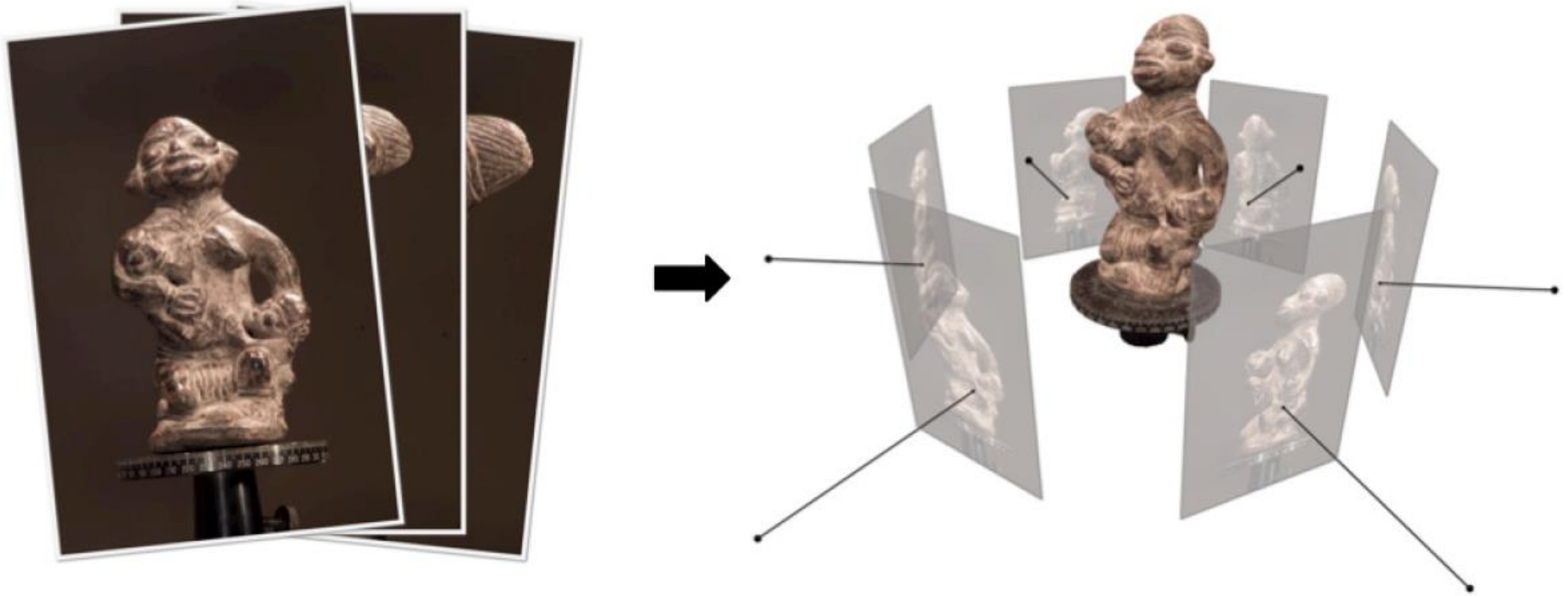
From Homography to MVSNet

Shenghua Gao

Multi-View Stereo

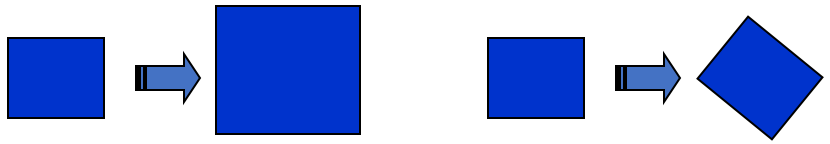
- Recovering the 3D structure of a scene from a set of calibrated images.
- We predict a depth map for each image and fuse the depth map.



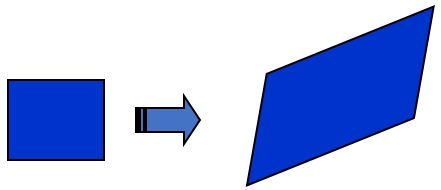
Multi-View Stereo: A Tutorial [Yasutaka Furukawa]

2D transformation models

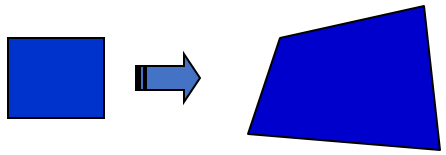
- Similarity (translation, scale, rotation)



- Affine

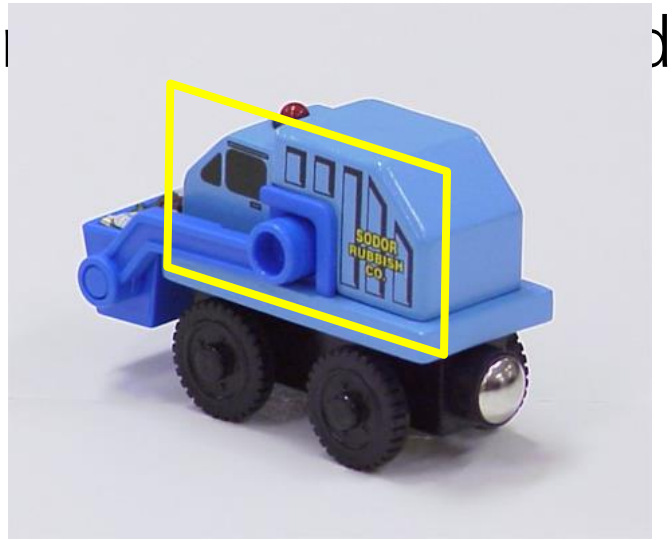


- Projective (homography)



Let’s start with affine transformations

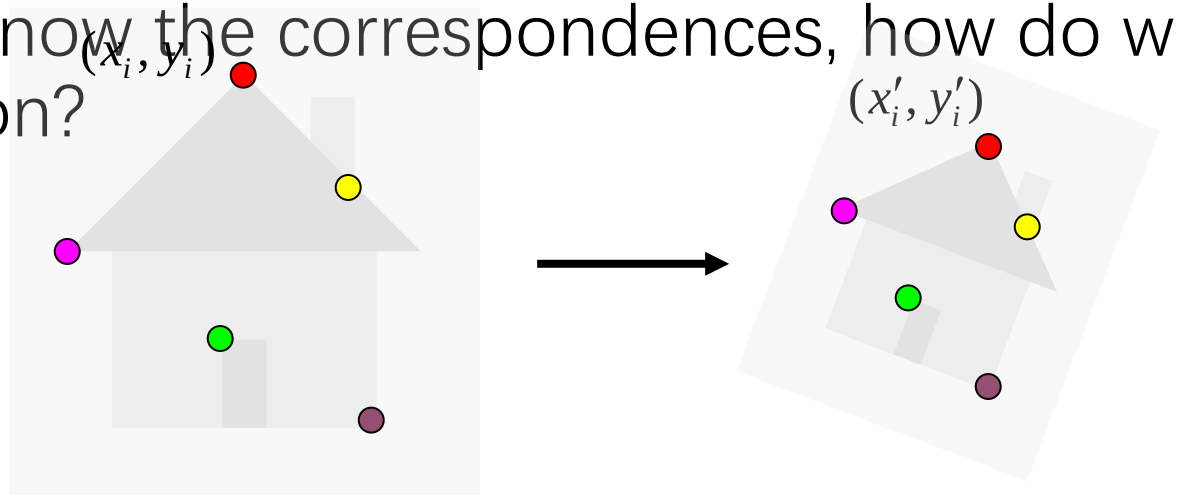
- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used for



dels

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

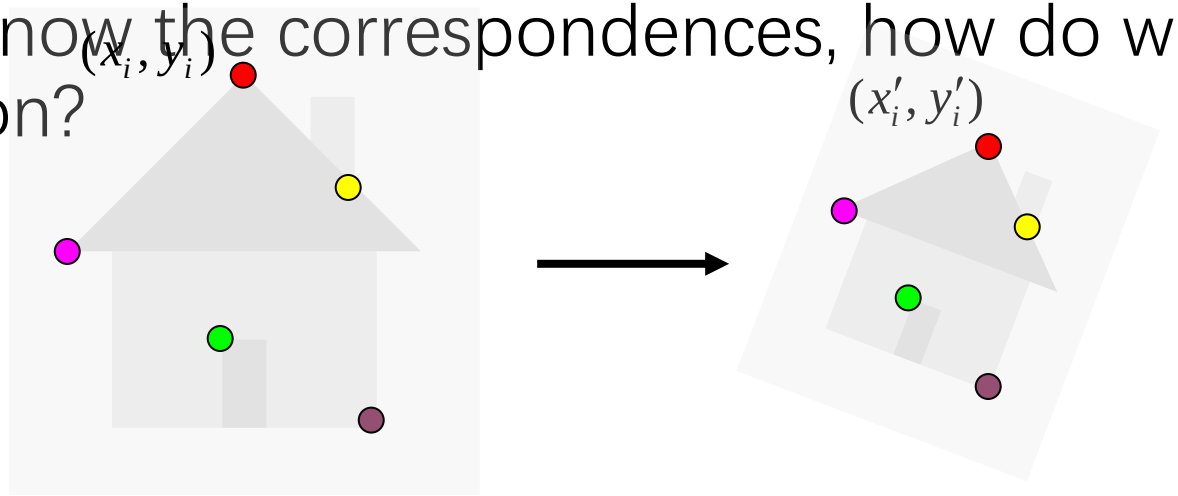
$$\mathbf{x}_i' = \mathbf{M}\mathbf{x}_i + \mathbf{t}$$

Want to find M, t to minimize

$$\sum_{i=1}^{n} \| \mathbf{x}_i' - \mathbf{M}\mathbf{x}_i - \mathbf{t} \|^2$$

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



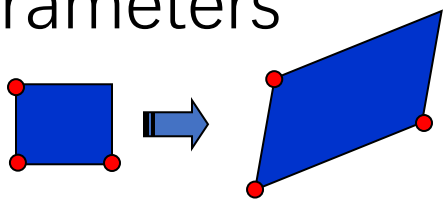
$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Fitting an affine transformation

$$\begin{bmatrix} & \cdots & & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & \cdots & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters



How about the translation only?

