Tornado Speed Estimation Using Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM)-Based Video Processing Approach

Anirudh Marathe, Prerit Daga, Sudha Radhika, and Yukio Tamura

Abstract Many natural processes in the universe occur in a rotational motion, such as the formation of drastic events including tornadoes and cyclones. For the past few decades, research has progressed to estimate the occurrence of such unpredictable small-scale meteorological events and their damage paths and to estimate the amount damage caused. In the case of short-lived yet disastrous tornadoes, it is possible to track the damage path. However, to estimate the damage through the Fujita Scale (*F*-Scale) or the Enhanced Fujita scale (EF-Scale), it is still necessary to rely on Radar Data Acquisition (RDA) systems working on Doppler effect to estimate the wind speed. For this, the equipment needs to be placed in the vicinity of where the tornadoes form, so they are often at risk of being damaged. Thus, in the current research, the tornado speed is estimated using video processing and AI techniques: Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) conjointly. A video model of a rotating tornado (without translational motion) is artificially generated with a tracking object inside it. The wind speed is estimated by tracking the speed of this object caught in the tornado's whirl. CNN in combination with LSTM effectively predicts the shift of the object in each frame of the video in comparison with a reference frame.

Keywords Wind engineering · Tornado · Video processing · CNN · LSTM · Python

A. Marathe · P. Daga · S. Radhika (✉)
BITS-Pilani Hyderabad Campus, Hyderabad, India
e-mail: sradhika@hyderabad.bits-pilani.ac.in

Y. Tamura
School of Civil Engineering, Research Center of Wind Engineering, Environment, and Energy, Chongqing University, Chongqing, China

Wind Engineering Research Center, Tokyo Polytechnic University Atsugi Campus, Atsugi, Japan

Y. Tamura
e-mail: yukio@arch.t-kougei.ac.jp

S. G. Rajasekharan et al. (eds.), *Proceedings of the 9th National Conference on Wind Engineering*, Lecture Notes in Mechanical Engineering,
https://doi.org/10.1007/978-981-99-4183-4_15

1 Introduction

A tornado is a violently rotating column of air that extends from a thunderstorm and comes into contact with the ground taking a massive shape and creating a lot of damage along the traversed course. These strong winds occur across the world, including the USA which experiences it more often than other regions. Every year, the USA reports about 1300 tornadoes of varying intensities [1]. The intensity of a tornado is measured on Enhanced Fujita (EF) Scale [2], which takes in to account 28 different damage indicators. The speed of tornado can describe tornadoes more accurately. However, it is difficult to estimate the speed due to the massive damage it can cause to the equipment like anemometer that is used to measure wind speed. Radar-based techniques [3], used by WSR-88D and Mobile Doppler Radar, provide some means for prediction and measurement of speed using Doppler effect. The WSR-88D Doppler radar network obtains weather information (precipitation and wind) based upon returned energy generated and received at the Radar Data Acquisition (RDA) unit. This helps in forecasting tornadoes, but due to physical limitations such as beam spreading and radar horizons, the rotation as well as speed of tornado cannot be measured. Mobile Doppler Radars can be positioned close enough to the storm to resolve the rotation within the tornado itself. This method is still not widely used to measure speed. This project aims at using image and video processing techniques to estimate the speed of rotation of still tornadoes. The image processing systems in development and use today have become more robust, accurate, and less expensive with the advancement in digital camera technology. The low cost of development, use and maintenance of digital cameras, and related processing equipment make them widely used for image processing-based applications. Various such techniques are already in use for tracking and measuring the speed of objects in linear motion like vehicles in traffic. Such techniques require capturing of video at known frame rate and observing the motion. They can also be used for the linear path traversed by a tornado. Using similar techniques with modifications, the speed of rotation can be estimated.

2 Methodology

The speed of rotation of an object is calculated as number of rotations completed in unit time. For identifying a complete rotation, we mark a reference point on the rotating object that appears at the same location after certain amount of time. Since a tornado is composed of winds, it is difficult to mark a single or group of identifiable points to make the required observations. Therefore, we choose an object stuck in the tornado and observe its location to identify the speed at which it rotates or appears to rotate, which is assumed to be the speed of the tornado [4].

The proposed methodology follows the approach of tracking an object stuck in the tornado to calculate the speed. Video processing is used to identify the object and its

position in each frame. First, the speed is calculated in terms of no. of frames traversed for unit rotation and then that is converted into the number of rotations per unit time based on the rate of frame capture. Furthermore, as the video obtained through a digital video camera is not continuous, we expect that the speed obtained through this process needs to be scaled to obtain the original speed. This scaling factor is obtained by first calculating the speed of a reference rotating object of known speed and comparing it with the actual speed. This scaling factor will be specific to camera frame rate and can be used to scale the speed obtained for the test object.

