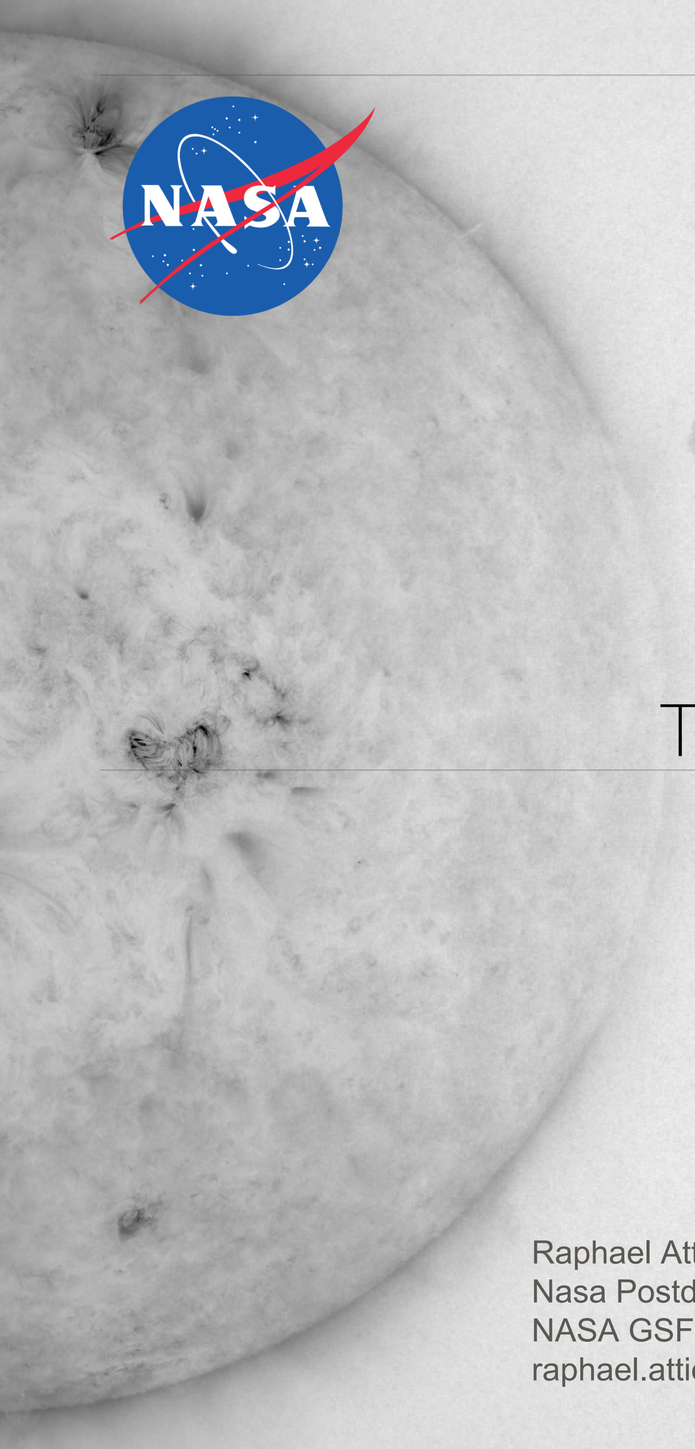


Tracking optical flows for better data mining on solar images

Raphael Attie, Barbara Thompson, Michael Kirk, Karin Muglach

Raphael Attie
Nasa Postdoctoral fellow (NPP)
NASA GSFC
raphael.attie@nasa.gov

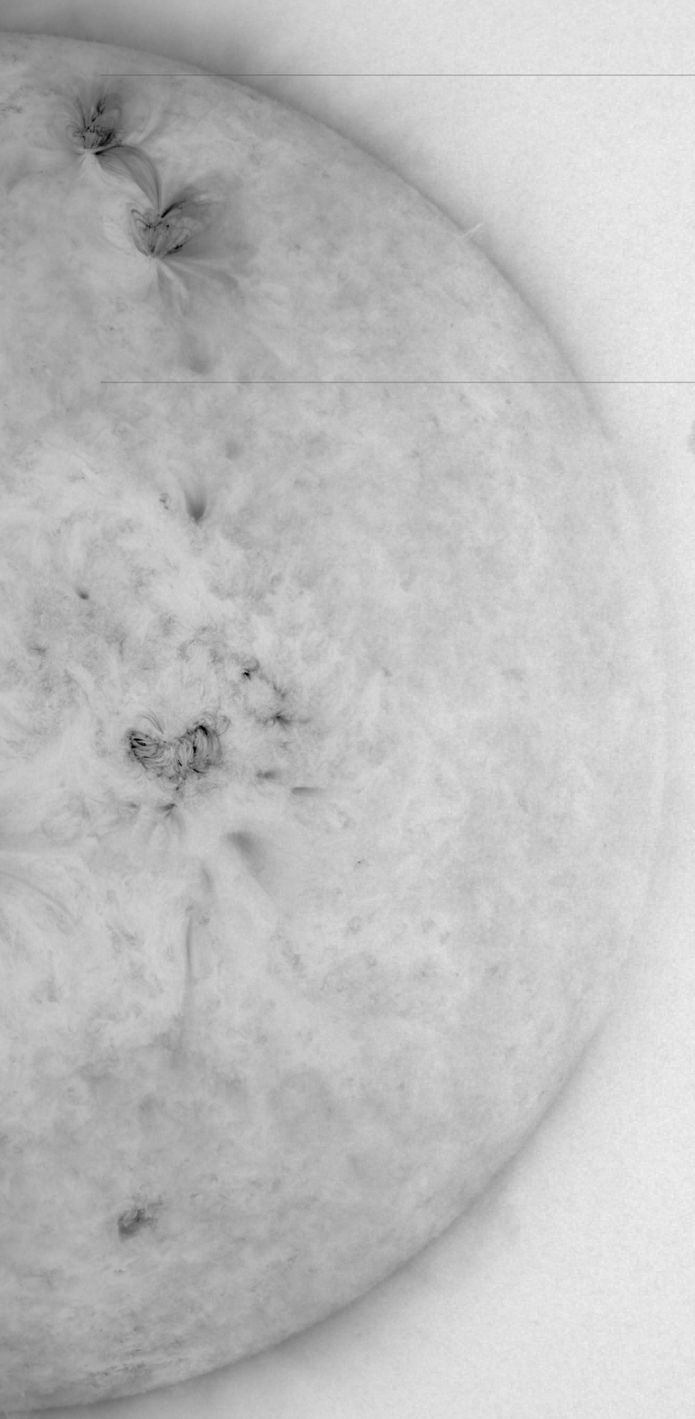
Goddard AI Workshop 2018
NASA/GSFC, Greenbelt, MD

Scientific objectives

Understand the physics of our nearest star:

- How does the plasma circulate on its surface?
- How does it interact with the solar magnetic field?
- How are solar energetic events (flares, CMEs, SEPs, …) triggered?
- To what extent can we help predict these events?

Outline



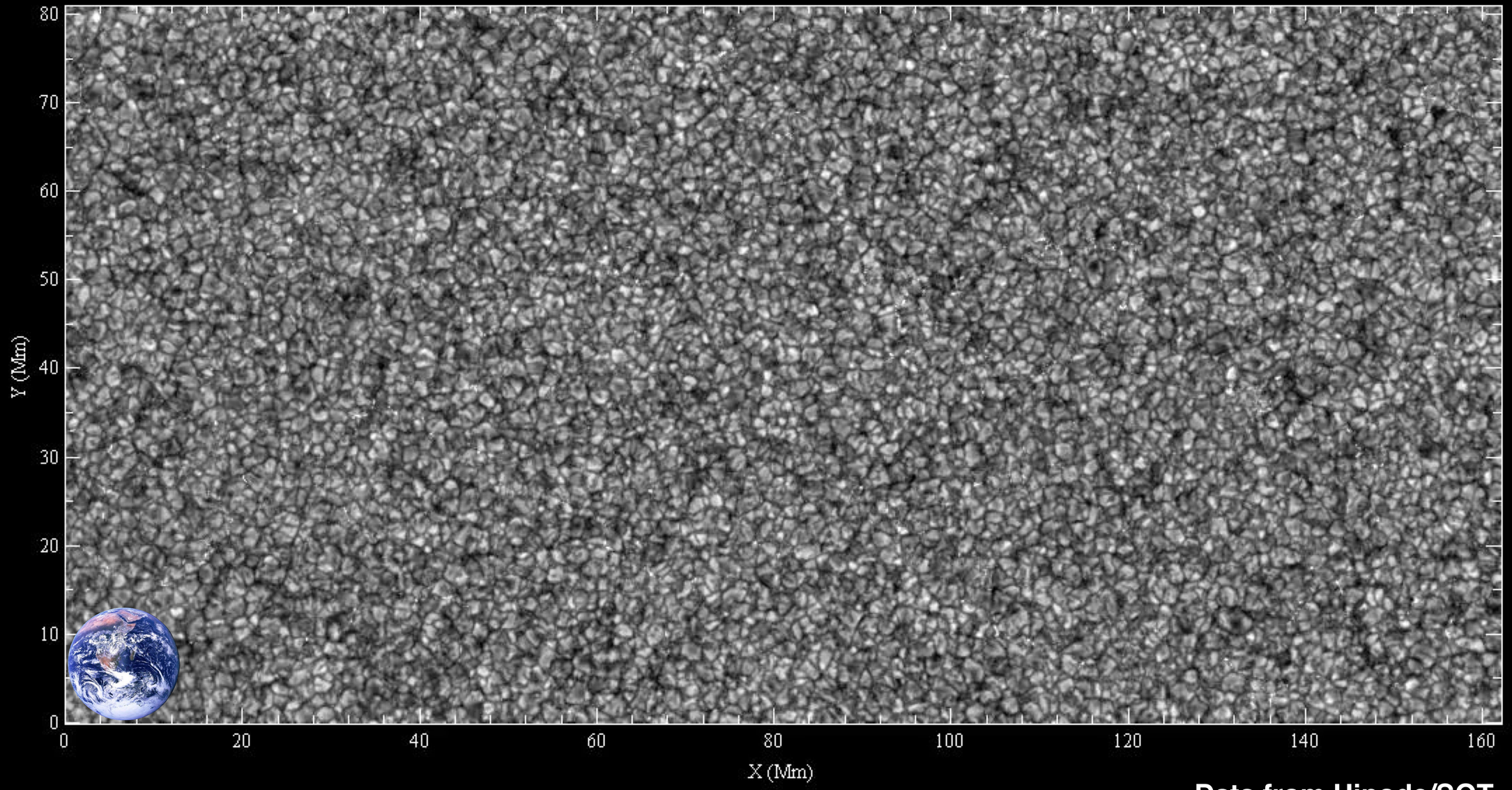
- Tracking optical flows at the solar surface
- Optical flows within the A.I industry
- Bridging both worlds…

Tracking optical flows at the solar surface

Photosphere: close-up showing the “granulation”

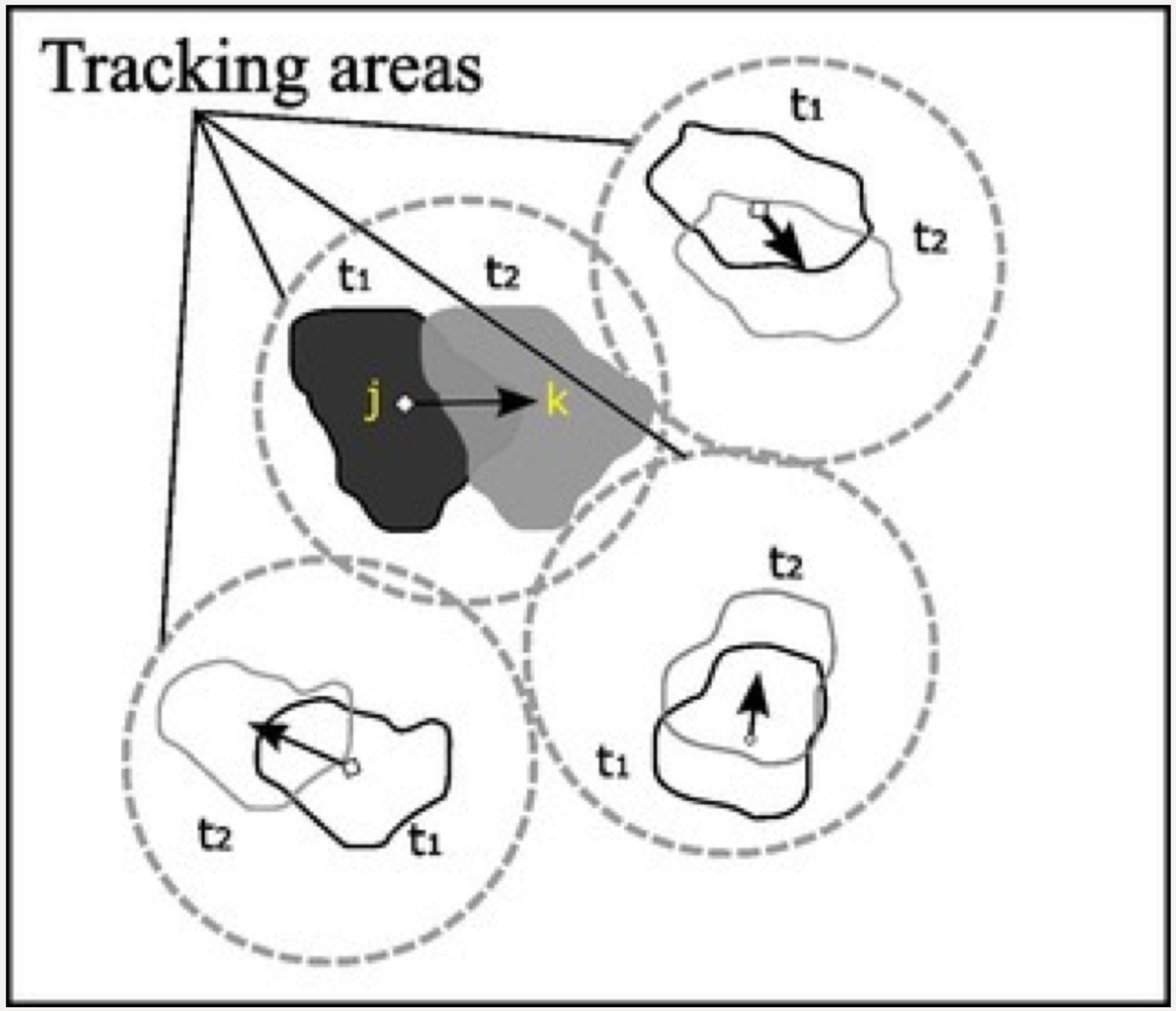
Data from SDO/HMI
HMI: Helioseismic and Magnetic imager
4k x 4k images
45s cadence
24/7

