architecture developed by Alex Kendall

There are some unique features of the SegNet which allow the architecture to segment an image into pixel-by-pixel classifications. First is the max-pool masking, which saves the exact indice a max-pool value is extracted from to be used in the upsample layers. The upsamples are performed using a bed-of-nails approach with locations given by the corresponding mask.

The next novelty in this architecture is combination of the max-pool masking with a post-upsampling convolution. This eliminates the need for a typical upconvolution layer and is intuitive in its performance. Deeper layers use the max-pool value and repeat the process. During upsampling the deeper layers max value is placed at the masking location. The effect this has is that the masking obtains the positions which are identified most as "road", and the upsampling-convolution then populates the surrounding pixels with values similar to the max value. The physical intuition is that if a human can identify the middle of a road with very high probability, then most likely the immediate area surrounding it is also road, with decreasing probability as you move away from the epicenter.

Additional benefits include minimal parameters compared to other segmentation techniques, meaning that full end-to-end training can be performed using nothing but vanilla SGD (or any other standard optimizer).

SegNet also allows each class to be weighted individually. This lets the user decide how much the loss is penalized for incorrectly classifying certain classes. In our case, we care very deeply about locations of road because those pixels are not only sparse, but important for our application. When training, the roads are weighted at 0.8, while 'not-road' is weighted at 0.2. This way the net is far more motivated to correctly identify the true-positives, but doesn't label every pixel as 'road' either.

In order to test different variations on SegNet, we utilized NVIDIA's DIGITS deep learning GPU training system[6]. The webapp lets users easily keep track of datasets, and nets, and provides visualization tools for the different filters, activations, and heatmaps.

Our best SegNet — coined "RoadNet" — was obtained from experiments discussed in section 5. It utilizes 4 encoding layers (conv1-conv4), followed by 4 decoding layers (conv5-conv8). Each layer consists of an 8 filter convolution which maintains image size, batch norm, the RELU non-linearity, and finally a max-pooling which divides each image dimension by 2. The architecture is summarized in table 1. First information is encoded down to 8 kernels of size 24x24, this information is then scaled up to the image input size during the decoder phase.

4. Dataset

We trained our models using images from the University of Toronto Road and Building Detection Datasets [4],

Layer	Type	Size
Input	RGB	375x375x3
Norm	LRN	375x375x3
Conv1+BN+RELU	8 Filters	375x375x8
Pool1	2x2 Max	188x188x8
Conv2+BN+RELU	8 Filters	188x188x8
Pool2	2x2 Max	94x94x8
Conv3+BN+RELU	8 Filters	94x94x8
Pool3	2x2 Max	47x47x8
Conv4+BN+RELU	8 Filters	47x47x8
Pool4	2x2 Max	24x24x8
Upsample4	Bed-of-Nails	47x47x8
Conv5+BN+RELU	8 Filters	47x47x8
Upsample3	Bed-of-Nails	94x94x8
Conv6+BN+RELU	8 Filters	94x94x8
Upsample2	Bed-of-Nails	188x188x8
Conv7+BN+RELU	8 Filters	188x188x8
Upsample1	Bed-of-Nails	375x375x8
Conv8+BN+RELU	8 Filters	375x375x8
ConvClassifier	1 Filter	375x375x2
Softmax		

Table 1: RoadNet architecture. 23,338 total parameters were learned.

which labels road pixels in aerial photographs taken from a relatively high altitude. To lower memory requirements, we split every image into 16 smaller 375x375 RGB images, for a total of 4500 data points. Figure 3 shows an example image-truth pair used in training. The left shows the input image and the right the ground-truth with white ("1") as road and black as not-road ("0").

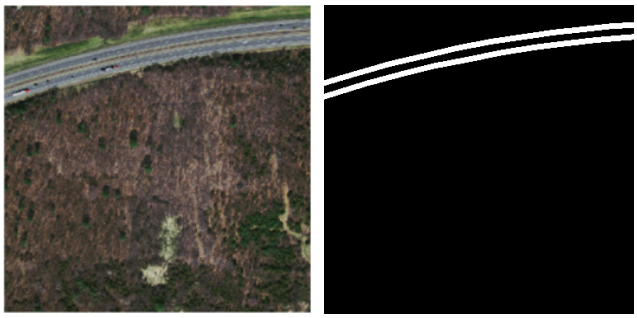

Figure 3: Training example: Image — Ground Truth

5. Experiments and Results

Described below are the results from the various configurations of SegNet we have experimented on.

