CSE 252D: Advanced Computer Vision

Manmohan Chandraker

Lecture 6: Structure and Motion

Virtual classrooms

- Virtual lectures on Zoom
  - Only host shares the screen
  - Keep video off and microphone muted
  - But please do speak up (remember to unmute!)
  - Slides uploaded on webpage just before class
- Virtual interactions on Zoom
  - Ask and answer plenty of questions
  - “Raise hand” feature on Zoom when you wish to speak
  - Post questions on chat window
  - Happy to try other suggestions!
- Lectures recorded and upload on Kaltura
  - Available under “My Media” on Canvas

Overall goals for the course

- Introduce the state-of-the-art in computer vision
- Study principles that make them possible
- Get understanding of tools that drive computer vision
- Enable one or all of several such outcomes
  - Pursue higher studies in computer vision
  - Join industry to do cutting-edge work in computer vision
  - Gain appreciation of modern computer vision technologies
- This is a great time to study computer vision!

Lighting Presentations

- Look out for email with steps for completing the presentation
  - Confirm the paper assignment when you receive the email
  - Send presentation to instructor and TA 3 days before class
  - Include script for narration
  - Incorporate feedback
  - Send final version and recorded video 1 day before class

- Order of presentation: alphabetic
  - Papers assigned by instructor
  - https://docs.google.com/spreadsheets/d/1JcM4V2GaPf7WF2YLFQ70Rn6BaLYEOqEzzj8rXOsb5bw/edit?usp=sharing

Lighting Presentations

- Time limit: 5 minutes ☺
  - **High-quality presentation:** well-practiced and fluent
  - Speak slowly, clearly and explain the content
- General rules
  - **Illustrate:** 1 image per slide, a few bullet points to explain it
  - **Related work:** don't list, rather contrast to 1-2 most important ones
  - **Experiments:** datasets, 1-2 key numbers or graphs
  - **Big no-no's:** giant tables of numbers, hyperparameters
- Template has been provided
  - Try to follow to the extent possible
  - Can replace prompts ("key design element") with relevant content
  - Avoid increasing the number of slides

Papers for Wed, Apr 21

- GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose
  - https://arxiv.org/abs/1803.02276

- Unsupervised Scale-Consistent Depth and Ego-Motion Learning from Monocular Video
  - https://arxiv.org/abs/1908.10553

Papers for Fri, Apr 23

- Building Rome in a Day
  - https://grail.cs.washington.edu/rome/rome_paper.pdf
- CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM
  - https://arxiv.org/abs/1804.00874
- Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry
  - https://arxiv.org/abs/1904.01892
- BA-Net: Dense Bundle Adjustment Network
  - https://arxiv.org/abs/1806.04807

Papers for Wed, Apr 28

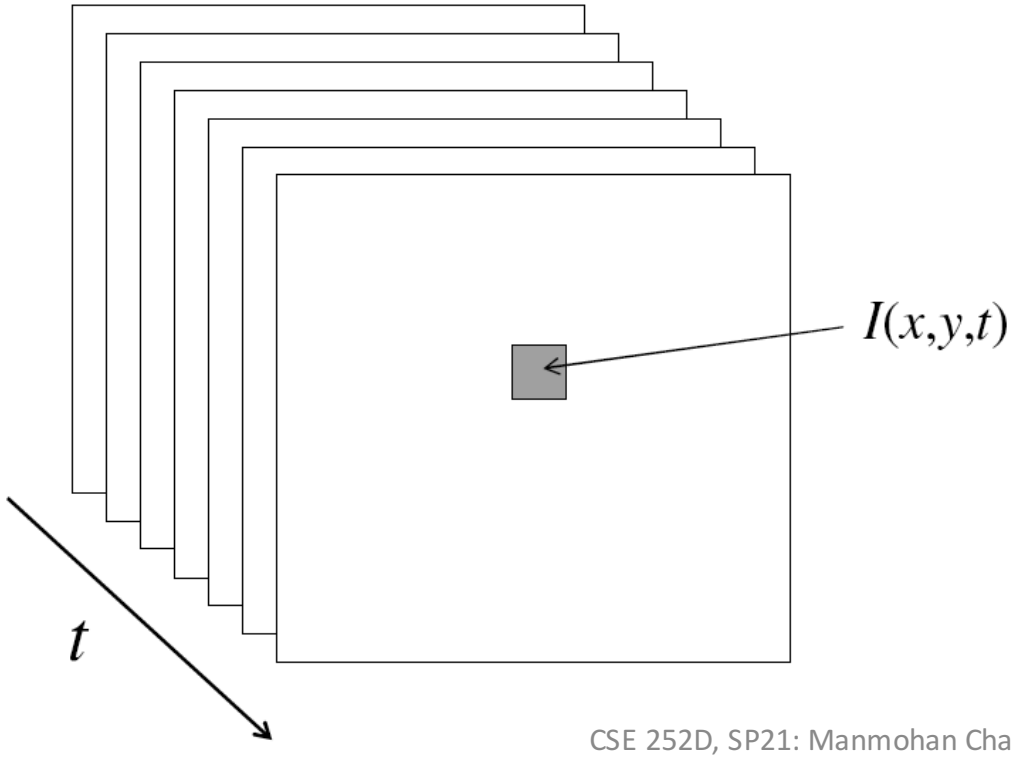
- Deep Fundamental Matrix Estimation
  - https://openaccess.thecvf.com/content_ECCV_2018/html/Rene_Ranftl_Deep_Fundamental_Matrix_ECCV_2018_paper.html
- DSAC - Differentiable RANSAC for Camera Localization
  - https://arxiv.org/abs/1611.05705
- Unsupervised Monocular Depth Estimation with Left-Right Consistency
  - https://arxiv.org/abs/1609.03677
- LSD-SLAM: Large-Scale Direct Monocular SLAM
  - https://vision.in.tum.de/_media/spezial/bib/engel14eccv.pdf

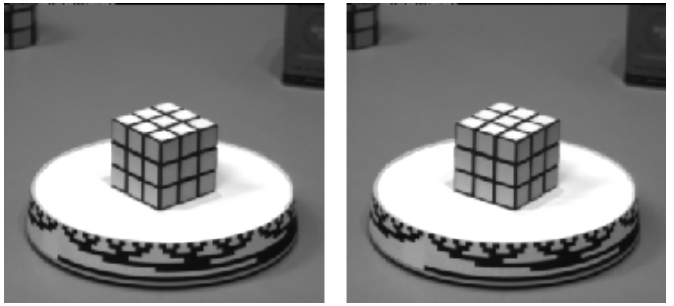
Recap

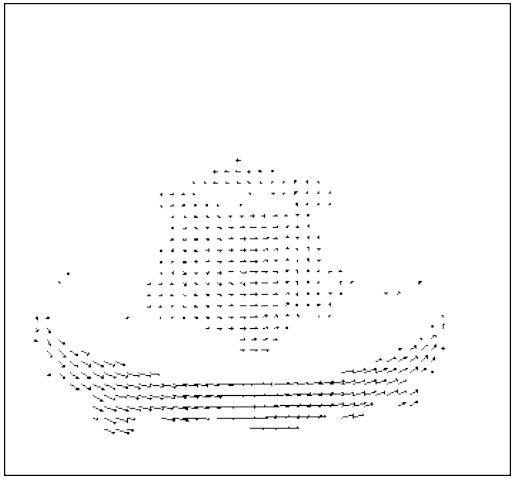
Optical flow

Brightness constancy constraint equation

$$I_x\, u + I_y\, v + I_t = 0$$

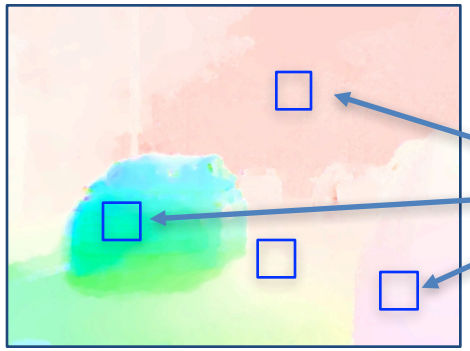






Optical flow: Lucas-Kanade

- Overconstrained linear system through patch coherence



Assume a constant flow in K x K patch

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

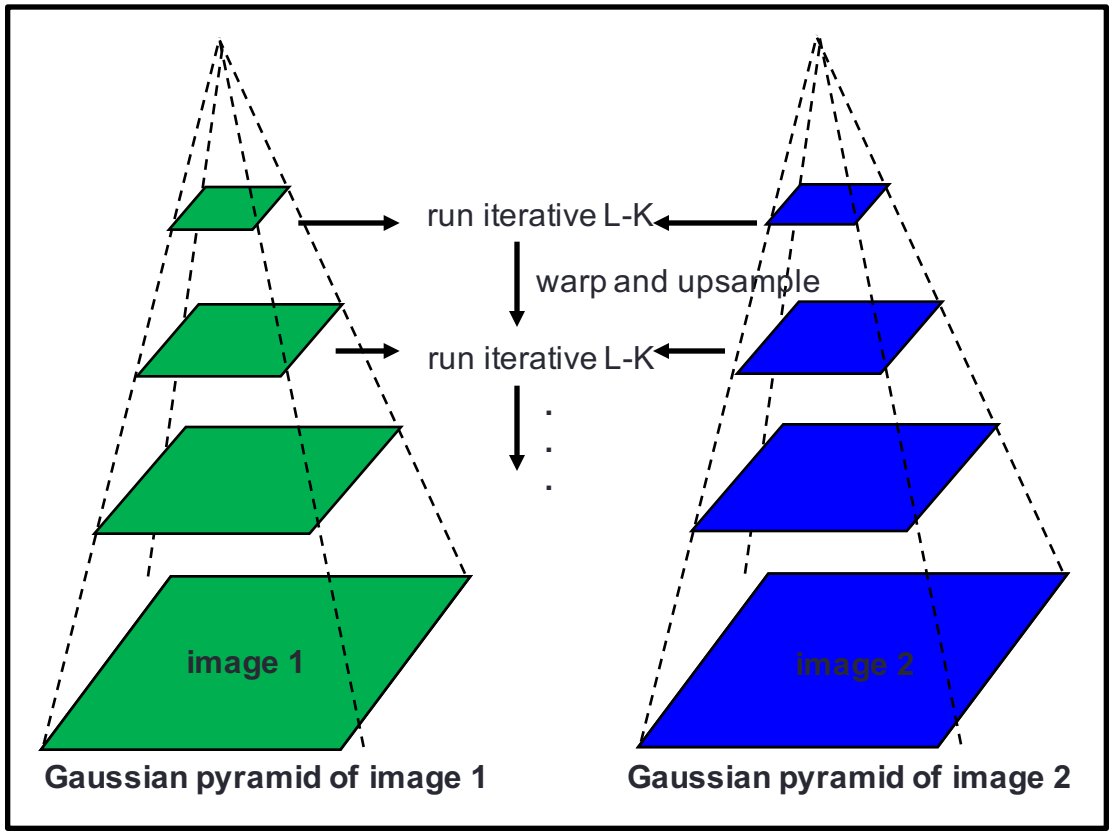
Least squares solution for *d* given by $(A^T A)\ d = A^T b$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

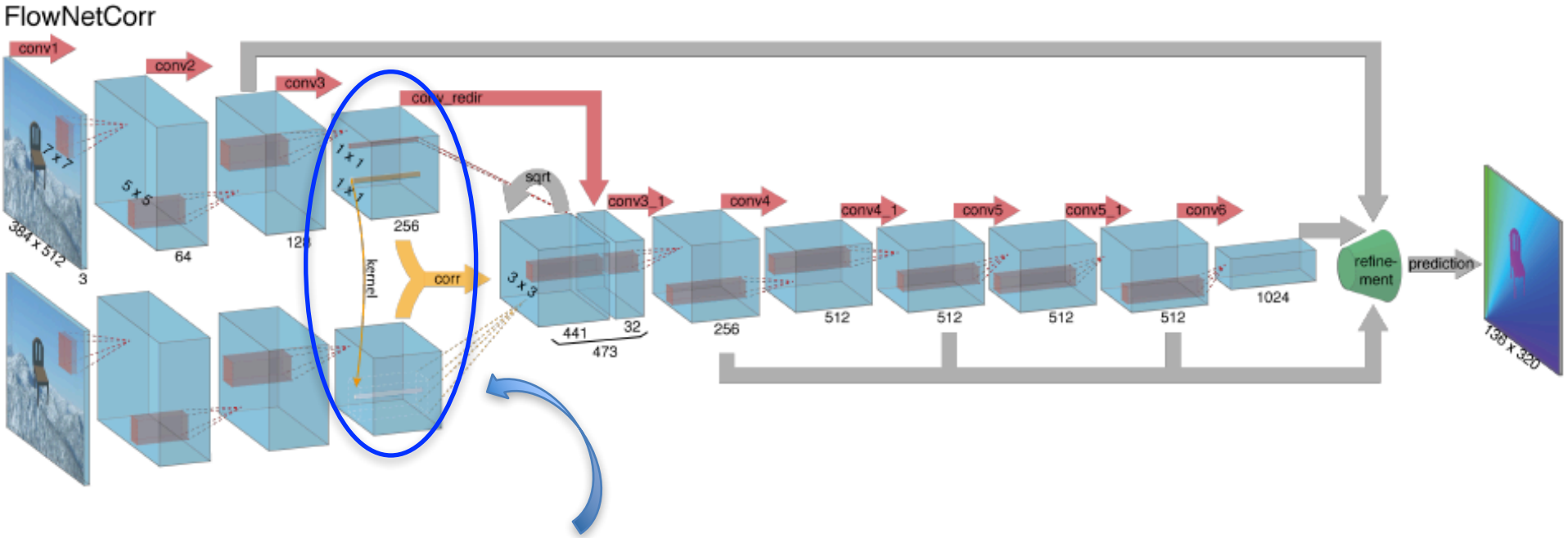
$A^T A$ $\qquad\qquad$ $A^T b$

The summations are over all pixels in the K x K window

Optical flow: Coarse-to-Fine



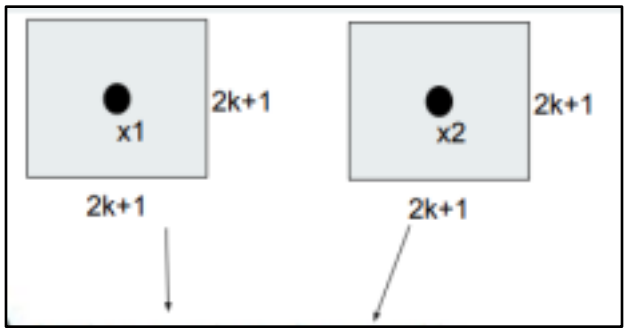
FlowNet: Correlation Layer



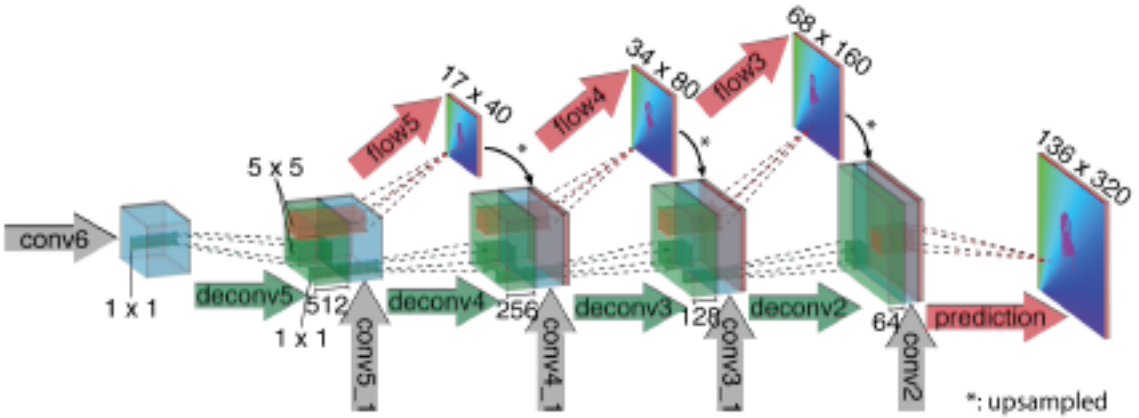
- Multi-channel feature maps $\mathbf{f}_1$ and $\mathbf{f}_2$, of size *w* x *h* and *n* channels
- Correlation of patches centered at $\mathbf{x}_1$ and $\mathbf{x}_2$:

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

- Number of trainable parameters?
- Cost of computing correlations?



