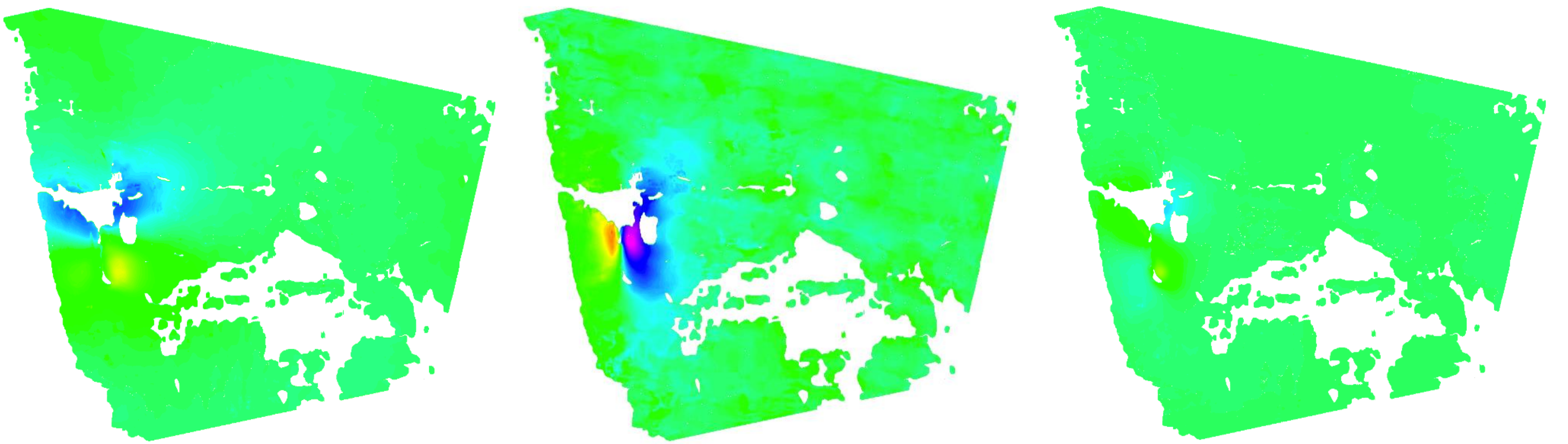
From InSAR range change to surface displacements and models



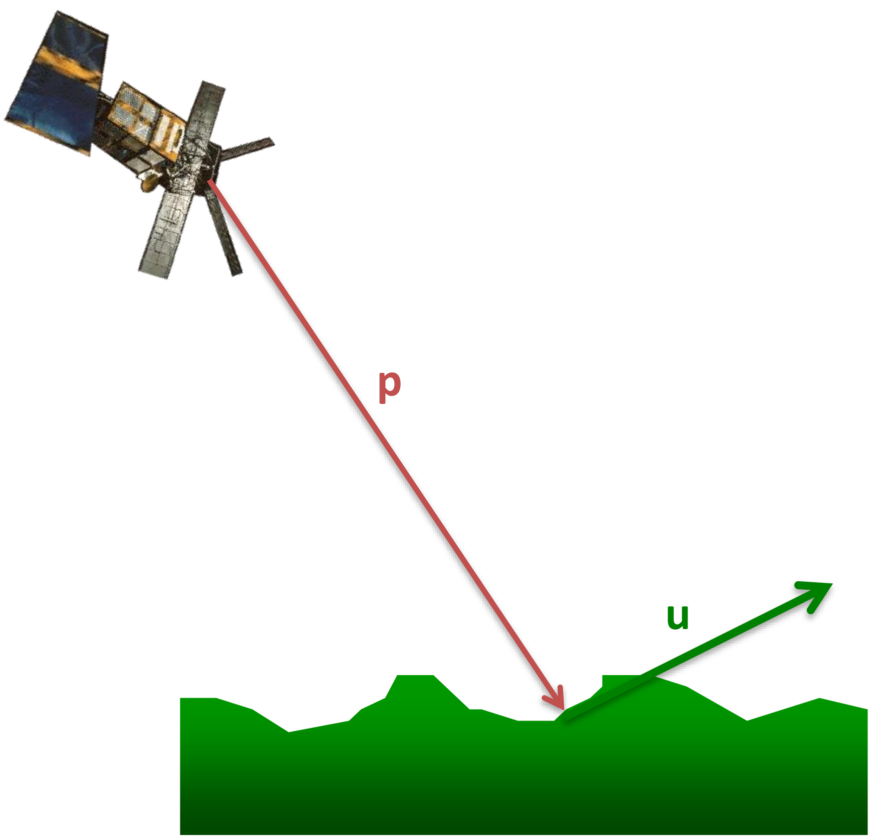
Gareth Funning

University of California, Riverside

Outline

- The line-of-sight vector: how surface displacements in 3D relate to range change
- Solving for 3D displacements; how to constrain the N-S component
- Data downsampling; preparation for modeling

Vector description of InSAR



u = ground displacement vector

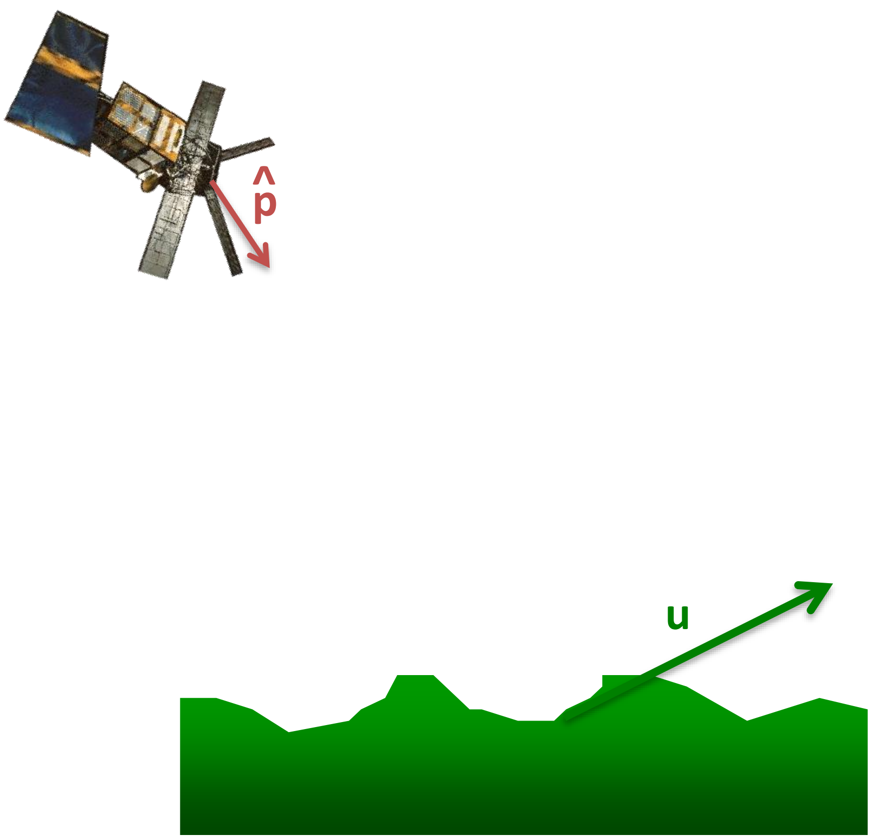
p = pointing vector (from satellite to ground target)

p is controlled by the satellite trajectory, beam mode (incidence angle) and position of the pixel within the swath

The unit pointing vector

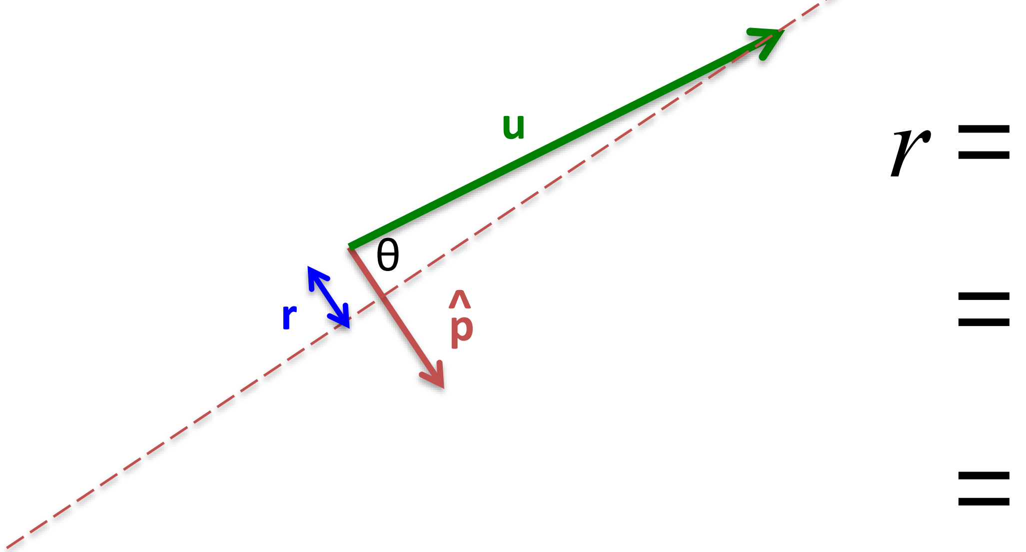
u = ground displacement vector

$\hat{\mathbf{p}}$ = <u>unit</u> pointing vector (from satellite to ground target)



Range change

the scalar (dot) product of **u** and $\hat{\mathbf{p}}$ is the 'range change' (r) we measure in interferograms



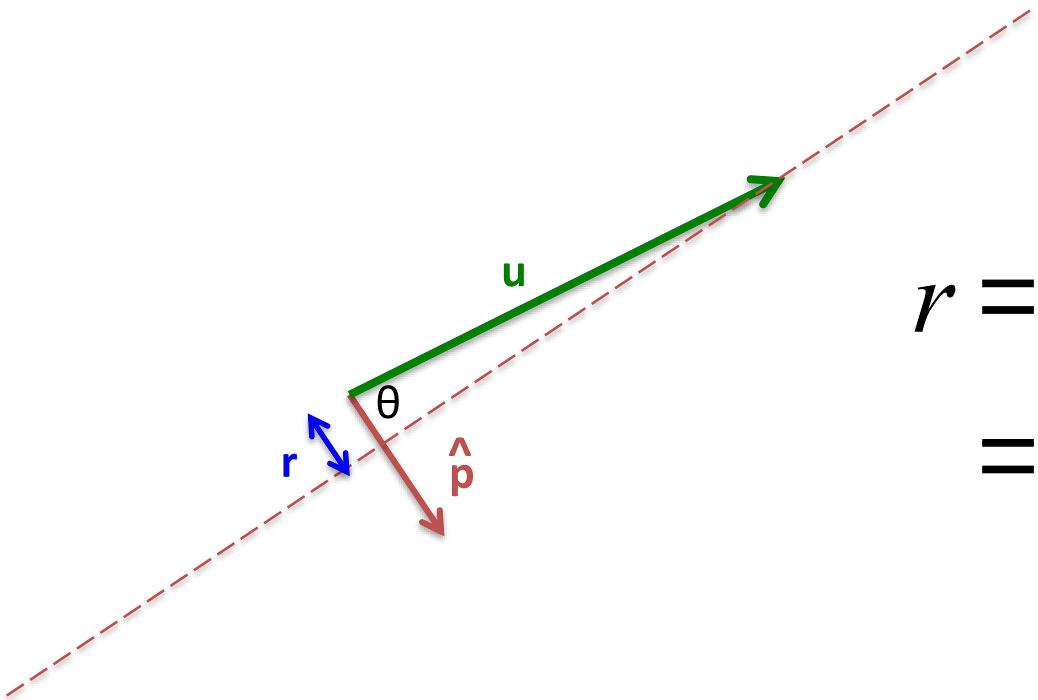
$$r = \mathbf{u} \times \hat{\mathbf{p}}$$

$$= |\mathbf{u}||\hat{\mathbf{p}}| \cos q$$

$$= |\mathbf{u}| \cos q$$

cross-section view

These vectors are 3D!