Optimizer - Adam The Adam optimizer was chosen over vanilla stochastic gradient descent because of the advantages it provides [7]. Adam uses the moving average of its parameters, allowing the optimizer to have a larger effective step size. The downside is that Adam requires additional memory and computation to store and use the moving average of each parameter.

Batch Size. A GTX 970 GPU was used to train the network. This GPU has 4 GB of RAM which can be used to store images in the mini-batch. When using SGD, the GPU can hold 6 input images taking up 3.8 GB of RAM. Due to Adam's higher memory requirements, 4 images were used in the mini-batch for the 2-layer nets, and 2 images mini-batch size for the larger 4-layer nets.

Evaluation Metric. As the class distribution on the data is highly uneven — road density is usually between 0.0 and 0.2, a pixel-wise global accuracy is a poor metric, as classifying every pixel as "not-road" would lead to an excellent score. Therefore, to evaluate the performance of our models, we use a per-class accuracy (Eq. 1).

$$acc_{road} = \frac{\sum_i \mathbb{1}\{y_i = \hat{y}_i = 1\}}{\sum_i \mathbb{1}\{y_i = 1\}}, \tag{1}$$

In this equation, road pixels are labeled as "1" and not-road pixels as "0". This accuracy function is representative of the ability of the network to classify sparse road pixels in an image mostly covered in fields, forests and buildings. Note that false-positives and false negatives penalize the accuracy, yet true negatives do not increase it. Moreover, most of the noise created by false-positives and negatives is rejected during post-processing.

5.1. Network parameters

As suggested in [9], we established our best network configuration using grid search. We focused on the kernel size and the number of filters per layer and established performance using the metric described above for 16 different configurations. For these analyses, we used our standard 4-layer architecture and a learning rate of 0.01. Figure 4 shows the outcome of this tuning analysis. By inspection, we conclude that a kernel size of 7x7 with 8 filters per layer lead to the best road accuracy. Consequently, this configuration is named "RoadNet" and is used for all subsequent analyses and demonstrations. Moreover, we can observe a few trends in the performance of these configurations. First, networks with lower filter sizes seem to benefit from a high number of filters per layer, which is typical for CNN's [8]. Second, conversely to most classifying CNN's, we obtain better performance with medium-sized filters. This could be explained by the spread-out nature of roads, which have

very low feature resolution and are characterized by bands spanning across a large number of pixels. In this case, larger filters potentially capture more of this larger scale characteristic and lead to less local over-fitting.

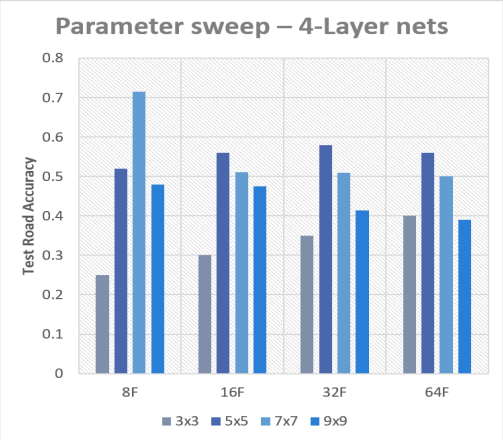

Figure 4: Sweep on kernel sizes and number of filters.

5.2. Learning Rate

The learning rate was varied to examine the effect it has on the learned parameters. Most nets were trained with $\eta = 0.01$ for 5-10 epochs, and many of the smaller nets converged onto their final within 2-4 epochs. With this learning rate the 2-layer nets only took 1 epoch to converge. In nearly all nets trained with the higher learning rate, dead filters were produced by convergence (Figure 5). Dead filters may be caused by a vanishing gradient or through too high of a learning rate. Additionally, the number of dead filters increases as the net becomes deeper. This means more and more filters don't contribute to accuracy but still require computations during the forward pass.