FlowNet: Refinement

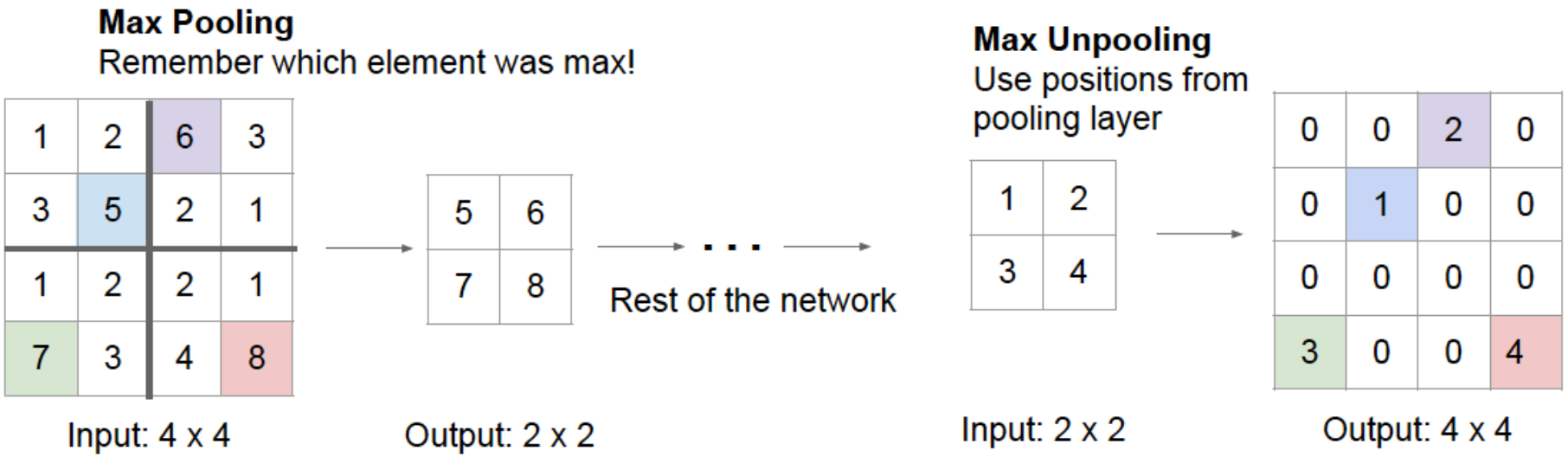


- Gradually upsample the low-dimensional feature.
- Concatenate with encoder feature of corresponding scale, to recover details.

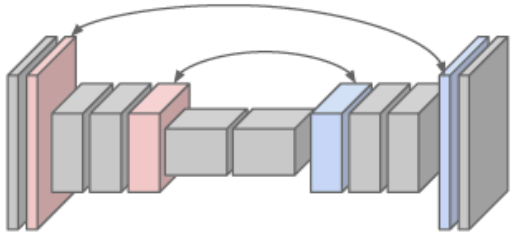
- In simplest form, upsampling can be implemented as bilinear interpolation.

- Can be learned as unpooling followed by convolution
- Can be learned as a transposed convolution filter

Refinement: Unpooling followed by convolution

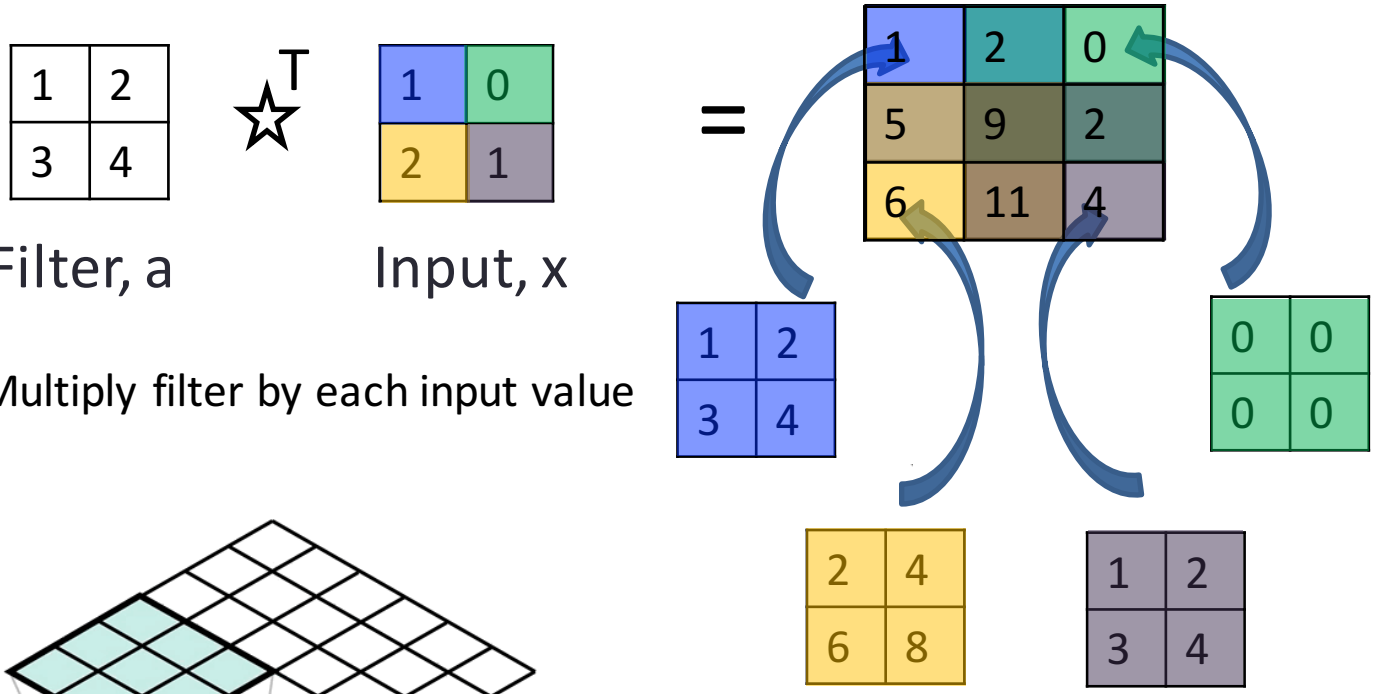


Corresponding pairs of downsampling and upsampling layers



[Figure from: L. Fei-Fei et al., CS 231n]

Transposed Convolutions

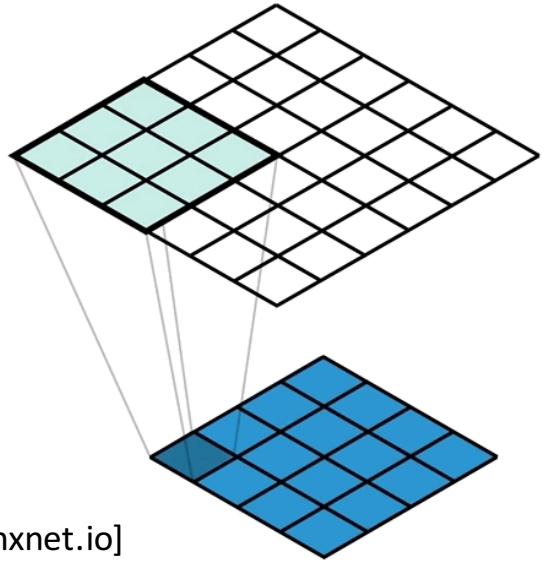


Filter, a

Input, x

Multiply filter by each input value

Tile the scaled filter at output locations
Add overlapping values



[Thom Lane, mxnet.io]

Transposed Convolutions

1	2
3	4

Filter, a

$\overset{T}{\star}$

1	0
2	1

Input, x

$=$

1	2	0
5	9	2
6	11	4

1	0	0	0
2	1	0	0
0	2	0	0
3	0	1	0
4	3	2	1
0	4	0	2
0	0	3	0
0	0	4	3
0	0	0	4

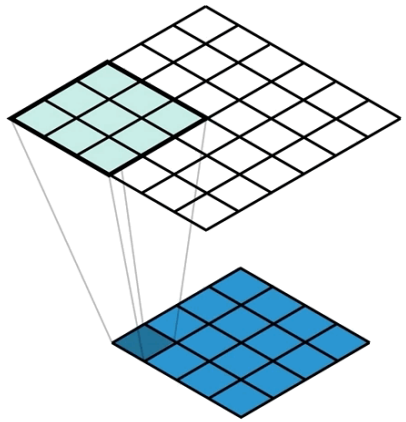
A^T

X

1
0
2
1

$=$

1
2
0
5
9
2
6
11
4



Spatial Pyramid Network

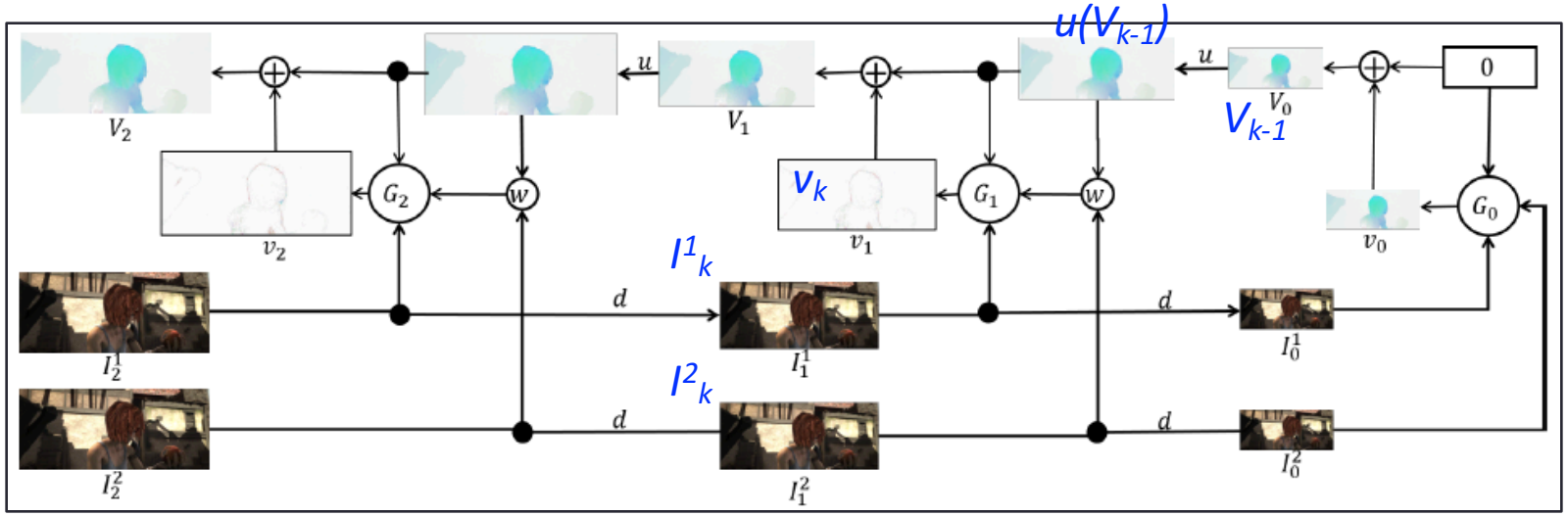
- Basic goal: learn a CNN G_k to predict residual flow at each level:

$$v_k = G_k(I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1}))$$

- At level k:
  - Use I^1_k and warped I^2_k, with upsampled flow from level k-1, to predict residual
  - Add residual to upsampled flow at level k-1 to obtain flow at level k
- Train each level G_k sequentially to predict residual at level k, given trained G_{k-1}
  - Ground truth residual = (Downsampled ground truth flow) – (Upsampled prediction)

