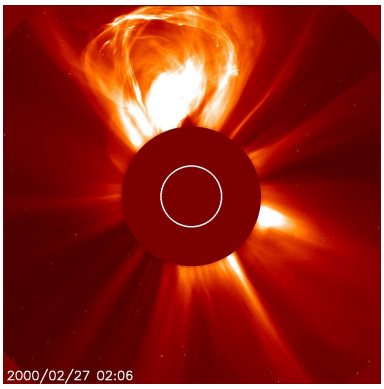
Development of Optical Flow Technique to track coronal transients in inner corona

N.Arutkeerthi-P49, Harini Suresh-P25

August 2022

1 Introduction

The study of space weather fuels both curiosity and applications like predicting geomagnetic storms,protecting satellites,etc
Coronal Mass Ejections (CMEs) are a well known phenomenon of the sun which tend to have an appreciable magnitude of effect on us.[1] They form from an eternal looping and tangling of magnetic fields on the corona layer of the sun.The magnetic fields twist and slowly build up potential beyond a breaking point which it gets ejected as jets, flares and mass ejections.[2][3]
Thus, predicting their velocities and trajectory would benefit in forecasting space weather and taking necessary actions.
In our project we would be applying a computer vision technique called optical flow to track coronal transients as close to the sun as possible. The initial phase of our project involves work on LASCO C2 data, which enables us to track transients from as close as 2 solar radii. With the Aditya L1 mission launching by March 2023, work on CME data close to 1.05 solar radii seems possible.



2 Optical Flow - The algorithm

Optical flow is a computer vision algorithm to find the velocity of pixels, the units that form the objects, moving in subsequent frames of a video.[4][5]
It works on an assumption that the brightness intensity in a small given patch is constant.
Therefore, the intensity between consequent frames is taken to be constant

$$I(x, y, t) = I(x + \partial x, y + \partial y, t + \partial t) \tag{1}$$

Next, we take the Taylor series approximation of RHS and eliminate the common terms.

$$I(x, y, t) = I(x, y, t) + (\partial I/\partial x)\delta x + (\partial I/\partial y)\delta y + (\partial I/\partial t)\delta t \tag{2}$$

$$(\partial I/\partial x)\delta x + (\partial I/\partial y)\delta y + (\partial I/\partial t)\delta t = 0 \tag{3}$$

Dividing by δt on both sides,

$$(\partial I/\partial x)u + (\partial I/\partial y)v + (\partial I/\partial t) = 0 \tag{4}$$

$$I_x.u + I_y.v + I_t = 0 - -- > EQUATION! \tag{5}$$

This requires simple matrix algebra henceforth to get the x direction and y direction velocity(u and v respectively in this case) if we get the matrix of I_x, I_y and I_t.

3 Corner detection algorithm: Shi-Tomasi

Why corners? Because, in the case of an edge or other dimensions' analysis, the velocity we would calculate would not be the actual velocity, rather, would be a component.
If we're scanning the image with a window just as we would with a kernel and we notice that there is an area where there's a major change(gradient change) no matter in what direction we actually scan, then we have a good intuition that there's probably a corner there.[6]

For a window(W) located at (X, Y) with pixel intensity I(X, Y), formula for Shi-Tomasi Corner Detection is –

$$f(X, Y) = \Sigma[I(X_k, Y_k) - I(X_k + \Delta x, Y_k + \Delta y)]^2 \tag{6}$$

Calculation of f(X,Y) would be time consuming. Hence we use Taylor Expansion to simplify the scoring function R

$$R = min(\lambda 1, \lambda 2)$$

where $\lambda 1, \lambda 2$ are eigenvalues of the resultant matrix M, where M is:

$$M = w(x,y) \begin{pmatrix} \Sigma I_x^2 & \Sigma I_x.I_y \\ \Sigma I_x.I_y & \Sigma I_y^2 \end{pmatrix} \tag{7}$$

