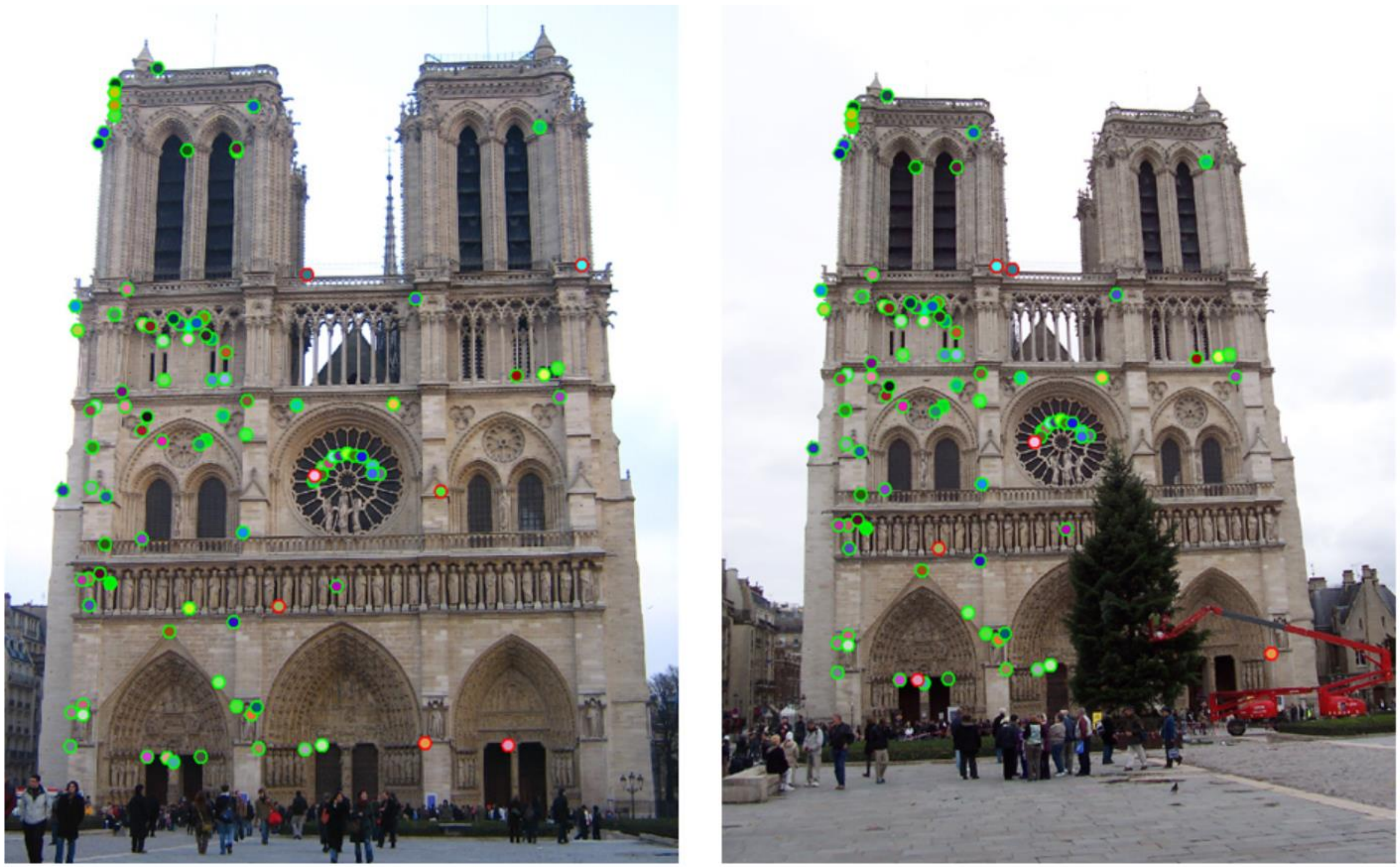
SIFT-based Image Alignment

Jianping Fan
Dept of Computer Science
UNC-Charlotte

Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

Project 2: SIFT-based Image Alignment

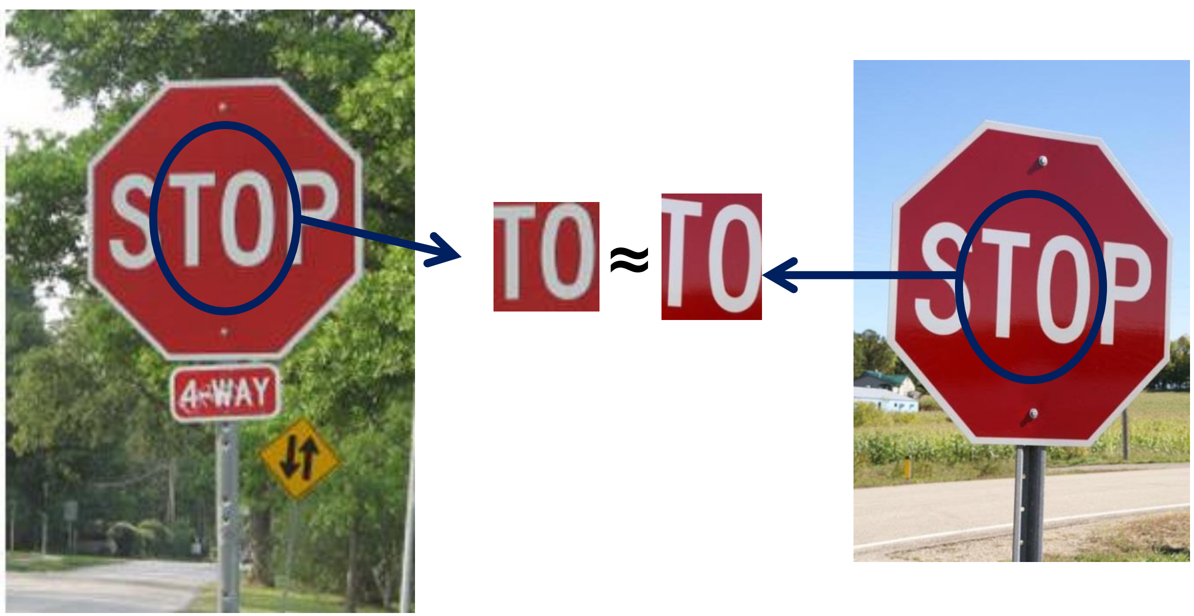


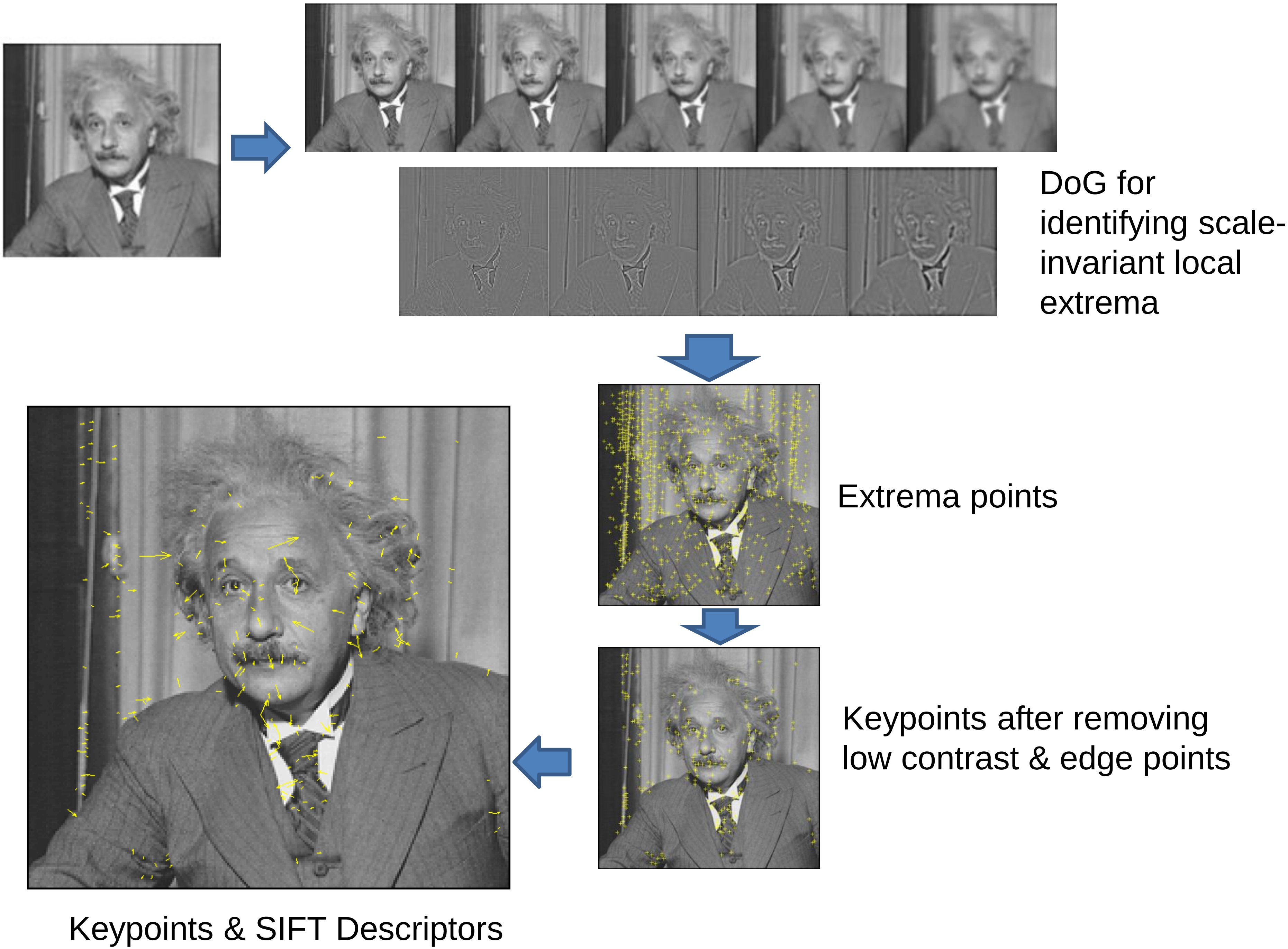
The top 100 most confident local feature matches from a baseline implementation of project 2. In this case, 93 were correct (highlighted in green) and 7 were incorrect (highlighted in red).

Project 2: Local Feature Matching

Correspondence and Alignment

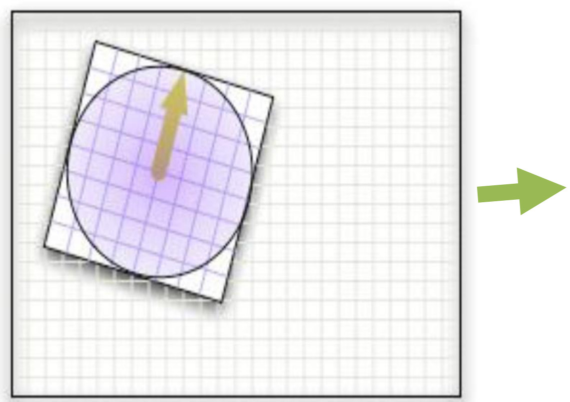
- **Correspondence**: matching points, patches, edges, or regions across images




DoG for identifying scale-invariant local extrema
Extrema points
Keypoints after removing low contrast & edge points
Keypoints & SIFT Descriptors

Keypoint & SIFT Descriptor

- 16x16 Gradient window is taken. Partitioned into 4x4 subwindows.
- Histogram of 4x4 samples in 8 directions
- Gaussian weighting around center(σ is 0.5 times that of the scale of a keypoint)
- 4x4x8 = 128



Keypoint Detection
& Orientation Determination
& Neighborhood Pattern

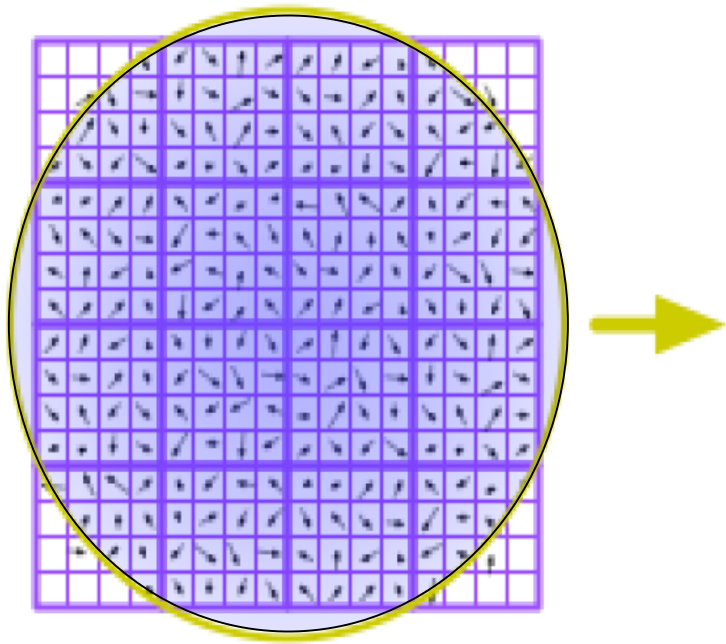
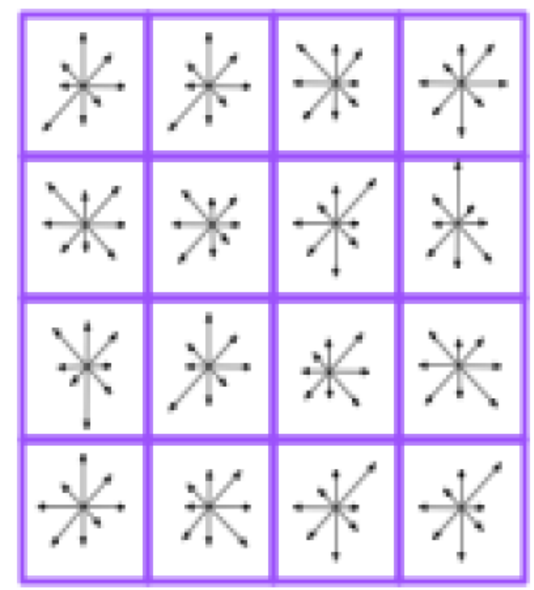
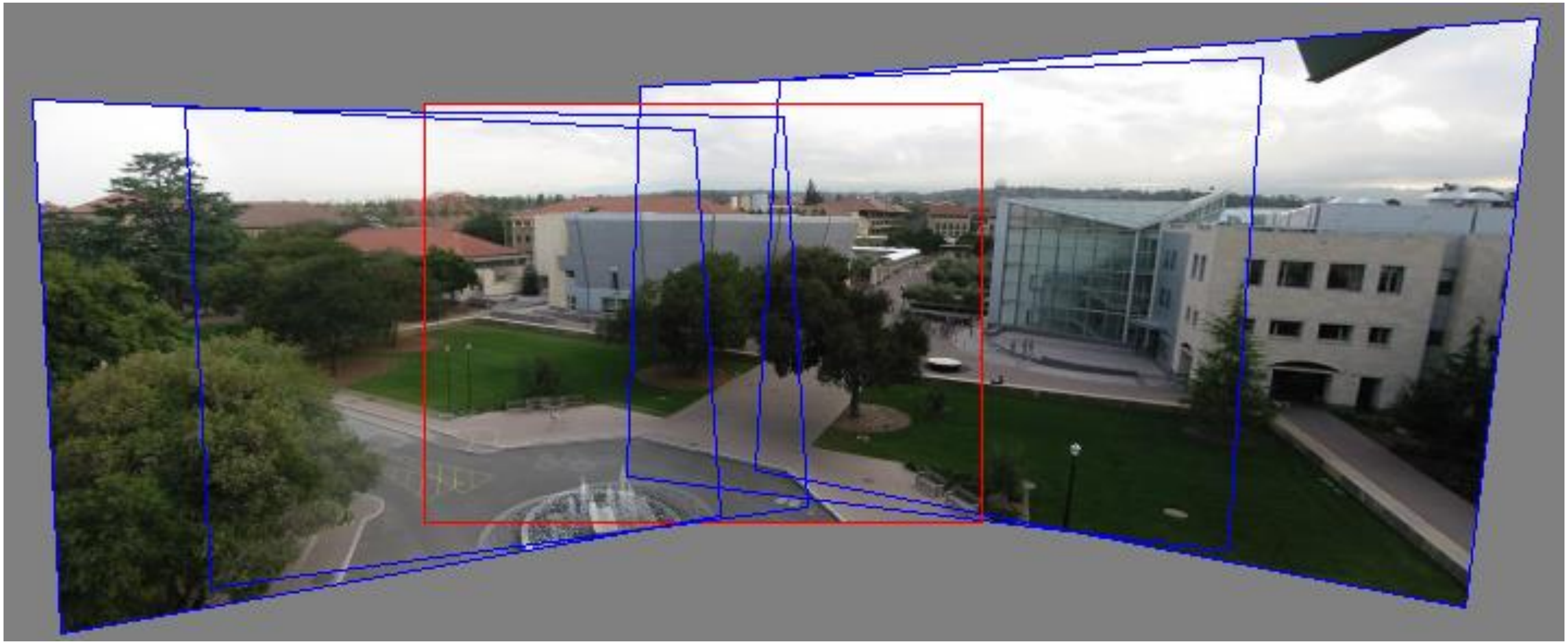


Image gradients



Keypoint descriptor

Image alignment



www.cs.unc.edu/~lazebnik/spring10/lec10_alignment

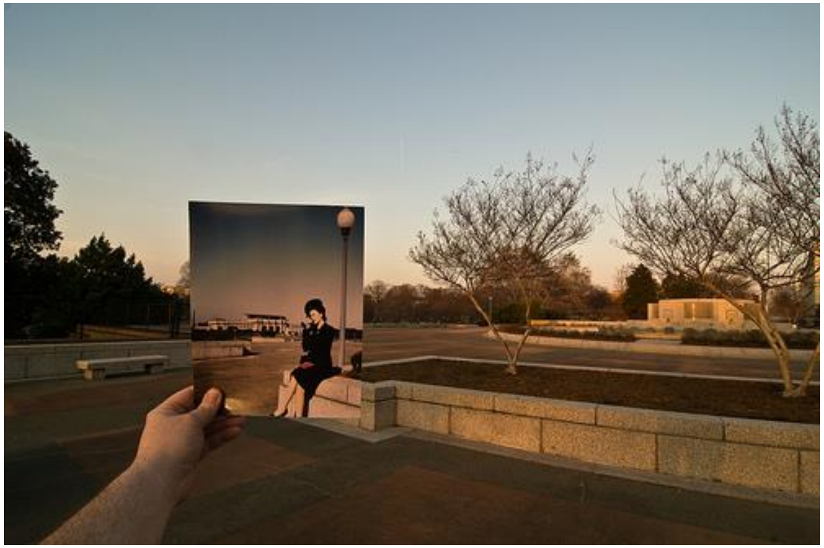
Image Alignment Algorithm

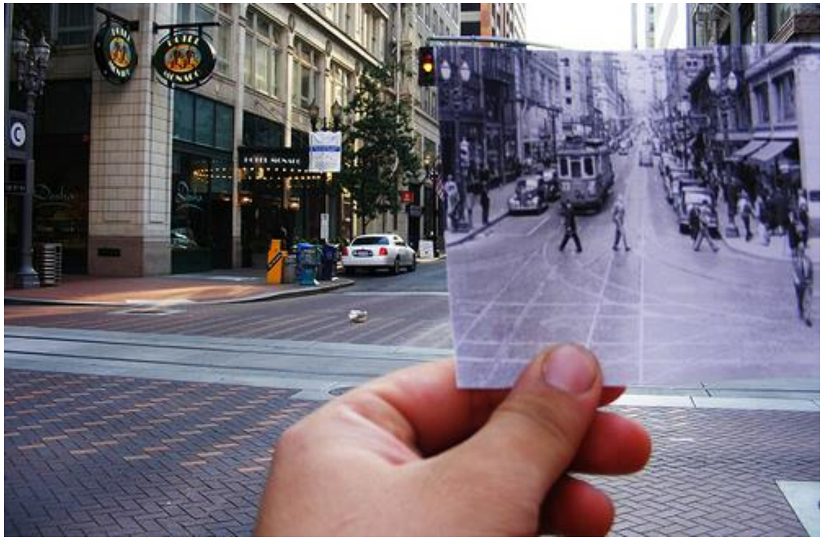
Given images A and B

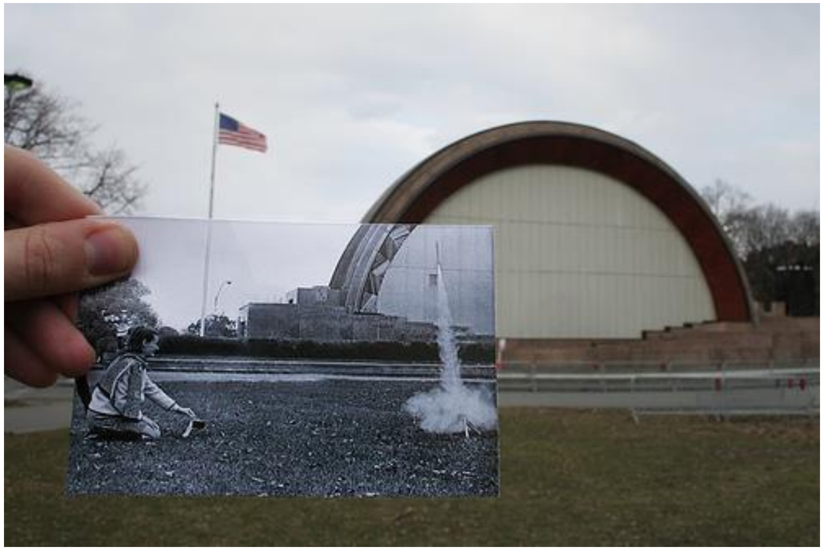
1. Compute image features for A and B
2. Match features between A and B
3. Compute homography between A and B using least squares on set of matches

What could go wrong?

A look into the past



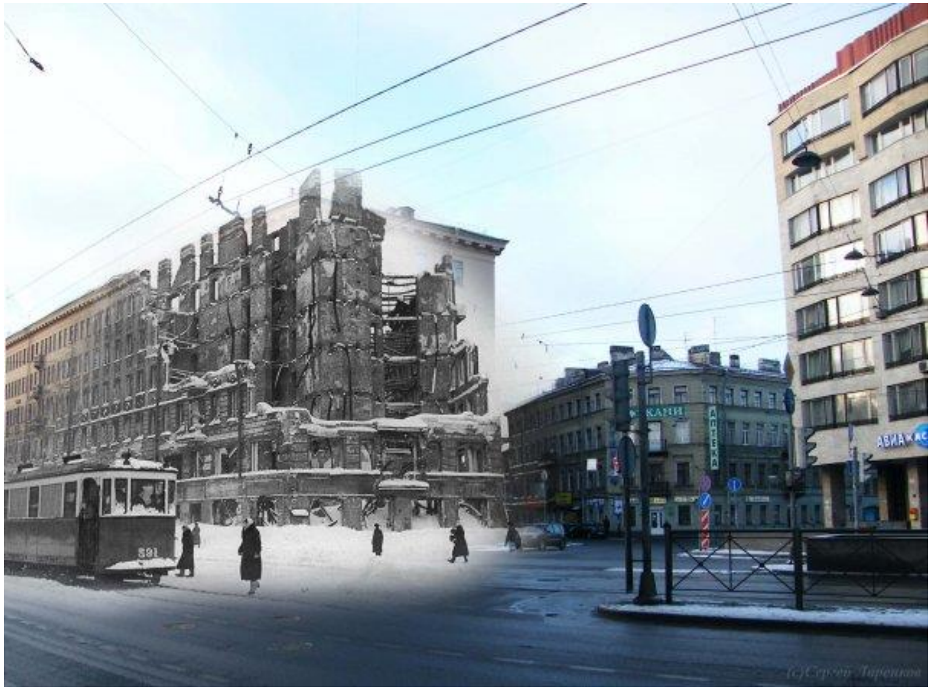




http://blog.flickr.net/en/2010/01/27/a-look-into-the-past/

A look into the past

- Leningrad during the blockade

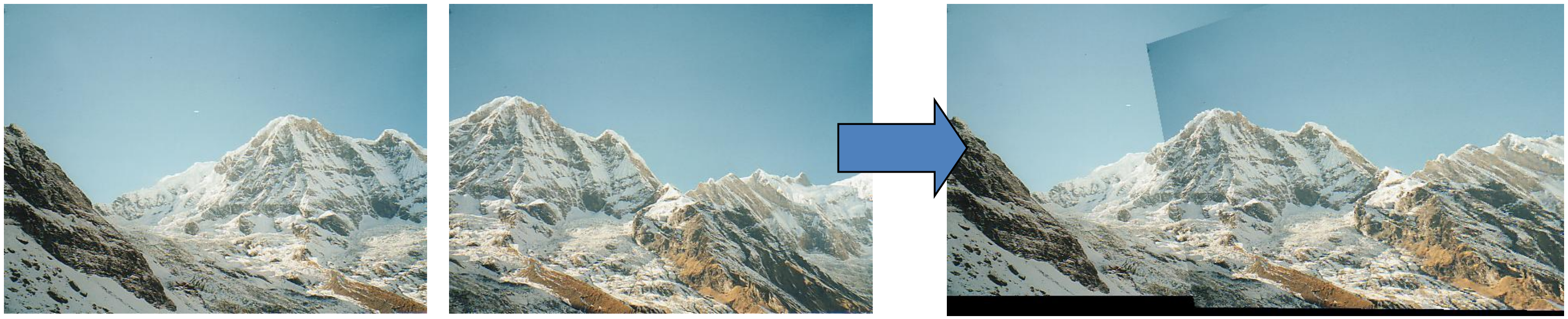


http://komen-dant.livejournal.com/345684.html

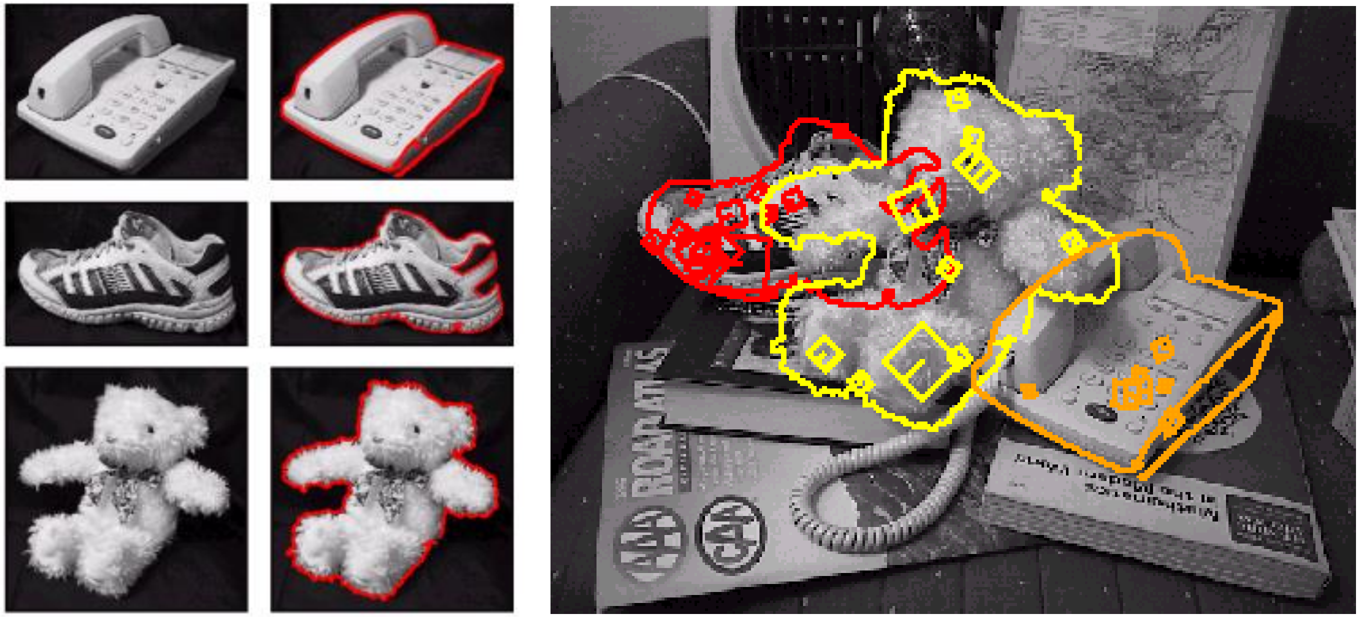
Bing streetside images

http://www.bing.com/community/blogs/maps/archive/2010/01/12/new-bing-maps-application-streetside-photos.aspx

