Deep neural network for precision multi-band infrared image segmentation

Thomas Lu[a], Alexander Huyen[a], Kevin Payumo[b], Luis Figueroa[c], Edward Chow[a], and Gil Torres[d]

[a]NASA/Jet Propulsion Lab/California Institute of Technology, Pasadena, CA, USA; [b]Univ. of Calif. Irvine, CA, USA; [c]Occidental College, CA, USA; [d]Naval Air Warfare Center, Point Mugu, CA, USA

ABSTRACT

Image segmentation is one of the fundamental steps in computer vision. Separating targets from background clutter with high precision is a challenging operation for both humans and computers. Currently, segmenting objects from IR images is done by tedious manual work. The implementation of a Deep Neural Network (DNN) to perform precision segmentation of multi-band IR video images is presented. A customized pix2pix DNN with multiple layers of generative encoder/decoder and discriminator architecture is used in the IR image segmentation process. Real and synthetic images and ground truths are employed to train the DNN. Iterative training is performed to achieve optimum accuracy of segmentation using a minimal number of training data. Special training images are created to enhance the missing features and to increase the segmentation accuracy of the objects. Retraining strategies are developed to minimize the DNN training time. Single pixel accuracy has been achieved in IR target boundary segmentation using DNNs. The segmentation accuracy between the customized pix2pix DNN and simple thresholding, GraphCut, simple neural network and ResNet models are compared.

Key words: Deep learning, Neural network, Machine learning, Training data, Precision segmentation, infrared IR target, Computer vision.

I. INTRODUCTION

Segmentation of an object from its background is the first step in object recognition and measurement. Object detection focuses on creating bounding boxes around specific objects within the frame [1, 2]. In many pattern recognition applications, the detection of the presence of an object and its location would be sufficient. However, in the case of object segmentation, a precise boundary between the object and its background environment needs to be drawn accurately. It could take an operator over 20 minutes to carefully draw the boundary of an object in a photo editing software such as Photoshop or ImageJ. An automated object segmentation software could free people from tedious manual work. Digital image processing, filtering and thresholding have been used in image segmentation [3, 4]. The segmentation accuracy suffers when the background is not uniform. Neural networks have been used to perform adaptive thresholding [5 - 7]. A simple neural network with a few hidden layers still would not handle complex background changes. Deep convolutional neural networks (CNNs) are commonly used to perform image classification tasks like object detection and semantic segmentation. Object detection using CNNs have been utilized in many applications, like multiple target tracking surveillance and autonomous vehicles. Semantic segmentation is the task of classifying pixels into distinct classes, resulting in well-defined segmentations in an image. These CNNs are comprised of an input and output layer, with hidden layers in between. Each layer consists of neurons which are connected to other neurons in different layers. The amount of connections between these neurons in different layers determines the amount of computation needed for the CNN. Fully convolutional

networks (FCN) contain only layers where every neuron is not connected to every neuron in the previous layer. Convolutional networks have been applied in segmentation tasks [8-10]. CNNs perform image classification tasks by fine tuning its connected neurons around a loss function which provides an objective score for the CNN's output. The loss function affects the quality of the output produced by the CNN, so this must be carefully selected based on the image type and desired results. We have used a conditional Generative Adversarial Network (cGAN) to perform the selection of this loss function and to simultaneously minimize the loss function for our particular task. The focus of this work is to precisely segment objects from noisy backgrounds in infrared images of the long-wavelength, mid-wavelength, and short-wavelength IR bands. The precision benchmark was done by taking individual pixel locations in the segmentation mask generated by the neural network and compared to its corresponding pixel location in the ground truth mask. Each pixel location is then checked for matching classifications between the neural network mask and ground truth mask. The total amount of mismatched pixel classifications is calculated to provide an error metric based on the area of the ground truth mask.

The intensity of the object and backgrounds in the images vary between frames and wavelengths, which contribute to the difficulty in achieving a precise segmentation between these two classes. High noise backgrounds with similar pixel intensities to the object results in ambiguous boundaries that can prove to be difficult for manual segmentation techniques. Low contrast and low intensity features of the objects in these infrared wavelengths also contribute to this difficulty in accurate classification. Small features are areas where precise segmentation is required for accurate pixel classifications, especially in noisy background environments.

In Section II, a customized pix2pix model for precision object segmentation is presented. In Section III, several DNN training optimization strategies are discussed to obtain optimum training results without prolonged training. In Section IV, we compare the performance of the model with commonly used segmentation methods, such as GraphCut, Thresholding, a simple neural network, and Deeplab ResNet.