Granules:
size ~1000 km , lifetime: ~5-10 min



Data from Hinode/SOT

Principle of “Local Correlation Tracking” (LCT)



Metric of similarity: correlation

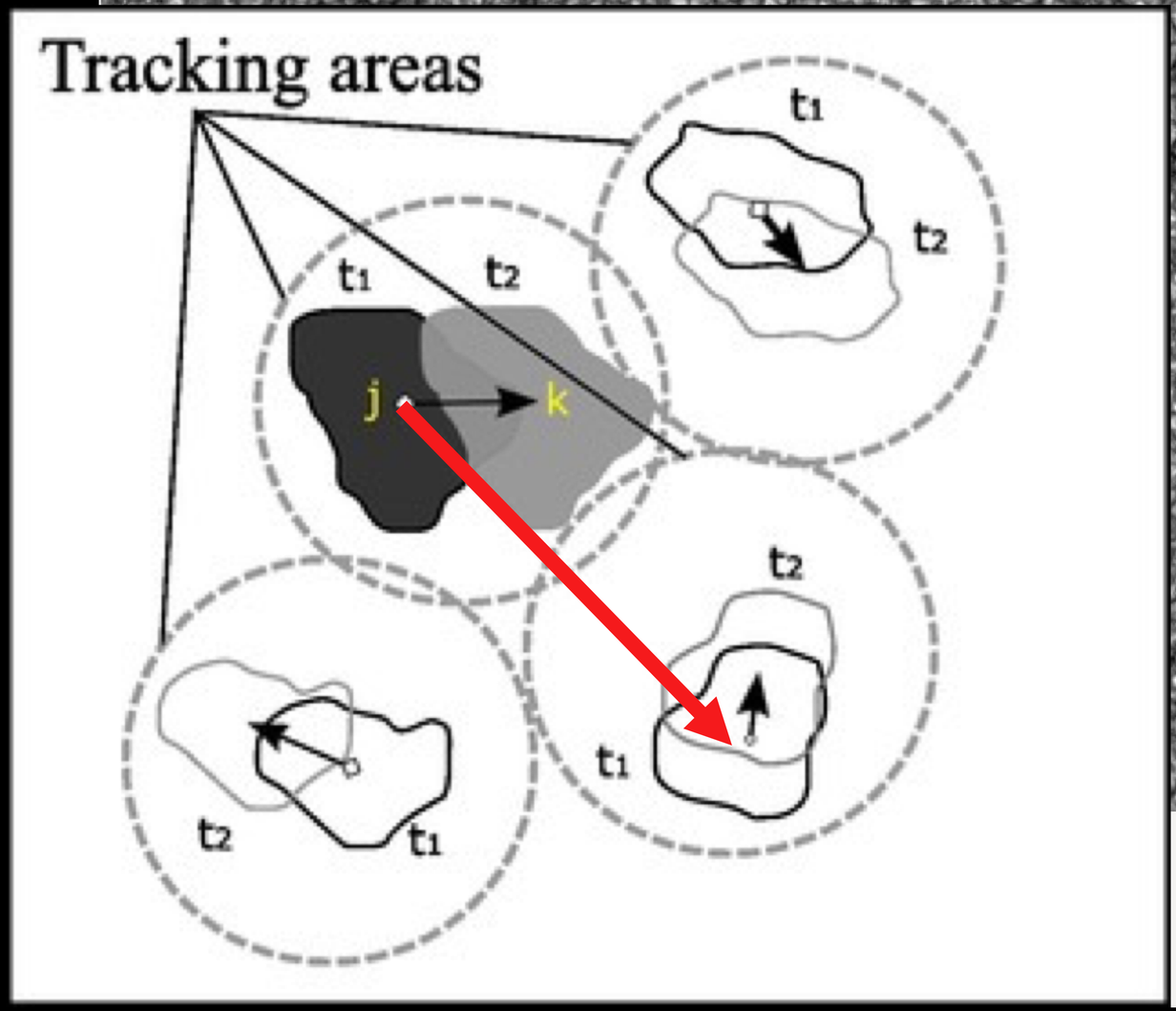
$$\sigma_{X^f} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i^f - \overline{X^f})^2}$$

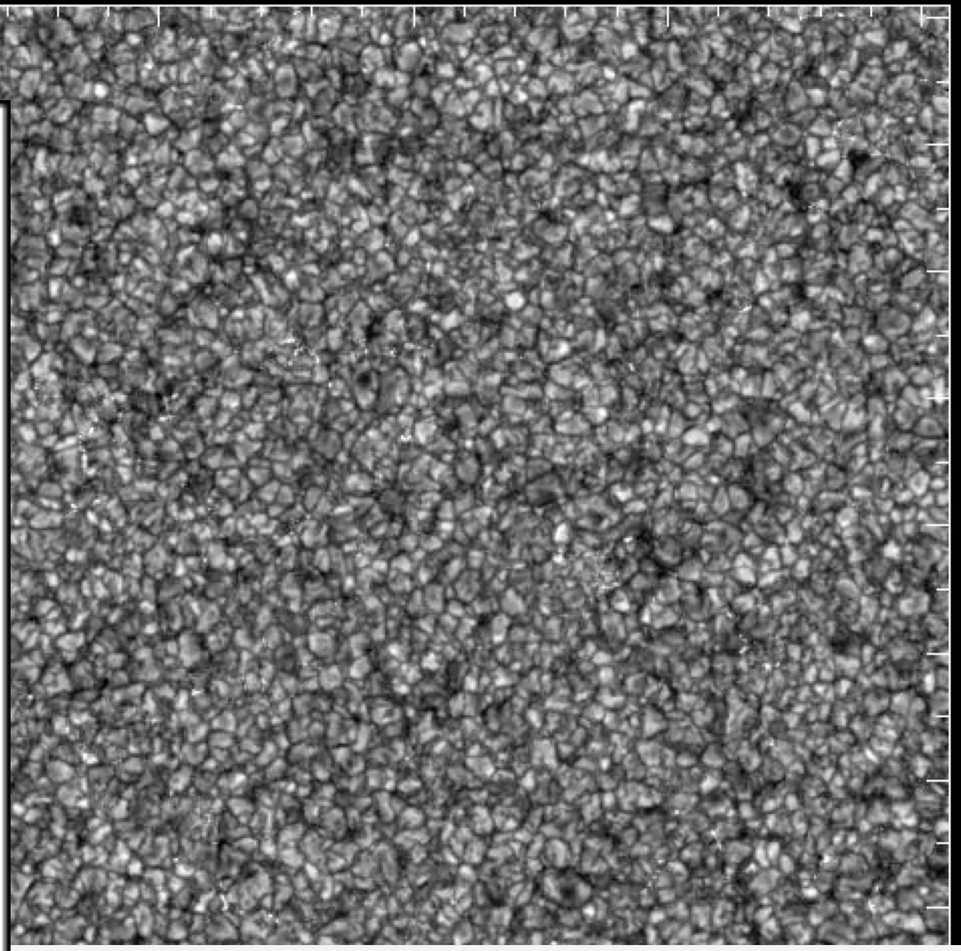
$$R(k) = \frac{\sigma_{X^1X^2}}{\sigma_{X^1}\sigma_{X^2}}$$

$$R(k) = \frac{\sum_{i=1}^{N}\left[(X_i^1 - \overline{X^1})(X_{i+k}^2 - \overline{X^2})\right]}{\sqrt{\sum_{i=1}^{N}(X_i^1 - \overline{X^1})^2}\sqrt{\sum_{i=1}^{N}(X_i^2 - \overline{X^2})^2}}$$

LCT relies on “correlation”, a number that tells by how much two areas in consecutive images look alike. The local displacement of the feature (here, j -> k) is the one for which that number is maximum.

ciple of "Local Correlation Tracking" (LCT)

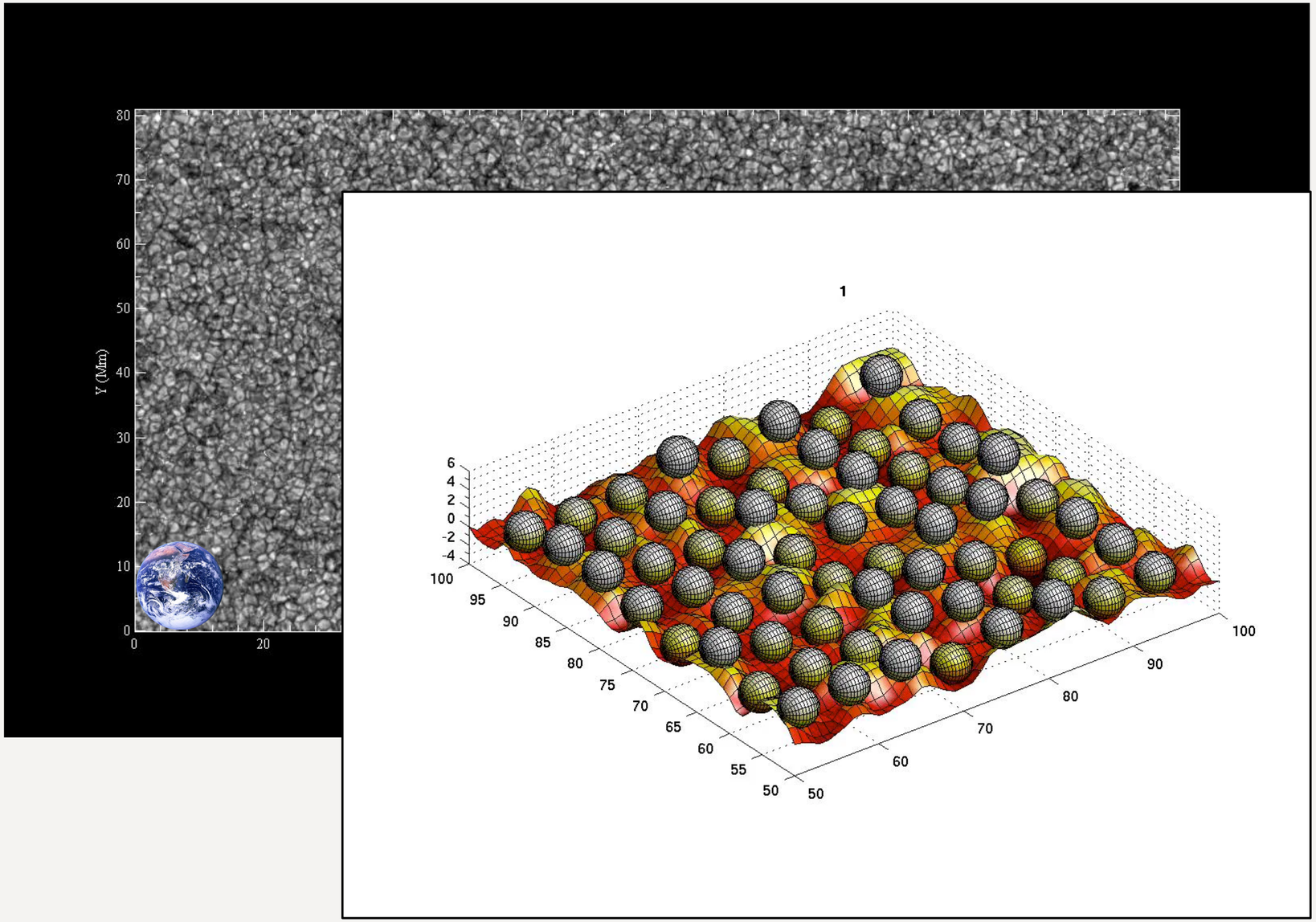




(…) number that tells by how much two areas in consecutive images look alike…
yet… granules look self-similar enough to "fool" this algorithm -> outliers!