Image alignment: Applications

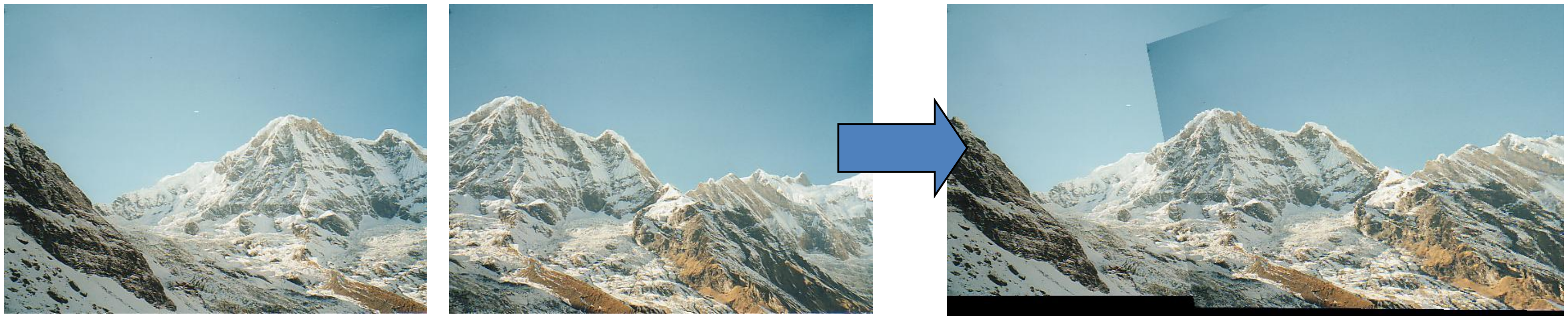


Panorama stitching



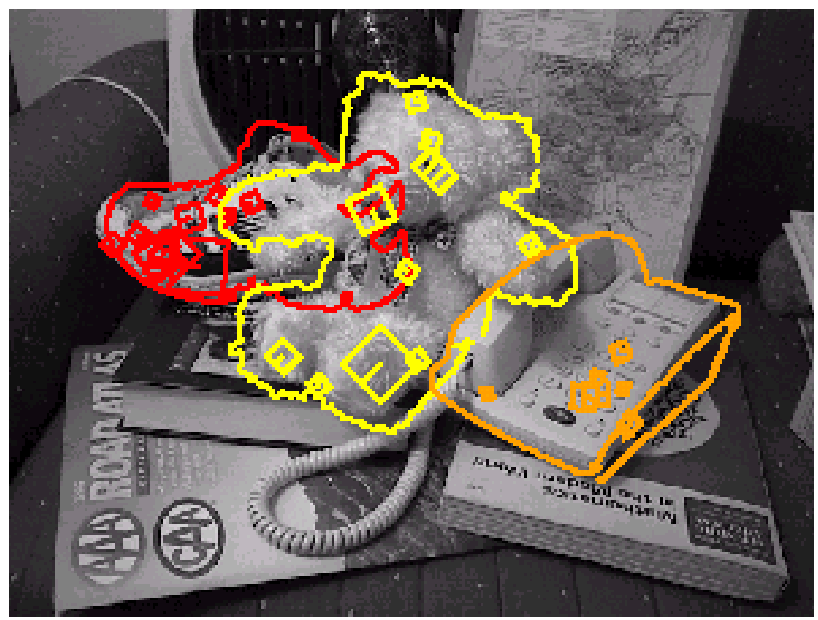
Recognition of object instances

Image alignment: Challenges



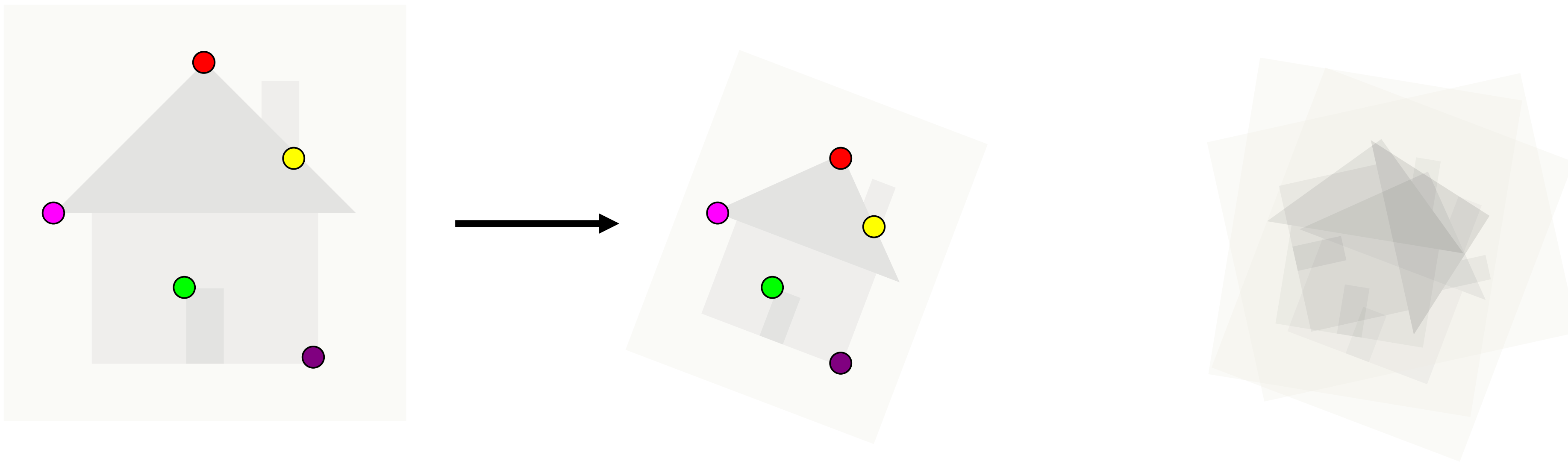
Small degree of overlap

Intensity changes



Occlusion, clutter

Image alignment

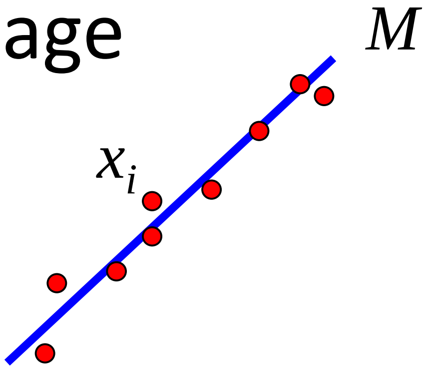


- Two broad approaches:
  - **Direct (pixel-based) alignment**
    - Search for alignment where most pixels agree
  - **Feature-based alignment**
    - Search for alignment where *extracted features* agree
    - Can be verified using pixel-based alignment

Image Alignment as Fitting

Alignment as fitting

- Previous lectures: fitting a model to features in one image

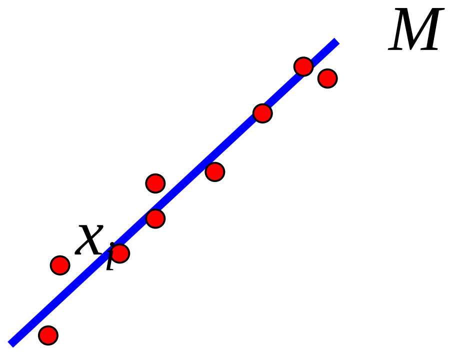


Find model *M* that minimizes

$$\sum_i \text{residual}(x_i, M)$$

Alignment as fitting

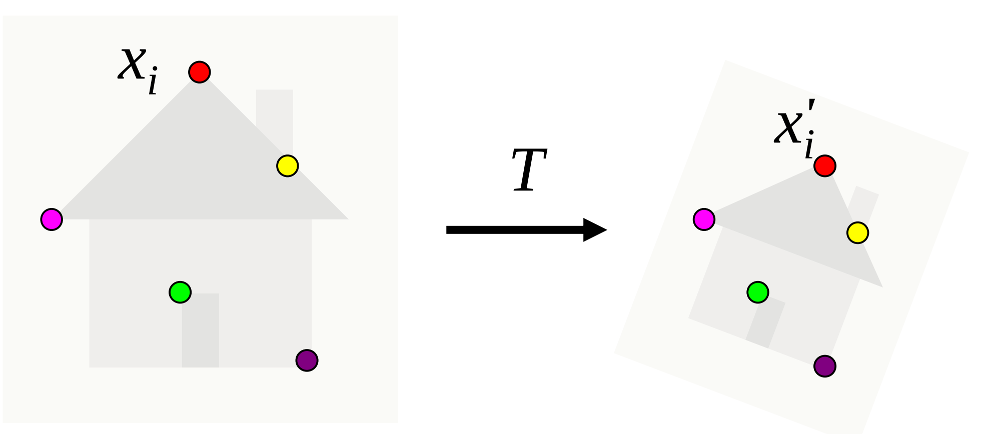
- Fitting a model to features in one image



Find model *M* that minimizes

$$\sum_i \text{residual}(x_i, M)$$

- **Alignment**: fitting a model to a transformation between pairs of features (*matches*) in two images

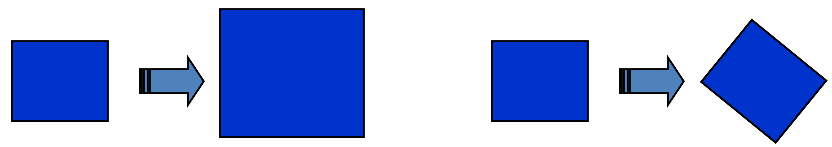


Find transformation *T*
that minimizes

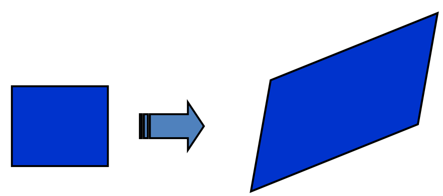
$$\sum_i \text{residual}(T(x_i), x_i')$$

2D transformation models

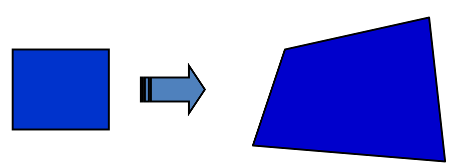
- **Similarity** (translation, scale, rotation)



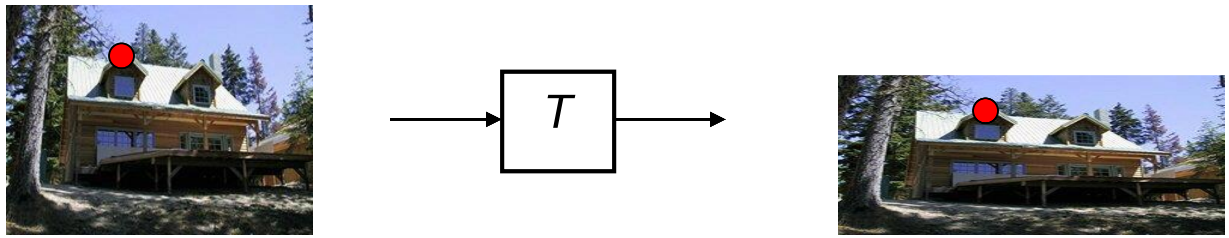
- **Affine**



- **Projective** (homography)



Parametric (global) warping



p = (x,y) **p'** = (x',y')

Transformation T is a coordinate-changing machine:

p' = *T*(p)

What does it mean that *T* is global?

– Is the same for any point p
– can be described by just a few numbers (parameters)

For **linear transformations**, we can represent T as a matrix

p' = **T**p

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Common transformations

original

Transformed

translation

rotation

aspect

affine

perspective

Slide credit (next few slides):
A. Efros and/or S. Seitz

Transformations

Original Image

$$x' = x + v$$

$$y' = y + u$$

Transformed Image

Transformations

Original Image

Rotation

Transformed Image

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Transformations

Original Image

Aspect

Transformed Image

$$x' = x/r$$

$$y' = y/t$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Transformations

Affine

Original Image

Transformed Image

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Transformations

Original Image

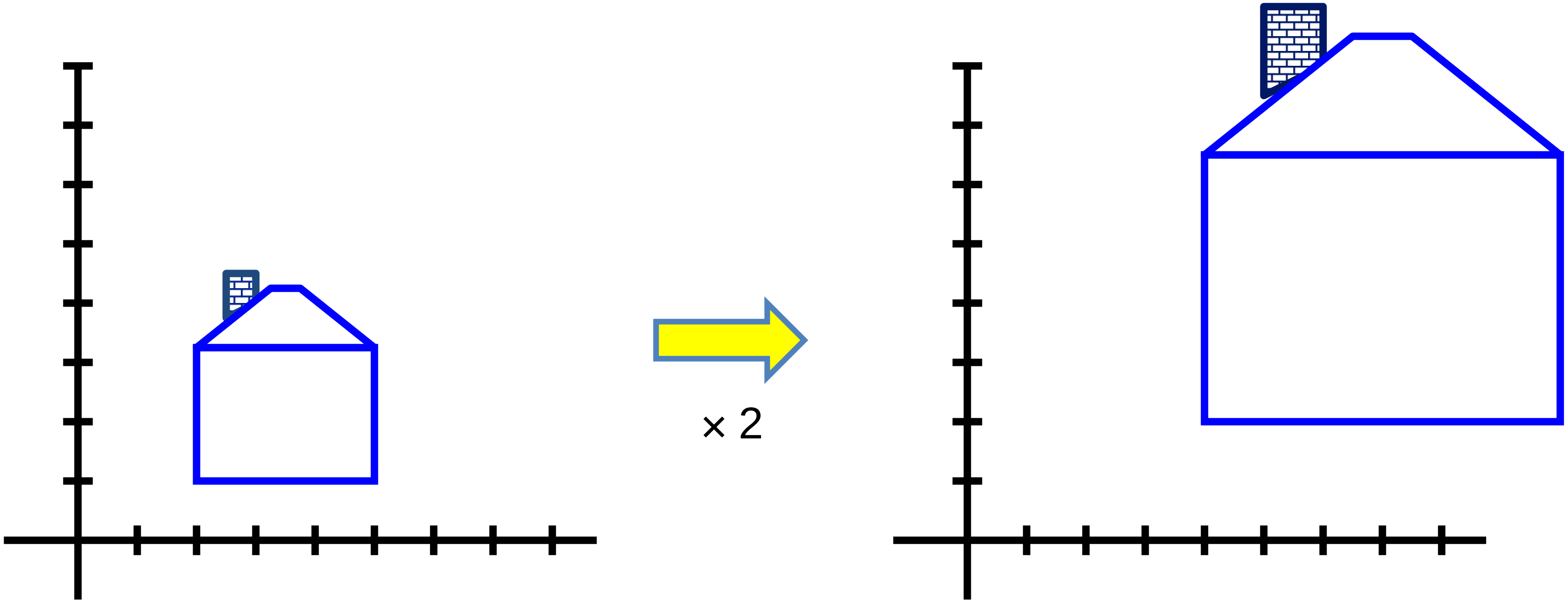
Perspective

Transformed Image

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

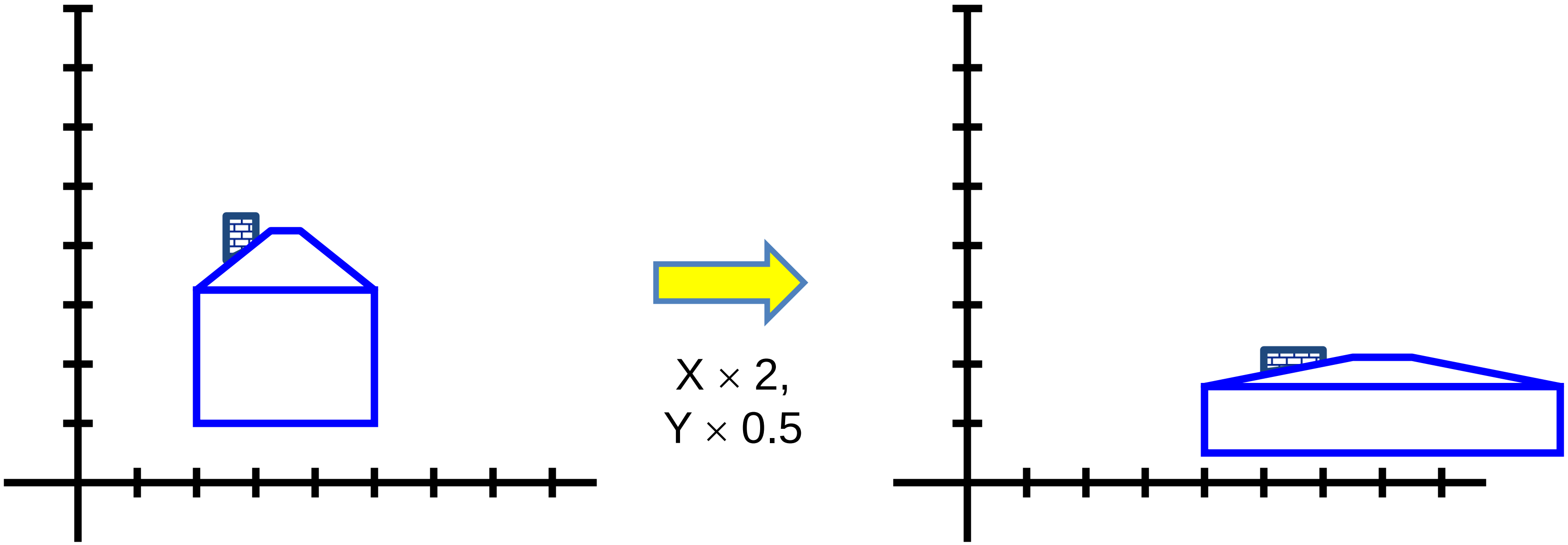
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



Scaling

- *Non-uniform scaling*: different scalars per component:



Scaling

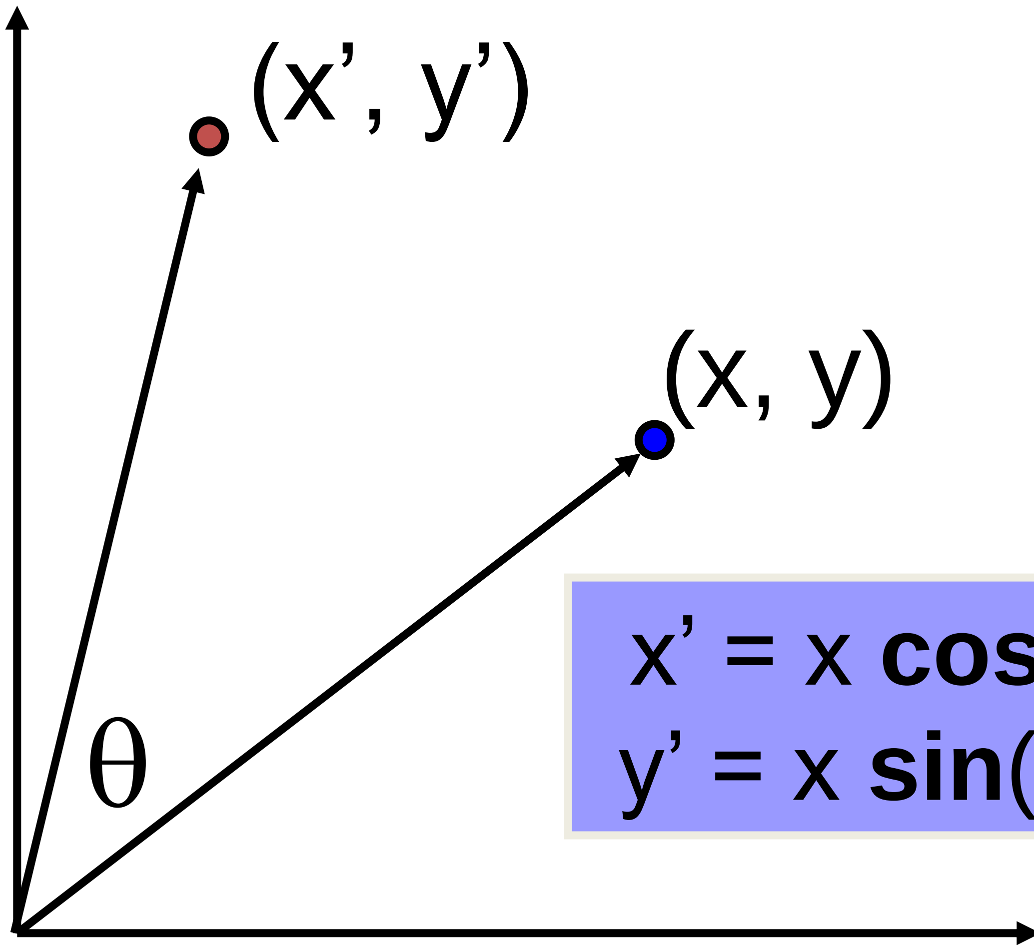
- Scaling operation:

$$x' = ax$$
$$y' = by$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

2-D Rotation



$$x' = x \cos(\theta) - y \sin(\theta)$$
$$y' = x \sin(\theta) + y \cos(\theta)$$

2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though sin(θ) and cos(θ) are nonlinear functions of θ,

- ***x' is a linear combination of x and y***
- ***y' is a linear combination of x and y***

What is the inverse transformation?

- Rotation by –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shear

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Projective Transformations

Projective transformations are combos of

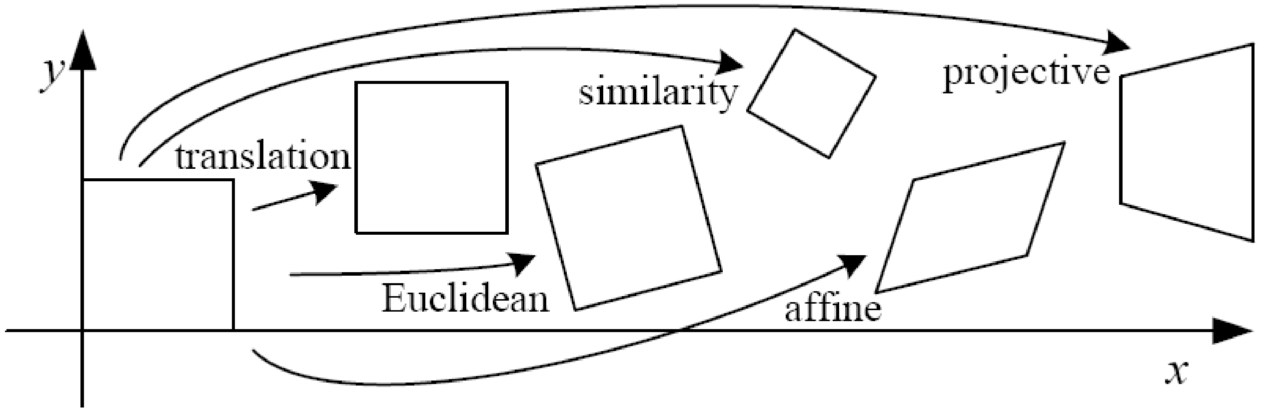
- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

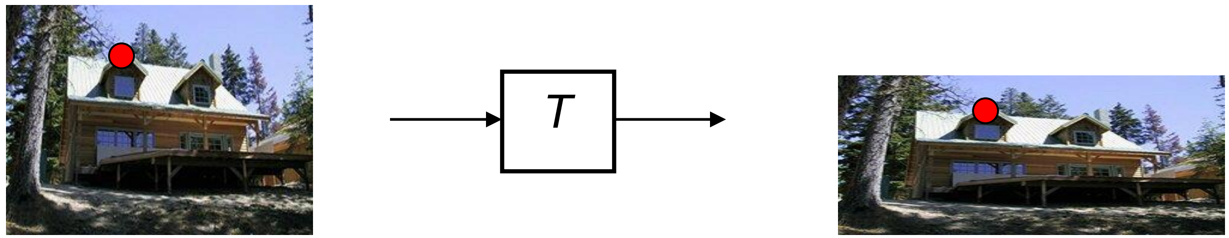
2D image transformations (reference table)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+ \cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+ \cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles $+ \cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism $+ \cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

Szeliski 2.1

Image matching under transformation



p = (x,y) **p’** = (x’,y’)

Transformation T is a coordinate-changing machine:

p’ = *T*(p)

What does it mean that *T* is global?