$$r = \mathbf{u} \times \hat{\mathbf{p}}$$

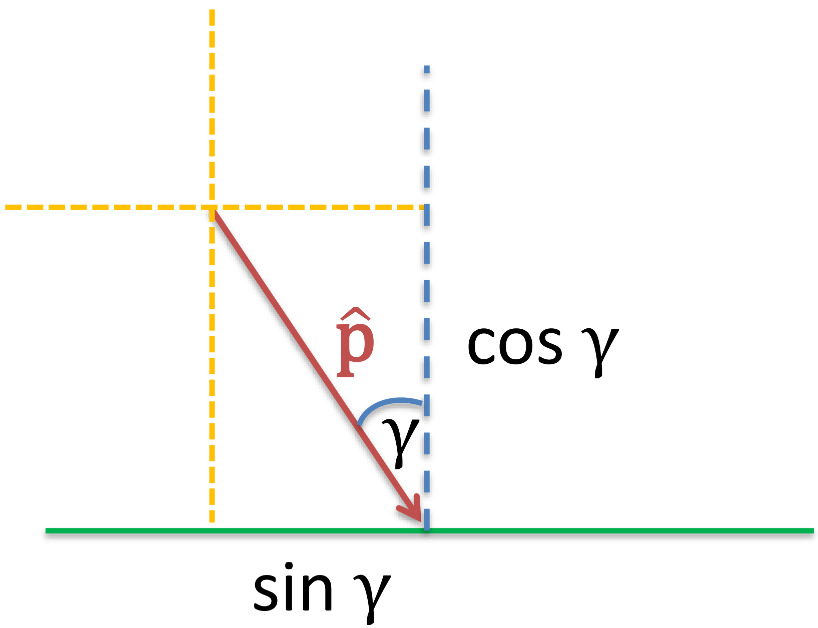
$$= u_x \hat{p}_x + u_y \hat{p}_y + u_z \hat{p}_z$$

typically, we decompose **u** and $\hat{\mathbf{p}}$ into their Cartesian components

cross-section view

Pointing vector components

cross-section view

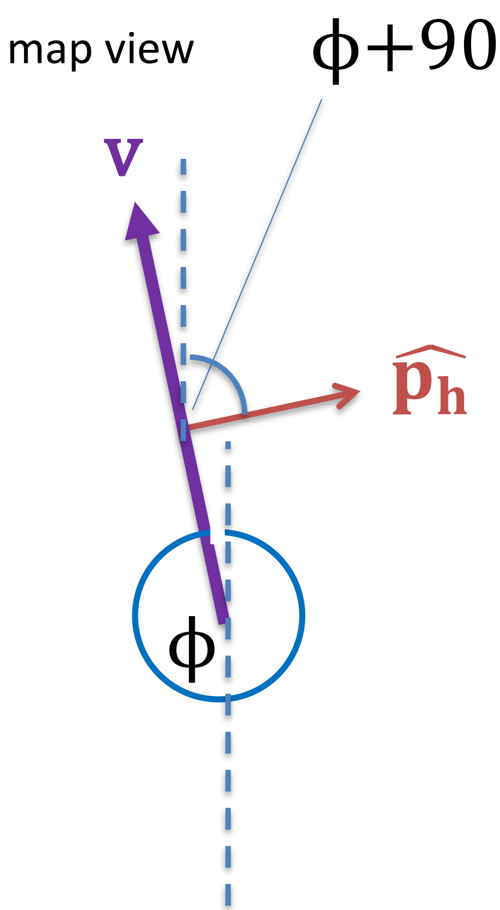


Angle of incidence of radar with ground: γ

Vertical component:
$p_z = -\cos\gamma$

Horizontal component:
$\sin\gamma = p_h = \mathrm{f}(p_x, p_y)$

Pointing vector components



'heading' (compass bearing of flight direction): ϕ

Horizontal component:

$\sin\gamma = p_h = f(p_x, p_y)$

$$p_x = \sin\gamma\cos\phi$$
$$p_y = -\sin\gamma\sin\phi$$

Pointing vector components

$$[p_x \ \ p_y \ \ p_z] = [\sin\gamma\cos\phi \quad -\sin\gamma\sin\phi \quad -\cos\gamma]$$

ϕ is the heading
γ is the incidence angle

Note:

1. Incidence can vary within a SAR image (8–10° for stripmap modes, up to ~20° for wide-swath)
2. In TOPS mode (e.g. for Sentinel-1), the heading can vary too!
3. Most processing software will output ϕ and γ (although not always in an easily parsed form…)

WARNING

ISCE does not output heading (azimuth) direction in a 'geographical' convention

- It uses a right-hand rule from east (i.e. 0=east, and counts positive degrees counter-clockwise from there)
- To convert to geographical heading, multiply the azimuth by -1 and add 90°

http://earthdef.caltech.edu/boards/4/topics/327

Home Projects Help — Sign in Register

Earthdef::ISCEForum

Wiki Documents Files **Forums**

Forums » ISCE Forum »

Generating pixel-by-pixel ENU LOS vectors

Added by Piyush Agram over 3 years ago

One can easily generate ENU vectors from the point on ground to the sensor using imageMath.py. This can be done with the LOS files in radar coordinates or geocoordinates

```
imageMath.py --eval='sin(rad(a_0))*cos(rad(a_1+90));sin(rad(a_0)) * sin(rad(a_1+90));cos(rad(a_0))' --a=los.rdr.geo -t FLOAT -s BIL
```

If you already have ENU displacements (3 band file) computed from a model on the same geocoded grid and want to project it to LOS:

```
imageMath.py --eval='a_0*b_0;a_1*b_1;a_2*b_2' --a=enu.rdr.geo --b=model.geo -t FLOAT -o model_LOS.geo
```

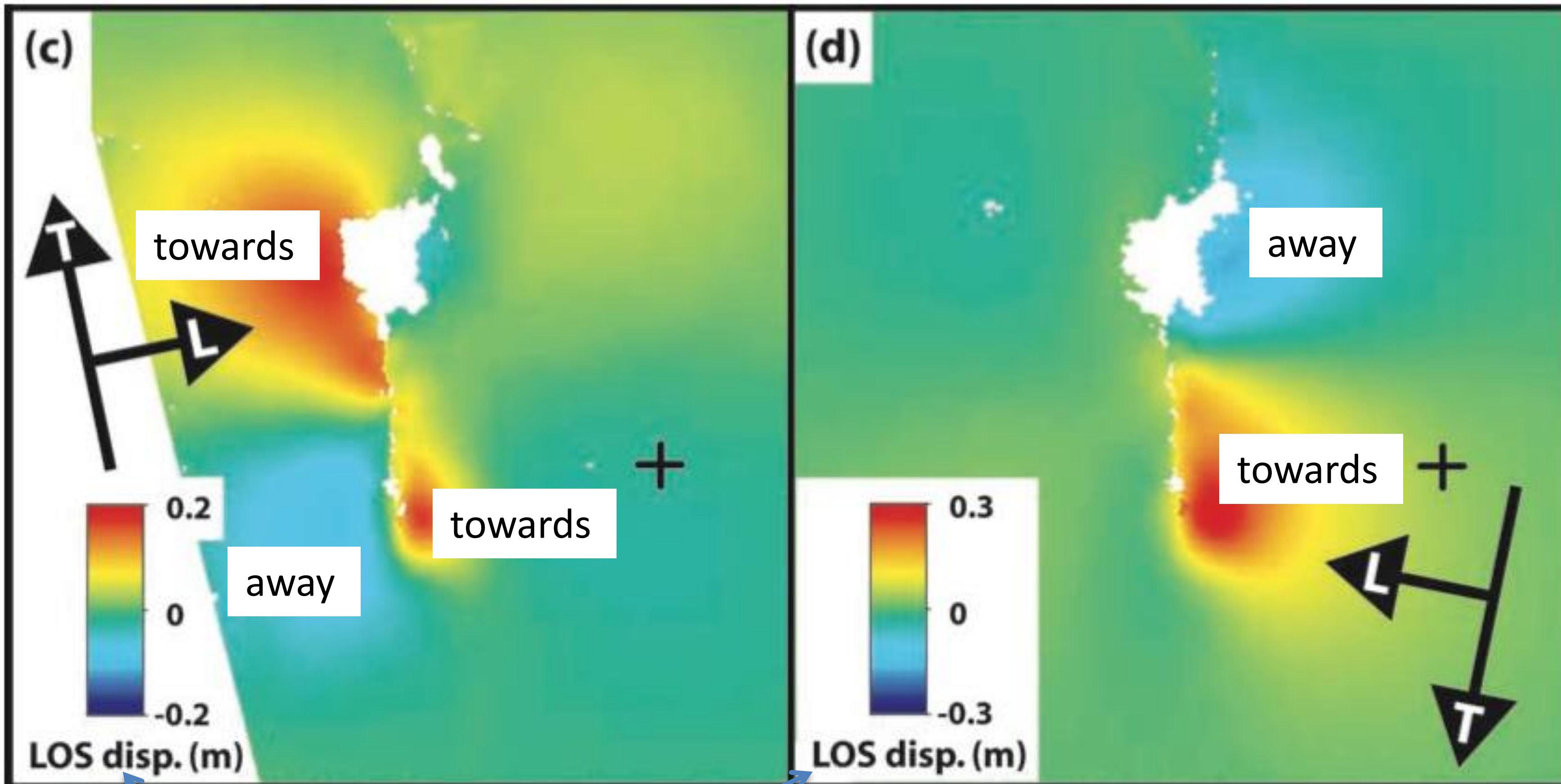
Piyush

ANOTHER WARNING

Historically, people did not all use the same sign conventions in InSAR (including me…)

