International Institute of Information Technology, Bangalore

Finding Out the Presence of Vortex And It's Rotation

(In the Guidance of Professor Shiva Kumar Malapaka)

Subhodeep Sahoo
(MT2020129)

Pushpendra Kumar
(MT2020135)

Vashu Chaudhary
(MT2020096)

Acknowledgements

We would like to thank Professor Shiva Kumar Malapaka for giving us the opportunity to work on the project and help us whenever we were struck by giving us ideas and resources to learn from. We are profoundly grateful for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

ABSTRACT

Vortex are structures formed due to presence of rotation and helicity in a three dimensional hydrodynamic helically rotating turbulence. The inverse cascade of magnetic helicity in three-dimensional magnetohydrodynamic (3D-MHD) turbulence is believed to be one of the processes responsible for large-scale magnetic structure formation in astrophysical systems. Aim of our project is to find the presence of these structures given a database as an input in hierarchical data format. And after detecting the structure we are finding out the rotation of these vortices using CNN model.

Table of Contents

Introduction

A vortex is a region in which the flow revolves around an axis line, which may be straight or curved. High mass planets leave remarkable features in their parent protoplanetary disks (PPDs), namely a gap, spiral waves, vortices, and eccentricities. In the context of planet formation, vortices are good candidates to trap dust grains allowing them to grow to planetesimal or planets sizes.

Our aim is to figure out these vortices (if present) formed due to presence of rotation and helicity in a three-dimensional hydrodynamic helically rotating turbulence in a database, provided as an input. And then find out the clockwise or anticlockwise rotation of those vortices. Our approach is as follows:

1. We are given a dataset in .h5 format. We are using a software tool named Visit Visualization that detects the vectors in the dataset and generate that vectored image in both 3D and 2D .
2. From an image we are detecting the voritices using Hough Circle Detection Algorithm (as in 2D vortex is displayed as circle) and crop the shape as square to detect the rotation of vector in that shape.
3. Now we are detecting the rotation of vectors using a Convolutional Neural Network Model. The training dataset for this model is generated by taking multiple screenshot of vector rotation from different position of original images. The test dataset is the cropped images of vortex.