$$\hat{v}_k = \hat{V}_k - u(V_{k-1})$$

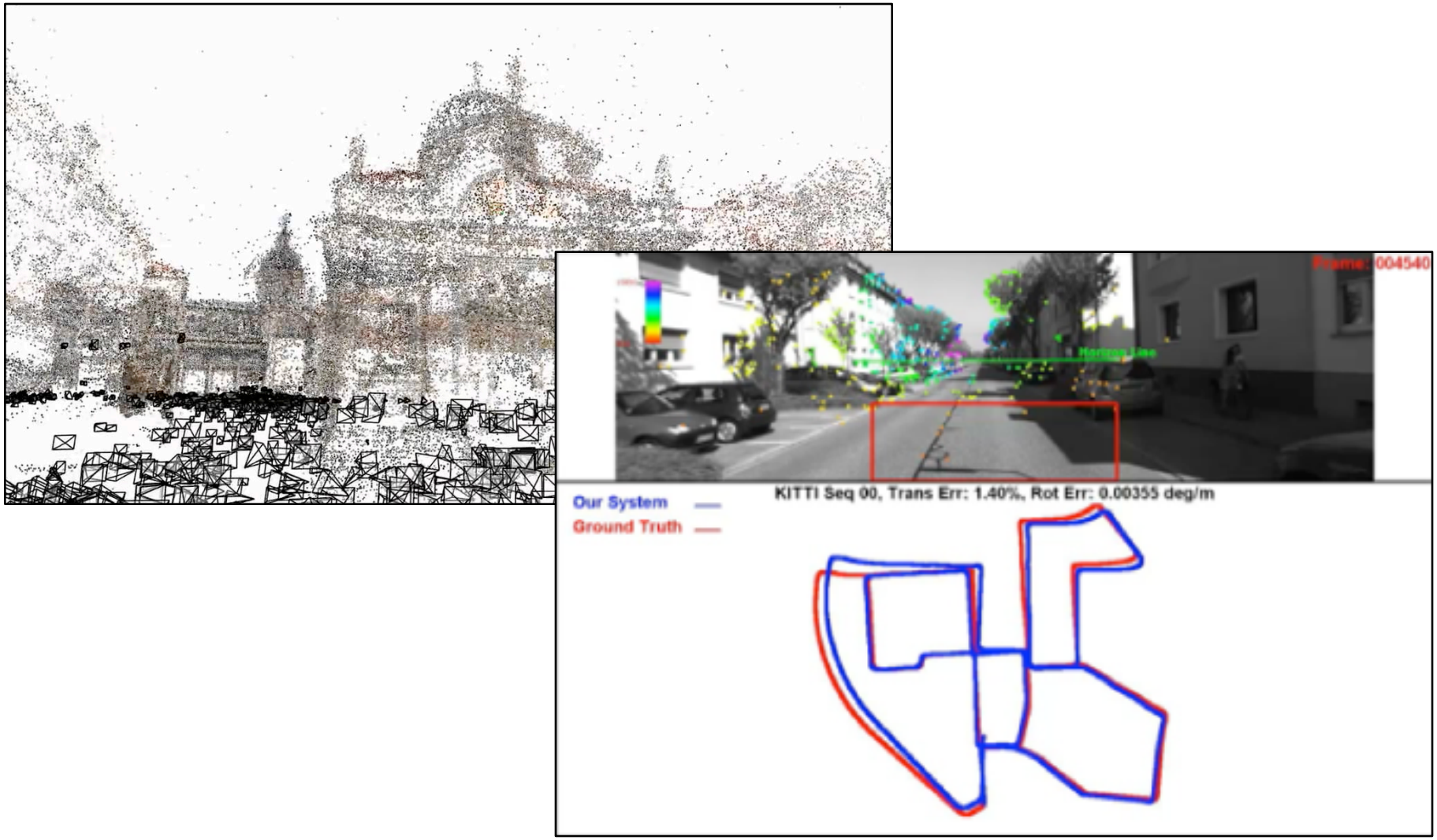
- Each level solves a simple problem, so each level G_k can have simple architecture



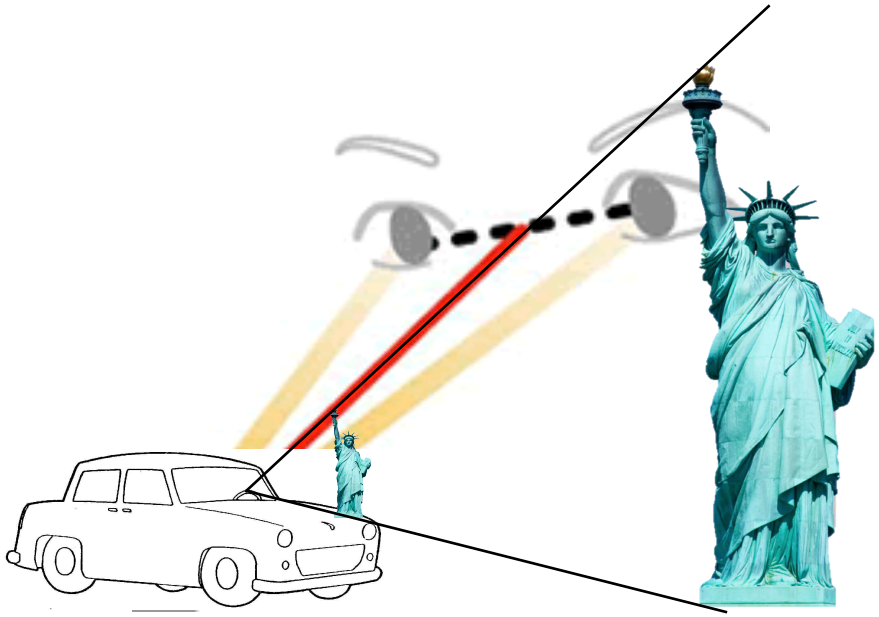
[Optical Flow Estimation Using a Spatial Pyramid Network, CVPR 2017]

Structure from Motion

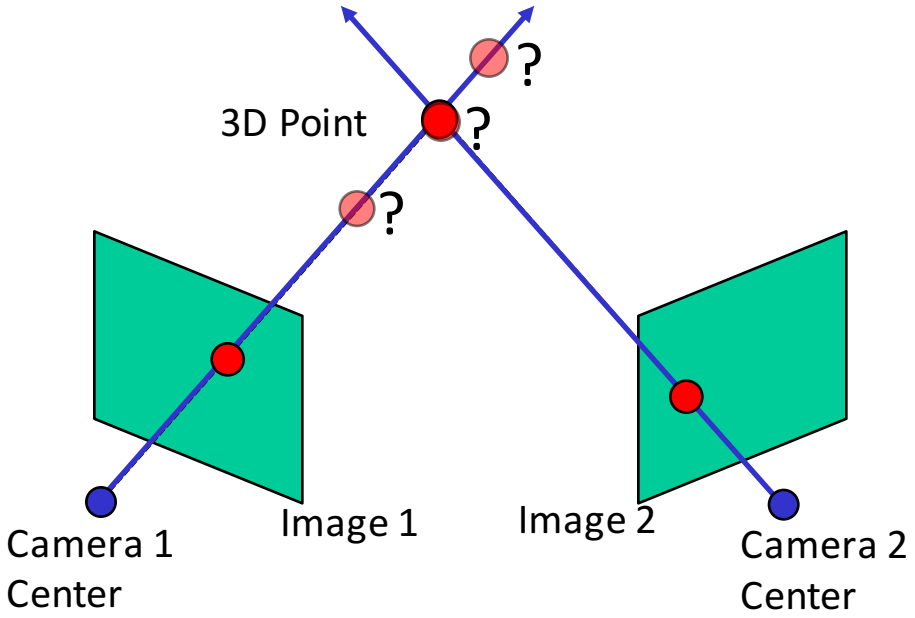
Unordered or Ordered Images



Correspondence is a vital 3D cue



Scale Ambiguity

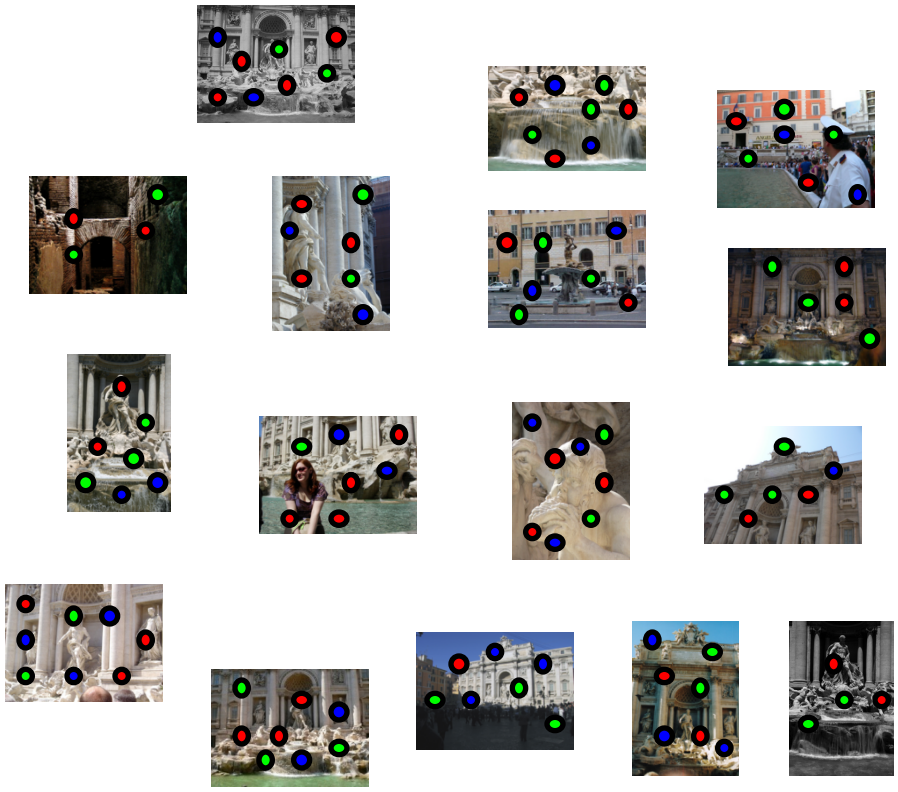


Feature detection

Several images observe a scene from different viewpoints

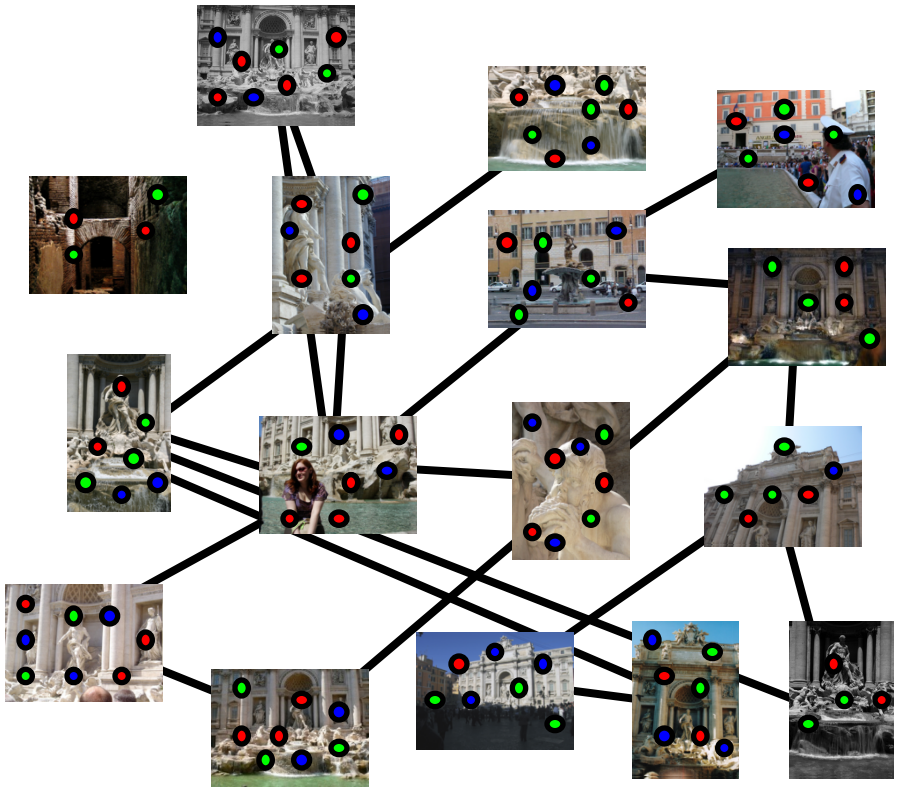
Feature detection

Detect features using, for example, SIFT [Lowe, IJCV 2004]

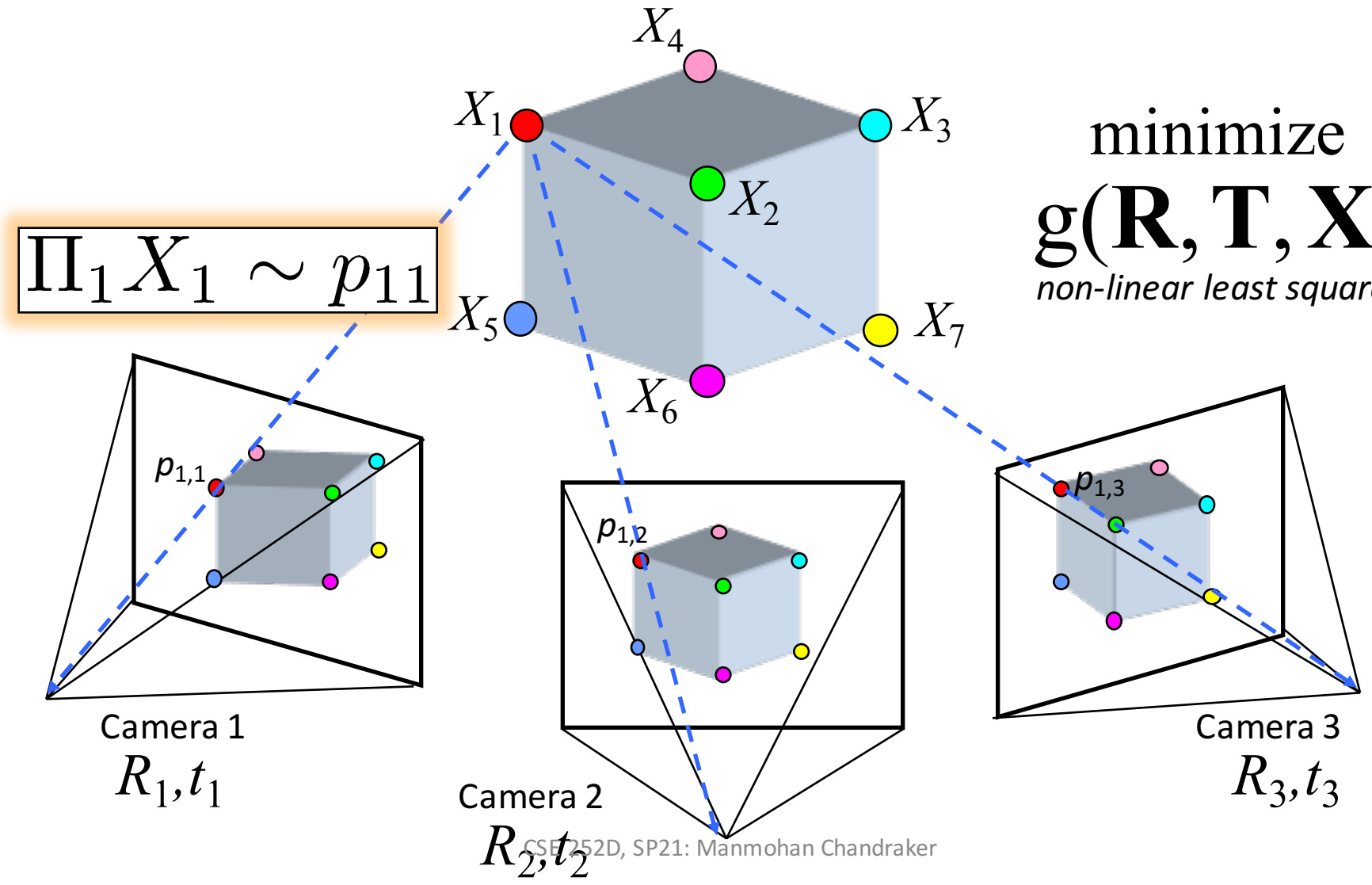


Feature matching

Match features between each pair of images



Structure from motion



Bundle adjustment

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

indicator variable: whether point *i* visible in image *j*

predicted image location

observed image location

  - Optimized with non-linear least squares
  - Levenberg-Marquardt is a popular choice
- Practical challenges?