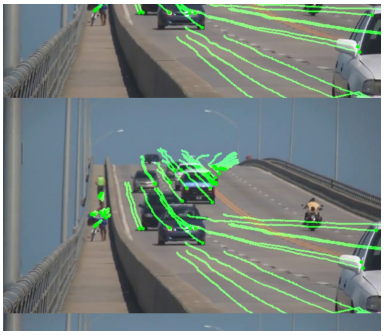
where w(x,y) is arbitrary window equation containing information about the size of the kernel analysed
Now, the value of R can be between 0 and 1. We can define the quality parameter above which we want the algorithm to consider a given moving patch as a corner. Typically since this algorithm is used in simple videos like car traffic, birds movements and analysing water streams, it is suggested to keep the quality parameter between 0.4-0.6.
Since we are dealing with coronagraphs here with less sharper features, we have to consider the parameters a bit lower between 0.1-0.2 to track the CMEs.

4 Algorithm application progress

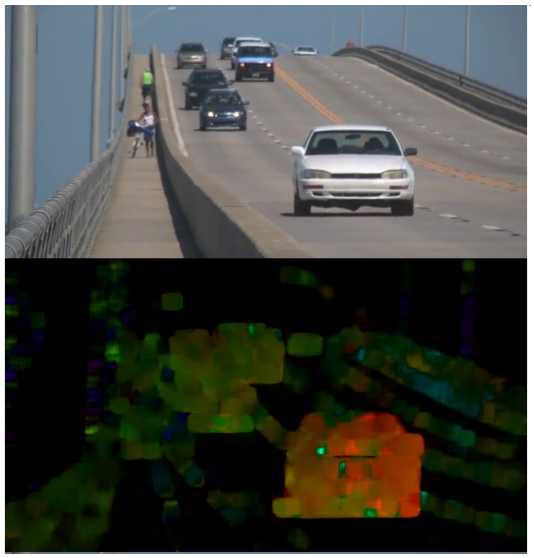
4.1 On simple videos like traffic

First we applied optical flow on simple slow traffic videos. We encountered upon mainly two types of optical flow techniques:SPARSE and DENSE.
The main difference between the two is that sparse gives the flow vectors of "interesting features" also known as corners,while dense gives flow vectors of the entire frame(all pixels-upto one flow vector per pixel).

Though incase of large streams/enhanced videos dense optical flow seems to be a better alternative,we will be working with LASCO C2 coronagraph initially which doesn't have the cme flow on the entire coronagraph making sparse flow the better algorithm to apply in our case
Below are the initial attempts of tracking objects and their flow(not their magnitudes just the flow) using both sparse and dense:



Sparse flow tracking



Denseflow tracking

After this we discovered a wonderful library called opyflow[7] which is a python package applying optical flow algorithm with Shi Tomsai corner detections and was able to give us both the flow vectors along with magnitudes which could be stored in csv files-Thus making Data analysis much more efficient for us.
The velocities of the cars come between 10-20 km/hr which seems fair enough for a slow traffic.