II. DEEP NEURAL NETWORK FOR PRECISION SEGMENTATION

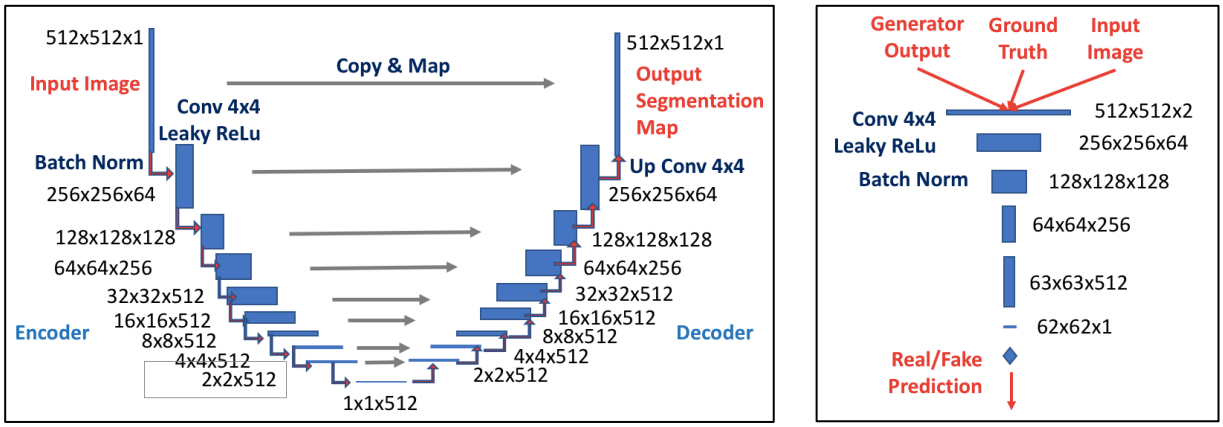
Deep learning is a machine learning technique that learns features and tasks directly from data and uses an artificial neural network architecture (ANN). Traditional neural networks contain one to three hidden layers, whereas deep learning models can contain as many as 100 hidden layers. As a result, this type of model is often referred to as a "Deep neural network" (DNN) [12]. Typically, DNN models learn classification tasks from large labeled datasets of images, texts, or audio through a process called feature extraction. The multi-layer nature of a DNN allows for each layer to encode distinctive features. When classifying features in an image, some layers can extract edge features while others can extract texture. As the layers advance towards the output layer, encoded features become more complex.

One of the drawbacks of the DNN is the requirement of a large number of training samples. It is time consuming to prepare training samples and ground truth data. Sometimes it is impossible to obtain many training samples. In this research we use pix2pix, a DNN model that makes use of a conditional generative adversarial network (cGAN) that learns a mapping from an input image to an output image [11]. By the nature of generative adversarial networks (GANs), the model generates a loss function directly from the training data. Pix2pix is composed of two adversarial neural networks, one network, dubbed the generator, learns to generate images similar to the input images while the other network, the discriminator, learns to distinguish whether the generated images are indistinguishable from the input images. In other words, these models are trained adversarially. As opposed to traditional neural network models that generate images by learning from random feature sets in the training data, pix2pix's cGAN model is conditioned to learn mappings from a dataset of paired images, a real image (left) and a ground truth (GT) (right), which is the precise boundary of the object. In this research, the model is conditioned to take an IR image of an object as an input and produce a binary mask of the object as an output. Using the feedback from the discriminator, the generator is able to produce images that resemble the ground truths from the input image.

Figure 1 shows the network architecture of the U-shaped generator (Unet) (Figure 1 (a)) and the discriminator (Figure 1 (b)). The generator architecture consists of a nine-layer encoder and nine-layer decoder with skip connections (copy & map) between each encoder and decoder layer. In this modified pix2pix implementation, an additional layer was added to both the encoder and decoder to increase the resolution from 256x256 to 512x512. As a result, a skip connection was added between the eighth encoder layer to the first decoder layer, and another skip connection was added between the first encoder layer and the eighth decoder layer.

An input image is resized into a 512x512 pixel image to match the size of the input layer. The input image comes in the first layer in the top left. The data flows through the network and the segmented map comes out of the top right side of the Unet. In each encoder layer, the network performs a 4x4 convolution, batch normalization, and an activation function (leaky rectified linear unit (LeakyReLU)). The network performs a 4x4 deconvolution, batch normalization, and an activation function (rectified linear unit (ReLU)) for each decoder layer.

The discriminator architecture consists of six layers. The first layer of the discriminator does not have a spatial batch normalization applied and the final layer retained the sigmoid function. Both the generator and discriminator used in this research were created with one input and output channel, compared to the original implementation of three input and output channels.



(a) (b)

Figure 1: A cGAN neural network for object segmentation: (a) the U-shaped generator; (b) the discriminator.