In this work, we have used a rotating cone as a simplified model of a tornado and a certain pattern on the cone as the object under observation. This approach helps in simplifying the creation of desired relations that can later be extended to complex tornado simulations.

The various steps that are involved are explained as follows:

2.1 Video Capturing

The video of a tornado is captured from a high-resolution digital camera at a known frame rate. A higher frame rate will ensure that less rotations are skipped while capturing the motion and better accuracy is expected.

2.2 Preprocessing

The video is to be processed frame by frame. Each frame is converted from RGB to grayscale image for performing fast operations.

2.3 Reference Object Selection

A distinguishable object that was stuck in the tornado is selected from a frame of the video in initial run. The location of the object in the frame is noted.

2.4 Reference Object Tracking

As the tornado rotates, the object location in the subsequent frames also changes. On completion of a rotation, the location on the corresponding frame is expected to be same as that in the initial frame. The location will repeat again in further rotations. This seems straightforward for human observer; however, to perform this

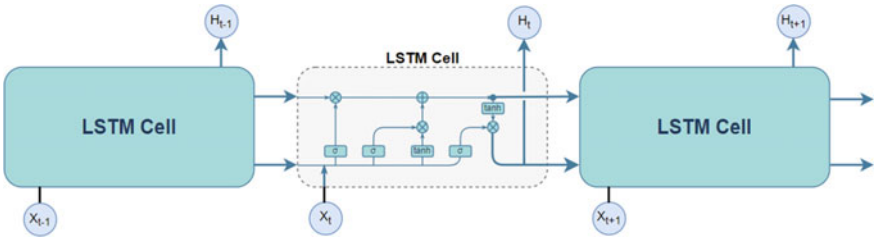


Fig. 1 LSTM internal structure

task through video processing, an object tracking algorithm is used. The results of this step are then used in speed estimation, as explained in next subsection.

The object tracking methodology used in this work makes use of CNN and LSTM that is explained as follows.

A reference object stuck in tornado is selected and cropped from the first frame of the video. To track its position in the subsequent frame, the same set of pixels needs to be searched. Now, since an object may change its orientation and shape while rotating, we cannot directly search the cropped image with good accuracy. Hence, we use Convolutional Neural Network (CNN) to predict the location of cropped image in the subsequent frames [6–8]. The learning method used is Long Short-Term Memory (LSTM) [9, 10].

LSTM. In LSTM, a training set of first few frames is given to the network. It stores in its memory buffer, the information gained from each training frame. Based on the information stored, it can predict the next frame of a video, using backpropagation method. The block diagram of an LSTM network is shown in Fig. 1. Each rectangular box is a neuron/cell. The internal working of an LSTM network is described as follows from left to right direction.

σ represents the sigmoid function which takes input as X_t and H_{t-1}. This sigmoid function is called as the forget gate. The task of the forget gate is to decide how much of the previous output data will be forgotten and how much of the previous data will be used in next steps. Both these inputs are multiplied with their corresponding weights and added and fed to the sigmoid function σ where the output could be in range of 0–1.

The middle σ and tanh function together comprises the input gate and new memory cell. They decide what relevant information can be added from the current step. The inputs to the new memory cell are X_t and H_{t-1}. These inputs are taken and multiplied together with their corresponding weights and fed to sigmoid function. The multiplication output of input gate and new memory cell could be any value between 0 and 1. The previous cell state input and the output from forget gate are multiplied and added with the total output of input gate and new memory cell calculated in the previous step to give the current cell state which would be given to the next cell [10].

The rightmost σ and tanh function together comprises the output gate. This gate tells the amount of relevant information to be taken from the cell state. The inputs X_t

and $H_{t\text{-}1}$ are taken and multiplied together with their corresponding weights and fed to sigmoid function. The output of sigmoid function and the final value of cell state obtained from by passing the cell state through the tanh function are both taken and multiplied to produce the hidden state output which would be given to the next cell.