Flow Diagram

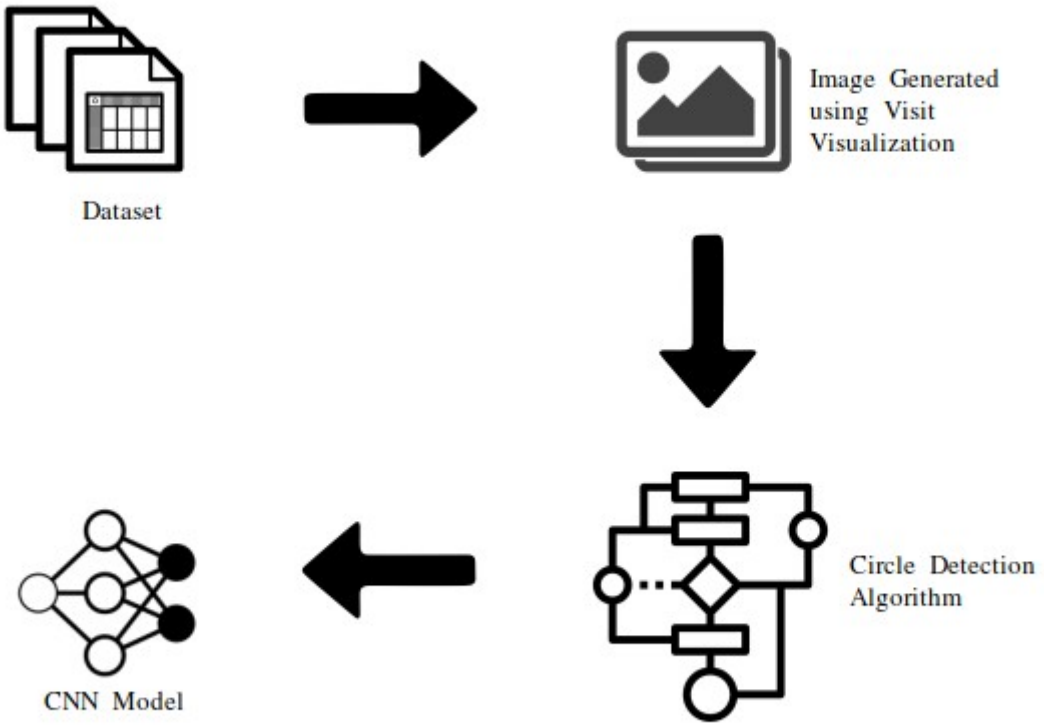


FIG: Flow Diagram of Our Project

I. Image Creation using Visit Visualization :

Dataset for this project was given in .h5 format and was not ready to use so we extracted images from data set using visit visualization tool.

1. Installing visit:

Installing visit on Linux is done using the “visit-install” script. To install an x86_64 version of visit 3.0.0, first download the tar file go to official website (https://visit-dav.github.io/visit-website/) and download the latest version. After downloading go to the directory where tar file is present and run following command

$ “visit-install 3.0.0 linux-x86_64 /usr/local/visit”

This command will install the 3.0.0 version of visit into the **/usr/local/visit** directory

2. Generating images

For generating images we have to define vector expressions. Scientific simulations often keep track of several dozen variables as they run. However, only a small subset of those variables are usually written to a simulation database to save disk space. Sometimes variables can be derived from other variables using a variable expression. VisIt provides variable expressions to allow scientists to create derived variables using variables that are stored in the database. Expressions are extremely powerful because they allow you to analyze new data without necessarily having to rerun a simulation. Variables created using expressions behave just like variables stored in a database, we have used expression for x,y and z coordinates.

Each expression also has a Type that specifies the type of variable the expression produces.Expression type determines the menu in which the variable appears and subsequently the plots that can operate on the variable. We are using vector type of expression. For our case we have 3D data so vector expression will be:

{<PS/vx>,<PS/vy>,<PS/vz>}

The Vector plot displays vector variables using glyphs that indicate the direction and magnitude of vectors in a vector field.

When visualizing a large database, a Vector plot will often have too many vectors. The Vector plot becomes incomprehensible with too many vectors. VisIt provides controls to thin the number of vectors to a number that looks appealing in a visualization. You can accomplish this reduction by setting a fixed number of vectors or by setting a stride. To set a fixed number of vectors, select the Fixed vectors radio button and enter a new number of vectors into the corresponding text field. To reduce the number of vectors by setting the stride, select the Stride radio button and enter a new stride value into the Stride text field.

And, Finally we can save image generated using SaveWindow() command the image will be saved in same directory where visit is installed.