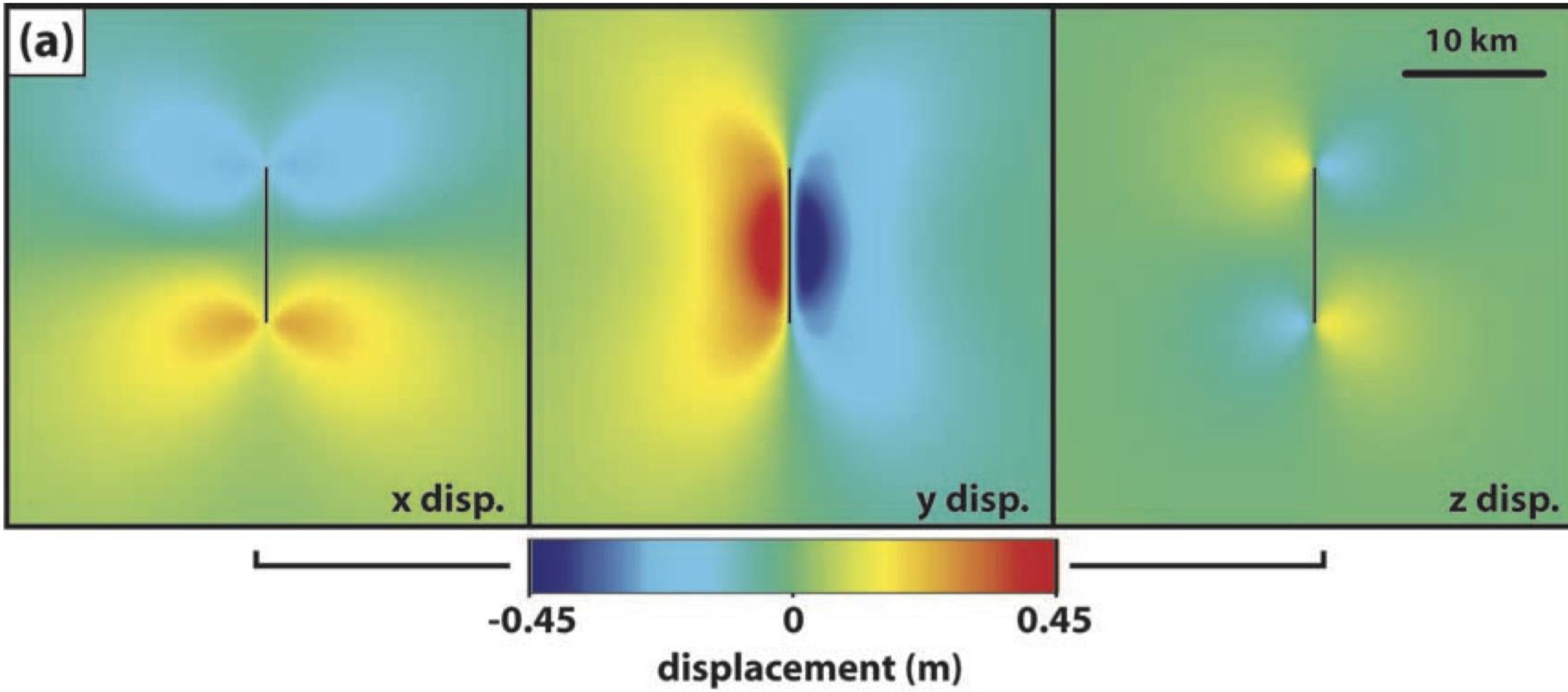
- Check whether your interferograms are ‘range change’ or ‘ground LOS displacement’
- Check if your pointing vectors are consistent with your interferograms (pointing from satellite to ground, or ground to satellite?)

Example: 2003 Bam, Iran



ground LOS displacement(!)

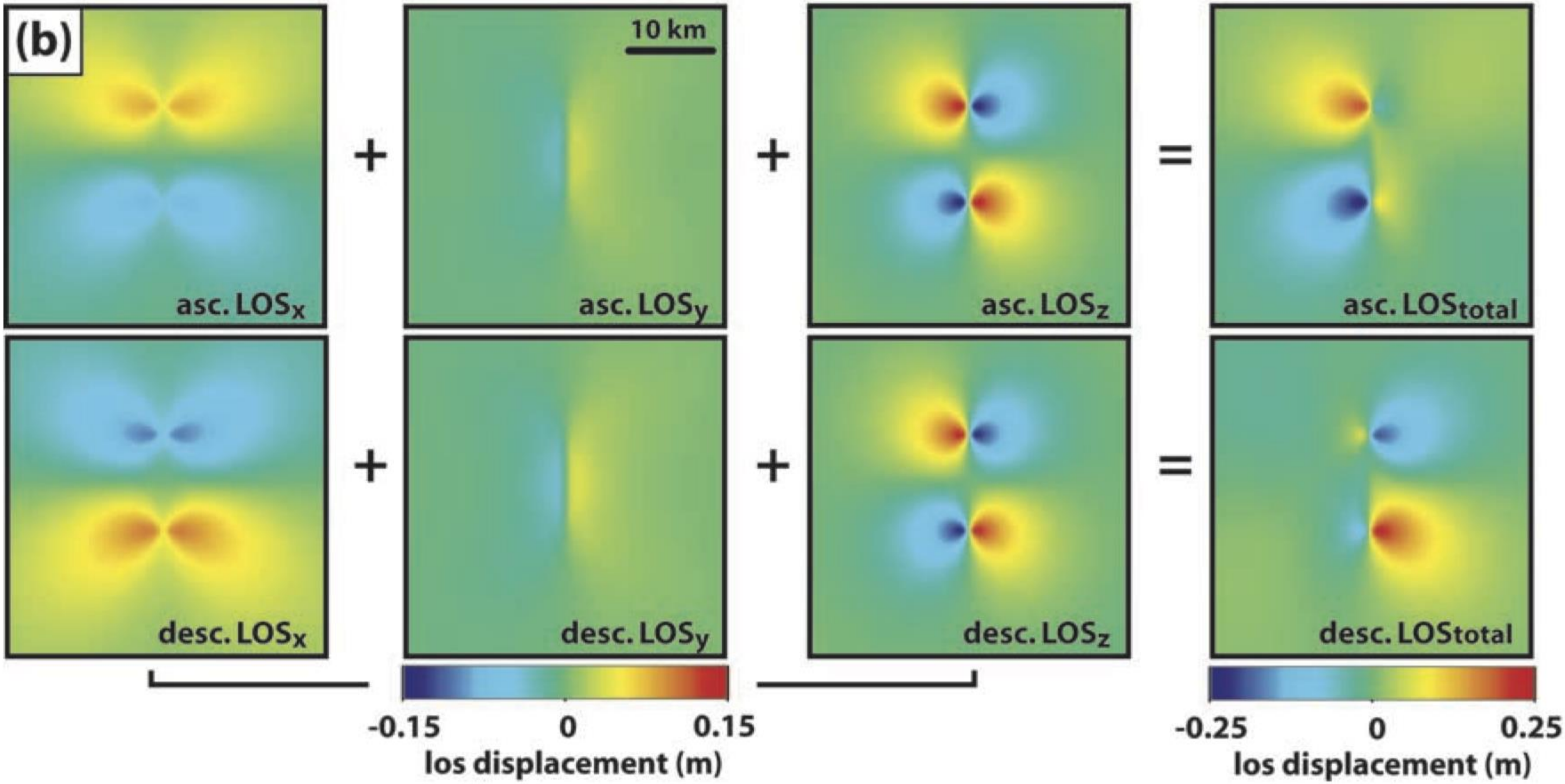
Funning et al., 2005



Forward model

- pure right-lateral strike-slip
- vertical dip
- N–S strike
- 1.8 m slip
- top 0.6 km depth
- bottom 13 km depth

Funning et al., 2005



When scaled by their pointing vector coefficients

- the north component contributes little to LOS
- the east and vertical components add on one side of the fault and cancel on the other

Funning et al., 2005

With 3 unknowns, we need 3 equations

(u_x, u_y, u_z)

3 different range-change observations…

$$r_1 = \mathbf{u} \times \hat{\mathbf{p}}_1$$

$$r_2 = \mathbf{u} \times \hat{\mathbf{p}}_2$$

$$r_3 = \mathbf{u} \times \hat{\mathbf{p}}_3$$

…from 3 different imaging geometries

[This is for a single pixel, assuming all data sets are detrended, referenced to a pixel in the far-field and sampled onto the same grid.]

expanding…

$$r_1 = \mathbf{u} \times \hat{\mathbf{p}}_1 = u_x(\hat{p}_x)_1 + u_y(\hat{p}_y)_1 + u_z(\hat{p}_z)_1$$

$$r_2 = \mathbf{u} \times \hat{\mathbf{p}}_2 = u_x(\hat{p}_x)_2 + u_y(\hat{p}_y)_2 + u_z(\hat{p}_z)_2$$

$$r_3 = \mathbf{u} \times \hat{\mathbf{p}}_3 = u_x(\hat{p}_x)_3 + u_y(\hat{p}_y)_3 + u_z(\hat{p}_z)_3$$

in matrix form, this is

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} (\hat{p}_x)_1 & (\hat{p}_y)_1 & (\hat{p}_z)_1 \\ (\hat{p}_x)_2 & (\hat{p}_y)_2 & (\hat{p}_z)_2 \\ (\hat{p}_x)_3 & (\hat{p}_y)_3 & (\hat{p}_z)_3 \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}$$

$$\mathbf{r} = \mathbf{P}\,\mathbf{u}$$

this can be solved by standard least squares methods:

$$\mathbf{r} = \mathbf{P}\,\mathbf{u}$$

$$\mathbf{P}^T\mathbf{r} = \mathbf{P}^T\,\mathbf{P}\,\mathbf{u}$$

$$(\mathbf{P}^T\,\mathbf{P})^{-1}\,\mathbf{P}^T\mathbf{r} = (\mathbf{P}^T\,\mathbf{P})^{-1}\,\mathbf{P}^T\,\mathbf{P}\,\mathbf{u}$$

$$\boxed{(\mathbf{P}^T\,\mathbf{P})^{-1}\,\mathbf{P}^T\mathbf{r} = \mathbf{u}}$$

$$\mathbf{u} = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{r}$$

if you have estimates of the uncertainties in range change, you can use them to weight the inversion…

uncertainty in r_1

$$\mathbf{E} = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix}$$

including weighting, we get:

$$\mathbf{u} = (\mathbf{P}^T \mathbf{E}^{-1} \mathbf{P})^{-1} \mathbf{P}^T \mathbf{E}^{-1} \mathbf{r}$$

with covariances in the estimate of **u** of:

$$\mathbf{C} = (\mathbf{P}^T \mathbf{E}^{-1} \mathbf{P})^{-1}$$

at least

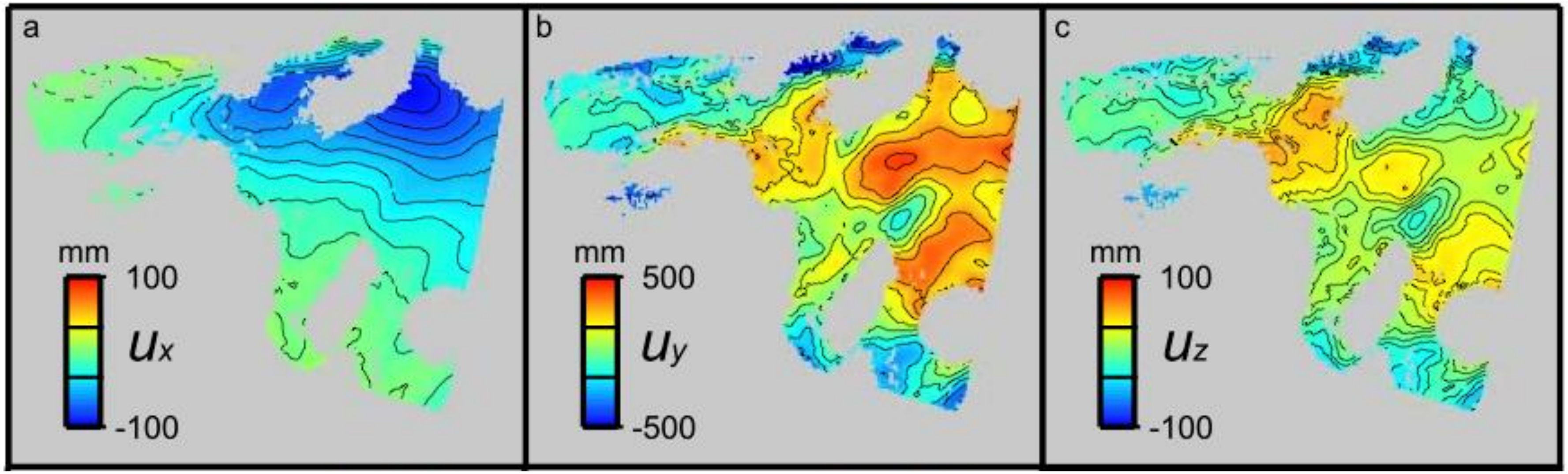
With 3 unknowns, we need ^ 3 equations

Ascending + descending + ?

- Interferogram with a different incidence angle?
- Left- and right-looking interferograms?
- Some other measure of displacement?

2002 Nenana Mountain, AK

σ_x = 6 mm σ_y = 286 mm σ_z = 41 mm



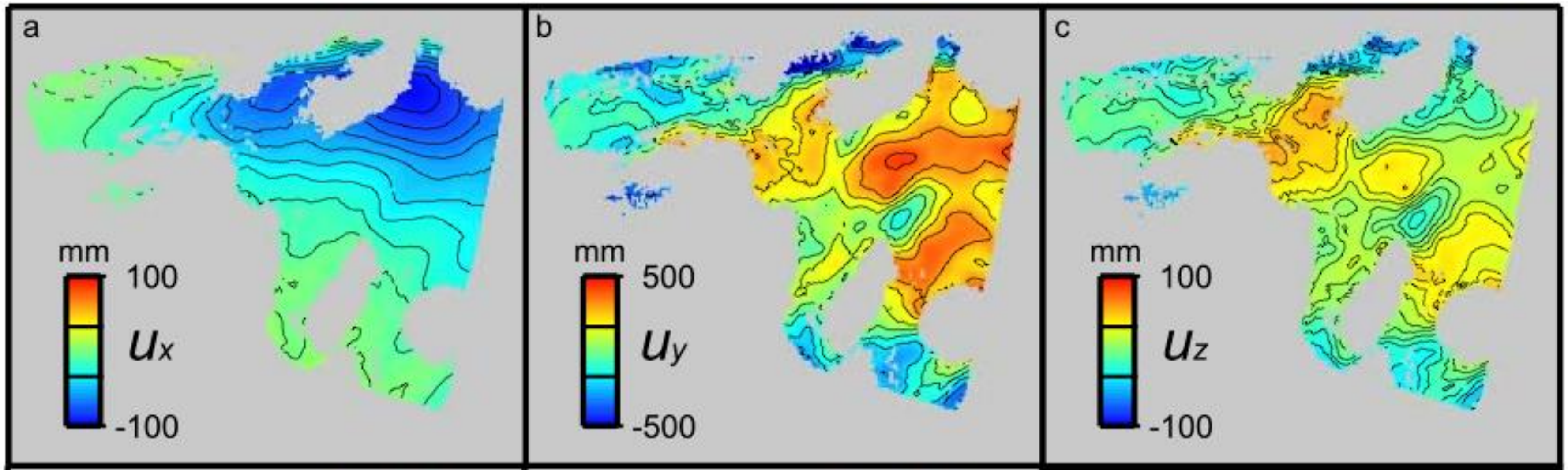
4 input interferograms:

Ascending and descending RADARSAT data with 24° incidence

Ascending and descending RADARSAT data with 45° incidence

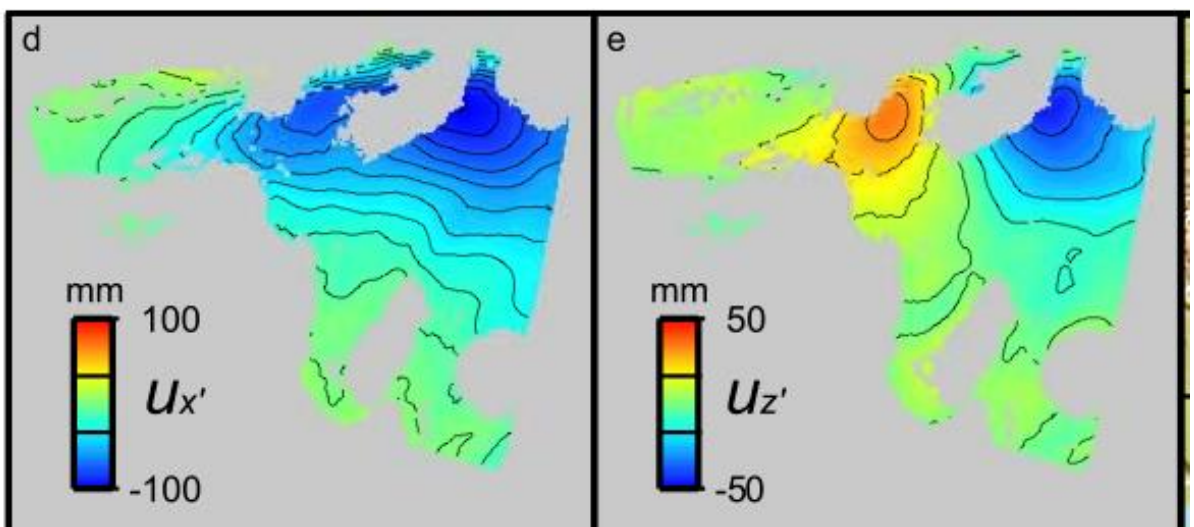
Wright et al., 2003

σ_x = 6 mm σ_y = 286 mm σ_z = 41 mm



Set u_y to zero, and you get:

$\sigma_{x'}$ = 6 mm $\sigma_{z'}$ = 4 mm



Wright et al., 2003

at least

With 3 unknowns, we need ^ 3 equations

Ascending + descending + ?

- Interferogram with a different incidence angle?
  doesn't work – not enough constraint on u_y
- Left- and right-looking interferograms?
  should work, but we have no satellites that can do it
- Some other measure of displacement?
  along-track component of deformation from azimuth offsets or MAI

[or… ascending + descending and only solve for 2!]

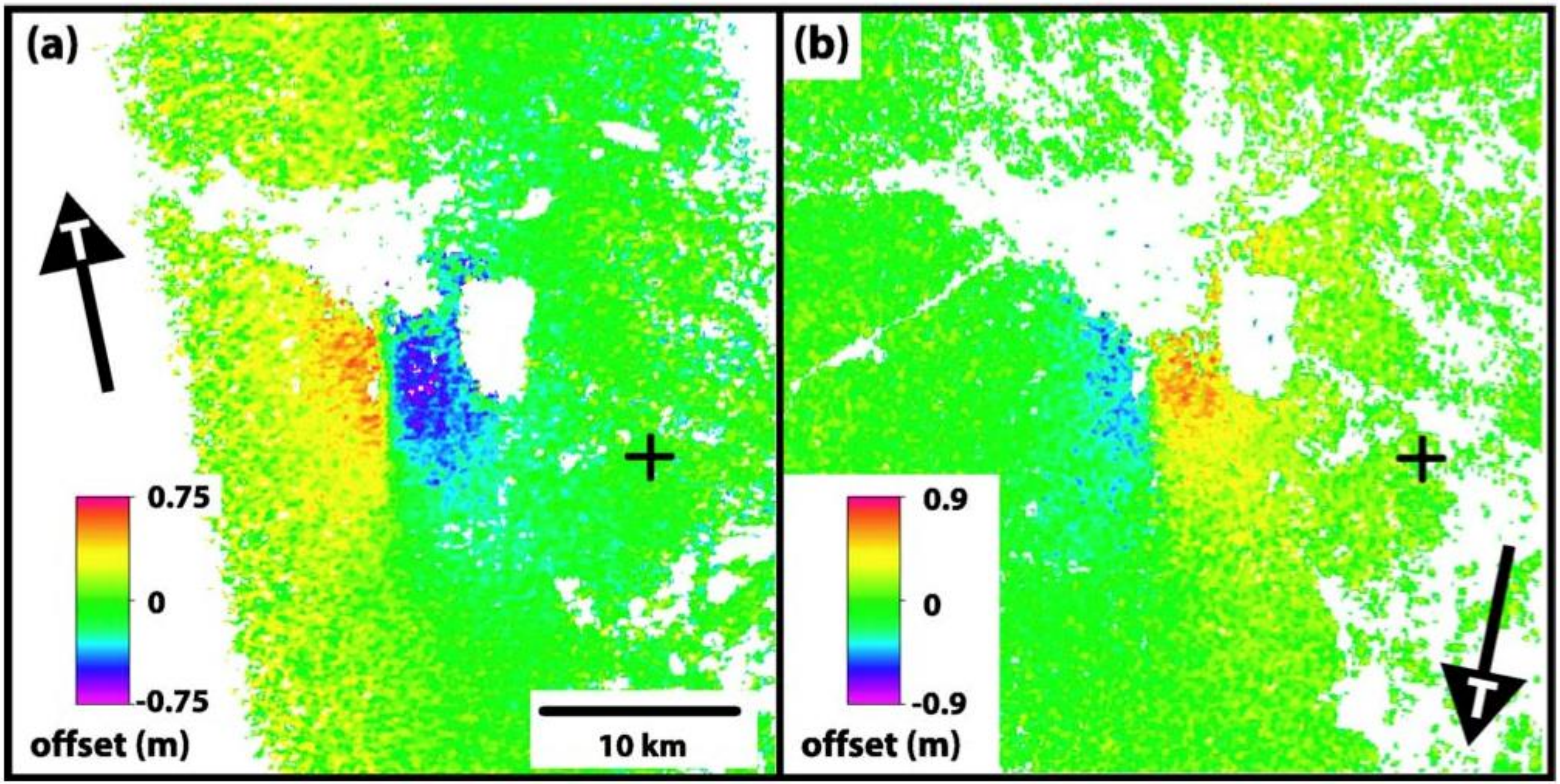
1) Azimuth offsets

- Distortion to post-earthquake SAR image caused by displacement in along-track direction
- Obtained by sub-pixel matching of the SLC images
- Same process as used to coregister SLCs for InSAR, but at much greater density (e.g. 4 range looks), with resulting increase in time taken
- Low precision compared with InSAR (10s of cm)