Using opyflow library depicting magnitudes as well

	A	B	C	D	E	F
1	Time_A	Time_B	X	Y	Vx	Vy
2	0.5	1.5	308.1624908	207.8433456	0.3249817	-0.31330872
3	0.5	1.5	177.926918	189.5297241	-0.14616394	-0.94055176
4	0.5	1.5	301.902832	108.7465553	-0.19433594	-0.50688934
5	0.5	1.5	304.8906555	98.74398422	-0.21868896	-0.51203156
6	0.5	1.5	304.8752899	103.7544365	-0.24942017	-0.491127
7	0.5	1.5	176.8641968	176.5275574	-0.27160645	-0.94488525
8	0.5	1.5	241.9809341	155.8336868	-0.038131714	-0.33262634
9	0.5	1.5	232.9363403	154.821785	-0.12731934	-0.35643005
10	0.5	1.5	232.9302063	148.8126907	-0.1395874	-0.37461853
11	0.5	1.5	266.9558105	155.8518143	-0.088378906	-0.29637146
12	0.5	1.5	240.9829407	145.8017807	-0.034118652	-0.3964386
13	0.5	1.5	267.0638275	103.6517639	0.12765503	-0.69647217
14	0.5	1.5	268.9617157	145.7832489	-0.0765686	-0.4335022
15	0.5	1.5	167.8935928	188.6317368	-0.21281433	-0.7365265
16	0.5	1.5	380.355896	206.9067688	0.711792	-0.1864624
17	0.5	1.5	323.5796051	100.659214	-0.8407898	-0.68157196
18	0.5	1.5	313.1057739	199.7590179	0.21154785	-0.4819641
19	0.5	1.5	315.6402283	99.72716904	-0.71954346	-0.5456619
20	0.5	1.5	306.0373077	246.8880615	0.07461548	-0.22387695
21	0.5	1.5	172.8015137	167.5180969	-0.39697266	-0.96380615
22	0.5	1.5	305.0941162	254.8943863	0.18823242	-0.21122742
23	0.5	1.5	258.9196472	118.8031807	-0.16070557	-0.3936386
24	0.5	1.5	306.0527802	239.9067764	0.1055603	-0.18644714
25	0.5	1.5	165.8429871	206.6036453	-0.31402588	-0.79270935
26	0.5	1.5	393.8647614	96.7110939	-0.2704773	-0.5778122
27	0.5	1.5	307.9249725	91.6986351	-0.15005493	-0.6027298
28	0.5	1.5	265.9722137	112.6221619	-0.05557251	-0.75567627
29	0.5	1.5	158.8188095	193.6749573	-0.36238098	-0.65008545
30	0.5	1.5	375.3479156	199.8895035	0.6958313	-0.22099304

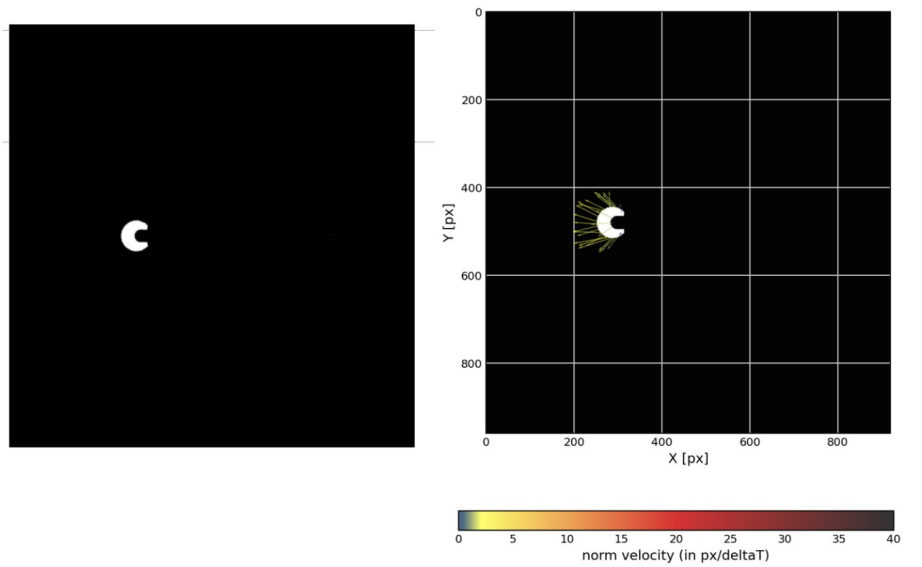
The magnitudes are stored in csv files which is preprocessed before analysis

4.2 On simulated CMEs

Next we applied optical flow on simulated CMEs[8]. We faced issues since it had a lot of noise,but then we played around with the quality level. A parameter of 0.1-0.3 worked well in this case giving us good vectors.

And the simulation was a coronagraph with 1 pixel equalling 2.5 arc seconds(earth reference-1 arc second 700 kms).