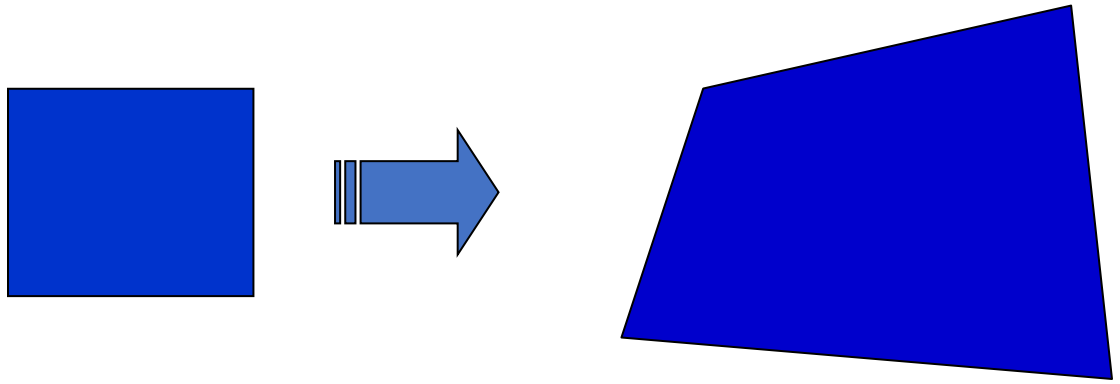
- Similarity transform

- M=I and t=?

- What's the minimum number of points needed for optimizing the objective?

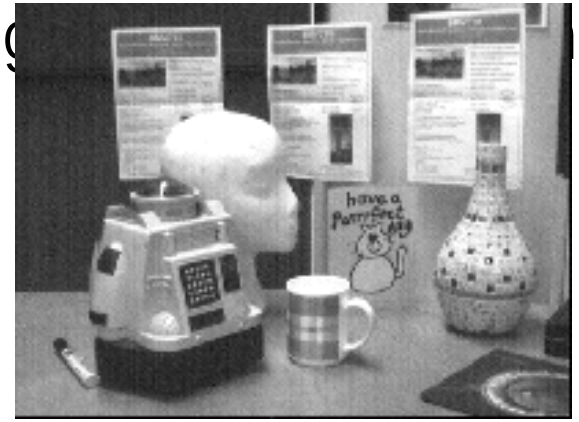
Fitting a plane projective transformation

- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)

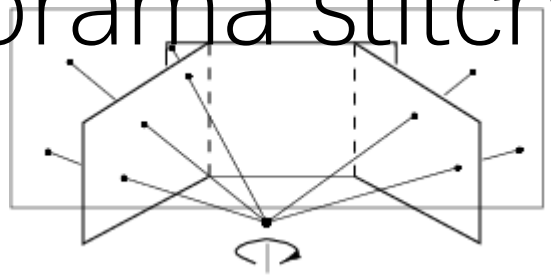


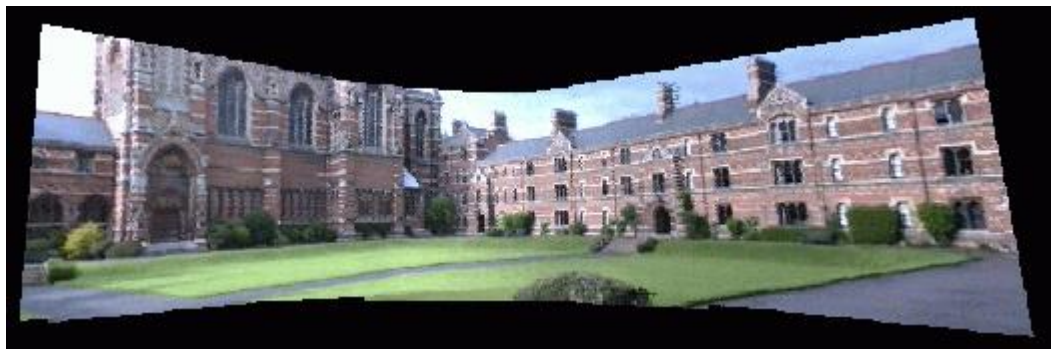
Homography

- The transforma  o  r surface

- The transforma  ag  eras that share the same cente

Application: Panorama stitching





Source: Hartley & Zisserman

Definition 2.11. Projective transformation. A planar projective transformation is a linear transformation on homogeneous 3-vectors represented by a non-singular 3×3 matrix:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \tag{2.5}$$

or more briefly, $\mathbf{x}' = \mathtt{H}\mathbf{x}$.

Note that the matrix H occurring in this equation may be changed by multiplication by an arbitrary non-zero scale factor without altering the projective transformation. Consequently we say that H is a *homogeneous* matrix, since as in the homogeneous representation of a point, only the ratio of the matrix elements is significant. There are eight independent ratios amongst the nine elements of H, and it follows that a projective transformation has eight degrees of freedom.

A projective transformation projects every figure into a projectively equivalent figure, leaving all its projective properties invariant. In the ray model of figure 2.1 a projective transformation is simply a linear transformation of $\mathbb{R}^3$.

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)

a b

Fig. 2.4. **Removing perspective distortion.** *(a) The original image with perspective distortion – the lines of the windows clearly converge at a finite point. (b) Synthesized frontal orthogonal view of the front wall. The image (a) of the wall is related via a projective transformation to the true geometry of the wall. The inverse transformation is computed by mapping the four imaged window corners to corners of an appropriately sized rectangle. The four point correspondences determine the transformation. The transformation is then applied to the whole image. Note that sections of the image of the ground are subject to a further projective distortion. This can also be removed by a projective transformation.*

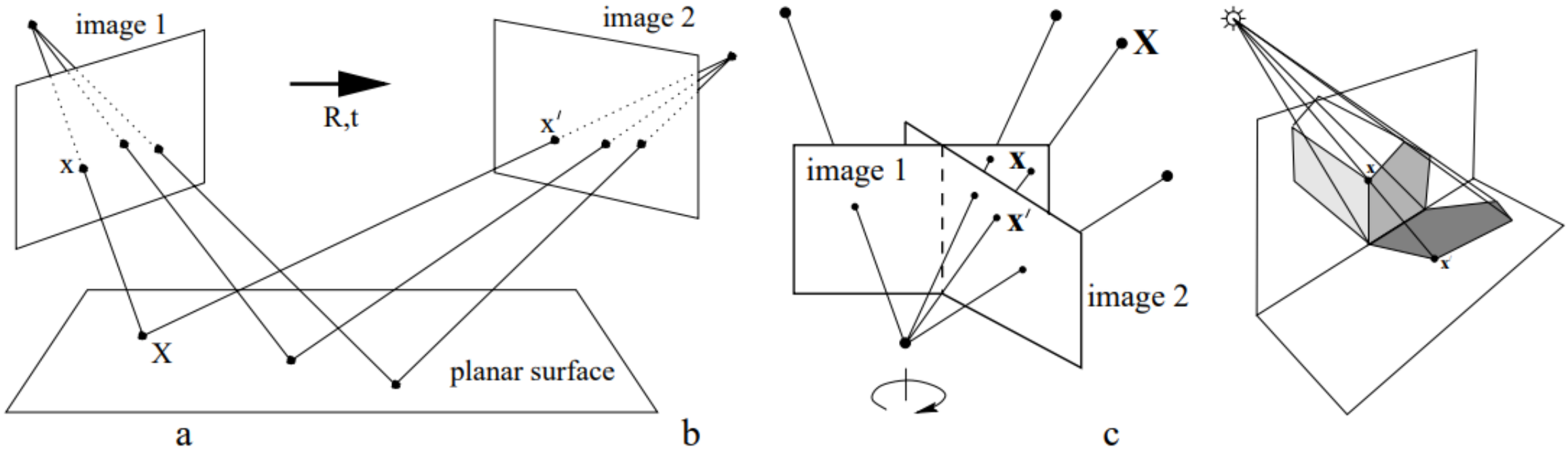
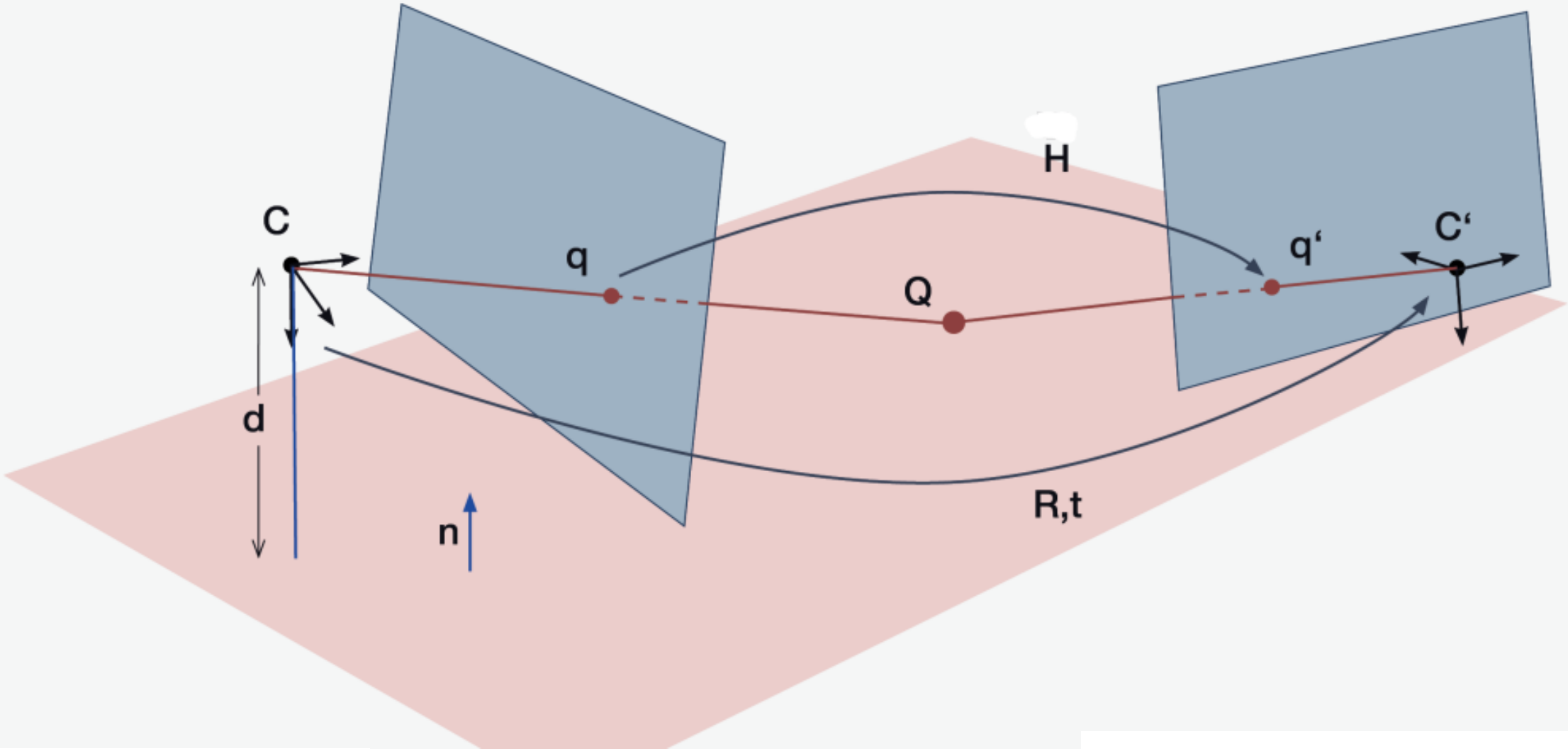
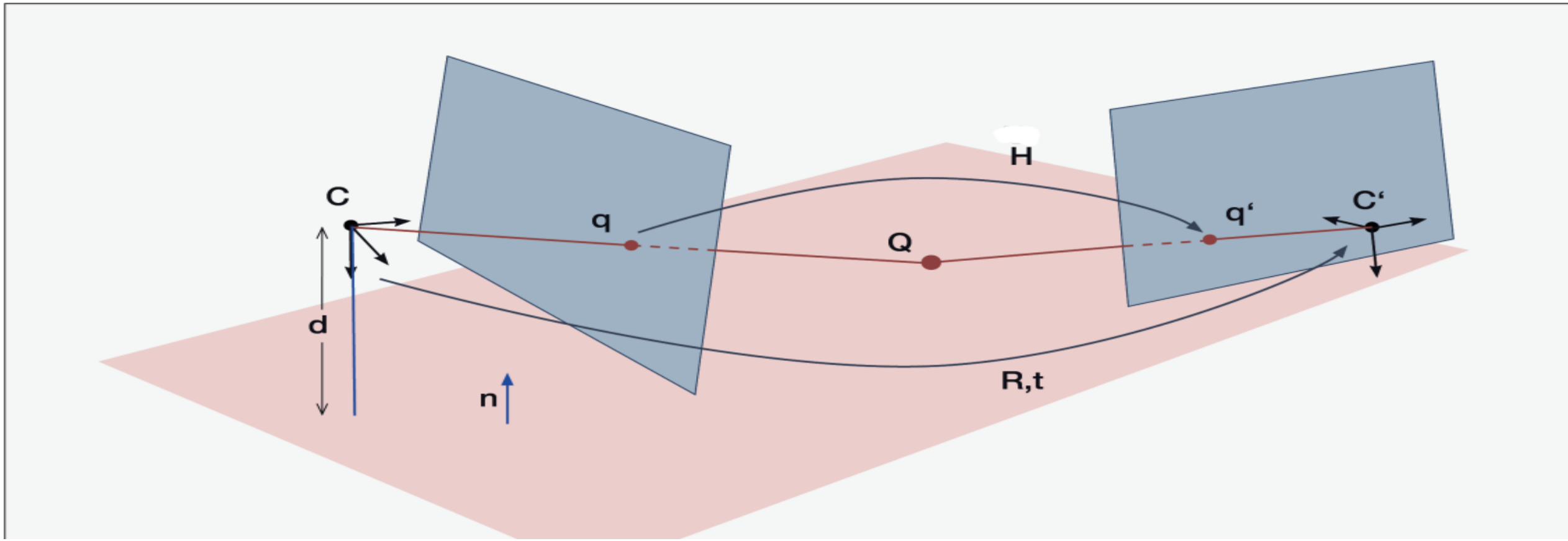