5.3. Filters and Activations

The filters have very obvious affects on the image. Figure 6 shows the activation map for each filter after the first convolutional layer. The dead filters can be ignored as they do not contribute. The middle filter has become a "gray" filter. The corresponding activation shows a very high activation for the road since its a solid gray chunk. Arbitrary gray pixels also get marginally activated throughout the image. In contrast, the top right filter is a "green" filter which highlights foliage and grass. The activation for this filter highlights all the greenery around the road. Such a neural net does an adequate job of identifying roads, but at its core the net just bisects the image into pixels of grayish color and greenish color. The level of grayness of the area around a pixel determines if its labeled road. Hence, the net does not learn abstractly what a road is. A cause of such behavior may be not enough training data, over-fitting on the current data, or too high of a learning weight. The common cause is learning rate, so a new batch of nets were trained which reduced this parameter.

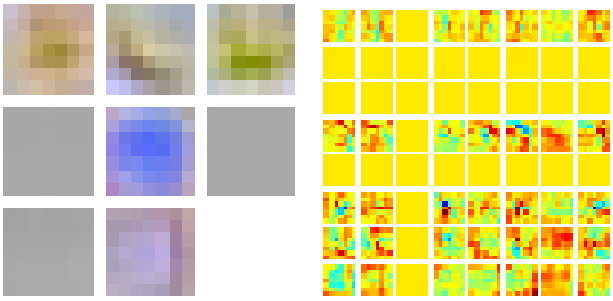

Figure 5: Filters for conv1 (left) and conv8 (right) with high learning rate.

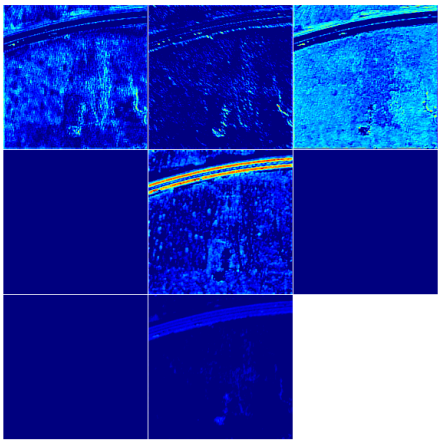

Figure 6: Activation maps for each filter after conv1.

The second set of nets were trained with $\eta = 0.0005$, a reduction of 20x. These nets were trained for 30 epochs, but mostly completed their training within 15 epochs. Figure 7 shows the weights of these nets. The difference is immediately apparent. No dead weights are present, but the filters are more difficult to physically interpret. Now the net has learned much better what a 'road' is abstractly.

This net is more robust to changes in environment, in fact it performs better in areas with less trees. The reason is because now the net looks for 'alleyways'; areas where a similar feature continues in a mostly straight line and is blocked on either side by different features. In fact, false positives will appear in tree-filled images when the boughs of the trees line up in a way that creates a natural pathway through the forest.

Interestingly, the conv8 layer, which operates after the upsample masking from conv1, appears to have edge detector filters. This layer is the final convolution before the

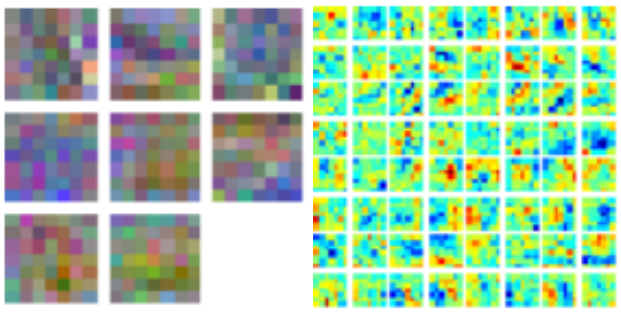

Figure 7: Filters for conv1 (left) and conv8 (right) with lower learning rate.

softmax, so it plays a major role in calculating the probability of each pixel. The edge detectors may act as a 'road boundary decider', where the network determines where a road ends and the terrain begins. The shape makes sense, as most every terrain-road boundary is locally a straight line.

5.4. Heatmaps and features

After the final layer, a heatmap of probabilities can be examined (Fig. 8). The areas of bright red indicate high likelihood of 'road', and the blue indicates 'not-road'. Both edges and color plays a role in determining the probability, as can be seen in the lower part of the image, where some parts of the foliage have decent probability due to the edges present, but lack the right colors. In contrast, the highway has both edges and grayish colors, so the heatmap shows a very high probability in that area.