Bundle adjustment

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

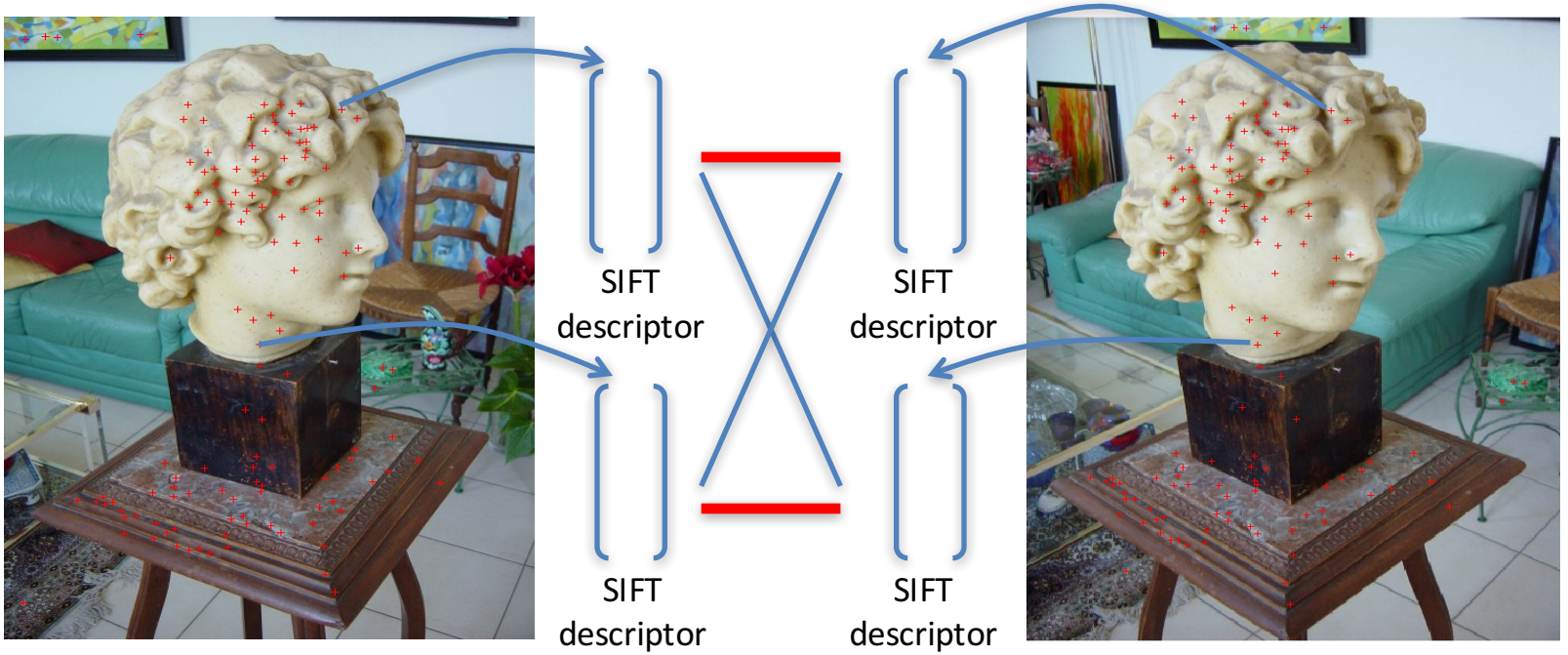
indicator variable: whether point *i* visible in image *j*

predicted image location

observed image location

  - Optimized with non-linear least squares
  - Levenberg-Marquardt is a popular choice
- Practical challenges?
  - Initialization
  - Outliers

Matching and Optimization



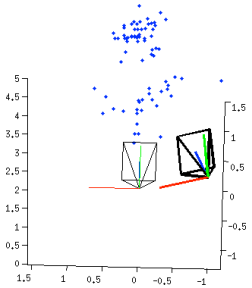
Master Plan! Repeat for all pairs?


Keypoints

Keypoints

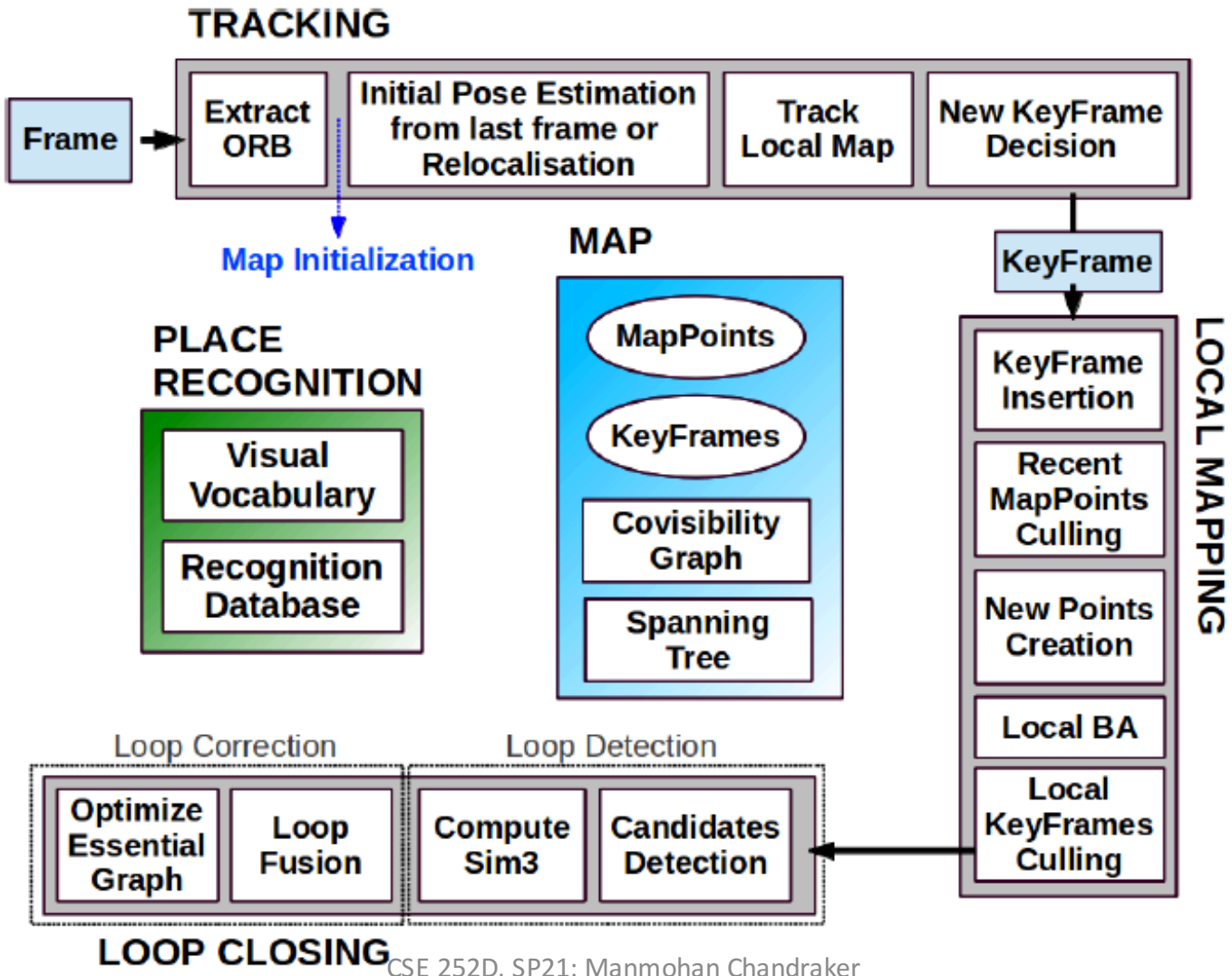
Match

Optimization for camera poses and 3D points



An Actual SFM System

- Use fundamental building blocks, but need to do a lot for robustness



[Mur-Artal et al., ORB-SLAM]

Some Recipes for SFM to Work

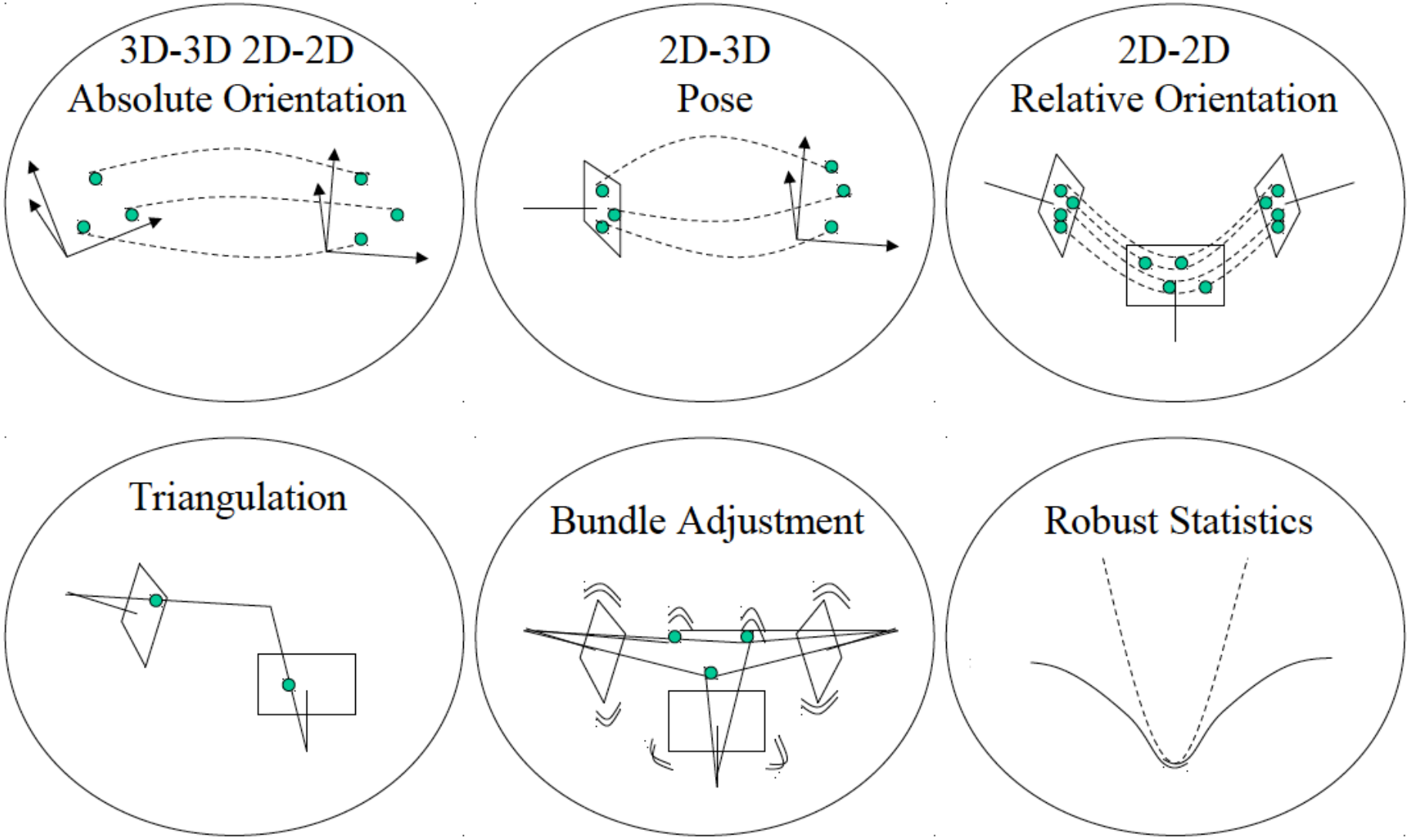
- Do everything you can to remove outliers

- Solve minimal problems to estimate geometric entities
  - Keeps RANSAC tractable
  - Typically, expect to spend 0.01ms

- Strategically consider what variables to optimize
  - Keyframe-based designs are successful
  - Try to robustly build long feature tracks
  - Do bundle adjustment whenever possible

- Drift is inevitable, so have a plan to address it
  - Local scale correction and global pose correction when possible

Toolkit for Practical SFM



Terminology and Concepts

Homogeneous coordinates

- Converting to homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

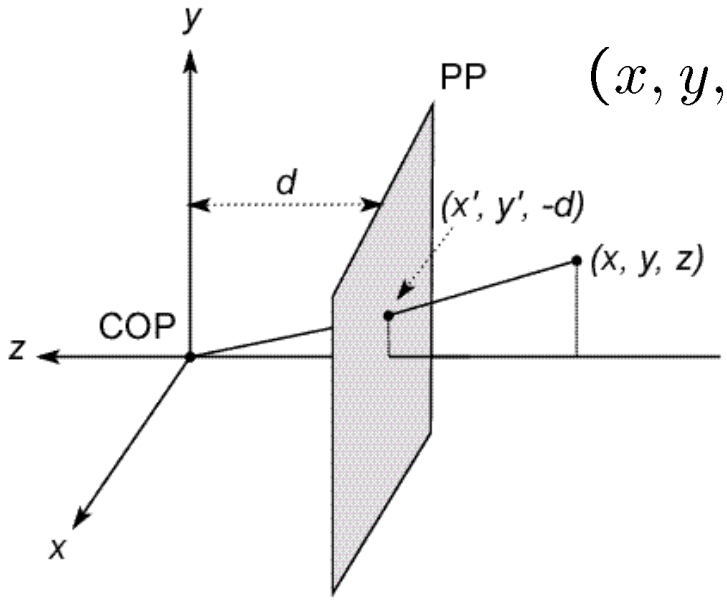
Homogeneous 2D point Homogeneous 3D point

- Converting from homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

- $(x, y, w)^\top$ and $(kx, ky, kw)^\top$ are the same point.