2.5 Speed Estimation

Let r_{obs} denote the number of rotations observed in f_{obs} frames. This gives speed in terms of number of rotations per frame. Multiply this with frame rate fps (frames per second) to obtain the observed speed, s_{obs}, in rotations per second, Eqs. (1 and 2).

$$s_{\text{obs}} = \frac{r_{\text{obs}}}{f_{\text{obs}}} \times \text{fps}, \tag{1}$$

$$s_{\text{obs}} = r_{\text{obs}}/\text{sec}. \tag{2}$$

Eq. (2) is called as the "observed speed", as a camera setup with frame rate less than the actual speed of tornado can skip a few rotations. Thus, a scaling factor is calculated from a reference training video, which is used here to scale the observed speed into actual speed, Eq. (3).

$$\text{scale} = \frac{r_{\text{actual}}'/\text{sec}}{r_{\text{obs}}'/\text{sec}}. \tag{3}$$

The scaling factor of Eq. (3) is obtained by first running the model on a video of an object of known speed. Then, we calculate the speed of rotation of desired object, s_{cal}, using Eq. (4).

$$s_{\text{cal}} = \text{scale} \times s_{\text{obs}}, \tag{4}$$

3 Experimental Setup and Results

The proposed method was implemented in Python, on a video sequence of a tornado model represented by a rotating cone Fig. 2, simulated through VPython [4], a python-based programming language to write programs that generate navigable real-time 3D animations.

The number of rotations to be simulated were manually given, and based on the time taken for the completion of simulation, the speed was calculated in rotations per second (rps) [5]. This gives 'actual speed' or 'known speed' [9]. Since the idea is to identify complete rotations of the cone, a texture was applied to the cone image.

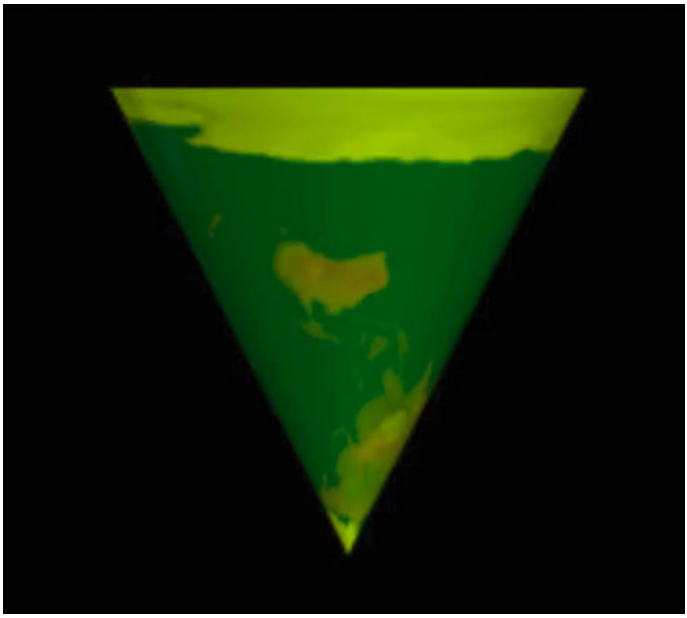


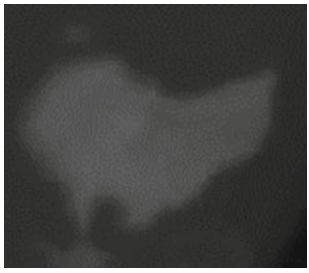
Fig. 2 Frame of simulated cone

The texture consisted of an inverted map of the Earth, on which any known and distinguishable place could be identified in each frame. Screen recorder was used to obtain the video of the rotating cone. The 'frame rate' of the screen recorder was taken to be 30 fps.

The reference object image chosen is an inverted map of Australia, Fig. 3.

The location of image and frame number is noted and compared. The difference between the first two frames that contain the reference image at almost the same location as the first frame is calculated, and subsequent frame numbers are estimated.