Fig. 2.5. **Examples of a projective transformation, $\mathbf{x}' = \mathtt{H}\mathbf{x}$, arising in perspective images.** *(a) The projective transformation between two images induced by a world plane (the concatenation of two projective transformations is a projective transformation); (b) The projective transformation between two images with the same camera centre (e.g. a camera rotating about its centre or a camera varying its focal length); (c) The projective transformation between the image of a plane (the end of the building) and the image of its shadow onto another plane (the ground plane). Figure (c) courtesy of Luc Van Gool.*



$$Q_2 = RQ_1 + t \qquad d = -n^T Q_1 \qquad \Rightarrow \qquad Q_2 = (R - t\frac{n^T}{d})Q_1$$



$$s_1 q_1 = K_1 Q_1$$

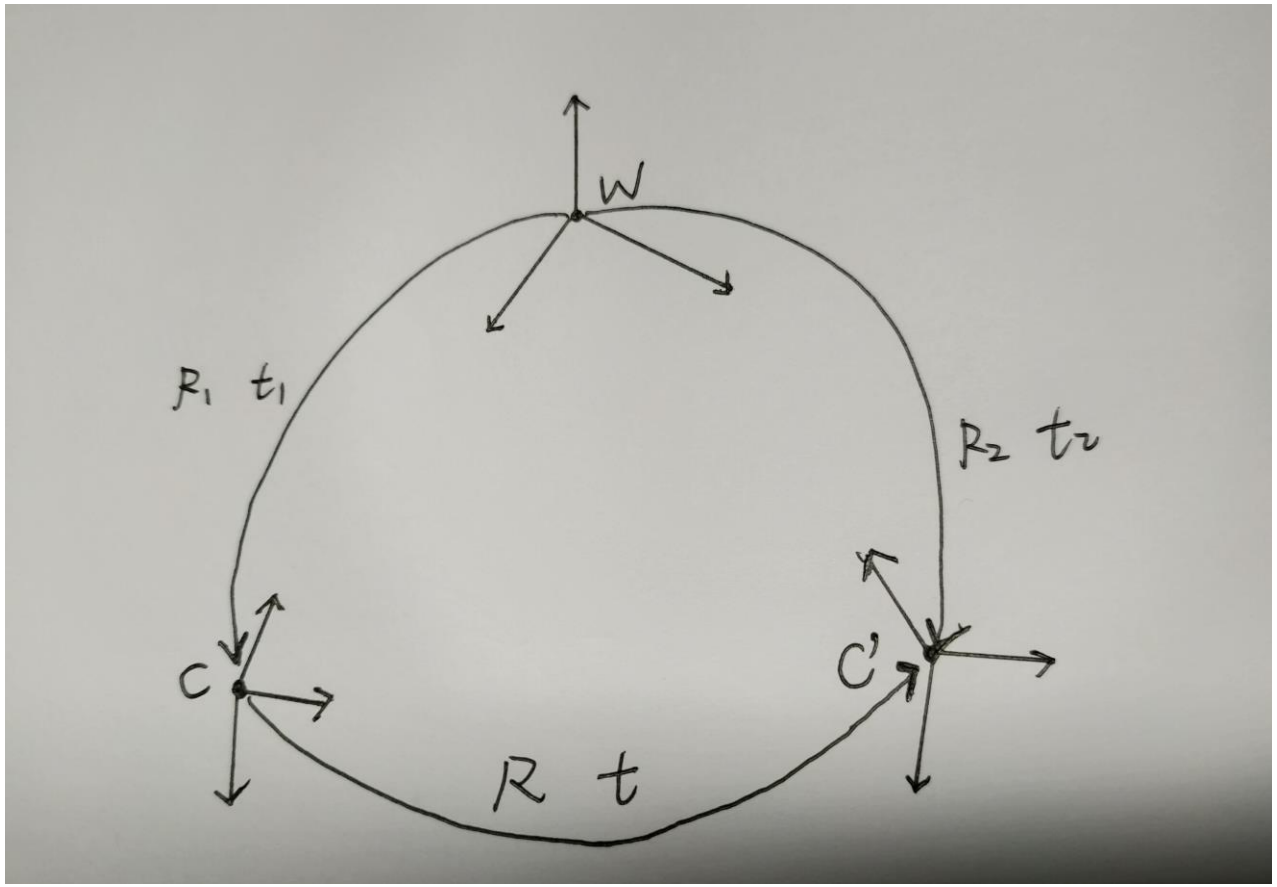
$$s_2 q_2 = K_2 Q_2$$

$$Q_2 = (R - t\frac{n^T}{d})Q_1$$

$$s_2 q_2 = K_2 (R - t\frac{n^T}{d})Q_1$$

$$q_2 = sK_2 (R - t\frac{n^T}{d})K_1^{-1} q_1$$

$$H = K_2 (R - t\frac{n^T}{d})K_1^{-1}$$



$$\begin{bmatrix} R_2 & t_2 \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}^{-1} \begin{bmatrix} R & t \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} R_1 & t_1 \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} P = P$$

$$\begin{bmatrix} R & t \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} R_2 R_1^{-1} & -R_2 R_1^{-1} t_1 + t_2 \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$$

$$R = R_2 R_1^{-1}$$

$$t = -R_2 R_1^{-1} t_1 + t_2$$

$$H = K_2 (R - t\frac{n^T}{d}) K_1^{-1}$$

$$H = K_2 (R_2 R_1^{-1} - (-R_2 R_1^{-1} t_1 + t_2)\frac{n^T}{d}) K_1^{-1}$$

$$H = K_2 R_2 (I - \frac{1}{d}(-R_1^{-1} t_1 + R_2^{-1} t_2) n^T R_1) R_1^T K_1^{-1}$$

