Multiview Geometry and Bundle Adjustment

CSE P576

David M. Rosen

Recap

Previously:

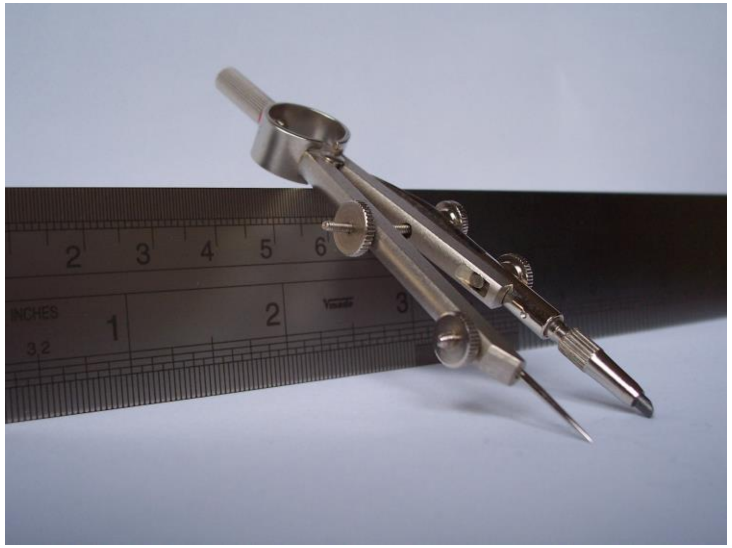
- Image formation
- Feature extraction + matching
- Two-view (epipolar geometry)

Today:

- Add some geometry, statistics, optimization
- Turn it up to ~~11~~ **N!**

Motivating example: Photogrammetry

The science of measurement using cameras



Application: Remote Sensing

Mars Reconnaissance Orbiter

- Launched 12 Aug 2005
- Entered orbit 10 Mar 2006
- ~ 112 minute orbital period
  $\Rightarrow$ ~ 12.8 orbits / (Earth) day

- Sensors:
  - High Resolution Imaging Science Experiment (HiRISE)
  - Context Camera
  - Mars Color Imager