Modeling projection



$$(x, y, z) \rightarrow (-d\frac{x}{z}, \ -d\frac{y}{z})$$

- A matrix multiplication using homogeneous coordinates

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \end{bmatrix} \rightarrow (-d\frac{x}{z}, \ -d\frac{y}{z})$$

From image plane to pixel coordinates

$$\begin{bmatrix} -d & 0 & 0 \\ 0 & -d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\mathbf{K}$ (intrinsics)

Projection (converts from 3D rays in camera coordinate system to pixel coordinates)

In general, $\mathbf{K} = \begin{bmatrix} -d & s & c_x \\ 0 & -d & c_y \\ 0 & 0 & 1 \end{bmatrix}$ (upper triangular matrix)

α : **aspect ratio** (1 unless pixels are not square)

s : **skew** (0 unless pixels are shaped like rhombi/parallelograms)

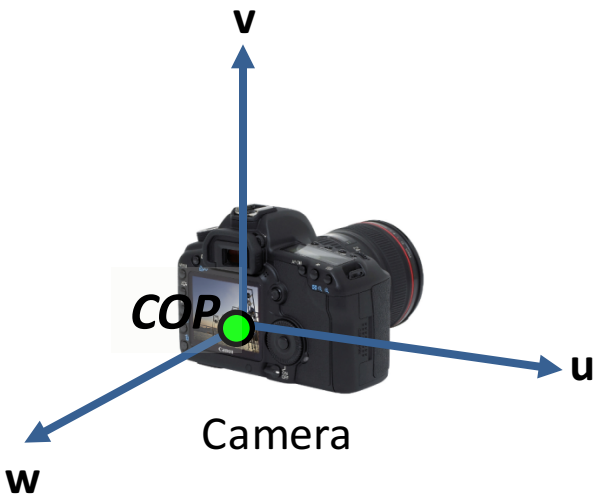
(c_x, c_y) : **principal point** ((0,0) unless optical axis doesn't intersect projection plane at origin)

Camera parameters

- How can we mathematically describe a camera?

- We need to describe its internal parameters
- We need to describe its *pose* in the world

A Tale of Two Coordinate Systems



Camera

"The World"

Two important coordinate systems:

1. *World* coordinate system
2. *Camera* coordinate system

Camera parameters

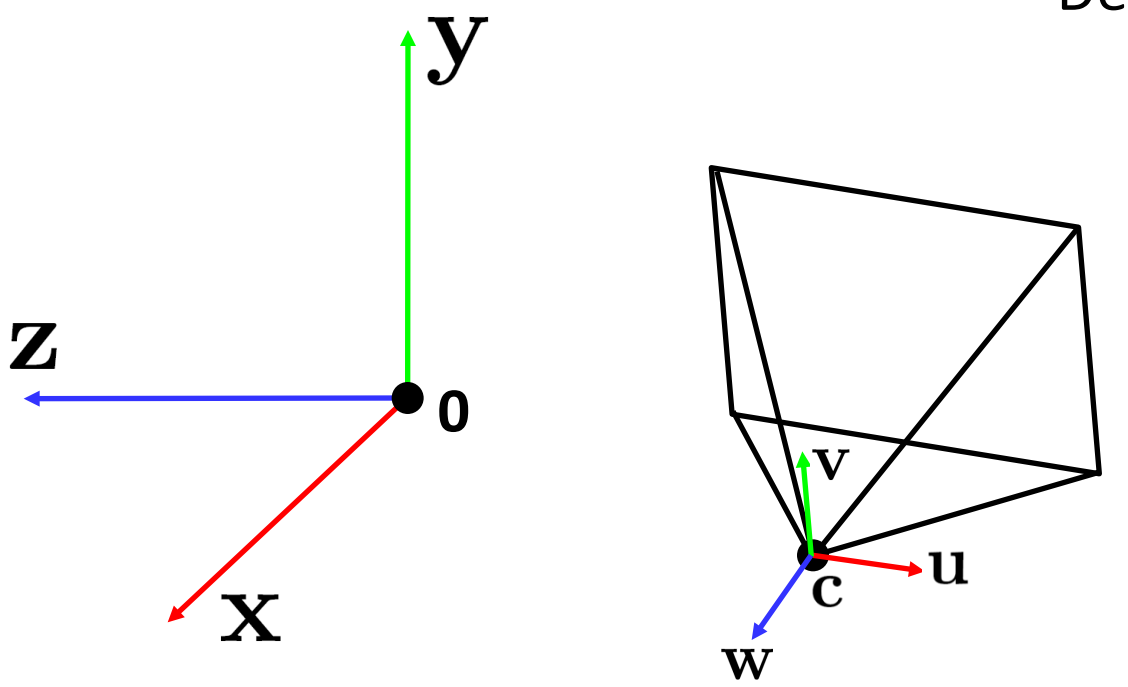
- To project a point (*x*,*y*,*z*) in *world* coordinates into an image
- First transform (*x*,*y*,*z*) into *camera* coordinates
- Need to know
  - Camera position (in world coordinates)
  - Camera orientation (in world coordinates)
- Then project into the image plane
  - Need to know camera *intrinsics*
- These can all be described with matrices.

Projection matrix

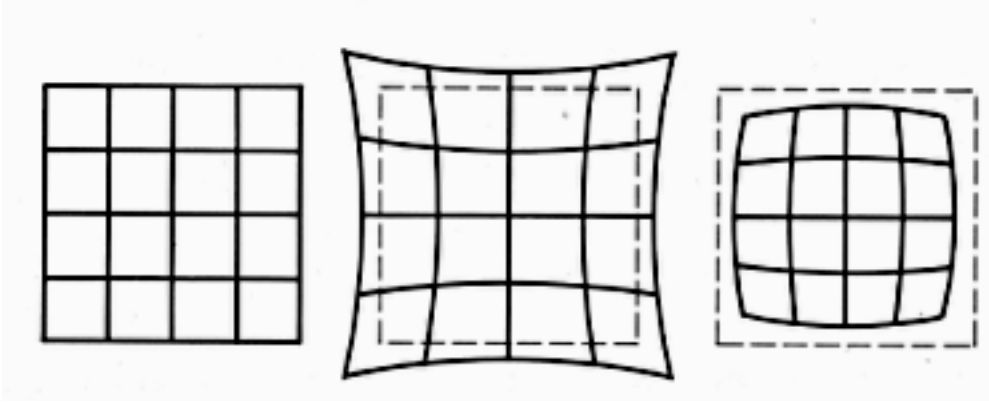
$$\Pi = \mathbf{K}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}\begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \;\; 0 \;\; 0 & 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{R} & -\mathbf{Rc} \end{bmatrix}$$

intrinsics projection rotation translation

Denote this by **t**



Distortion



No distortion Pin cushion Barrel

- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens

from Helmut Dersch

Modeling distortion

Project $(\hat{x}, \hat{y}, \hat{z})$ to "normalized" image coordinates

$$x'_n = \hat{x}/\hat{z}$$
$$y'_n = \hat{y}/\hat{z}$$

$$r^2 = {x'_n}^2 + {y'_n}^2$$

Apply radial distortion

$$x'_d = x'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$
$$y'_d = y'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$

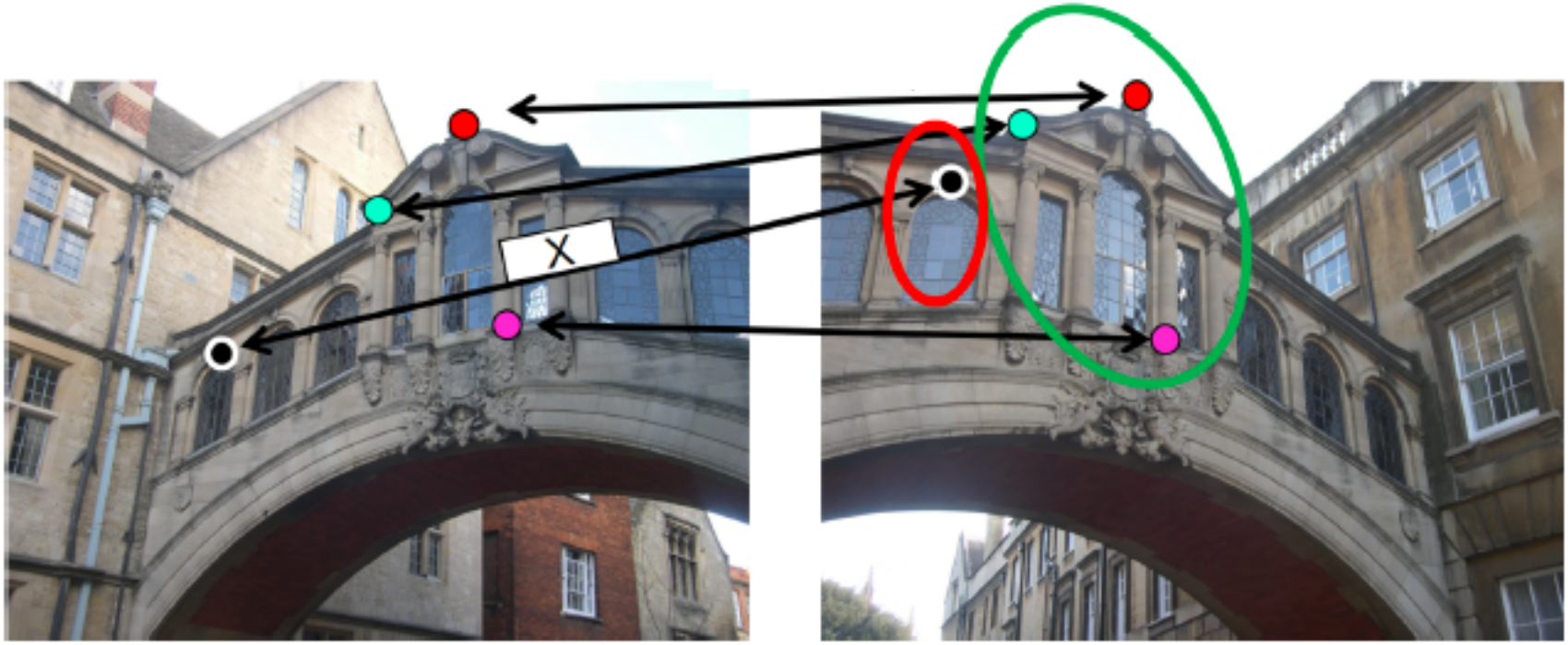
Apply focal length translate image center

$$x' = f x'_d + x_c$$
$$y' = f y'_d + y_c$$

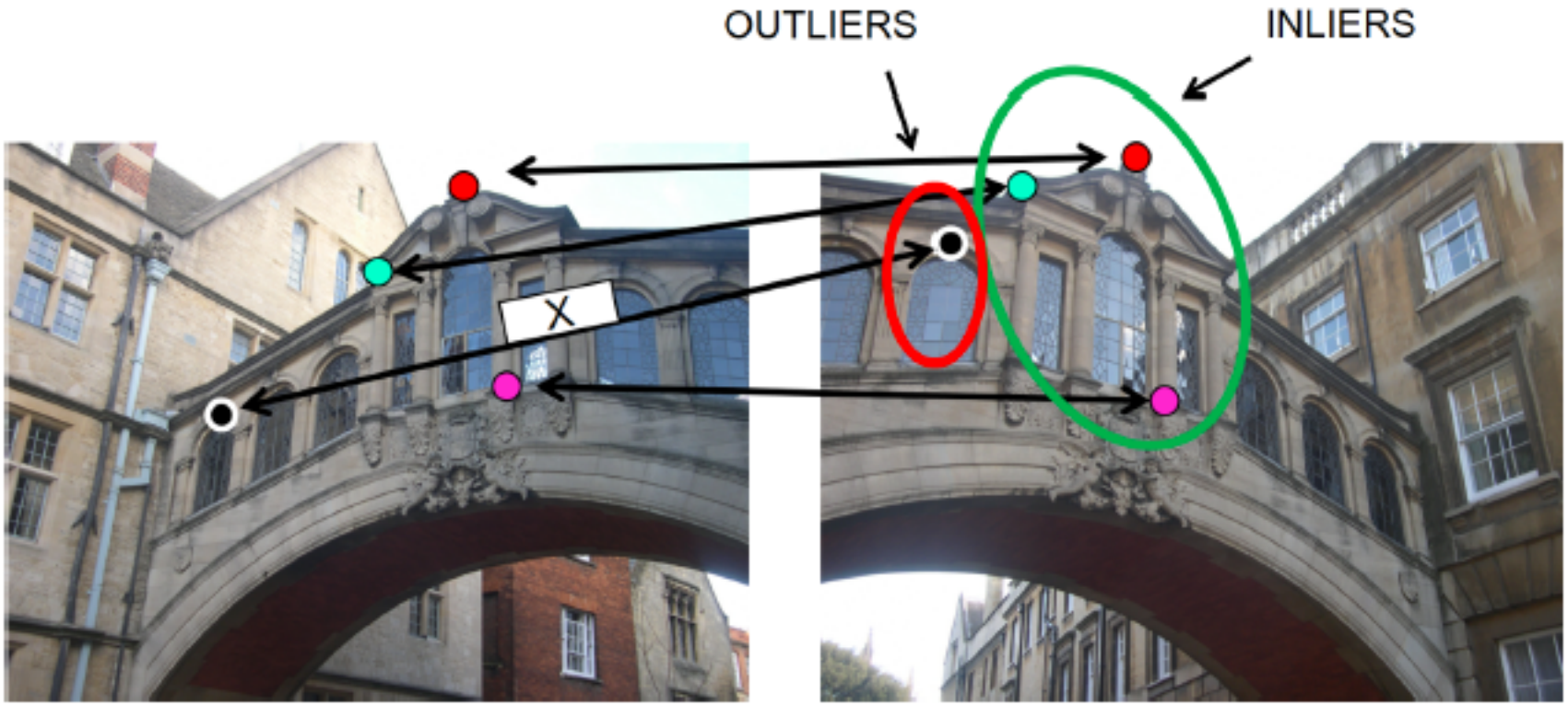
- To model lens distortion
  - Use above projection operation instead of standard projection matrix multiplication

