Vortex detection, tracking and trajectory prediction: a machine learning approach

May 10, 2015

1 Introduction

The coherent vortices in two-dimensional turbulence has been a subject of intense study as its importance in many geophysical fluid systems. For example, the Earth weather is strongly influenced by tropical cyclones in lower latitudes and blocking systems in higher latitudes (Fig. 1 left). In the Earth ocean, the ocean eddies play an important role in transporting heat and nutrients (Fig. 1 middle). Jupiter's atmosphere features hundreds of vortices, notably the Great Red Spot, which has existed for at least three hundred years (Fig. 1 right). Despite the ubiquitousness of the vortices, our current understanding about them is very limited. One big challenge to study these vortices is that although these oval-shaped structures are easily picked up by eyes, it is difficult to tell a computer to detect and track them. Therefore, although the vortices can be well simulated by the state of art numerical models, it is difficult to analyze the vast amount of data (Guido Boffetta1 and Robert E. Ecke, 2012, Two dimensional turbulence). The current vortex detection and tracking methods are mostly ad-hoc and can only be applied to the data it is developed for. For example, in decaying turbulence, the McWilliams vortex census procedure are usually used, which first identify extrema as possible vortices and then apply a set of criteria and eliminate the false positive ones (1990). A threshold vorticity need to be specified otherwise there is no isolated extrema. This method is only suitable for decaying turbulence, in which vortices are very coherent and isolated, and have much larger vorticity than the background. For hurricane detection, another set of criteria is developed, which is based on vorticity and maximum wind speed (2009, Zhao). The tracking algorithm is even less well-developed. It usually involve connecting nearby vortices in two consecutive frames. However, considering the complicated dynamics of vortices involving merging and splitting, and the positive false alarm and undetection, such an naive tracking algorithm has a lot of shortcomings.

While vortex detection and tracking remain a major challenge in the fluid dynamics community, there are significant breakthroughs in the machine learning community dealing with similar problems. The facial recognition is an intensively studied problem which is quite similar to vortex detection. The cascading classifier first developed by Viola and Jones (2001) has realized real time facial detection with a detection rate of 95%. The classifier can be trained to detect other objects, so it it promising to use it to detect vortices. In modern multiple target tracking

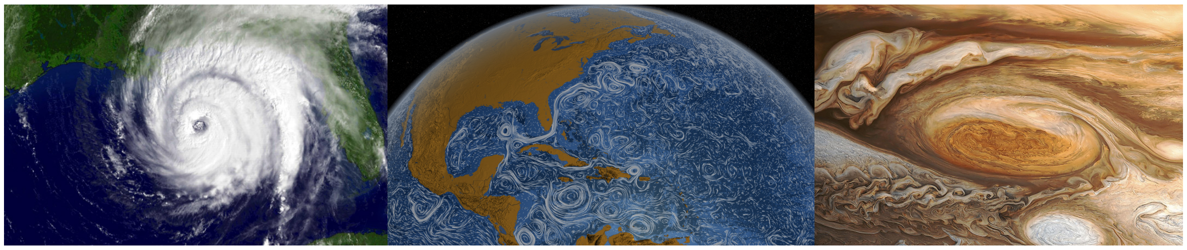

Figure 1: (Left) hurricane in Earth's atmosphere (from NOAA, http://www.noaanews.noaa.gov/stories2005/s2438.htm); (middle) animation of ocean surface currents in the Atlantic Ocean, and (right) the Great Red Spot on Jupiter, which has existed for over three hundred years (from NASA, http://ircamera.as.arizona.edu/NatSci102/NatSci102/lectures/jupiter.htm, http://www.nasa.gov/topics/earth/features/perpetual-ocean.html#.VU2El3UVhBc).

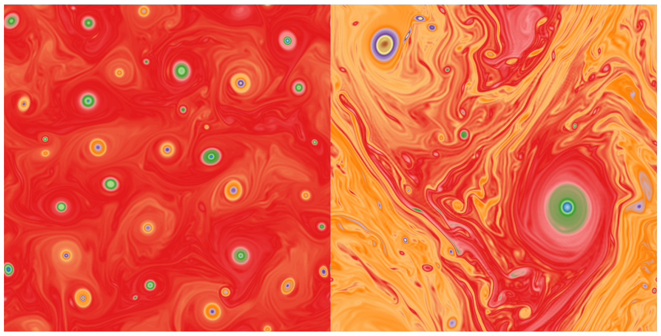

Figure 2: Visualizations of vorticity field from (left) DataI and (right) DataII. The colormap is chosen so that vorticity gradient is highlighted.

systems, it is widely accepted that the multiple hypothesis tracking (MHT) is the preferred method (Blackman 2003). MHT can be also used to track vortices if properly set up. Both cascading classifier and MHT take an probabilistic approach, which makes them generic and robust. In this work, we aim to apply these methods to detect and track vortices from a set of numerical simulations.

In a addition to detection and tracking, a further problem is prediction the movement of vortices. Although there is sophisticated fluid dynamic model to do that, it is time consuming. Instead, statistical models are orders of magnitude cheaper once properly trained. Furthermore, statistical models may provide some insight into the vortex motion apart from the complicated fluid dynamics equations. In this work, we choose to predict the hurricane trajectories, which have a huge social and economical impact.

2 Datasets and pre-processing

We have two set of simulation data from a numerical fluid dynamics model with different settings. Each data set contains the vorticity field for 2000 consecutive times with resolution 256×256, and it is stored in NetCDF format. One data set, denoted as DataI hereinafter, is from a decay turbulence simulation and thus features well separated vortices as illustrated in Fig. 2 (left). The other data set, denoted as DataII thereinafter, is from a evolving turbulence simulation. As illustrated in Fig. 2 (right), the fluid motion is much more energetic and contains vortices with widely different sizes, less well separated and less coherent.