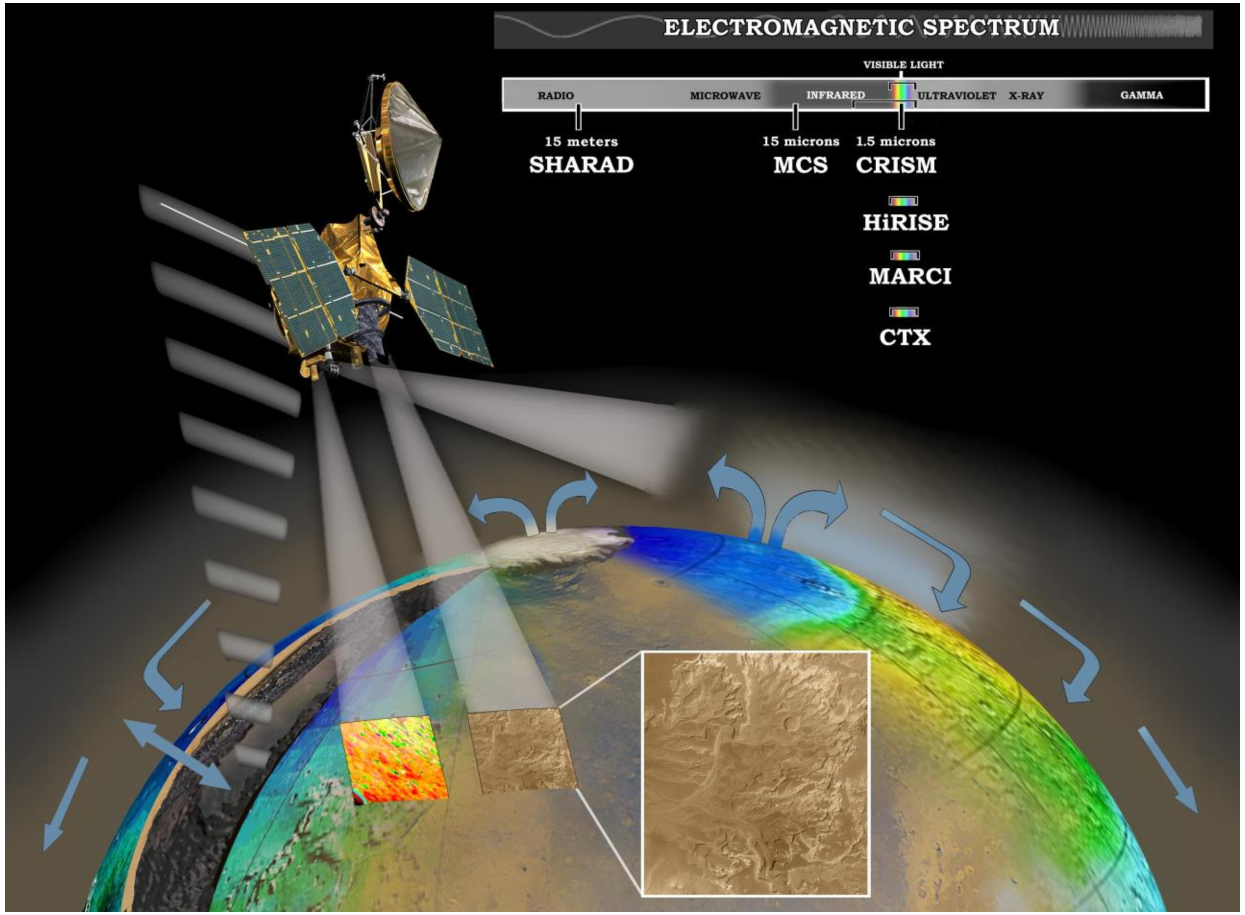


Image credit: NASA/JPL

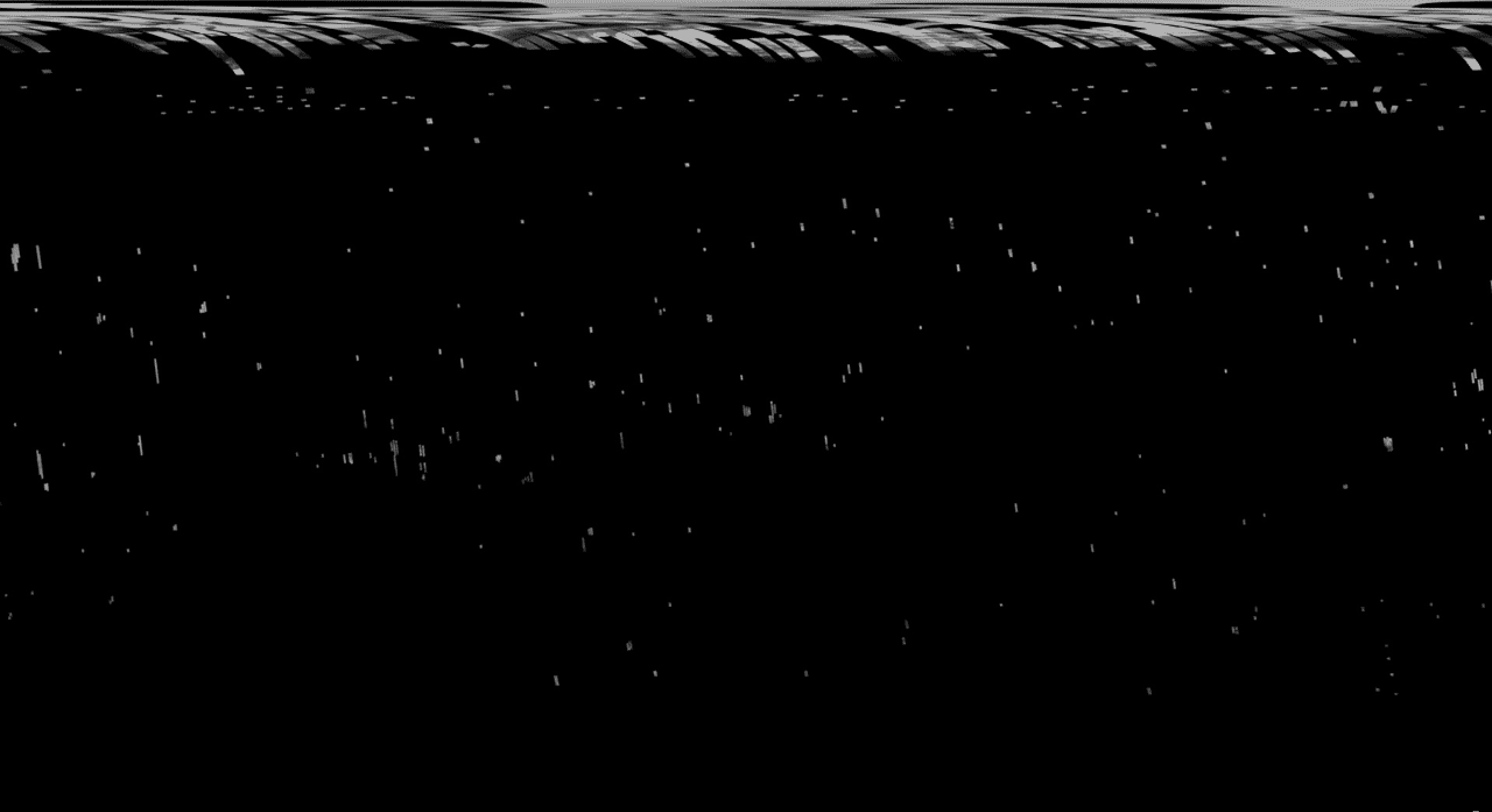


Image credit: NASA/JPL-Caltech/MSSS

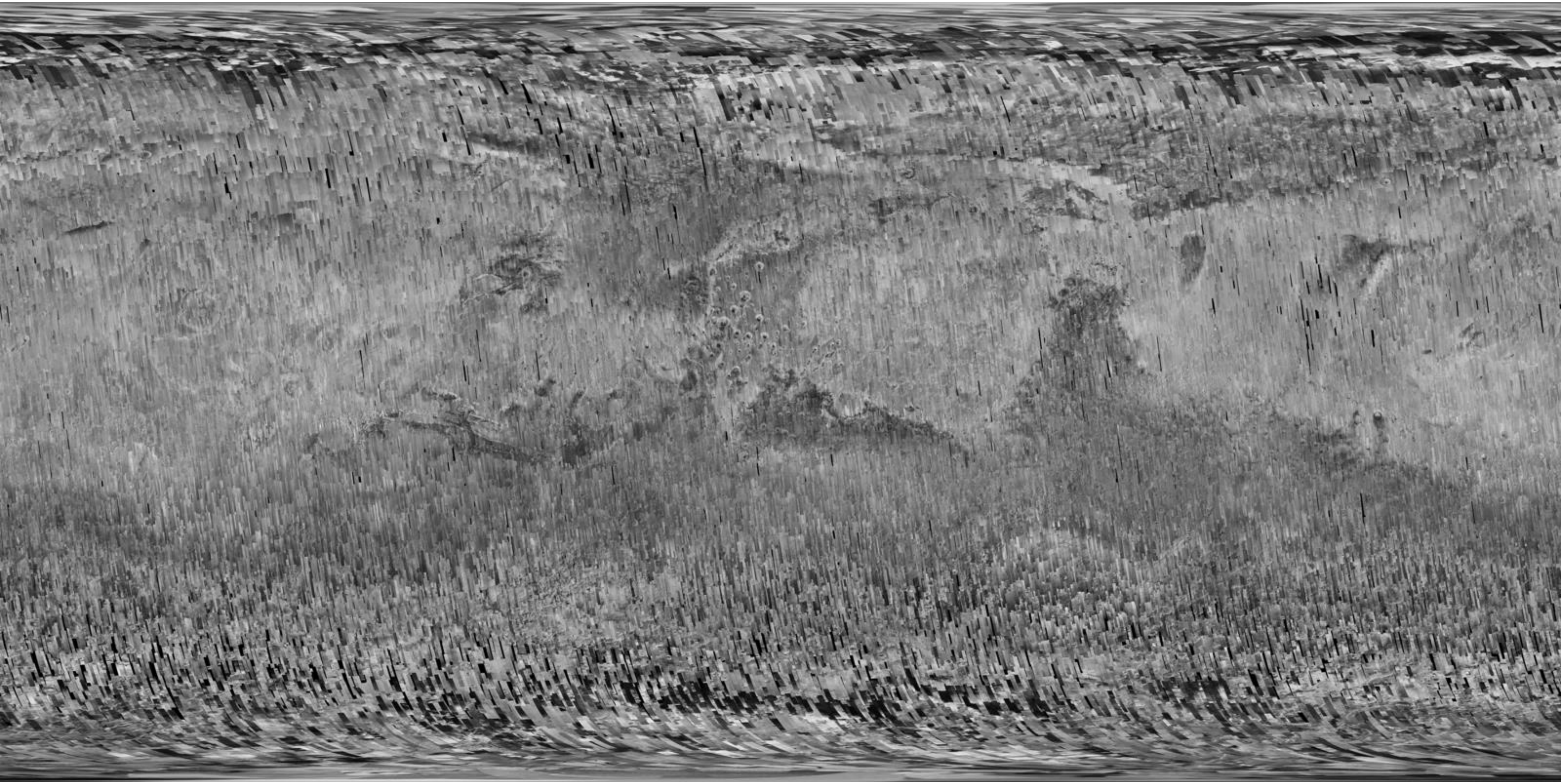


Image credit: NASA/JPL-Caltech/MSSS

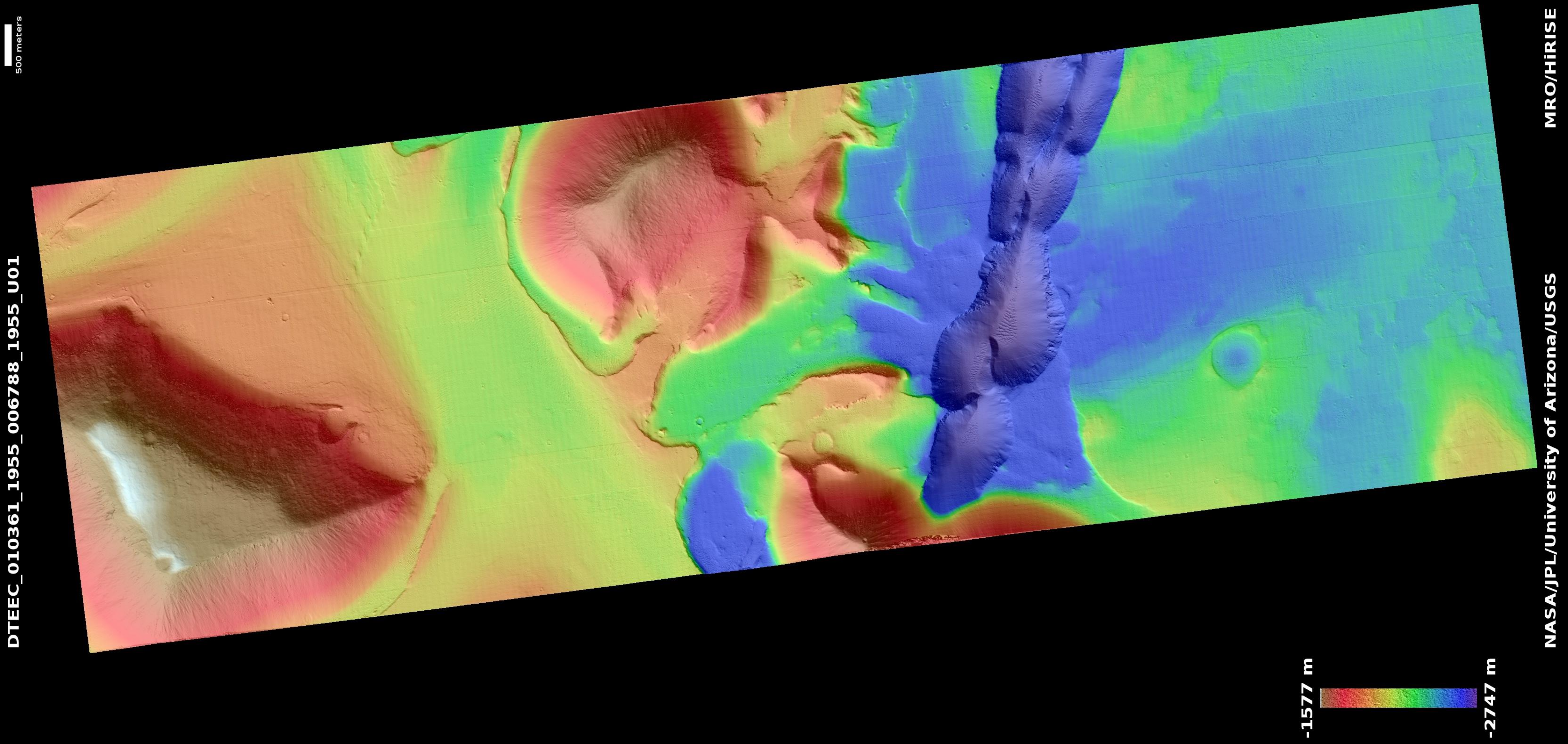


Image credit: NASA/JPL/University of Arizona/USGS

Image credit: NASA / JPL-Caltech / UA / Kevin M. Gill

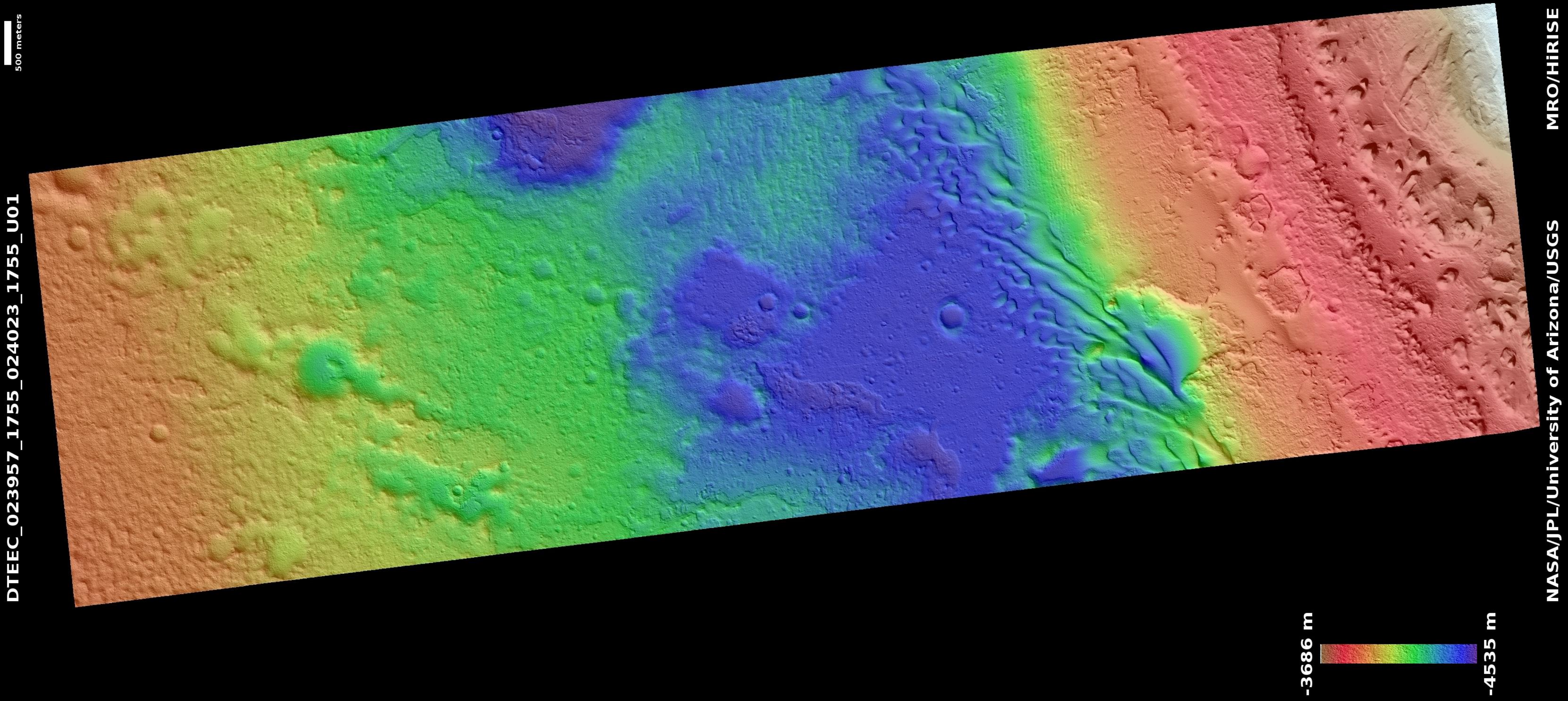
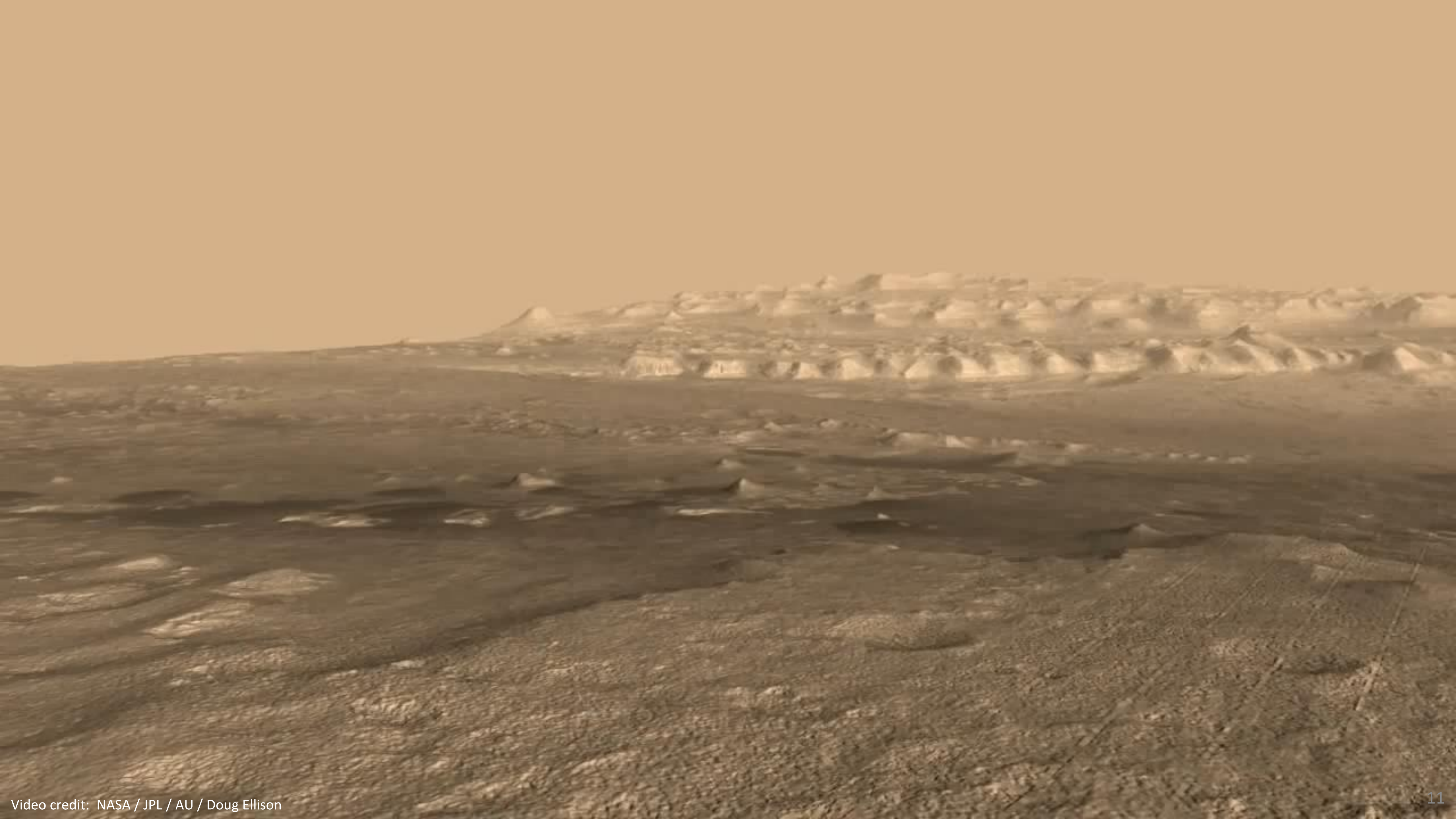


Image credit: NASA/JPL/University of Arizona/USGS

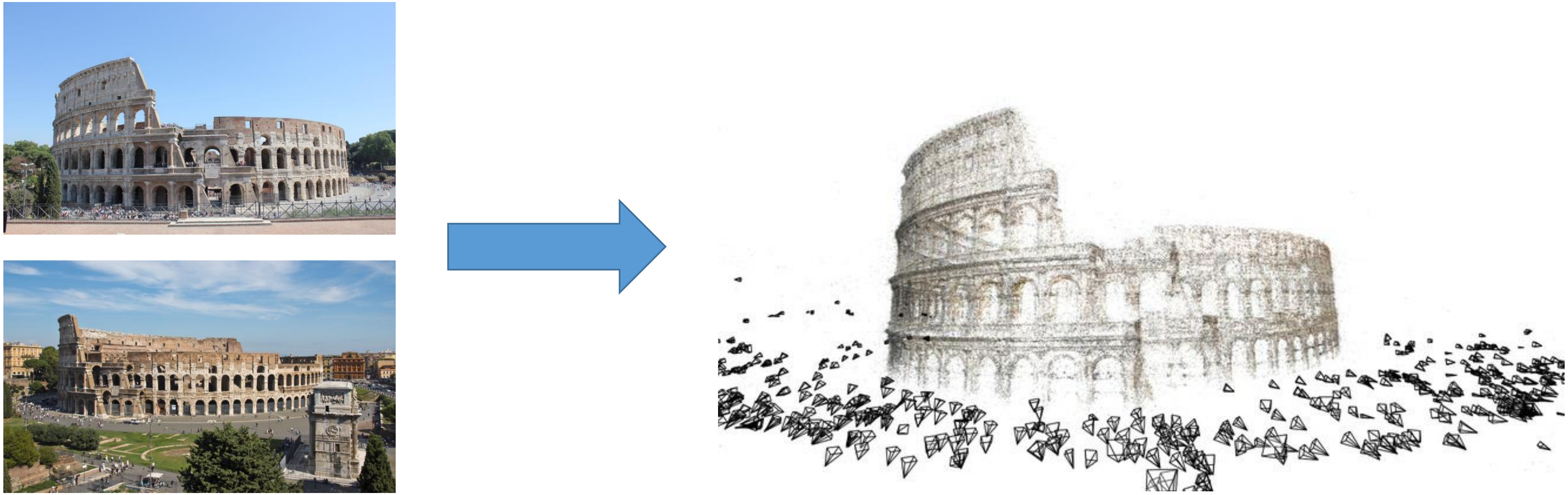
Image credit: NASA / JPL-Caltech / UA / Kevin M. Gill



Video credit: NASA / JPL / AU / Doug Ellison

Application: 3D Reconstruction

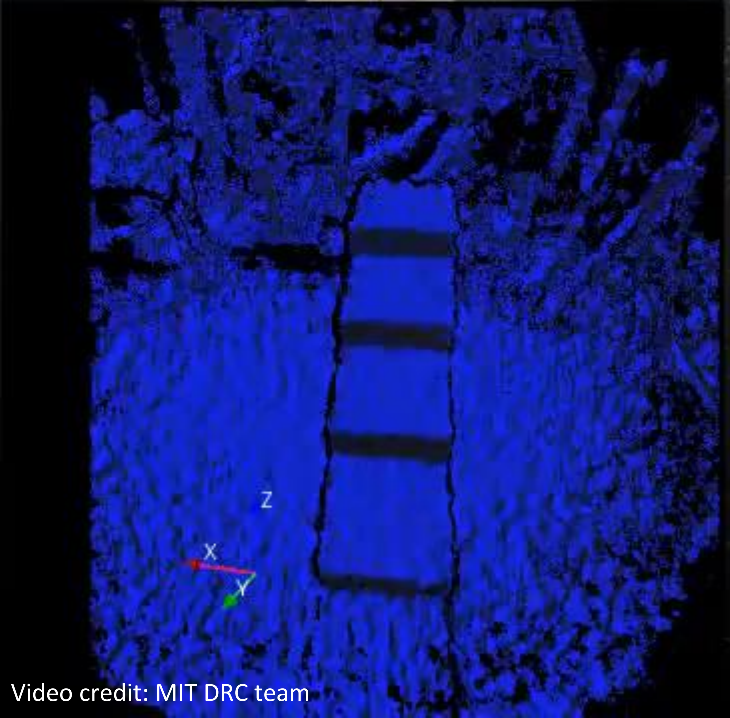
Goal: Build a 3D model of a scene from a collection of images

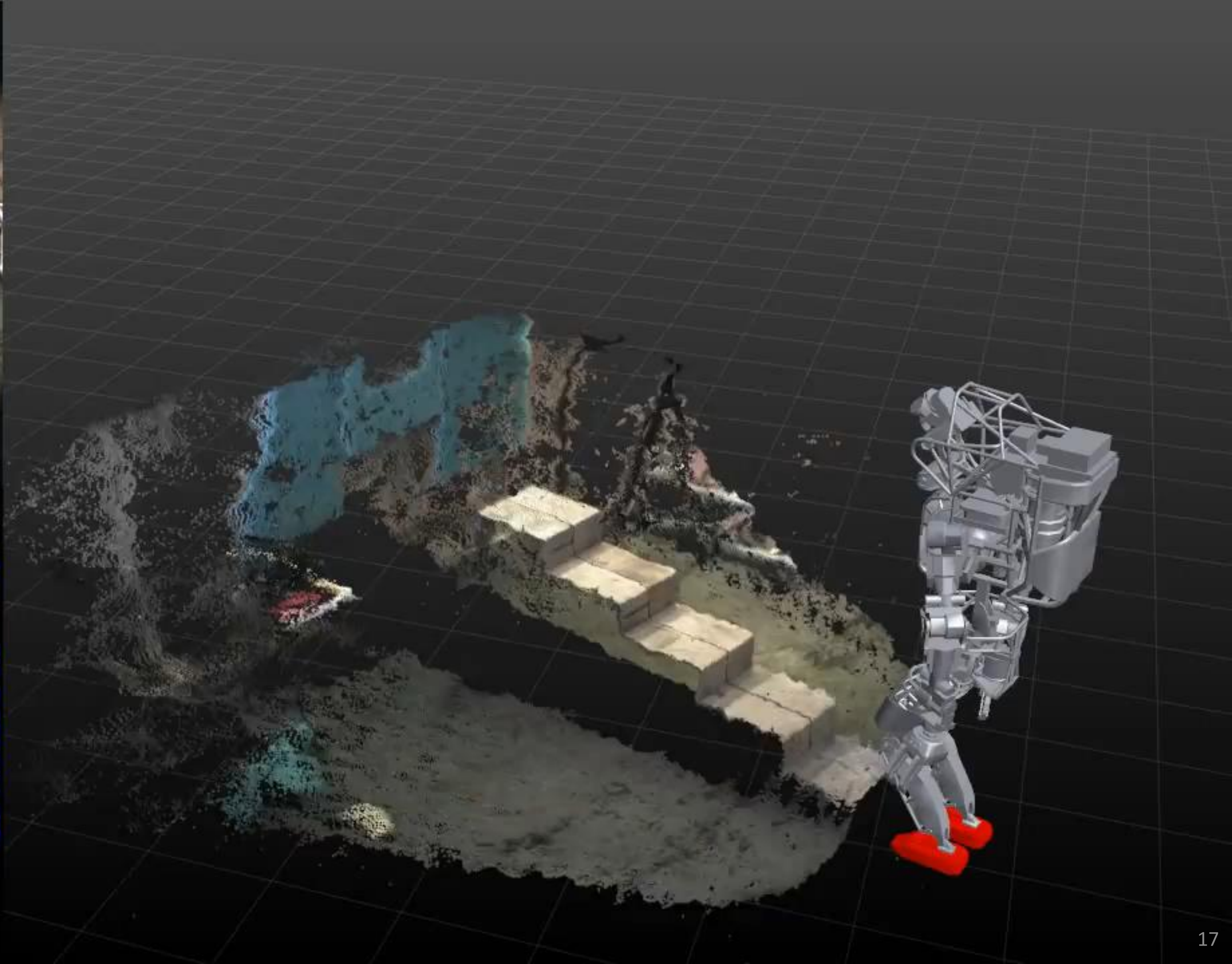


[S. Agarwal et al., "Building Rome in a Day", Communications of the ACM, 2011]