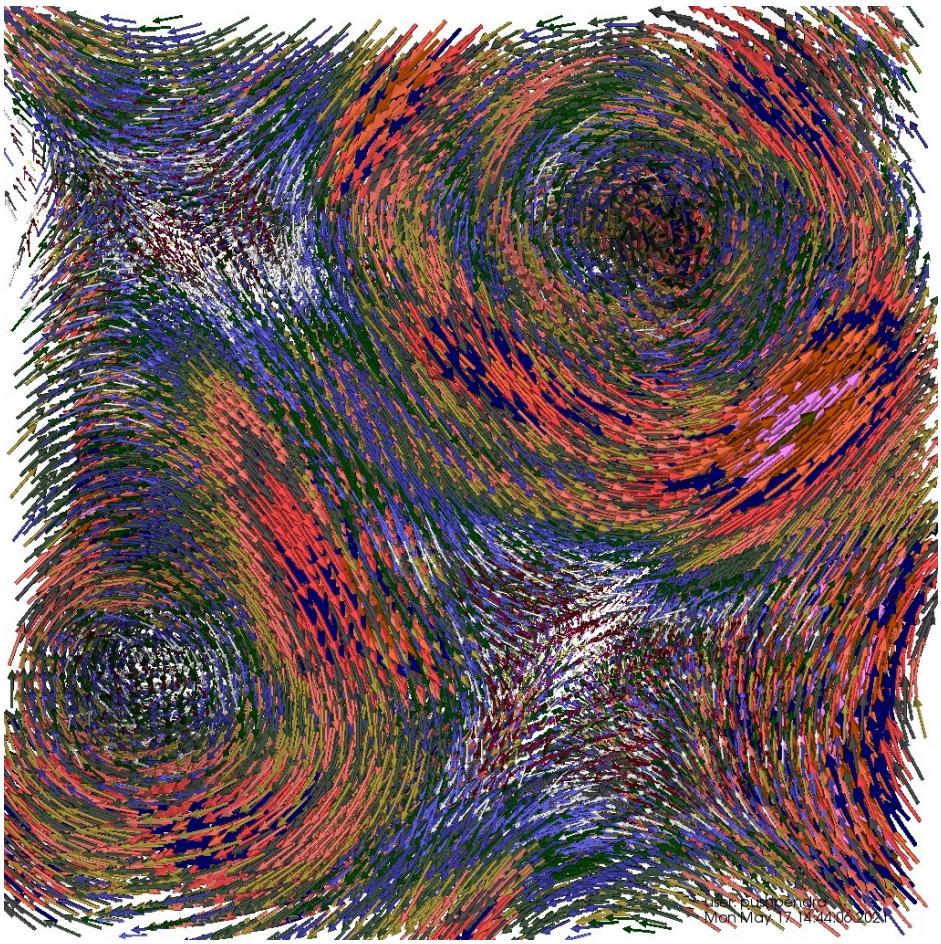


FIG: Image Generated from highrotminhel (velocity.800000.h5) dataset

II. Detecting Vortex from Images

Generally in 2D dimension vortex are circle and we can use Hough Circle Detection algorithm to detect these circles. There are following steps to detect these circles.

1. Preprocess the image

At first, we read the images as array (in RGB format) using opencv imread functions. We are converting the image to gray scale as grayscale simplifies the algorithm and reduces computational requirements. Now we are using Blurring to remove high frequency content (like, noise, edges) from the image. We are using Gaussian Blur and set different kernel size in parameter to check which kernel give the better result. Finally, we are using thresh holding to reduce more computational requirements. In thresh holding if the pixel value is smaller than the threshold, it is set to minimum value, otherwise it is set to a maximum value.

2. Hough Circle Detection algorithm

The parameters used in this algorithm is as follows ,

- thresh : Input image.
- HOUGH_GRADIENT : Define the detection method. Currently this is the only one available in OpenCV.
- dp = 1: The inverse ratio of resolution.
- min_dist : Minimum distance between detected centers.
- Param_1 : Upper threshold for internal Canny edge detector.
- param_2 : Threshold for center detection.
- min_radius: Minimum radius to be detected.
- max_radius : Maximum radius to be detected.

The output circles is vector that stores sets of 3 values: x_c , y_c , r for each detected circle.

3. Save each vortex from a image

After detecting those circles we have have the co-ordinate of center and radius of the circle. Now we are croping the those circular vortex in square shape where width is in range
(x-coordinate of center – radius) to (x-coordinate of center + radius)
and height is in range
(y-coordinate of center – radius) to (y-coordinate of center + radius)