Some “Practical” Steps

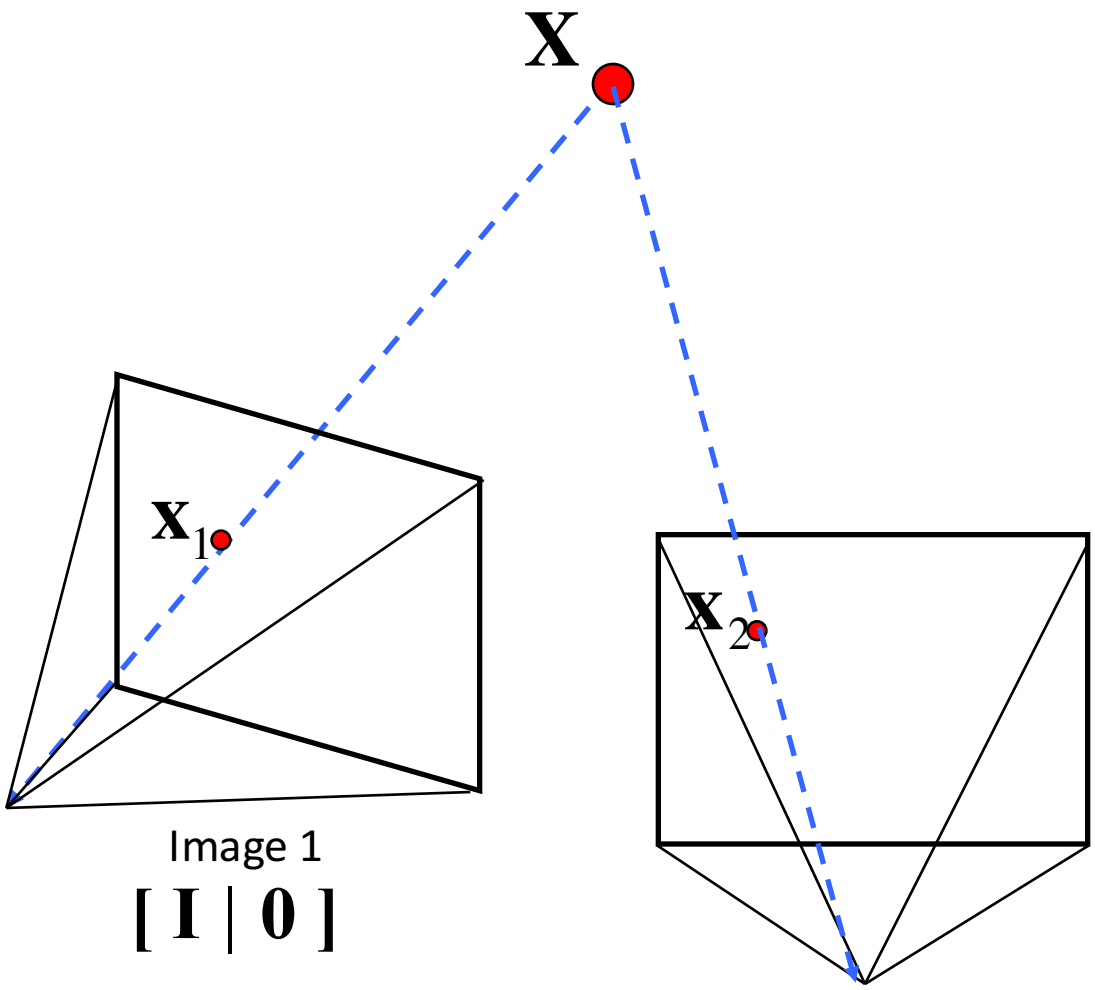
Feature matching



Feature matching



Fundamental Matrix



$$\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$$

$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$

$$\mathbf{F} = \mathbf{K}_2^{-\top} \mathbf{E} \mathbf{K}_1^{-1}$$

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$$

Estimating F: Linear Method

- The fundamental matrix F is defined by

$$\mathbf{x}'^{\mathsf{T}}\mathbf{F}\mathbf{x} = 0$$

for any pair of matches x and x' in two images.

- Let **x** = $(u,v,1)^{\mathsf{T}}$ and **x'** = $(u',v',1)^{\mathsf{T}}$, $\mathbf{F} = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix}$
- Each match gives a linear equation:

$$uu'f_{11} + vu'f_{12} + u'f_{13} + uv'f_{21} + vv'f_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0$$

Direct Linear Transform Method

Given *n* point correspondences, set up a system of equations:

$$\begin{pmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

- In reality, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$, using SVD.

Enforcing Rank Condition after DLT

- **F** should have rank 2 ($\mathbf{F} = \mathbf{K}_2^{-\top}\mathbf{E}\mathbf{K}_1^{-1}$, $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$)

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

$$\mathbf{t} \times \tilde{\mathbf{p}} = [\mathbf{t}]_\times \tilde{\mathbf{p}}$$

What is the rank of $[\mathbf{t}]_x$?

Enforcing Rank Condition after DLT

- **F** should have rank 2 ($\mathbf{F} = \mathbf{K}_2^{-\top}\mathbf{E}\mathbf{K}_1^{-1}$, $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$)

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

$$\mathbf{t} \times \tilde{\mathbf{p}} = [\mathbf{t}]_\times \tilde{\mathbf{p}}$$

What is the rank of $[\mathbf{t}]_x$?

Rank 2, since **t** is a null vector of $[\mathbf{t}]_x$
$[\mathbf{t}]_x\mathbf{t} = \mathbf{t} \times \mathbf{t} = \mathbf{0}$

Enforcing Rank Condition after DLT

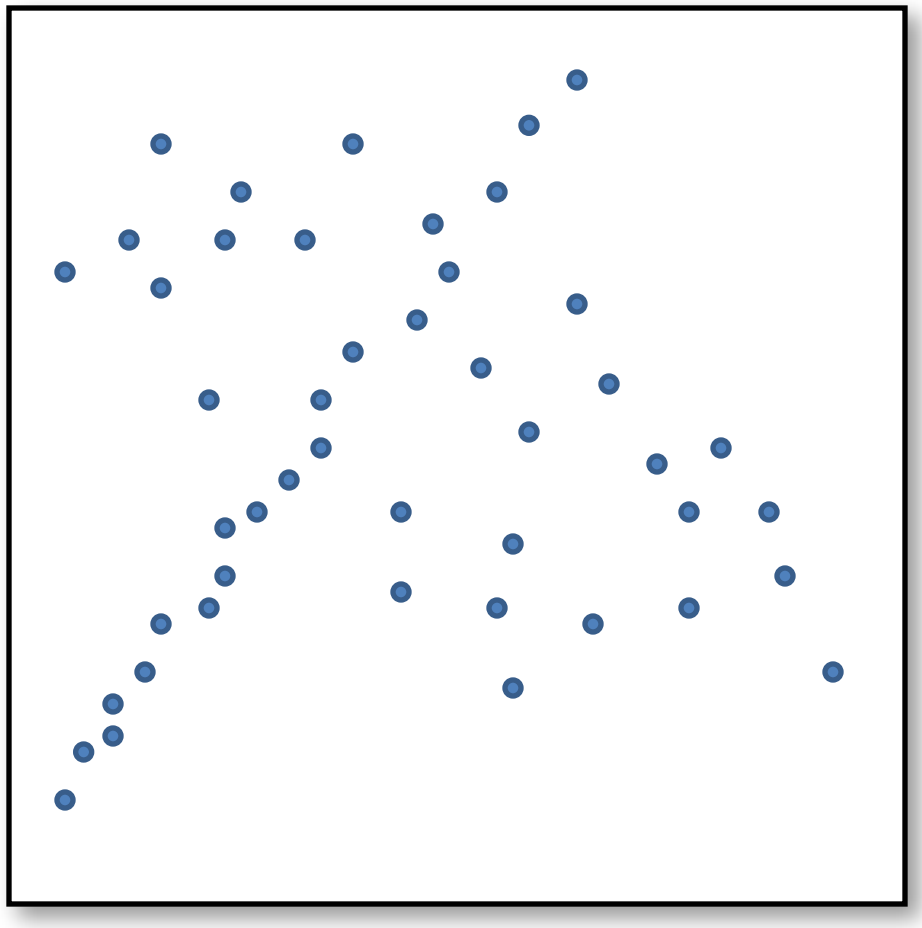
- **F** should have rank 2 ($\mathbf{F} = \mathbf{K}_2^{-\top}\mathbf{E}\mathbf{K}_1^{-1}$, $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$)
- To enforce that **F** is of rank 2, **F** is replaced by the closest **F'** that obeys the rank constraint.

- This is achieved by SVD. Let $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}$, where

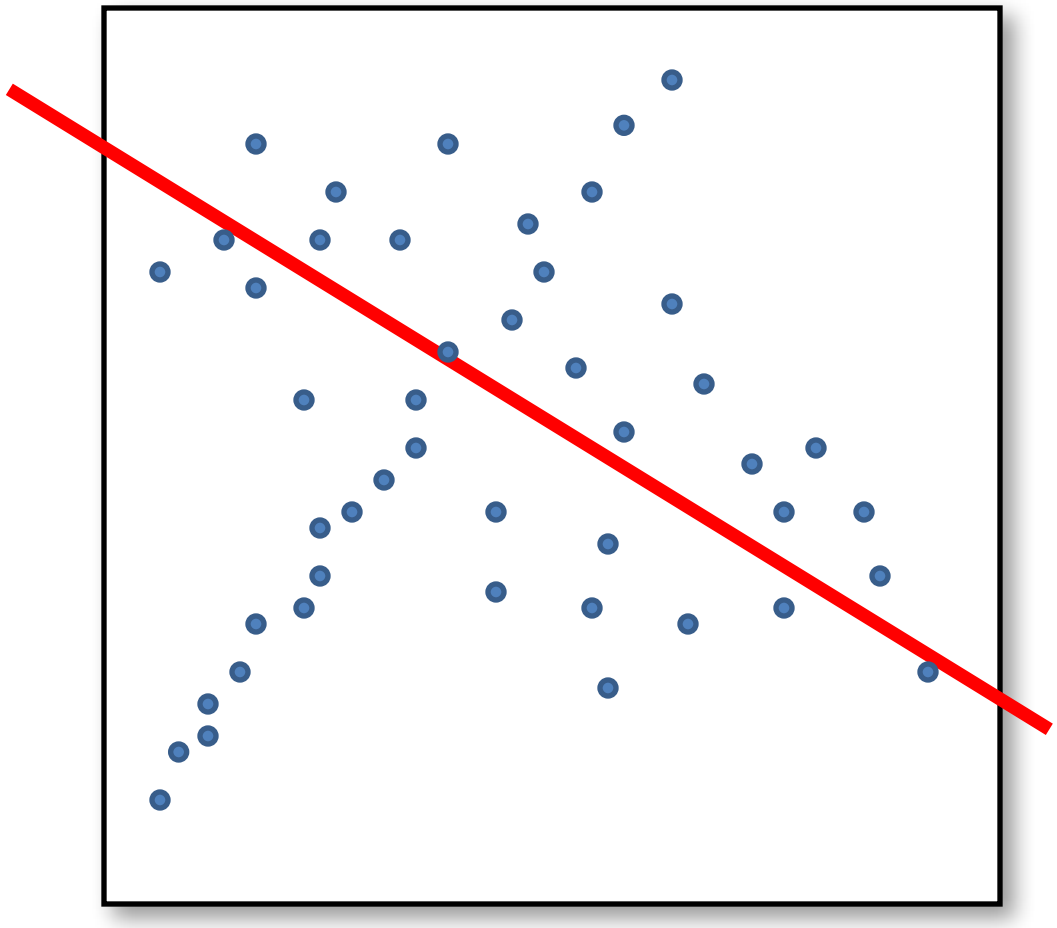
$$\Sigma = \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{pmatrix}. \quad \text{Let } \Sigma' = \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

then $\mathbf{F}' = \mathbf{U}\Sigma'\mathbf{V}$ is the solution.