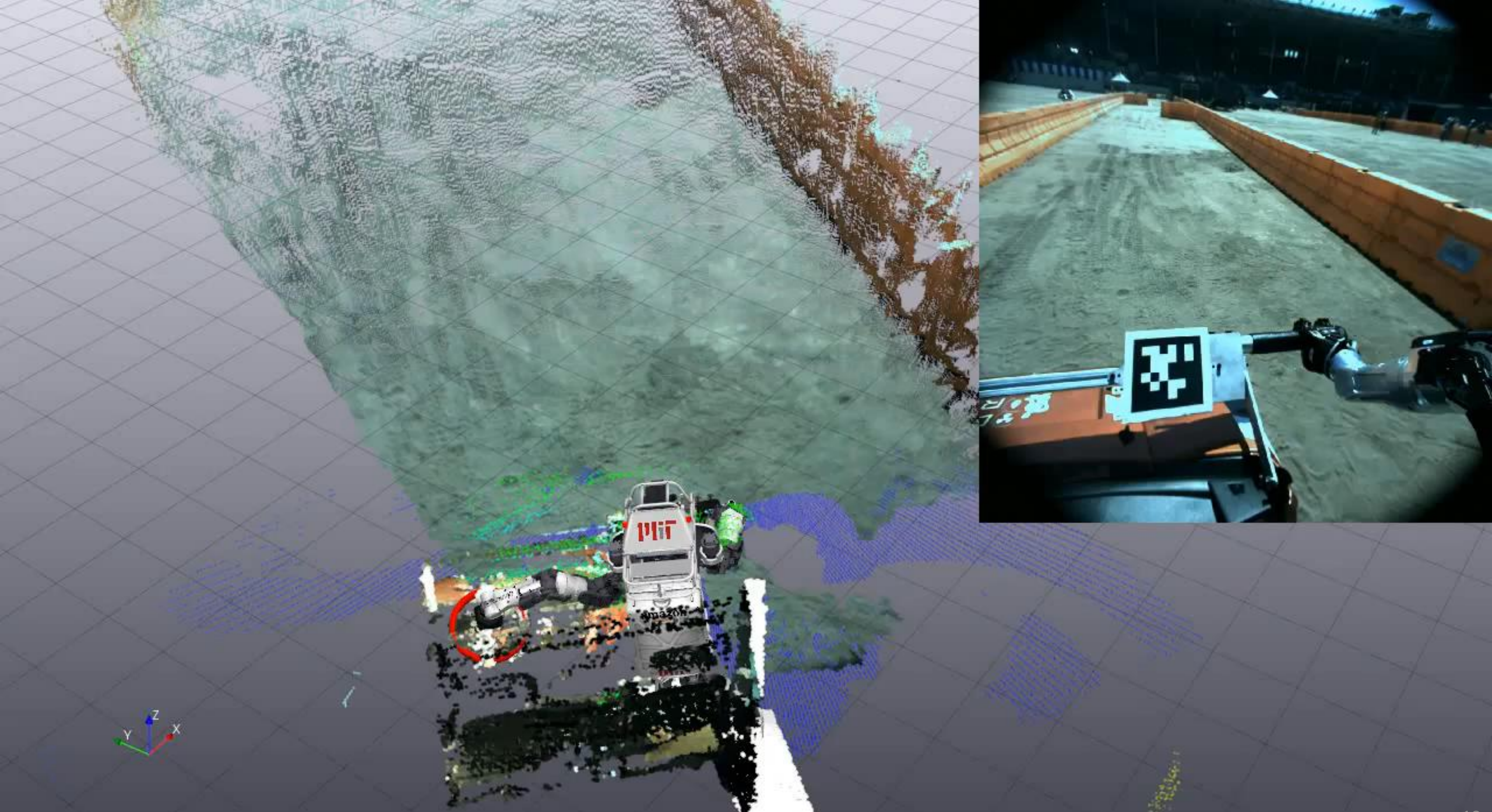
Application: Robotics

Video credit: MIT DRC team





Video credit: MIT DRC team



Video credit: MIT DRC team

ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras

Raúl Mur-Artal and Juan D. Tardós

raulmur@unizar.es tardos@unizar.es

In this lecture

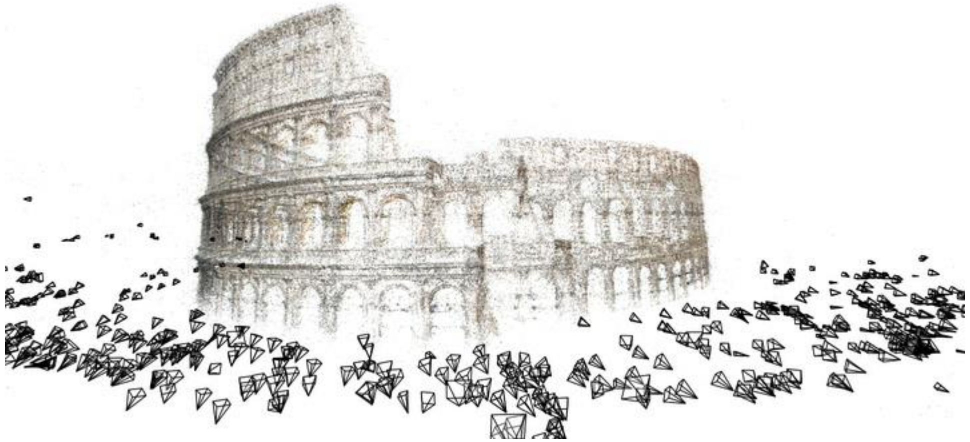
- **Photogrammetry:** The problem of measurement using imagery

- **Maximum-likelihood estimation** and **bundle adjustment:** Solving the photogrammetry problem

- **Practicalities**
  - Problem scale
  - Robust estimation
  - Representation of rotations

Photogrammetry: The Problem

- **Given:** A collection of images
- **Estimate:**
  - 3D positions of imaged points
  - Poses of the imaging cameras
  - Intrinsic parameters of the imaging cameras

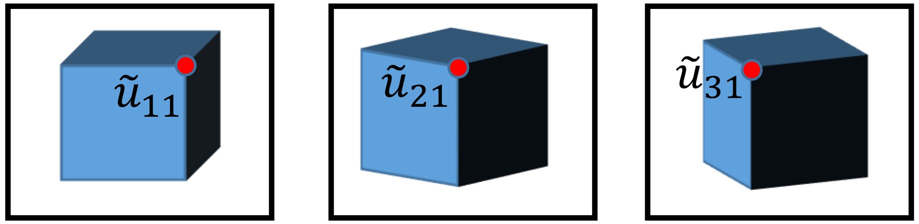


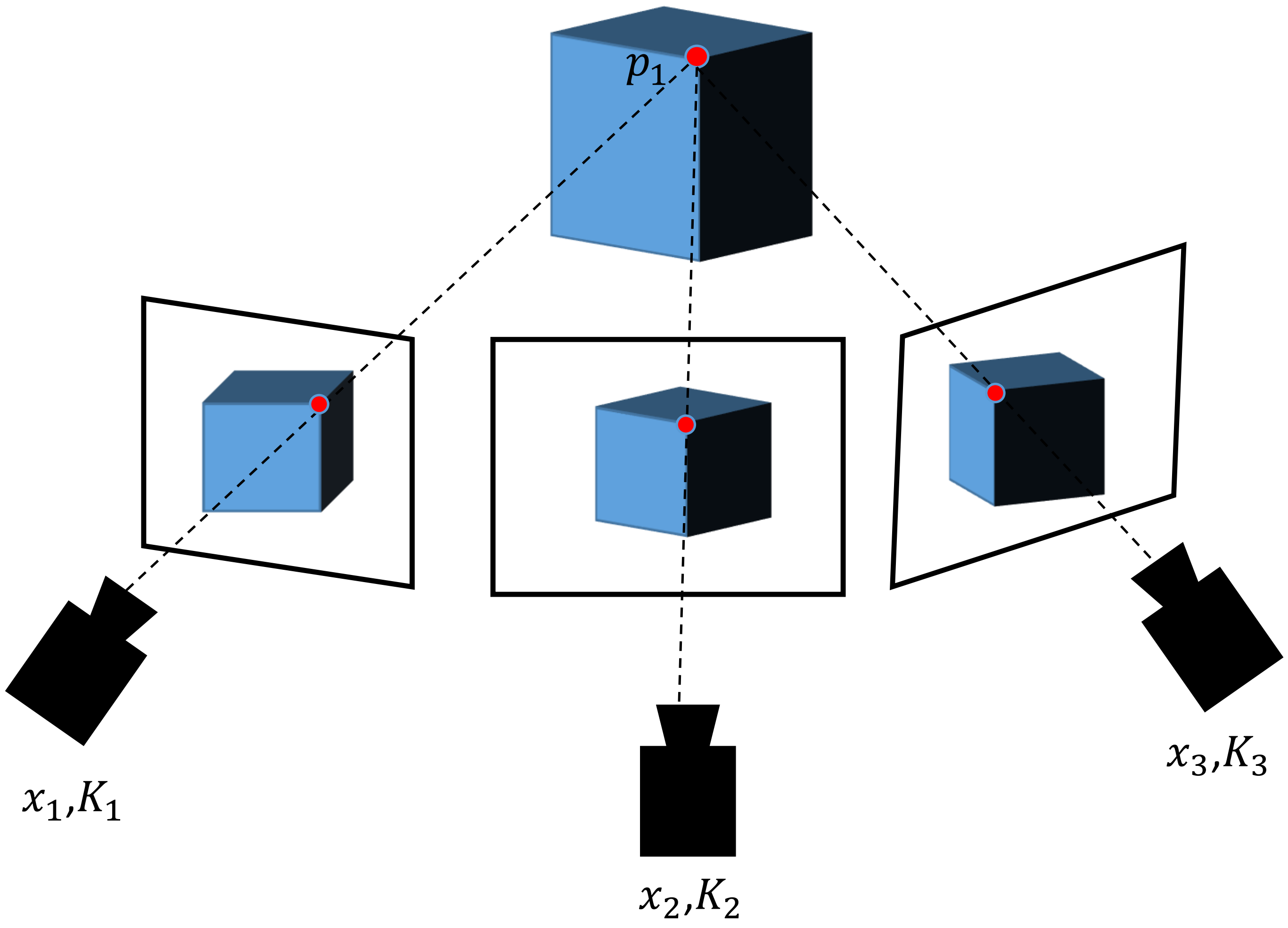
Photogrammetry: Generative model

Q: How are ***variables of estimation***:

- 3D point positions p_j
- Camera poses $x_i = (t_i, R_i)$
- Camera intrinsics K_i

related to ***images***?



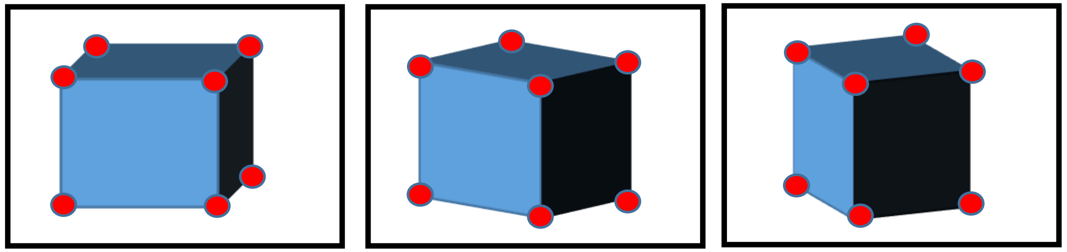


$$\tilde{u}_{ij} = f(x_i, K_i, p_j)$$

where f is the *camera projection function* (from Lecture 1)

