The Virginia Journal of Business, Technology, and Science
VaJBTS

SAR MOCOMP by Machine Learning

Christensen, Tamminga, and Chang

Azusa Pacific University

April 8, 2021

Abstract

All unmanned aerial vehicles that use synthetic aperture radar (SAR) systems are equipped with inertial navigation systems (INS) to reduce motion error. Additional motion compensation (MOCOMP) from the data itself is still necessary to achieve required accuracy of a SAR. An affordable method for small drones has yet to be created. We propose machine learning with deep convolutional neural network (CNN) to extract motion error such as sway (right and left) and surge (forward). Results show that the CNN is capable of recognizing gradual drone motion deviations. It has the potential to pick up sudden motion error as well, overcoming major deficiencies of traditional MOCOMP methods, and the need for INS.

1 Introduction

For radar imaging, long and straight antennas produce better resolution. Synthetic aperture radar (SAR) mimics long antennas by sending multiple radar pulses from a short antenna traveling across a distance equivalent to the size of the synthetic "large" antenna. SAR is useful for ground imaging when it is mounted on aerial vehicles like airplanes, helicopters, and satellites at appropriate altitudes [1,2]. SAR allows for weather-resistant and any time of day imaging, with resolution comparable to optical imaging, as long as the radar platform maintain a steady velocity in a straight trajectory. Unmanned aerial vehicles (UAV) fly lower and weigh less than other space-borne and larger airborne vehicles and are more susceptible to turbulence and trajectory deviation. This distorts data even when UAV are equipped with inertial navigation system (INS)/global positioning system (GPS). Highly accurate INS is expensive and heavy for light-weight drones and the motion compensation dependent on INS for data extraction is for gradual (across 1 meter) deviations. Other existing algorithms for radar data extraction mainly correct for sudden motion deviations, leaving out the gradual deviations.

An alternative solution to the motion correction of SAR imaging is presented in this paper. We propose machine learning technique, namely a deep convolutional neural network (deep CNN). Neural networks have been used to classify subjects of optical images such as objects, places, and signatures, and has been used to improve the resolution of optical images [3,4]. A CNN was chosen because it minimizes weights by assembling simple patterns of weights to save computational time [5,6]. Neural nets rely on randomized gradient decent techniques to minimize the error of the prediction and actual sway by training with data sets and the answer key; no hardware is required. When the neural net is trained with lots of simulated data, the resulting algorithm is generalizable to various real data. A set of radar data holds the information necessary to train the net for deviation prediction. Our CNN primarily pick up gradual motion errors in radar data to predict the path deviation, leaving the more rapidly varying errors for traditional MOCOMP algorithms. This technique is a lighter and cheaper alternative to INS.

2 The background of SAR and SAR motion error

In this simulation, the radar is directed perpendicular to the flight trajectory. Point target reflectors backscatter the radar signal and their range is estimated by the time at which the signal returns. Surge does not greatly impact the range estimate, because its motion is perpendicular to the signal. Drones fly over fields of interest at altitudes of ten to several hundred feet. When the range is larger than the height of the drone the effective heave is also small compared to sway. Because surge and heave produce less motion error, this research is focused on predicting sway, although the discussion below is more general and encompasses both sway and surge.

Figure 1 illustrates the motion error with sway and surge. If the flight is expected to be along the x-axis, at constant height z, the actual path, starting from the origin o, is the expressed as $D(x', y', t)$. Expected position is $D(x_0, y_0, t)$. The

motion deviation, a Lorentzian-like Power Spectral Density (PSD) was used. The PSD is an inverse-power law but with an appropriate cutoff parameter at small wave numbers. A power of three was chosen for sway. The final solution for sway looks like

$$\text{Sway} = \mathcal{R}\text{e}\left(\mathcal{F}^{-1}\left(\sqrt{\text{PSD}} * \text{Z}(j)\right)\right), \quad (1)$$

where it takes the real part of the inverse Fourier transform, $\mathcal{F}^{-1}$ of the square-root of the PSD multiplied by Z(j), the unit variance Gaussian distribution of random complex numbers. The j refers to the index in the along-track wavenumber domain.