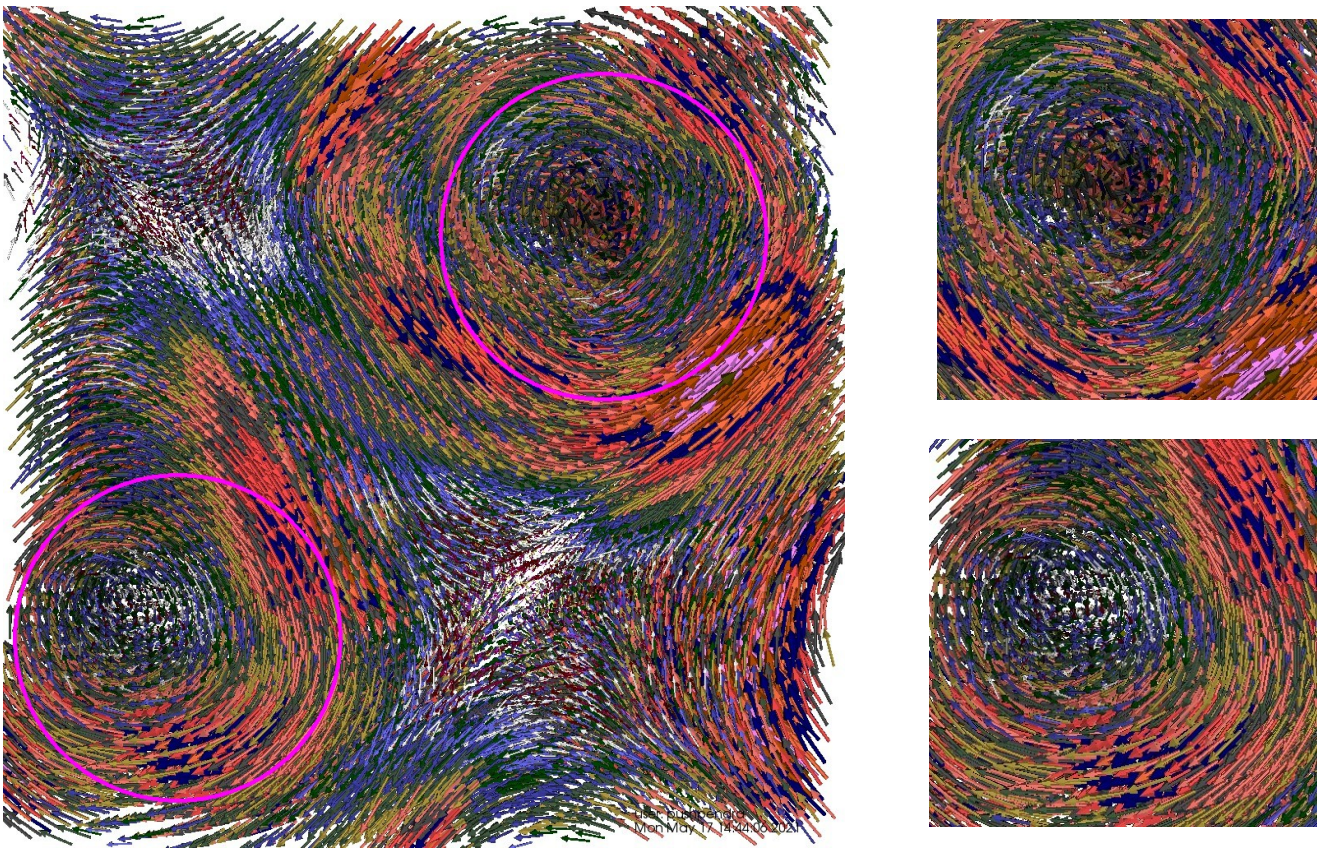


FIG: Detecting Vortex and cropping the Vortex

III. Detecting Rotation of Vortex :

We are using Convolutional Neural Network (CNN) to find out ‘clockwise’ or ‘anticlockwise’ direction of the vectors of our cropped images. The steps are as follows –

1. Generating training dataset

We are generating the training dataset to feed our CNN model by taking screenshot from original images generated from h5 dataset. We have 150 ‘clockwise’ and 150 ‘anticlockwise’ image set to train our model. Our data is classified in a folder meant for that purpose, in which every class has its own subfolder.

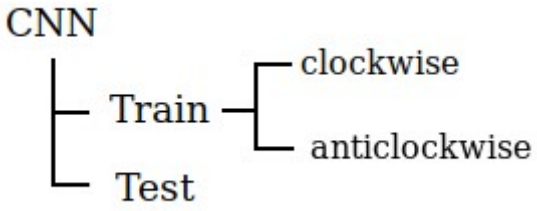


FIG: Folder Structure of Dataset

2. Importing libraries and modules

We are importing

- **numpy** to use numpy array for faster operation
- **os** to get the path of directories
- **opencv** to read, resize images
- **pickle** to convert the array into a character stream
- **matplotlib** for plotting purposes
- **tensorflow** to use prebuilt keras model for training our CNN model.
- **PIL** to display our images in jupyter notebook

3. Preparing our dataset to train

We are defining a list CATEGORIES = ["anticlockwise", "clockwise"] that contains category of training dataset. Now we are reading the images using opencv module in ‘img_array’ and resizing our images so that every image will have same dimension. We also have a ‘class_num’ variable that has the label of our category (0 for ‘anticlockwise’ 1 for ‘clockwise’). We are storing

the images and labels in training_data list as [new_array, class_num]. Now we are creating 2 list ‘X’ and ‘y’ that will have all features (image array) and labels. We are converting those 2 lists as numpy array for faster operations. Now we are using pickling to convert the X and y array into a byte stream. Pickle.dump and pickel.load is used to save the list as pickle file and load the the list back from pickle file.