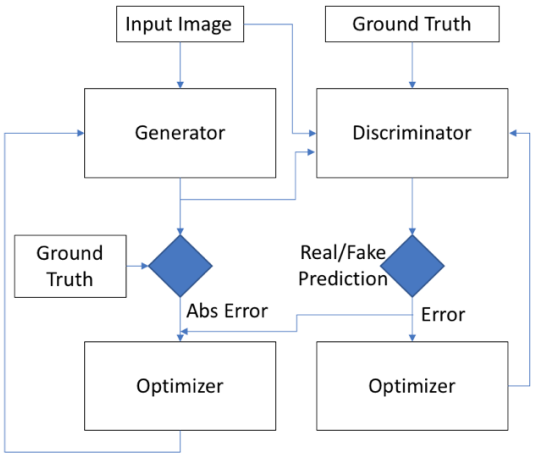


Figure 2: Generator & discriminator training process.

DNN Training Process

The cGAN network [11] training process shown in Figure 2 starts with using the generator to create an output image. This output image is passed to the discriminator for updating the discriminator weights. A corresponding ground truth image is also passed to the discriminator to produce a second classification error. Both classification errors from the generator output and ground truth are used to update the discriminator. The updated discriminator trains the generator by providing its classification error and this process is repeated until the maximum number of training iterations has been reached.

Discriminator Training
The discriminator is passed in two images, an input image and either a ground truth image or generator output image. The discriminator is unaware of the second image type and must predict whether it is a real (ground truth) or fake (generator) output. The discriminator produces classification errors for both pairs of input images. An Adam optimizer [13] is used to update the weights of the discriminator based on the resulting classification errors.

Generator Training
The generator is passed in an input image and produces an output image. This output image is used to compute a mean absolute error [14] against the ground truth and a classification error from the discriminator. Both of these errors are passed to the Adam optimizer to update the weights of the generator.

III. OPTIMIZATION OF DNN TRAINING

The base cGAN model is modified with an additional two layers, one added to the encoder and the other to the decoder. These additional layers increase the resolution in the initial and end layers from 256 to 512. The increased resolution improves the accuracy of segmentations, at the cost of increasing training time. The Torch input image loader is also modified to process images in grayscale from the default RGB color channels, since images used in this infrared wavelength context do not utilize these extra channels.

The accuracy of the DNN segmentation is evaluated by comparing the DNN binary output mask with the GT mask. An XOR operation is performed on the DNN output and the GT:

$$E = \frac{\sum_{i=1}^{w}\sum_{j=1}^{h} x_{i,j} \oplus y_{i,j}}{\sum_{i=1}^{w}\sum_{j=1}^{h} x_{i,j}} * 100 \qquad (1)$$

Where X represents GT and Y represents the DNN output. “w” and “h” represent the width and height of the image, respectively. If the DNN output is identical to the GT, then E = 0%.

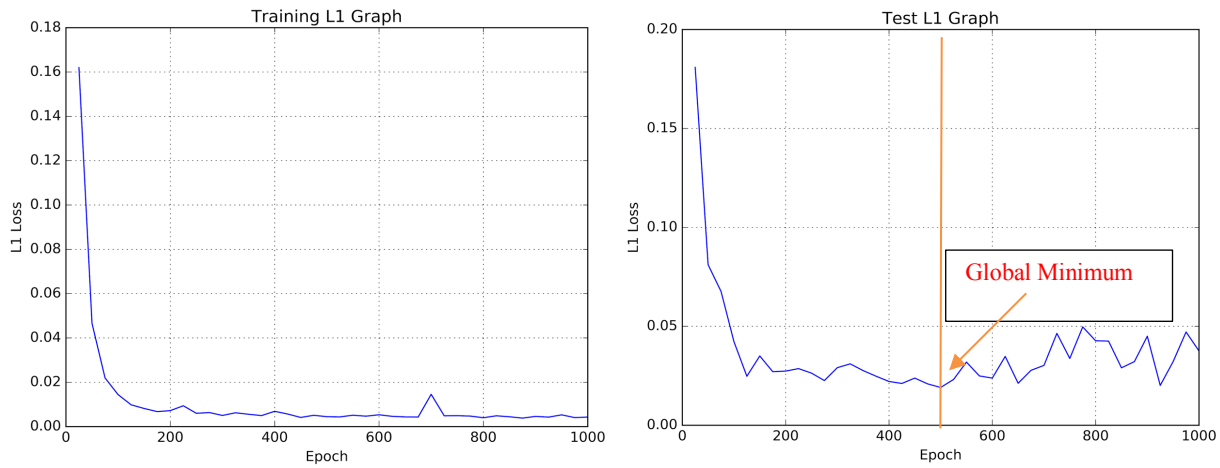
The DNN produces a confidence map with each pixel between [0, 255], where low values represent the likelihood of background and high values represent foreground. An overall confidence value of the DNN output is calculated in the following equation:

$$C = \frac{\dfrac{\sum_{i=1}^{w}\sum_{j=1}^{h} y_{i,j}\,[\,x_{i,j} = 1]}{\sum_{i=1}^{w}\sum_{j=1}^{h} x_{i,j}}}{\dfrac{\sum_{i=1}^{w}\sum_{j=1}^{h} y_{i,j}[\,x_{i,j} = 0]}{\sum_{i=1}^{w}\sum_{j=1}^{h} 1[x_{i,j} = 0]} + 255} * 100 \qquad (2)$$

During the DNN training, it is important to monitor the training and testing error graph, such as the Training and Testing L1 graphs in Figure 3. This training and testing L1 error is calculated from the mean absolute error [14] between the ground truth and the generator output. The quality of the DNN can be measured by the trend and convergence of the training and testing error curves. A good training is represented by both training and testing error curves trending lower together closely. If the training error does not go down, the DNN architecture needs to be adjusted. If the testing error does not go down along with the training error, then the training sample set needs to be expanded to represent the test samples. The testing error curve could go up and down. A global minimum point can be found to obtain the best DNN model. Due to limited on board memory size on the GPU board, the DNN can be trained by a portion of the training samples in each epoch. In this case, a moving average of the L1 error in multiple epochs would be more representative of DNN performance in the global minimum search.

A significant part of achieving high accuracy while maintaining a small dataset is through a systematic process of analyzing the results of a model, choosing an improved dataset, retraining the model, and repeating the process.

Through this process the DNN model is "fine-tuned" while maintaining a small dataset and reducing the total time for trainings and retraining. Each model that produces accurate results is chosen as a standard weight matrix to retrain in the next step.



(a) (b)

Figure 3: DNN training L1 loss graphs: (a) Training error curve; (b) Testing error curve.

Selecting the new training images is based on empirical results of the model's confidence masks. To reduce the time for each training cycle and optimize and increase the accuracy of the model, we compared two methods for choosing new images for retraining. The enhancement training images are divided into two categories: full frame (FF) images and feature cropped (FC) images (see an example in Figure 4). Given a DNN output confidence map from a model, new training images are selected that contain similarities to the specific features to be enhanced as opposed to arbitrary images. In Figure 4 (b), the confidence mask is missing a part of the rotor, as such, a similar image (Figure 4 (c)) is selected to enhance this particular feature — the enhancement images. While selecting new training images, a small dataset is maintained by selecting a minimal number of enhancement images per retraining.

(a) (b) (c) (d)

Figure 4: Example of an Augmented FF Dataset, (a) Original image, (b) GT, and an example of an Augmented FC Dataset: (c) missing rotor part, (d) GT of the missing rotor.

An open-source image processing software GNU Image Manipulation Program (GIMP) is used to manipulate the IR images. Every FC image is created by removing the target from the background by utilizing GIMP's Resynthesizer tool. This tool works by taking surrounding pixels of a sample image texture and filling in the desired area with new texture from the surrounded data. The target is removed and the surrounding area's texture is taken to fill in the missing target creating an image with only the desired feature. After removing the object, the cropped feature is overlayed on top of the background image. The goal of creating FC enhancement images was to increase the loss contribution of the object's features during the model's retraining. By increasing the number of training images containing these loss contributions, the desired features were enhanced.