A different motion error is generated for each dataset, simulated on Matlab. Since the signal beam decays with increasing angle, the beam function should look like a sinc-function, or roughly a Gaussian function. Targets of unknown material will have a variety of phase shifts, which can be expressed in a complex number. The final solution for the received signal is

$$\text{Sig(i)} = \text{B}(\theta_i, \text{b})\ \text{P}(\text{r}_p; r_i, \text{w})\ \text{O}(\text{r}_i)\ \text{H(i)}, \quad (2)$$

where i refers to a specific target. The B is the Gaussian beam function, determined by the target angle and beam width. The P is the Gaussian Pulse function, determined by the pulse length, the target placement, and the pulse width. The O is the complex phase function for the propagation to and from the target. The H is the Gaussian distribution of random complex numbers with unit variance. The signal, will have a slightly altered amplitude and a phase shift, complying with the natural properties of each ground target reflector. The dataset is the sum of the signals of each target at each position of the drone.

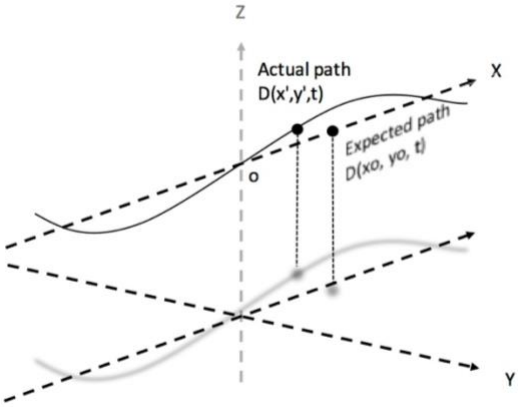

Figure 1: The sway and surge. The expected path is along the x-axis whereas the actual path deviates from this. The two dots refer to the position expected at time t and the actual position.

3 The background of Convolutional Neural Network

For the CNN, Tensorflow was the engine, and NumPy and matplotlib were computational tools used in a python environment. Convolutional layers save computational time by introducing filters to convolve the data. These filters minimize weights by sliding across a whole dataset (with stride set as two) to produce feature maps based off each filter's characteristics. The CNN consists of seven layers, a convolution layer, a maxpooling layer, a second convolution layer, a second maxpooling layer, a third convolution layer, a flatten layer, and finally, a dense layer.

The maxpooling layer is between each convolutional layer to decrease the resolution to about one-fourth, saving computation time. This is done by taking four neighboring pixels and replaces it with a single pixel of maximum strength. The flatten layer transforms the square datasets into a 1-D array to be used in the dense layer. The dense layer will make the CNN densely connected to the final output, so that all the data is used to predict sway. The CNN will output an array of 1x200 for each data set which will be a function of position x.

Table 1 is a summary of the CNN model. In the second column, the dimensions of the filters are given. The first two numbers are filter sizes and the third number is the filters. The same applies to the third column, which are the feature maps. The fourth column is the count of all the weights of each layer. The total number of weights are listed in the bottom, which are all trainable. The model was trained with 8,000 datasets along with their sway. Each epoch is responsible for minimizing the cost function by randomly selecting parameters to adjust, averting local minimums. The cost function does not change appreciably after ten epochs. Tensorflow compared the predictions to the actual deviation information and made adjustments to the weights to minimize the error. The other two-2000 datasets were used to test the trained model.

Model: sequential			
Layer (type)	Filter size	Output shape	Parameter number
Convolution 2D (1)	(9,9,48)	(96, 96, 48)	3936
MaxPooling (1)	(None)	(48, 48, 48)	0
Convolution 2D (2)	(4,4,32)	(23 23, 32)	24608
MaxPooling (2)	(None)	(11, 11, 32)	0
Convolution 2D (3)	(3,3,24)	(9, 9, 24)	6936
Flatten	(None)	(1944)	0
Dense	(None)	(200)	389000
Trainable parameters: 424, 480		Total parameters: 424,480	
Non-trainable parameters: 0		Epochs: 10	

Table 1: A summary of the CNN. The types, filter size, output shape, and parameter numbers are all listed. On the bottom all the parameters are listed as trainable. The number of epochs are given.