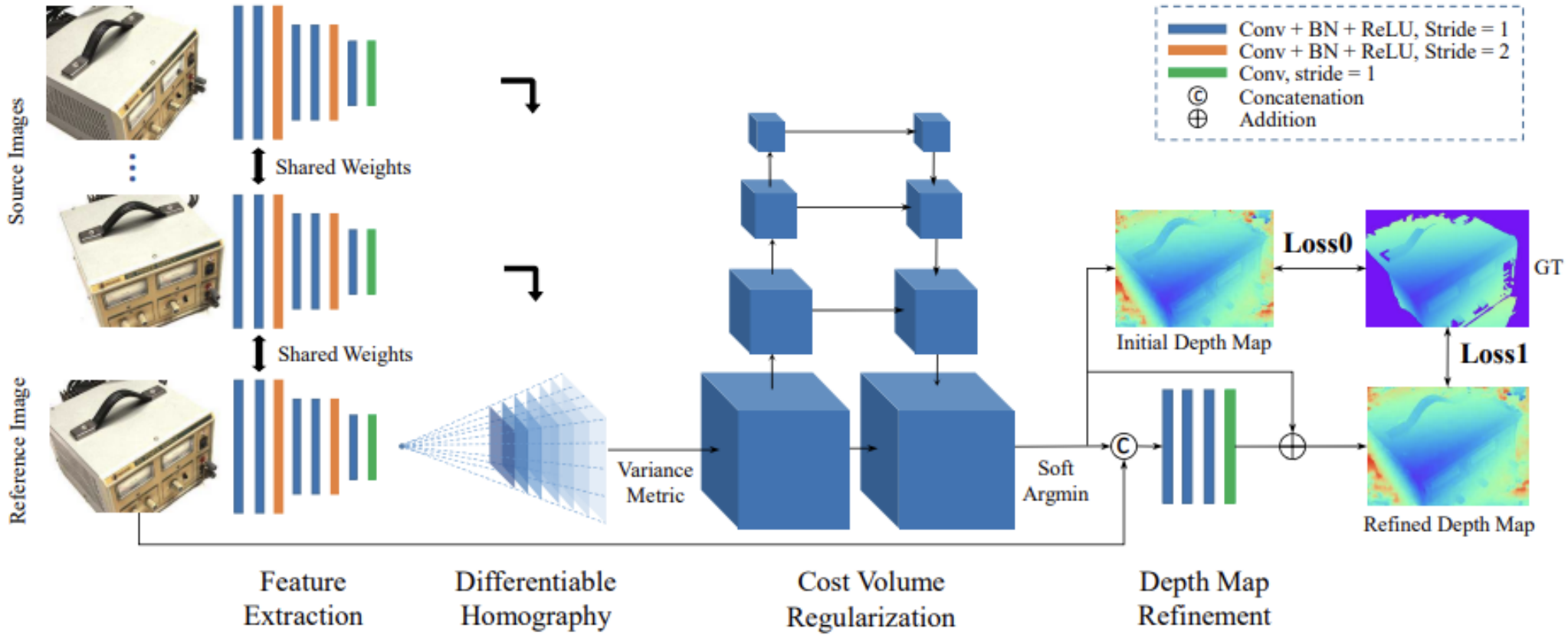


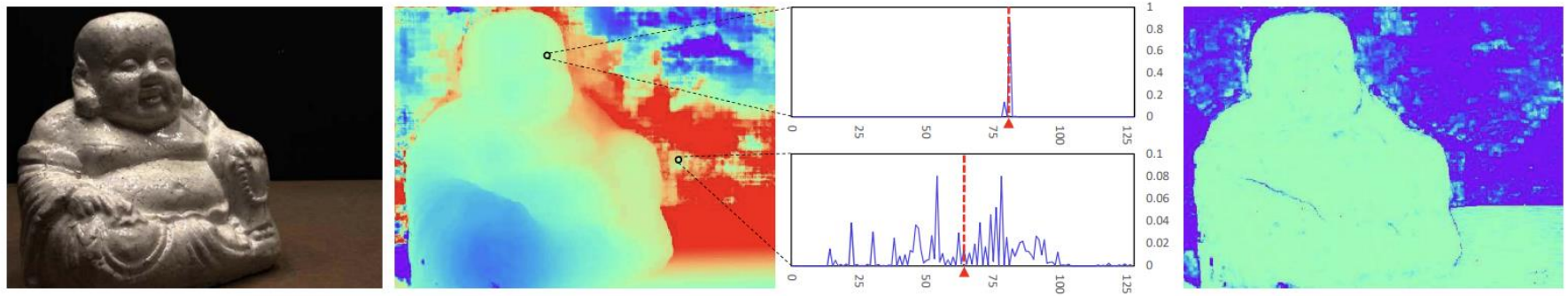
Fig. 1: The network design of MVSNet. Input images will go through the 2D feature extraction network and the differentiable homograph warping to generate the cost volume. The final depth map output is regressed from the regularized probability volume and refined with the reference image

MVSNet

Differentiable Homography All feature maps are warped into different fronto-parallel planes of the reference camera to form N feature volumes $\{\mathbf{V}_i\}_{i=1}^{N}$. The coordinate mapping from the warped feature map $\mathbf{V}_i(d)$ to $\mathbf{F}_i$ at depth d is determined by the planar transformation $\mathbf{x}' \sim \mathbf{H}_i(d) \cdot \mathbf{x}$, where '$\sim$' denotes the projective equality and $\mathbf{H}_i(d)$ the homography between the i^{th} feature map and the reference feature map at depth d. Let $\mathbf{n}_1$ be the principle axis of the reference camera, the homography is expressed by a 3×3 matrix:

$$H = K_2 R_2 (I - \frac{1}{d}(-R_1^{-1} t_1 + R_2^{-1} t_2) n^T R_1) R_1^T K_1^{-1}$$

$$\mathbf{C} = \mathcal{M}(\mathbf{V}_1, \cdots, \mathbf{V}_N) = \frac{\sum_{i=1}^{N} (\mathbf{V}_i - \overline{\mathbf{V}_i})^2}{N}$$



(a) Reference image (b) Inferred depth map (c) Probability distribution (d) Probability Map

Fig. 2: Illustrations on inferred depth map, probability distributions and probability map. (a) One reference image of *scan* 114, *DTU* dataset [1]; (b) the inferred depth map; (c) the probability distributions of an inlier pixel (top) and an outlier pixel (bottom), where the x-axis is the index of depth hypothesis, y-axis the probability and red lines the soft argmin results; (d) the probability map. As shown in (c), the outlier's distribution is scattered and results in a low probability estimation in (d)

$$\mathbf{D} = \sum_{d=d_{min}}^{d_{max}} d \times \mathbf{P}(d)$$

3.4 Loss

Losses for both the initial depth map and the refined depth map are considered. We use the mean absolute difference between the ground truth depth map and the estimated depth map as our training loss. As ground truth depth maps are not always complete in the whole image (see Sec. 4.1), we only consider those pixels with valid ground truth labels:

$$Loss = \sum_{p \in \mathbf{p}_{valid}} \underbrace{\|d(p) - \hat{d}_i(p)\|_1}_{Loss0} + \lambda \cdot \underbrace{\|d(p) - \hat{d}_r(p)\|_1}_{Loss1} \tag{4}$$

Where $\mathbf{p}_{valid}$ denotes the set of valid ground truth pixels, $d(p)$ the ground truth depth value of pixel p, $\hat{d}_i(p)$ the initial depth estimation and $\hat{d}_r(p)$ the refined depth estimation. The parameter λ is set to 1.0 in experiments.

(a) *Family*

(b) *Panther*

(c) *Horse*

(d) *Playground*

(e) *Francis*

(f) *Train*

(g) *Lighthouse*

(h) *M60*

Fast-MVSNet: Sparse-to-Dense Multi-View Stereo With Learned Propagation and Gauss-Newton Refinement

Zehao Yu and Shenghua Gao

ShanghaiTech University

上海科技大学

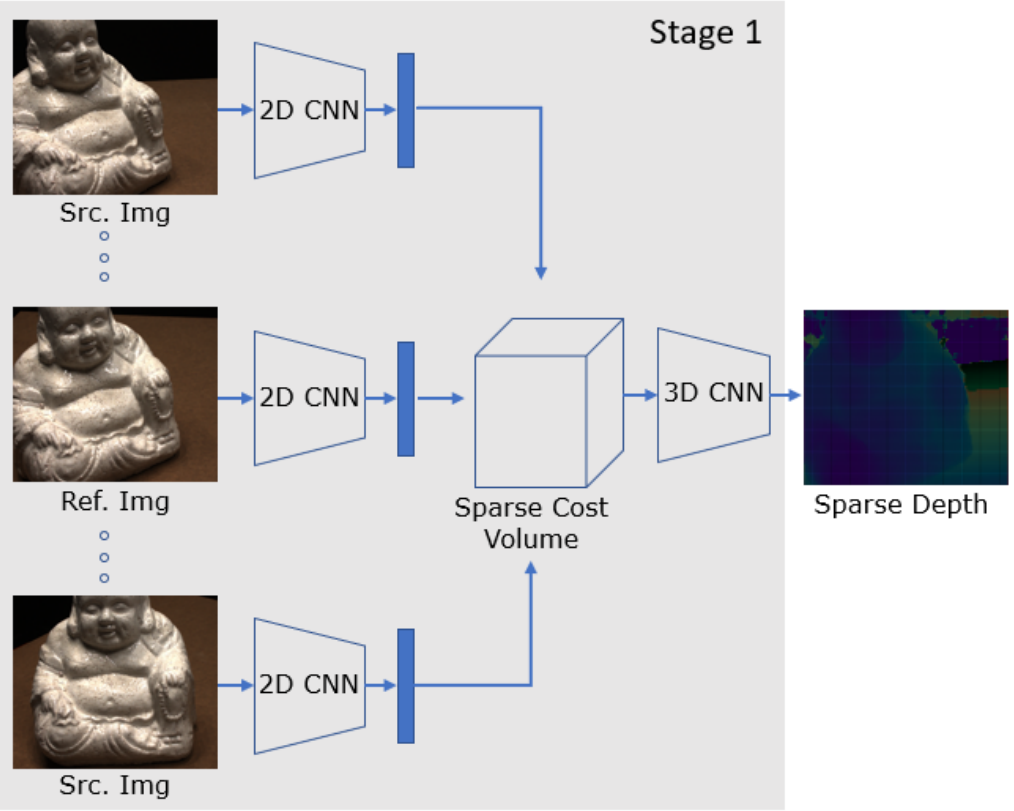
ShanghaiTech University

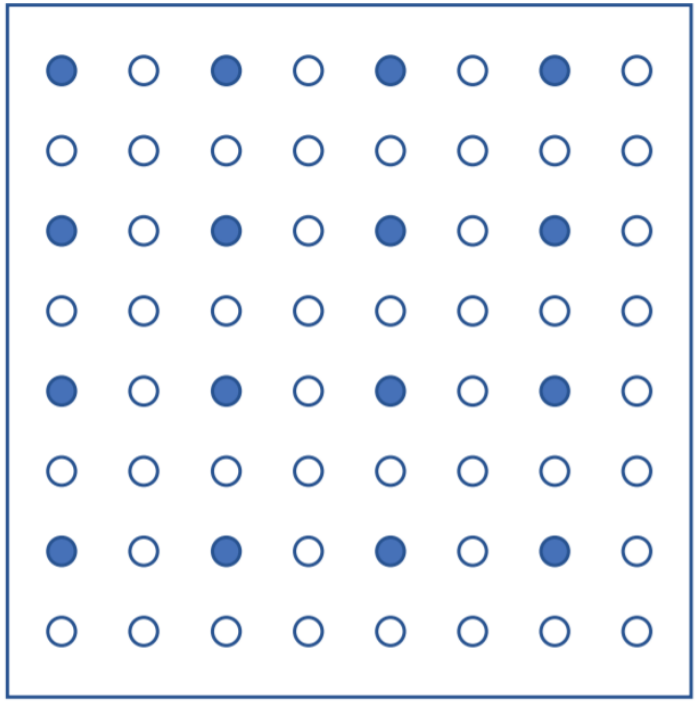
Motivation

- Both quality and efficiency are desirable features for MVS in real scenarios
- High-resolution depth map contains finer details. But predicting a high-resolution depth map from a 3D cost volume is computationally expensive and memory-intensive.
- Low-resolution depth can be predicted at much lower cost but with much fewer details.