IV. EXPERIMENTAL RESULTS

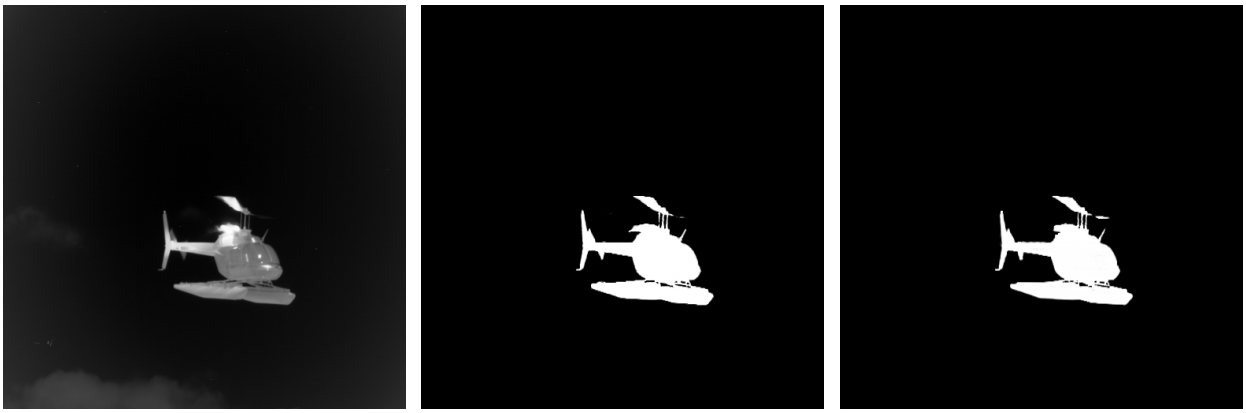
The DNN models were trained using a NVIDIA GeForce 1080TI card on a server with an Intel Xeon(R) E5-2620 v2 @ 2.10GHz CPU. The modified pix2pix, DeepLab v2 ResNet and Matlab neural network models were trained on identical datasets. The XOR test error for each method was calculated by running each model on a 11-image standard test dataset outside of the training set.

4.1 Retraining Strategy

To investigate and determine the best process for optimizing the models through retraining, two methods have been compared:

- Retraining on an augmented dataset. This includes the initial images with which the model was trained originally, and the addition of the new enhancement frames.
- Retraining on a dataset of just enhancement frames. This dataset discards the initial frames from the original training.

From these two procedures, four different training datasets were then collected to optimize the DNN performance with respect to the validation set. After training the modified pix2pix with an augmented dataset of feature crop images, cases of better performance are demonstrated. Figure 5 shows an example of retraining a DNN with an augmented FC dataset to enhance features: (a) the original image; (b) initial DNN output missing the right side of rotor blade; (c) the retrained DNN output showing the missing rotor blade feature reconstructed.



(a) (b) (c)

Figure 5: Retraining a DNN to enhance features: (a) the original image; (b) initial DNN output missing the right side of rotor blade; (c) the retrained DNN output showing the missing rotor blade part is reconstructed.

In an experiment, the pix2pix model was trained using an initial subset of 12 images from the 68 ground truths used for all neural network trainings. This initial training on the 12 images was trained in 1.73 hours to 1000 epochs. A second training session was done by adding 8 new images to the previously used 12-image set from the 68-image set. The epoch with the lowest L1 validation score from the 12-image training session was selected as the weight matrix to begin retraining for the 20-image training. This 20-image training session was run for 1.96 hours to 1000 epochs. The same process was repeated for the final training, with 16 new images being added to the 20-image dataset. The lowest L1 validation scoring weight matrix from the 20-image set was used to continue training for this new 36-image set. This 36-image training was done for 2.9 hours to 1000 epochs. A final training was done for another 4.05 hours to 1000 epochs on top of the weight matrix from the 36-image set with the lowest L1 validation score using the entire set of 68 images. This process of increasing the training set in batches and retraining using a previous optimal weight matrix generated more accurate results than the regular method of training. In comparison,

using the conventional method of training from scratch with the entire set of 68 images to 10,000 epochs for 42.45 hours achieved an average XOR error percentage of 4.98 % compared to 10.64 hours of training for a total of 4000 epochs and a resulting average XOR error percentage of 4.23%.

Figure 6 illustrates the comparison of the retraining sessions (A, B, C, D) and a single training session (E). It is apparent that the four-step retraining achieved an error rate of 4.23% in total 10.64 hours training. In comparison, a single training session yielded higher error rate of 4.98% and 4 times longer training time.

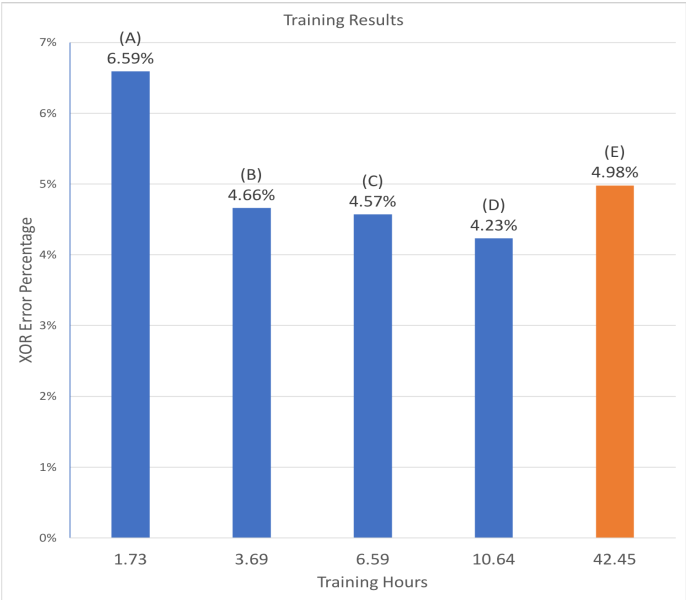


Figure 6: Performance (error) comparison of retraining process: Four-step retraining results (A, B, C, D) compared against single training results (E). The 4-step training (Blue bars) achieved lower error rate in a quarter training time as compared to a single training session (Orange bar). The XOR error percentages for each training iteration was generated by calculating the average results from testing on an 11-image standard test dataset that was not included in the training dataset.