Value of one of the trials gave:
Minimum velocity:146.79 km/s
Mean:298.83 km/s
Maximum:439.04 km/s
which is decent enough given the simulated cmes had velocities between 150-400 km/s with the mean being around 250 km/s



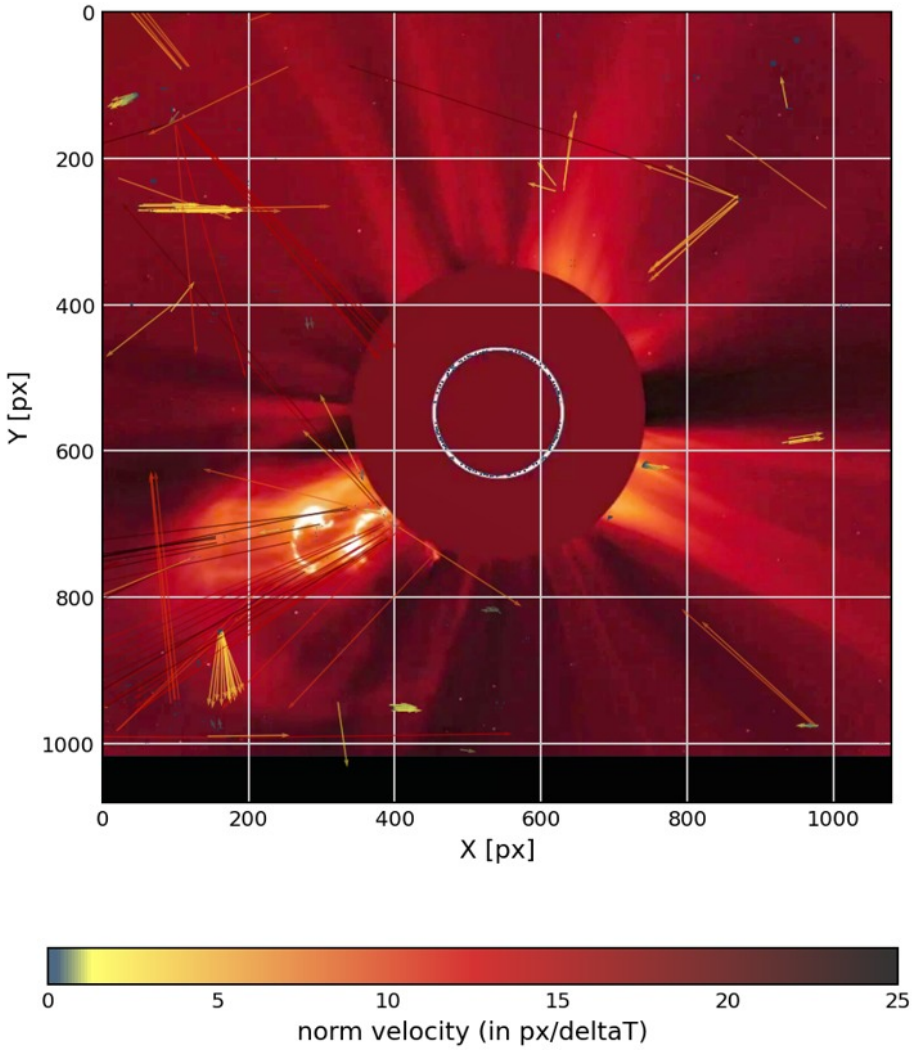
Simulated CME and optical flow on it

4.3 On Original Data

We now just started to apply optical flow on real LASCO C2 coronagraph data.

We chose random days of observation from 2014-15 owing to the solar maximum so that we could track more CMEs

And now the real challenges started,the significant CMEs were not even detected...Reason?Their brightness was changing very rapidly as we could see and the whole algorithm works on the assumption that the brightness remains constant for a local patch in a given time.