Fast-MVSNet

- Predict a sparse high-resolution depth map with low cost.

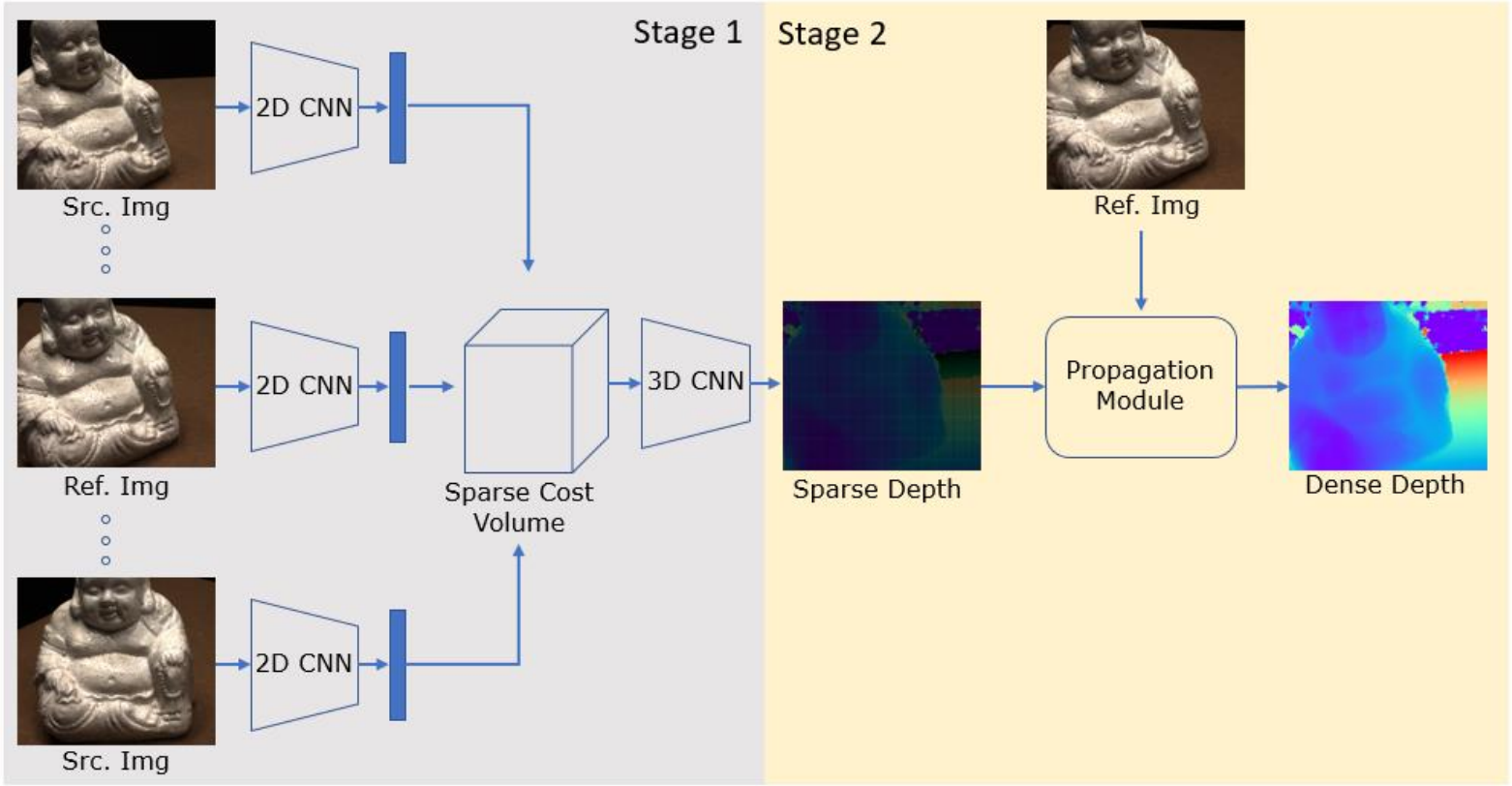




sparse high-resolution depth map

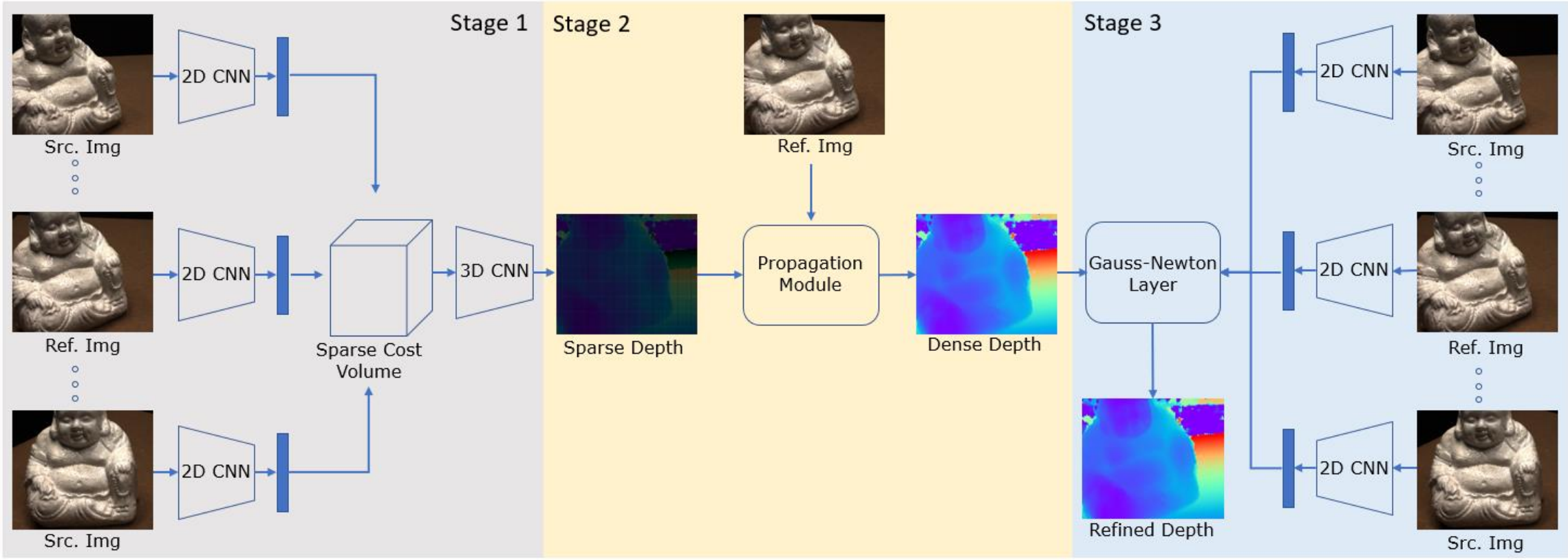
Fast-MVSNet

- Learning the propagation of the sparse depth map to a dense depth map.



Fast-MVSNet

- Use a differentiable Gauss-Newton layer to further refine the depth map.



Gauss-Newton Refinement

- Compute residual:

$$r_i(p) = F_i(p'_i) - F_0(p)$$

- Compute derivative w.r.t. depth map:

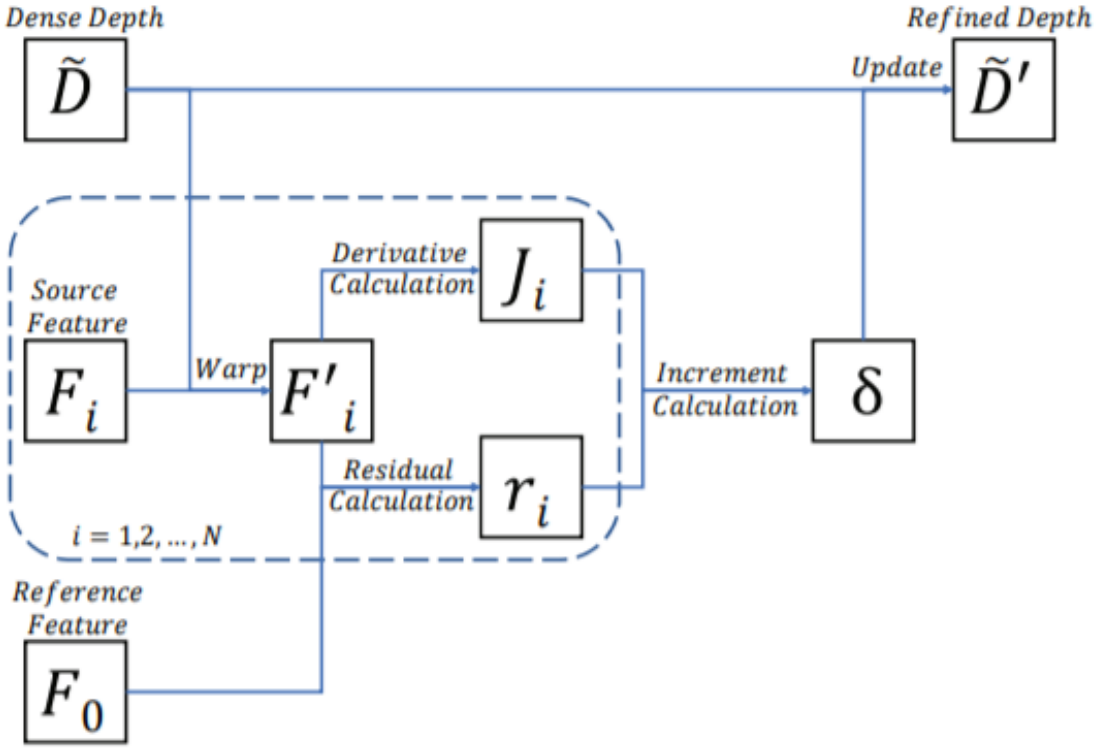
$$J_i(p) = \frac{\partial F_i(p'_i)}{\partial p'_i} \cdot \frac{\partial p'_i}{\partial \tilde{D}(p)}$$