4. Train our neural network model

In our neural network model we have 3 Convolutional layers and 2 Hidden layer (Dense layer). As input, a CNN takes tensors of shape (image_height, image_width, color_channels), ignoring the batch size.

The Conv2D layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. We are using ‘ReLU’ as activation function. MaxPooling2d downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. The window is shifted by strides along each dimension. The output of every Conv2D and MaxPooling2D layer is a 3D tensor of shape (height, width, channels). The width and height dimensions tend to shrink as you go deeper in the network.

To complete the model, we feed the last output tensor from the convolutional base into one or more Dense layers to perform classification. Dense layers take vectors as input (which are 1D), while the current output is a 3D tensor. First, we will flatten (or unroll) the 3D output to 1D, then add one or more Dense layers on top. Our dataset has 2 output classes, so you use a final Dense layer with 2 outputs.

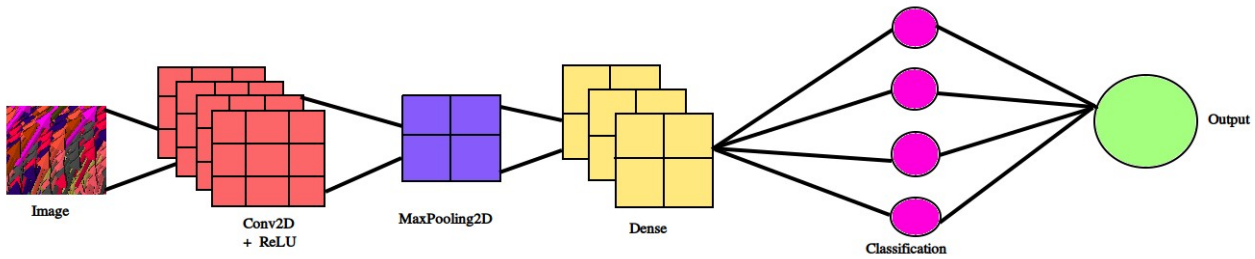


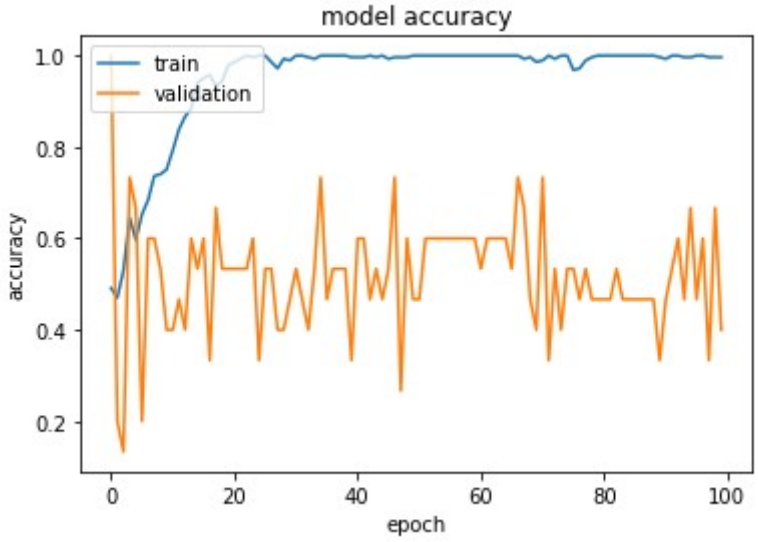
FIG: CNN model

Now we comile the model using following parameter ,

- Loss Function is **SparseCategoricalCrossentropy** that computes the crossentropy loss between the labels and predictions.
- Optimizer that implements the **Adam** algorithm.
- We set **Accuracy** as matrices.