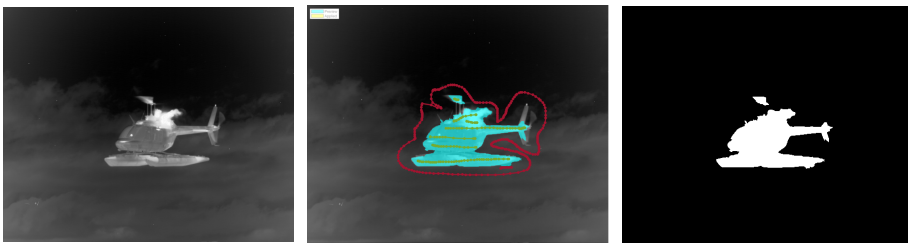
4.2 Comparison of Segmentation Methods

We compare the modified pix2pix DNN segmentation technique with other segmentation methods, such as the simple thresholding methods, GraphCut, DeepLab v2 ResNet model and a simple neural network model.

4.2.1 GraphCut Segmentation Method

GraphCut is a semi-manual segmentation technique which divides the entire frame into regions and weighs each based on the similarity to nearby regions. Edges where the weights are low are separated into foreground and background. A manual selection of samples of the foreground and background are needed by GraphCut to distinguish the class of its regions. More samples can be added to further refine the segmentation, however the high similarity between some regions in the frame will cause misclassifications that cannot be corrected by an increased number of manual samples. Another drawback of GraphCut is that the relative size of regions determines the size of the smallest feature that can be segmented. The implementation of GraphCut found in Mathworks' Matlab Image Processing Toolbox was used.

The level of manual work required for applying GraphCut segmentation is shown in Figure 7. A red line denotes a manually drawn boundary to mark background, and a green line is used to mark the foreground. Roughly five minutes were spent detailing regions in each frame until a satisfactory mask was generated by the GraphCut program. The rotor system and small background voids in the pontoon rigging of the object were areas of difficulty where further manual markings did not improve segmentation precision.



(a) (b) (c)

Figure 7: GraphCut results: (a) original image; (b) manual drawing of rough boundary (red lines) and object intensity (green lines); (c) segmentation result. The GraphCut method misses part of the target.

4.2.2. Manual & Automatic Thresholding using ImageJ

ImageJ is an open source image processing tool used for scientific applications. Binary segmentation masks were created from image thresholding tools in this program by applying an automatically generated threshold. The default automatic threshold function in ImageJ was used to perform the segmentations. This threshold function implements an algorithm based on iterative inter-means or IsoData [4]. This algorithm separates foreground and background with a starting threshold, which is then updated by calculating an average of two averages from values of the pixels above and below the initial threshold. The initial threshold is updated until it is higher than the average of two averages from pixels higher and lower than it.

ImageJ was also used to perform segmentation by manually thresholding the image. The threshold value was selected based on the accuracy of segmented features and amount of noise generated. A higher threshold tends to remove a majority of noise in the segmentation, but resulted in missed low intensity features like the antennae and rotor blades.

The automatic thresholding tool required no manual input, while the manual thresholding method took one to two minutes per frame to select an optimal threshold level based on the amount of noise and pixel values of the input image. This manual thresholding method in ImageJ was also applied to generate ground truths for training data, with differences in the procedure than previously explained. A large number of regions in the input image are hand selected to be used for the application of individual local thresholds. Regions are created based on object features and boundary intensity values. Small size and low intensity features like antennae, rotor systems, and the pontoon rigging were contained into small regions to apply thresholds values well under the average of the object's pixel values. The manual localized thresholding method described creates optimal training data with consistently defined boundaries between foreground and background. A large amount of effort and time are needed to produce high quality ground truths, with each input image requiring 20 - 30 minutes to complete manual classification. Figure 8 shows an example of an image been segmented by auto threshold (Figure 8(b)), and manual threshold (Figure 8(c)). Both the auto and manual threshold are not able to segment the object properly.

(a) (b) (c)

Figure 8: Segmentation results from ImageJ: (a) Original image; (b) segmentation by auto threshold. Background is not properly thresholded; (c) segmentation by manual single threshold. Partial target is missing.