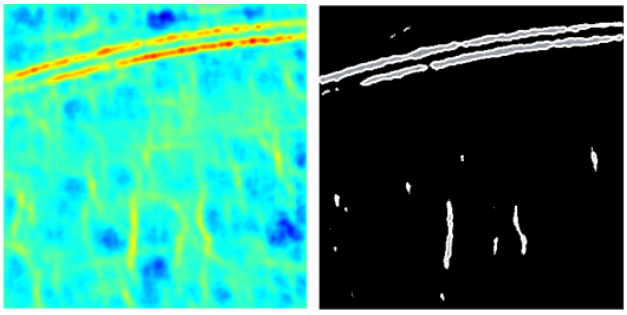

Figure 8: Final heatmap of RoadNet and the corresponding inference.

5.5. Training epochs

The true-positive road accuracy was examined as a function of the epoch count for RoadNet. What was found is that road accuracy follows a parabolic trajectory, where the best accuracy occurs at epoch 11. Because of the large number of parameters learned in RoadNet, and the relative simplicity of a two-class road segmentation problem. This is indicative of the net over-fitting after many training epochs.

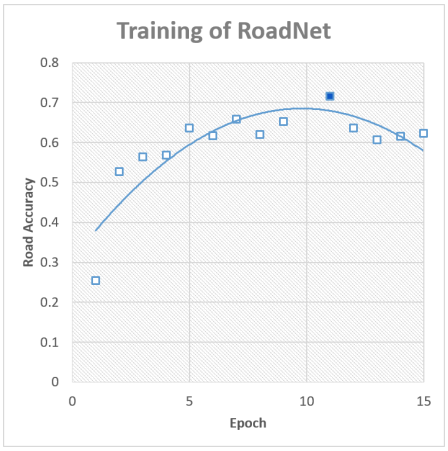

Figure 9: Road accuracy over many epochs.

5.6. Loss and Convergence

The Segnets were trained using the cross-entropy loss over the total image, where each pixel contributes equally to the total loss (Eq. 2). Backpropagation is executed normally, the only difference is now you have one loss per pixel, which each add onto the full backprop gradient. Due to the importance of correctly classifying roads, that class was weighted 4x the not-road category.

$$loss = -\frac{\sum_i w_i \log(\exp(f_{y_i}))}{\sum_j \exp(f_j)} \tag{2}$$

Figure 10 shows the loss and validation accuracy over 30 epochs for a net trained with lower training rate ($\eta = 0.0005$). Notice that within two epochs the training loss has reduced significantly, and validation accuracy is already at 95%. As previously mentioned, validation accuracy is not a good metric for road detection performance, so the reduction in training loss is more focused on, as this takes into account the weighting of the different classes.

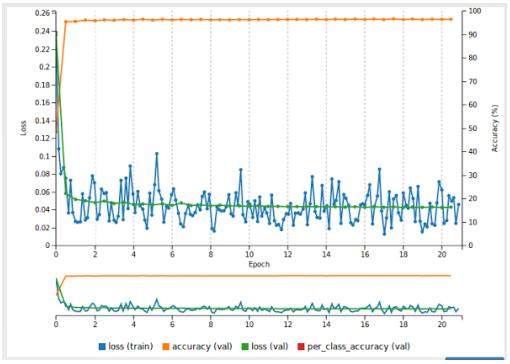

Figure 10: Loss and accuracy convergence over 30 epochs with low learning rate.

Throughout training the validation accuracy slowly de-

creases, indicating that the net is still improving its total accuracy, if only marginally. The oscillations in training accuracy is due to the small batch size and variance in training data.

5.7. Results

We now compare the performance of our best net to a rudimentary version of SegNet. Global and road accuracies are shown in table 2. We clearly observe the increase in both global and local accuracy when compared to the literature. Moreover, these results justify the extra complexity of adding two extra layers to the SegNet-2layer architecture. In comparison, Mnih obtains a high of 90% global recall on the same data set. It is fair to say we obtain similar performance on binary road/no-road segmentation.

Network	Global accuracy	Road accuracy
SegNet-2layer	0.90	0.40
RoadNet	0.96	0.72

Table 2: Comparative results.