Fig. 3 Reference object



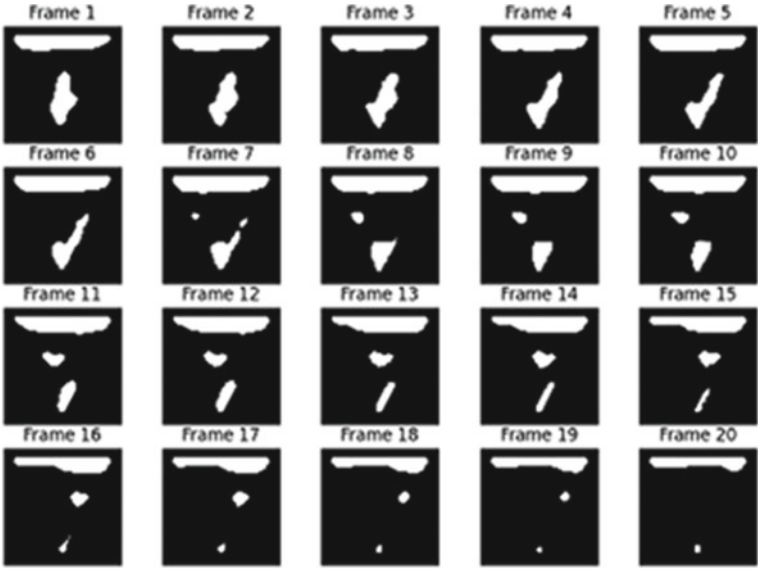


Fig. 4 Dataset for training

3.1 Results for LSTM Network Next Frame Prediction Computation

A dataset is created by taking 6000 image frames from a sample video. Out of these, 20 images are taken as training dataset, Fig. 4.

After the model has been trained, all the weights of trained model are then provided as the input to the prediction module. The LSTM model is tested for the next frame prediction. And, the next frame prediction is done with a frame rate of 5 fps as shown in Fig. 5.

3.2 Scale Calculation from Video of Known Speed

The model was first run on a rotating cone with speed = 4 rps. Figure 6 shows a frame where the reference object is marked.

Table 1 shows the estimated and actual values of frame numbers along with the location of the reference image.

Each row in the table corresponds to a calculated rotation of the object image which is identified with position. From Table 1, we observe that though the first rotation takes about 14 frames per observed rotation the subsequent rows show an average of about *15 frames per observed rotation.* Using this value and frame rate of 30 fps in Eq. (2), we get

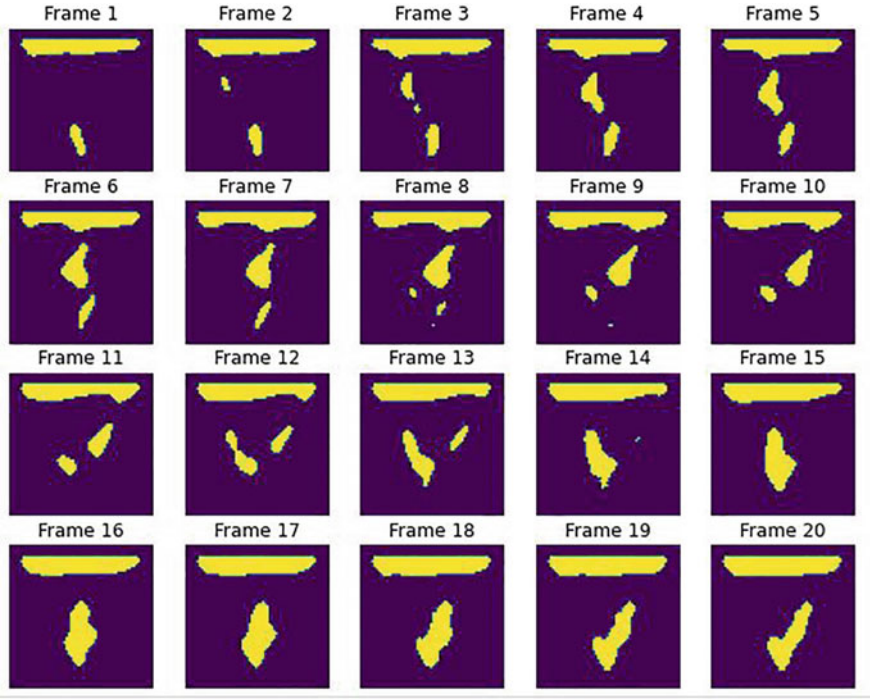


Fig. 5 Predicting the next video frames using dataset for the trained model

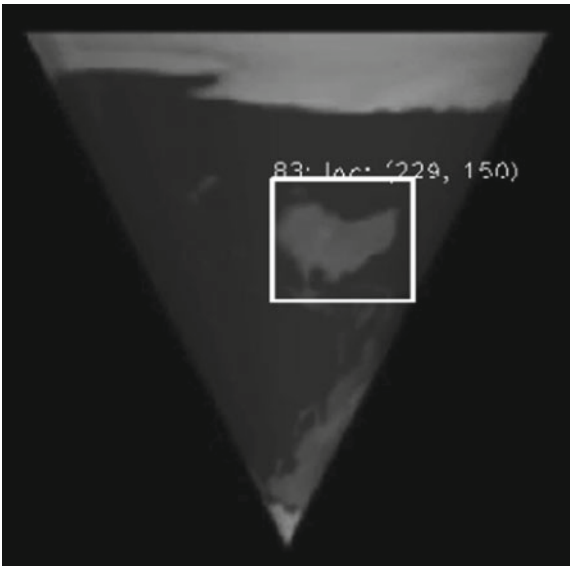


Fig. 6 Reference object identified and marked

Table 1 Predicted and actual values of frame numbers along with the location of the reference image (for reference video)