Photogrammetry: Estimation procedure

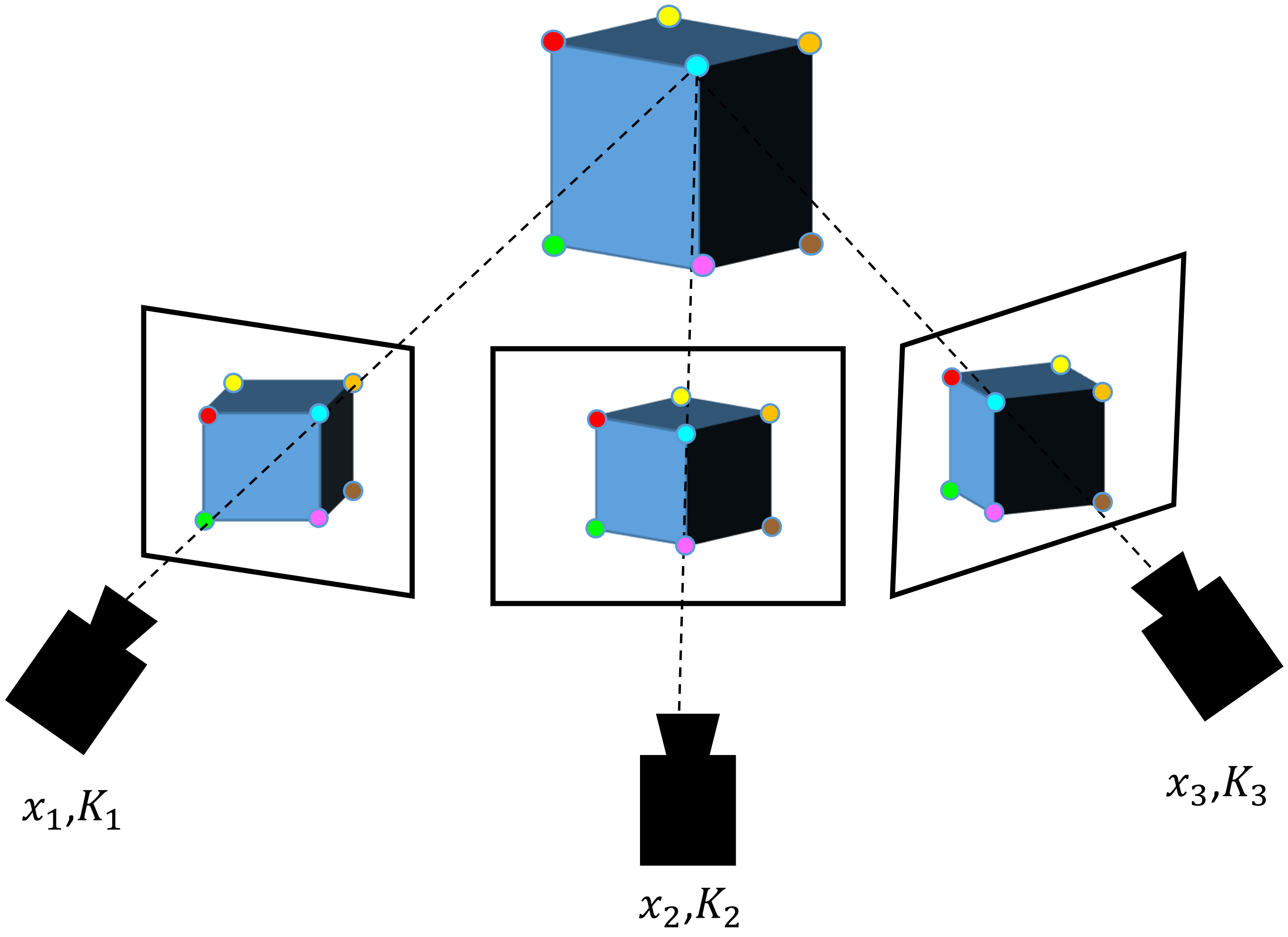
Main idea: Given a set of images



1. Extract features
2. Match features
   (identify the set of 3D points)
3. Estimate parameters so that:

$$\tilde{u}_{ij} = f(x_i, K_i, p_j)$$

for all point projections $\tilde{u}_{ij}$



The problem of measurement noise

We want to find x_i, K_i, p_j so that

$$\tilde{u}_{ij} = f(x_i, K_i, p_j)$$

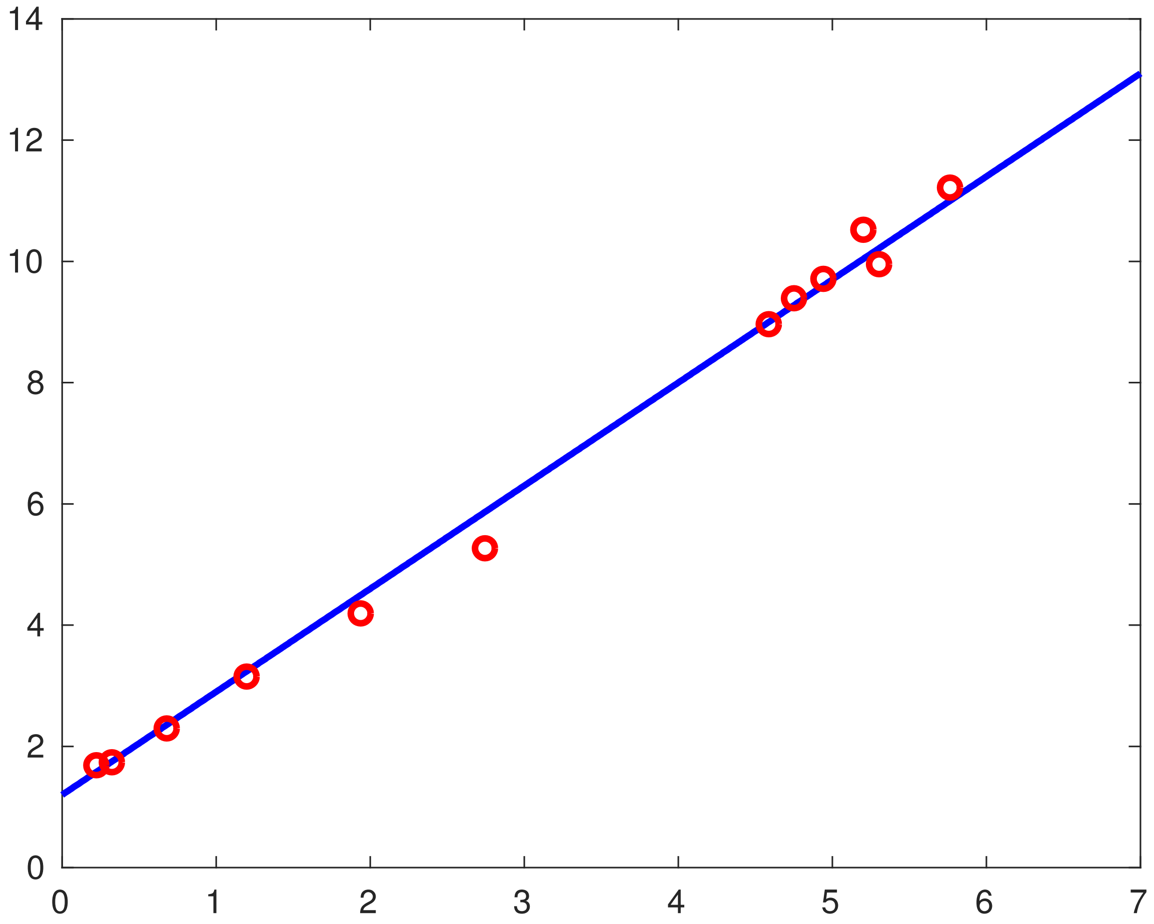
(i.e. our model matches the data) for all $\tilde{u}_{ij}$.

But: All real-world measurements have *errors*

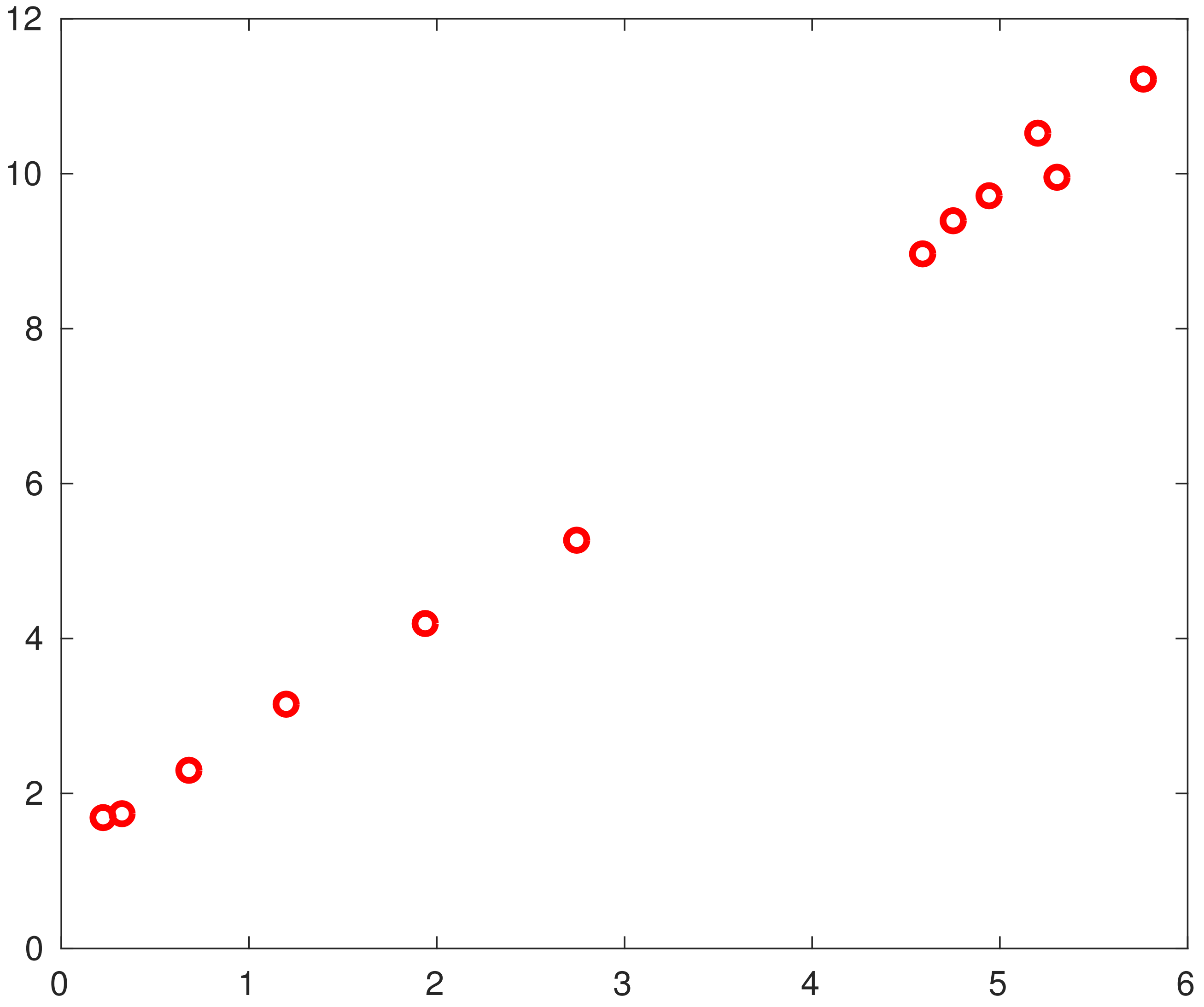
$\Rightarrow$ What we ***actually*** measure is:

$$\tilde{u}_{ij} = f\left(x_i, K_i, p_j\right) + \varepsilon_{ij}$$

$\Rightarrow$ We cannot find parameters x_i, K_i, p_j that fit the measured projections $\tilde{u}_{ij}$ *exactly* …



Example: Linear regression



Maximum likelihood estimation

MLE is a method for *fitting parameters* θ to *noisy data* $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_n)$, given a *sampling model* $y \sim p(\cdot \mid \theta)$.

Basic idea: Choose the θ that *best fits* the data $\tilde{y}$.

But: How can we *measure* goodness of fit?

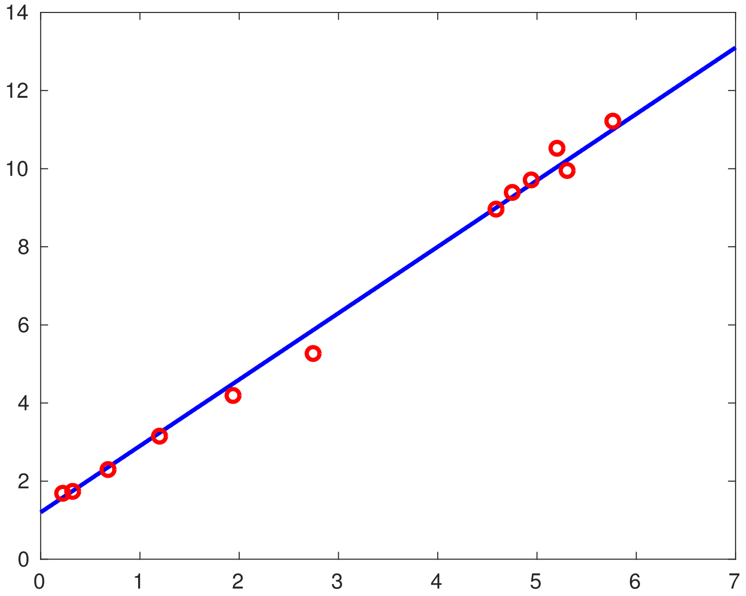
Likelihood function: $L(\theta) \triangleq p(\tilde{y}|\theta)$.

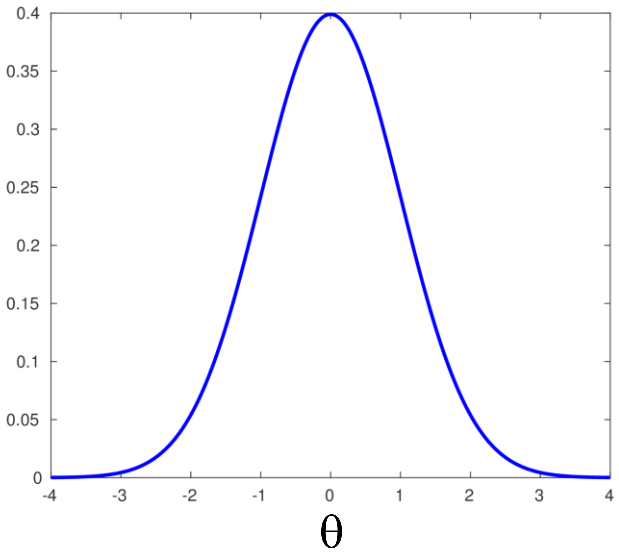
Measures *how likely* the data $\tilde{y}$ is for each choice of θ.

MLE principle: "Best fit" ⇔ "Maximum likelihood"

⇒ Pick the θ that *maximizes the likelihood* of data $\tilde{y}$:

$$\hat{\theta} = \max_{\theta} L(\theta)$$

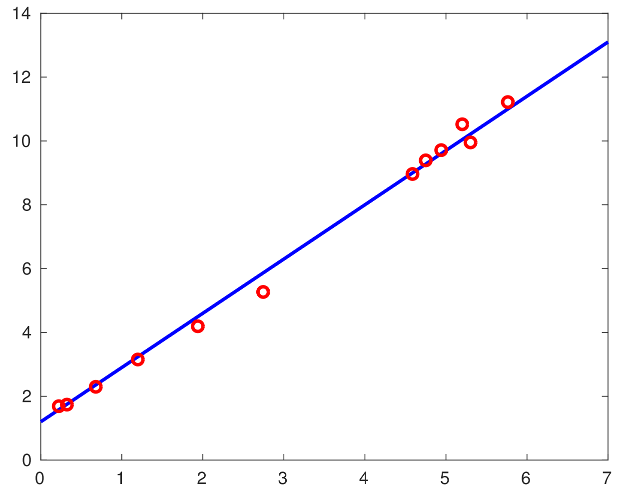




Example: Regression under Gaussian noise

Consider fitting a function $f(\cdot\,;\theta)$ to data $D = \{(x_i, y_i)\}_{i=1}^{N}$ under the model:

$$y_i = f(x_i;\ \theta) + \varepsilon_i, \qquad \varepsilon_i \sim N(0, \Sigma_i)$$

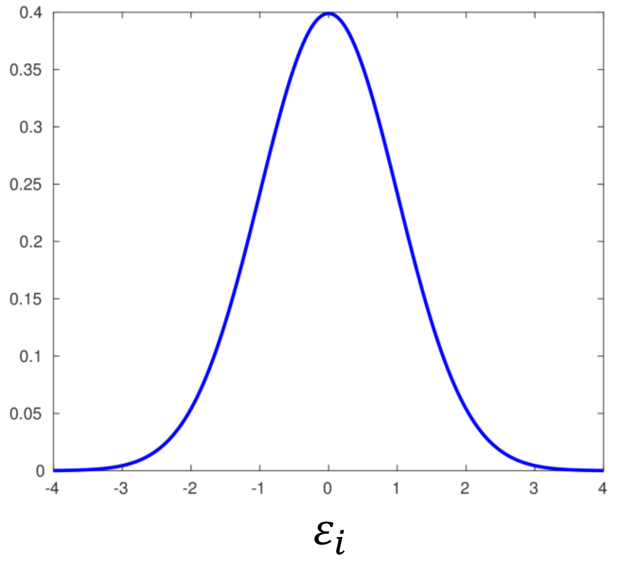


For each choice of θ, for each (x_i, y_i):

$$\varepsilon_i = y_i - f(x_i; \theta)$$

The pdf for ε_i is:

$$p(\varepsilon_i) = \det(2\pi\Sigma_i)^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}\varepsilon_i^T \Sigma_i^{-1} \varepsilon_i\right)$$