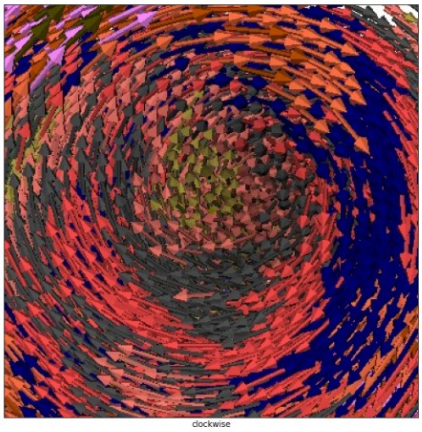
Now we run our model for batch size = 32, number of epochs = 100 and validation data is 5% of training data. Finally we save our model as **CNN.model.**

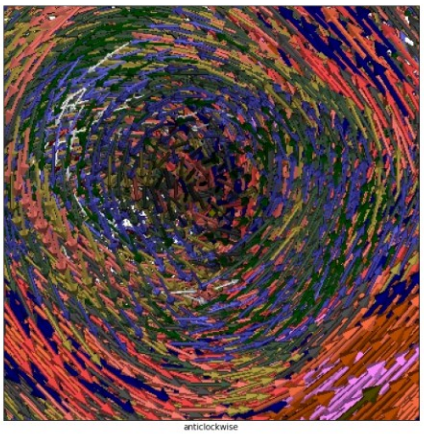
5. Model accuracy

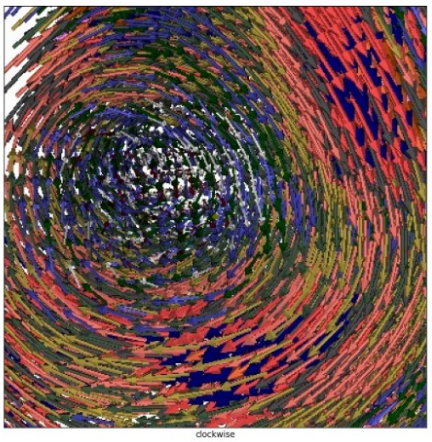


6. Test our CNN model

Now, you can use your neural network to predict our cropped images. The function prepare(file) allows us to use an image of any size, since it automatically resize it to the image size we defined in the first program. Now we are reading all the test images and predict the rotation and plot the image and its category using PIL.







Conclusion

Vortex flows exist across a broad range of spatial and temporal scales in the universal atmosphere. Small-scale vortices are thought to play an important role in energy transport in the universal atmosphere. However, their physical properties remain poorly understood due to the limited spatial resolution of the observations. In our work we are detecting the vortex and its co-ordinate and radius. And finally its rotation by processing the vector diagram.

This work is a very challenging task because of selection of proper parameters for both Circle Detection Algorithm and Convolutional Neural Network Model. Proper tuning of hyperparameters were required in both algorithm order to determine a model which will produce a reliable prediction. We would like to conclude that we were able to come up with an efficient model to detect all vortices and predict the rotation of all vortices . This work obviously enabled us to come up with a neural network model which would reliably predict the clockwise or anticlockwise rotation of a particular vortex.

References

[1] Shiva Kumar Malapaka and Wolf-Christian Müller. “Large-Scale Magnetic Structure Formation In Three-Dimensional Magneto-Hydrodynamic Turbulance”.
[Online]. Available: ”https://iopscience.iop.org/article/10.1088/0004-637X/778/1/21”
[2] Documentation of Visit Visualiztion tool.
[Online]. Available:
“https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/gui_manual/index.html”
[3] OpenCV: Hough Circle Transform - OpenCV documentation.
[Online]. Available:
“https://docs.opencv.org/master/da/d53/tutorial_py_houghcircles.html”
[4] Govinda Dumane. “Introduction to Convolutional Neural Network (CNN) using Tensorflow”. [Online]. Available:
“https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83”
[5] Convolutional Neural Network (CNN) using Tensorflow documentation.
[Online]. Available: “https://www.tensorflow.org/tutorials/images/cnn”