4 Results

The simulated radar data is a 200x200 dataset collected along a track distance of 100 meters. The range of the targets are 20-120 meters. Arbitrary target placement is represented by bright dots on the scene, which are proportional to the absolute value of its signal strength in Fig. 2a. The corresponding raw radar data in an ideal, error-free environment has smooth and symmetric arches which are the radii from a target to the drone at each along-track position (Fig. 2b). The peaks correlate to the target positions and agree with Fig. 2a. In a natural setting, raw radar data has motion error, as shown in Fig. 3a. The arches are distorted by sway and surge. At each position along the track, the radius of each target has the same sway and surge displacement since each signal is offset the same amount by the drone's movement. The dotted lines in Fig. 3 show where the bumps in the predicted sway (b) visibly affect the radar data (a). The position along the zero in (b) refers to the ideal trajectory, which the blue function deviates away from, at a maximum of -1 meter range at position 150. Along-track trajectory is calculated in increments of every half-meter which corresponds to the resolution. The dotted lines show the visible effects that sway has on the data. Peaks appear where there are sudden changes in direction of sway. Gradual changes are not as easily distinguished. Nonetheless, the CNN uses these distortions to predict a sway similar to Fig. 3b.

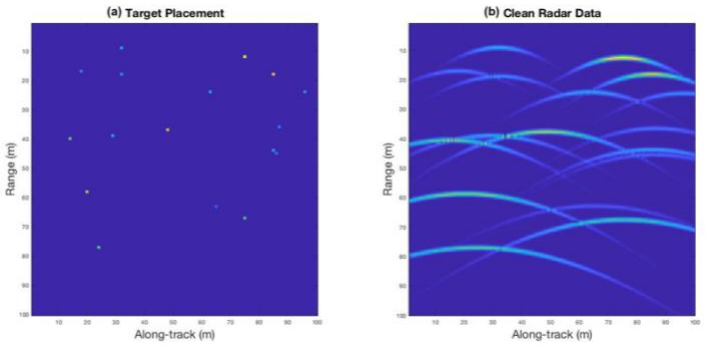

Figure 2: (a) Arbitrary target placement and signal strength (brightness). (b) The raw radar data collected in an ideal, motion-error-free environment.

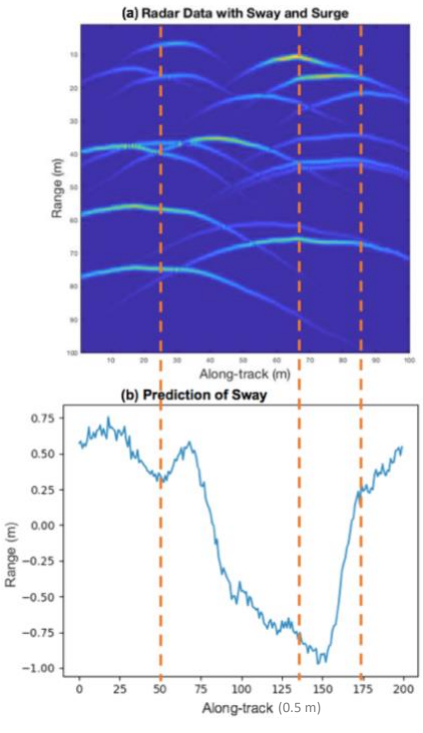

Figure 3: (a) The raw radar data collected in a non-ideal motion-error environment. (b) The prediction of the sway motion error.