- Is the same for any point p
- can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix

p’ = **T**p

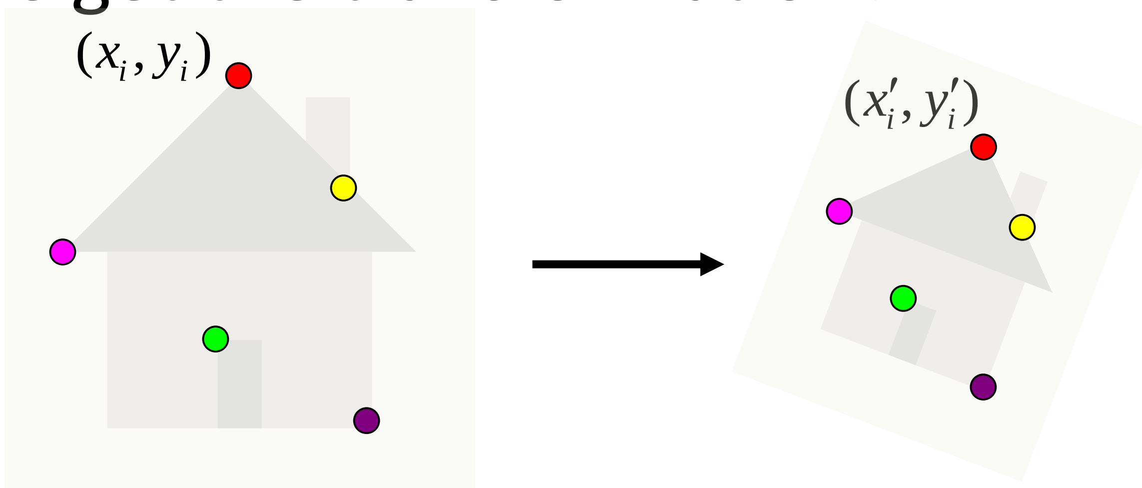
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

affine transformations

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

Fitting an affine transformation

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



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

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

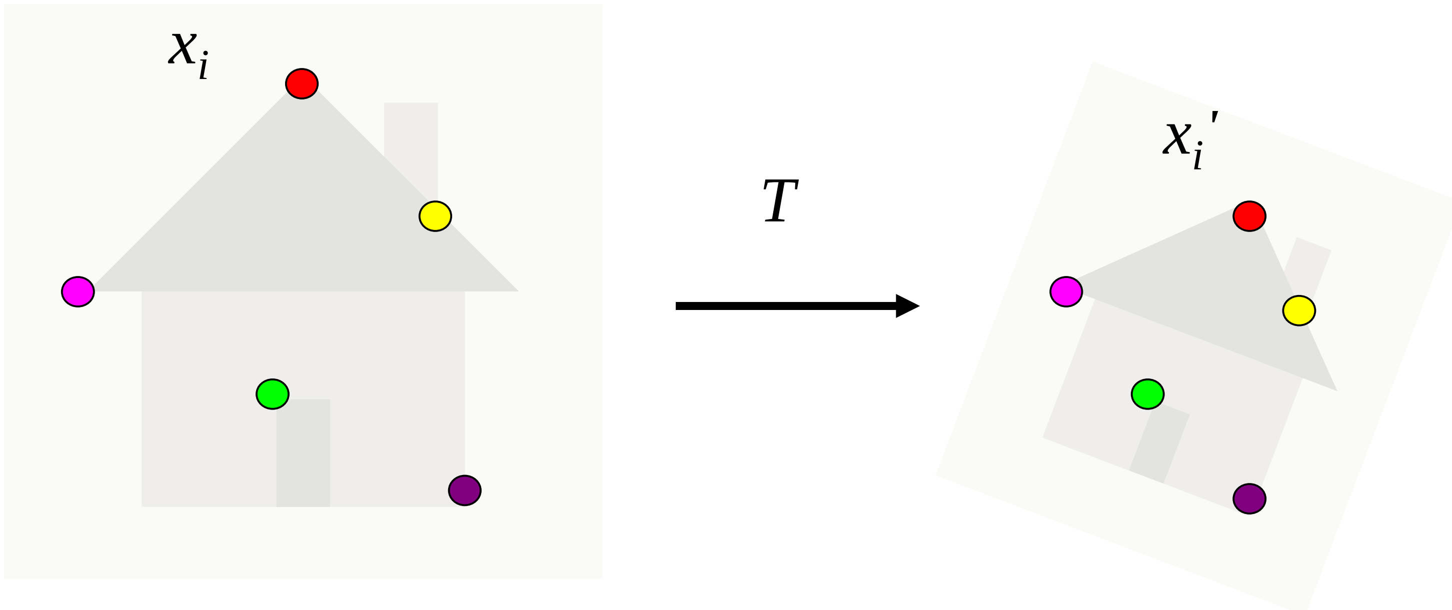
Fitting an affine transformation

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

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

Alignment as fitting

- Transformation between pairs of features (*matches*) in two images



Find transformation *T*
that minimizes

$$\sum_i \text{residual}(T(x_i), x_i')$$

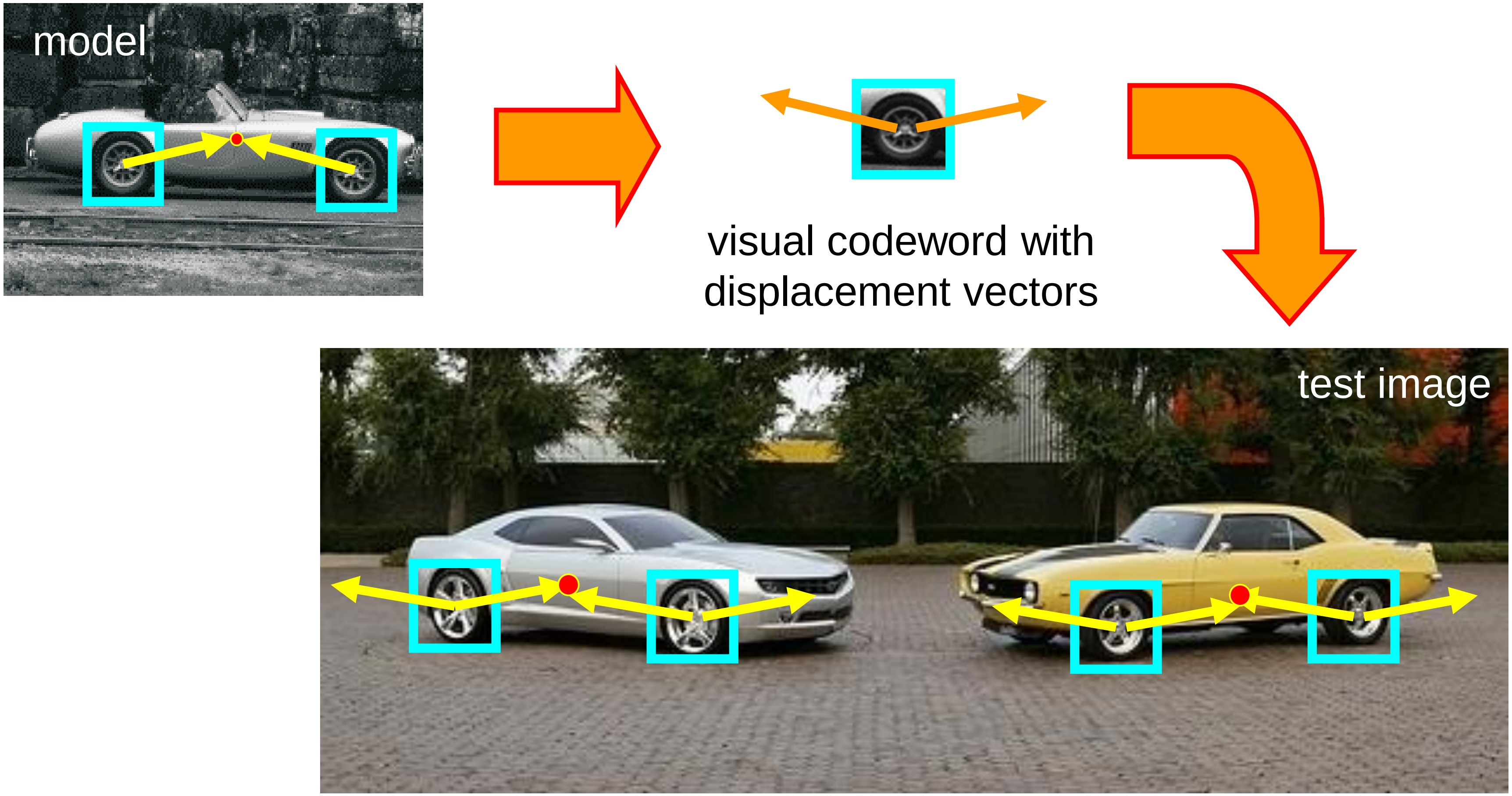
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

T

Hough Transformation for Alignment

Hough transform

- Recall: Generalized Hough transform

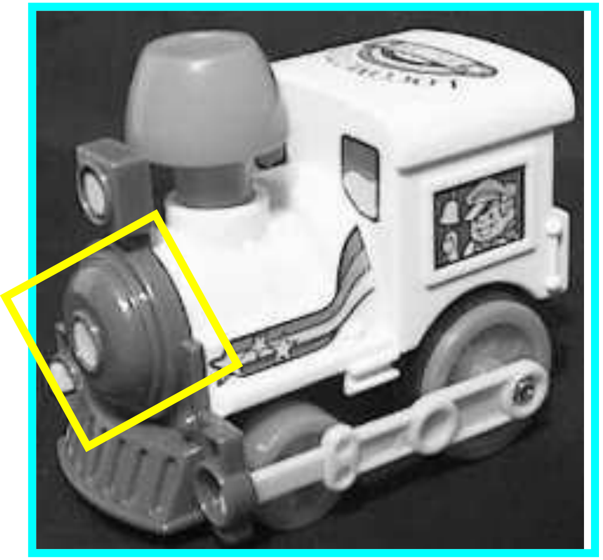


B. Leibe, A. Leonardis, and B. Schiele, [Combined Object Categorization and Segmentation with an Implicit Shape Model](), ECCV Workshop on Statistical Learning in Computer Vision 2004

Hough transform

- Suppose our features are adapted to scale and rotation
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation)

model

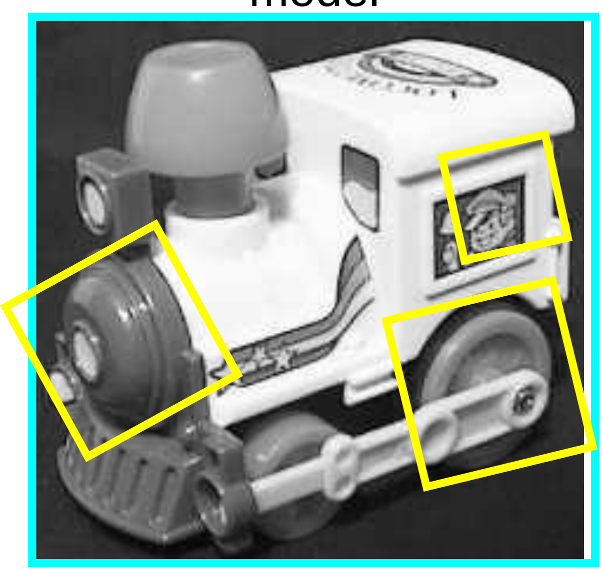


David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

Hough transform

- Suppose our features are adapted to scale and rotation
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation)
  - Of course, a hypothesis obtained from a single match is unreliable
  - Solution: let each match vote for its hypothesis in a Hough space with very coarse bins

model



David G. Lowe. **["Distinctive image features from scale-invariant keypoints."]** *IJCV* 60 (2), pp. 91-110, 2004.

Hough transform details (D. Lowe's system)

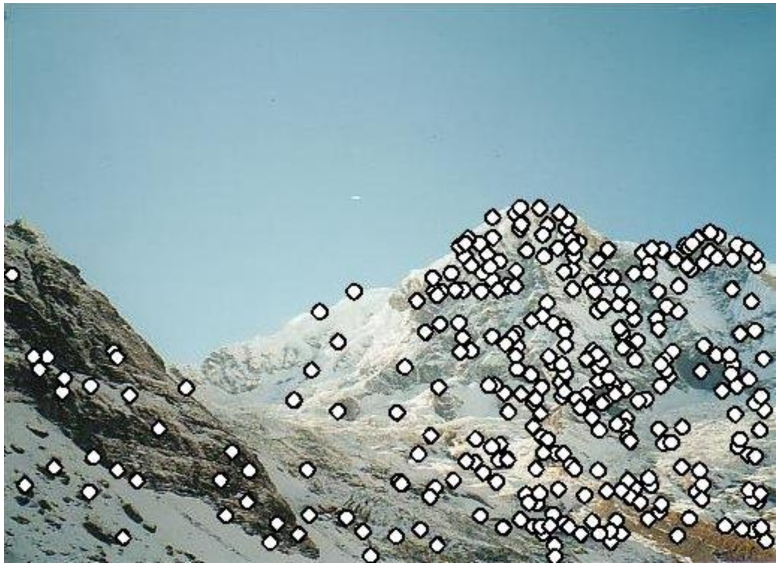
- **Modeling phase:** For each model feature, record 2D location, scale, and orientation of model (relative to normalized feature frame)
- **Test phase:** Let each match between a test and a model feature vote in a **4D Hough space**
  - Use broad bin sizes of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times image size for location
  - Vote for two closest bins in each dimension
- **Find all bins with at least three votes and perform geometric verification**
  - Estimate least squares *affine* transformation
  - Use stricter thresholds on transformation residual
  - Search for additional features that agree with the alignment

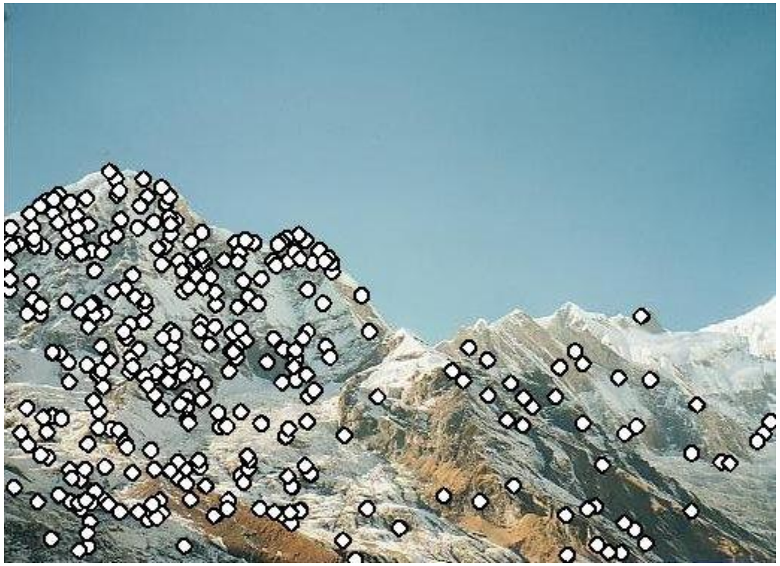
David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

Features for Image Alignment

Feature-based alignment outline

Feature-based alignment outline



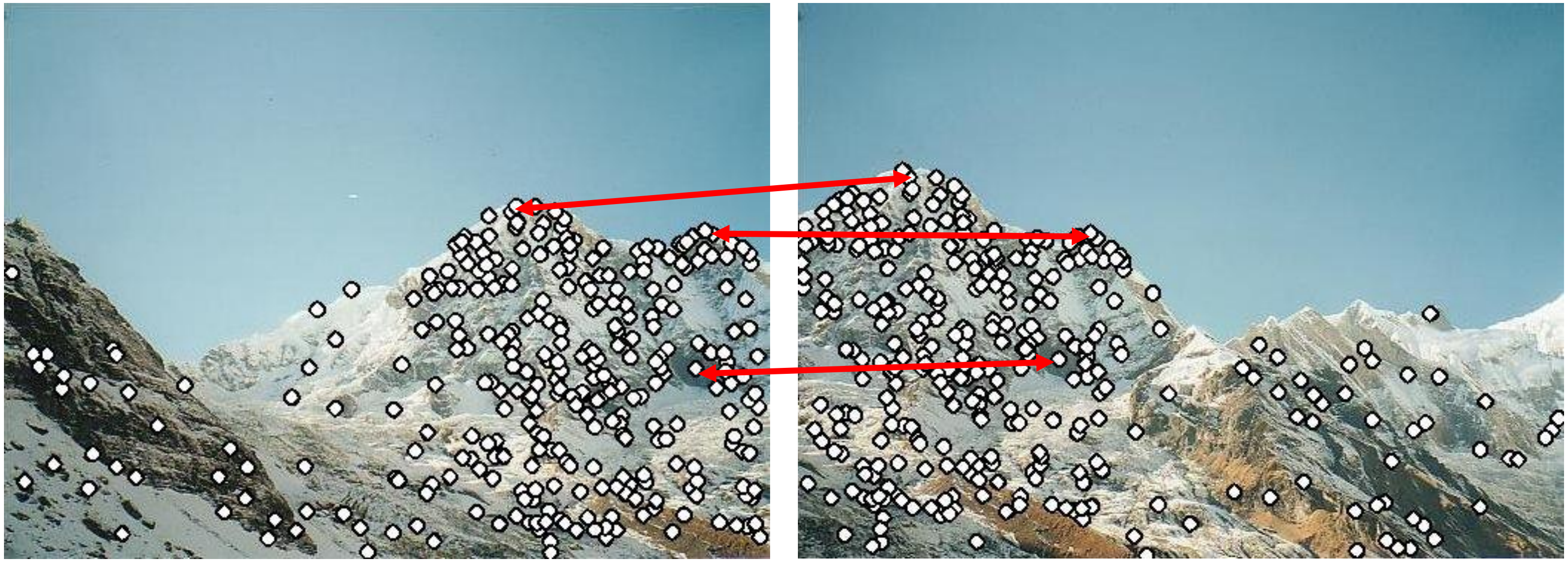


- Extract features

Feature-based alignment outline

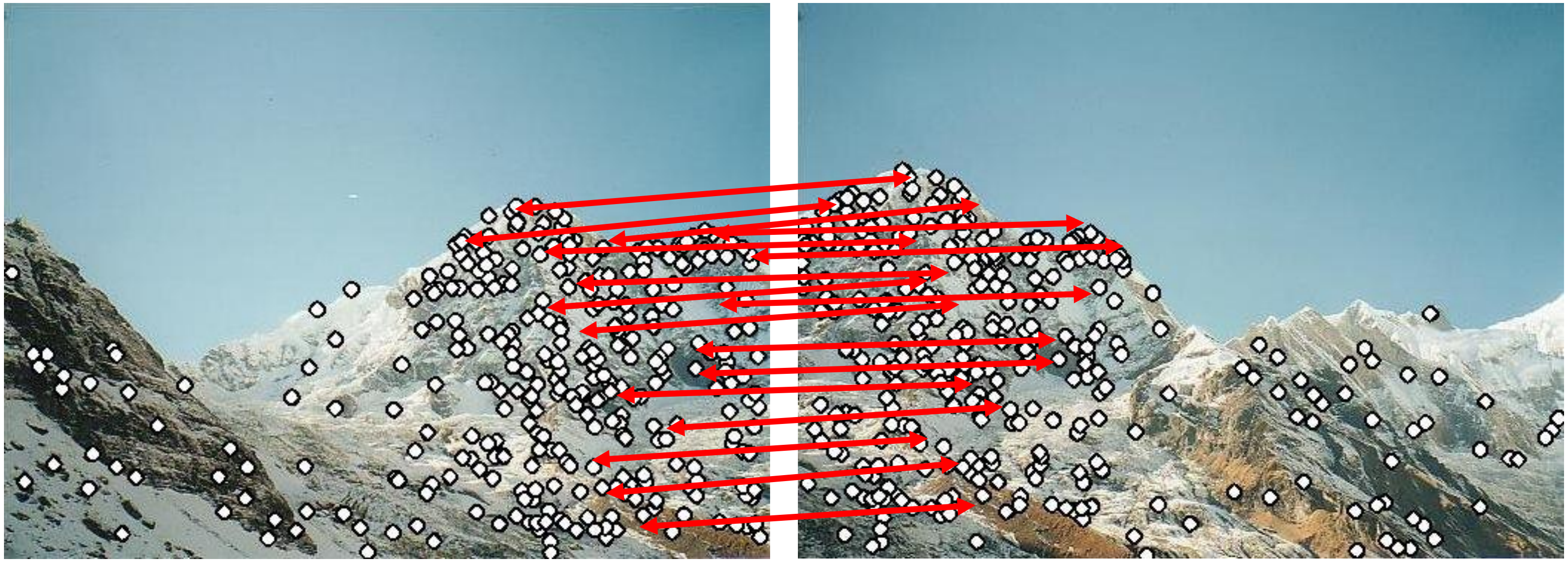
- Extract features
- Compute *putative matches*

Feature-based alignment outline



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

Feature-based alignment outline

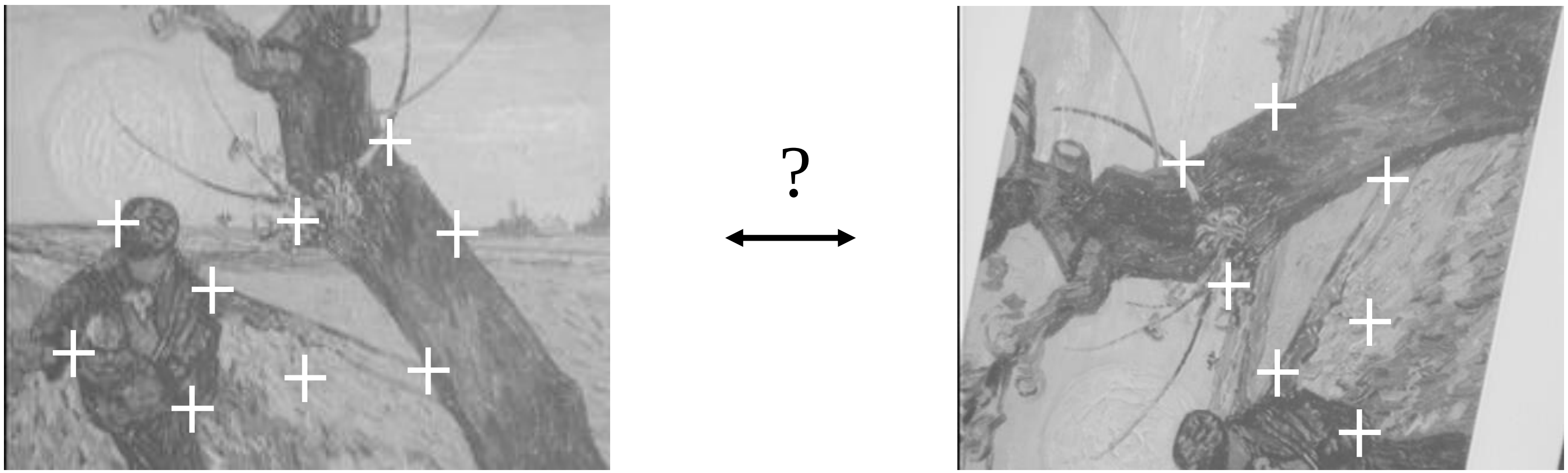


- Extract features
- Compute *putative matches*
- Loop:
  - ***Hypothesize* transformation *T***
  - ***Verify* transformation** (search for other matches consistent with *T*)

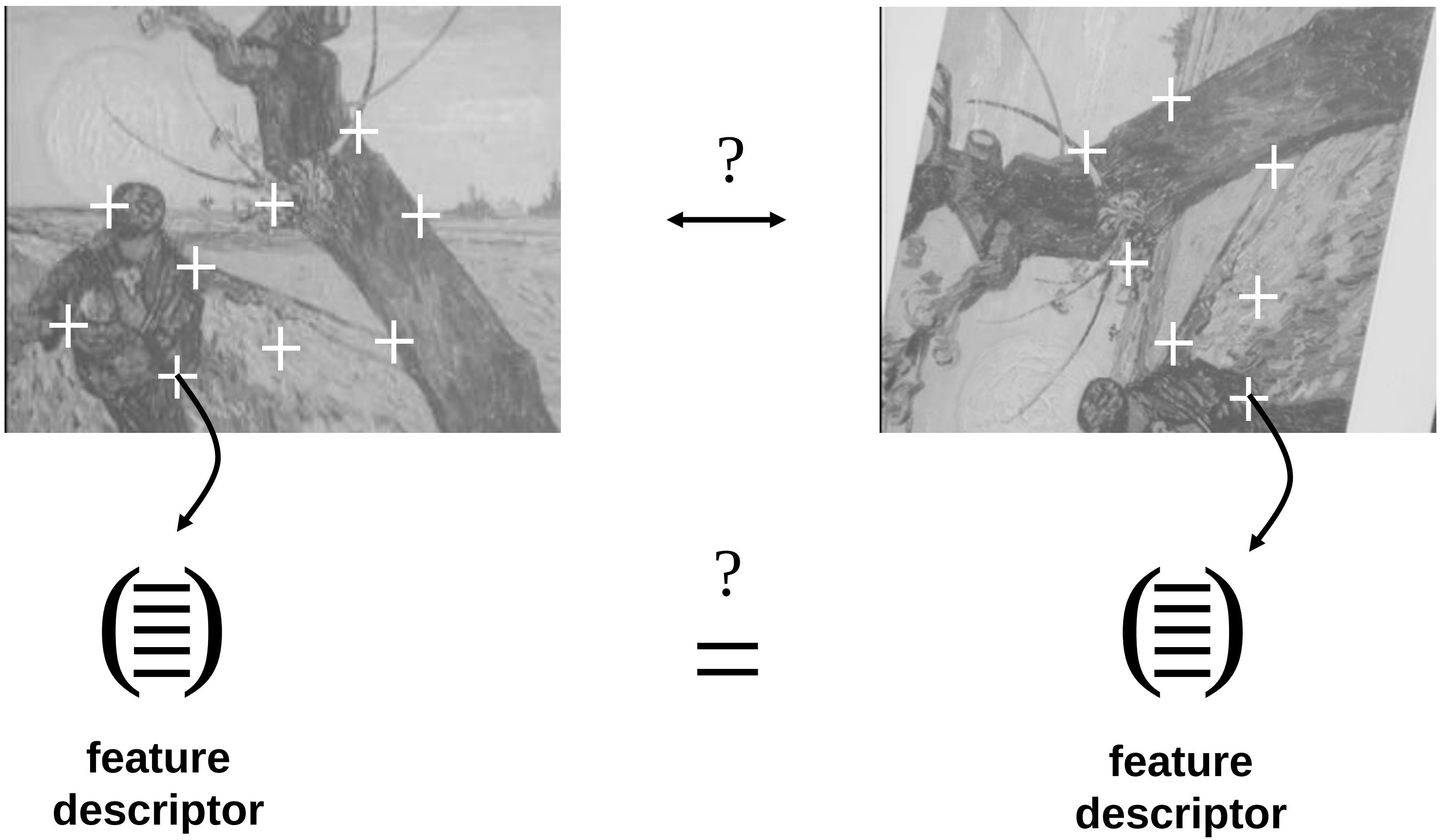
Feature-based alignment outline

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

Generating putative correspondences



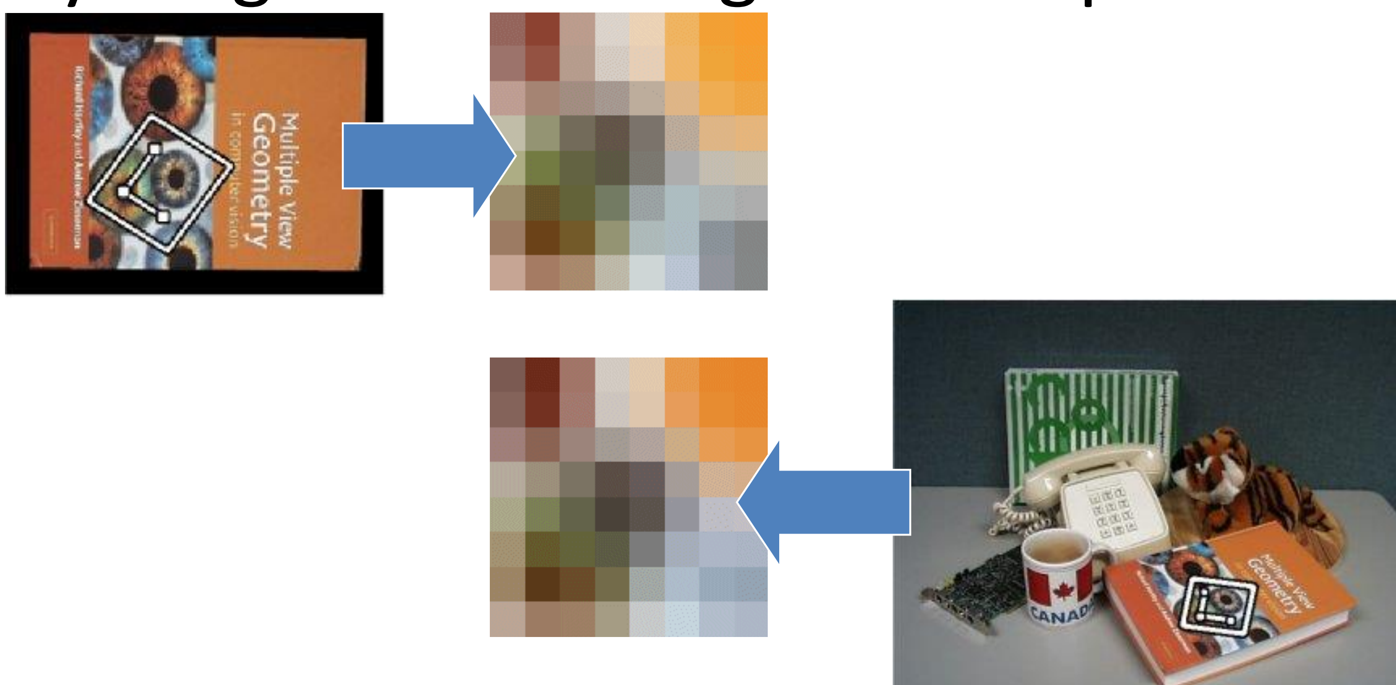
Generating putative correspondences



- Need to compare *feature descriptors* of local patches **surrounding interest points**