RANSAC: Counting Inliers

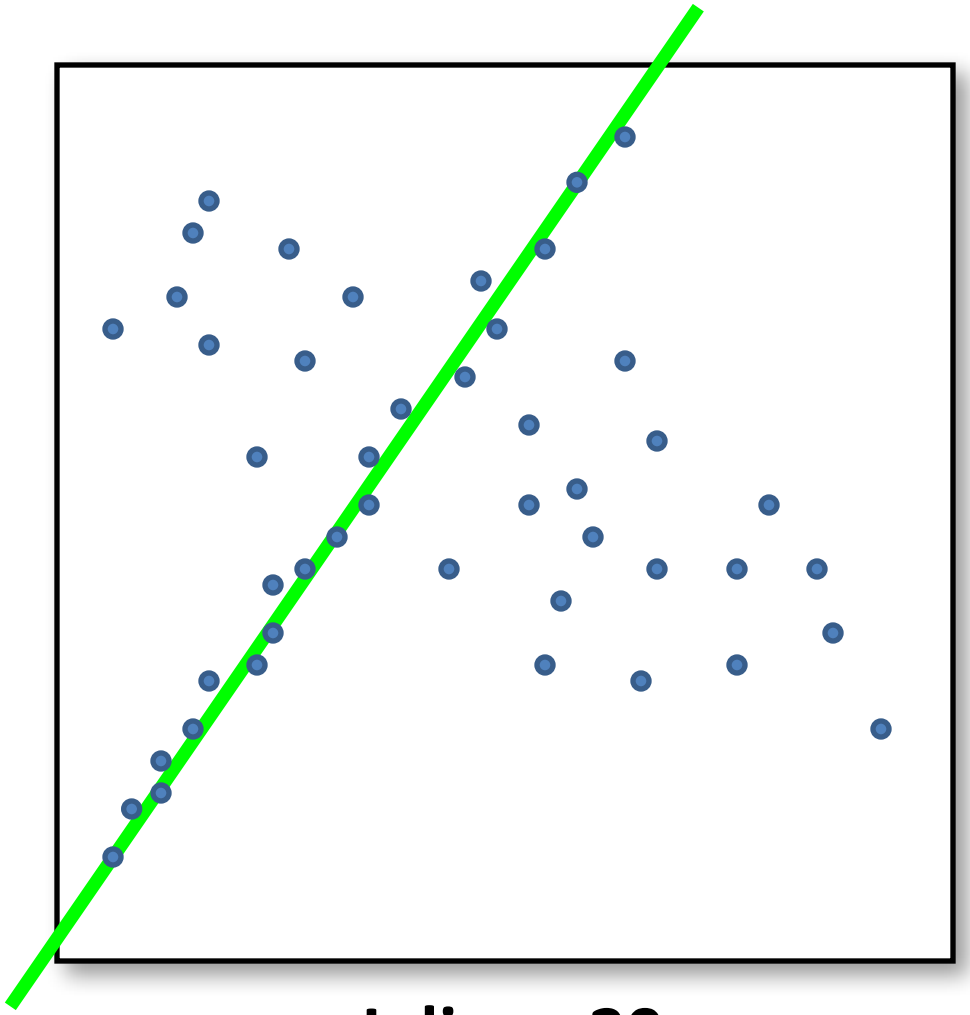


RANSAC: Counting Inliers



Inliers: 3

RANSAC: Counting Inliers



Inliers: 20

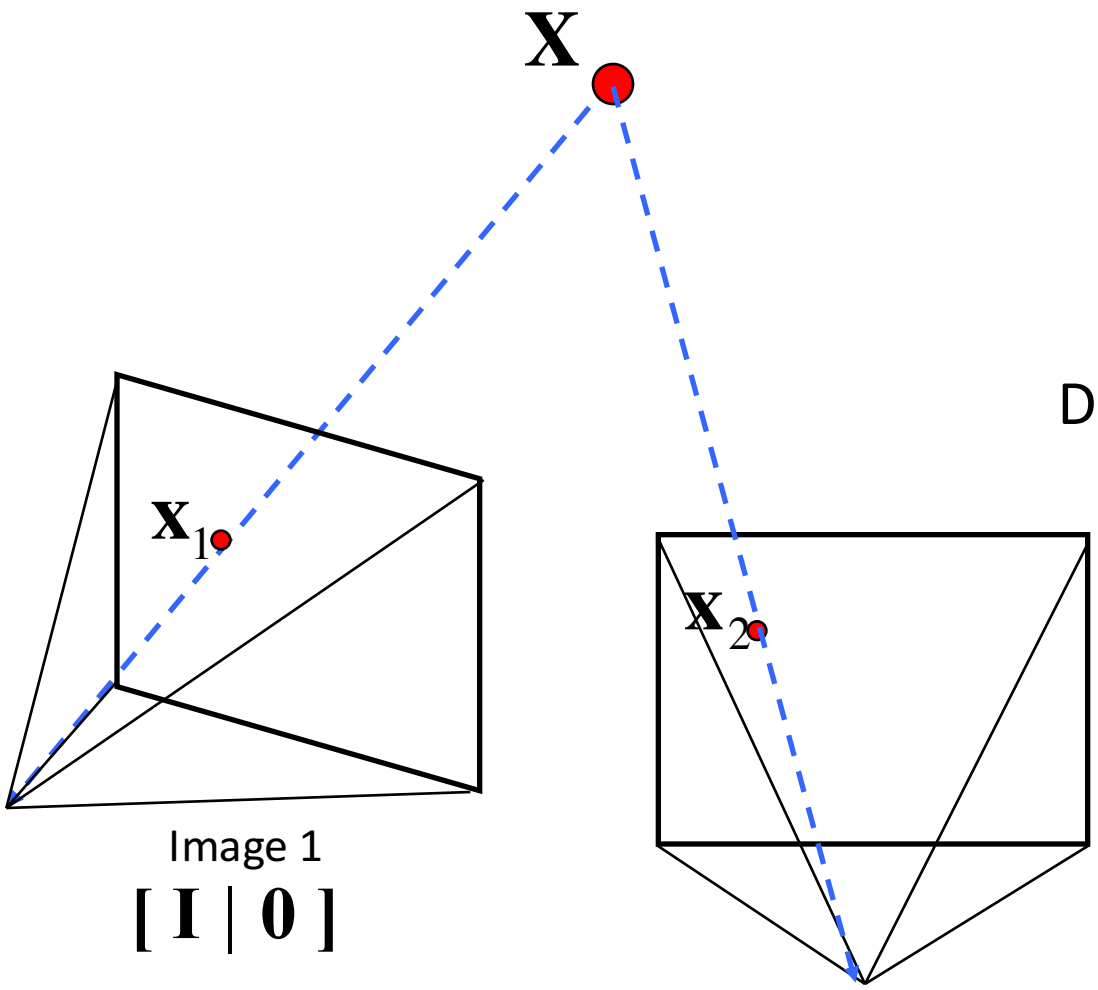
How do we find the best line?

- Unlike least-squares, no simple closed-form solution
- Hypothesize-and-test
  - Try out many lines, keep the best one
  - RANSAC: Random Sample Consensus
- Number of samples depends on
  - Outlier ratio
  - Probability of correct answer
  - Model size

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size to fit a model
  2. Fit a model (say, line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model with the largest set of inliers

Fundamental Matrix

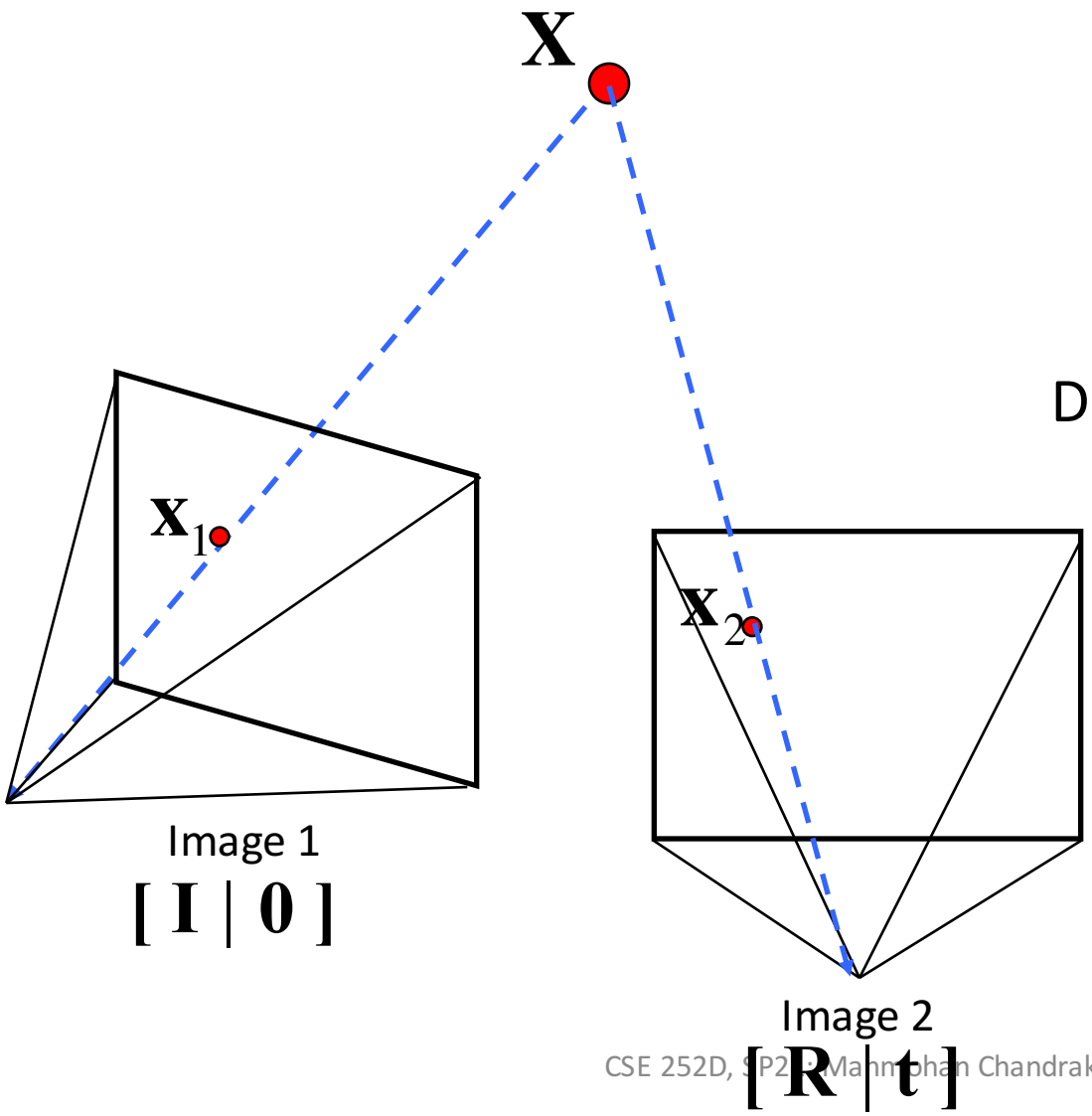


$$\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$$

$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$

Degrees of freedom for F: 7

Fundamental Matrix



$$\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$$

$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$

Degrees of freedom for F: 7

So, 7 points suffice to find F, but use 8 for linear method

RANSAC to Estimate Fundamental Matrix

- For *N* times
  - Pick 8 points
  - Compute a solution for $\mathbf{F}$ using these 8 points
  - Count number of inliers with $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2$ close to 0
- Pick the one with the largest number of inliers

RANSAC

- Adaptively determine number of iterations based on outlier proportion

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}$$

Sample size	Proportion of outliers ϵ						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Values of N for p = 0.99

Motion from correspondences

- Use 8-point algorithm to estimate F
- Get E from F:

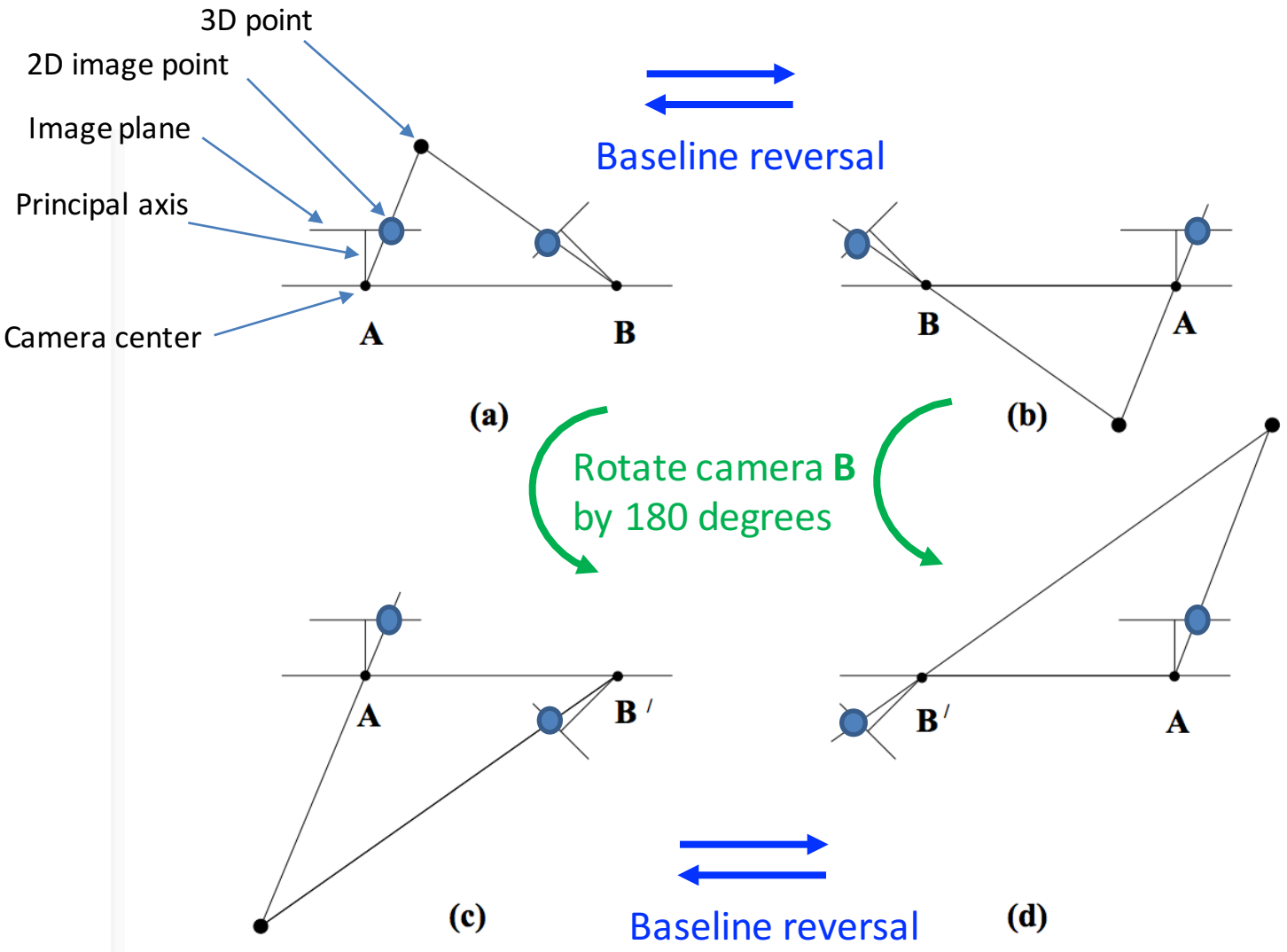
$$\mathbf{F} = \mathbf{K}_2^{-\top}\mathbf{E}\mathbf{K}_1^{-1}$$

$$\mathbf{E} = \mathbf{K}_2^{\top}\mathbf{F}\mathbf{K}_1$$

- Decompose E into skew-symmetric and rotation matrices:

$$\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R}$$

Four Possible Solutions



Bundle adjustment

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

indicator variable: whether point *i* visible in image *j*

predicted image location

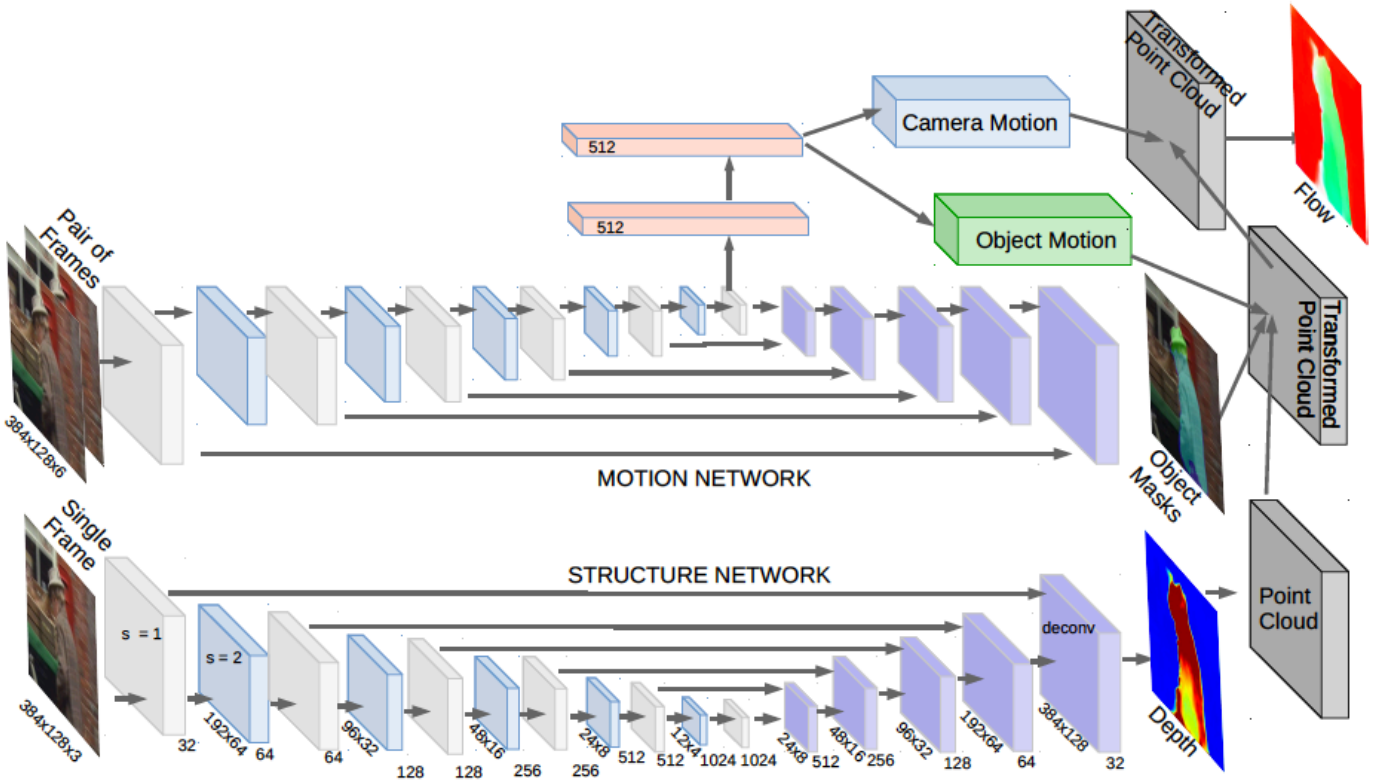
observed image location

  - Optimized with non-linear least squares
  - Levenberg-Marquardt is a popular choice
- Practical challenges?
  - Initialization
  - Outliers