Optical flow on original data having issues with brightness fluctuation

We would discuss on how we plan to solve this obstacle in the next section

5 Challenges faces and proposed solutions

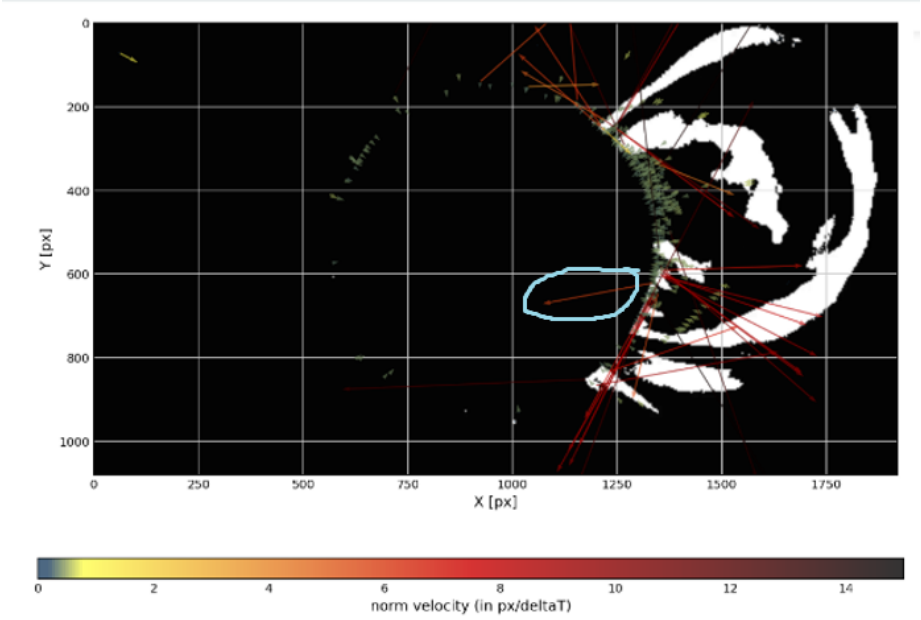
5.1 Back propagating vectors

When we were working with the simulated data,few of the frames gave us some peculiar observations

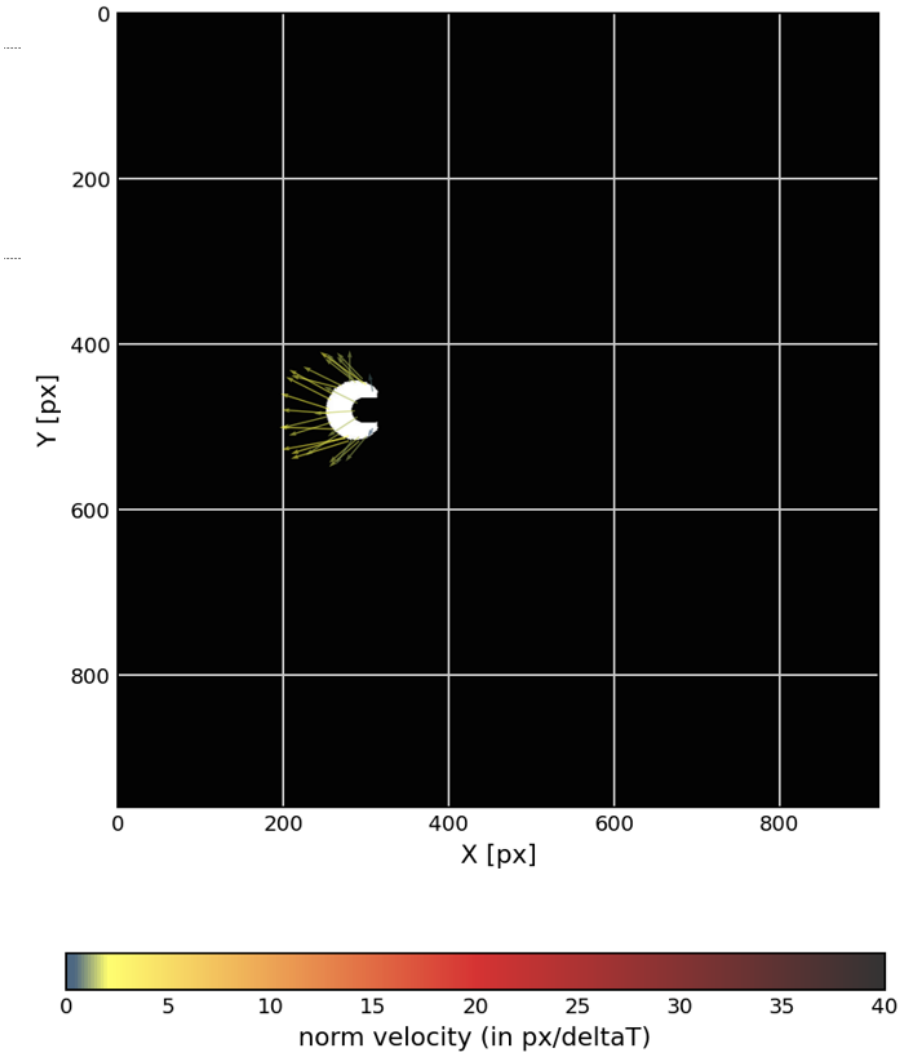
We were able to see back flow vectors.Now in original data its possible,but in the case of simulation thats purely forward thrusted such back pointing vectors