4.2.3. Segmentation using Simple Neural Network

A small scale neural network was used to perform these segmentations. This neural network consisted of one input layer, one hidden layer and one output layer. The hidden layer of this simple neural network contained only 25 neurons. This neural network was trained on the same dataset as the deep neural network used for precise segmentation. Further details of this neural network can be found in [6]. Figure 9 shows that the simple neural network gives decent segmentation result.

(a) (b)

Figure 9: Segmentation result from simple neural network: (a) Original image; (b) Simple neural network segmentation mask. The simple neural net performs a relatively clean segmentation of the target.

4.2.4. Segmentation using DeepLab-ResNet

DeepLabv2 (ResNet-101) is an implementation of deep convolutional networks for the purpose of semantic segmentation. The specific implementation used here features atrous convolution and atrous spatial pyramid pooling, without the application of conditional random fields for post processing. Atrous convolution refers to retrieving full resolution features by using upsampled filters in the convolutional layers. Atrous spatial pyramid pooling (DeepLab-ASPP) is used during the feature abstraction stages to increase resilience against shifts in scale by utilizing multiple filters with various sampling rates [8]. This DNN was trained using the same dataset as the simple neural network and pix2pix DNN. The training data was converted into 1-D labels by translating the ground truth foreground pixel values to match its class label in this network.

Figure 10 shows the results of the segmentation by the ResNet in Figure 10 (b) and Pix2pix in Figure 10 (c). ResNet is capable of identifying the object and separating the object from background. However, the ResNet model was not able to draw an accurate boundary of the object, missing small details of the object. In comparison, the pix2pix DNN performs accurate segmentation and is capable of segmenting small features of the object.

(a) (b) (c)

Figure 10: Segmentation results: (a) Original image; (b) ResNet segmentation output; Partial features of the target is missing; (c) Pix2pix segmentation output. A good target segmentation is performed.

This implementation of DeepLab-ResNet in Tensorflow achieved a comparable PASCAL VOC2012 performance score of 75.8%, to the performance score of 79.7% referenced in [8].

4.2.5. Segmentation using Pix2pix cGAN architecture

As described in Section II, Pix2pix is a DNN utilizing conditional adversarial networks to perform image to image translations. Once it is trained to map a reference image to an output image with structurally similar features, it can be used to translate similar images to the style of the original training output images. This cGAN model was modified by adding extra layers and reducing the number of color channels to perform precise segmentation on infrared images consisting of noisy backgrounds and an object with small, low intensity features. Pixel intensities within the object in these infrared images can also be identical to the pixel intensities of the background noise, which often cause other methods to fail. This model was trained on the same 68-image set as the simple neural network and DeepLab ResNet models.

The modified pix2pix model trained for 10.65 hours to a total of 4000 epochs, the DeepLab v2 ResNet model trained for roughly 34 hours to a total of 120000 steps, and the Matlab neural network trained for around 30 seconds to a total of 48 epochs. Figure 11 shows the comparison of the segmentation results of the above methods. The Matlab neural network, DeepLab v2 network, and modified pix2pix network were all trained using an identical dataset. Both the Matlab network and DeepLab v2 network contained only two classes; background and foreground. Black and red was used to classify background and foreground in the DeepLab v2 network, the other segmentation methods utilized only the colors black and white.

The modified pix2pix model has shown to be able to generate significantly more accurate segmentation masks than the thresholding methods, Matlab GraphCut, DeepLab v2 model and the simple neural network model. Although the simple neural network achieved an average XOR percentage of 9.2048% with less than a minute of training, manual input is required. Manual selection of coordinates to create a bounding box around the object is needed for both training and prediction. The simple neural network also has difficulty with segmenting fine features around the low intensity regions.