Learning Structure and Motion

Typical Way to Learn Structure and Motion

- Estimate depths (convert to 3D points given calibration) in frame t
- Estimate motion from frame t to t+1 for background and objects
- Project 3D points to frame t+1 using the estimated motions
- Use a consistency condition to declare matches as good



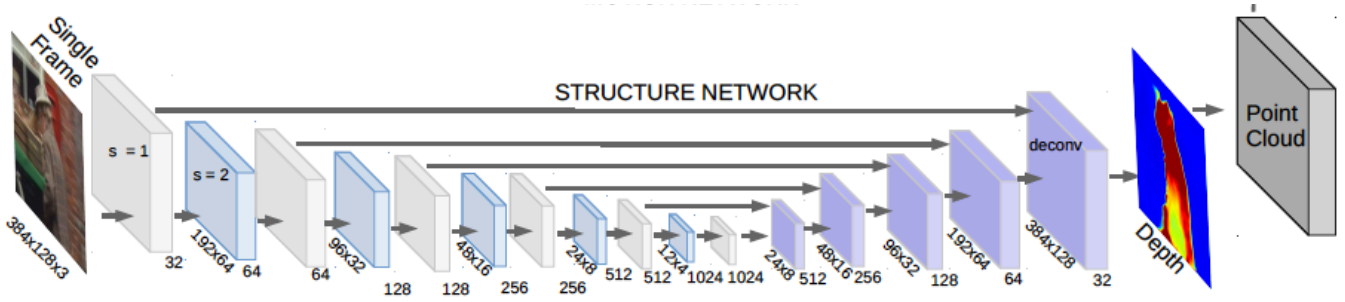
[SFM-Net, ICCV 2017]

Estimate Depth in an Image

- Depth in frame t : d_t , camera pose : $\{R_t^c, t_t^c\}$, K object motions : $\{R_t^k, t_t^k\}$

- Standard encoder-decoder for depth estimation
- For pixel (x_t^i, y_t^i), with camera intrinsics (c_x, c_y, f), 3D points are

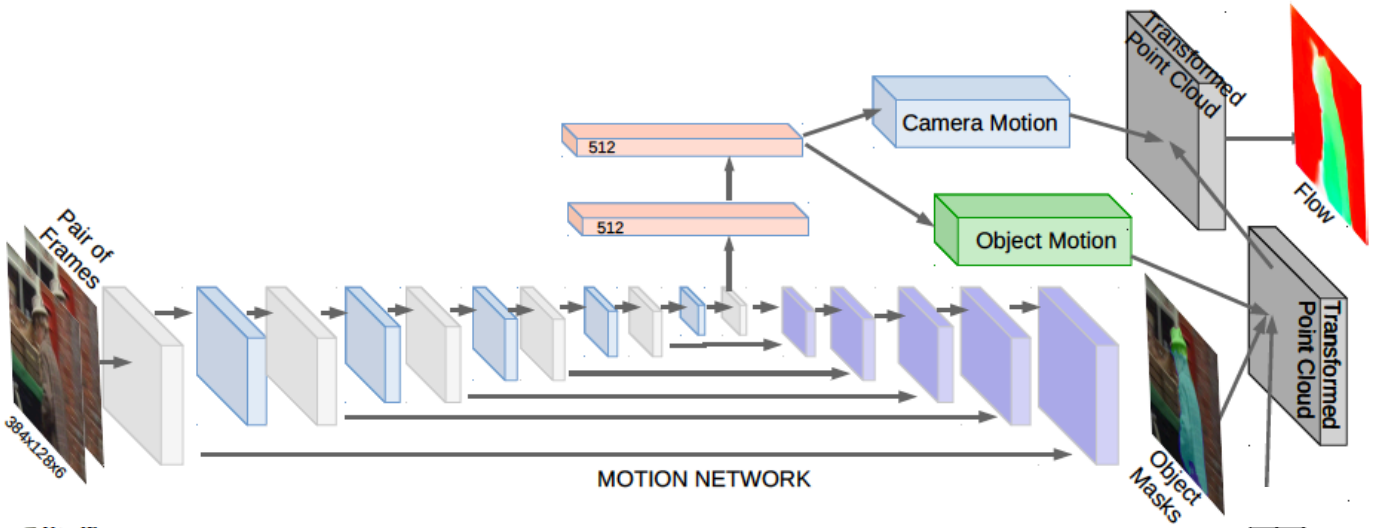
$$\mathbf{X}_t^i = \begin{bmatrix} X_t^i \\ Y_t^i \\ Z_t^i \end{bmatrix} = \frac{d_t^i}{f} \begin{bmatrix} \frac{x_t^i}{w} - c_x \\ \frac{y_t^i}{h} - c_y \\ f \end{bmatrix}$$



[SFM-Net, ICCV 2017]

Predict Motion across Frames

- Motion network takes frames t and t+1 as input
- Two fully-connected layers to predict camera and K object motions
- Transposed convolution on same embedding to get motion masks
- Same pixel can belong to multiple motions (articulations), $\mathbf{m}_t^k \in [0,1]^{(h \times w)}$



[SFM-Net, ICCV 2017]

Estimate Displacement in Image

- Apply object transformations on 3D points in frame t:

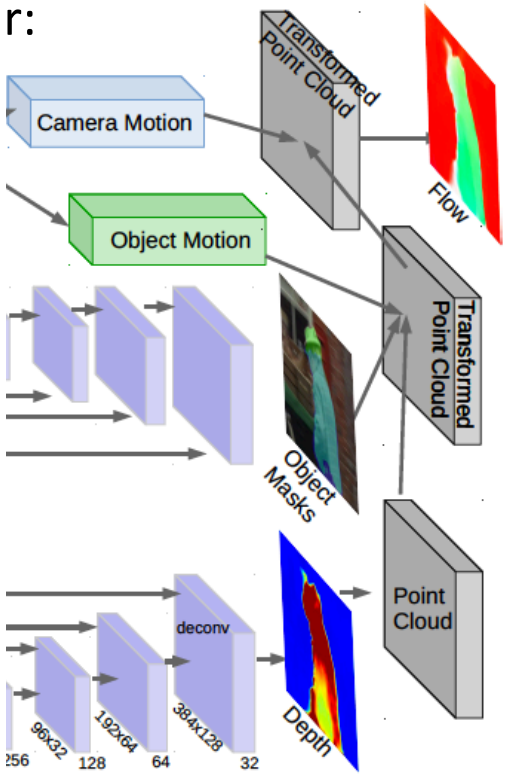
$$\mathbf{X}'_t = \mathbf{X}_t + \sum_{k=1}^{\mathrm{K}} \mathrm{m}_t^k(i)(R_t^k(\mathbf{X}_t - p_k) + t_t^k - \mathbf{X}_t)$$

- Apply camera transformations on 3D points: $\mathbf{X}''_t = R_t^c(\mathbf{X}'_t - p_t^c) + t_t^c$

- Project to frame t+1 and compute flow vector:

$$\begin{bmatrix} \frac{x_{t+1}^i}{w} \\ \frac{y_{t+1}^i}{h} \end{bmatrix} = \frac{f}{Z''_t} \begin{bmatrix} X''_t \\ Y''_t \\ f \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}$$

$$(U_t(i), V_t(i)) = (x_{t+1}^i - x_t^i, y_{t+1}^i - y_t^i)$$



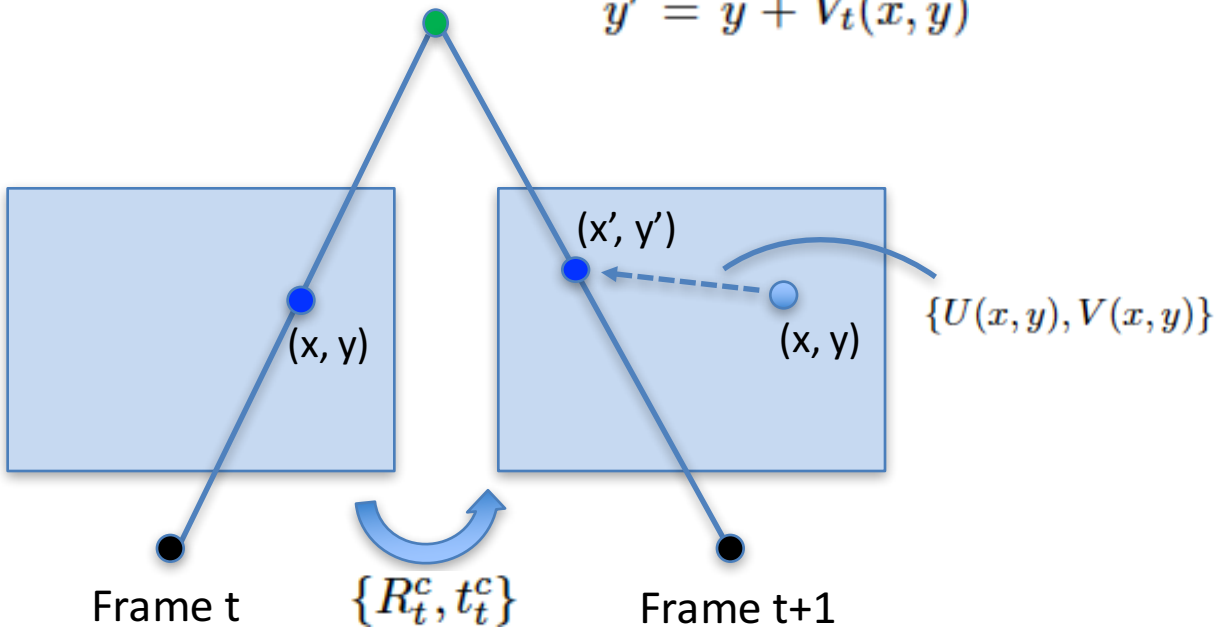
[SFM-Net, ICCV 2017]

Types of Supervision: Photoconsistency

- Image intensity at a point remains the same
- Structure and motion estimates most consistent with above
- Minimize the error :

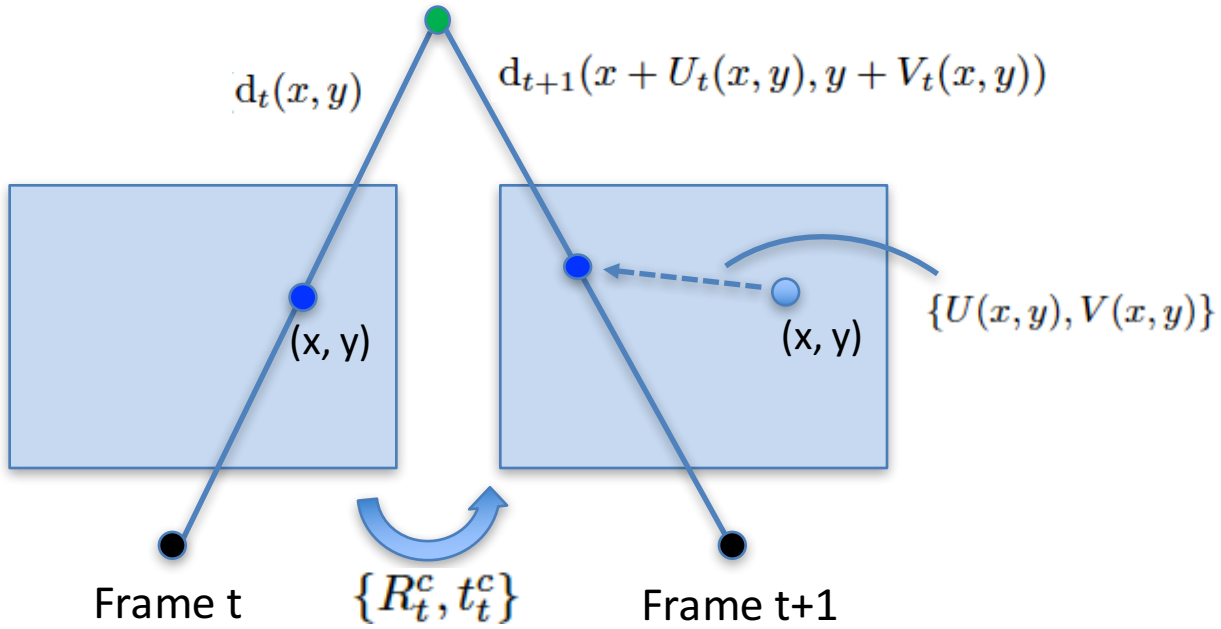
$$\sum_{x,y} \|I_t(x,y) - I_{t+1}(x',y')\|_1$$

$$x' = x + U_t(x,y)$$
$$y' = y + V_t(x,y)$$



Types of Supervision: Left-Right Consistency

- Depths of same point in two frames must be consistent with motion
- Estimate depths in frame t
- Apply estimated motion, W_t, to 3D points
- Estimate depth in frame t+1 at flow-displaced location
- Minimize error : $|(\mathrm{d}_t(x,y) + W_t(x,y)) - \mathrm{d}_{t+1}(x + U_t(x,y), y + V_t(x,y))|$



Types of Supervision: Ground Truth

- In some cases, ground truth depth is known (LIDAR, Kinect) at some points

$$\sum_{x,y} \text{dmask}_t^{GT}(x,y) \cdot \|\text{d}_t(x,y) - \text{d}_t^{GT}(x,y)\|_1$$

- Sometimes, ground truth motion is known (GPS, IMU)

$$R_t^{\text{err}} = \text{inv}(R_t^c) R_t^{c-GT} \;, \quad t_t^{\text{err}} = \text{inv}(R_t^c)(\text{t}_t^{c-GT} - \text{t}_t^c)$$