S No.	Frame No.		x-location	Frame difference	Remarks
	Predicted	Actual			
1	–	8	251	–	
2	–	22	253	14	
3	36	37	198	15	
4	50	52	241	15	x-location to far to be considered as complete rotation
5	64	60	214	8	
6	78	76	255	16	
7	92	91	249	15	
8	106	106	243	15	
9	120	121	237	15	

$$s'_{\text{obs}} = 2\,\text{rps}. \tag{5}$$

We use this value and the actual speed of 4 rps in Eq. (4) to obtain scaling factor, Eq. (6):

$$\text{scale} = \frac{4\,\text{rps}}{2\,\text{rps}} = 2. \tag{6}$$

Thus, one calculated rotation at a frame rate of 30 fps is equivalent to two actual rotations. We used this factor to estimate the speed of another rotating object.

3.3 Speed Estimation for Video 2

A rotating cone was again simulated with a speed of rotation = 2.11 rps. This was captured as a video at 30 fps. Table 2 shows the various values of frame location for each match of reference object image.

Following the procedure mentioned earlier, we obtained *estimated speed* $\approx$ 2.14 rps. This shows that our model estimates the speed with about 98.6% accuracy.

Table 2 Predicted and actual values of frame numbers along with the location of the reference image (for test video)

S No.	Frame No.		*x*-location	Frame difference
	Predicted	Actual		
1	–	7	212	–
2	–	33	205	14
3	59	63	198	15
4	85	78	220	15
5	111	108	214	8
6	137	138	207	16

4 Conclusion

The proposed method can be used to measure the rotational speed of a tornado from a video sequence. The method works quite well for simulated models of tornado and can be extended to tornados in real world, with constraint of objects stuck in the tornado being identifiable by the code. Since models are usually of lesser speed as compared to the actual tornados, videos captured at higher frame rate can be helpful. This work does not consider the orientation and position changes of the reference object and the hiding and unhiding of the object that happens in real tornado. A more robust object tracking can be used to enhance the results on real tornados.

References

1. Radhika S, Tamura Y, Matsui M (2012) Use of post-storm images for automated tornado-borne debris path identification using texture-wavelet analysis. J Wind Eng Ind Aerodyn 107:202–213
2. Sabareesh GR, Matsui M, Tamura Y (2011) Characteristics of surface pressures on a building under a tornado-like flow at different swirl ratios. J Wind Eng 8(2):30–40
3. Radhika S, Tamura Y, Matsui M (2017) Application of remote sensing images for post-wind storm damage analysis. In: Remote sensing of hydro-meteorological hazards. 1st edn. Taylor & Francis
4. Sabareesh GR, Matsui M, Tamura Y (2013) Characteristics of internal pressure and resulting roof wind force in tornado-like flow. J Wind Eng 112:52–57
5. Elhamod M, Levine MD (2013) Automated real-time detection of potentially suspicious behavior in public transport areas. In: IEEE transactions on intelligent transportation systems, vol 14(2), pp. 688–699
6. Freer JA, Beggs BJ, Fernandez-Canque HL, Chevrier F, Goryashko A (1996) Automatic video surveillance with intelligent scene monitoring and intruder detection. In: 30th annual 1996 international carnahan conference in security technology in security technology, pp 89–94
7. Lai TY, Kuo JY, Liu C-H, Wu YW, FanJiang Y-Y, Ma S-P (2013) Intelligent detection of missing and unattended objects in complex scene of surveillance video. In: 2012 international symposium on consumer and control in computer (IS3C), pp 662–665
8. Chowdhury A, Tripathy SS (2014) Detection of human presence in a surveillance video using fuzzy approach. In: 2014 international conference on signal processing and integrated networks (SPIN), pp 216–219

9. Bird ND, Masoud O, Papanikolopoulos NP, Isaacs A (2005) Detection of loitering individuals in public transportation areas. In: 2005 IEEE transactions on intelligent transportation systems, pp 167–177
10. Ajitha A, Goel M, Assudani M, Radhika S, Goel S, Design and development of residential sector load prediction model during COVID-19 pandemic using LSTM based RNN. Electric Power Syst Res