Feature descriptors

- Assuming the patches are already normalized (i.e., the local effect of the geometric transformation is factored out), how do we compute their similarity?
- Want invariance to intensity changes, noise, perceptually insignificant changes of the pixel pattern



Feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors?
  - **Sum of squared differences** (SSD)

$$\mathrm{SSD}(u,v) = \sum_i \left(u_i - v_i\right)^2$$

    - Not invariant to intensity change

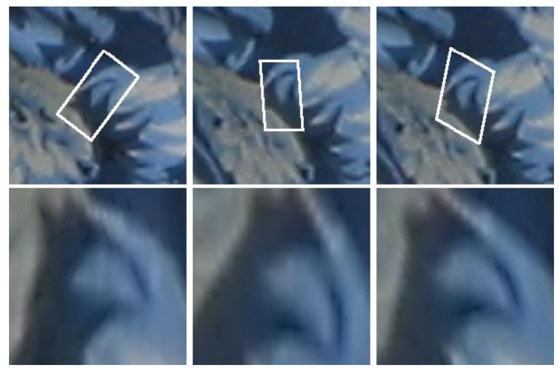
  - **Normalized correlation**

$$\rho(u,v) = \frac{\sum_i (u_i - \overline{u})(v_i - \overline{v})}{\sqrt{\left(\sum_j (u_j - \overline{u})^2\right)\left(\sum_j (v_j - \overline{v})^2\right)}}$$

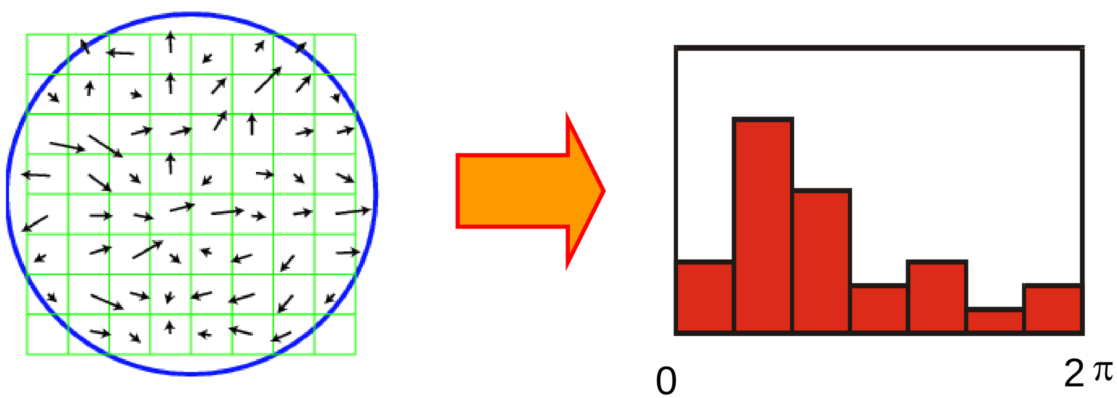
    - Invariant to affine intensity change

Feature descriptors

- **Disadvantage of patches as descriptors:**
  - Small shifts can affect matching score a lot

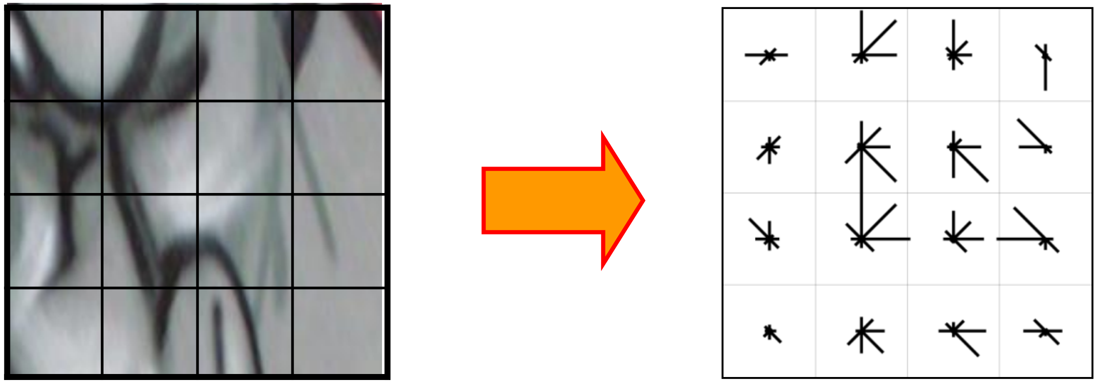


- Solution: histograms



Feature descriptors: SIFT

- **Descriptor computation**:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions



David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Feature descriptors: SIFT

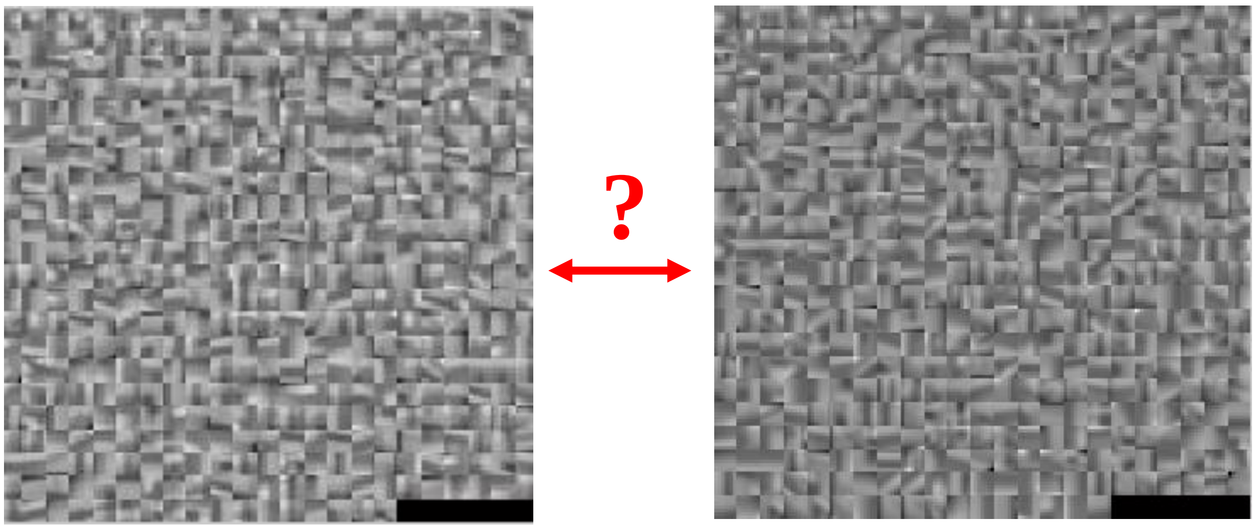
- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions

- Advantage over raw vectors of pixel values
  - Gradients less sensitive to illumination change
  - Pooling of gradients over the sub-patches achieves robustness to small shifts, but still preserves some spatial information

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

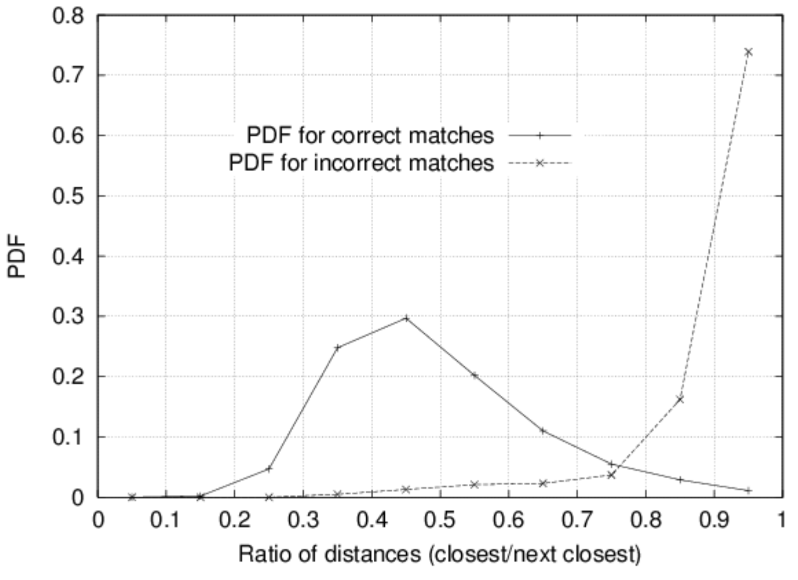
Feature matching

- Generating *putative matches*: for each patch in one image, **find a short list of patches** in the other image that could match it based solely on appearance



Feature space outlier rejection

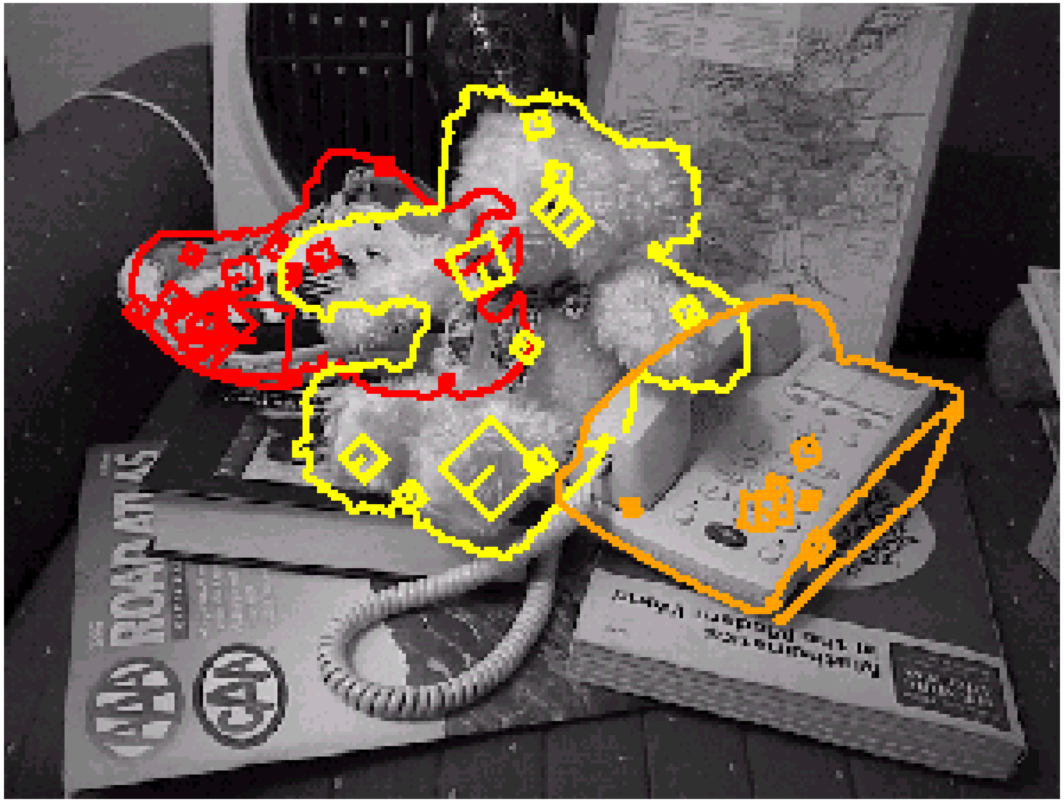
- How can we tell which putative matches are more reliable?
- Heuristic: compare distance of **nearest** neighbor to that of **second** nearest neighbor
  - Ratio of closest distance to second-closest distance will be *high* for features that are *not* distinctive



Threshold of 0.8 provides good separation

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Reading



David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

RANSAC Technique

What's RANSAC ?

- **RANSAC** is an abbreviation for "RANdom SAmple Consensus".

- It is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers.

- Non-deterministic algorithm.

http://en.wikipedia.org/wiki/RANSAC

Why RANSAC ?

- RANSAC can estimate a model which ignored outliers.

- Example:
  - To fit a line
    - Least Squares method:
      - Optimally fitted to all points including outliers.
    - RANSAC:
      - **Only computed from the inliers**.

Inliers ***vs.*** **Outliers**

http://en.wikipedia.org/wiki/RANSAC

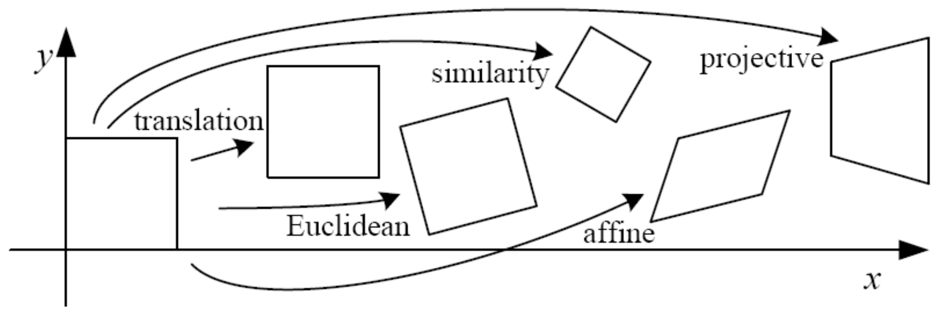
RANSAC

General version:

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

How big is *s*?

- For alignment, depends on the motion model
  - Here, each sample is a correspondence (pair of matching points)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\, I \mid t \,\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\, R \mid t \,\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\, sR \mid t \,\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\, A \,\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\, \tilde{H} \,\right]_{3\times 3}$	8	straight lines	

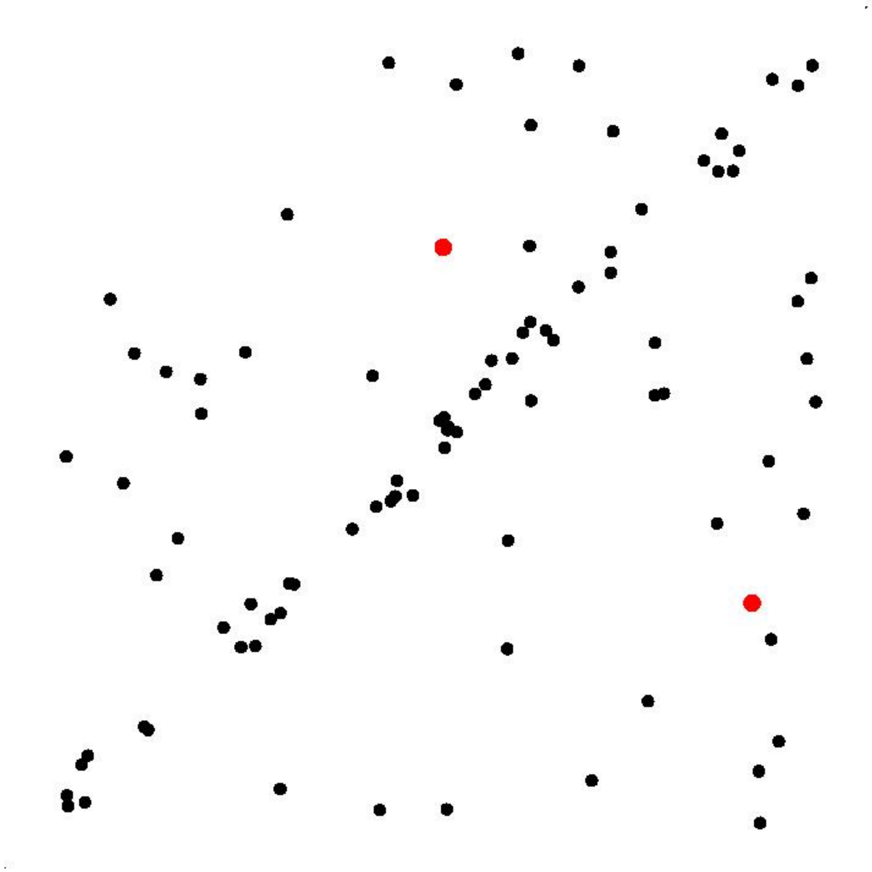
RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Parameters to tune
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios
  - We can often do better than brute-force sampling

Illustration of RANSAC

cmp.felk.cvut.cz/~matas/papers/presentations/viva.ppt

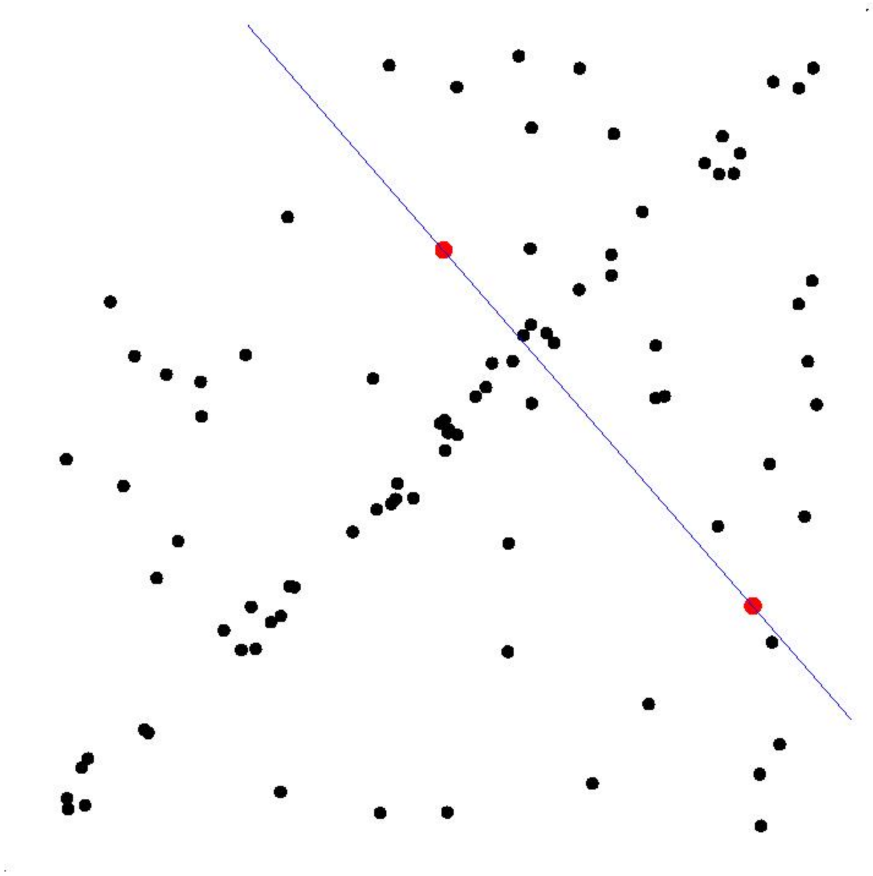
Illustration of RANSAC



- **Select sample of m points at random**

cmp.felk.cvut.cz/~matas/papers/presentations/viva.ppt

Illustration of RANSAC

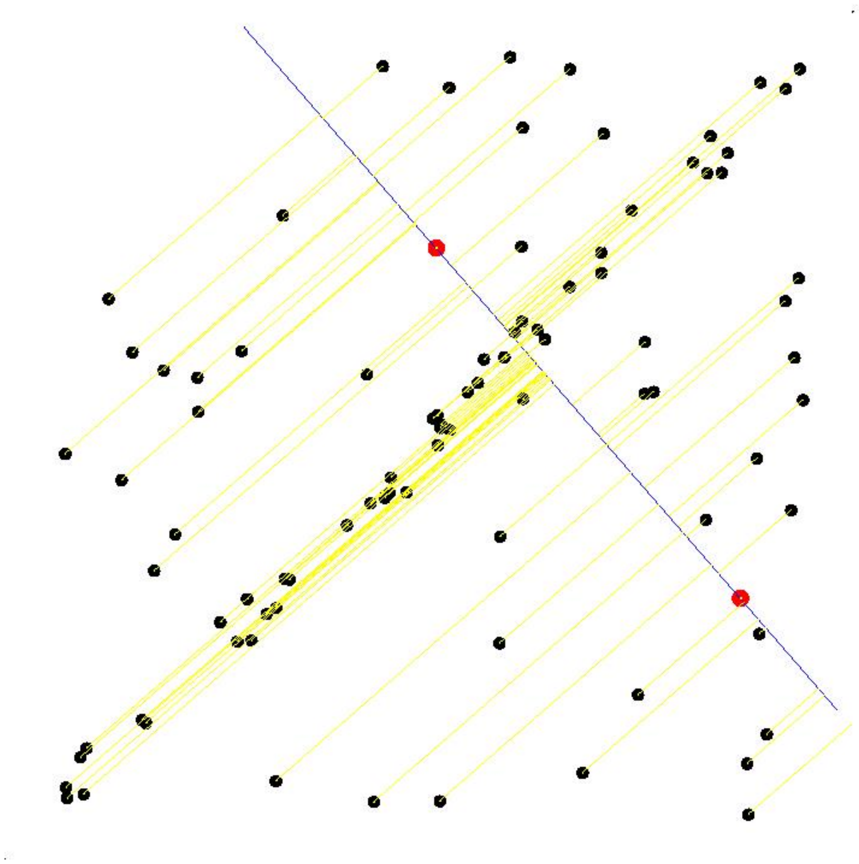


- Select sample of m points at random

- **Calculate model parameters that fit the data in the sample**

cmp.felk.cvut.cz/~matas/papers/presentations/viva.ppt

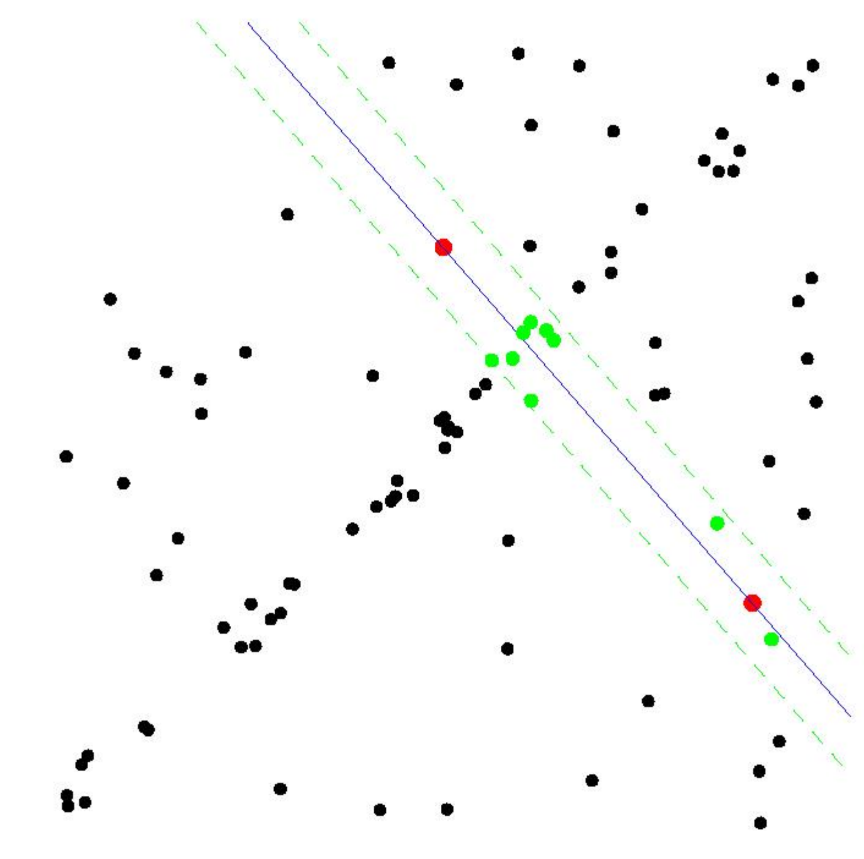
Illustration of RANSAC



- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- **Calculate error function for each data point**

cmp.felk.cvut.cz/~matas/papers/presentations/viva.ppt

Illustration of RANSAC



• Select sample of m points at random

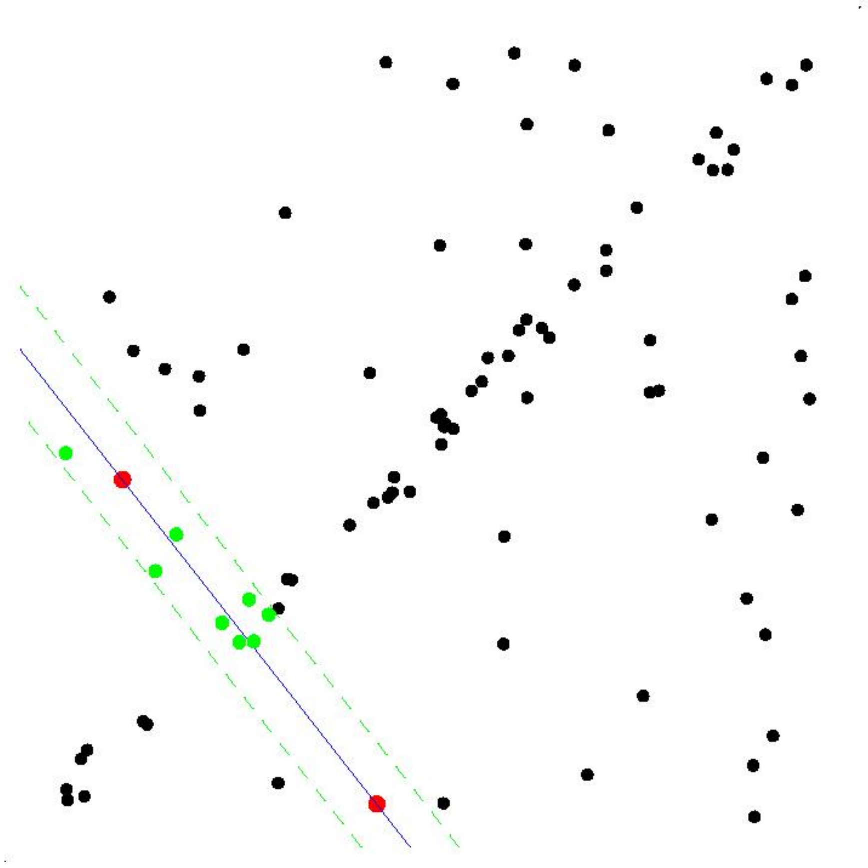
• Calculate model parameters that fit the data in the sample