Multiple datasets and their corresponding original sway (red) and predicted sway (blue) is shown in Fig. 4. The predicted sway graph is the same track-distance as the data but elongated to show detail. The test datasets have their own unique placements of targets and sway which the CNN had never trained with before. The predictions follow the general pattern of the real sway. Unlike standard SAR MOCOMP techniques, the CNN picks up gradual motion deviations easily. The red lines tend to peak faster than the blue, because the CNN is thus far incapable of predicting the rapidly varying sway.

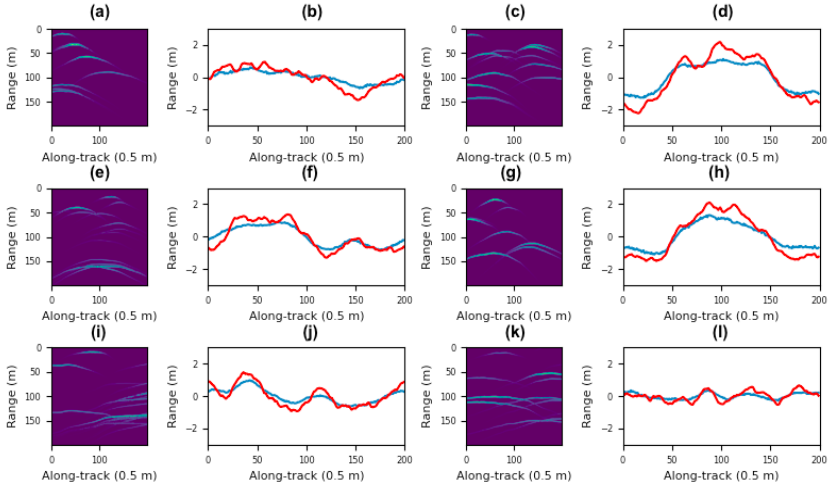

Figure 4: Examples of radar data with predictions on their right. The real sway (red) and the prediction (blue) are both normalized so that the average position is zero.

Possible training to pick up rapidly varying sway remains inconclusive. But traditional MOCOMP can already compensate for this and could possibly reinforce the CNN.

The convolutional filters grab the "key elements" of the data to find sway. Figure 5 shows six of the 48 filters from the first convolutional layer. The combined weights create unique gashes, bars, and holes (see orange circles). What data they single out can be seen by their feature maps. To the right of each filter is a feature map they produced from a single scene. The filter a.i picks up positive slopes, because its feature map only shows the left side of the data arches, and f.i appears to

do the same thing. Filter, e.i, picks up the negative surroundings by creating a feature map that is mostly light purple.

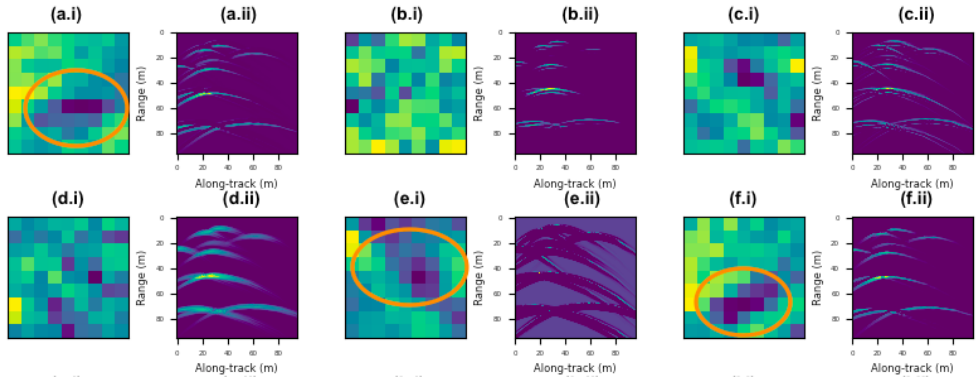

Figure 5: Six different filters (a.i-f.i) of the first convolutional layer with size 9x9. Bright pixels are a higher value than the darker pixels. Their corresponding feature maps (a.ii-f.ii) are of size 96x96.