⇒The likelihood of the *i*th data point (x_i, y_i) is:

$$p(x_i, y_i|\theta) = \det(2\pi\Sigma_i)^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(y_i - f(x_i;\theta))^T \Sigma_i^{-1} (y_i - f(x_i;\theta))\right)$$

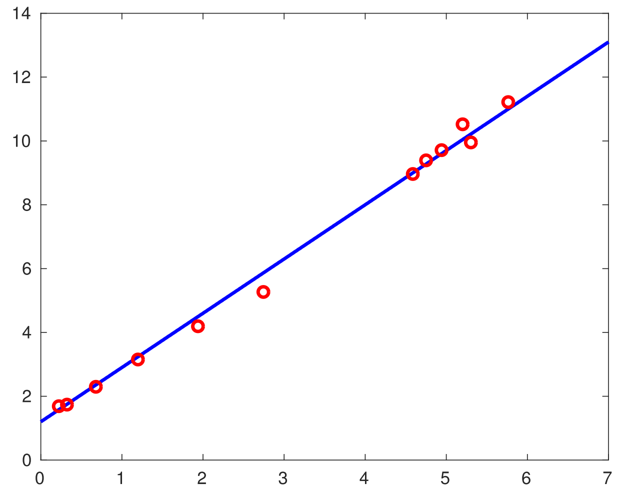
⇒ The likelihood for the *entire* dataset *D* is:

$$L(D|\theta) \propto \prod_{i=1}^{N} \exp\left(-\tfrac{1}{2}(y_i - f(x_i;\theta))^T \Sigma_i^{-1} (y_i - f(x_i;\theta))\right)$$

Example: Regression under Gaussian noise

Consider fitting a function $f(\cdot\,; \theta)$ to data $D = \{(x_i, y_i)\}_{i=1}^{N}$ under the model:

$$y_i = f(x_i;\ \theta) + \varepsilon_i, \qquad \varepsilon_i \sim N(0, \Sigma_i)$$

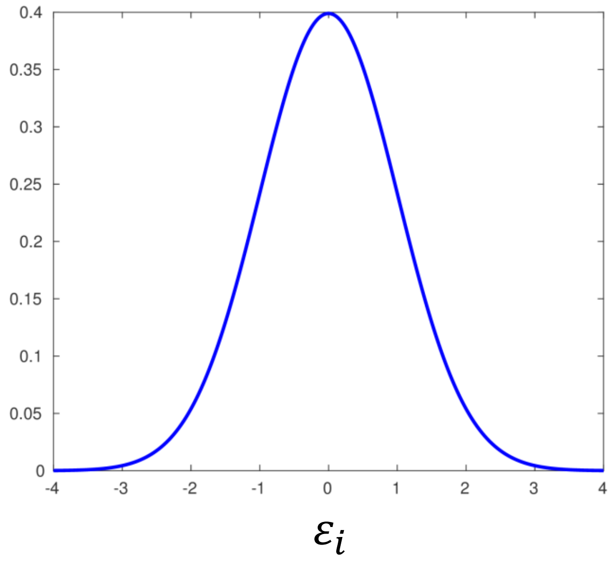


$\Rightarrow$The likelihood for the *entire* dataset *D* is:

$$L(D|\theta) \propto \prod_{i=1}^{N} \exp\left(-\tfrac{1}{2}(y_i - f(x_i;\theta))^T \Sigma_i^{-1} (y_i - f(x_i;\theta))\right)$$

Taking the logarithm:

$$\log L(D|\theta) = c - \frac{1}{2}\sum_{i=1}^{N} \|y_i - f(x_i;\theta)\|_{\Sigma_i}^2$$



$\Rightarrow$ MLE under additive Gaussian noise is a *nonlinear least-squares problem*:

$$\hat{\theta} = \min_{\theta} \sum_{i=1}^{N} \|y_i - f(x_i;\theta)\|_{\Sigma_i}^2$$

Exercise: Linear regression

Consider fitting a *linear* function to the data:

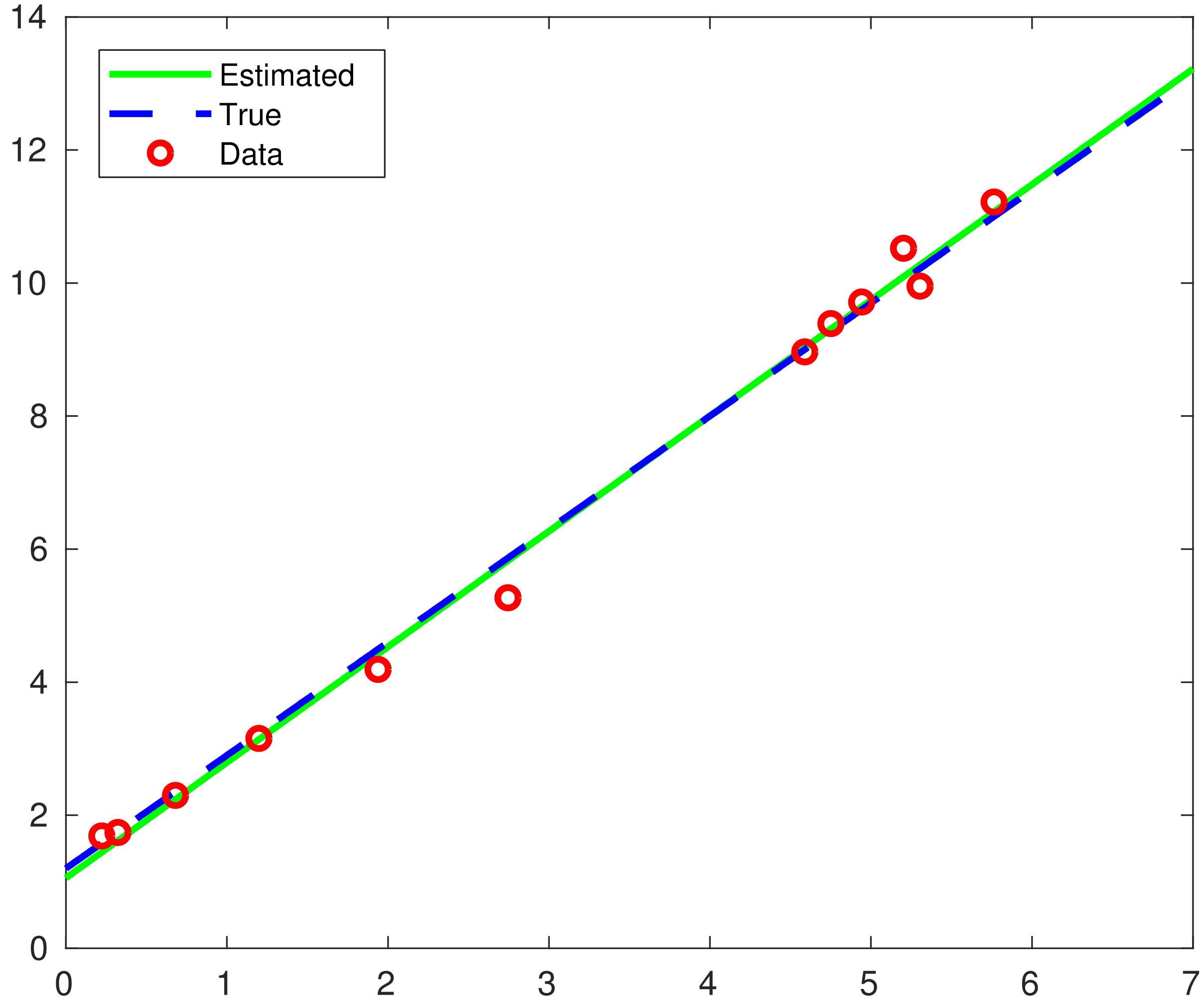
x	4.75	5.30	5.20	2.75	4.59	1.20	4.94	0.22	1.94	0.32	0.68	5.76
y	9.39	9.95	10.52	5.27	8.96	3.15	9.71	1.69	4.19	1.74	2.23	11.22

under the model:

$$\tilde{y}_i = a x_i + b + \varepsilon_i, \qquad \varepsilon_i \sim N(0, .35^2)$$

Exercise: Linear regression

x	*y*
4.75	9.39
5.30	9.95
5.20	10.52
2.75	5.29
4.59	8.96
1.20	3.15
4.94	9.71
0.22	1.69
1.94	4.19
0.32	1.74
0.68	2.30
5.76	11.22



Model:

$$y_i = ax_i + b$$

Estimated:

- a = 1.73
- b = 1.06

True:

- a = 1.70
- b = 1.20

Bundle adjustment

Recall: Given a set of point projections $\tilde{u}_{ij}$, we want to estimate:

- 3D point positions p_j
- camera poses x_i
- camera intrinsics K_i

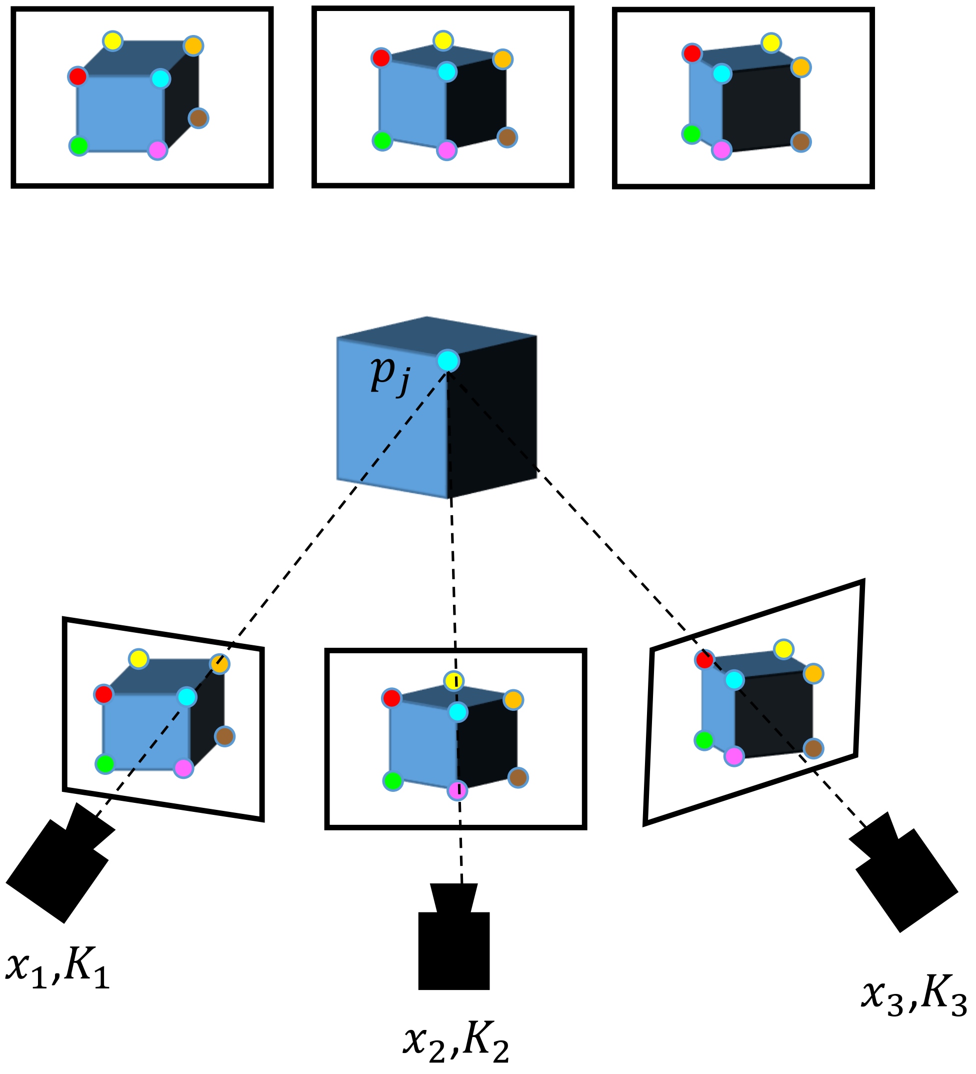
Assuming the measurement model:

$$\tilde{u}_{ij} = f\left(x_i, K_i, p_j\right) + \varepsilon_{ij}, \quad \varepsilon_i \sim N(0, \Sigma_i)$$

Maximum-likelihood estimation is then:

$$\hat{x}_i, \hat{K}_i, \hat{p}_j = \min_{x_i, K_i, p_j} \sum_{i,j} \left\| \tilde{u}_{ij} - f(x_i, K_i, p_j) \right\|_{\Sigma_{ij}}^2$$

$\Rightarrow$ *Minimize (weighted) reprojection error*

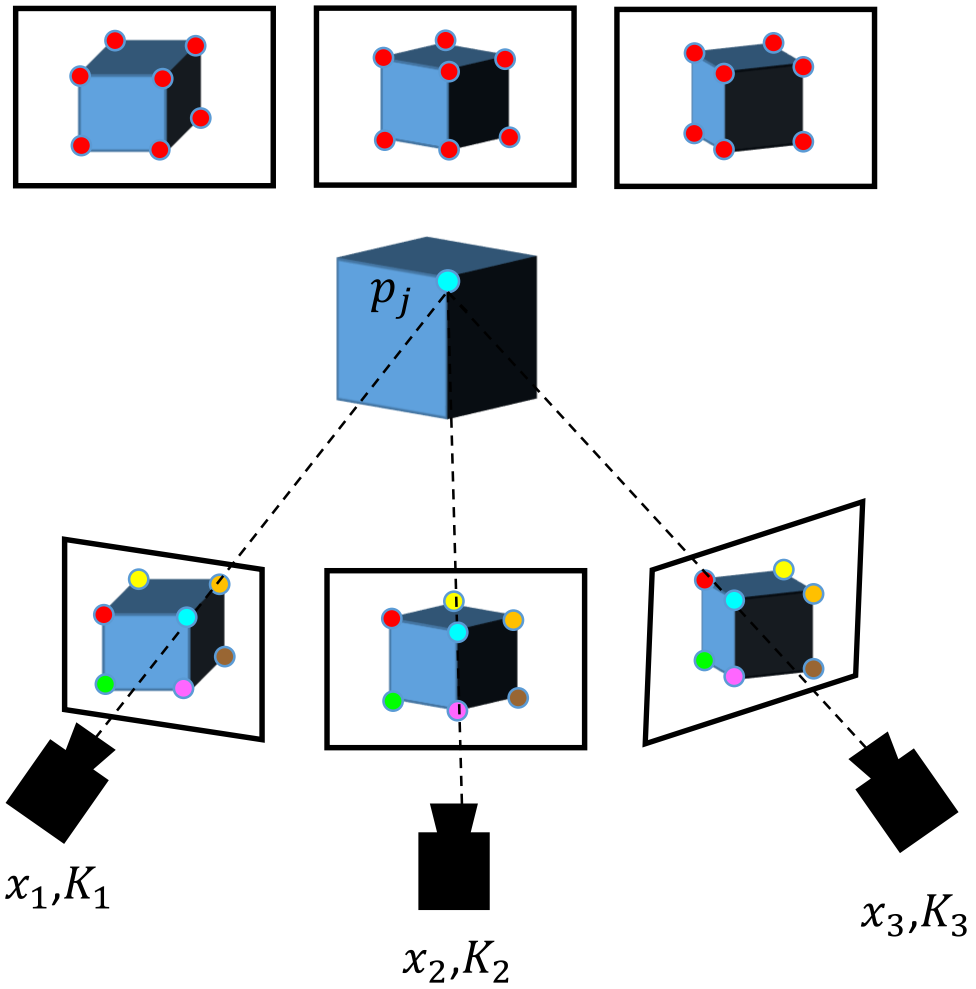


Photogrammetry and Bundle Adjustment

Given: A set of images

1. Extract features
2. Match features (identify 3D points)
3. Bundle adjust (minimize reprojection error):

$$\hat{x}_i, \hat{K}_i, \hat{p}_j = \min_{x_i, K_i, p_j} \sum_{i,j} \left\| \tilde{u}_{ij} - f(x_i, K_i, p_j) \right\|_{\Sigma_{ij}}^2$$