- Compute increment:

$$\delta = -(J^T J)^{-1} J^T r$$

- Update depth map:

$$\tilde{D}'(p) = \tilde{D}(p) + \delta.$$



Experiments

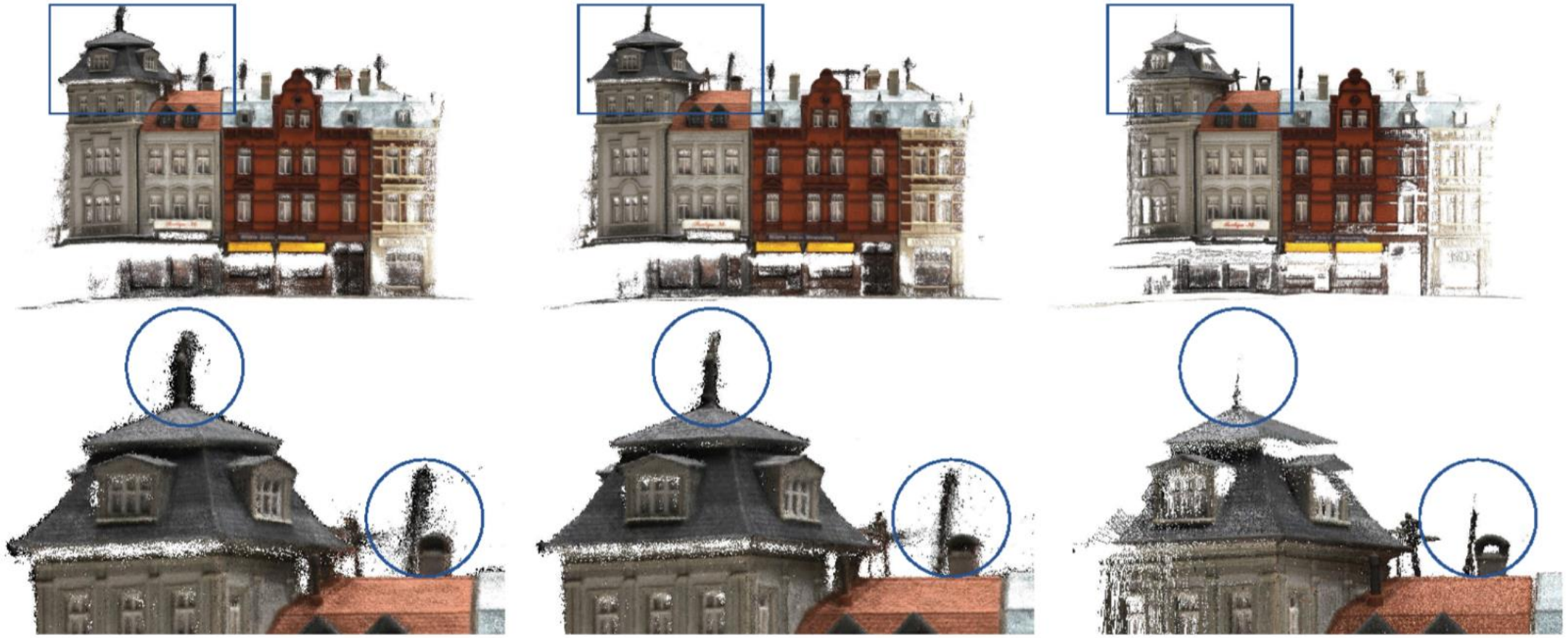
The DTU dataset

	Acc. (mm)	Comp. (mm)	Overall (mm)	Depth Map Res.	GPU Mem. (GB)	Runtime (s)
MVSNet[42]	0.456	0.646	0.551	288×216	10.8	1.05
R-MVSNet[42]	0.385	0.452	0.417	400×300	6.7	9.1
Point-MVSNet [7]	0.361	0.421	0.391	**640×480**	8.7	3.35
Ours	**0.336**	**0.403**	**0.370**	**640×480**	**5.3**	**0.6**

The Tank and Temples dataset

	Mean	Family	Francis	Horse	Lighthouse	M60	Panther	Playground	Train
MVSNet [42]	43.48	55.99	28.55	25.07	**50.79**	**53.96**	**50.86**	47.90	34.69
R-MVSNet [43]	**48.40**	**69.96**	**46.65**	32.59	42.95	51.88	48.80	52.00	42.38
Point-MVSNet [7]	48.27	61.79	41.15	34.20	**50.79**	51.97	50.85	52.38	**43.06**
Ours	47.39	65.18	39.59	**34.98**	47.81	49.16	46.20	**53.27**	42.91

Visualization on the DTU dataset



Point-MVSNet [7] | Ours | Ground Truth

Visualization on the Tank and Temples dataset






Ablation Study

method	Acc.	Comp.	Overall
low res.	0.517	0.557	0.537
sparse high res.	0.394	0.478	0.436
sparse high res. + prop.	0.370	0.448	0.409
sparse high res. + prop. + GN	**0.336**	**0.403**	**0.370**
low res. + *PointFlow* [7]	0.361	0.421	0.391
low res. + GN.	0.376	0.417	0.396

Conclusion

- A sparse-to-dense coarse-to-fine framework for MVS. Sparse-to-dense strategy guarantees the efficiency while the coarse-to-fine strategy guarantees the effectiveness.
- Learning the depth dependencies for pixels within a local region with a small-scale convolutional neural network to densify the sparse depth map
- A differentiable Gauss-Newton layer to optimize the depth map.
- Achieves better or comparable reconstruction results compared to state-of-the-art methods while being much more efficient and memory-friendly.

Fitting a homography

- Equation for homography:

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad \lambda \mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$$

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0$$

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h_1}^T \mathbf{x}_i \\ \mathbf{h_2}^T \mathbf{x}_i \\ \mathbf{h_3}^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y_i' \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x_i' \mathbf{h}_3^T \mathbf{x}_i \\ x_i' \mathbf{h}_2^T \mathbf{x}_i - y_i' \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

$$\begin{bmatrix} 0^T & -\mathbf{x}_i^T & y_i' \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x_i' \mathbf{x}_i^T \\ -y_i' \mathbf{x}_i^T & x_i' \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

3 equations, only 2 linearly independent

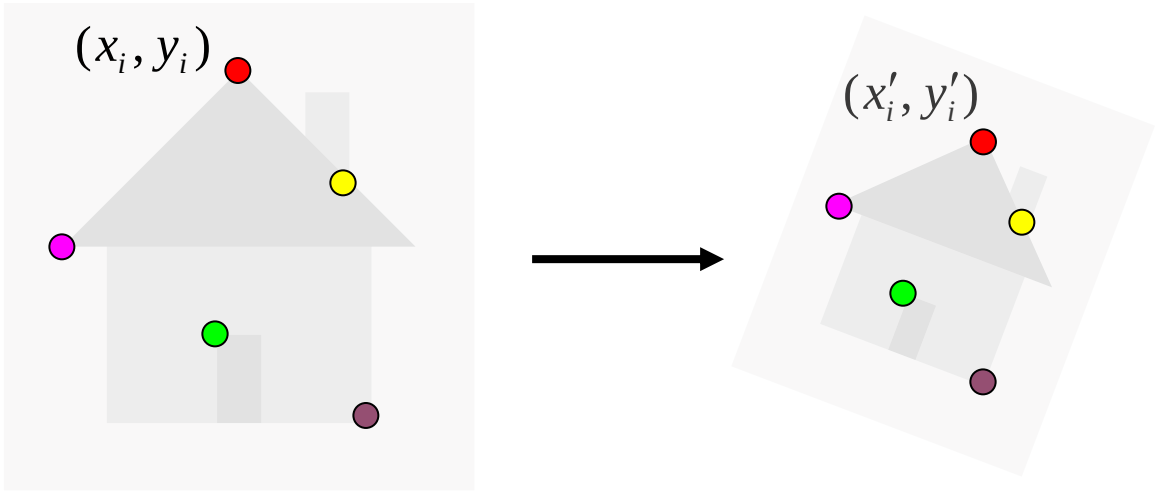
Fitting a homography

$$\begin{bmatrix} 0^T & \mathbf{x}_1^T & -y_1'\mathbf{x}_1^T \\ \mathbf{x}_1^T & 0^T & -x_1'\mathbf{x}_1^T \\ \cdots & \cdots & \cdots \\ 0^T & \mathbf{x}_n^T & -y_n'\mathbf{x}_n^T \\ \mathbf{x}_n^T & 0^T & -x_n'\mathbf{x}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \qquad \mathbf{A}\mathbf{h} = 0$$

- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Homogeneous least squares: find $\mathbf{h}$ minimizing $\|\mathbf{A}\mathbf{h}\|^2$
  - Eigenvector of $\mathbf{A}^\mathbf{T}\mathbf{A}$ corresponding to smallest eigenvalue
  - Four matches needed for a minimal solution

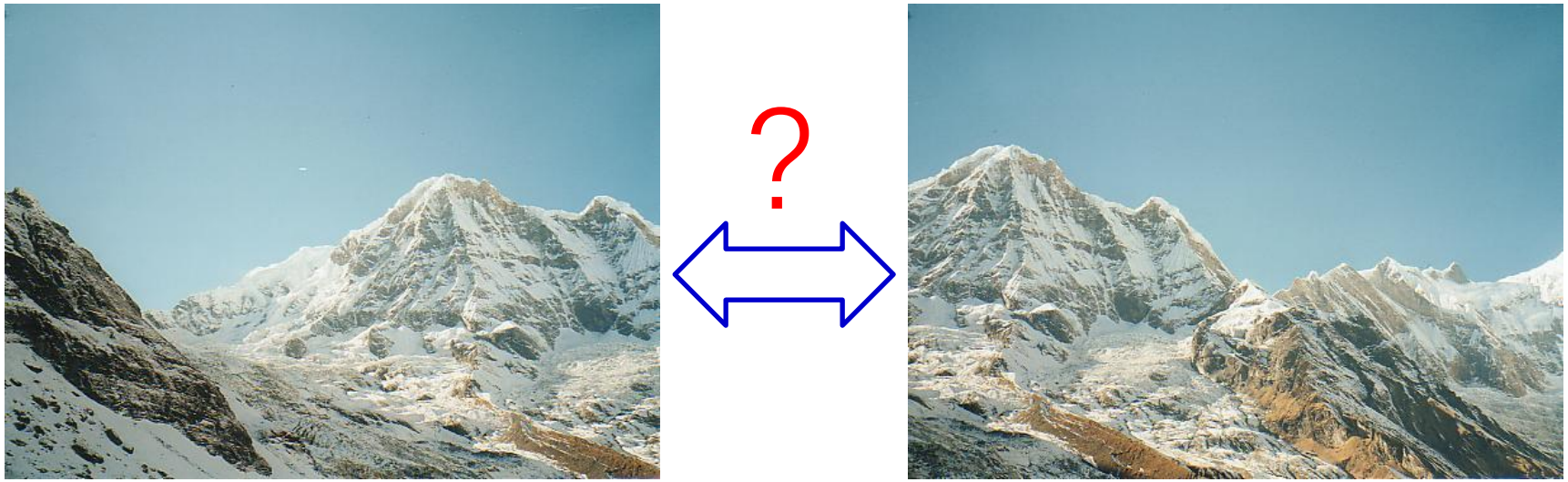
Robust feature-based alignment

- So far, we've assumed that we are given a set of "ground-truth" correspondences between the two images we want to align
- What if we don't know the correspondences?