Balltracking:

- Put balls between the granules
- Settle into local minima
- Granules push the balls around
- Differentiate final positions
- Average (time & space)



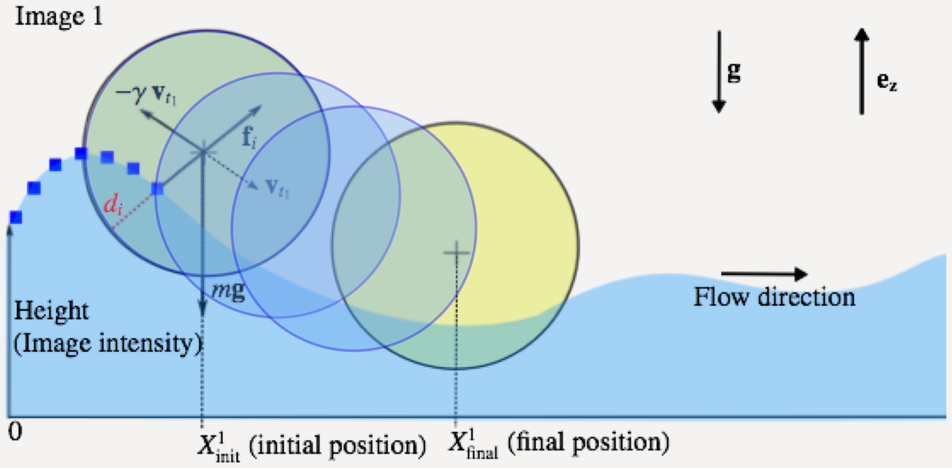


Image 1
$-\gamma\,\mathbf{v}_{t_1}$
$\mathbf{f}_i$
$\mathbf{v}_{t_1}$
d_i
$m\mathbf{g}$
g
$\mathbf{e}_z$
Height
(Image intensity)
Flow direction
0
X^1_{init} (initial position)
X^1_{final} (final position)

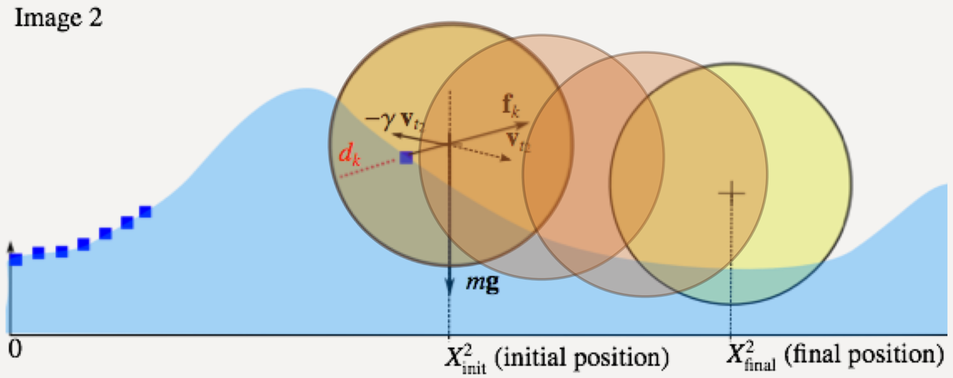


Image 2
$-\gamma\,\mathbf{v}_{t_2}$
$\mathbf{f}_k$
$\mathbf{v}_{t_2}$
d_k
$m\mathbf{g}$
0
X^2_{init} (initial position)
X^2_{final} (final position)

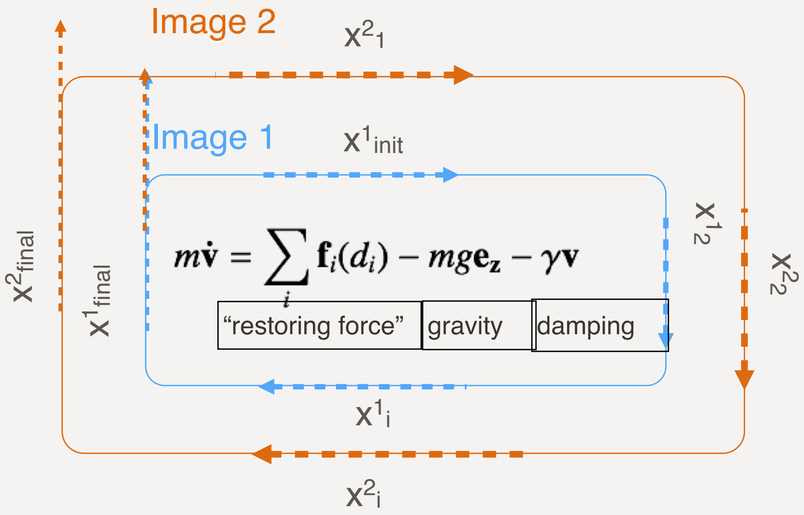
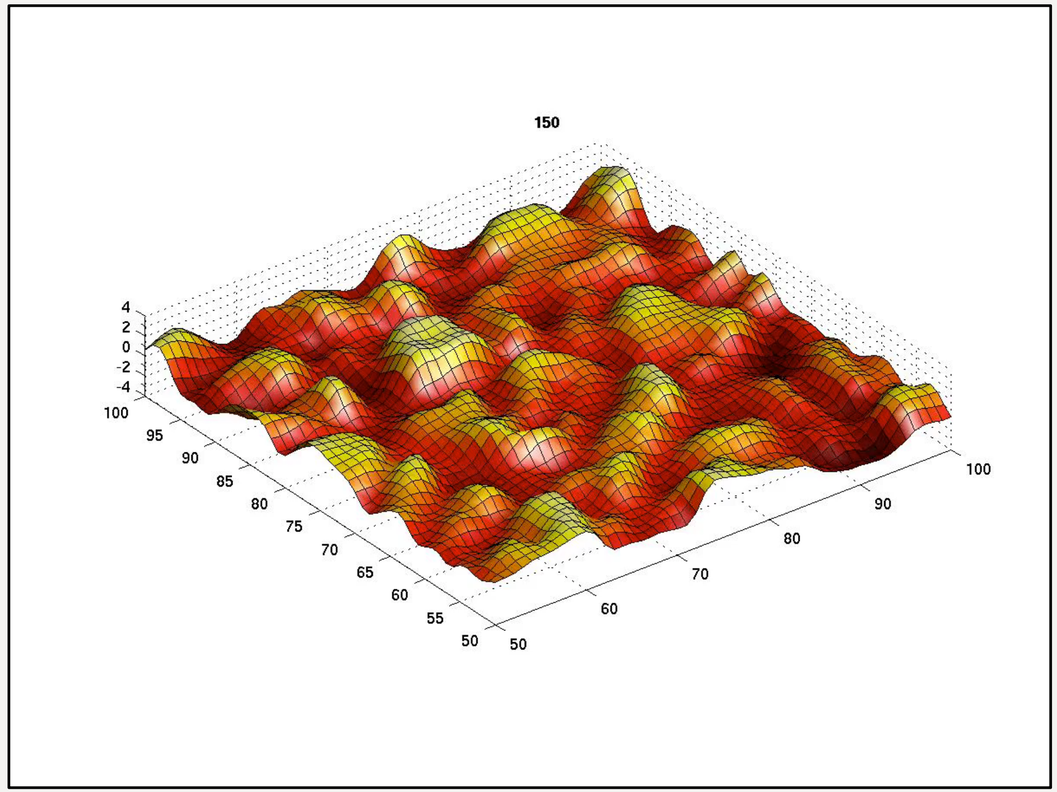
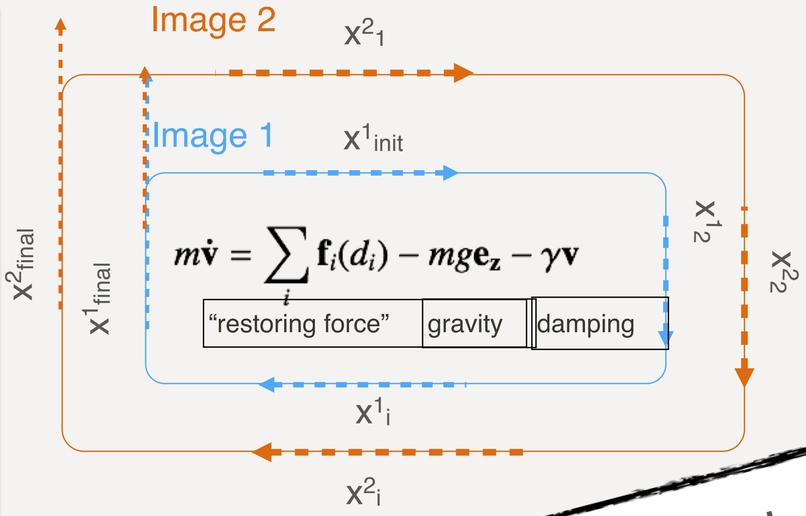


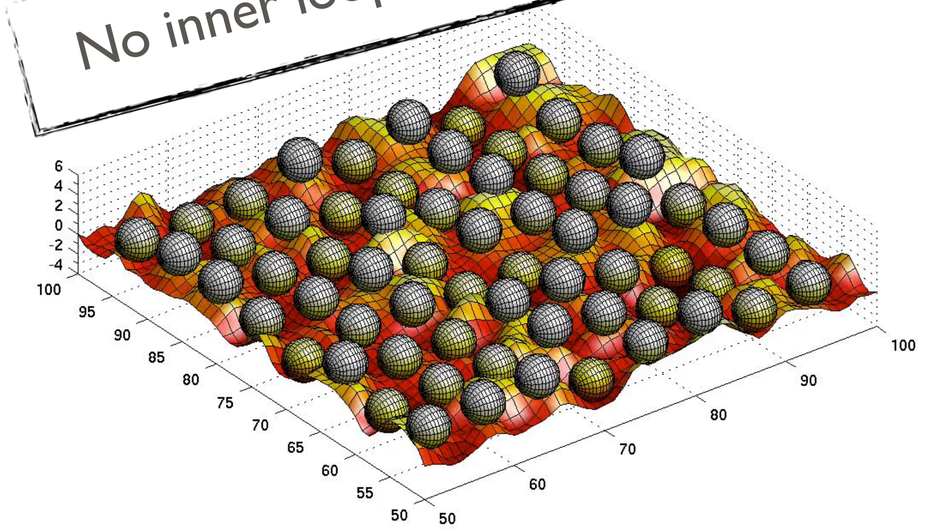
Image 2
$x^2{}_1$
Image 1
$x^1{}_{\text{init}}$
$x^1{}_2$
$x^2{}_2$
$x^2{}_{\text{final}}$
$x^1{}_{\text{final}}$
$m\dot{\mathbf{v}} = \sum_i \mathbf{f}_i(d_i) - m\mathbf{g}\mathbf{e}_z - \gamma\mathbf{v}$
"restoring force"
gravity
damping
$x^1{}_i$
$x^2{}_i$


150
4
2
0
-2
-4
100
95
90
85
80
75
70
65
60
55
50
50
60
70
80
90
100

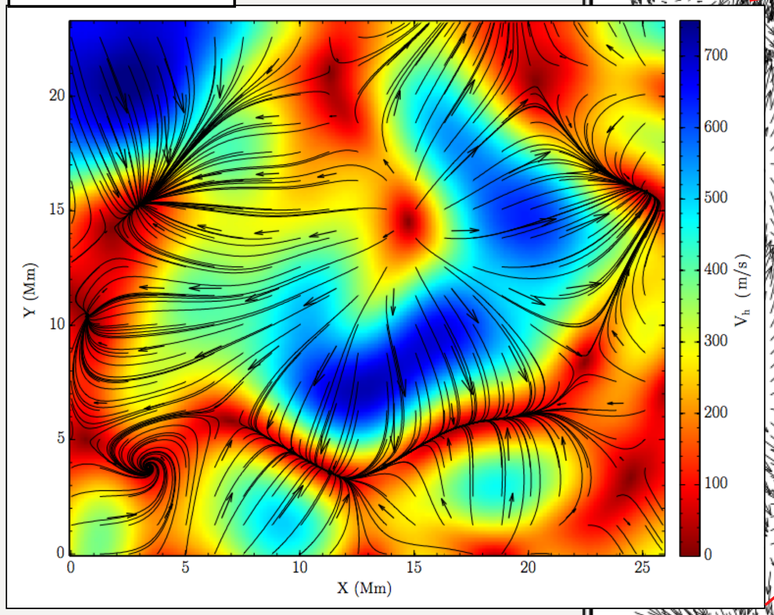
```python
def integrate_motion(bt, surface, return_copies=False):

    # Get the valid balls & unpack vector components for better readability
    #print("Get the valid balls")
    xt, yt, zt = bt.pos[:, bt.new_valid_balls_mask]
    vxt, vyt, vzt = bt.vel[:, bt.new_valid_balls_mask]
    # Update the balls grids with current positions
    # bcols and brows have dimensions = [prod(ballgrid.shape), nballs]
    # Clipping balls rows and cols. Stay away from borders by 1 px
    # because of the interpolation scheme
    #print("Clipping balls rows and cols.")
    bcols = np.clip(bt.bcols + xt, 1, bt.nx - 2)
    brows = np.clip(bt.brows + yt, 1, bt.ny - 2)
    # "ds" stands for "data surface"
    # These interpolations gives identical results in Matlab
    ds = cinterp.bilin_interp2f(surface, bcols, brows)
    fxt, fyt, fzt = compute_force(bt, brows, bcols, xt, yt, zt, ds)

    # Integrate velocity
    vxt += fxt
    vyt += fyt
    vzt += fzt
    # Integrate position including effect of a damped velocity
    # Damping is added arbitrarily for the stability of the code.
    xt += vxt * bt.td * (1 - bt.e_td)
    yt += vyt * bt.td * (1 - bt.e_td)
    zt += vzt * bt.zdamping * (1 - bt.e_tdz)

    bt.pos[0, bt.new_valid_balls_mask] = xt
    bt.pos[1, bt.new_valid_balls_mask] = yt
    bt.pos[2, bt.new_valid_balls_mask] = zt
    # Update the velocity with the damping used above
    bt.vel[0, bt.new_valid_balls_mask] = vxt * bt.e_td
    bt.vel[1, bt.new_valid_balls_mask] = vyt * bt.e_td
    bt.vel[2, bt.new_valid_balls_mask] = vzt * bt.e_tdz

    if return_copies:
        force = np.array([fxt, fyt, fzt])
        return bt.pos.copy(), bt.vel.copy(), force
```