6. Simulation

To simulate using hardware-in-the-loop, a real-time simulation was implemented in the Python Pygame/OpenGL environment. The ground images are scraped from Google using their Maps static API and a GPS longitude-latitude pair (x,y). Figure 11 shows the flow of information during the simulation. A local image — representing what a ground-facing gimbled camera would capture — is passed on to RoadNet for segmentation. We then use the Hough transform, a line segment regression algorithm described in [10], to extract the individual lines longer than a certain threshold from the inferred road pixels. Among the Hough lines with highest score, we use the one closest to the aircraft as an input to the line follower. This block determines the command heading ψ_c by weight-averaging the direction to the line and the direction to the objective (end of the line). To fly closely above the road, the former is weighed with a factor of 3.0. The command heading is fed into nested-loop controller, which uses a gain on the heading error $(\psi_c - \psi)$ to set the the command bank angle ϕ_c, and another gain on the bank error $(\phi_c - \phi)$ to set the aileron deflection δ_a. This aileron command is fed into the aircraft — a simplified 6-DOF rigid body with actual UAV dynamics — and produces the required rolling moment to turn and reach the command heading, i.e. follow the road. An example of a real-time simulation over CA-120 in Yosemite National Park can be accessed on our YouTube page [13], here.

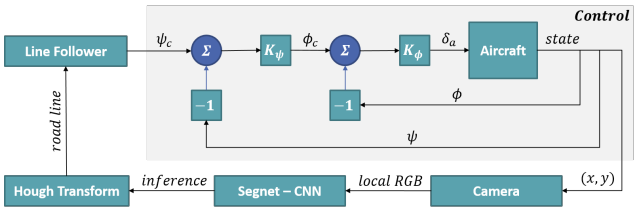

Figure 11: Hardware-in-the-loop simulation.

7. Conclusion

In the scope of this project, we trained convolutional neural networks with Alex Kendall's SegNet Encoding-Decoding architectures to detect roads on aerial images in real-time. Following a hyperparameter grid sweep, we reached a peak road accuracy of 72% with RoadNet, a CNN with 4 encoding-decoding, 8 filter deep layer pairs. Quantitative and qualitative analyses were performed, yielding strong conclusions regarding the effects of learning rate on filter convergence and general network performance. Moreover, we tested different network complexities and depths to understand trends in overfitting. Finally, we integrated RoadNet in a real-time simulation loop, allowing a realistic UAV model to accurately navigate over roads.

8. Future Work

The next thing to do is implement RoadNet on the real hardware we have (Fig. 12). For this setup, the sensing is performed through a gimbaled downward facing, body-fixed GoPro whose images are fed to an Arduino hooked up to an onboard GPU for real-time inference.

Another big step would be implementing a higher-level path-planner so that line segments can be converted into nodes & edges for an algorithm such as A* to operate on in order to determine which path to take when passing over crossroads. Low-level controls would be handled by a pixhawk controller embedded on the plane.

Figure 12: UAV for real-life testing.

References

[1] Jain, Ramesh, Rangachar Kasturi, and Brian G. Schunck. Machine vision. Vol. 5. New York: McGraw-Hill, 1995.

[2] Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." arXiv preprint arXiv:1511.00561 (2015).

[3] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[4] Mnih, Volodymyr. Machine learning for aerial image labeling. Diss. University of Toronto, 2013.

[5] SegNet Github. https://github.com/alexgkendall/caffe-segnet

[6] DIGITS Github. https://github.com/NVIDIA/DIGITS

[7] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[8] Li, Fei-Fei, Andrej Karpathy, and J. Johnson. "CS231n: Convolutional neural networks for visual recognition." University Lecture (2015).

[9] Bengio, Yoshua. "Practical recommendations for gradient-based training of deep architectures." Neural networks: Tricks of the trade. Springer Berlin Heidelberg, 2012. 437-478.

[10] Duda, Richard O., and Peter E. Hart. "Use of the Hough transformation to detect lines and curves in pictures." Communications of the ACM 15.1 (1972): 11-15.

[11] G. Brostow, J. Fauqueur, and R. Cipolla, Semantic object classes in video: A high-definition ground truth database, PRL, vol. 30(2), pp. 8897, 2009.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html

[13] T. Buckley, F. Crevier, "CS231n Project Simulation Flight" Video. https://www.youtube.com/channel/UC3cpWY9fDpq4we-nlUnFAYA

[14] Laptev, Ivan, et al. "Automatic extraction of roads from aerial images based on scale space and snakes." Machine Vision and Applications 12.1 (2000): 23-31.

[15] Huang, Xin, and Liangpei Zhang. "Road centreline extraction from highresolution imagery based on multiscale structural features and support vector machines." International Journal of Remote Sensing 30.8 (2009): 1977-1987.