21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

An Effective Vortex Detection Approach for Velocity Vector Field

Shen-Shyang Ho
School of Computer Engineering
Nanyang Technological University
Singapore 639798
ssho@ntu.edu.sg

Abstract

Detection of vortices, which are rotating flow features, is an important task to identify, analyze, and understand flow dynamics in a fluid. For example, it can be used to accurately tag nonrigid salient rotation features from large amount of wind vectors captured by orbiting satellites for hurricane research. In this paper, we describe in detail a general vortex detection algorithm motivated by Hough transform and flow vector tree structures. The vortex detection algorithm allows one to find the exact vortex center efficiently if it is in the vector field. A special case of the algorithm has been successfully applied to cyclone annotation and tracking using QuikSCAT satellite wind measurements.

1. Introduction

A vortex is any circular or rotating flow in a fluid. The amount of rotation at each point for a given 3D velocity vector field $\vec{f} = (u, v, w)$ is quantified as following.

$$\vec{\omega} = \nabla \times \vec{f} = det \begin{vmatrix} \hat{e}_x & \hat{e}_y & \hat{e}_z \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u & v & w \end{vmatrix} \quad (1)$$

such that $\hat{e}_x$, $\hat{e}_y$, and $\hat{e}_z$ are unit vectors along x-, y-, and z-axis, respectively. The vorticity $\vec{\omega}$ is the curl of the velocity field. The direction of $\vec{\omega}$ is along the axis of rotation.

Vortices are very common in fluid (e.g. water and air) flows. The identification of vortices in fluid is important for the analysis and understanding of flow dynamics in the oceanography and atmospheric sciences. These salient features could indicate mesoscale convective systems, early stage of cyclonegenesis, or developed cyclonic events.

The main contribution of the paper is a detailed exposition of a generalized vortex detection algorithm motivated by Hough transform and tree constructed based on computed normal vector in 2D velocity vector fields. The algorithm allows one to find the exact vortex center efficiently if it is in the vector field. Here, we demonstrate the vortex detection algorithm using the wind measurements from QuikSCAT [6] satellite observing tropical cyclone events. A simple version of the algorithm has been successfully applied to cyclone eye identification in wind field given a cyclone trajectory [5] and it is also integrated into a cyclone tracking system prototype that uses multiple satellite sources [7].

2. Related Work

Previous research focused on estimating fluid flows such as spiral motion and vortices using image sequences [2, 4]. Vortex detection is often used in cyclone detection. For example, Pao and Yeh [8] proposed a procedure based on image processing and morphology techniques to recognize and locate cyclones from JPEG-format infrared satellite images. Wong et al. [10] implemented a genetic algorithm approach to locate cyclone eye using radar reflectivity and the corresponding Doppler velocity. Wong, Yip, and et al. [9] proposed cyclone eye identification using motion field analysis and template matching using spiral equation $r = ae^{\theta \cot \alpha}$, where a and α are domain specific constants, for flow patterns [12] and region of interest [11].

3. Methodology

The proposed vortex detection algorithm for velocity vector field is motivated by Hough transform [3] and tree constructed using computed normal vectors.

3.1. Intuition

The Hough transform was initially proposed to detect lines and curves in a digital image [3] and later extended to find arbitrary complex shapes [1]. The main idea behind Hough transform is to compute the best parameters that describe a line (or a shape) based on a voting scheme for all the data points in the parameter space. Before performing a voting procedure on the image data, one would need to preprocess the raw image data to obtain image or duality features such as binary edge points [1] or point-to-line transformations [3].

The intuition for our approach is that most of the normal vectors in a velocity field near the vortex center point towards the vortex center (see left diagram in Figure 1). A directed tree is constructed by linking each measurement (or pixel) based on the direction of its normal vector starting from a pixel that is likely to be a vortex center. Assuming that the vector field is uniformly gridded, a normal vector points to one of its eight neighbors (see right diagram in Figure 1). It points to a particular neighbor if it is within the 45^o bounded direction criterion. For example, from the right diagram in Figure 1, a pixel at the center with its normal vector pointing upward within the inverted 45^0 cone will have a linked neighbor at pixel 2. There can only be one directed edge pointing from a pixel to another pixel.