No inner loop, fully vectorized over all balls

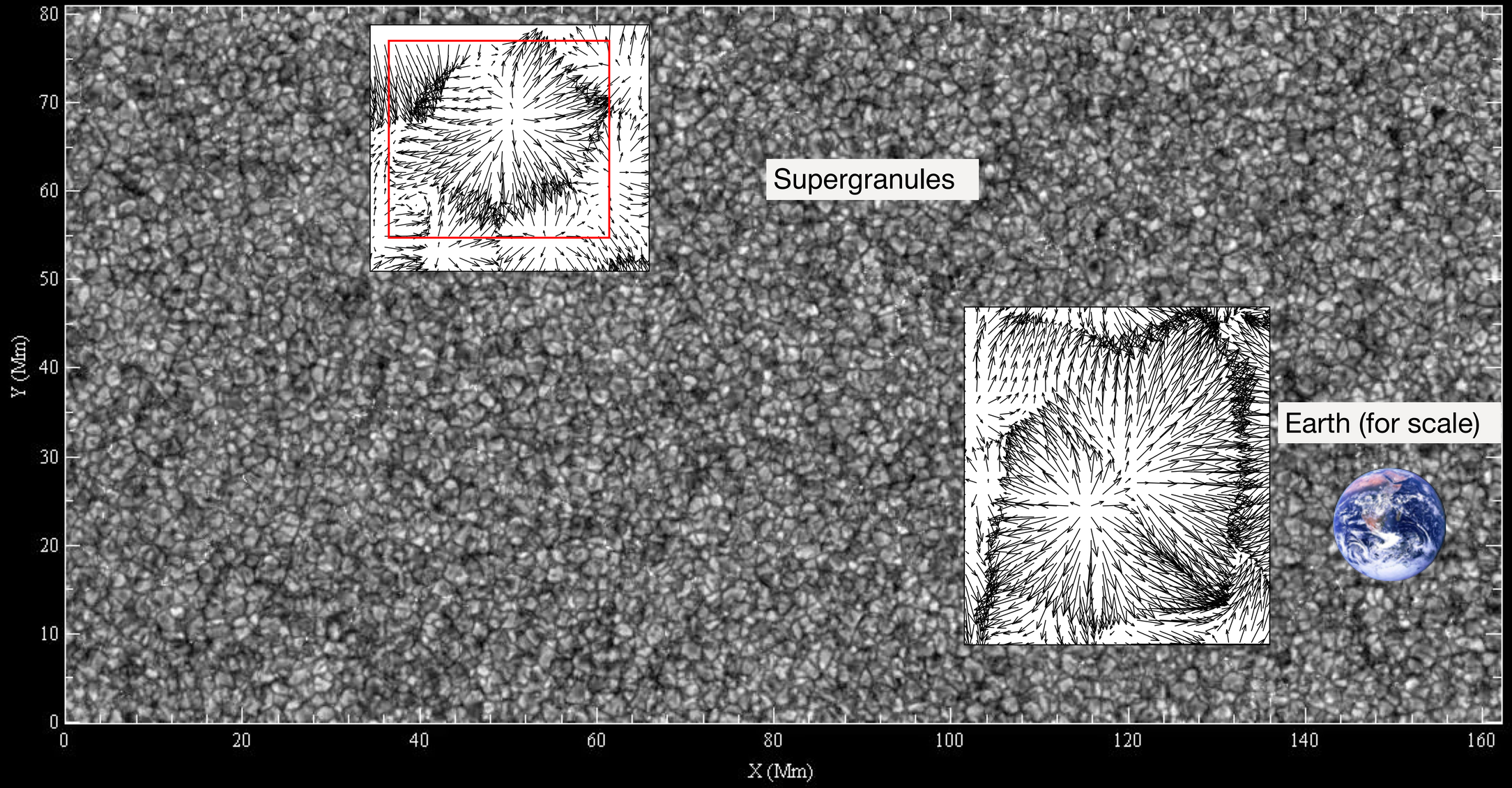


Streamlines



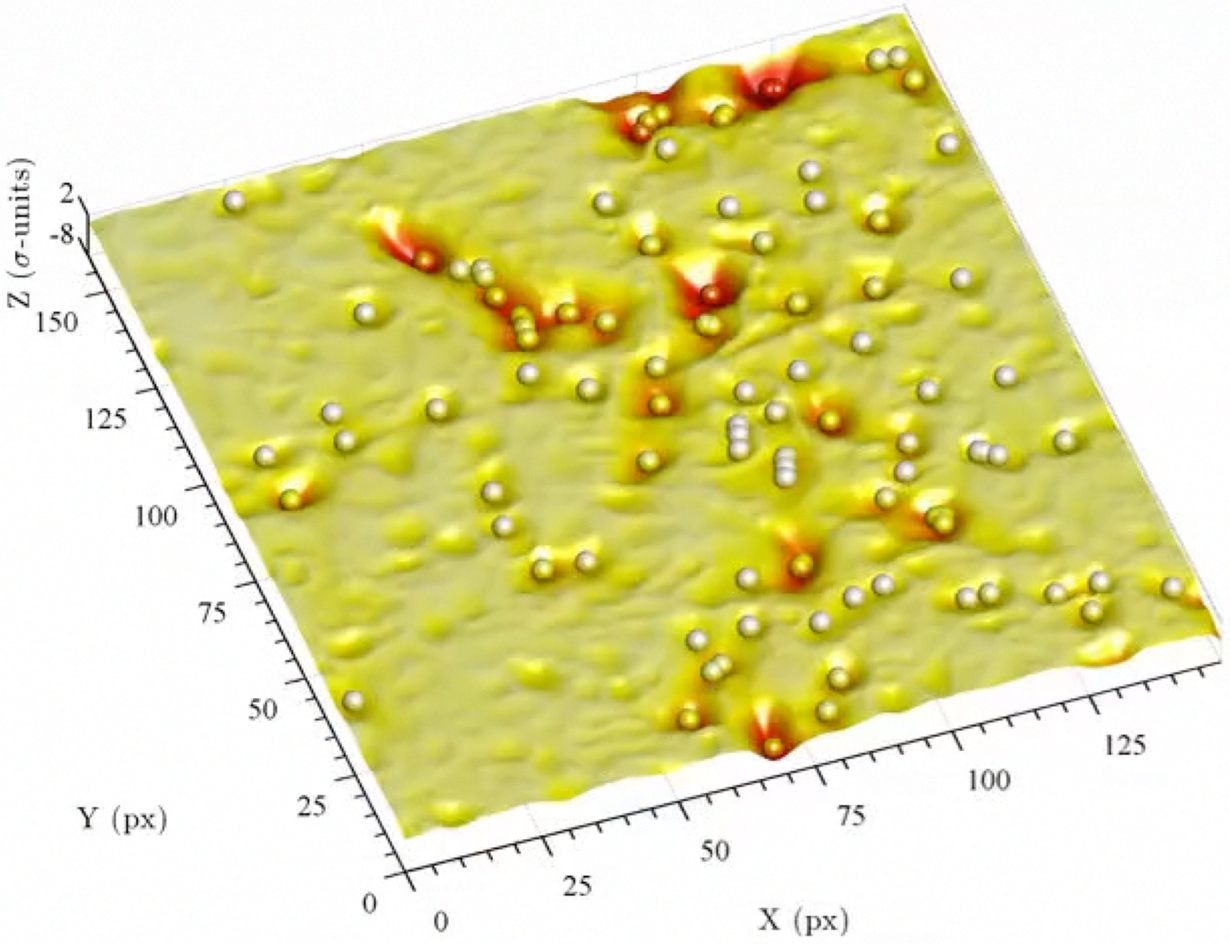
Flow map (2D vector field)