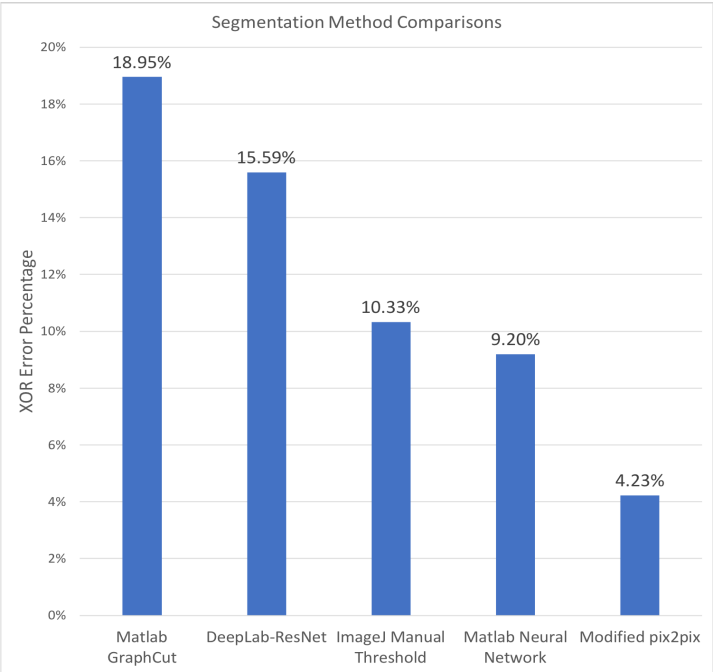


Figure 11: Comparison of the segmentation results of GraphCut, DeepLab-ResNet, ImageJ Manual Threshold, simple neural network and the modified Pix2pix methods.

The segmentation processing times were comparable between the DeepLab v2 network and the modified pix2pix network at less than a second per frame. These two networks utilized a 1080TI GPU to perform the segmentations. The simple neural network segmented frames at a rate of ten seconds per image on an Intel Xeon(R) E5-2620 v2 CPU.

The modified pix2pix model segmented objects most accurately compared to the other neural networks, the Matlab GraphCut tool and automatic iterative inter-means thresholding [4]. Many small and low intensity features were classified correctly by the pix2pix model. However, the nature of the data tested contained only one object class with noisy backgrounds; whereas the semantic segmentation DNN models tested here are concerned with accurately classifying a large range of objects in all of its classes. Some failure modes of the image translation DNN for precise segmentation have been identified where a unique orientation of the object is not well represented in the training set.

V. CONCLUSIONS

The results from the modified pix2pix architecture have demonstrated the successful application of pix2pix, an image translation DNN, to precisely segment objects from a noisy background in infrared wavelength images. Low intensity features and gradient edges are clearly extracted by the modified pix2pix model where methods like iterative inter-means thresholding, manual thresholding, and GraphCut generally fail. The pix2pix DNN can be trained by a relatively small number of training images due to the cGAN architecture. It demonstrated robust object detection, identification and precision segmentation in highly cluttered backgrounds. Other DNN architectures focused on semantic segmentation are able to generate segmentation masks better than thresholding methods but often miss major features of the object.

ACKNOWLEDGEMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology and was supported in part by the U.S. Department of Defense, Test Resource Management Center, Test & Evaluation/Science and Technology (T&E/S&T) Program under NASA prime contract NAS7-03001, Task Plan Number 81-12346.

REFERENCES

1. Lewis, G., “Object Detection for Autonomous Vehicles”, (2014).
2. Teichman, A., and Thrun, S., "Practical object recognition in autonomous driving and beyond," Advanced Robotics and its Social Impacts, Half-Moon Bay, CA, pp. 35-38, (2011).
3. Boykov, Y., Veksler, O., and Zabih, R., "Fast approximate energy minimization via graph cuts", Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, 1999, pp. 377-384 vol.1.
4. Ridler, T., and Calvard, S., "Picture Thresholding Using an Iterative Selection Method," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 8, no. 8, pp. 630-632, (1978).
5. Lu, T., Chao, T-H., Chen, K., Leung, A., Dewees, M., Yan, X., Chow, E., and Torres, G., "Cross-correlation and image alignment for multi-band IR sensors", Proc. SPIE 9845, (2016).
6. Lu, T., Luong, A., Heim, S., Patel, M., Chen, K., Chao, T.-H., Chow, E., and Torres, G., “Intelligent multi-spectral IR image segmentation”, vol. 1020303, no. May 2017, p. 1020303, (2017).

7. Lu, T., and Mintzer, D., “Hybrid neural networks for nonlinear pattern recognition,” Optical Pattern Recognition, ed. by F. T. S. Yu & S. Jutamulia, Cambridge University Press, (1998).
8. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A.L., "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 834-848, (2018).
9. Ronneberger, O., Fischer, P., Brox, T., “U-Net: Convolutional Networks for Biomedical Image Segmentation”, Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer, (2015).
10. Shelhamer, E., Long, J., and Darrell, T., “Fully Convolutional Networks for Semantic Segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 4, pp. 640–651, (2017).
11. Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. “Image-to-image translation with conditional adversarial networks”, CVPR, (2017).
12. Bengio, Y., Courville, A., and Vincent, P., "Representation Learning: A Review and New Perspectives", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, (2013).
13. Kingma, D., and Ba, J., “Adam: A method for stochastic optimization”, ICLR, (2015).
14. Willmott, C.J., and Matsuura, K., “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” Climate Research, vol. 30, no. 1, pp. 79–82, (2005).