Special Case: Perspective-n-Point (PnP)

Given:

- Known point positions p_j
- Known camera intrinsics K

Estimate: Camera pose $x = (R,t)$

$$\hat{x} = \min_{x} \sum_{j=1}^{N} \left\| \tilde{u}_j - f(x, K, p_j) \right\|_{\Sigma_j}^2$$

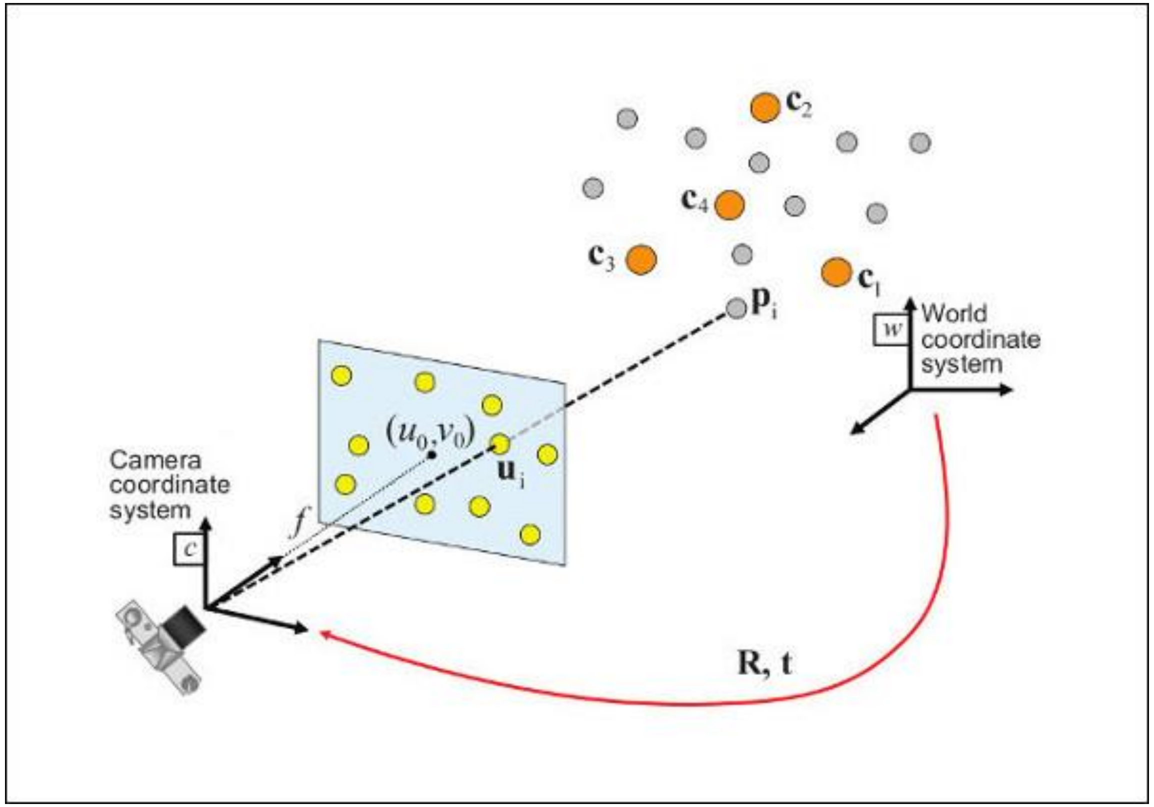


Image credit: OpenCV

Special Case: Camera calibration

Given:

- Known point positions p_j (on calibration object)

Estimate:

- Camera poses x_i
- Intrinsic parameters K

$$\hat{x}_i, \hat{K} = \min_{x_i, K} \sum_{i,j} \left\| \tilde{u}_{ij} - f(x_i, K, p_j) \right\|^2_{\Sigma_{ij}}$$

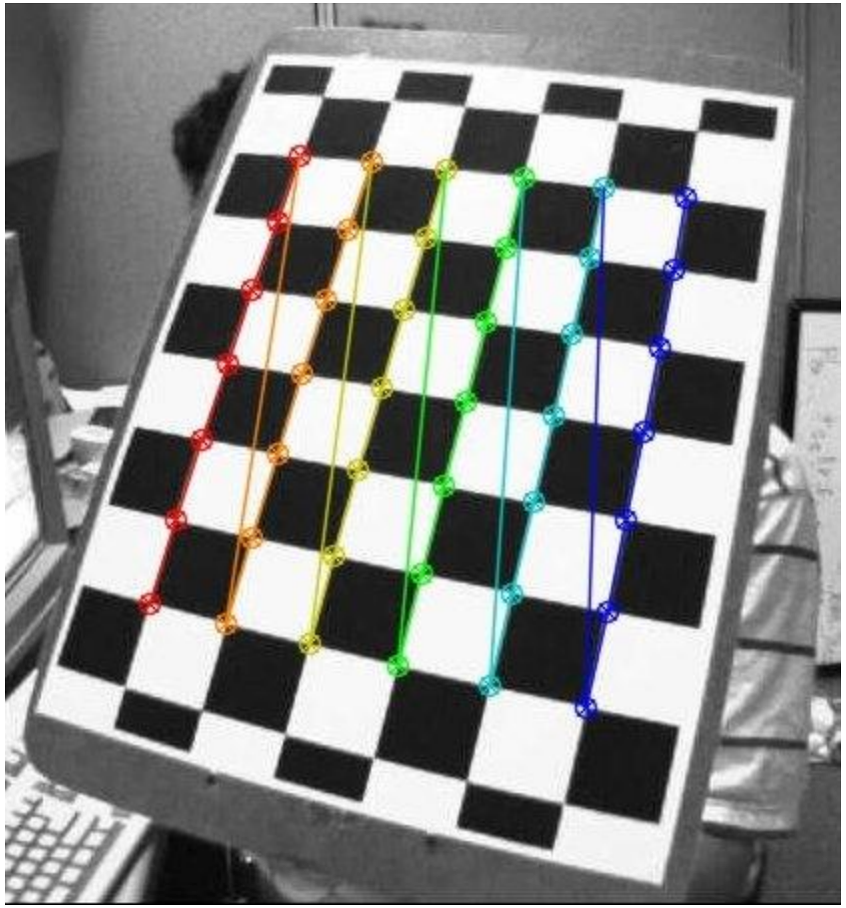


Image credit: OpenCV

Special Case: Stereovision

Given: A pair of cameras with:

- Known poses x_1, x_2
- Known intrinsics K_1, K_2

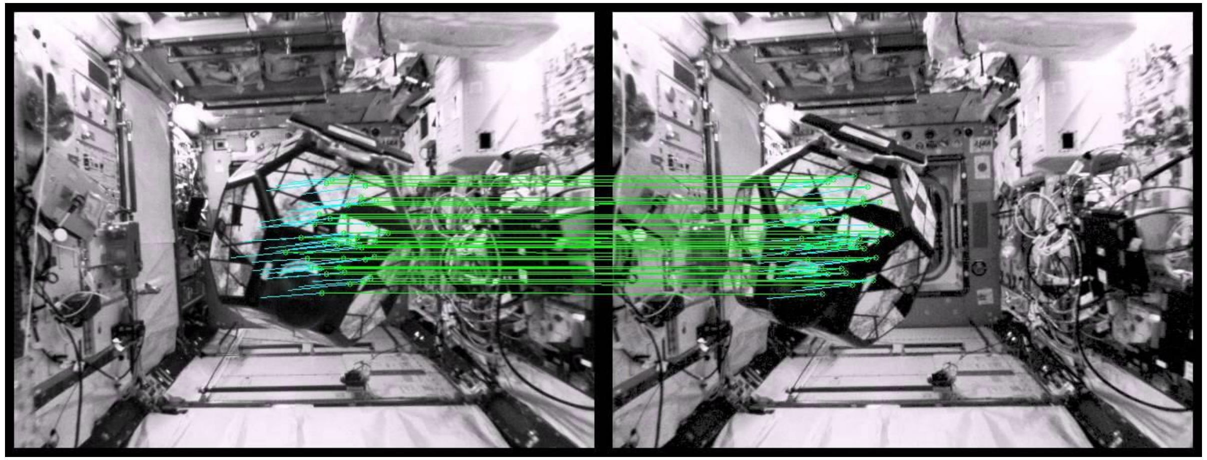


Image credit: MIT Space Systems Laboratory

Estimate: 3D point positions p_j

$$\hat{p}_j = \min_{p_j} \sum_j \left\| \tilde{u}_{1j} - f(x_1, K_1, p_j) \right\|^2_{\Sigma_{1j}} + \left\| \tilde{u}_{2j} - f(x_2, K_2, p_j) \right\|^2_{\Sigma_{2j}}$$

$$\Rightarrow \quad \hat{p}_j = \min_{p_j} \left\| \tilde{u}_{1j} - f(x_1, K_1, p_j) \right\|^2_{\Sigma_{1j}} + \left\| \tilde{u}_{2j} - f(x_2, K_2, p_j) \right\|^2_{\Sigma_{2j}}$$

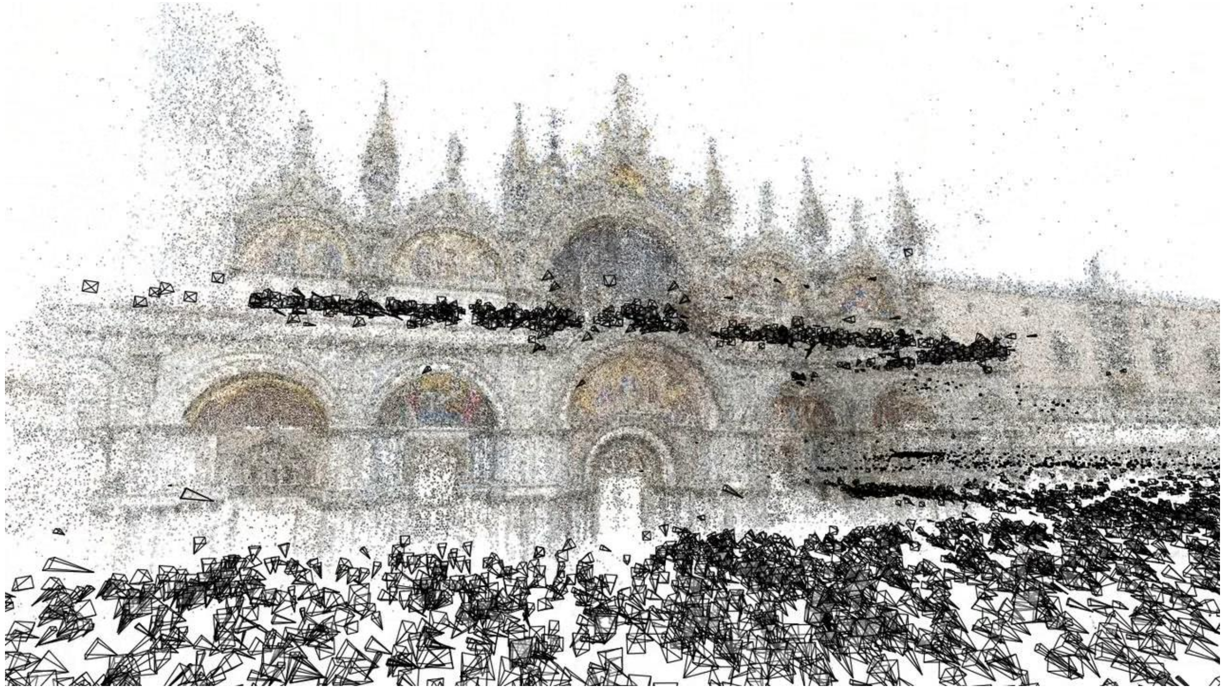
independently for all *j*

Practicality: Problem Scale

Piazza San Marco reconstruction:

- ~14,000 images
- ~4.5 million points

Assuming each point is observed 20x, what is the size of the BA problem?



Practicality: Problem Scale

Camera variables (0-skew, equal pixel scaling):

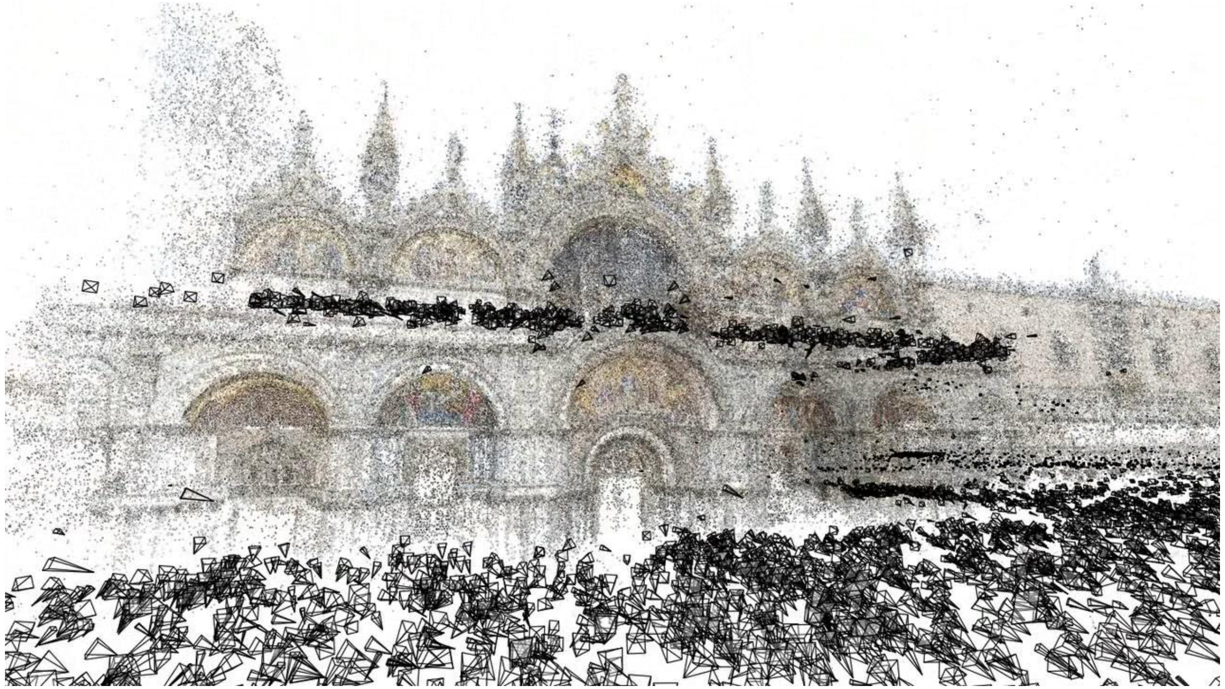
$$14{,}000 \cdot (6 + 3) = 126{,}000$$

Point variables:

$$4{,}500{,}000 \cdot 3 = 13{,}500{,}000$$

Point observations:

$$4{,}500{,}000 \cdot 20 \cdot 2 = 180{,}000{,}000$$



Totals:

- 13,626,000-dimensional **state** vector
- 180,000,000-dimensional **residual** vector

$\Rightarrow$ This is a ***huge*** optimization problem!