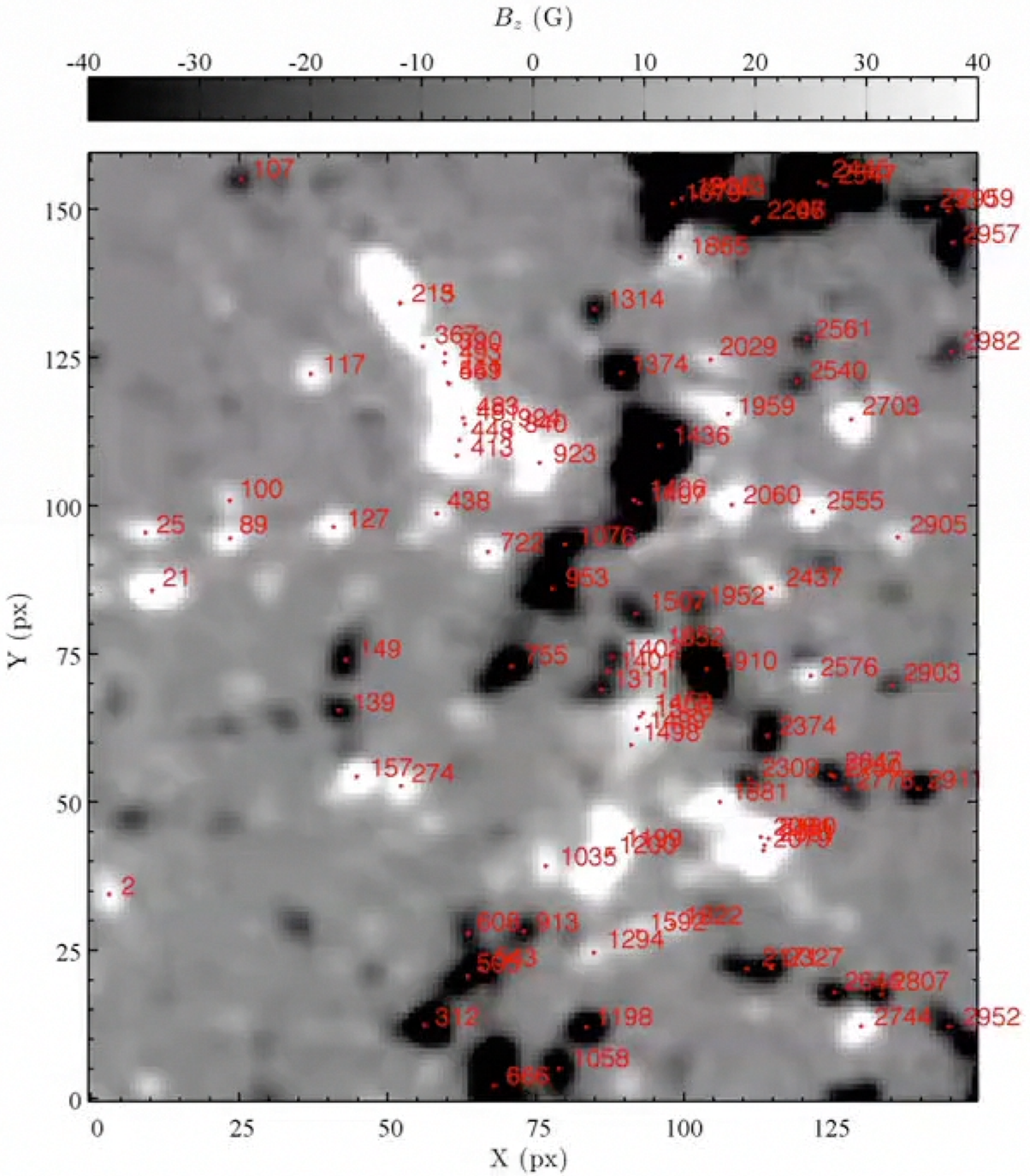
$< \vec{V} > = 350$ m/s

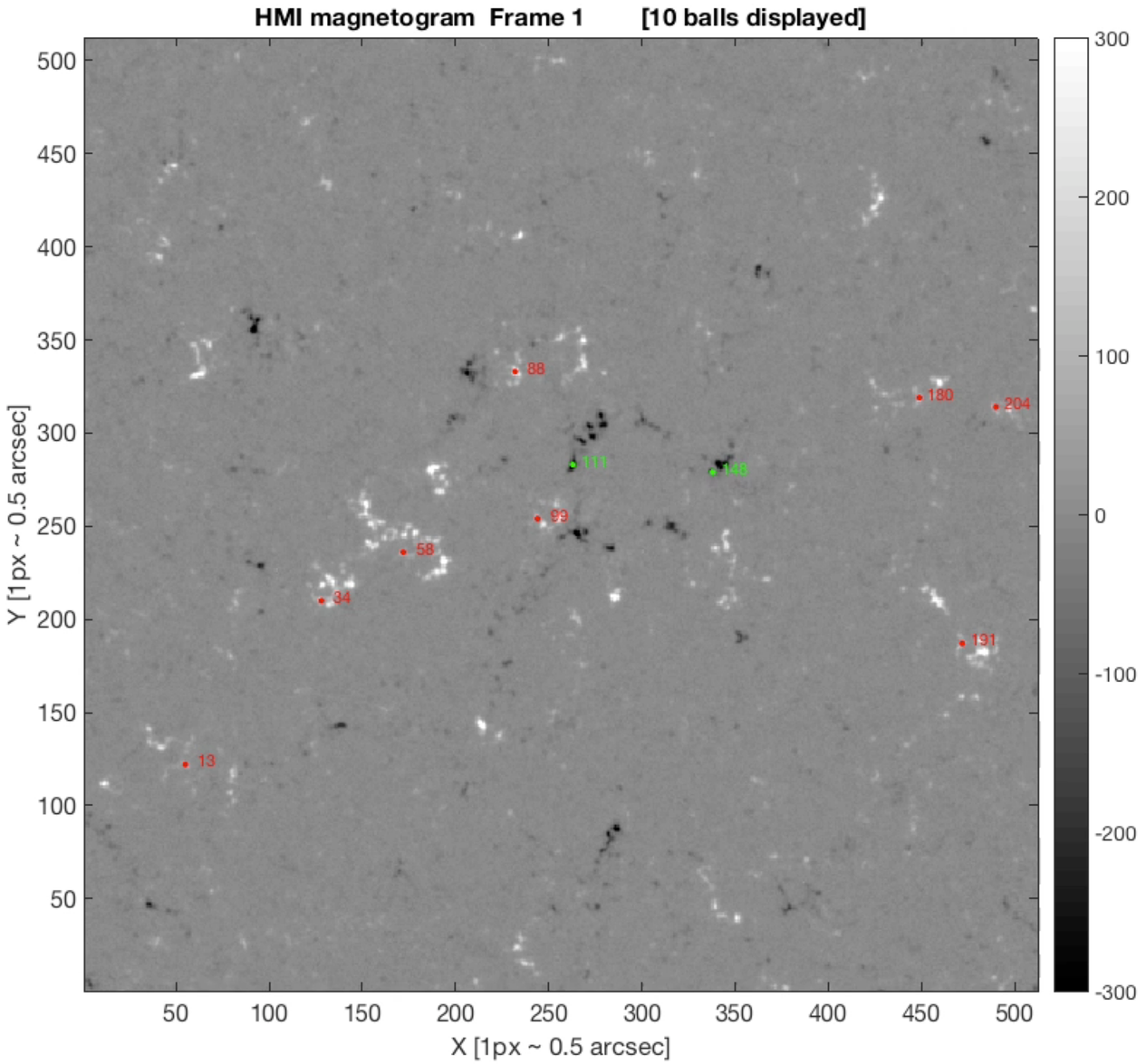

Supergranules
Earth (for scale)
Y (Mm)
X (Mm)
0
10
20
30
40
50
60
70
80
0
20
40
60
80
100
120
140
160

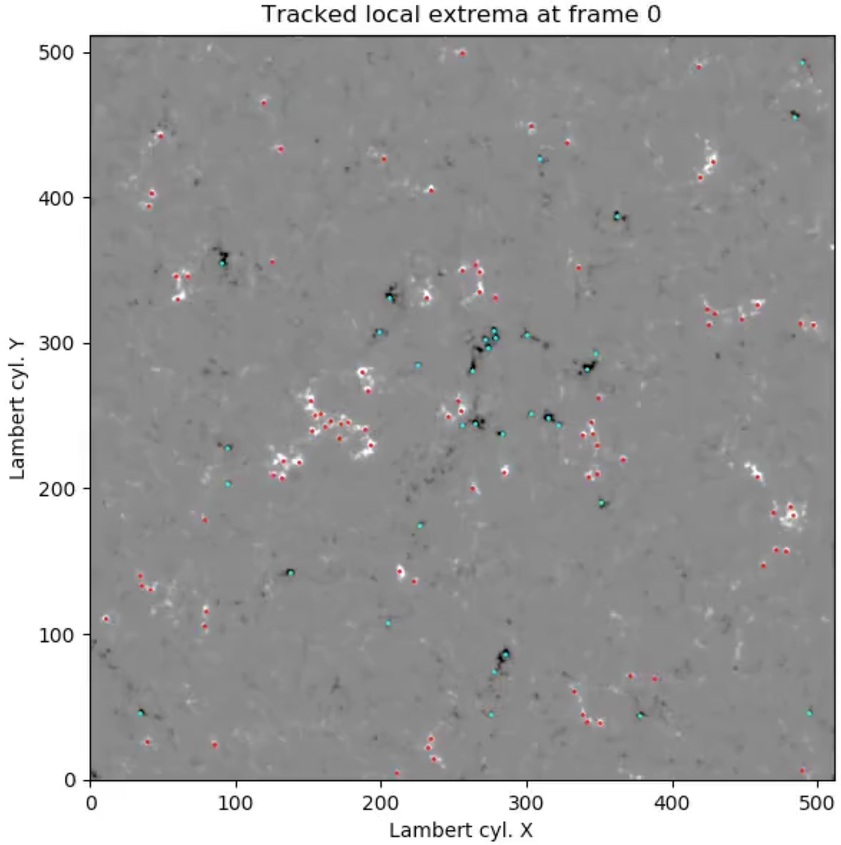
“Magnetic Balltracking”:

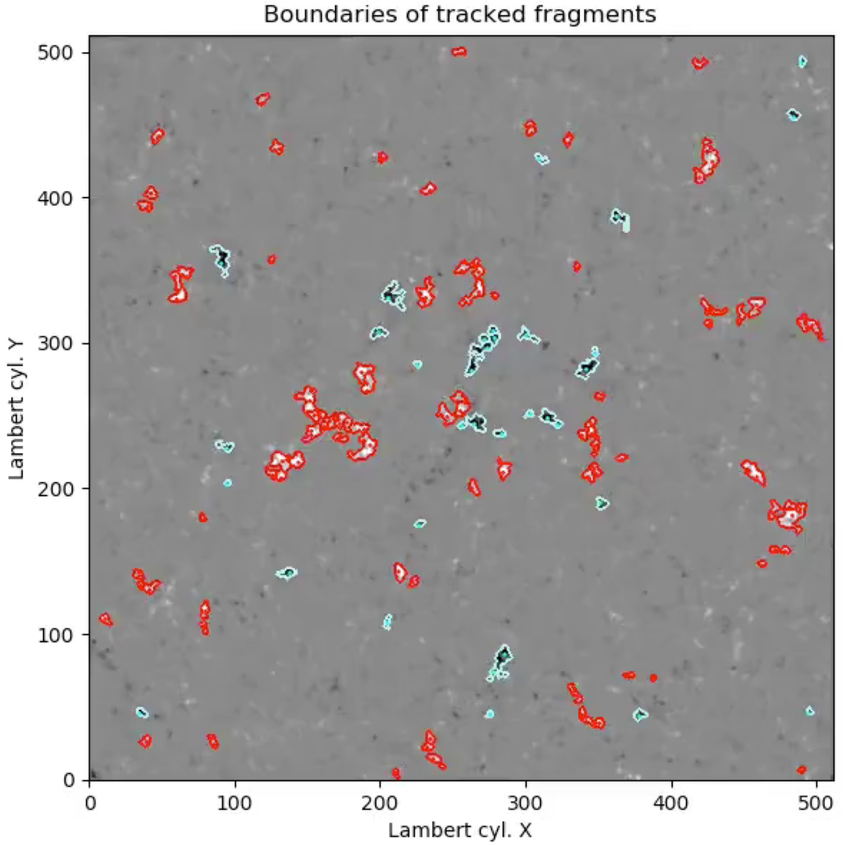
Highly efficient tracking and univocal labelling of moving magnetic features