• Calculate error function for each data point

• **Select data that support current hypothesis**

cmp.felk.cvut.cz/~matas/papers/presentations/viva.ppt

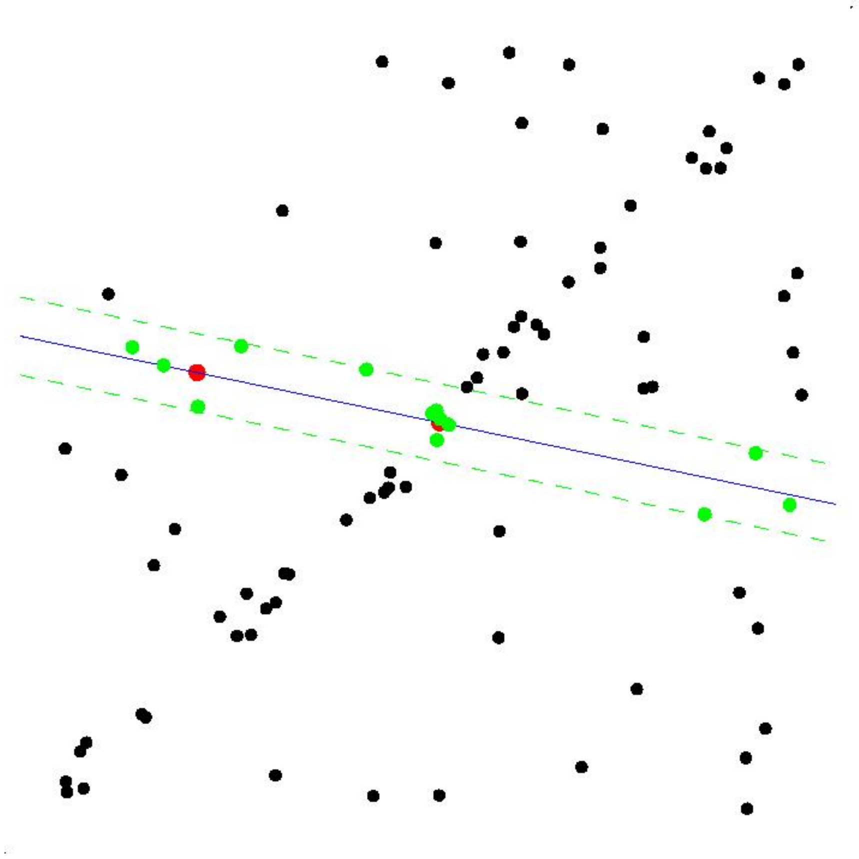
Illustration of RANSAC



- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

cmp.felk.cvut.cz/~matas/papers/presentations/viva.ppt

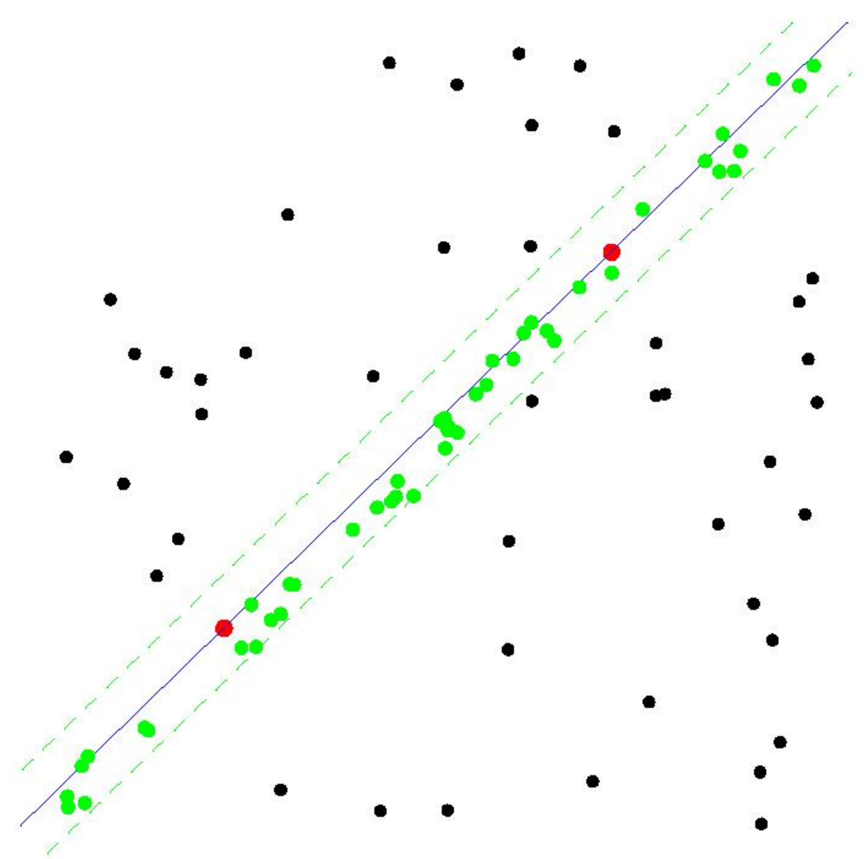
Illustration of RANSAC



- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

cmp.felk.cvut.cz/~matas/papers/presentations/viva.ppt

Illustration of RANSAC



ALL-INLIER SAMPLE

RANSAC time complexity

$$t = k(t_M + \overline{m}_s N)$$

k … number of samples drawn

N … number of data points

t_M … time to compute a single model

m_S … average number of models per sample

cmp.felk.cvut.cz/~matas/papers/presentations/viva.ppt

RANSAC Algorithm

- Input:
  - data: a set of observations
  - model: a model that can be fitted to data
  - n: the minimum number of data required to fit the model
  - k: the maximum number of iterations allowed in the algorithm
  - t: a threshold value for determining when a datum fits a model
  - d: the number of close data values required to assert that a model fits well to data
- Output:
  - best_model : model parameters which best fit the data (or nil if no good model is found)
  - best_consensus_set : data point from which this model has been estimated
  - best_error : the error of this model relative to the data

http://en.wikipedia.org/wiki/RANSAC

RANSAC Algorithm

```
iterations := 0
best_model := nil
best_consensus_set := nil
best_error := infinity
while iterations < k
    maybe_inliers := n randomly selected values from data
    maybe_model := model parameters fitted to maybe_inliers
    consensus_set := maybe_inliers

    for every point in data not in maybe_inliers
        if point fits maybe_model with an error smaller than t
            add point to consensus_set

    if the number of elements in consensus_set is > d
        (this implies that we may have found a good model,
        now test how good it is)
        better_model := model parameters fitted to all points in consensus_set
        this_error := a measure of how well better_model fits these points
        if this_error < best_error
            (we have found a model which is better than any of the previous one
            keep it until a better one is found)
            best_model := better_model
            best_consensus_set := consensus_set
            best_error := this_error

    increment iterations

return best_model, best_consensus_set, best_error
```


http://en.wikipedia.org/wiki/RANSAC

Parameters

$$1 - p = \left(1 - w^{n}\right)^{k}$$

$$k = \frac{\log\left(1 - p\right)}{\log(1 - w^{n})}$$

k: Iteration times.
n: Selected points in one iteration.
p: Probability in k iteration selects only inliers.
w: Probability of a point which is a inlier.

In general, the p is unknown. If we fixed p, the k increased when n increased.

http://en.wikipedia.org/wiki/RANSAC

RANSAC for Image Alignment

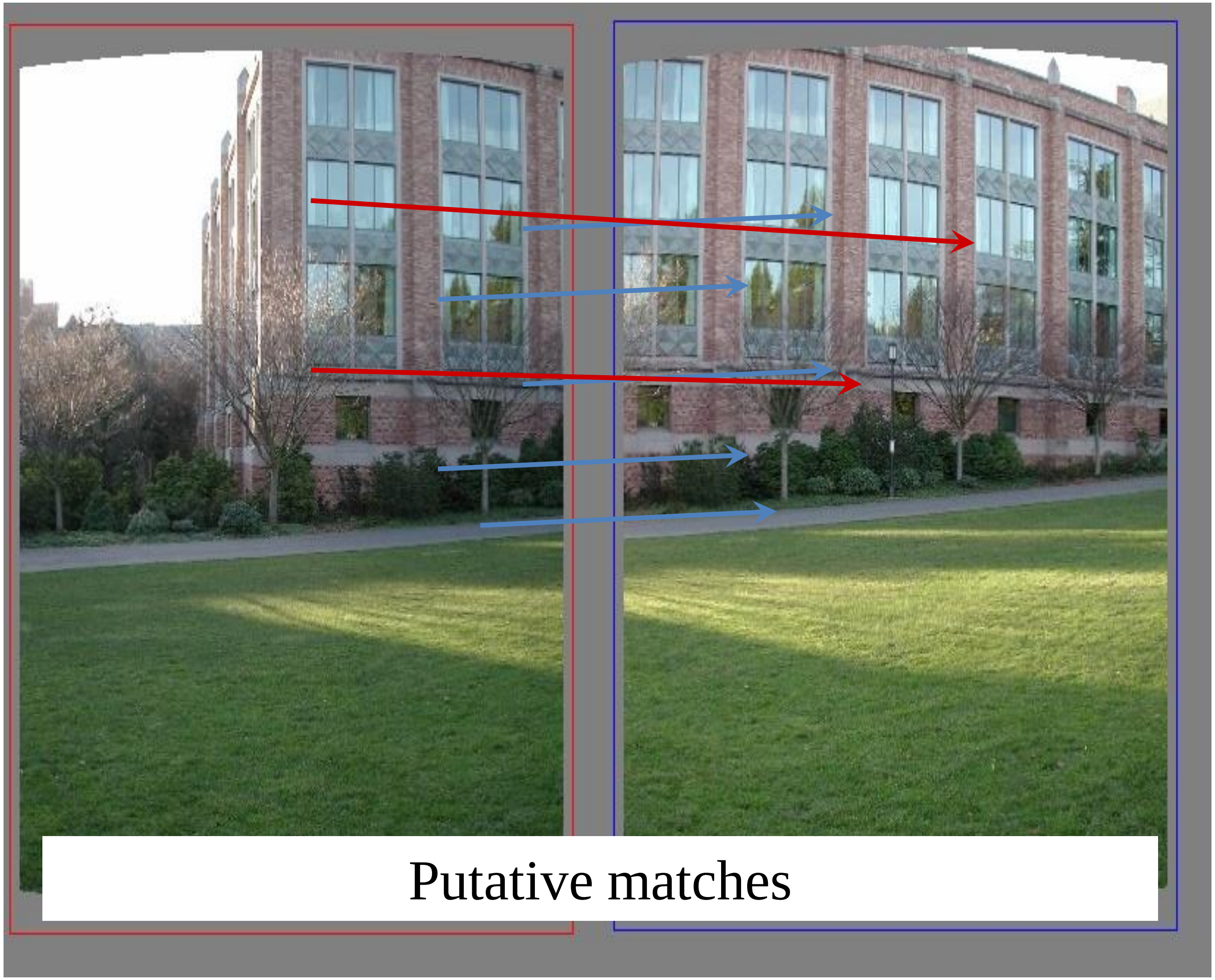
RANSAC

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

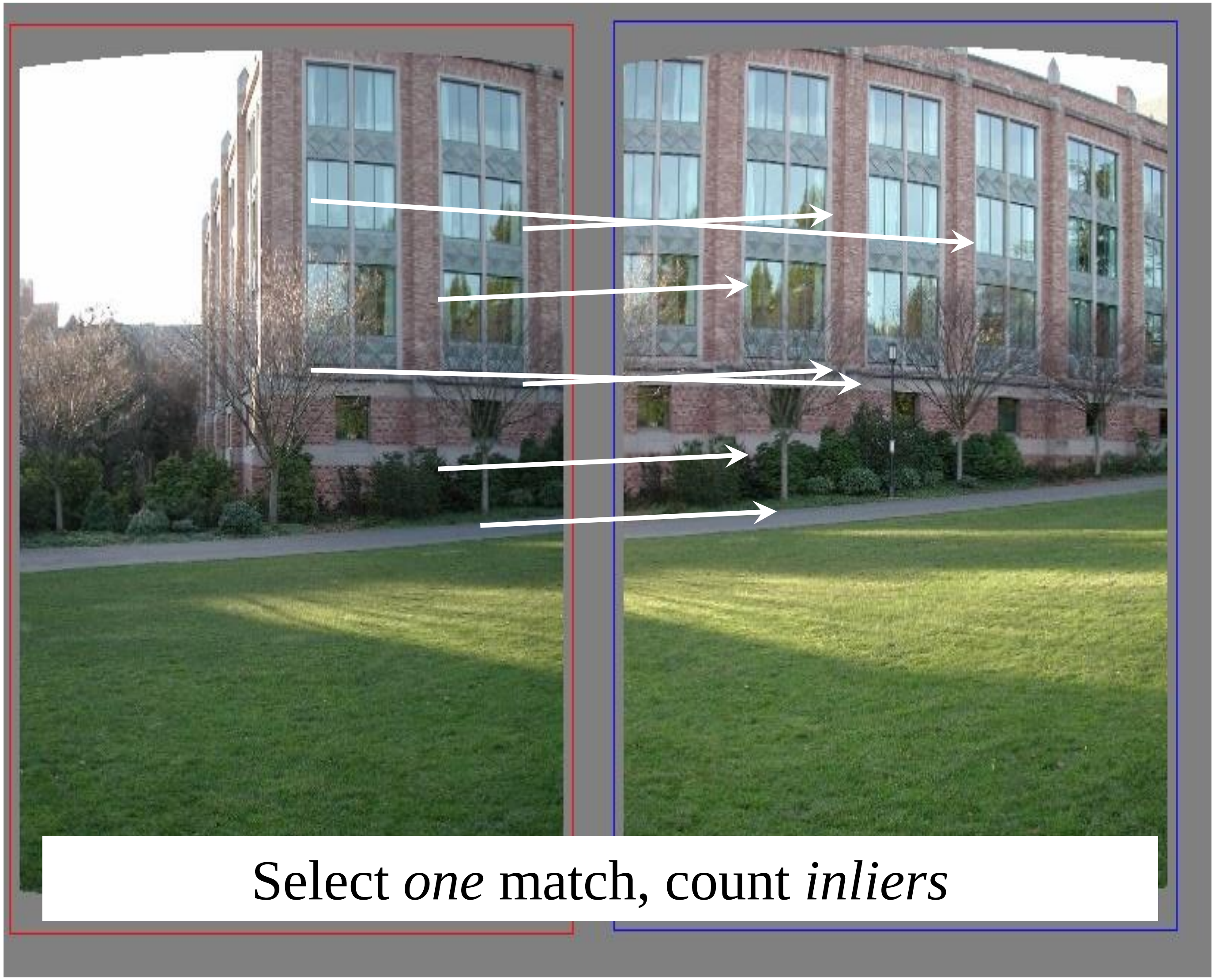
- **Keep the transformation with the largest number of inliers**

RANSAC example: Translation



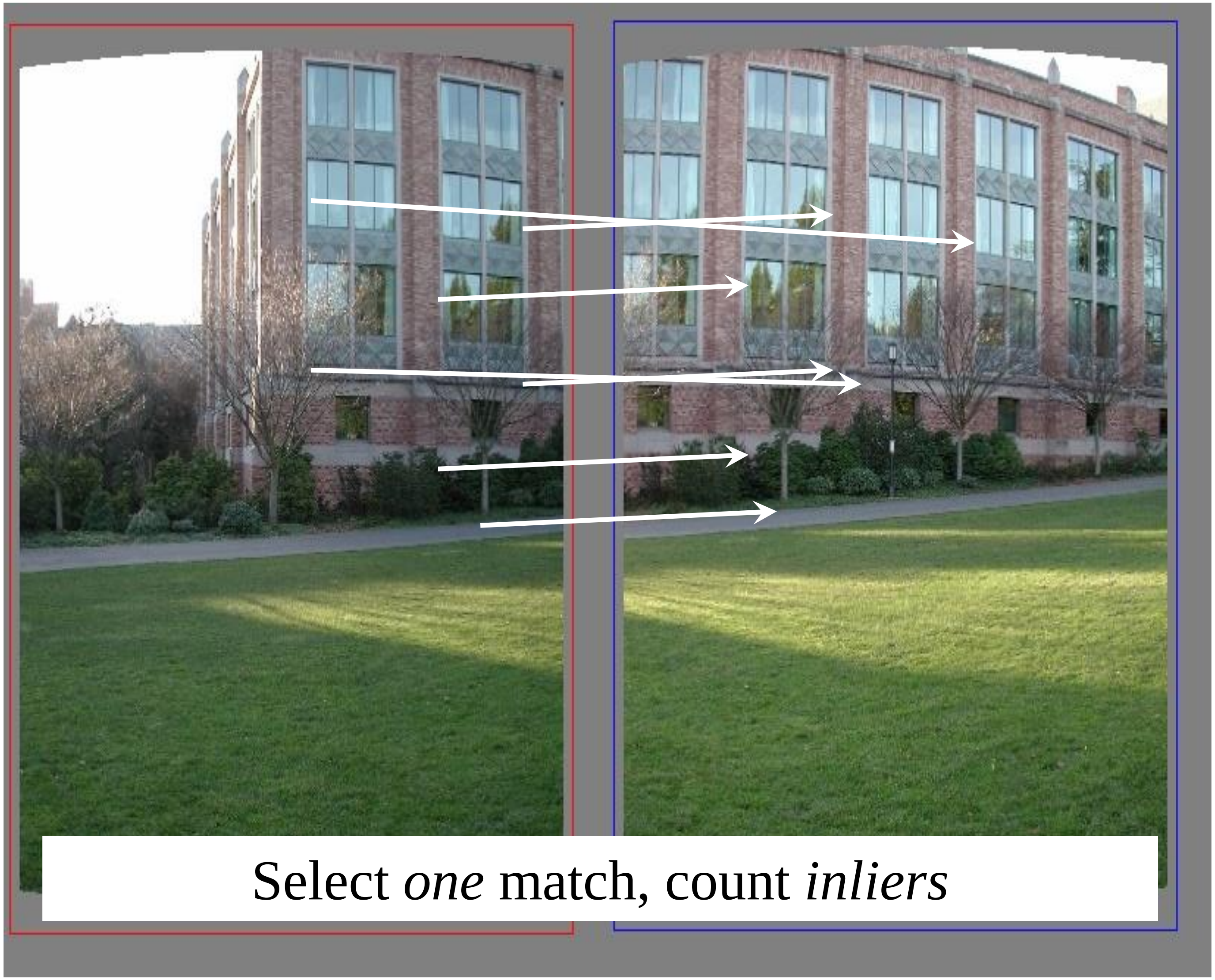
Putative matches

RANSAC example: Translation



Select *one* match, count *inliers*

RANSAC example: Translation



RANSAC example: Translation

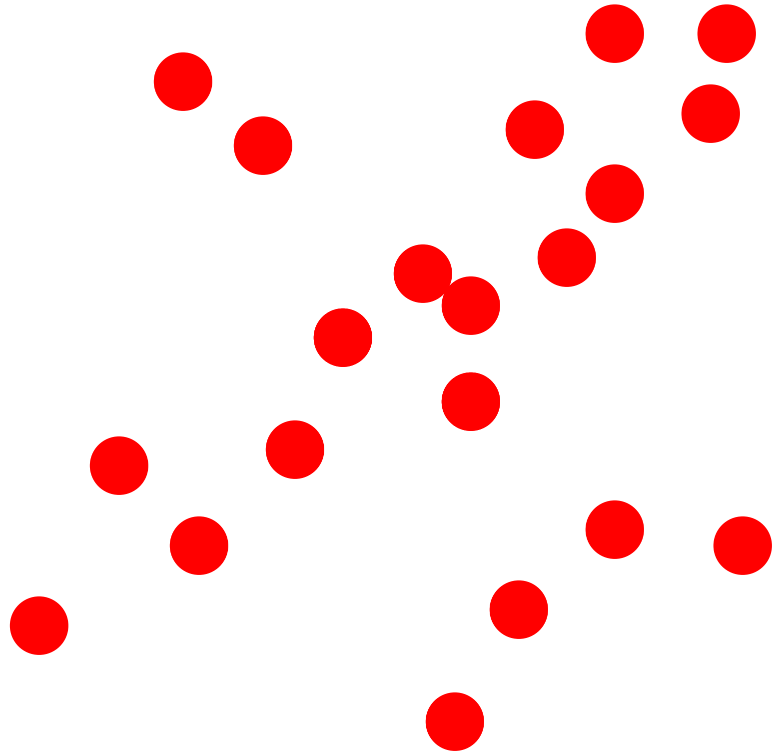
Problem with RANSAC

- In many practical situations, the **percentage of outliers** (incorrect putative matches) is often very high (90% or above)
- **Alternative strategy**: Hough transform

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



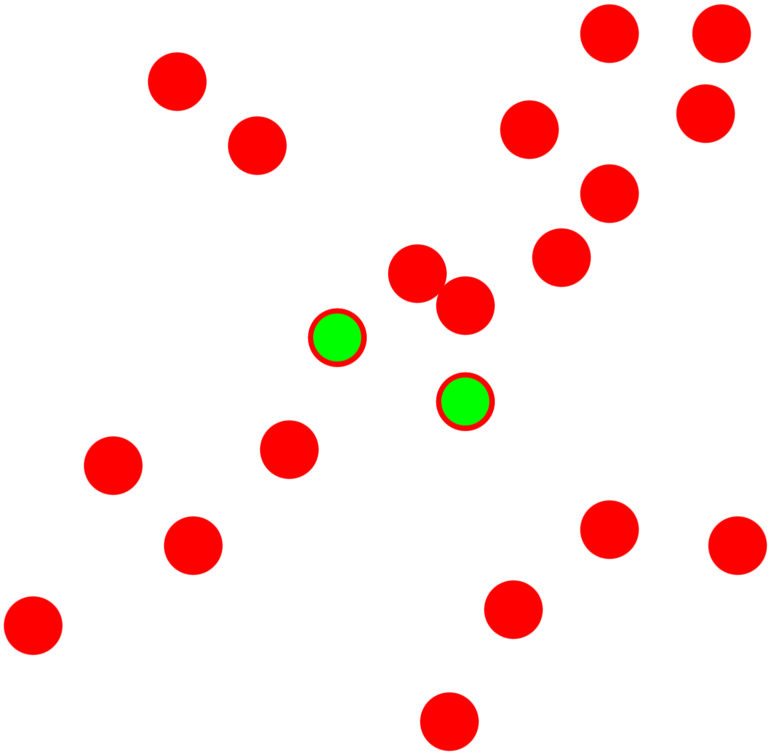
Algorithm:

1. **Sample** (randomly) the number of points **required to fit the model**
2. **Solve** for model parameters using samples
3. **Score** by the **fraction of inliers** within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



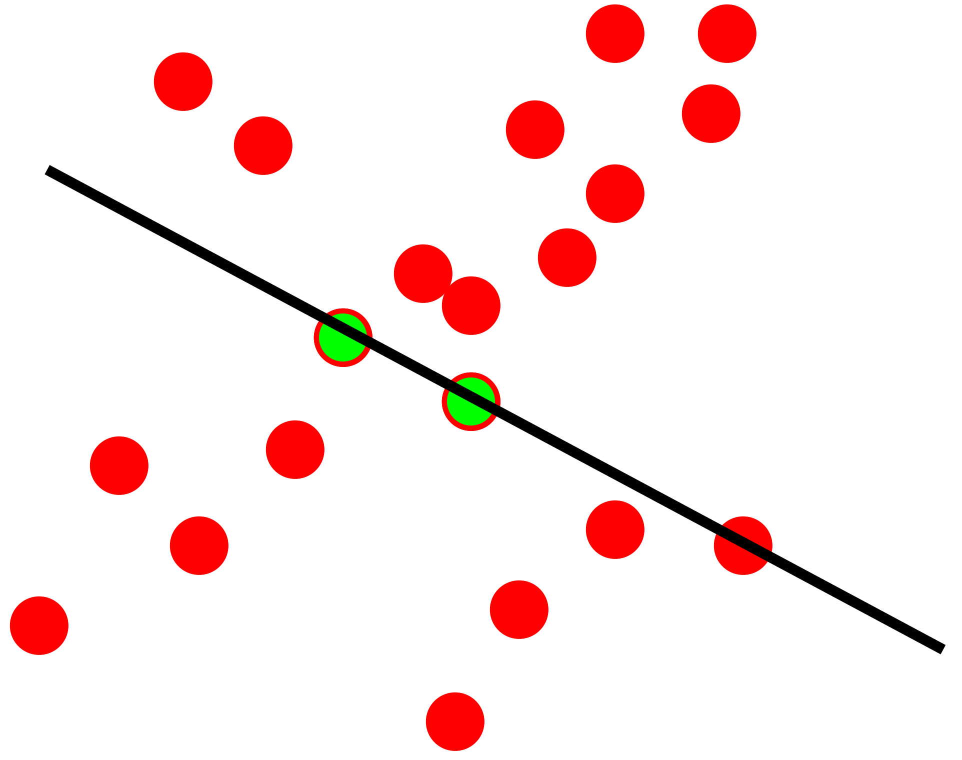
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC



Line fitting example

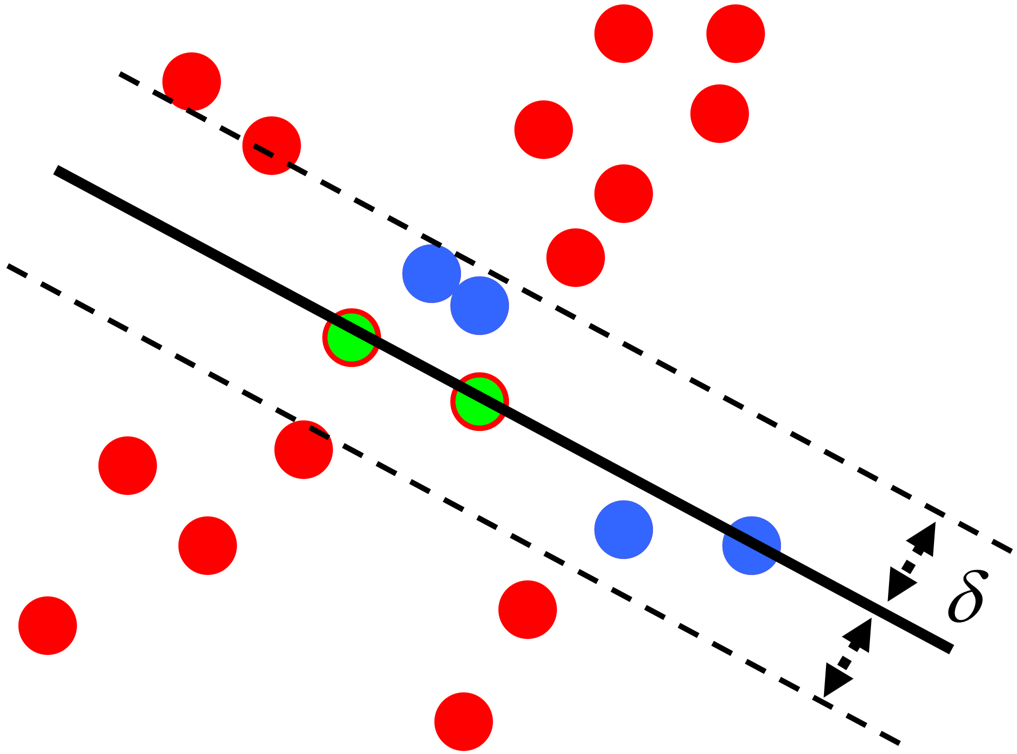
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



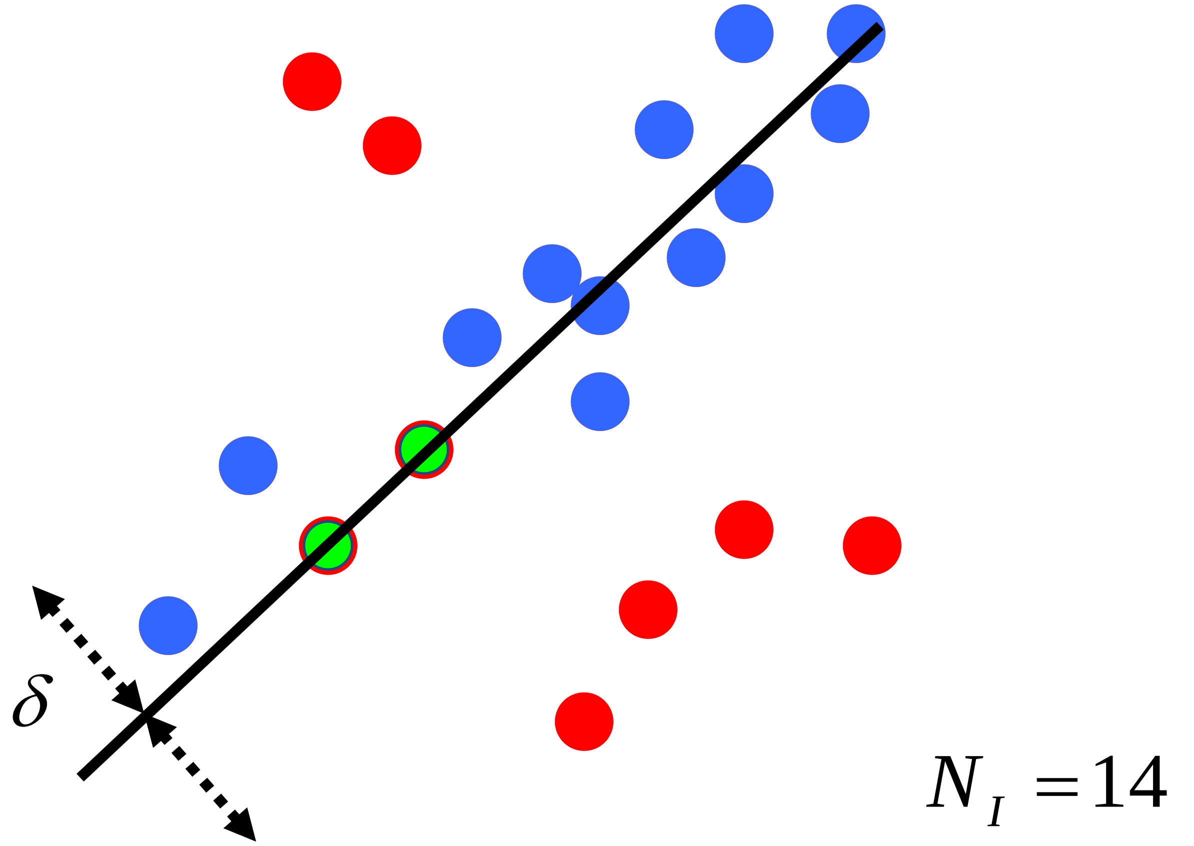
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \log(1-p) / \log\left(1-(1-e)^s\right)$$

	proportion of outliers *e*						
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

For p = 0.99

modified from M. Pollefeys

RANSAC conclusions

Good

- Robust to outliers
- Applicable for larger number of model parameters than Hough transform
- Optimization parameters are easier to choose than Hough transform

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not good for getting multiple fits

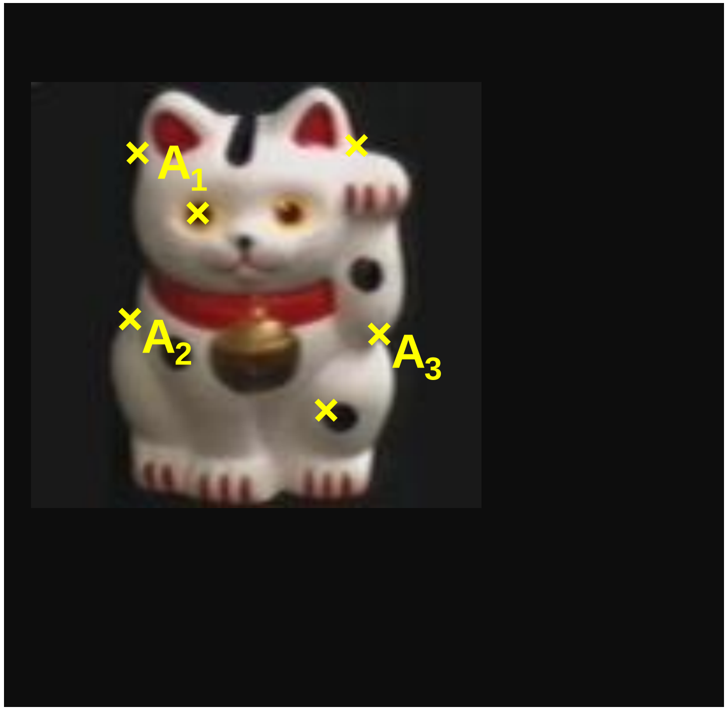
Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

Alignment

- Alignment: find parameters of model that maps one set of points to another
- Typically want to solve for a global transformation that accounts for *most* true correspondences
- Difficulties
  - Noise (typically 1-3 pixels)
  - Outliers (often 50%)
  - Many-to-one matches or multiple objects

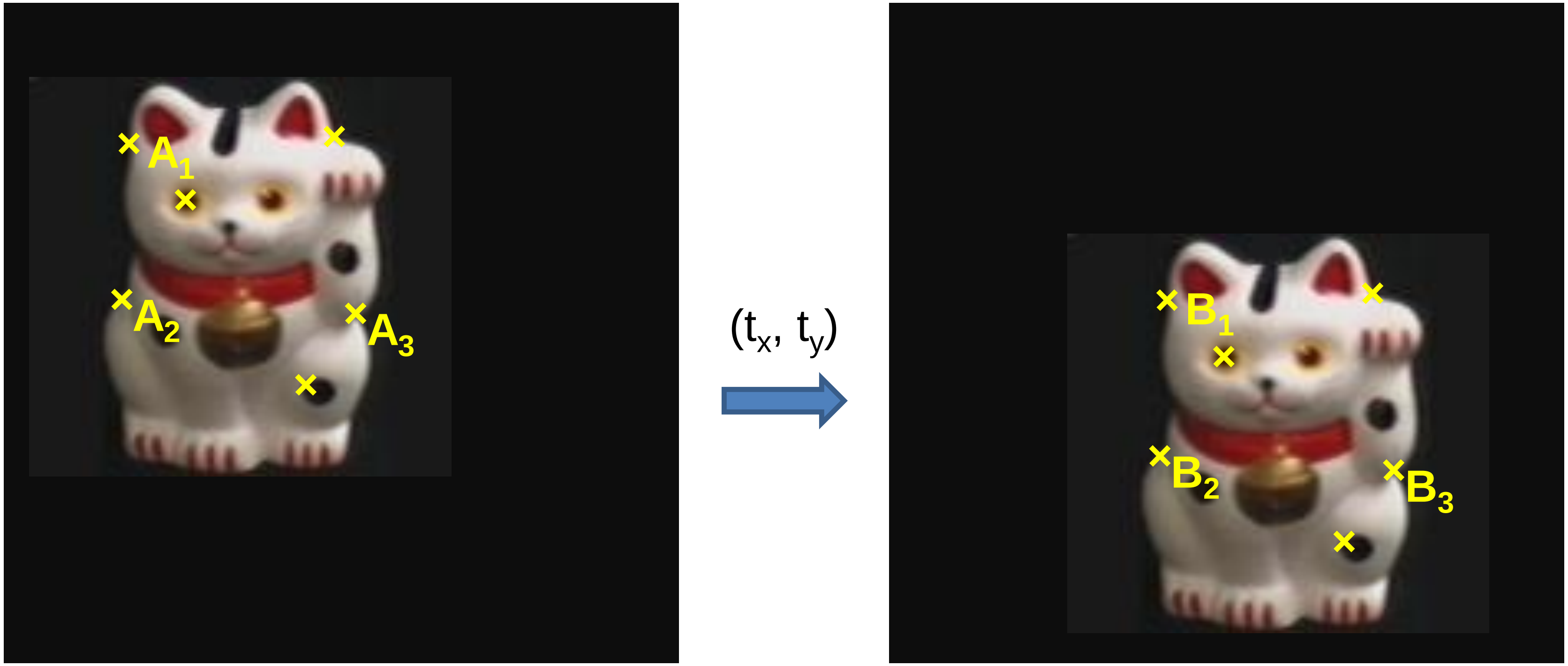
Example: solving for translation



Given matched points in {A} and {B}, **estimate the translation of the object**

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



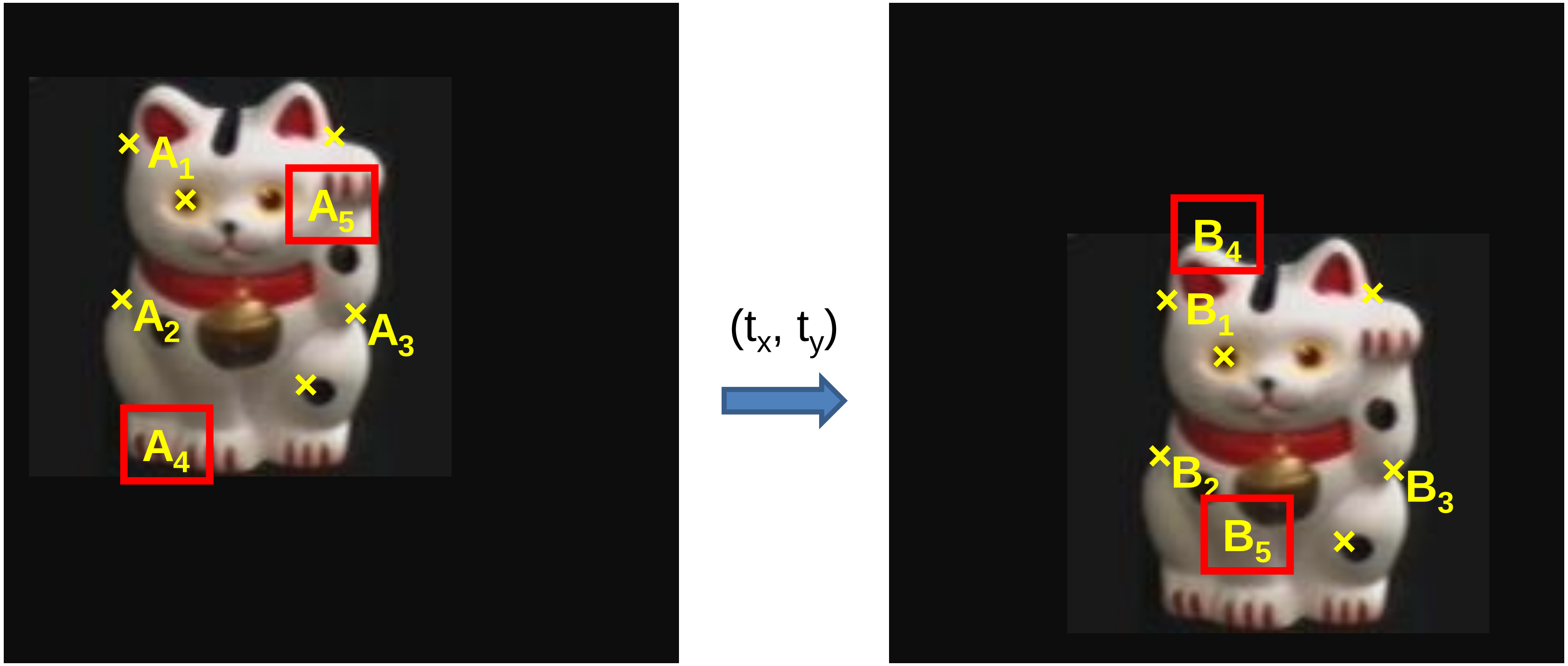
Least squares solution

1. Write down objective function
2. Derived solution
   a) Compute derivative
   b) Compute solution
3. Computational solution
   a) Write in form Ax=b
   b) Solve using pseudo-inverse or eigenvalue decomposition

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

Example: solving for translation



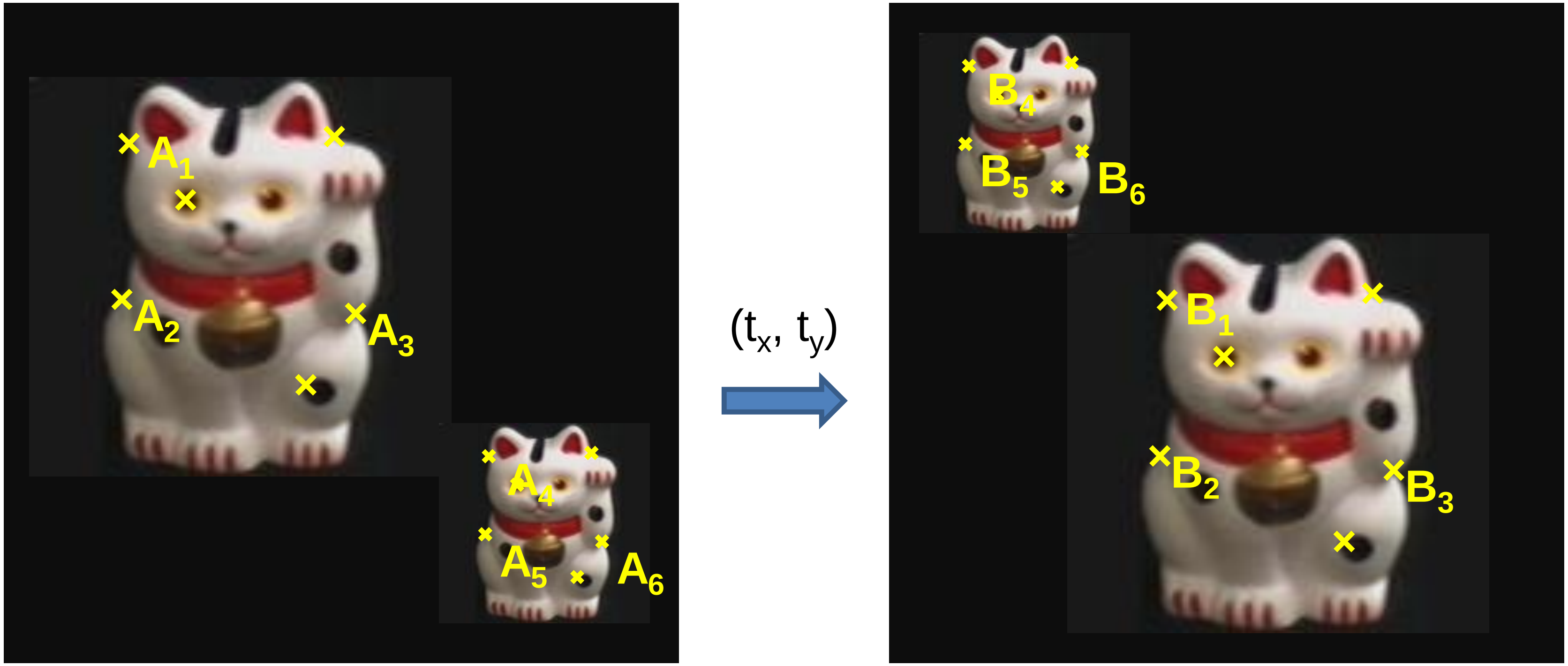
Problem: outliers

RANSAC solution

1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



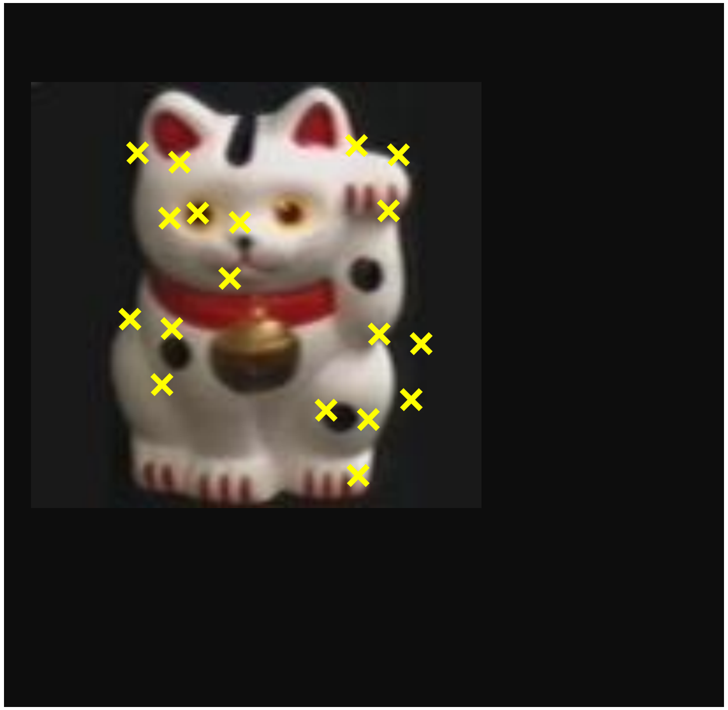
Problem: outliers, multiple objects, and/or many-to-one matches

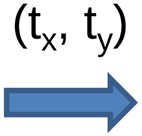
Hough transform solution

1. Initialize a grid of parameter values
2. Each matched pair casts a vote for consistent values
3. Find the parameters with the most votes
4. Solve using least squares with inliers

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation





Problem: no initial guesses for correspondence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

ICP for Image Alignment

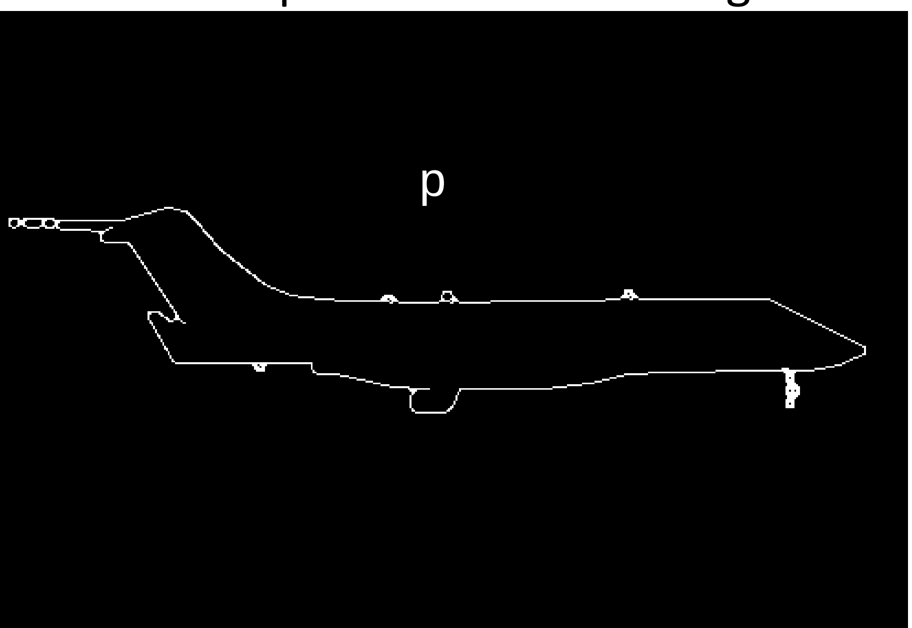
Iterative Closest Points (ICP) Algorithm

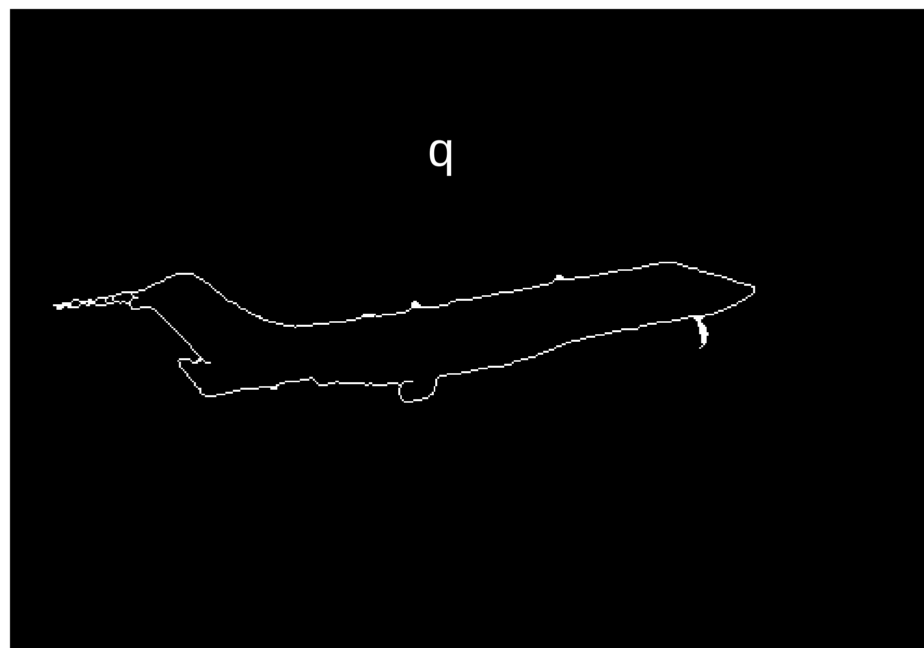
Goal: estimate transform between two dense sets of points

1. **Initialize** transformation (e.g., compute difference in means and scale)
2. **Assign** each point in {Set 1} to its nearest neighbor in {Set 2}
3. **Estimate** transformation parameters
   – e.g., least squares or robust least squares
4. **Transform** the points in {Set 1} using estimated parameters
5. **Repeat** steps 2-4 until change is very small