Practicality: Feature mismatches

Recall: We construct the bundle adjustment problem:

$$\hat{x}_i, \widehat{K}_i, \hat{p}_j = \min_{x_i, K_i, p_j} \sum_{i,j} \left\| \tilde{u}_{ij} - f(x_i, K_i, p_j) \right\|_{\Sigma_{ij}}^2$$

using *estimated* feature matches.

But: What happens if these are *mis-estimated*?

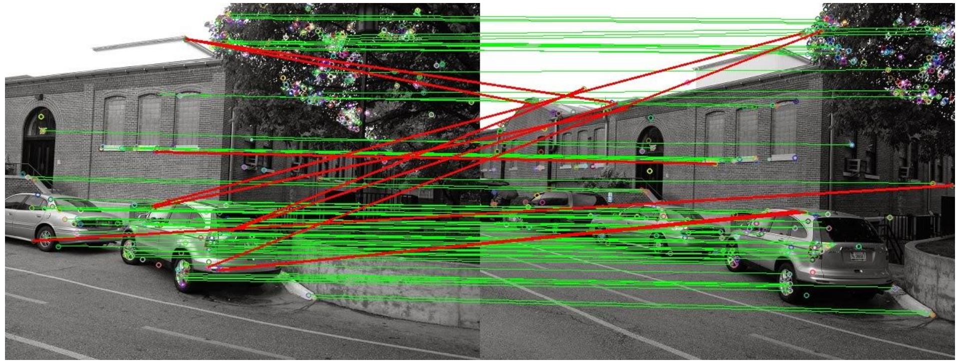
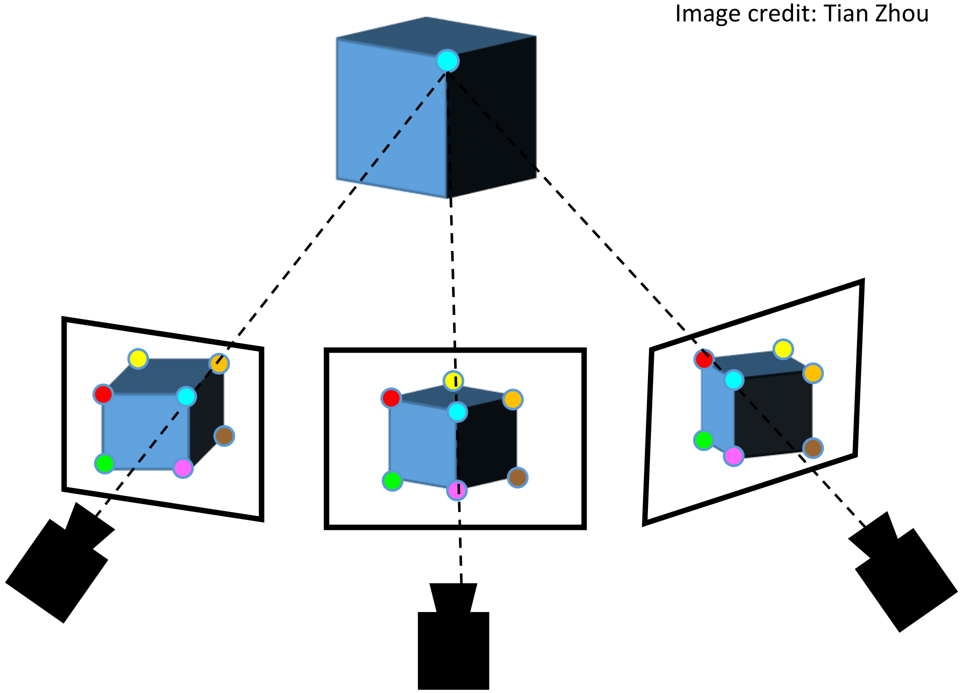


Image credit: Tian Zhou

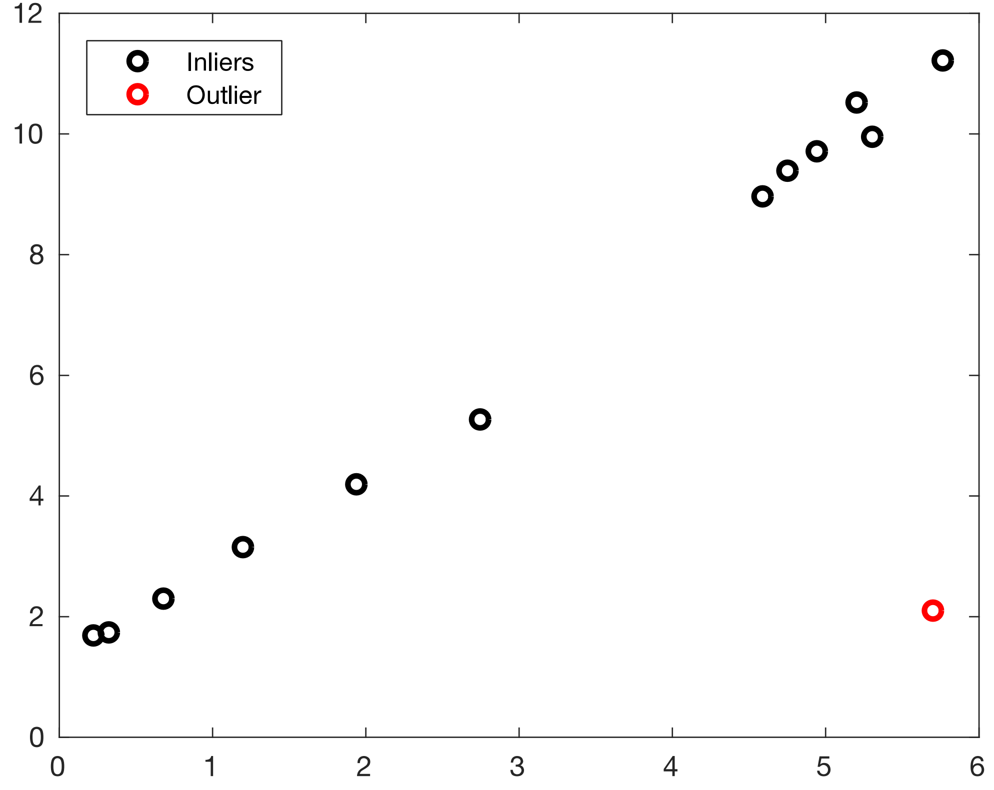


Example: Contaminated linear regression

Consider fitting the linear model:

$$\tilde{y}_i = ax_i + \varepsilon_i, \quad \varepsilon_i \sim N\left(0, .35^2\right)$$

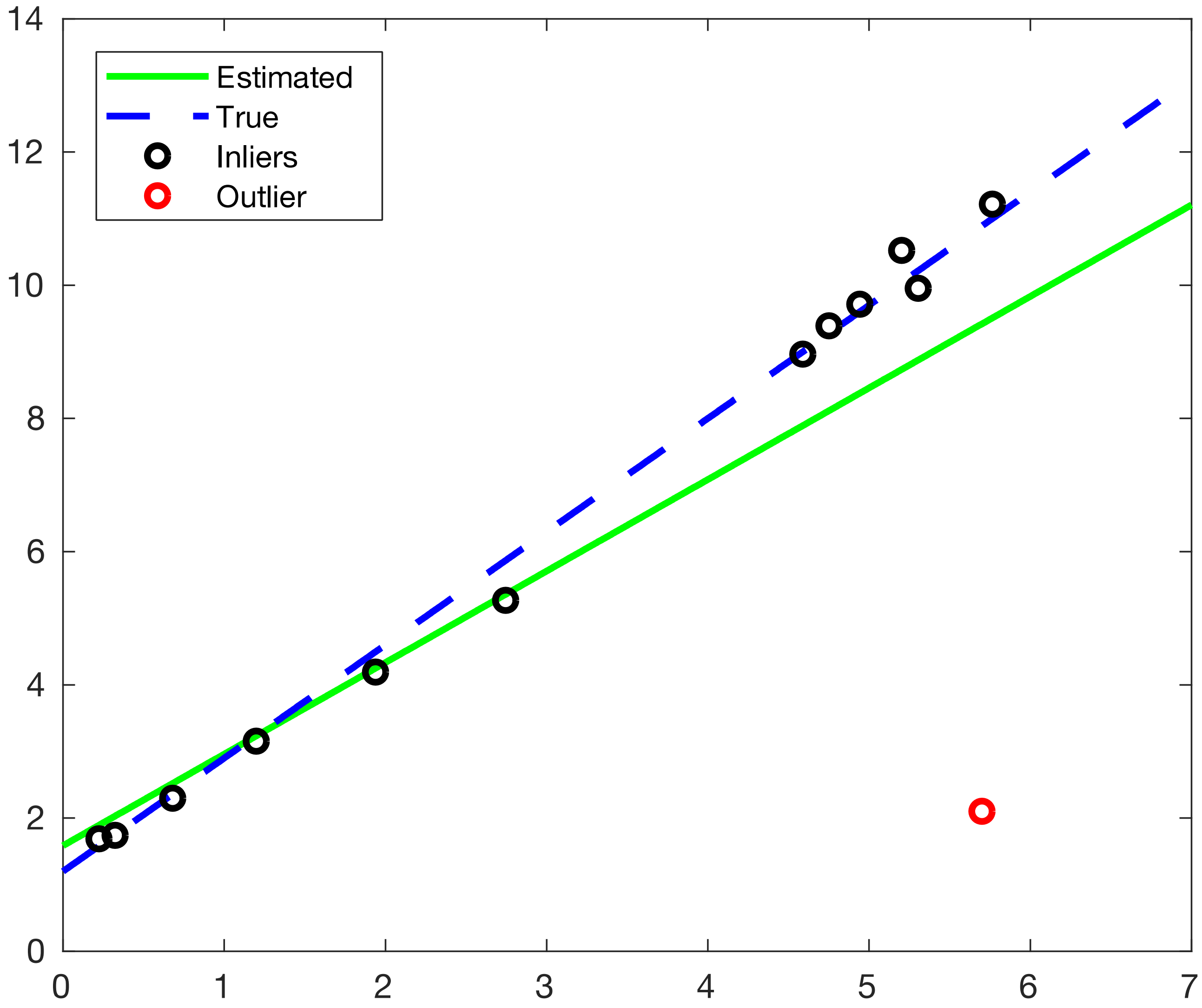
to a *contaminated* data set



x	4.75	5.30	5.20	2.75	4.59	1.20	4.94	0.22	1.94	0.32	0.68	5.76	**5.70**
y	9.39	9.95	10.52	5.27	8.96	3.15	9.71	1.69	4.19	1.74	2.23	11.22	**2.10**

Exercise: Contaminated linear regression

x	y
4.75	9.39
5.30	9.95
5.20	10.52
2.75	5.29
4.59	8.96
1.20	3.15
4.94	9.71
0.22	1.69
1.94	4.19
0.32	1.74
0.68	2.30
5.76	11.22
5.70	**2.10**



Model:

$$y_i = ax_i + b$$

Estimated:

- a = 1.37
- b = 1.58

True:

- a = 1.70
- b = 1.20

The problem of outliers

Recall: We assumed *additive Gaussian image noise*:

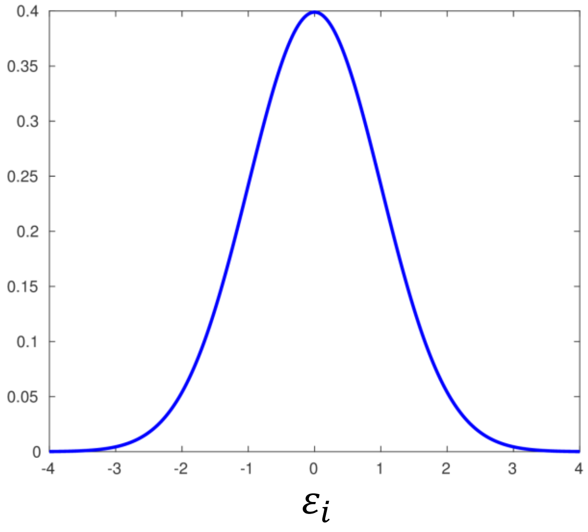
$$\tilde{u}_{ij} = f\left(x_i, K_i, p_j\right) + \varepsilon_{ij}, \qquad \varepsilon_{ij} \sim N(0, \Sigma_{ij})$$

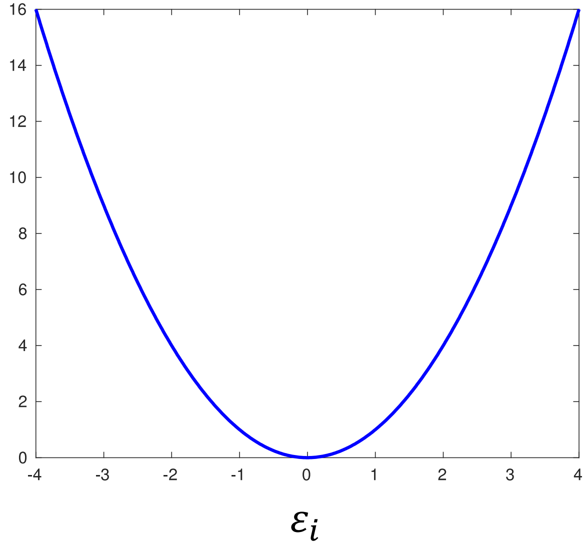
and obtained a *nonlinear least-squares* problem:

$$\hat{x}_i, \widehat{K}_i, \hat{p}_j = \min_{x_i, K_i, p_j} \sum_{i,j} \left\| \tilde{u}_{ij} - f(x_i, K_i, p_j) \right\|_{\Sigma_{ij}}^2$$

NB: This loss weights extreme errors *very* heavily.

$\Rightarrow$ This estimator is *not robust*!