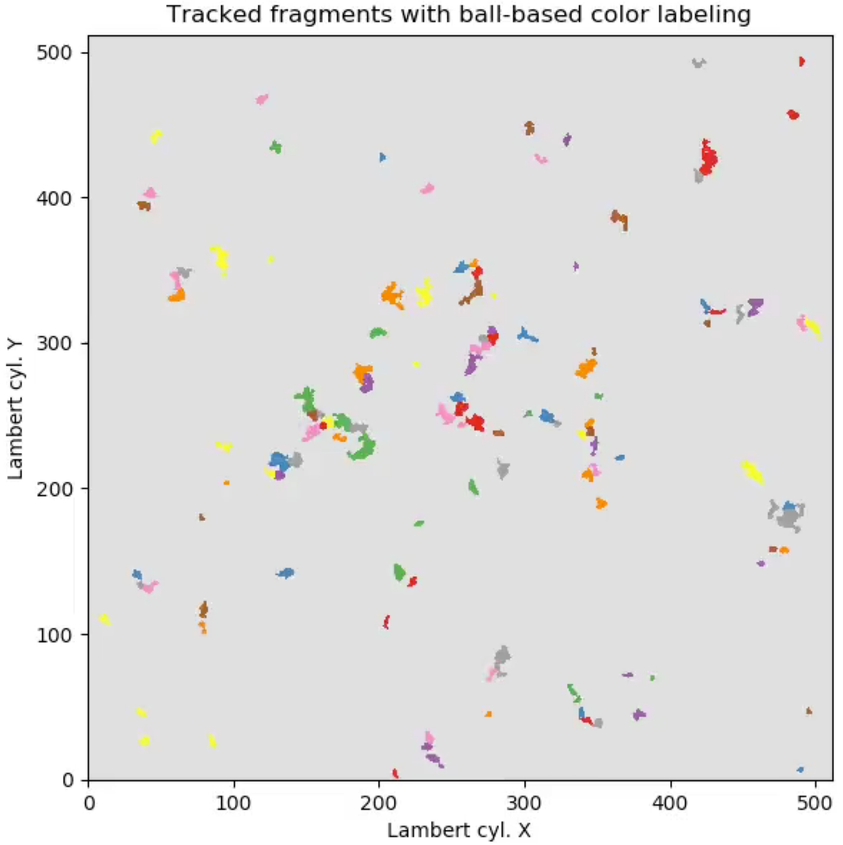


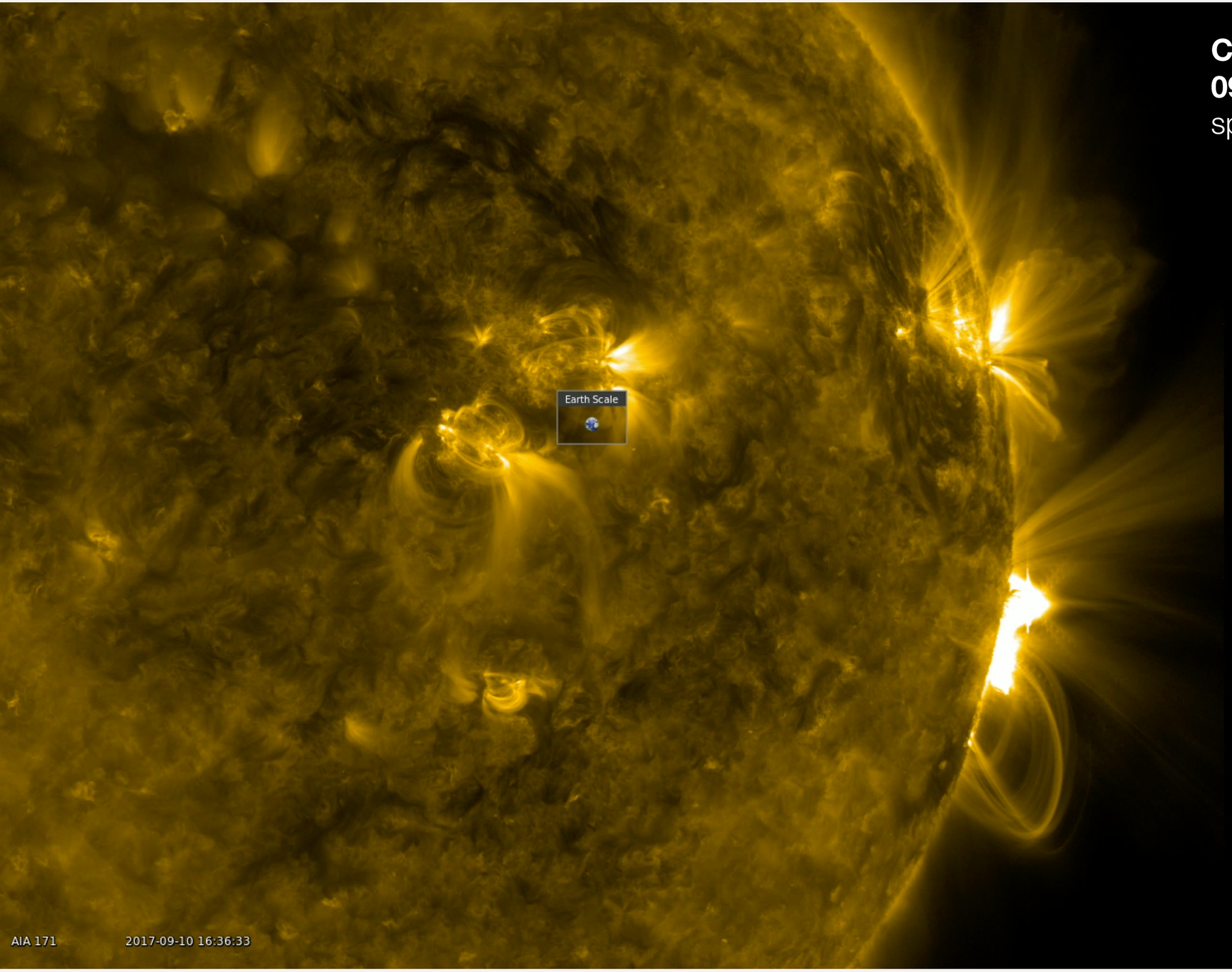



HMI magnetogram Frame 1 [10 balls displayed]
Y [1px ~ 0.5 arcsec]
X [1px ~ 0.5 arcsec]
88
180
204
111
148
99
58
34
191
13
300
200
100
0
-100
-200
-300


Tracked local extrema at frame 0
Lambert cyl. Y
Lambert cyl. X
0
100
200
300
400
500


Boundaries of tracked fragments
Lambert cyl. Y
Lambert cyl. X
0
100
200
300
400
500


Tracked fragments with ball-based color labeling
Lambert cyl. Y
Lambert cyl. X
0
100
200
300
400
500



Case study of the X-class flare from 09/10/2017: Tracking optical flows to support space weather forecasting efforts

www.helioviewer.org

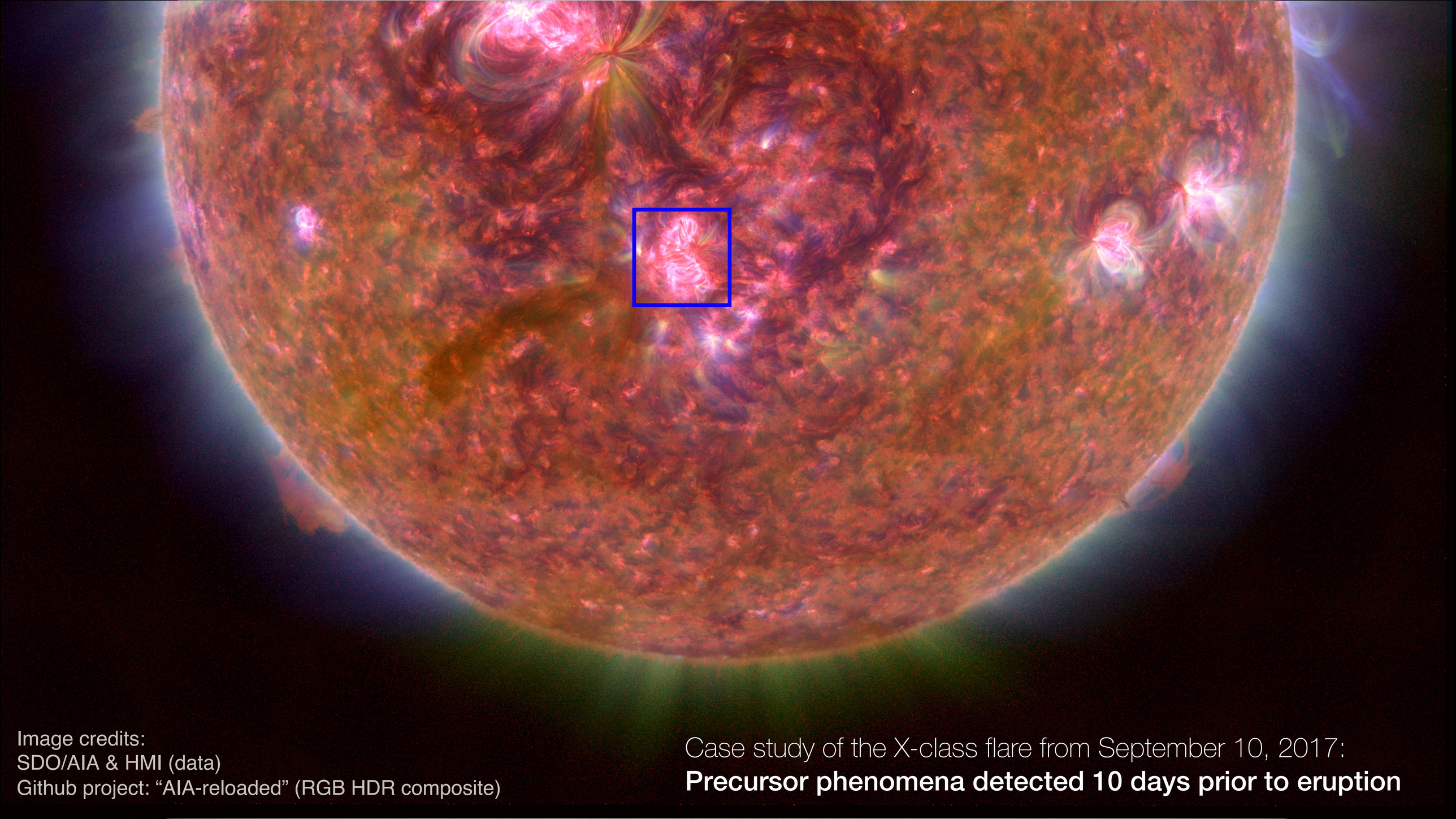
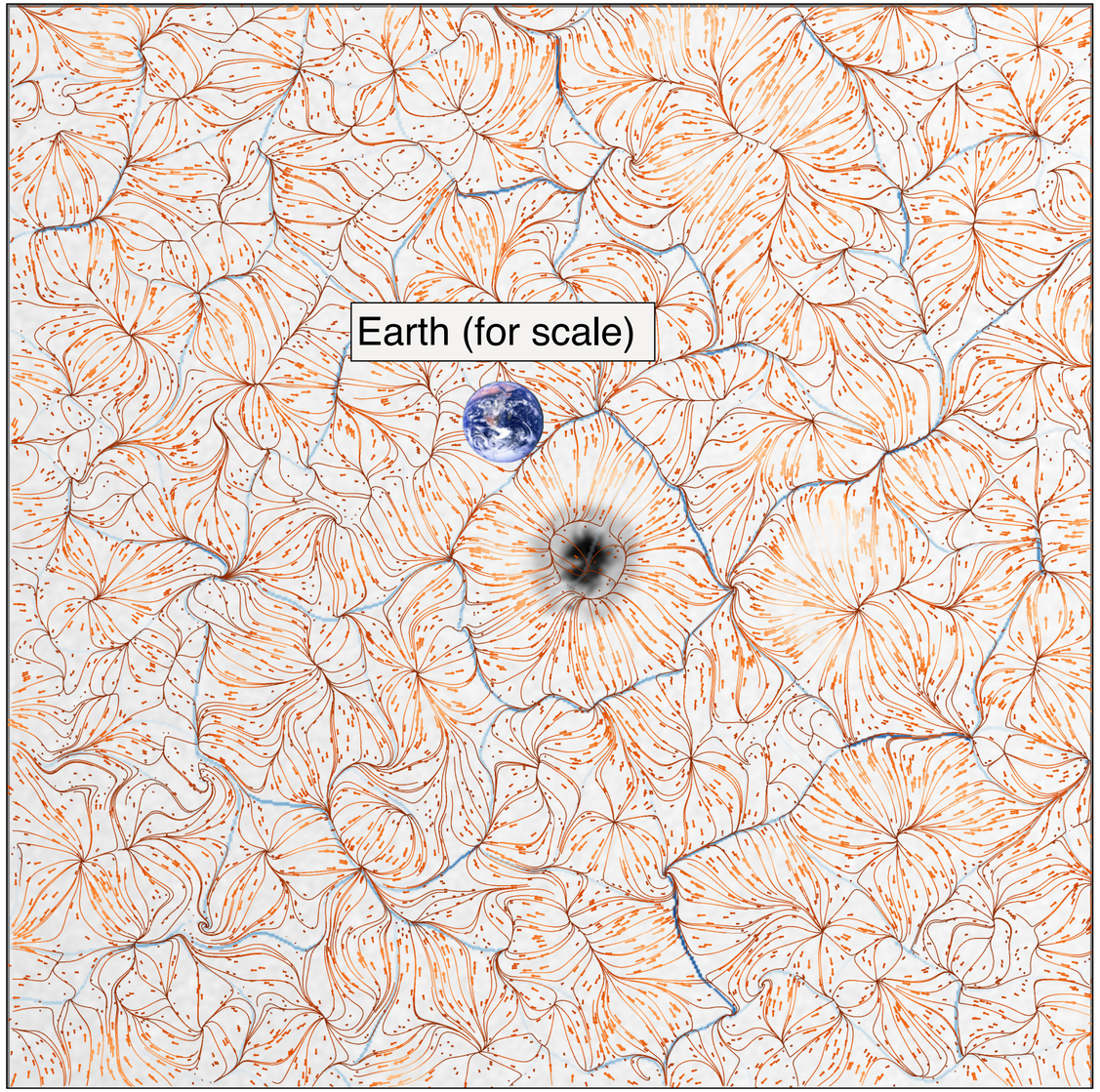
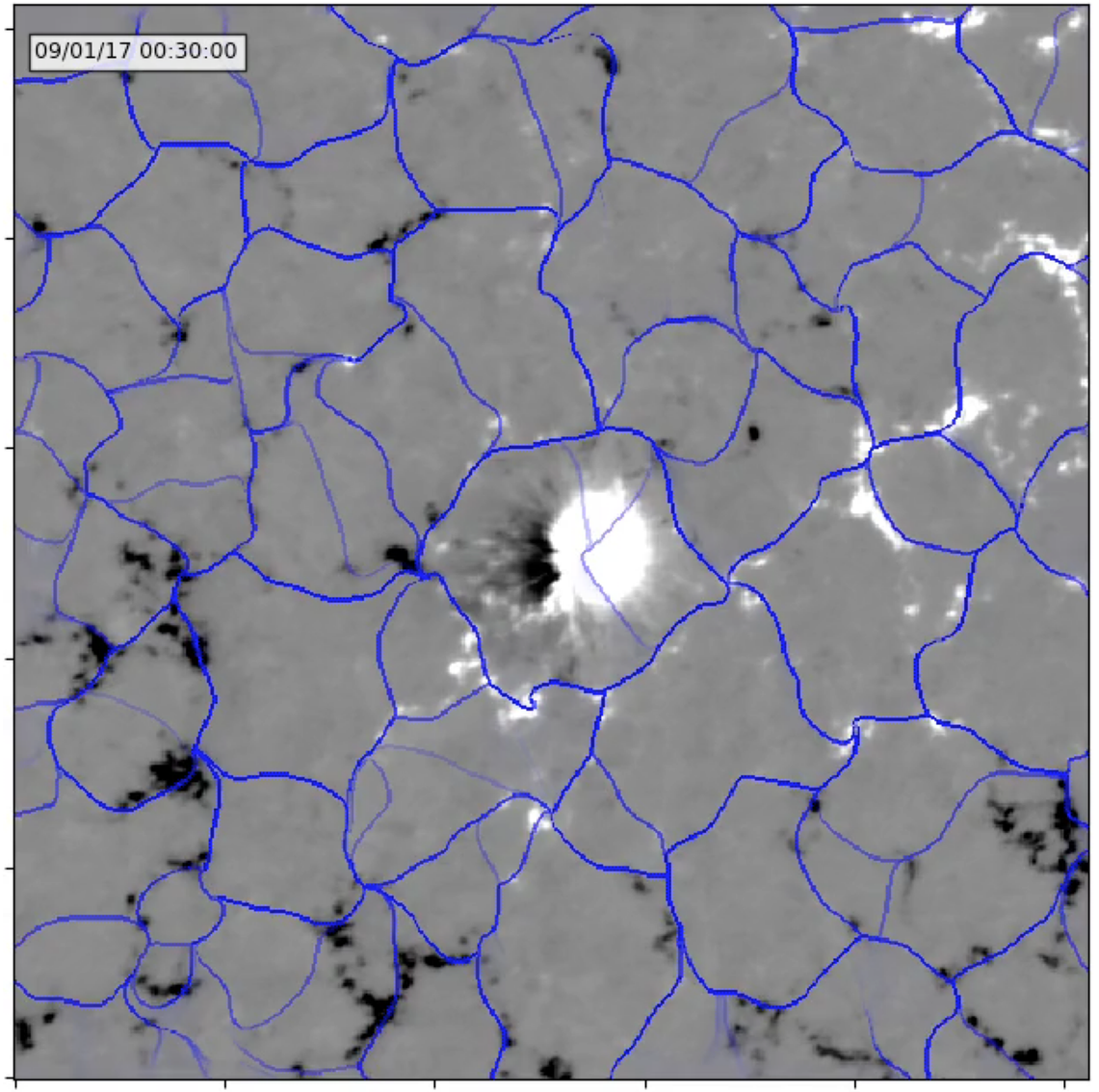