The most likely vortex centers are those pixels that are the root nodes in trees that have large numbers of connected pixels. If there is only one vortex in the vector field, the tree containing the largest number of nodes is most likely to be the vortex. If there are multiple vortices in the vector field, non-overlapping "highly balanced" trees are vortices in the vector field. The first case is possible if one can use domain knowledge to segment regions containing vortices. For example, the wind magnitude can be used to isolate tropical cyclones in a wind field [5, 7].

3.2. Algorithm

Algorithm 1 details the steps for our general vortex detection that supersedes the algorithm in [5, 7] for single vortex detection in the wind vector field for cyclone detection. This algorithm works on any vector field with multiple vortices. Moreover, it also approximates the (pixel-based) radii of the detected vortices. Algorithm 1 consists of two main procedures: (i) normal vector and neighborhood in-

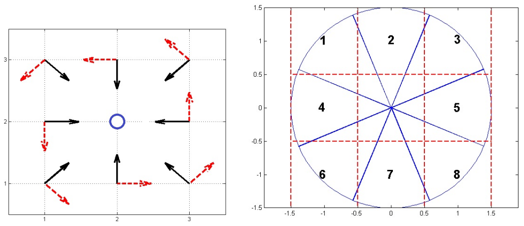


Figure 1. Left: Normal vectors (solid arrows) of velocity vectors (dashed arrows) for a simple anti-clockwise rotating vortex with center at o; Right: Direction criteria (eight 45^o cones) for a normal vector at the center of the grid.

formation computation (Line 1-6) and (ii) vortex center and radius approximation (Line 7-22).

For a vector field to contain vortices, there must be pixels with at least four neighbors pointing towards them (Line 7-10). For a pixel to be a vortex center, it must satisfy two criteria: (i) **Symmetry:** the neighbors pointing towards it must come from all four predefined neighborhood region: $\{1,2\},\{3,5\},\{4,6\},\{7,8\}$ according to the neighbor conventions in the right diagram of Figure 1 (predicate function **SymCond**) and (ii) **Size:** the tree size must be greater than a tree-depth (r) dependent threshold, $\mathbf{Floor}(\pi r^2)$ (Line 17) such that r is the mode of set of all child node depths (Line 16). Algorithm 2 (**TreeSize**) is a recursive function to compute the directed spanning tree size (nc) and the depths of all the child nodes (D) in Line 15 of Algorithm 1.

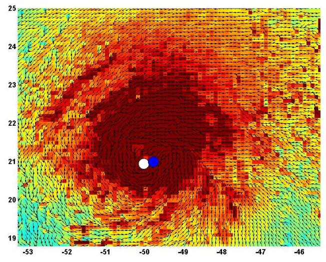


Figure 2. Blue ○: Cyclone eye located using the vortex detection algorithm. White ○: National Hurricane Center best track eye estimate.

Next, we demonstrate some important steps in the vortex detection algorithm using a uniformly gridded QuikSCAT wind vector field (see Figure

```
Input: Uniformly Gridded Vector Field with
       $m$ measurement vector points.
Output: $S$, vortex centers; $R$, vortex radii
1: for Point $i$ in the vector field do
2:    Compute the normal vector $\hat{n}_i$ to the
      direction vector $\hat{d_i}$;
3:    Calculate which neighbor $\hat{n}_i$ is pointing;
4:    Update the neighbor count $N_k$
      for pixel $k$ $\hat{n}_i$ is pointing;
5:    Update $l_k$, list of neighbor pixels,
      pointing at pixel $k$;
6: end for
7: $MaxNeighbor := \max_{1 \leq k \leq m} N_k$;
8: if $MaxNeighbor < 4$ then
9:    Return $0, 0$;
10: end if
11: $VC := \{i \mid N_i \geq MaxNeighbor - 1\}$;
12: m := 0; S = {}; R = {};
13: for $j \in VC$ do
14:    if SymCond($l_j$) == 1 then
15:       $[nc, D, d] :=$ TreeSize($j, l_j$);
16:       $r =$ Mode($D$);
17:       if $nc >$ Floor($\pi r^2$) then
18:          $m = m + 1$;
19:          $S(m) = j, R(m) = r$;
20:       end if
21:    end if
22: end for
23: Return $S, R$;
```