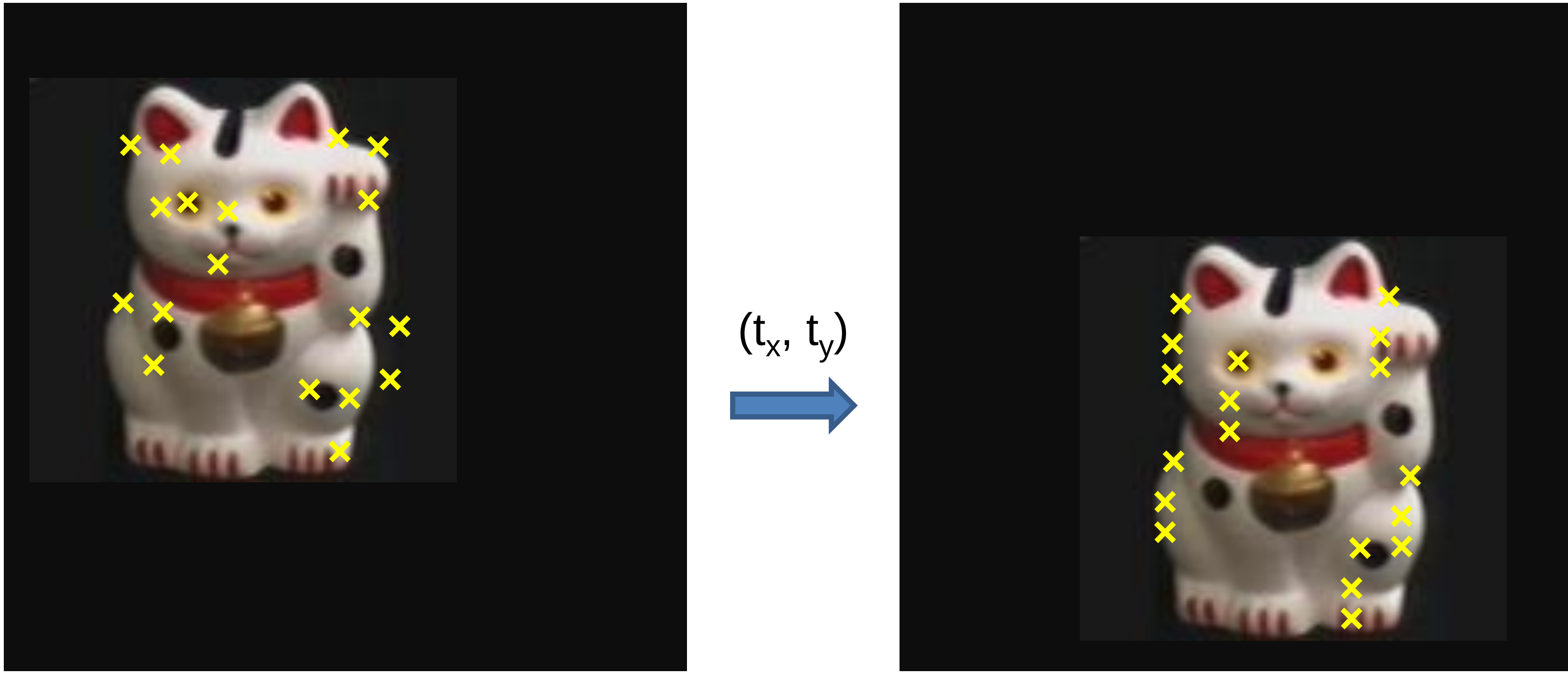
Example: **aligning boundaries**

1. Extract edge pixels $p_1..p_n$ and $q_1..q_m$
2. Compute initial transformation (e.g., compute translation and scaling by center of mass, variance within each image)
3. Get nearest neighbors: for each point p_i find corresponding $\text{match(i)} = \underset{j}{\text{argmin}}\, dist(pi, qj)$
4. Compute transformation ***T*** based on matches
5. Warp points ***p*** according to ***T***
6. Repeat 3-5 until convergence





Example: **solving for translation**



Problem: no initial guesses for correspondence

ICP solution

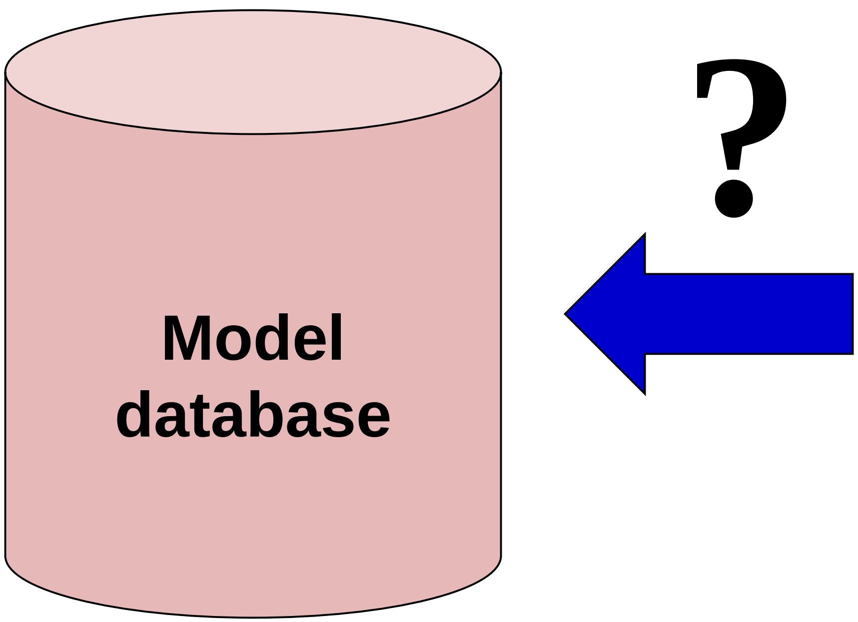
1. Find nearest neighbors for each point
2. Compute transform using matches
3. Move points using transform
4. Repeat steps 1-3 until convergence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Applications of Feature Matching

Scalability: Alignment to large databases

- What if we need to align a test image with thousands or millions of images in a model database?
  - Efficient putative match generation
    - Fast nearest neighbor search, inverted indexes

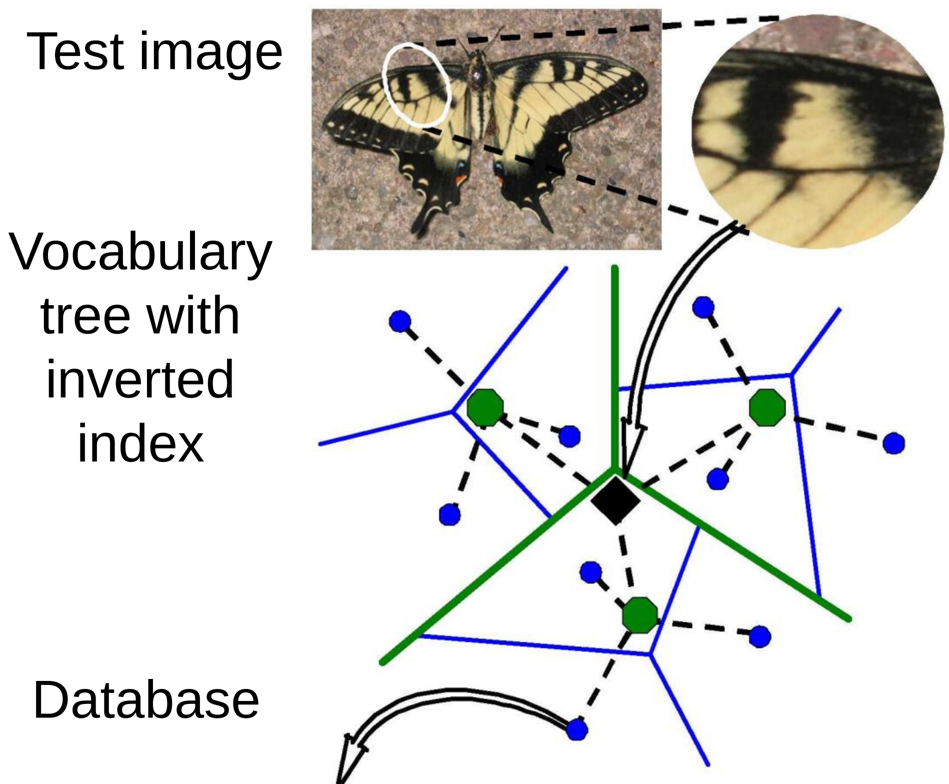


Test image

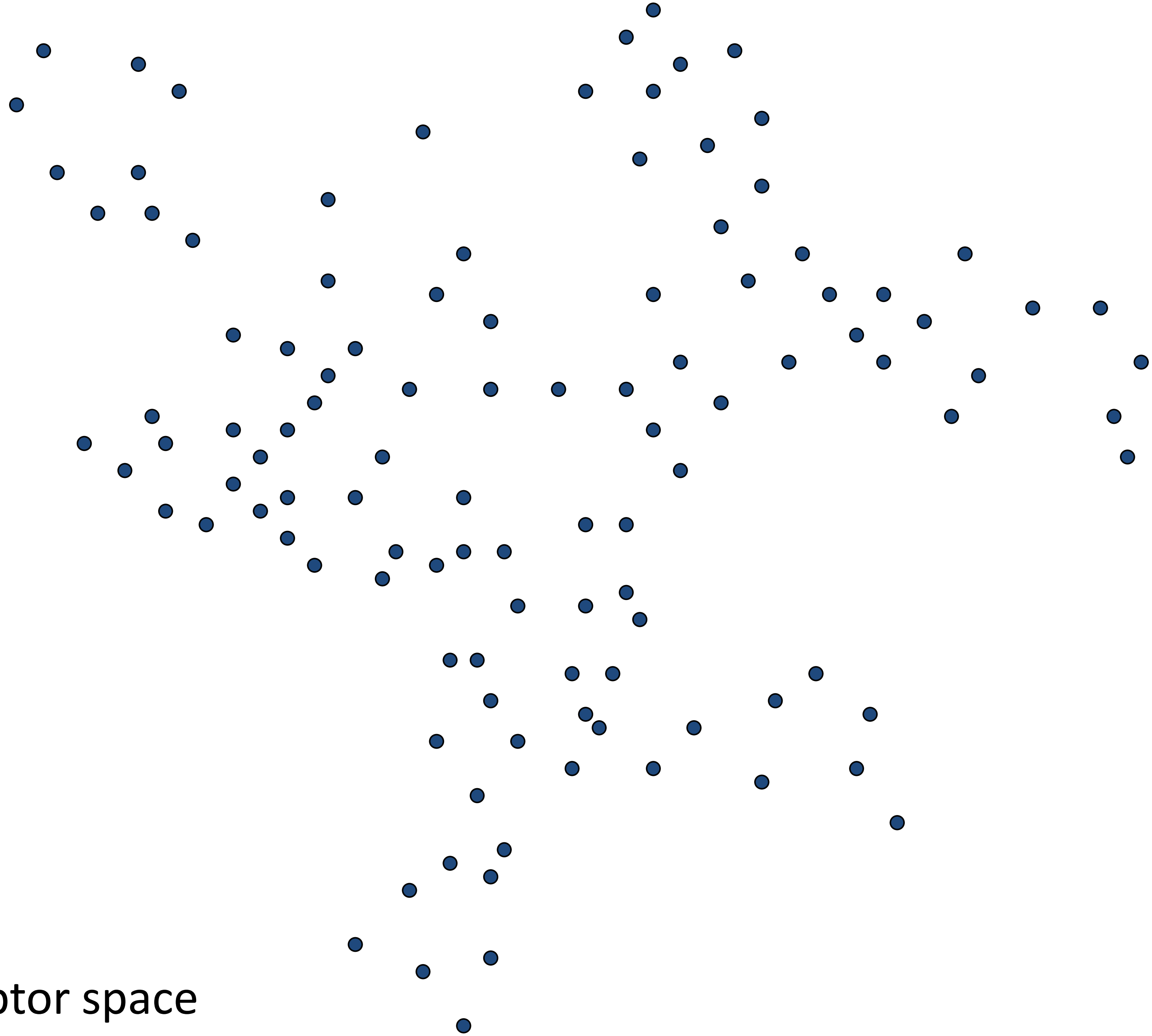
Scalability of SIFT Matching

Scalability: Alignment to large databases

- What if we need to align a test image with thousands or millions of images in a model database?
  - Efficient putative match generation
    - Fast nearest neighbor search, inverted indexes

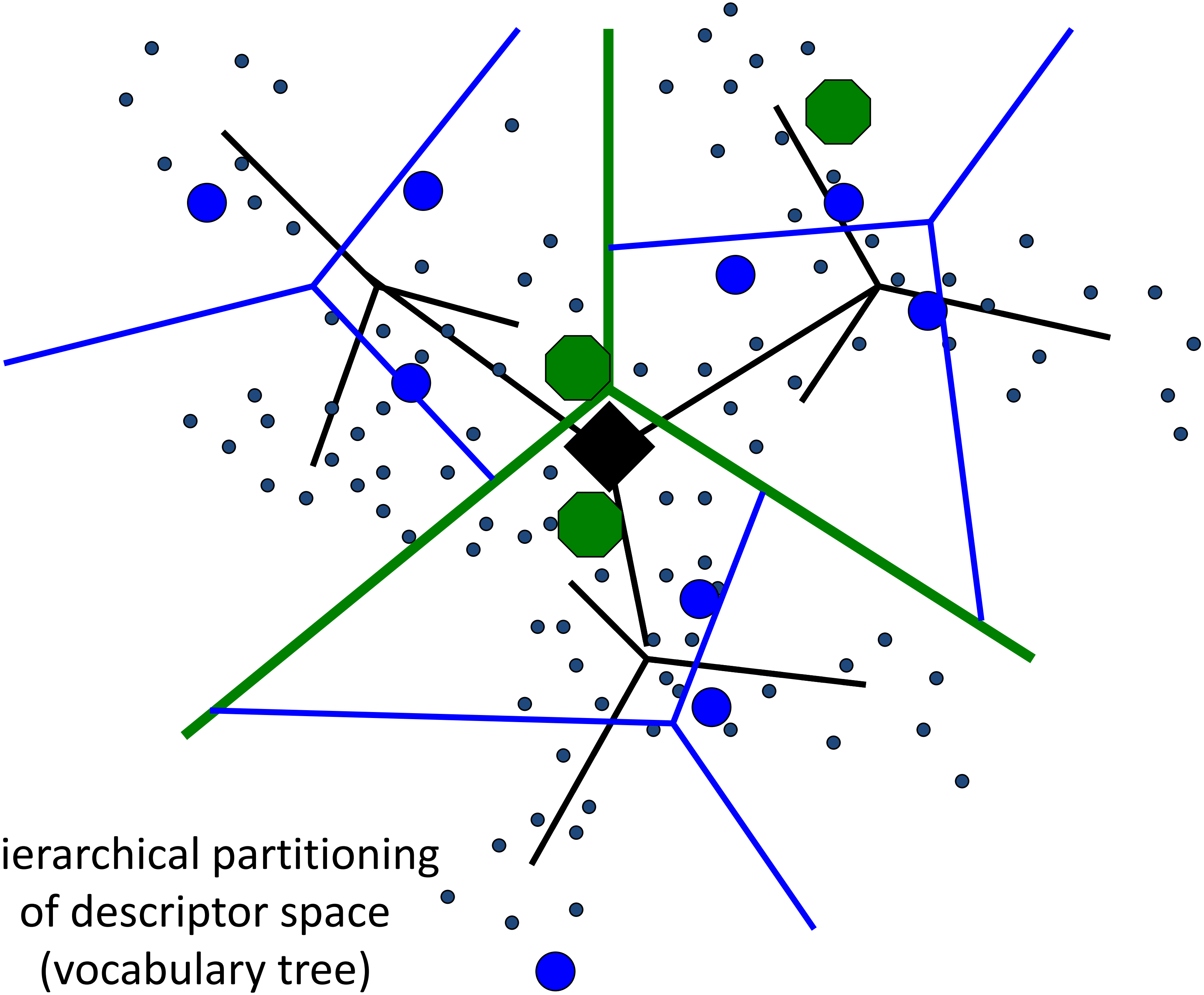


D. Nistér and H. Stewénius, [Scalable Recognition with a Vocabulary Tree](), CVPR 2006



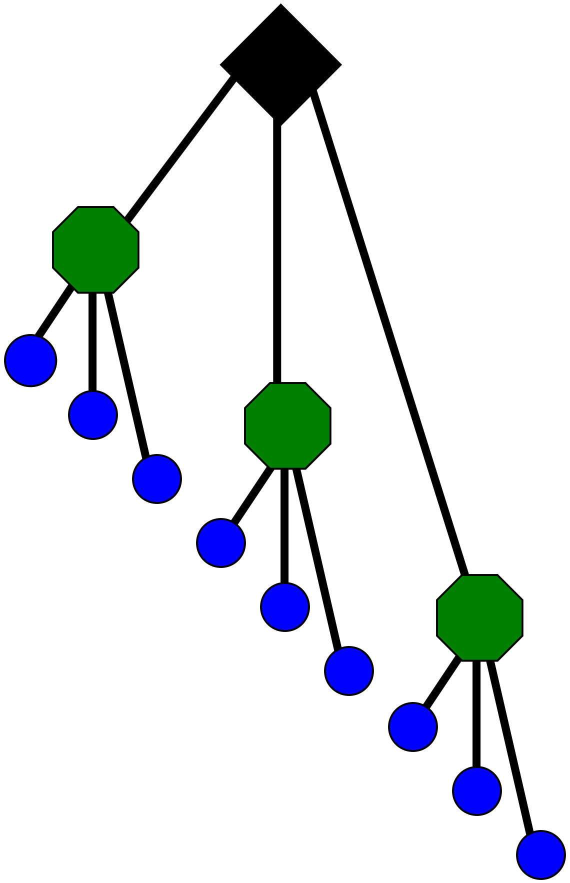
Descriptor space

Slide credit: D. Nister



Hierarchical partitioning
of descriptor space
(vocabulary tree)

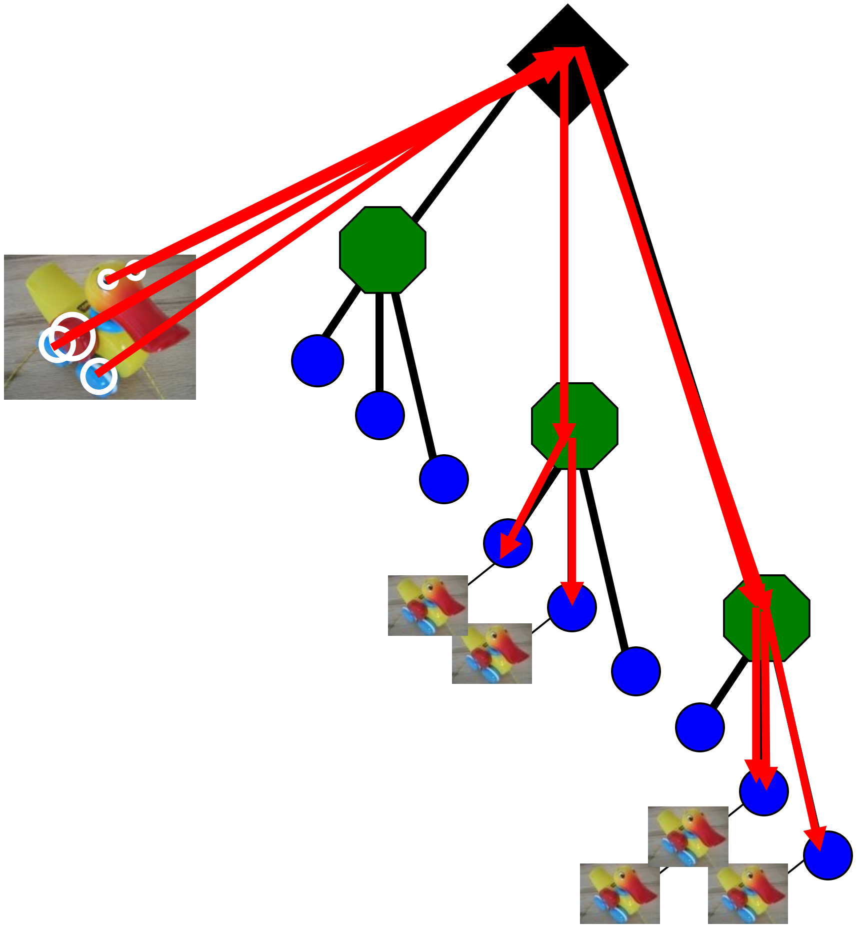
Slide credit: D. Nister



Vocabulary tree/inverted index

Slide credit: D. Nister

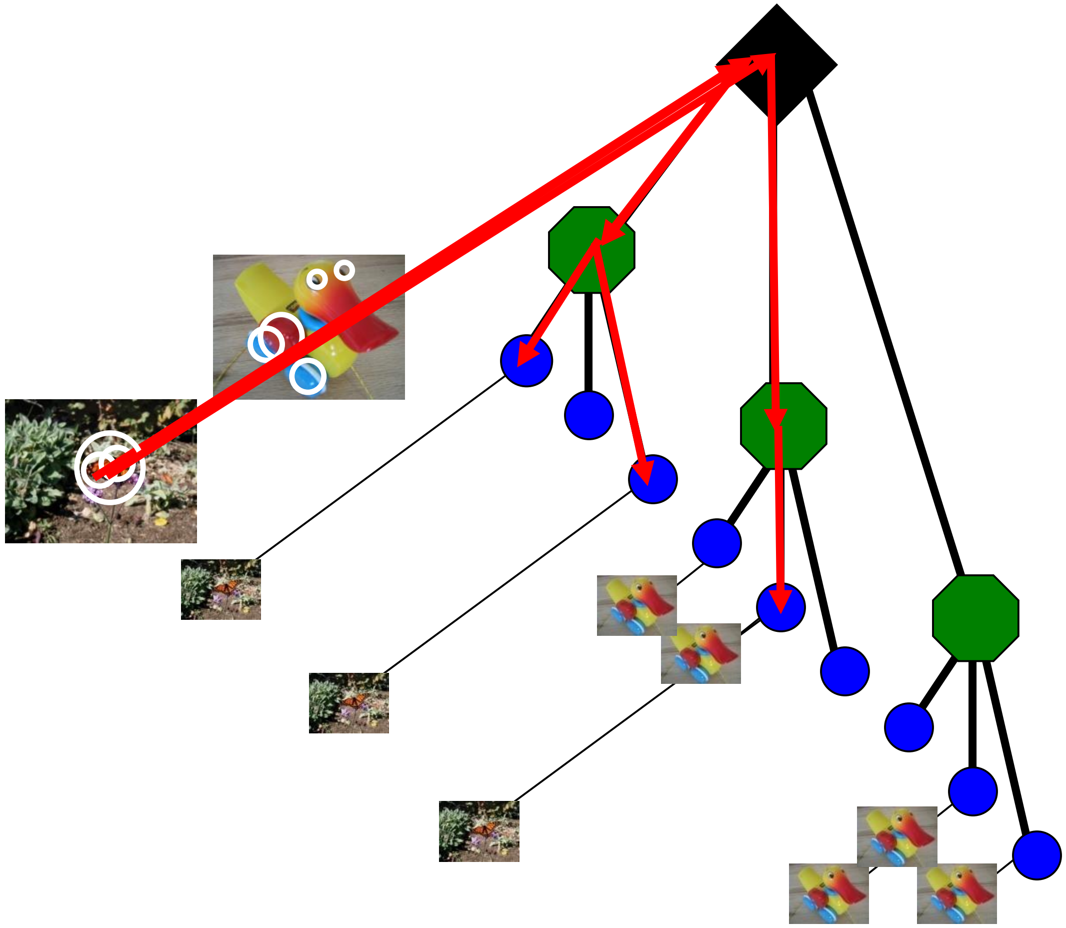
Model images



Populating the vocabulary tree/inverted index

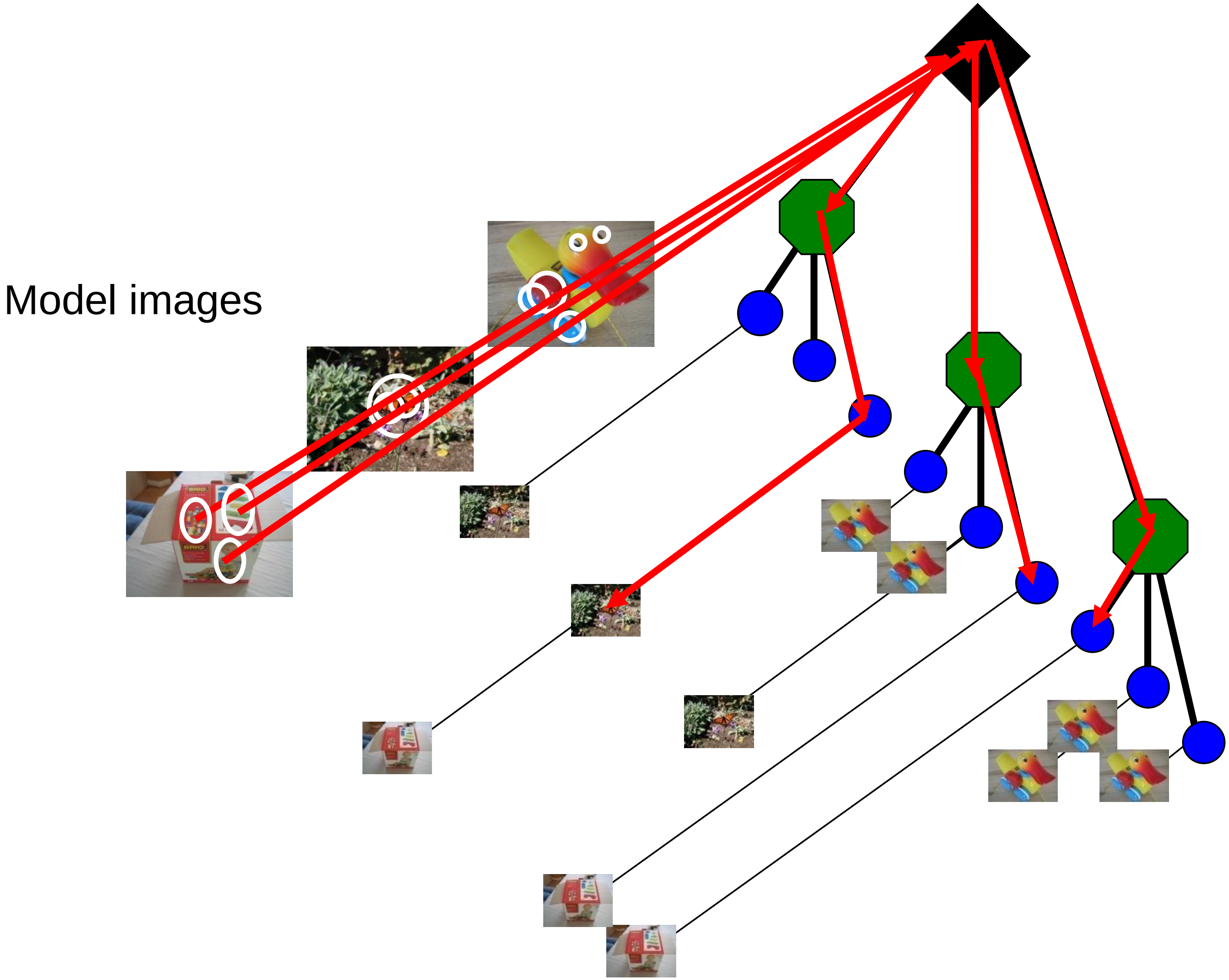
Slide credit: D. Nister

Model images



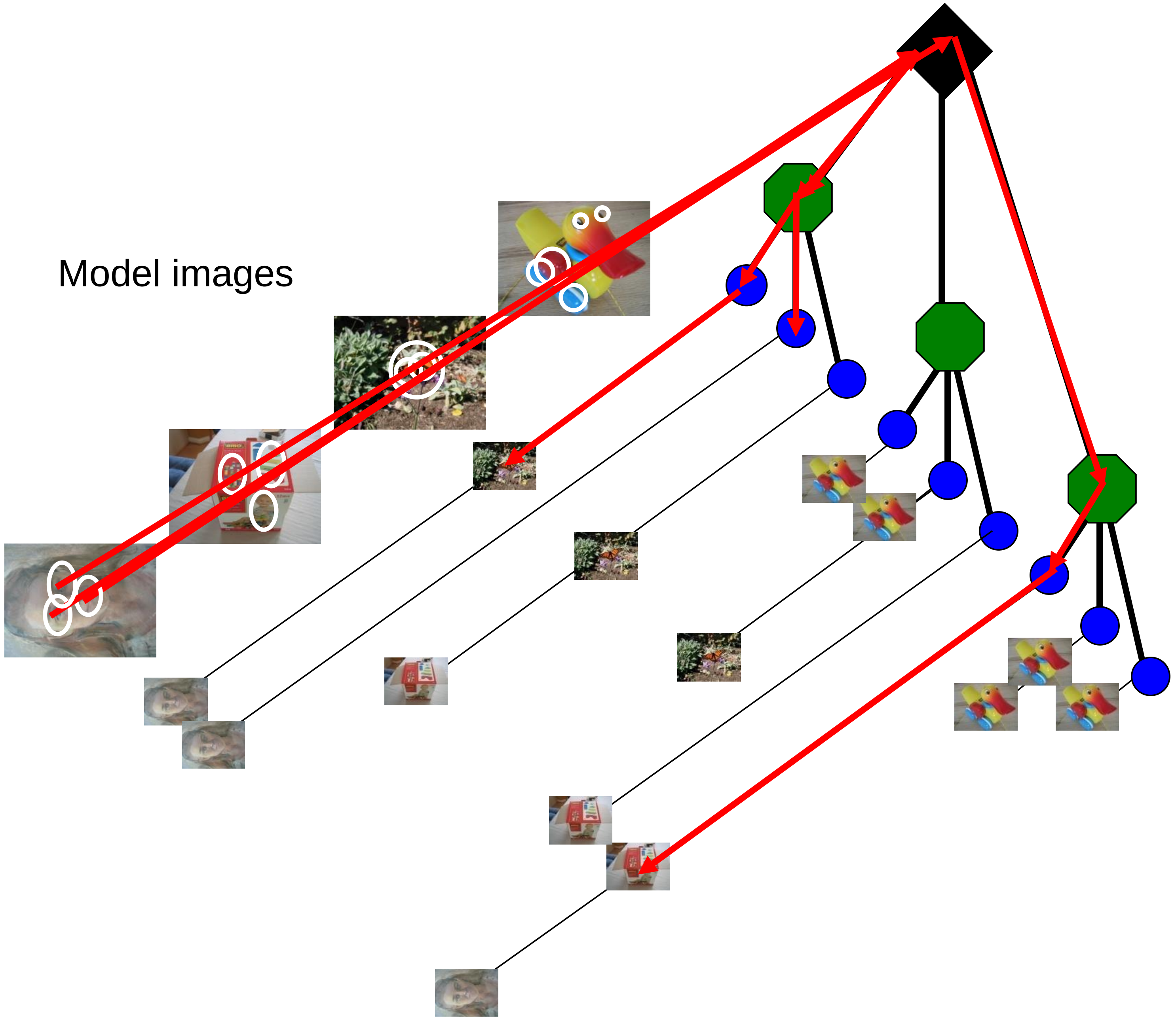
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