Image credits:
SDO/AIA & HMI (data)
Github project: “AIA-reloaded” (RGB HDR composite)

Case study of the X-class flare from September 10, 2017:
Precursor phenomena detected 10 days prior to eruption





Attie et al. (2018): https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018SW001939

Optical flows in industry: self driving cars & drones

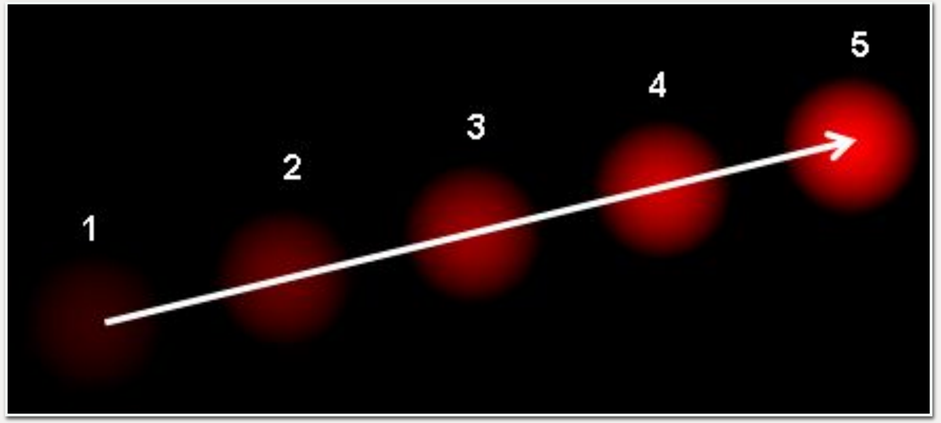
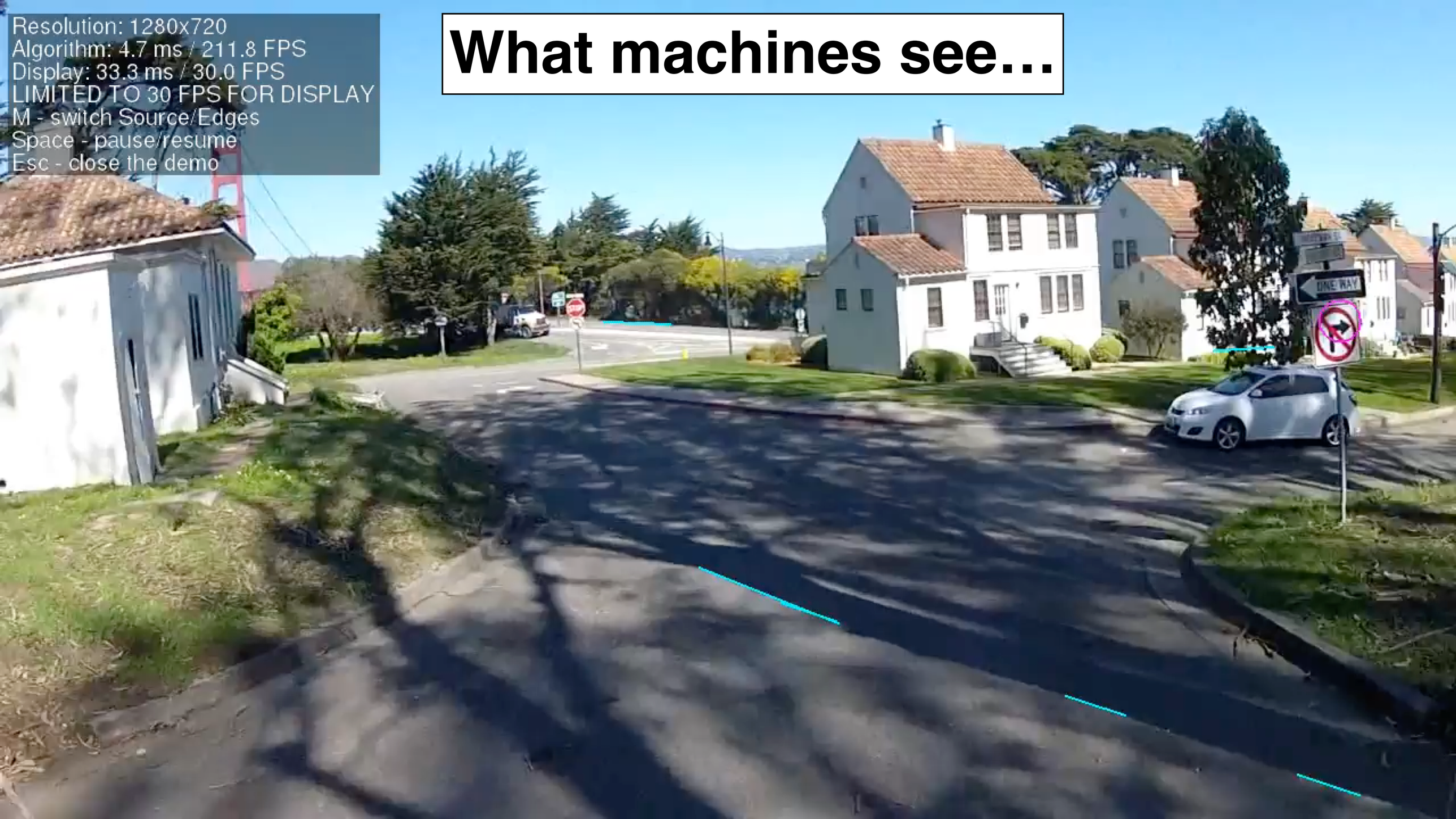
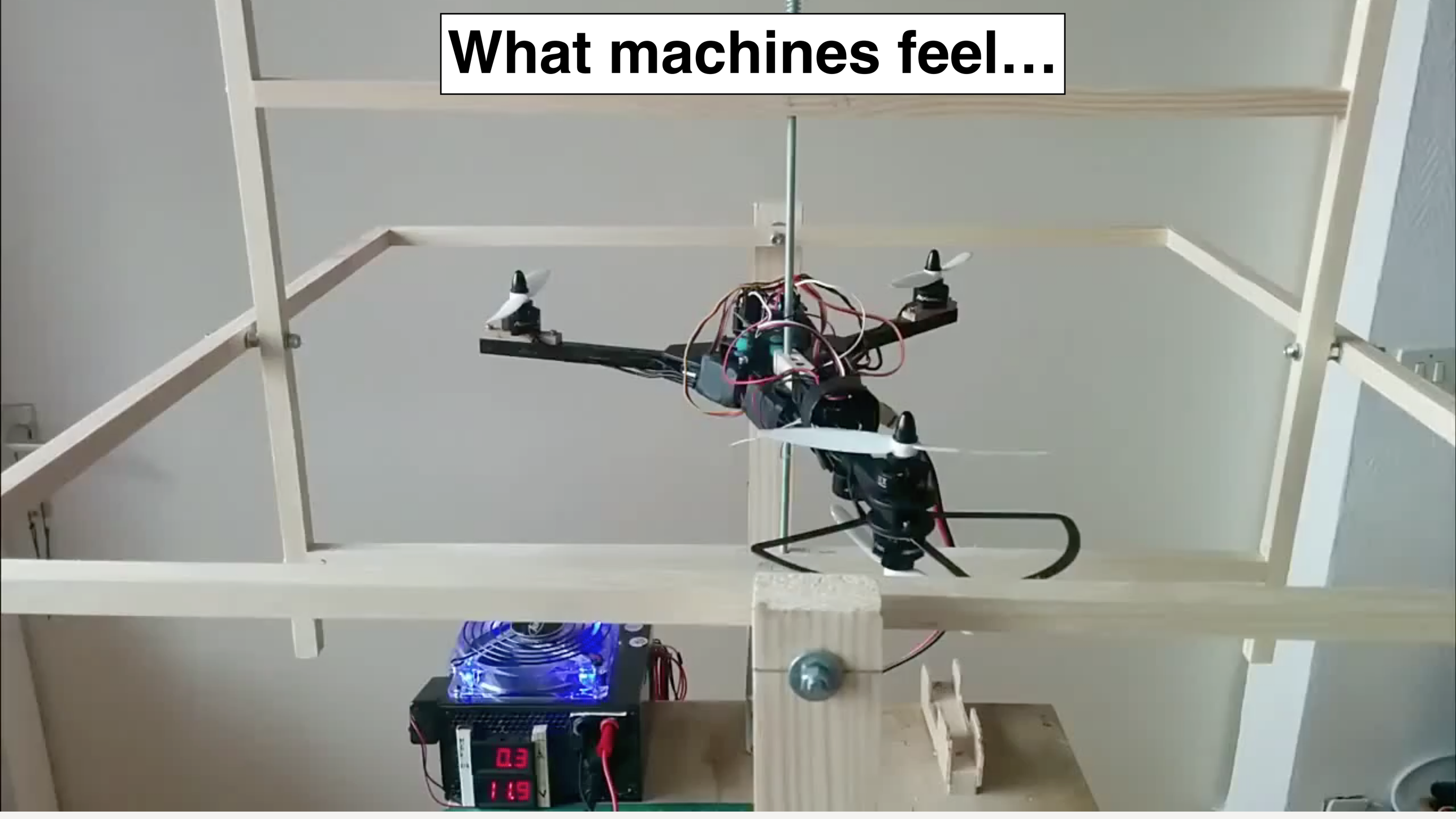


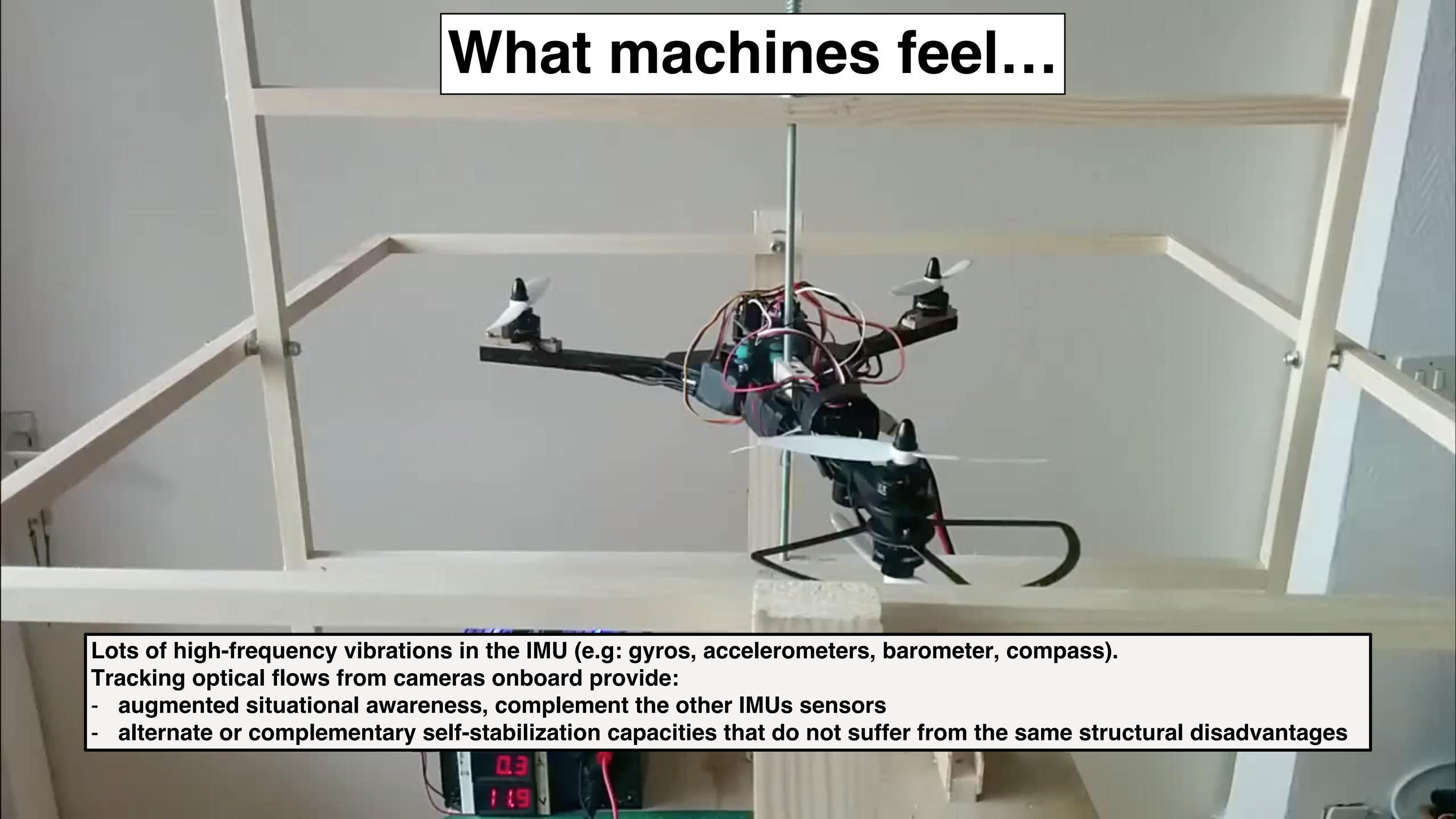
Image credits: opencv
https://docs.opencv.org/3.4/d7/d8b/tutorial_py_lucas_kanade.html