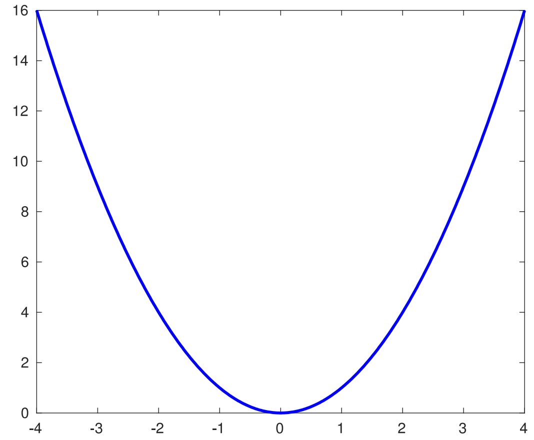


Robust loss functions

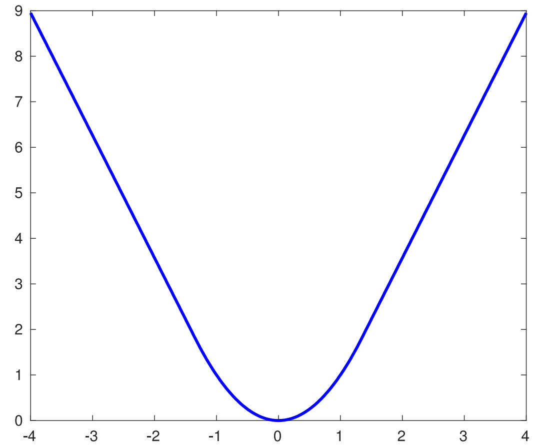
One solution: Replace quadratic loss with a function that *attenuates gross errors*

Tradeoff:

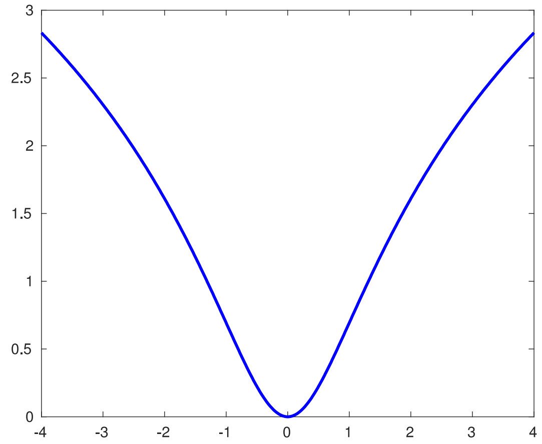
- More robust to *outliers*
- (Slightly) less *statistical* power



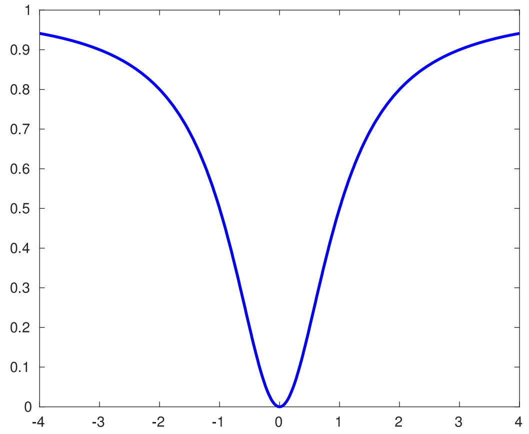
Quadratic



Huber

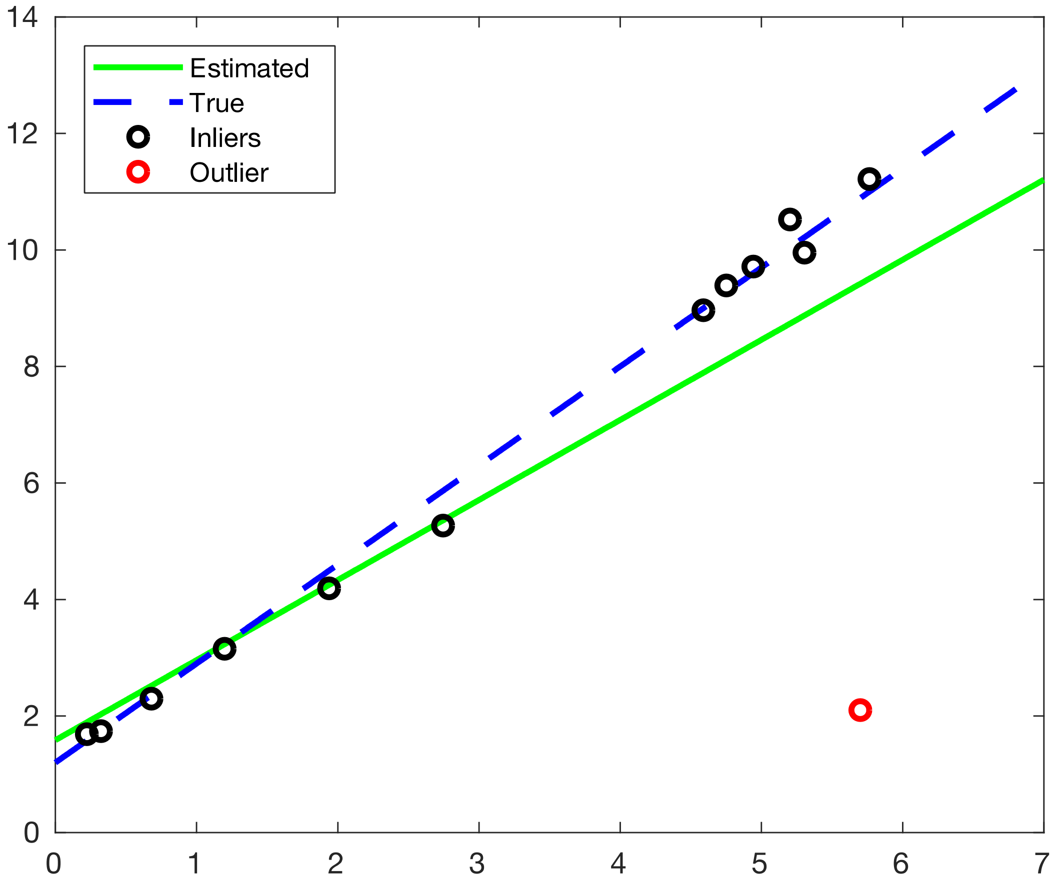


Cauchy

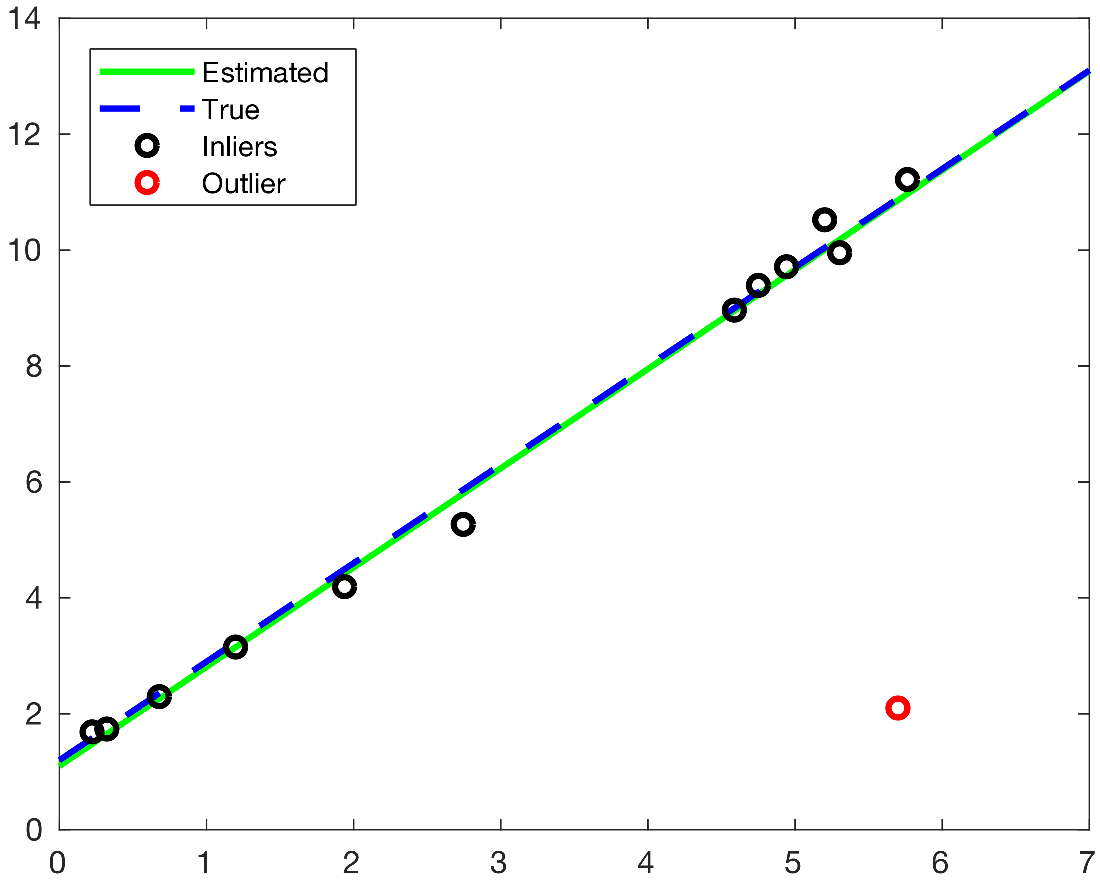


Geman-McClure

Example: Contaminated linear regression



Least-squares loss



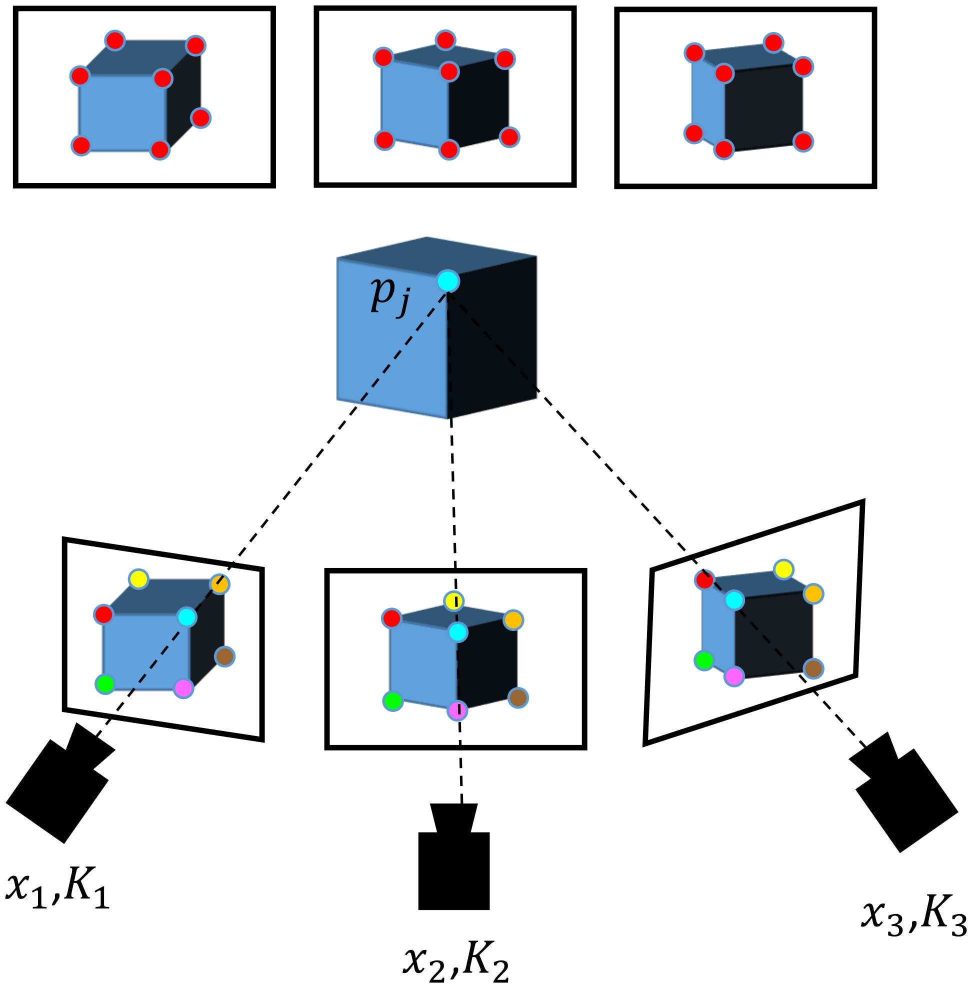
Huber loss

Photogrammetry and Bundle Adjustment: Summary

Given: A set of images

1. Extract features
2. Match features (identify 3D points)
3. Bundle adjust using a *robust loss function ρ*:

$$\hat{x}_i, \hat{K}_i, \hat{p}_j = \min_{x_i, K_i, p_j} \sum_{i,j} \rho\left(\left\| \tilde{u}_{ij} - f(x_i, K_i, p_j) \right\|_{\Sigma_{ij}} \right)$$



Practicality: Representing rotations

So far, we've represented rotations using *rotation matrices*:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad R^T R = I_3, \;\; \det(R) = +1$$

Pro: Trivial point operations

Con: *Over-parameterized*

$\Rightarrow$ Not super convenient for optimization (requires *constraints*)

Euler's Rotation Theorem

Theorem: Every rotation of 3D space has a fixed axis ***e***.

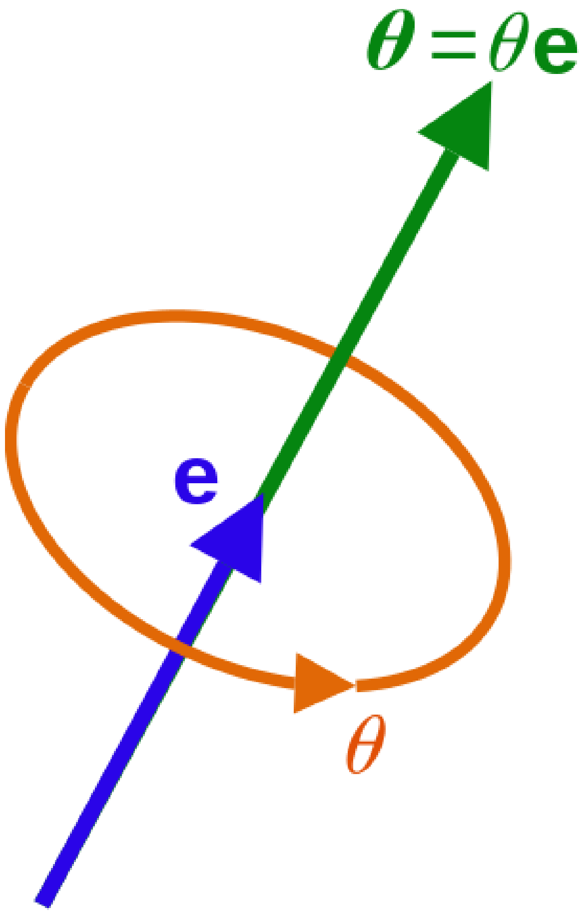
$\Rightarrow$We can describe a rotation using:

- An *axis* (*unit* vector ***e***)
- (Right-handed) rotation *angle* θ

This is the *axis-angle* parameterization of rotations.

Can also combine these into a single *axis-angle vector*:

$$\boldsymbol{\theta} = \theta \boldsymbol{e}$$



Rodrigues' Formula

How does a rotation parameterized as $\boldsymbol{\theta} = \theta \boldsymbol{e}$ act on points?

Rodrigues' formula: Given a vector $\boldsymbol{v}$,

$$\boldsymbol{v}_{rot} = \boldsymbol{v}\ \cos\theta + \sin\theta\,(\boldsymbol{e} \times \boldsymbol{v}) + (1 - \cos\theta)(\boldsymbol{e} \cdot \boldsymbol{v})\boldsymbol{e}$$

NB:

- ***Any*** 3D vector $\boldsymbol{\theta}$ determinates a valid rotation
- Rodrigues' formula is differentiable in $\boldsymbol{\theta}$

$\Rightarrow$ Axis-angle is *much* more convenient for use in optimization!

Rodrigues' Formula

How does a rotation parameterized as $\boldsymbol{\theta} = \theta \boldsymbol{e}$ act on points?

Rodrigues' formula: Given a vector $\boldsymbol{v}$,

$$\boldsymbol{v}_{rot} = \boldsymbol{v} \cos\theta + \sin\theta\,(\boldsymbol{e} \times \boldsymbol{v}) + (1 - \cos\theta)(\boldsymbol{e} \cdot \boldsymbol{v})\boldsymbol{e}$$

Matrix form:

$$R(\theta) = I + (\sin\theta)\,E + (1 - \cos\theta)E^2\,,$$

where

$$E = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix}$$

Photogrammetry and Bundle Adjustment: Summary

Given: A set of images

1. Extract features
2. Match features (identify 3D points)
3. Bundle adjust using a *robust loss function ρ*:

$$\hat{x}_i, \hat{K}_i, \hat{p}_j = \min_{x_i, K_i, p_j} \sum_{i,j} \rho\left(\left\| \tilde{u}_{ij} - f(x_i, K_i, p_j) \right\|_{\Sigma_{ij}} \right)$$