2003 Bam, Iran earthquake

σ = 114 mm

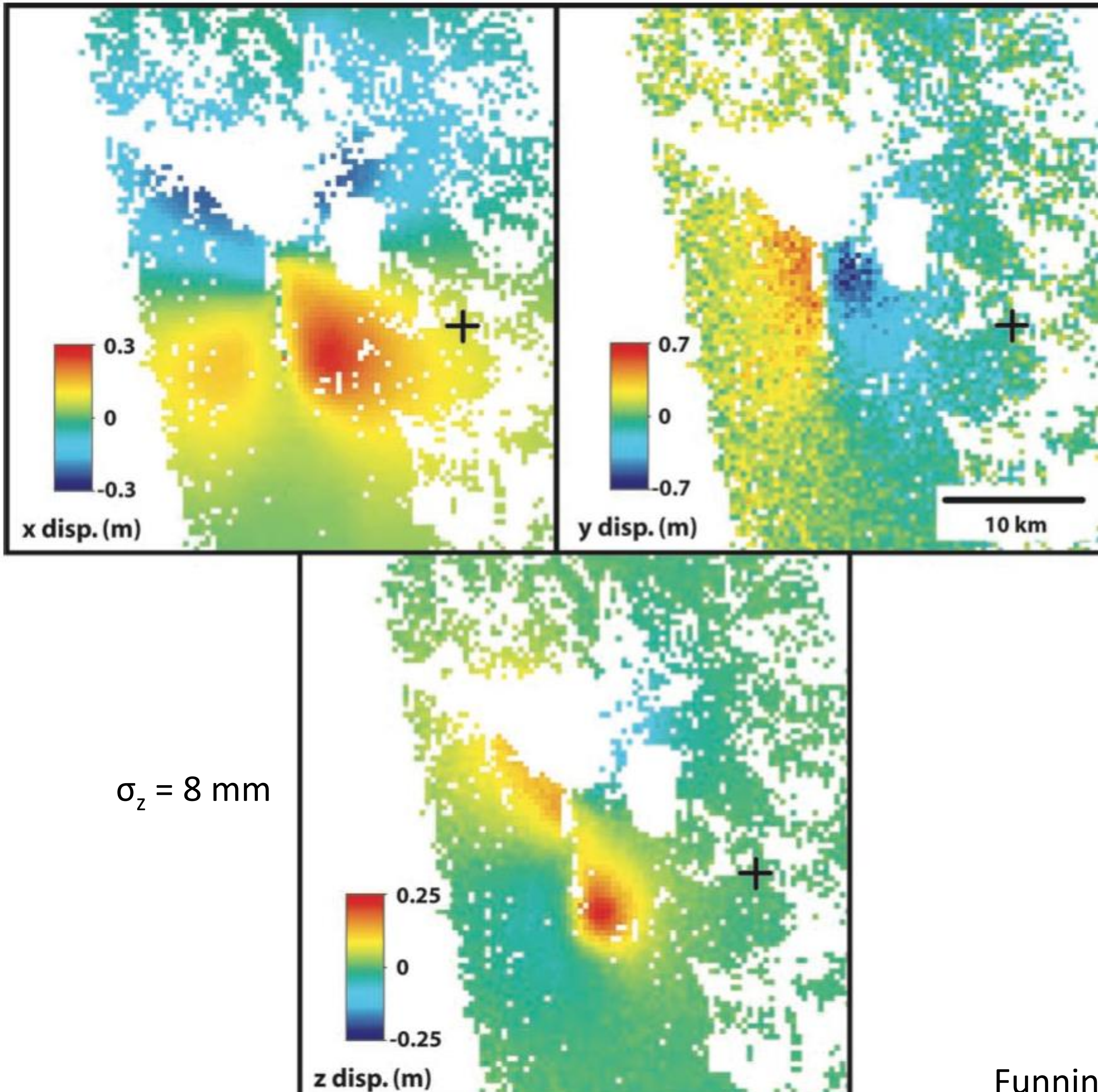
σ = 117 mm



Funning et al., 2005

σ_x = 9 mm

σ_y = 89 mm

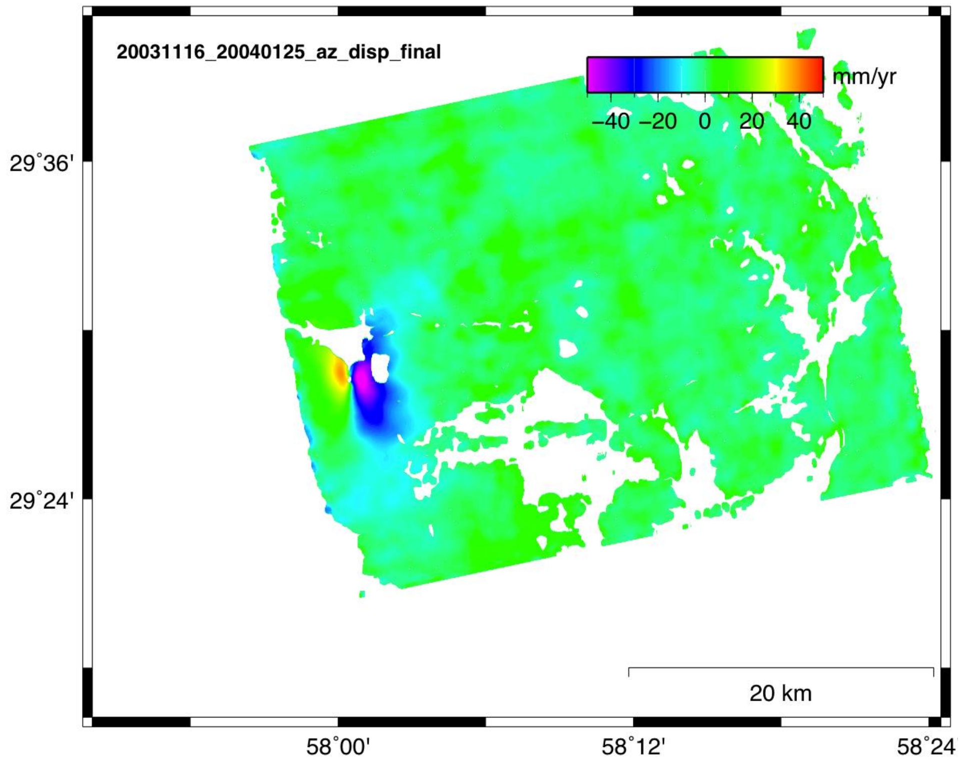


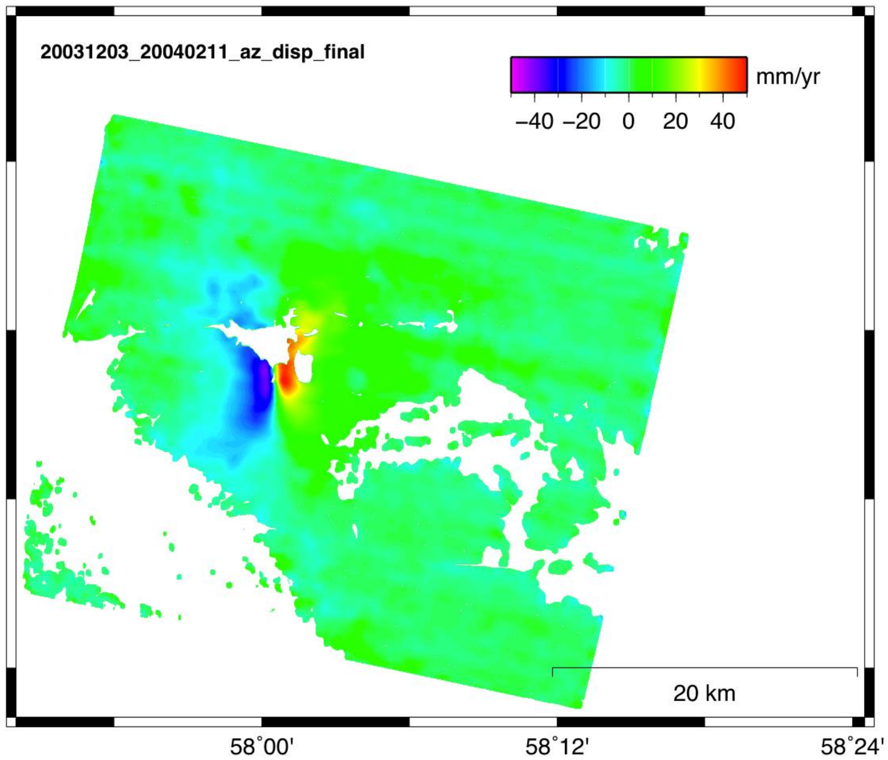
σ_z = 8 mm

Funning et al., 2005

2) Enhanced Spectral Diversity

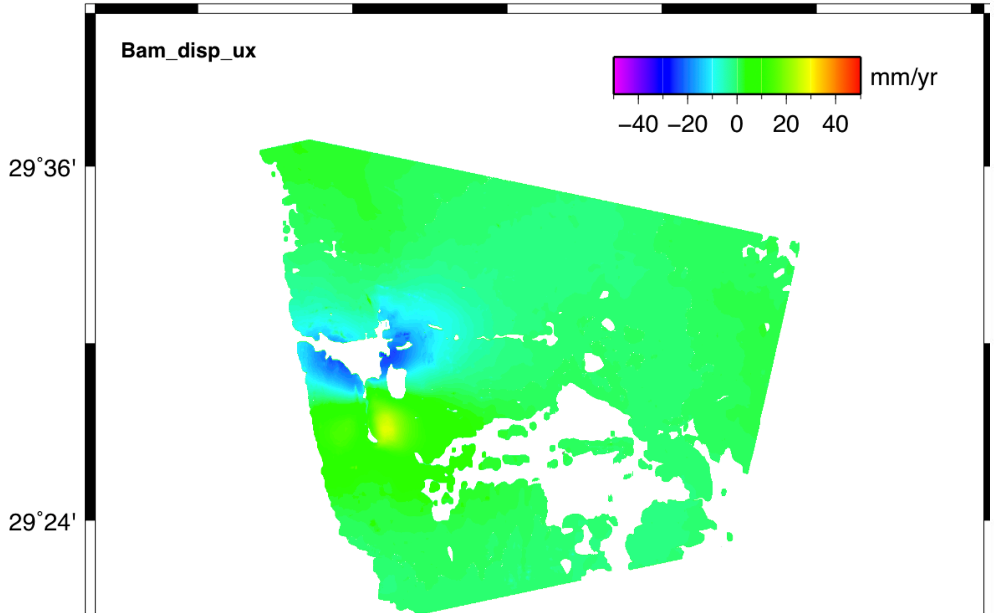
- Split the synthetic aperture for both SAR images into forward- and backward-looking apertures
- Form forward- and backward-looking SLCs, and then interferograms
- The difference of those two interferograms is a measure of the along-track displacement
- Much faster to compute than azimuth offsets
- Lower signal-to-noise and precision than regular InSAR