The data collected for two target rich scenes and two target lacking scenes is shown in Fig. 6. For the top two predictions, they predict accurately except at the previously mentioned sharp humps, which is the case for all the predicted sway. For the bottom two predictions, they predict just as accurately as the top data. Unless there are weak targets in Fig 6.e and 6.g that are observable only to the CNN, it is robust with few targets.

5 Conclusion

This paper summarizes the characteristics and results of a machine learning technique to predict motion error. The CNN produces predictions resembling the gradual motion deviations in the sway model. The CNN is robust in handling the range of targets that comprise the datasets. However, the full capacity of the machine learning technique has not been tested. The CNN has room for improvement. The results showed a model with first convolutional layer that had filter sizes large enough to find gradual motion deviation. However, creating smaller filters might force them to match smaller motion deviations. An additional way to improving predictions would involve an iterative process of correcting data with the predictions. This corrected data could be run through a new prediction model to find the residual sway until desired results are produced. Since traditional MOCOMP does not prove useful for gradual motion deviations, a hybrid solution of MOCOMP and this technique is worthy of researching.

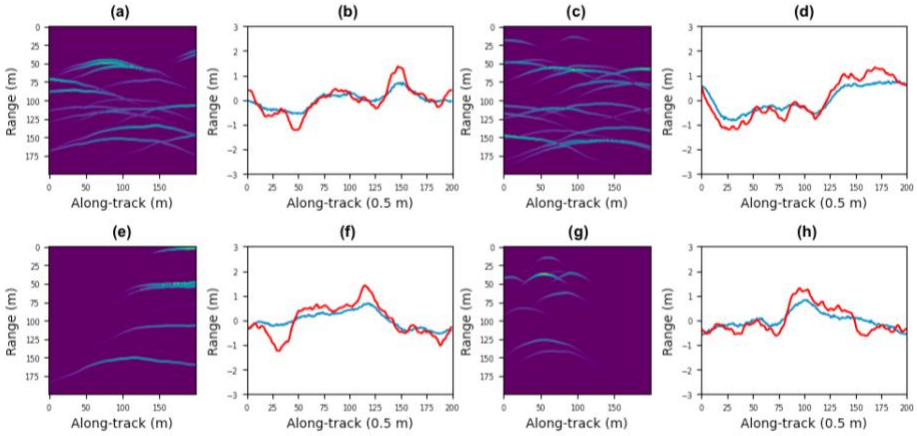

Figure 6: Two data sets with plenty of targets and two data sets with not enough targets. To the right are their predictions.

The machine learning with convolutional neural networking technique is easy to use and quick to run, perfect for aerial platforms such as small drones. Overcoming many of the limitations of traditional MOCOMP, it has the potential to replace these traditional techniques.

References

[1] M. Xing, X. Jiang, R. Wu, F. Zhou, and Z. Bao. “Motion compensation for UAV SAR based on raw radar Data,”. IEEE Trans. Geosci. Remote Sens., vol 47, no. 8, pp 2870-2883, Aug. 2009.

[2] J. R. Moreira, “A new method of aircraft motion error extraction from radar raw data for real time motion compensation,” IEEE Trans. Geosci. Remote Sens., vol. 28, no. 4, pp. 620–626, Jul. 1990.

[3] G. Alvarez, B. Sheffer, and M. Bryant. “Offline Signature Verification with Convolutional Neural Networks,” Stanford University CS231n, 2016.

[4] W. Yang, X. Zhang, Y. Tian, W. Wang, J. H. Xue, and Q. Liao. “Deep learning for single image super-resolution: a brief review,” arXiv:1808.03344v3, Jul 2019.

[5] C. Dong, C. C. Loy, K. He, and X. Tang. “Image super-resolution using convolutional network,” arXiv:1501.0092v3, Jul 2015. https://arxiv.org/pdf/1501.00092.pdf

[6] L. Xu, J. SJ. Ren, C. Liu, and J. Jia. “Deep convolutional neural network for image deconvolution,” http://www.cse.cuhk.edu.hk/~leojia/papers/deconv_nips14.pdf