Resolution: 1280x720
Algorithm: 4.7 ms / 211.8 FPS
Display: 33.3 ms / 30.0 FPS
LIMITED TO 30 FPS FOR DISPLAY
M - switch Source/Edges
Space - pause/resume
Esc - close the demo
What machines see…
ONE WAY

What machines feel…

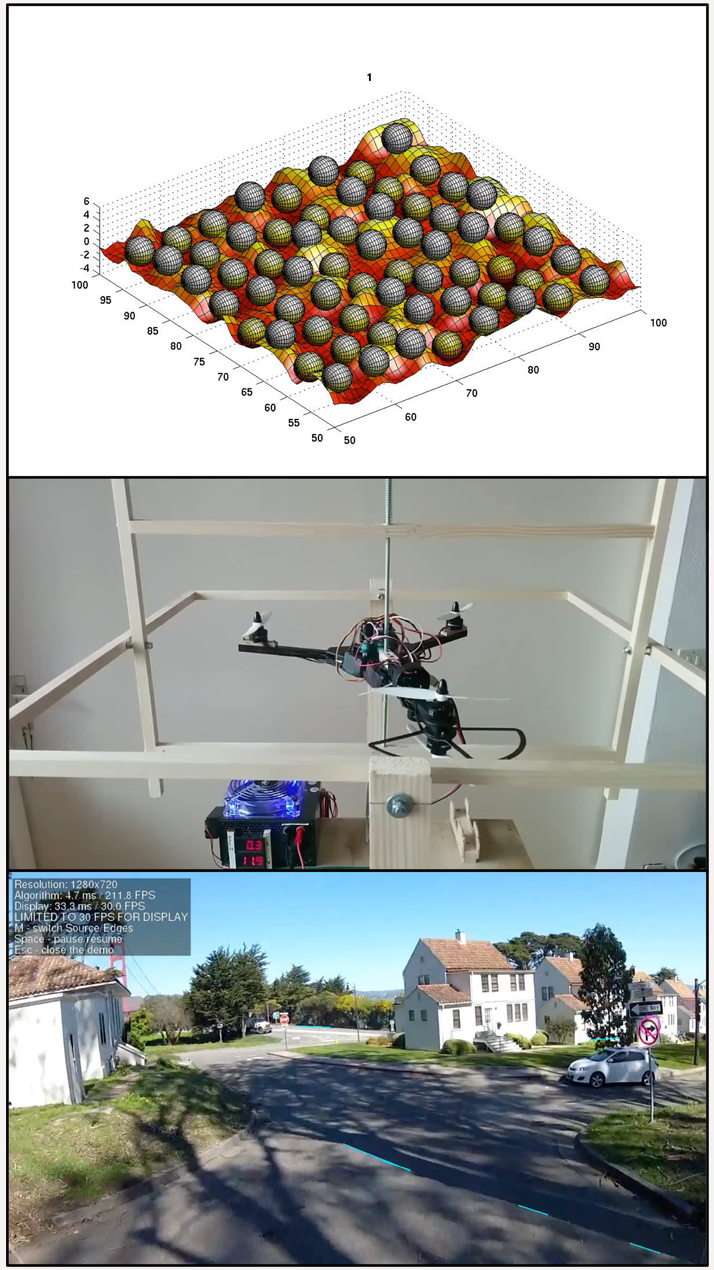


What machines feel…

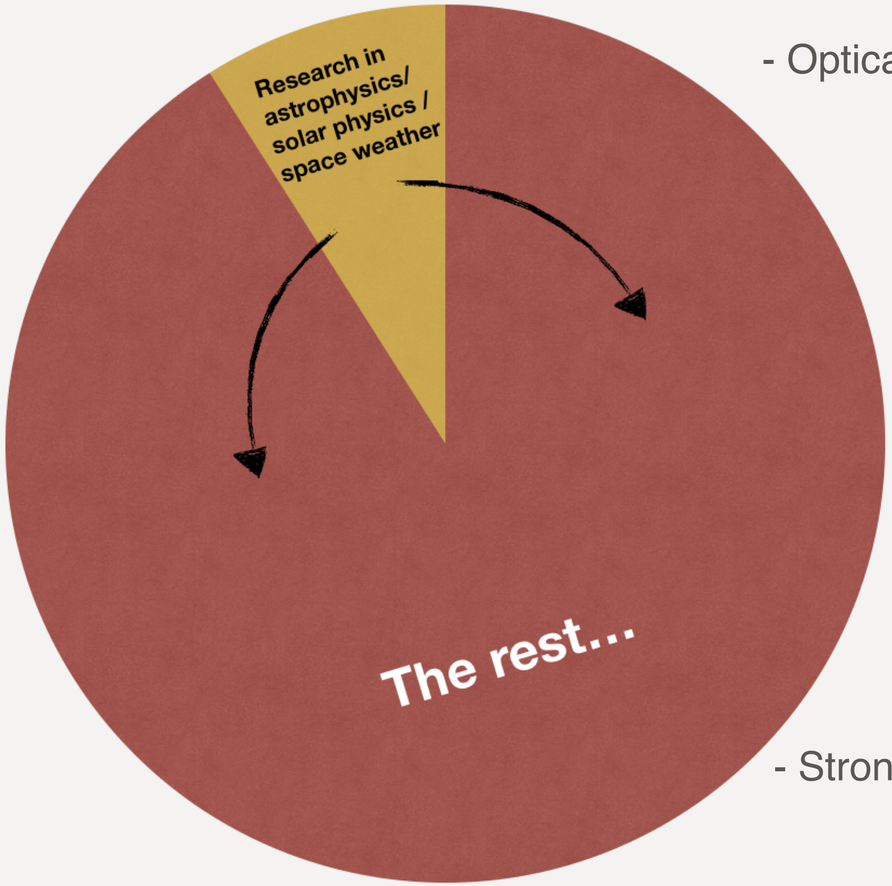


Lots of high-frequency vibrations in the IMU (e.g: gyros, accelerometers, barometer, compass).
Tracking optical flows from cameras onboard provide:

- augmented situational awareness, complement the other IMUs sensors
- alternate or complementary self-stabilization capacities that do not suffer from the same structural disadvantages



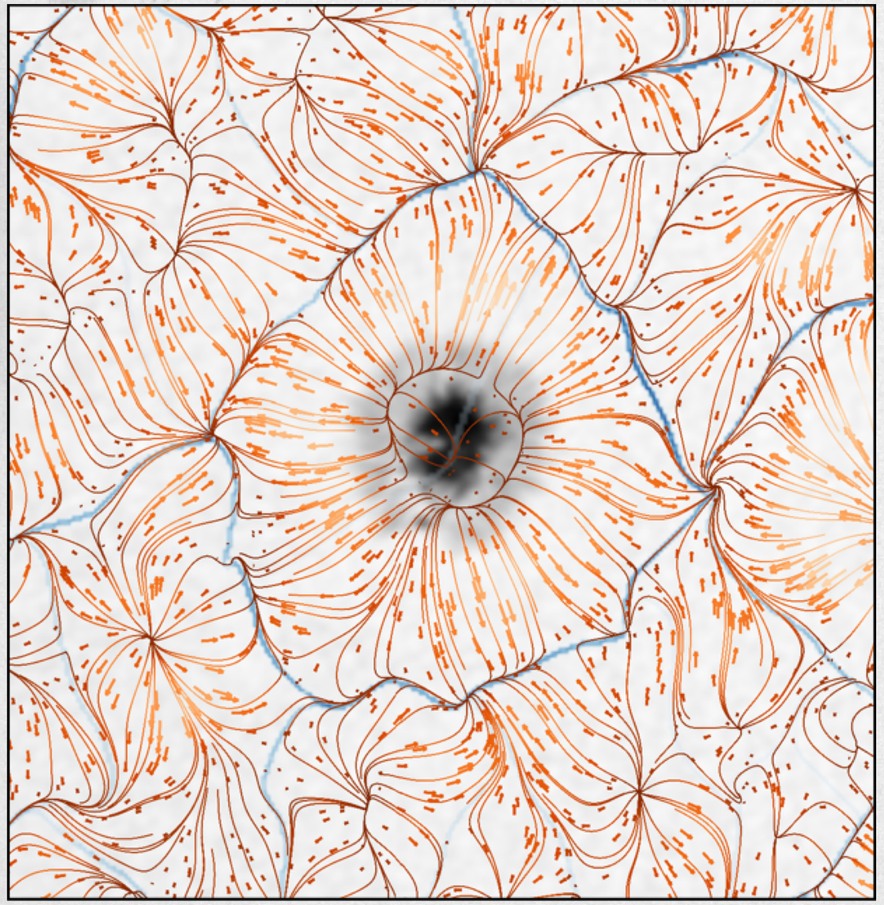
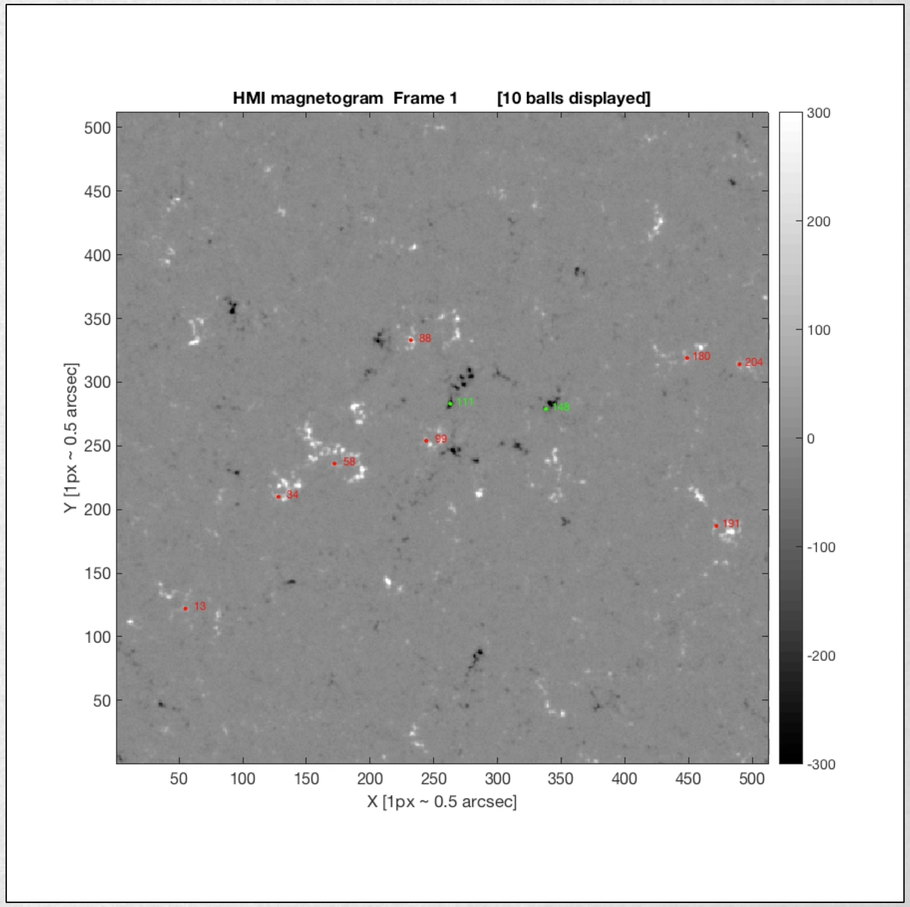
Bridging both worlds…



- Optical flows wherever we put cameras

- A whole industry pushing toward efficient algorithms involving A.I

- Worth venturing outside the academic “comfort zone”

- Strong potential to augment our discovery space

- **Experienced in optical flows? Try tracking SDO data (freely available).**
- **Tracking framework is an open source code available on [Bitbucket:](#) sdo-tracking-framework**
- **More info: raphael.attie@nasa.gov**