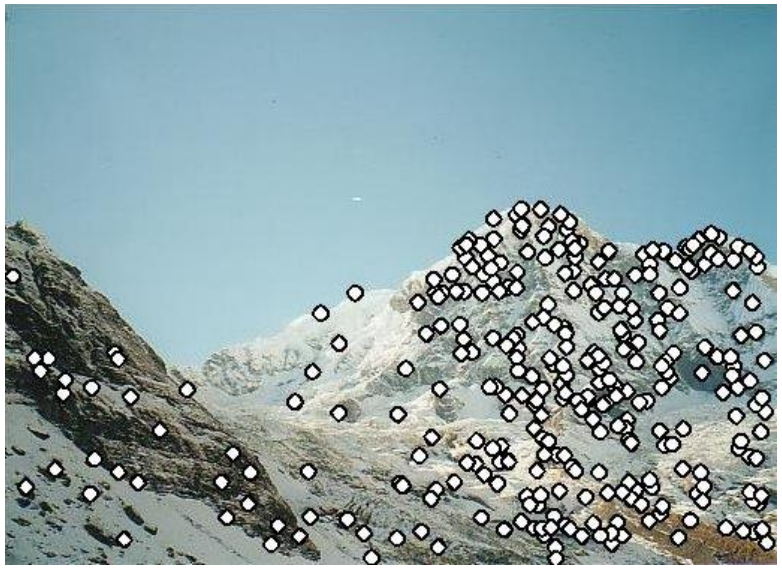
Robust feature-based alignment

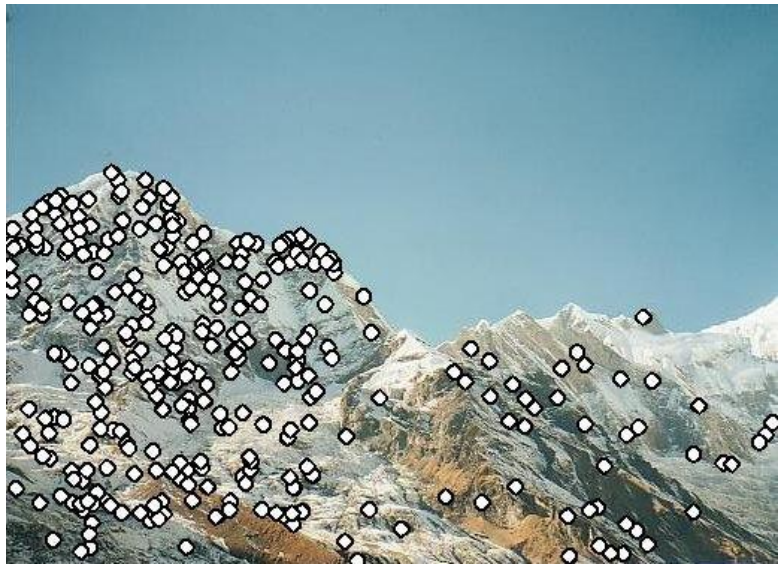
- So far, we've assumed that we are given a set of "ground-truth" correspondences between the two images we want to align
- What if we don't know the correspondences?



Robust feature-based alignment

Robust feature-based alignment



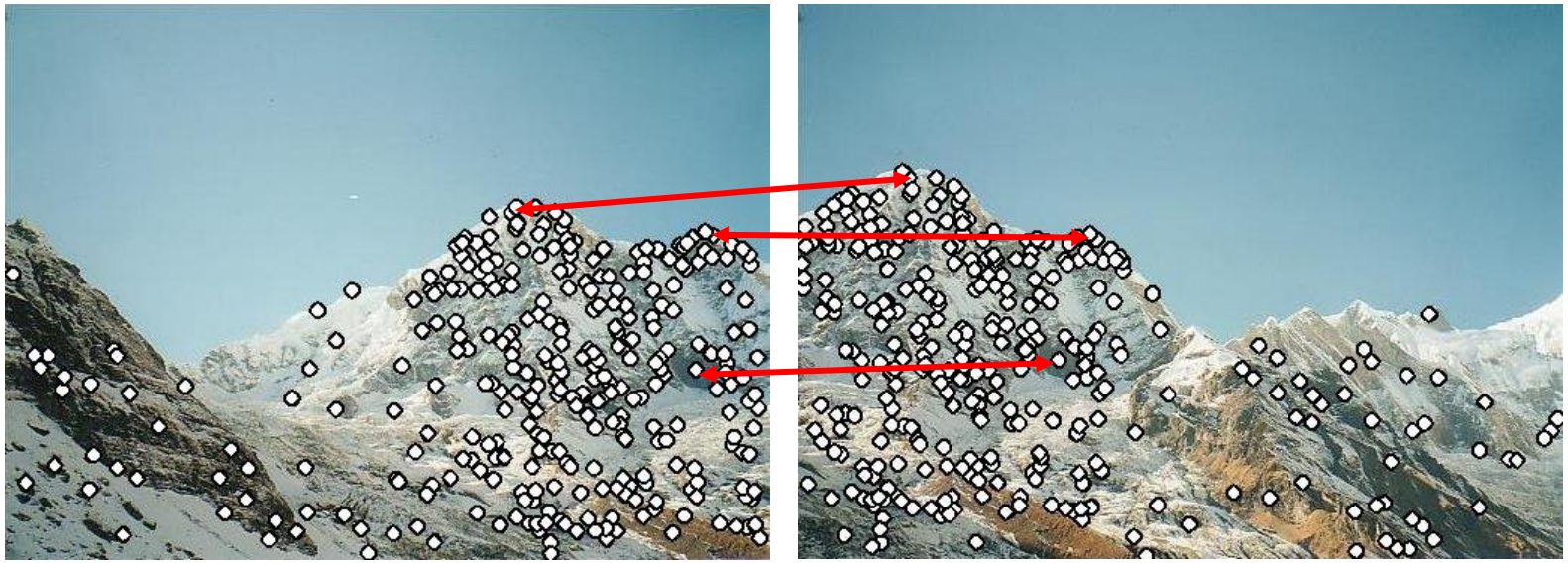


- Extract features

Robust feature-based alignment

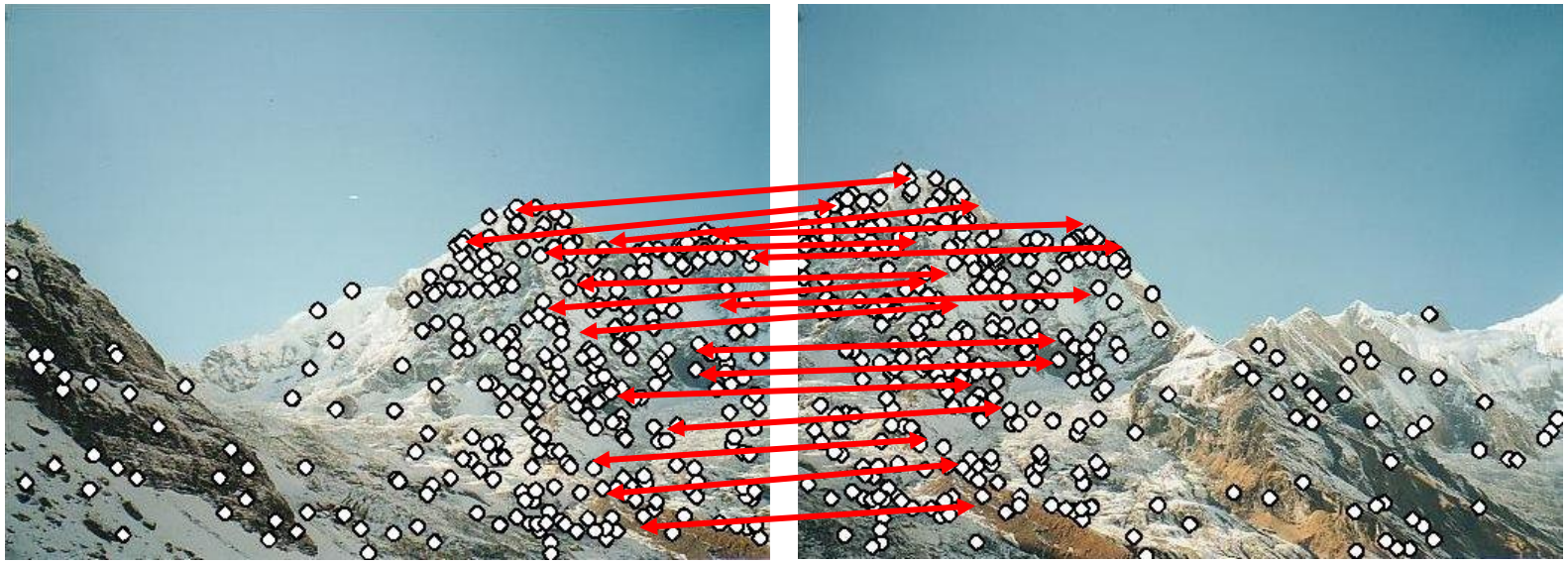
- Extract features
- Compute *putative matches*