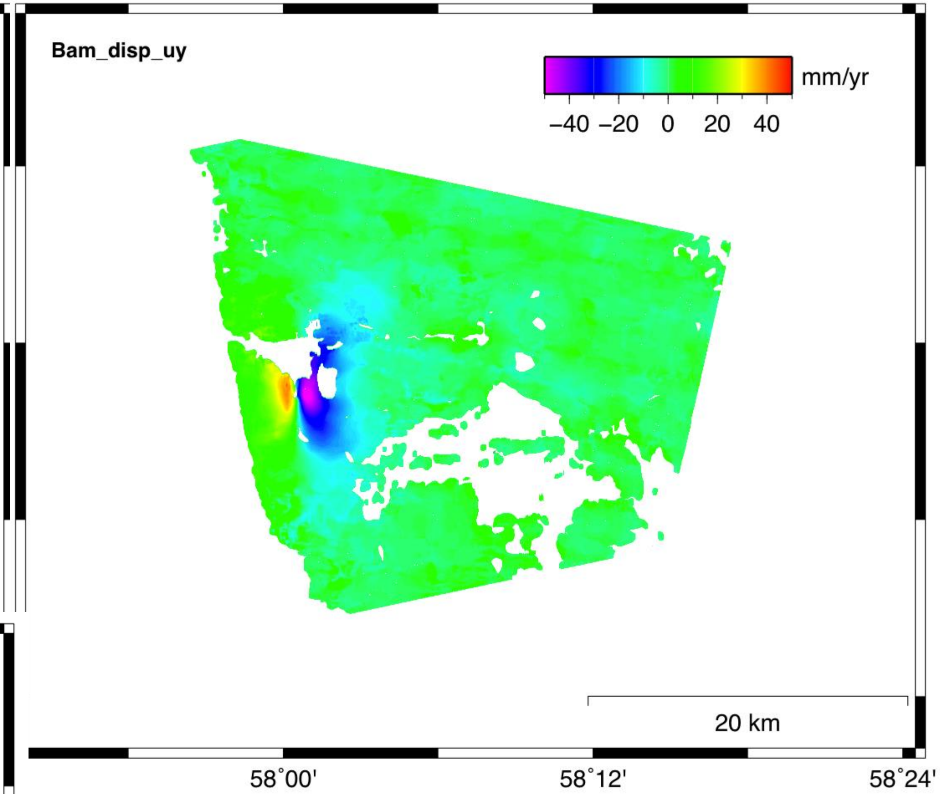


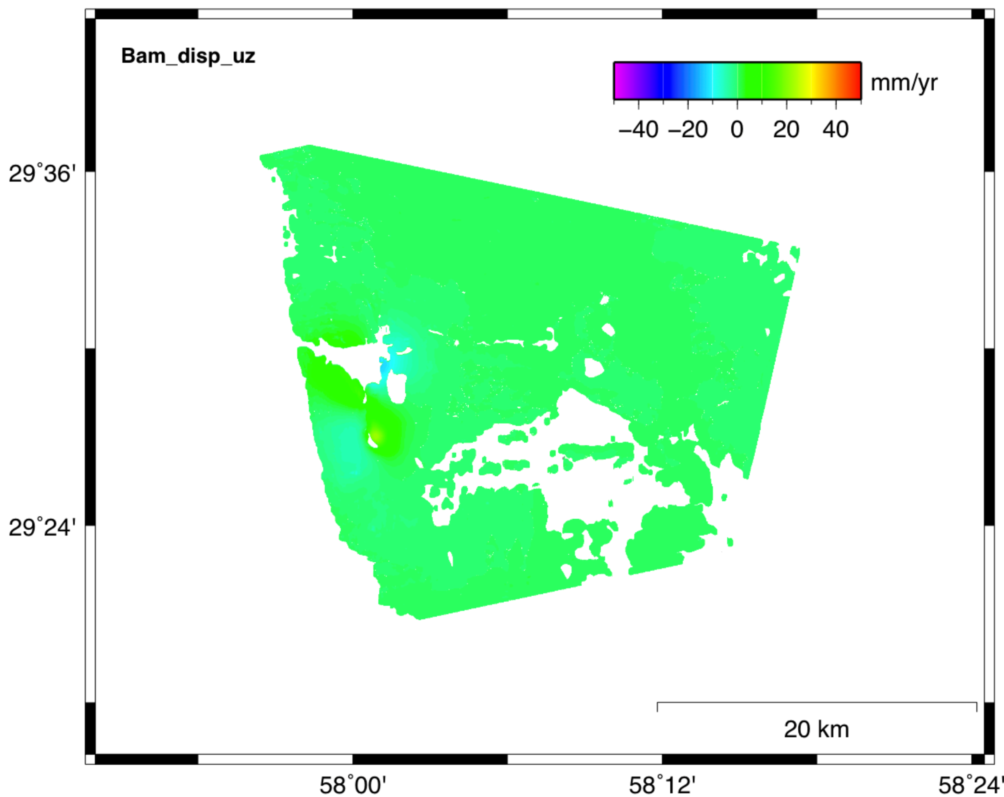


ESD result is much 'cleaner' (less noisy) than the azimuth offsets

Hyung-Sup Jung







Largest improvement is in u_y – this strongly depends on the ESD data

Hyung-Sup Jung

Modeling your InSAR data

Along with a model code that produces surface displacements for your application of interest[1], you will need:

i. A manageable number of data points[2]
ii. Line of sight information for those points

---

[1] if you're modeling fault slip, I can help with this

[2] usually of the order of a few thousand or fewer

Why downsample InSAR data?

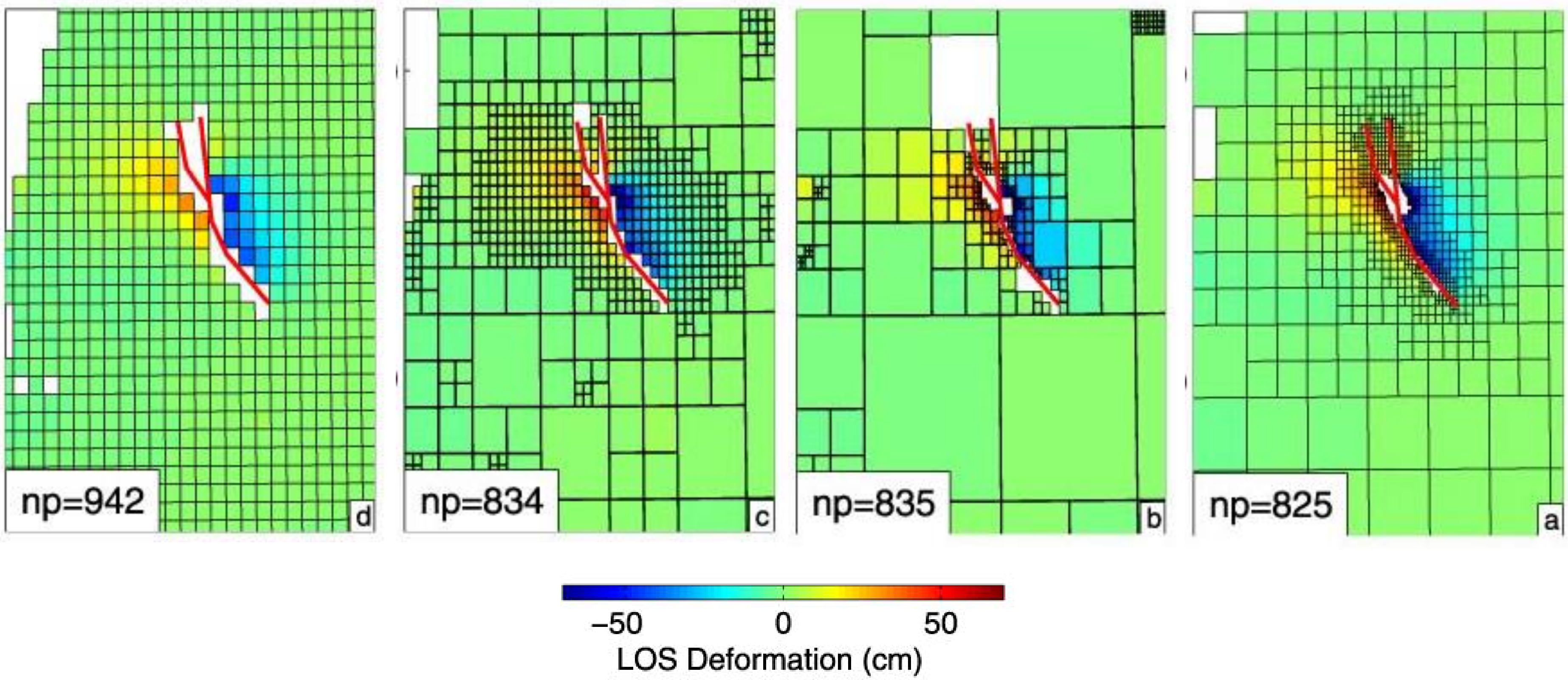
Even when multilooked, an individual interferogram can have a very large number of pixels (e.g. single frame Sentinel-1 TOPS at 90 m resolution has ~4 million pixels!)