Algorithm 1: Vortex Detection Algorithm.

2). Figure 3 (a) uses color to show the number of neighbor pixels pointing at each pixel after Line 6 in Algorithm 1. Brighter color (e.g., red) implies larger number of neighbor pixels compared to the duller color (e.g., dark blue pixels has no pixel pointing at them). The four pixels that have four pixels pointing towards them are in the two circles in Figure 3 (a). The bigger circle contains three likely vortex centers and the smaller circle has one. These four pixels are stored in VC (Line 11). By iterating Line 14-21 on the four pixels without the **symmetry** condition (Line 14), we have four trees (see Figure 3 (b)). If **symmetry** condition is applied, we have the tree at the bottom left of Figure 3 (b) that also satisfies the **size** condition (Line 17). Hence, its location and radius are stored (Line 19).

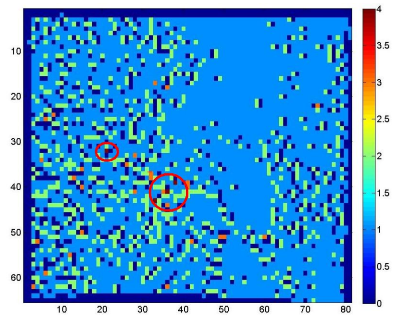
A pixel with a larger number of neighbors pointing at it may generate a smaller tree than one that has less neighbors pointing at it. Hence, one may need to consider a larger VC search space by using a smaller threshold to populate VC. This threshold is $MaxNeighbor - 1$ in our algorithm (Line 11).

The normal vector and neighbor information computation in Algorithm 1 scans through the vec-

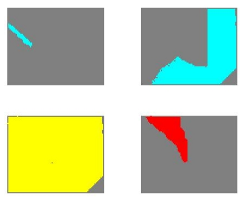
```
Input: $node, l_{node}, D = \{\}, d = 0$
Output: $nc$, $D$, $d$
TreeSize($node, l_{node}, D, d$)

1: if Empty($l_{node}$) then
2:    $D = D \cup \{d\}$;
3:    $d = d - 1$;
4:    Return $1, D, d$;
5: else
6:    for $c \in l_{node}$ do
7:       $d = d + 1$;
8:       $[nc, D, d] =$ TreeSize($c, l_c, D, d$);
9:       $nc = nc + 1$;
10:      $d = d - 1$;
11:      Return $nc, D, d$;
12:   end for
13: end if
```

Algorithm 2: Depth-first search algorithm **TreeSize** for computing the directed spanning tree size and child node depths.



(a)



(b)

Figure 3. (a) Number of neighbors pointing to each pixel for the example in Figure 2. Two red circles contain the four pixels that have four neighbors pointing towards them. (b) The areas covered by the four trees.

tor field to compute the normal vector for each pixel and updates neighbor information. This takes $O(m)$ time. To compute the tree size for the likely vortex centers in VC, we use a depth-first search (Algorithm 2) which has a time complexity of $O(b^d)$ such that b is the branching factor and d is the maximum depth. We note that b is bounded by $MaxNeighbor \leq 8$. The number of depth-first search depends on the number of pixels in VC. Hence, the time complexity of detecting vortices is $O(|VC| \cdot MaxNeighbor^d)$. One notes that in practice (i) the time complexity is much lower as b decreases when the depth increases and (ii) $|VC|$ depends on $MaxNeighbor$.