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

Robust feature-based alignment

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

RANSAC

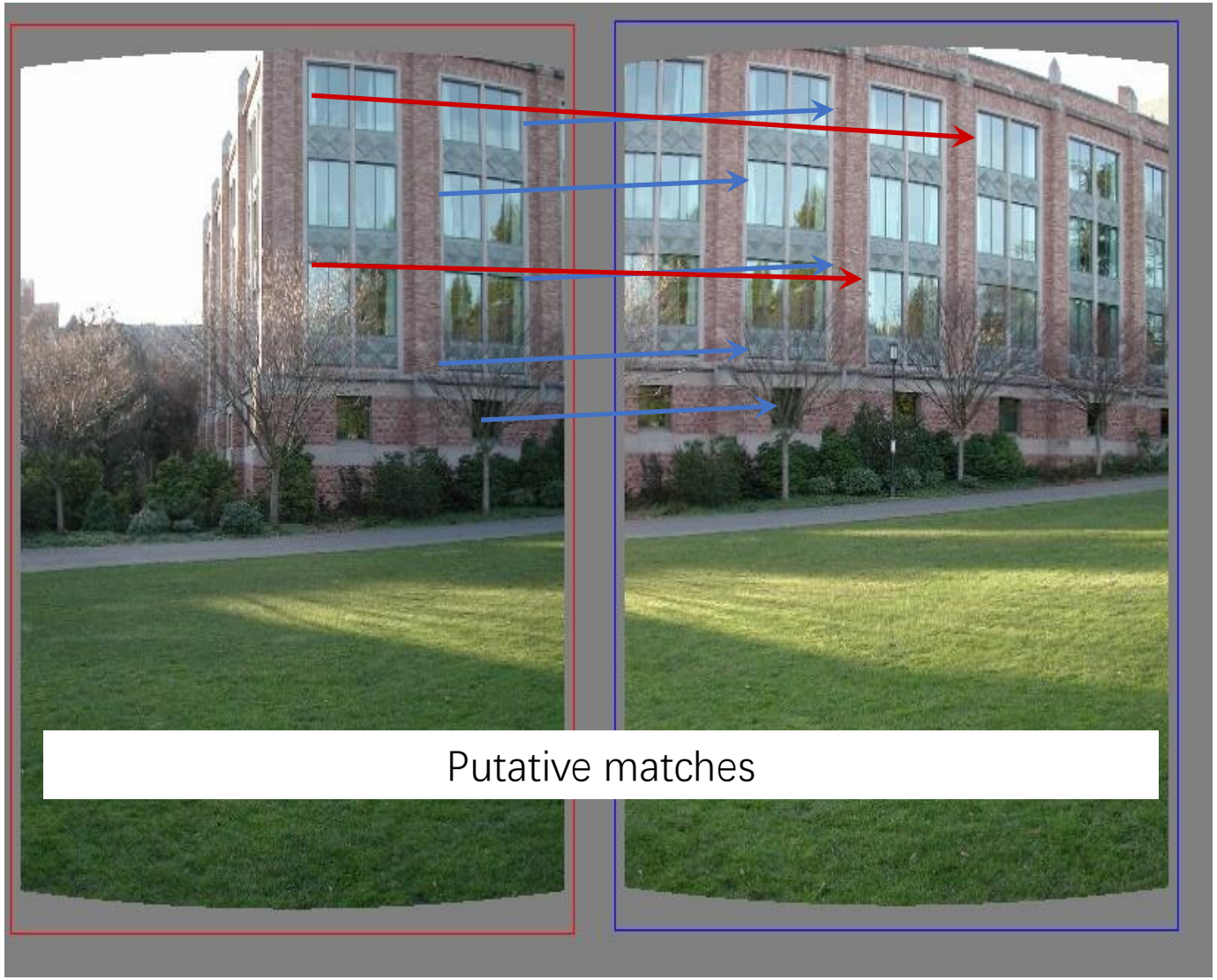
- The set of putative matches contains a very high percentage of outliers

- **RANSAC loop:**

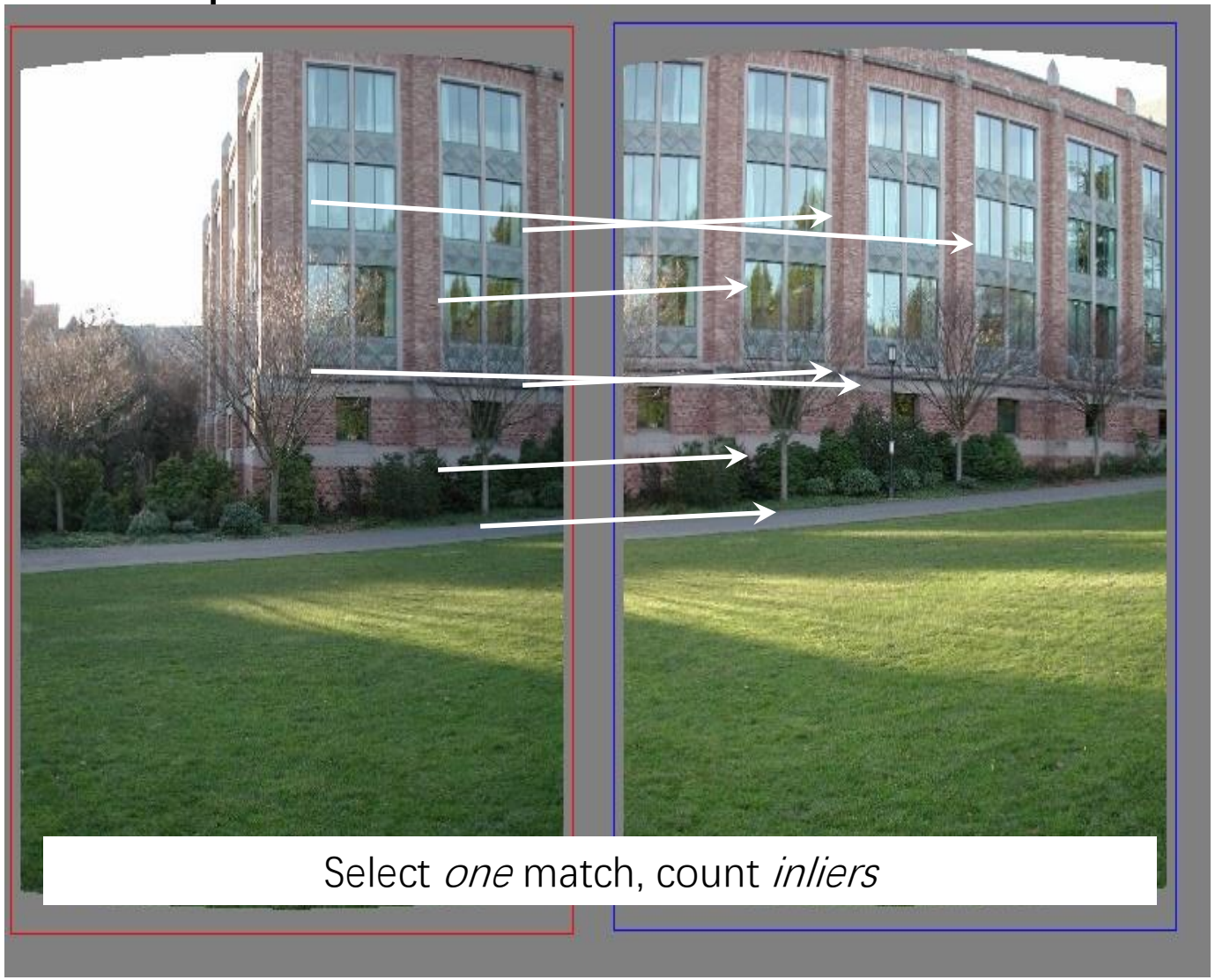
1. Randomly select a *seed group* of matches
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

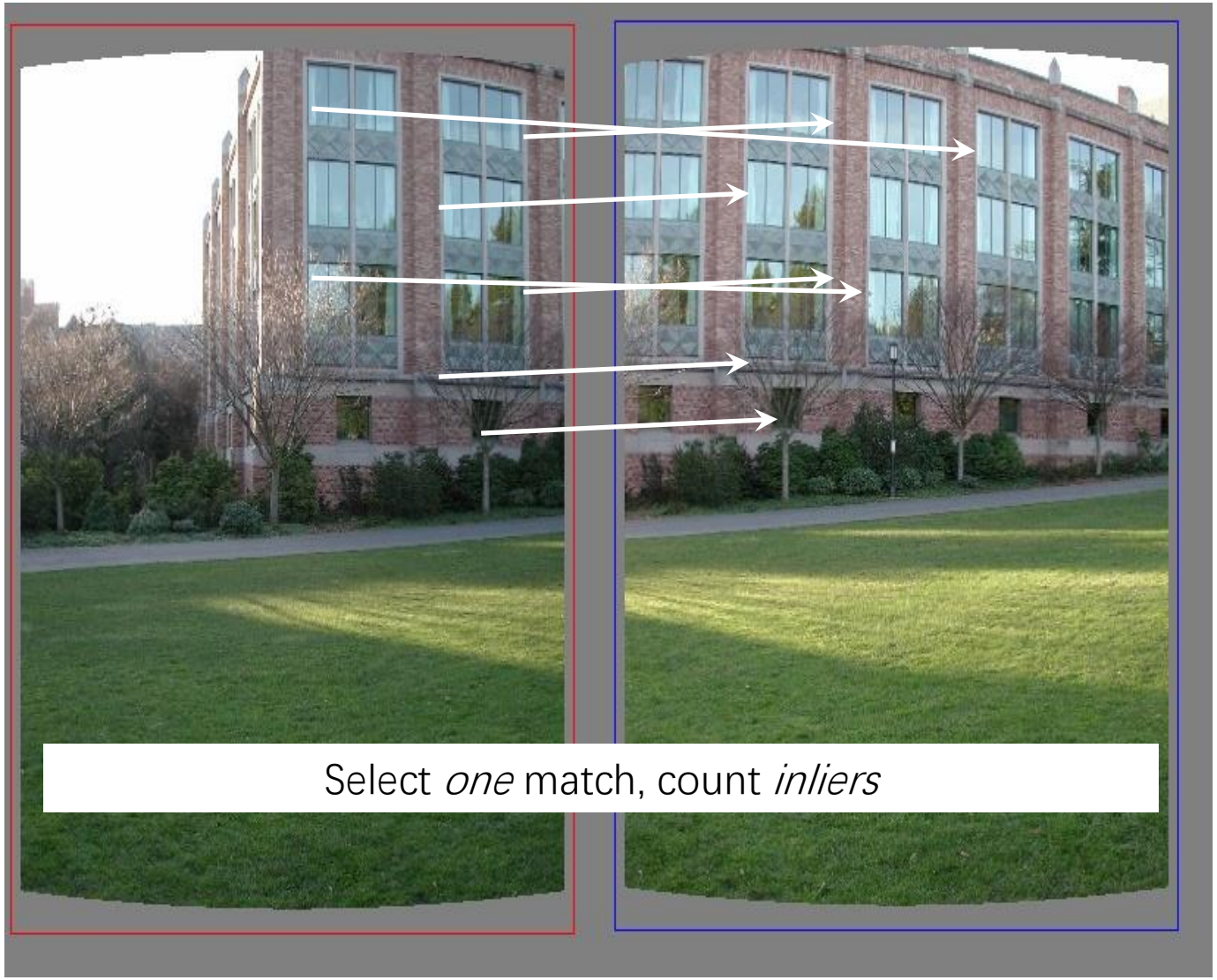
RANSAC example: Translation



RANSAC example: Translation



RANSAC example: Translation



RANSAC example: Translation