For many geophysical applications, the data are highly correlated (i.e. pixels in close proximity have near-identical displacements)

=> You do not need every single pixel to represent the displacement pattern

=> Including many similar (and/or unimportant) pixels, might bias your model towards fitting those

Uniform | Variance quadtree | Curvature quadtree | Resolution-based



Lohman and Simons, 2005

Uniform sampling

Advantages:

- Simple to implement

Disadvantages:

- Can weight the far field (more points) over the near field (fewer points)
- One sample spacing is unlikely to capture details of the near field as well as the high correlation of the far field
- Models can lack detail or contain biases

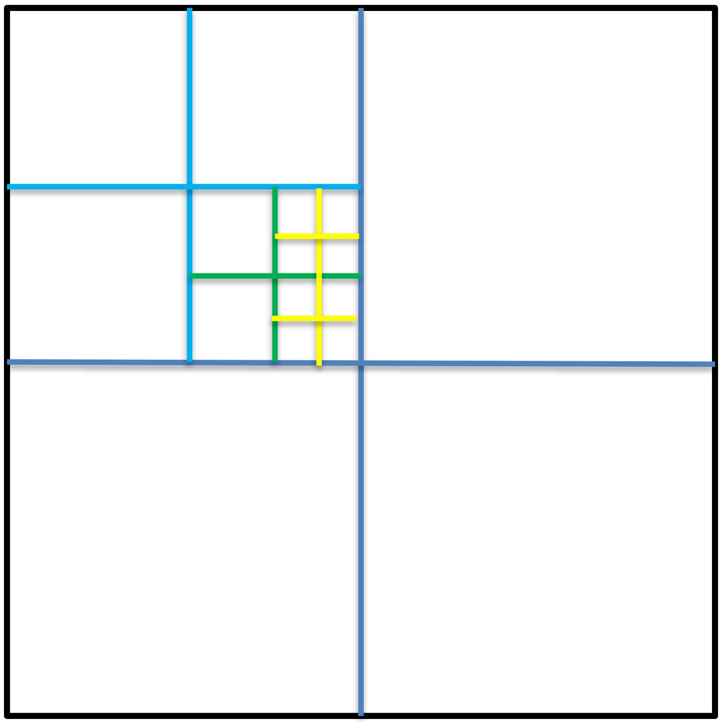
[One remedy: 'zoned' uniform sampling]

Variance quadtree

[e.g. Jonsson et al., 2002]

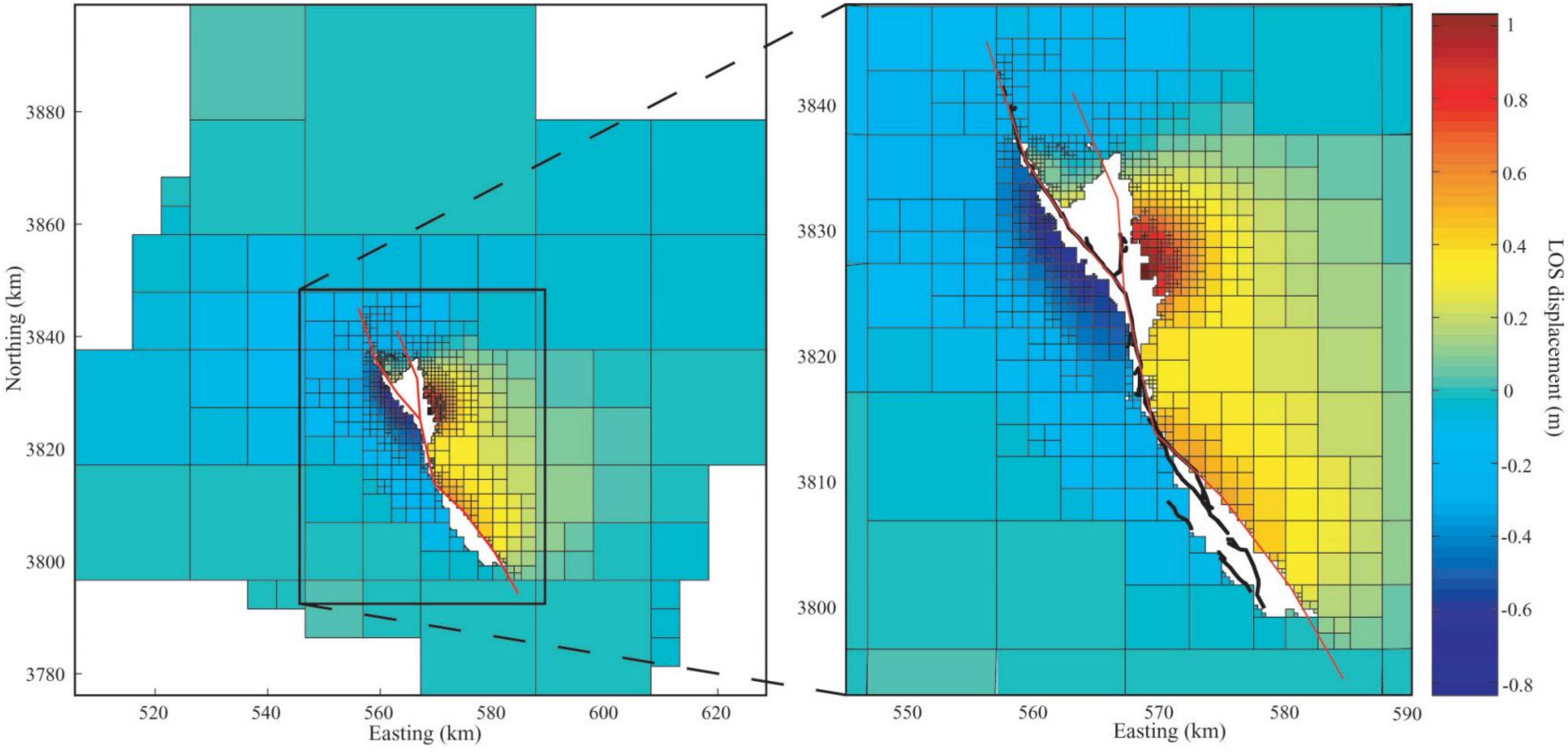
Concept:

1. sample data uniformly (and coarsely)
2. compute variance for each sample 'cell'
3. if variance exceeds threshold, divide cell into 4 portions
4. goto 2



Variance quadtree

[e.g. Jonsson et al., 2002]



Variance quadtree

[e.g. Jonsson et al., 2002]

Advantages:

- Captures the correlations in the data (short wavelength near field, long wavelength far field)

Disadvantages:

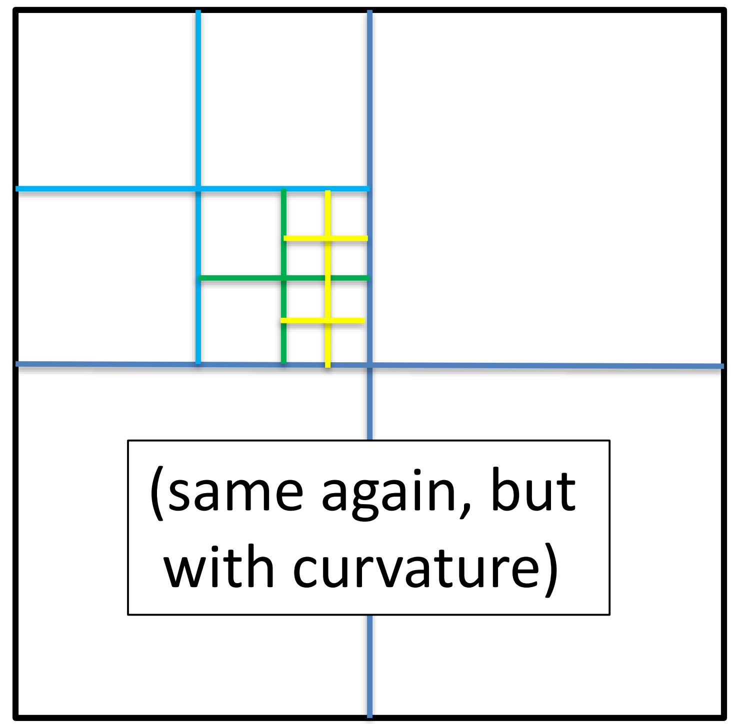
- Can be complicated to implement (although several codes exist and are available)
- Focuses on area of maximum LOS displacement gradient; may not capture features of interest
- Can be 'distracted' by noise in the data

Curvature quadtree

[e.g. Simons et al., 2002]

Concept:

1. sample data uniformly
2. compute curvature for each cell (remove ramp, compute variance)
3. if curvature exceeds threshold, divide cell into 4
4. goto 2



Curvature quadtree

[e.g. Simons et al., 2002]

Advantages:

- Captures the correlations in the data
- Constrains, in particular, edges of features in the data (and therefore likely in the model, too)

Disadvantages:

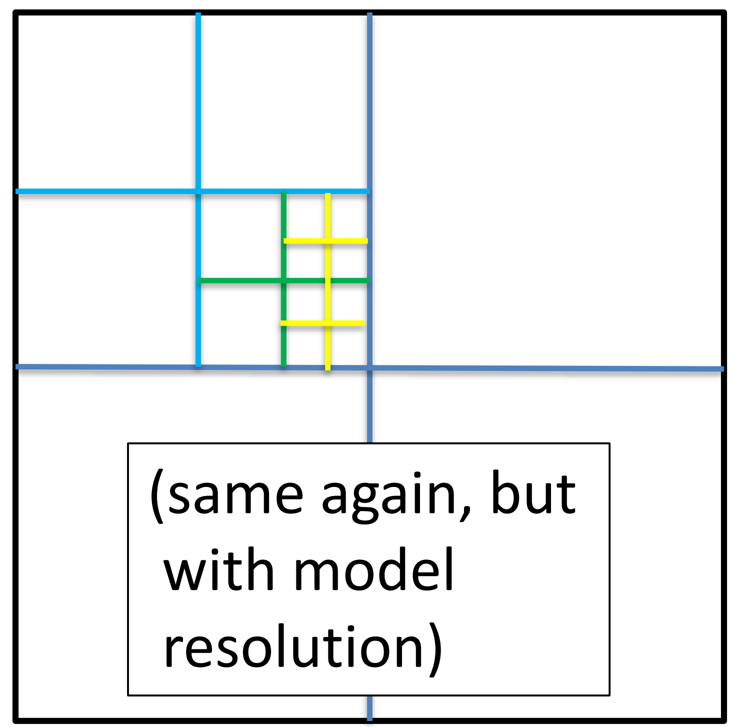
- Implementation (but codes exist, see above)
- Curvature may not capture features of interest
- Can be 'distracted' by noise in the data

Resolution-based sampling

['rosampling'; e.g. Lohman and Simons, 2005]