is definitely an error by the algorithm. Probing into the reason-We discovered it was because of a large window kernel.This gives rise to relativistic(not lorentzian but newtonian) velocities where 2 corners get confused as the same and thus even though the corner moves forward,some other corner that enters the window is seen as the same one and relatively it has moved backwards. We solve this problem by a simple parameter change-we reduce the window kernel for analysation.Of course we might still have some rare instances of this error still but it reduces drastically. We reduced the window size from 20x20 to 9x9.

Can we reduce further? We absolutely could but that would mean an upper limit on the velocity vector we would be getting,so we should have better enhanced coronagraphs with real time data with as less cadence as possible or we have to just have the optimal amount of window size-neither too big nor too small



Backward vectors on simulated data(blue marked shows one of them)

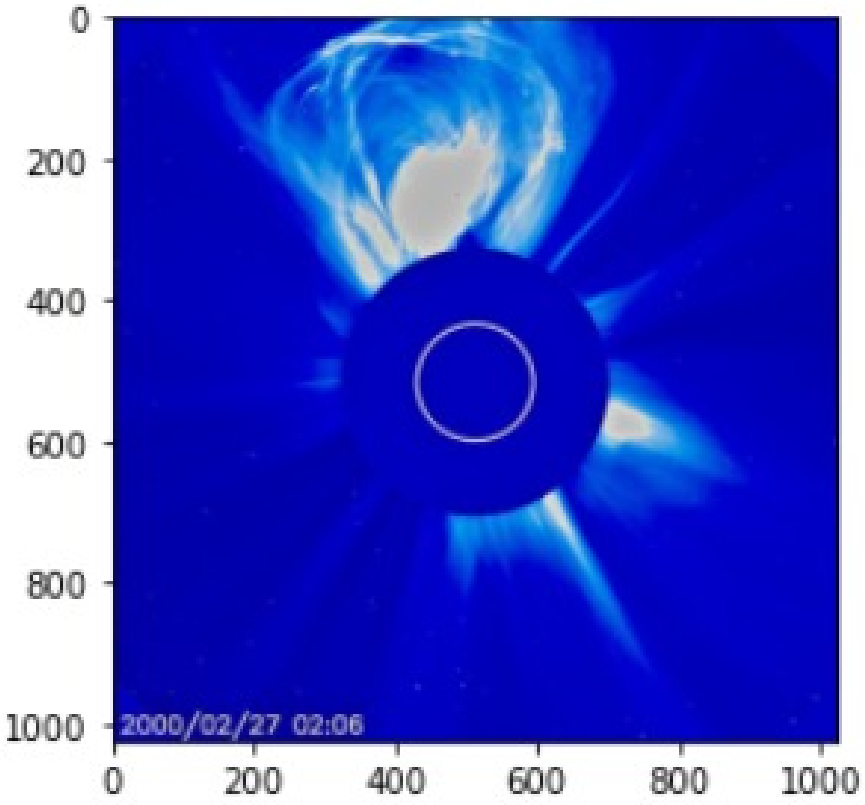


Most backward errors eliminated when window kernel size for analysis made smaller.Outliers could be eliminated during preprocessing

5.2 Highly fluctuating brightness in real data

Now,as we saw the algorithm works on an assumption that the intensity/brightness remains constant over a small patch for a small amount of time. But we know that the cadence of coronagraph is high and the brightness naturally fluctuates very rapidly as it is a CME.Thus the algorithm even failed to recognize few good CMEs that were taken from during the 2014-2015 regime during the solar maximum.
Thus we need to normalize the brightness and then apply the algorithm on it

to see if it works.We currently are applying Multi normalized gaussian(mgn) technique to normalize the brightness over the coronagraph's pixels after which analysis should be fruitful.[9]



A random CME image after multi gaussian normalization

6 Conclusion and Future possibilities

So now the algorithm has been understood the challenges it faces has been identified.
Now we need to apply it on real data of big sample sizes and see how it turns out.We would categorise and analyse on both the solar minimum and maximum and we would also try to keep as least cadence as possible for closer analysis on the CMEs.[10]