The positive training samples for cascading classifiers, i.e., vortices examples, must be prepared manually. To assist preparing training samples, we developed a helper program with a Graphical User Interface using Python and Tkinter as shown in Fig. 3. This program can visualize vorticity field read from NetCDF file on a canvas, and allow user to zoom in and zoom out on the canvas. The user can then select a vortex by dragging a box enclosing the vortex with mouse. After confirming the selection by double-click, the location of the box is output and the vorticity field inside the box can be further converted into images using different colormaps. We find this program to be very useful and may be relevant for other object detection studies. It is made open sourced and can be find on our Github repository (`https://github.com/mathsam/vortex_tracker`).

Ideally, we would like to apply our vortex detection and tracking algorithms to the real meteorological data and find out the trajectories of the hurricanes, and then make prediction on the hurricane paths. However, to expedite the process of this project, we did the trajectory prediction in parallel with vortex detection and tracking. Therefore, we downloaded historical hurricane trajectories from the National Hurricane Center of the National Weather Service (`http://www.nhc.noaa.gov/data/`). The data contains hurricane trajectories from 1851-2014, and we used part of the data from 1979, which is more reliable since satellite observations is used. Each trajectory contains name/id of the hurricane, its position in logitude/latitude format at 6 hours frequency, and its maximum wind speed.

3 Vortex detection

We used the cascade object detection algorithm first proposed by Viola (2001) and later extended by Rainer (2002). The detection algorithm works on gray scale images. It first transform pixel information into a set of Harr-like

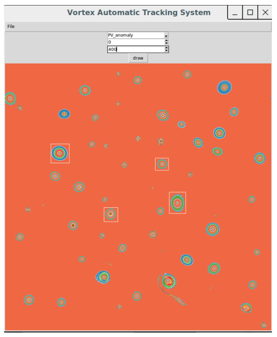

Figure 3: The helper program which visualize the vorticity field read from the NetCDF file, and each vortex can be selected by mouse and thus be logged.

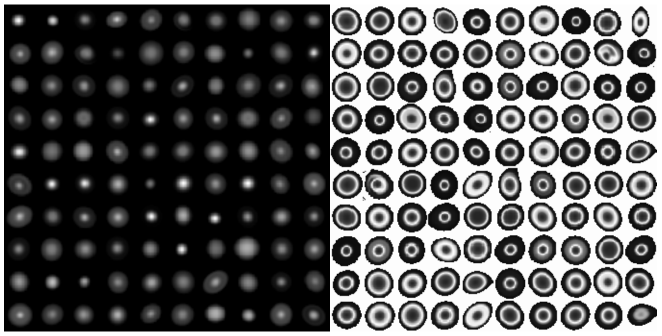

Figure 4: Sample training vortices prepared using (left) linear colormap and (right) nonlinear colormap which creates more edges and highlights the gradients in vorticity field.

features, which are basically the difference of sums of pixel intensity in adjacent rectangular regions. Each Harr-like feature is only a weak classifier, therefore, the detection algorithm uses AdaBoost to select the best features and then construct a strong classifier using the weighted sum of weak classifiers. These strong classifiers are then combined into a degenerated decision tree, which is referred as the cascade of classifiers. In this way, the first layer of the cascade rejects most of the uninteresting regions and then the next stage can focus on the selected regions, thus spending most of the time on regions with high probability to be object. This algorithm has been implemented in OpenCV library (2000), which provides API in C++ and Python. We used this library extensively in our vortex detection task.

The cascade detection algorithm works on gray scale images, therefore, we first need to transform vorticity field, which is an array of float numbers ranging from about -2000 to 2000, to an array of integers ranging between 0 to 255. The straight forward method is to linearly rescale the vorticity field as $255 \times (\zeta - \zeta_{min})/(\zeta_{max} - \zeta_{min})$ and truncate it to integers. As we do not differentiate between cyclonic and anti-cyclonic vortices (vortices with positive and negative vorticity at the center), we use the absolute value of the vorticity field before rescaling. Sample training vortices prepared using this linear colormap is shown in Fig. 4 (left).

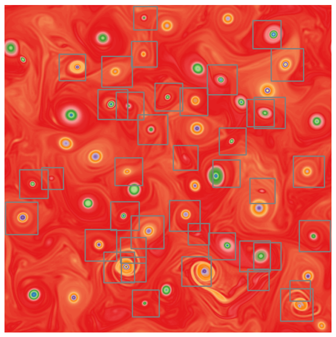

Figure 5: An example of vortices detected by our trained cascade classifier on DataI. The rectangular denotes the area which is identified to be vortex by the classifier.

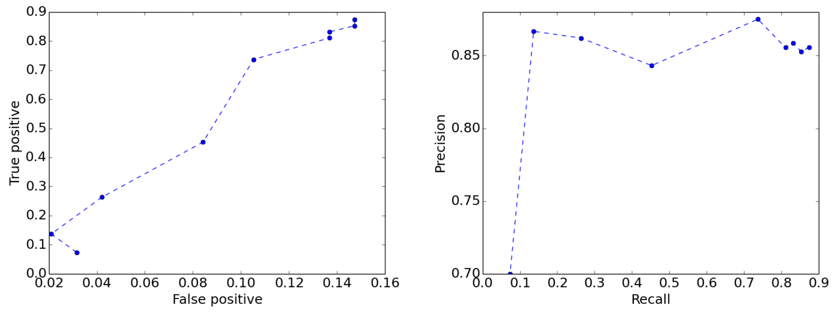

Figure 6: (Left) ROC and (right) precision-recall curve tested on DataI using linear colormap transformation.