4. Experimental Results and Discussions

Our vortex detection algorithm is clearly useful in real world application (e.g., cyclone tracking) shown in [7]. The performance of our algorithm on 12.5 km spatial resolution L2B QuikSCAT wind vector field for 4 independent tropical cyclones with two different thresholds to populate VC (Algorithm 1 Line 11) is shown in Table 1. The error (in degree for GB-I and GB-II in Table 1) is computed as the absolute difference between the detected vortex center and the cyclone eye estimate via interpolation between two cyclone eyes along a cyclone track obtained from the National Hurricane Center website[1]. Thresholds for GB-I and GB-II are $MaxNeighbor$ and $MaxNeighbor-1$, respectively. The later threshold considers a larger search space

Hurricane	Number of Images	GB-I	GB-II
Isabel 2003	21	0.9510	0.7852
Maria 2005	14	1.4612	0.8088
Fengshen 2008	10	2.5760	1.4612
Matsa 2005	13	2.0875	1.2917

Table 1. Experimental Results.

than the former and hence its accuracy is better. We note from Figure 4 that the cyclone eye estimate via interpolation may not be a good approach for vortex center estimation. Hence, our performance evaluation may be biased. On the other hand, it shows clearly that our algorithm is much more robust in detecting vortex centers in vector fields than by interpolation.

[1]http://www.nhc.noaa.gov/pastall.shtml

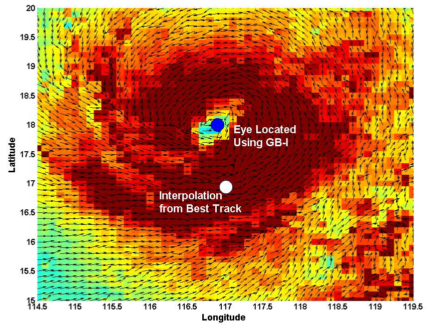


Figure 4. Our approach detects the vortex center accurately (blue dot) while the interpolated eye (white dot) is way out.

References

[1] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] T. Corpetti, É. Mémin, and P. Pérez. Dense estimation of fluid flows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):365–380, 2002.

[3] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[4] I. Herlin, I. Cohen, and S. Bouzidi. Image processing for sequences of oceanographic images. *Journal of Visualization and Computer Animation*, 7(3):169–176, 1996.

[5] S.-S. Ho and A. Talukder. Utilizing spatio-temporal text information for cyclone eye annotation in satellite data. In *IJCAI workshop on Cross-Media Information Access and Mining*, 2009.

[6] T. Lungu and et. al. *QuikSCAT Science Data Product User's Manual.* 2006.

[7] A. Panangadan, S.-S. Ho, and A. Talukder. Cyclone tracking using multiple satellite image sources. In *GIS*, pages 428–431, 2009.

[8] T.-L. Pao and J.-H. Yeh. Typhoon locating and reconstruction from the infrared satellite cloud image. *Journal of Multimedia*, 3(2):45–51, 2008.

[9] K. Y. Wong and C. L. Yip. An intelligent tropical cyclone eye fix system using motion field analysis. In *ICTAI*, pages 652–656, 2005.

[10] K. Y. Wong, C. L. Yip, and P. W. Li. Automatic tropical cyclone eye fix using genetic algorithm. *Expert Syst. Appl.*, 34(1):643–656, 2008.

[11] K. Y. Wong, C. L. Yip, P. W. Li, and W. W. Tsang. Automatic template matching method for tropical cyclone eye fix. In *ICPR (3)*, pages 650–653, 2004.

[12] C. L. Yip and K. Y. Wong. Identifying centers of circulating and spiraling flow patterns. In *ICPR (1)*, pages 769–772. IEEE Computer Society, 2006.