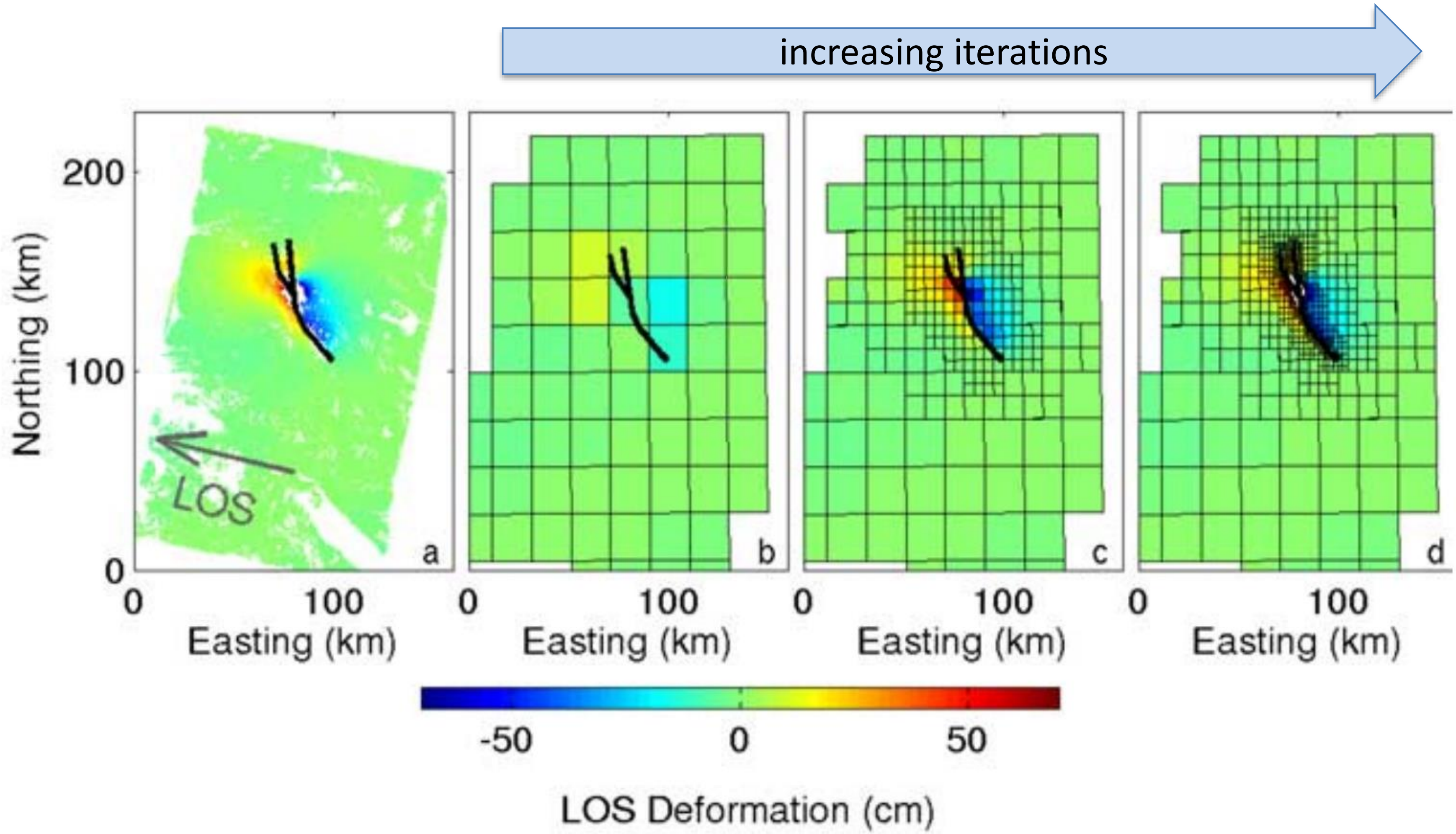
Concept:

1. sample data uniformly
2. compute resolution matrix for model with that data distribution
3. if resolution of a sample cell below threshold, divide into 4
4. goto 2



Resolution-based sampling

['rosampling'; e.g. Lohman and Simons, 2005]



Resolution-based sampling

['rosampling'; e.g. Lohman and Simons, 2005]

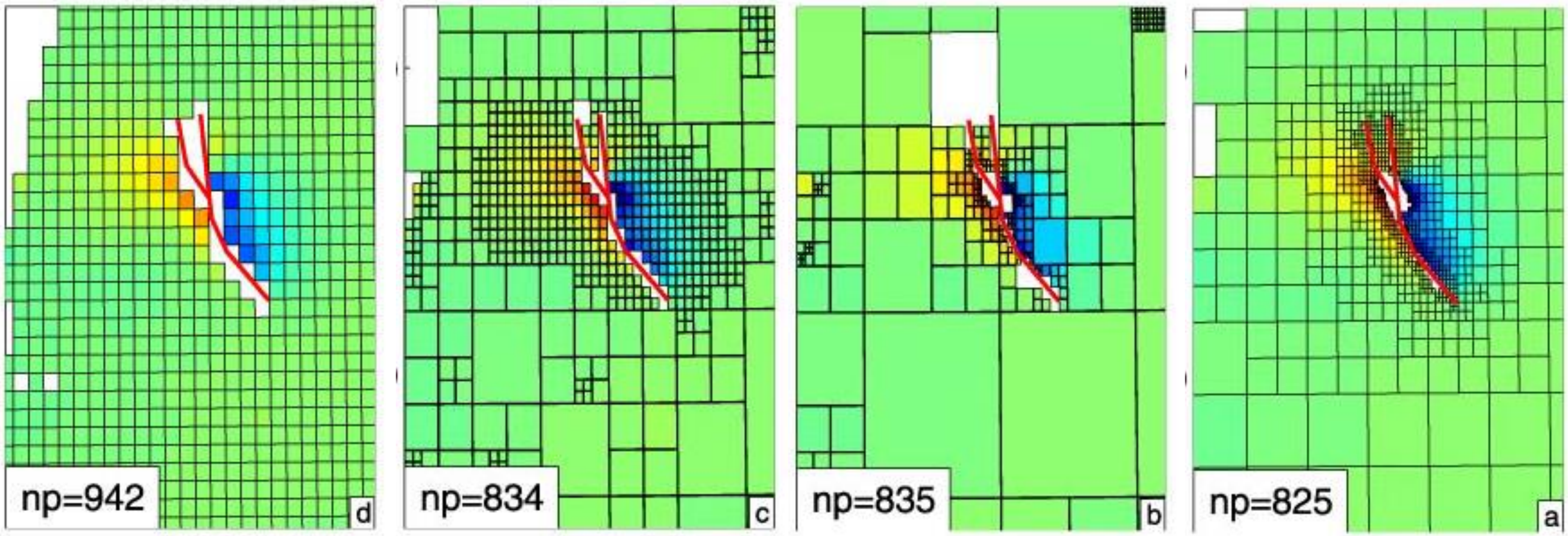
Advantages:

- Chooses locations of data points to optimize their resolving power in the model

Disadvantages:

- Implementation
- Model-based, not data-based (i.e. requires to know the geometry of your model in advance – great if you do, may lead to biases if you don't)

Uniform | Variance quadtree | Curvature quadtree | Resolution-based

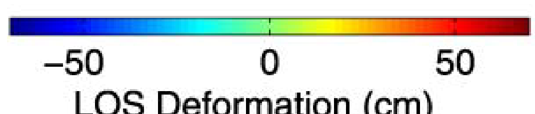


Low resol-ution in near field

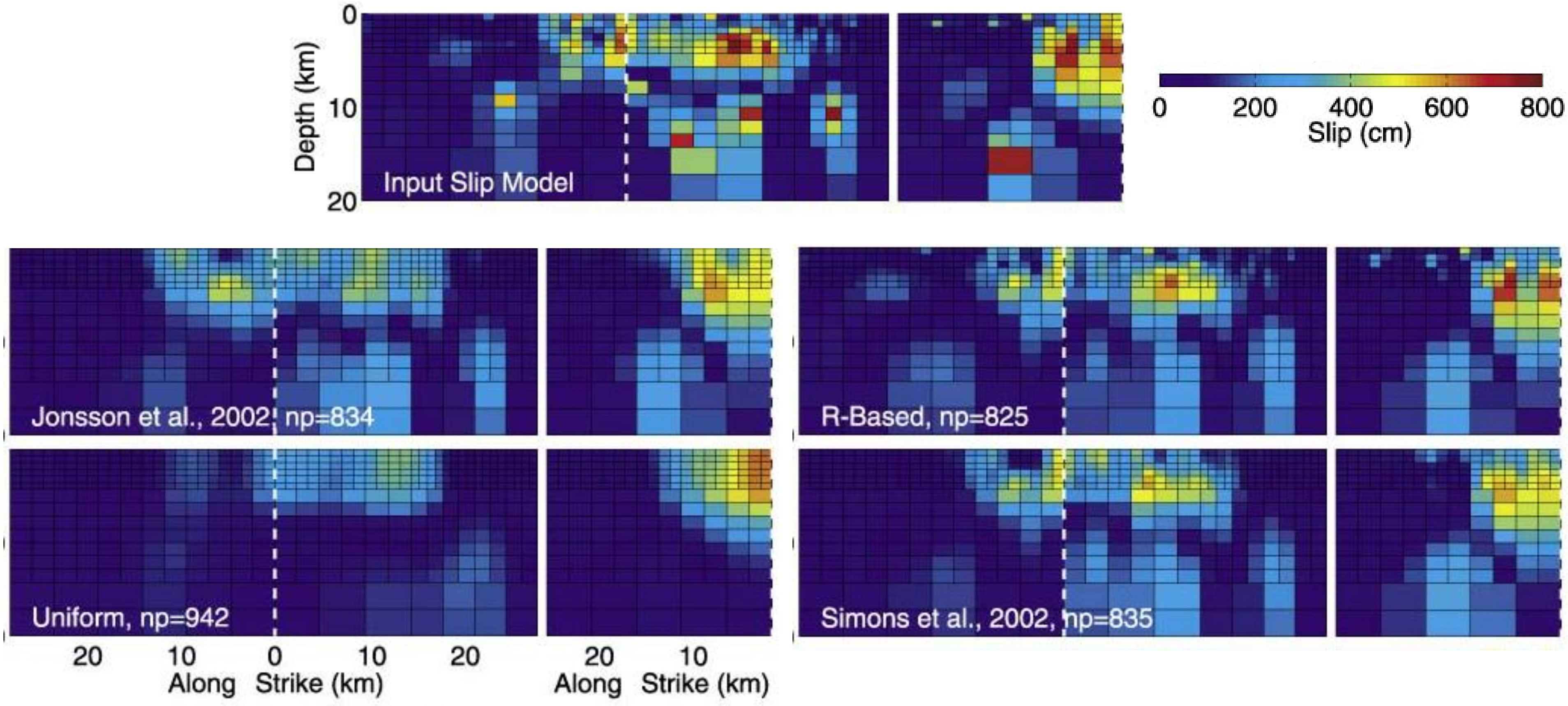
High resolu-tion in near field

Focused on the edges of features

'Zoned' – cells larger with distance



Lohman and Simons, 2005



Model inversion test with synthetic data: resolution-based method does best at reproducing original, then curvature and variance quadtree.

Uniform sampling is worst – shallow: fuzzy, deep: washed out. Don't do it!

Lohman and Simons, 2005

Downsampling tips

- Coarsest sampling scale should be ~the correlation length scale of noise in the data
- If you know your model geometry, resolution-based sampling works well (if not, try curvature quadtree)
- Quadtree methods can have problems with decorrelated (holey) data – try interpolating first, then quadtree, then apply quadtree grid to original data
- Anything is better than uniform sampling! If you must, try 'zoned' sampling – increase density in the near field
- Python and Matlab codes for these methods are out there (e.g. pyrocko/kite)