Future possibilities include parallel computing where one patch of CME is detected and analysed,processed as separate units in automation. We could also try to inculcate some predicitive algorithms along with this like the Hough Transform so that we could do both predictive analysis and real time analysis.

We thank our mentor Prof Dipankar Banerjee and co mentor Vaibhav Pant who guided us and will still be guiding us along the project.
Special mention to Ritesh Patel and Abhas Pradhan-Our seniors that have guided us along all the way.
All our codes and works are added in the github link given below:
$https://github.com/Arut123/Optical_flow_cme$

References

[1] M. Stix, *The Sun: an introduction*. Springer Science & Business Media, 2004.

[2] T. Howard, *Coronal Mass Ejections-An Introduction*. Springer Science & Business Media, 2011.

[3] D. F. Webb and T. A. Howard, "coronal mass ejections: Observations," *Living Reviews in Solar Physics*, vol. 9, no. 1, pp. 1–83, 2012.

[4] D. Sun, S. Roth, J. P. Lewis, and M. J. Black, "learning optical flow," in *European Conference on Computer Vision*, pp. 83–97, Springer, 2008.

[5] R. C. Colaninno and A. Vourlidas, "analysis of the velocity field of cmes using optical flow methods," *The Astrophysical Journal*, vol. 652, no. 2, p. 1747, 2006.

[6] M. Bansal, M. Kumar, M. Kumar, and K. Kumar, "an efficient technique for object recognition using shi-tomasi corner detection algorithm," *Soft Computing*, vol. 25, no. 6, pp. 4423–4432, 2021.

[7] groussea, "opyflow:python package for optical flow measurements,"

[8] R. Patel, V. Pant, D. Banerjee, A. Kumar, *et al.*, "Onboard automated CME detection algorithm for the visible emission line coronagraph on ADITYA-L1," *Solar Physics*, vol. 293, no. 7, pp. 1–25, 2018.

[9] H. Morgan and M. Druckmüller, "multi-scale gaussian normalization for solar image processing," *Solar physics*, vol. 289, no. 8, pp. 2945–2955, 2014.

[10] H. Morgan and M. B. Korsós, "tracing the magnetic field topology of the quiet corona using propagating disturbances," *The Astrophysical Journal Letters*, vol. 933, no. 2, p. L27, 2022.