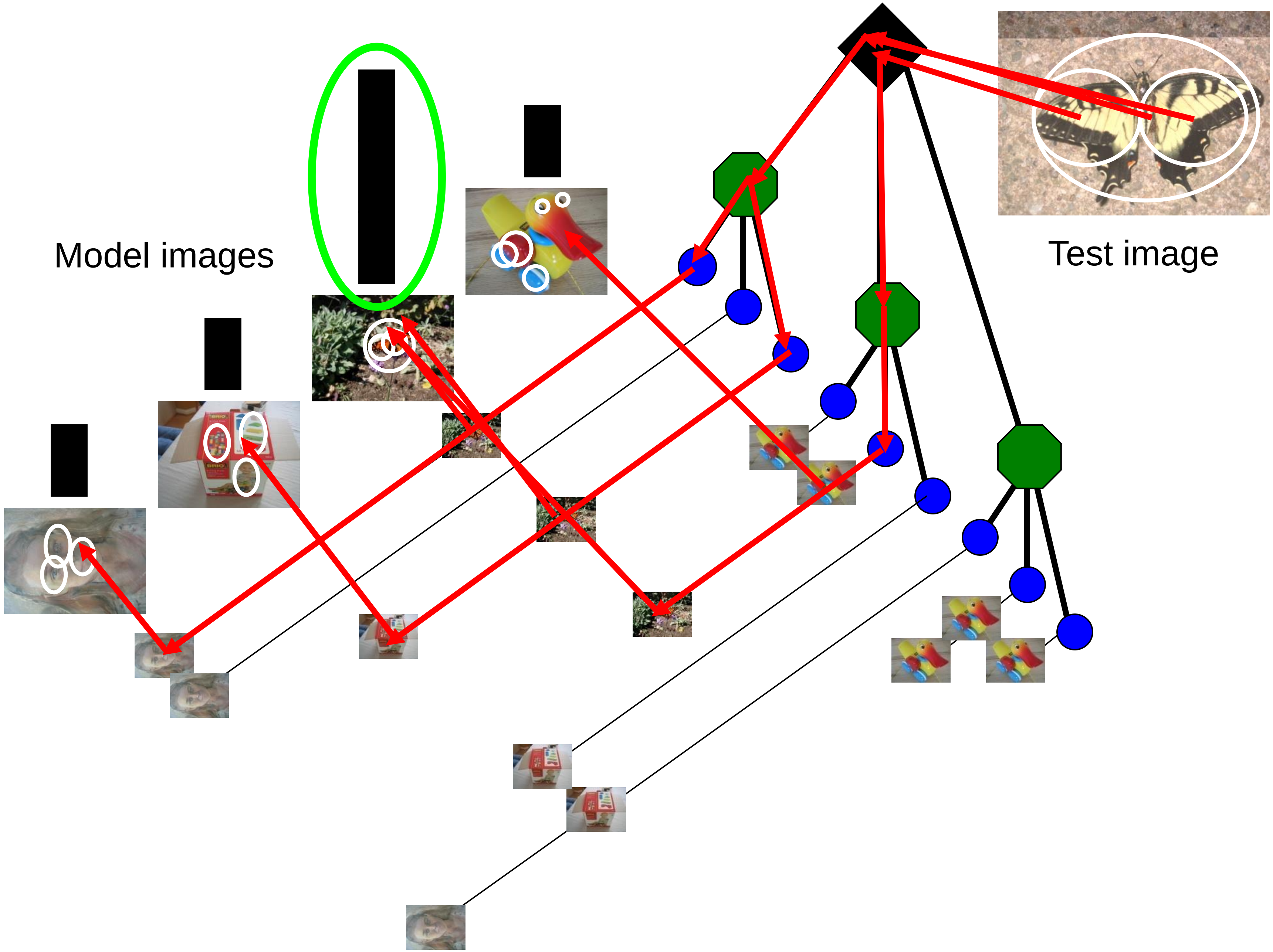
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



Populating the vocabulary tree/inverted index

Slide credit: D. Nister

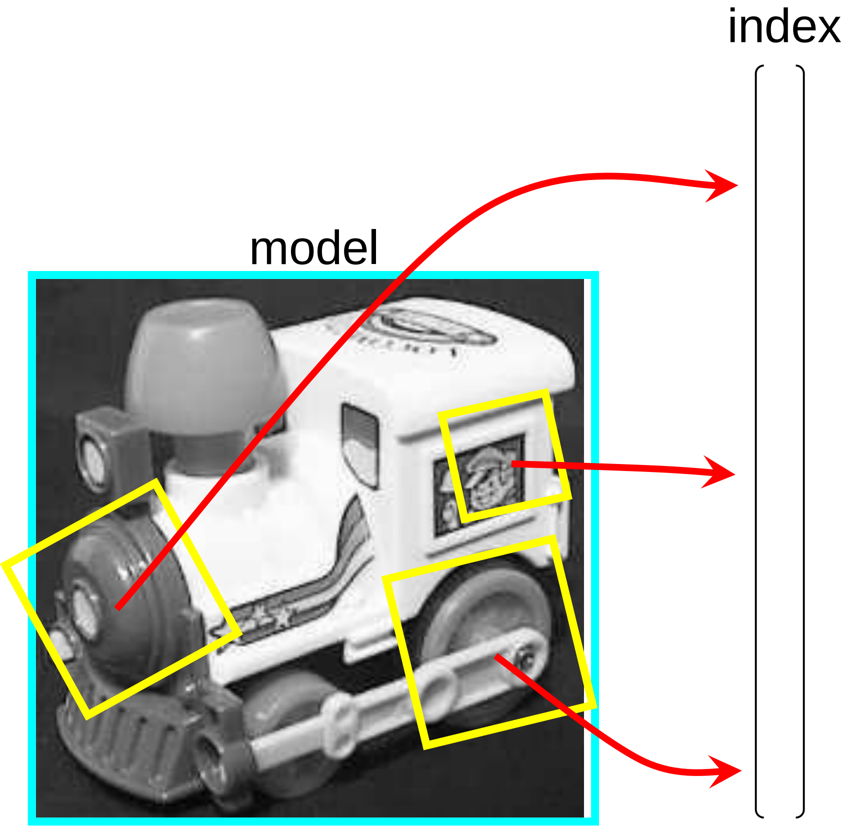


Looking up a test image

Slide credit: D. Nister

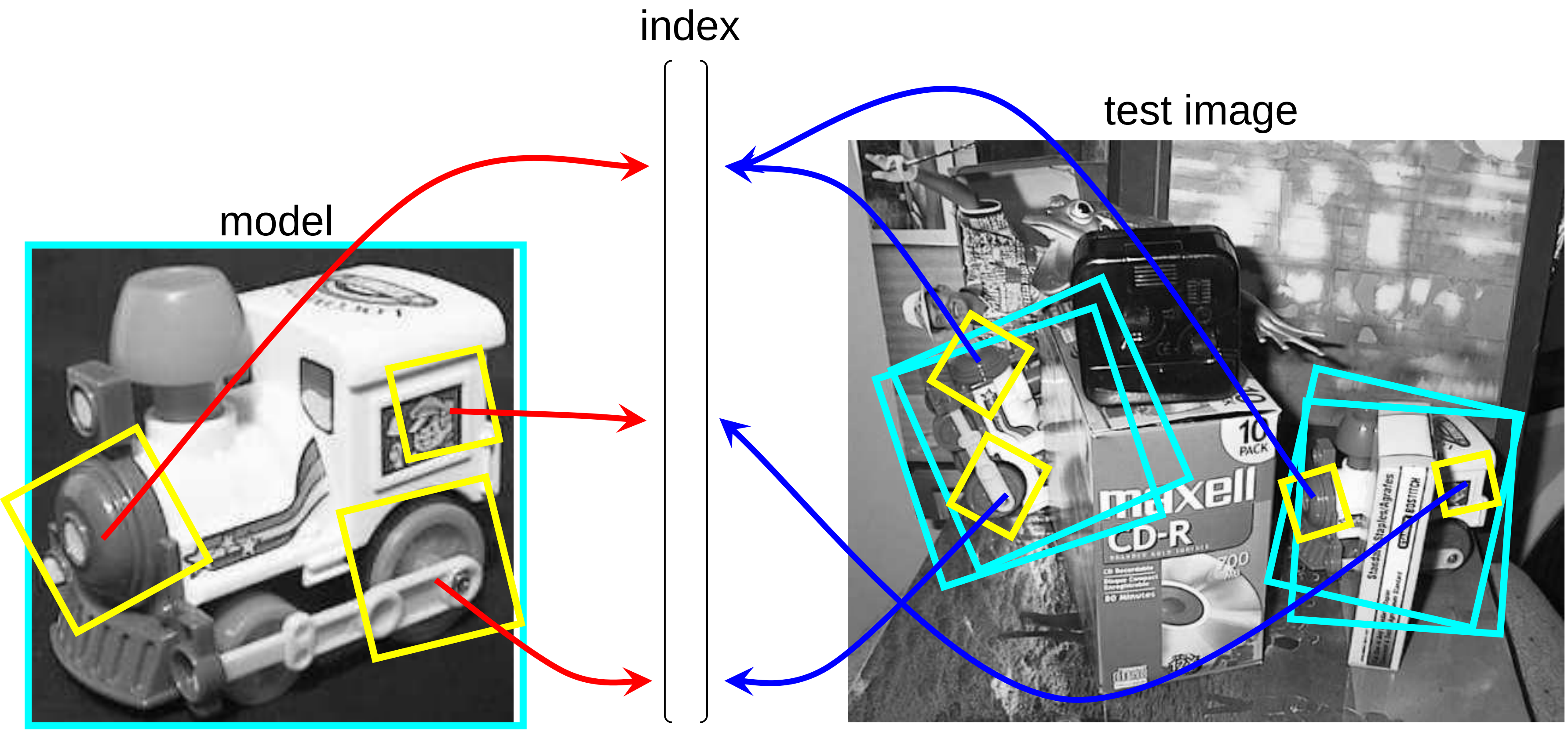
Indexing with geometric invariants

- A match between invariant descriptors can yield a transformation hypothesis



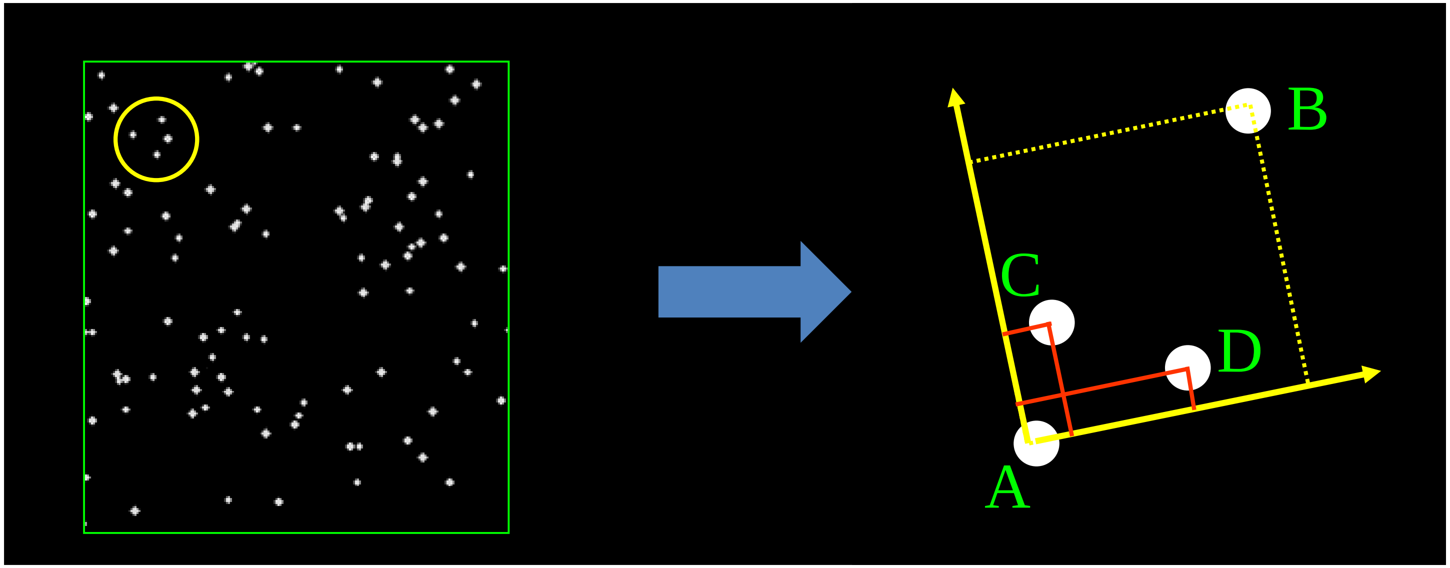
Indexing with geometric invariants

- A match between invariant descriptors can yield a transformation hypothesis



Indexing with geometric invariants

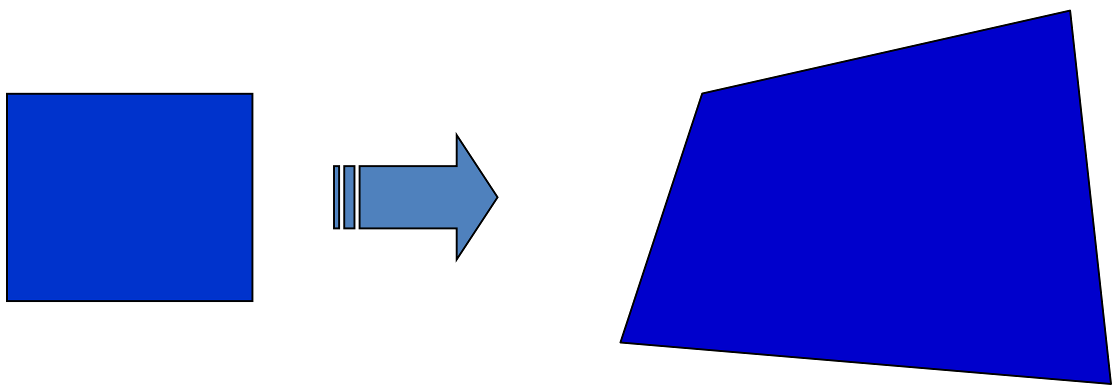
- When we don't have feature descriptors, we can take n-tuples of neighboring features and compute invariant features from their geometric configurations
- Application: searching the sky: http://www.astrometry.net/



Projective (Homography) Transformation

Beyond affine transformations

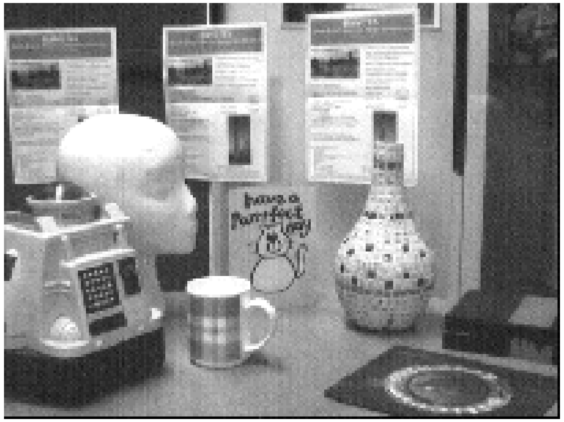
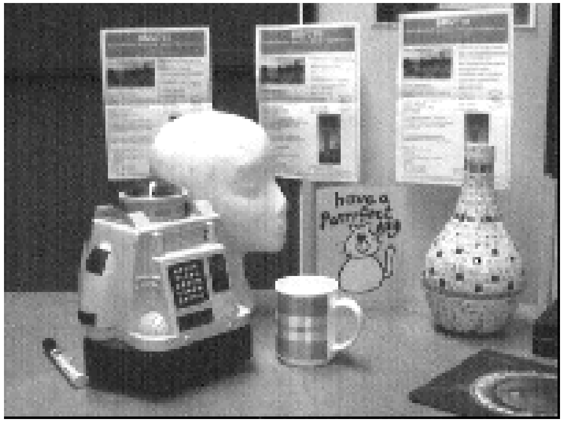
- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)



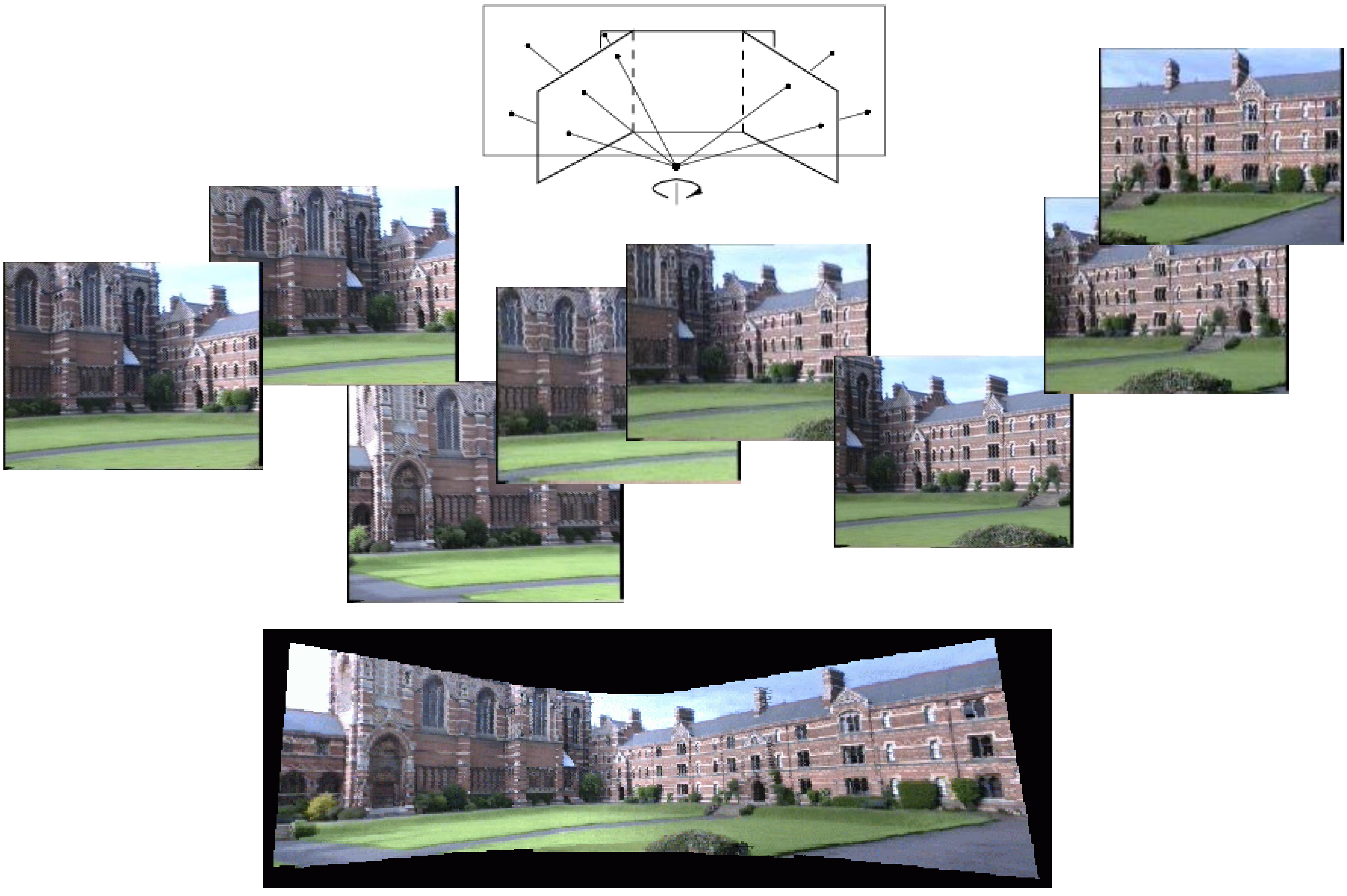
Homography

- The transformation between two views of a planar surface


- The transformation between images from two cameras that share the same center

Application: **Panorama stitching**



Fitting a homography

- Recall: homogenenous coordinates

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

Converting *to* homogenenous image coordinates

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

Converting *from* homogenenous image coordinates

Fitting a homography

- Recall: homogeneous coordinates

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

Converting *to* homogeneous image coordinates

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

Converting *from* homogeneous image coordinates

- Equation for homography:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Fitting a homography

- Equation for homography:

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

$$\lambda \mathbf{x}_i' = \mathbf{H} \mathbf{x}_i$$

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

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

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

3 equations, only 2 linearly independent

Direct linear transform

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

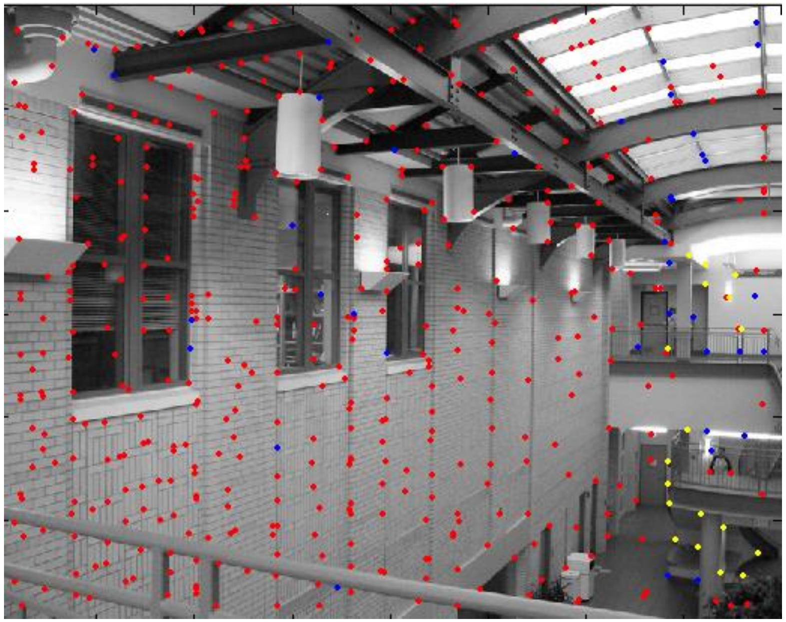
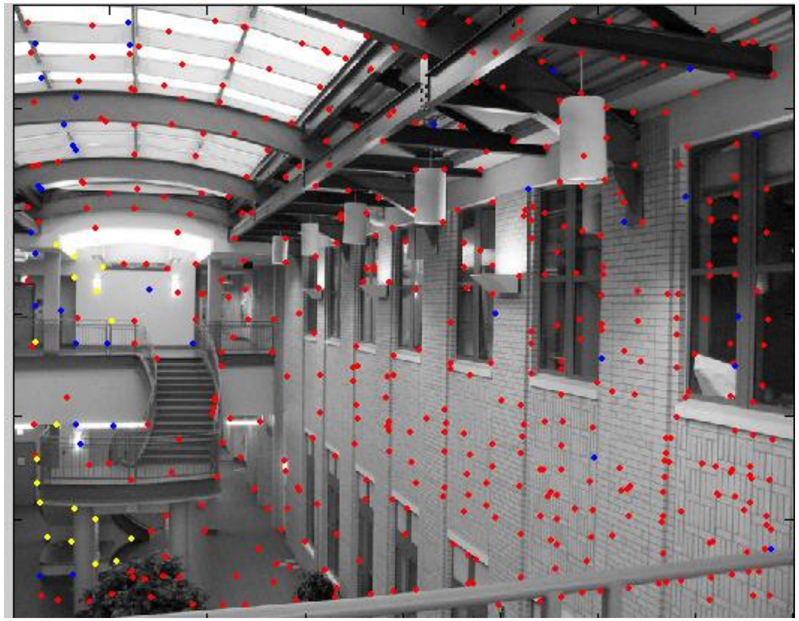
- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Four matches needed for a minimal solution (null space of 8x9 matrix)
- More than four: homogeneous least squares

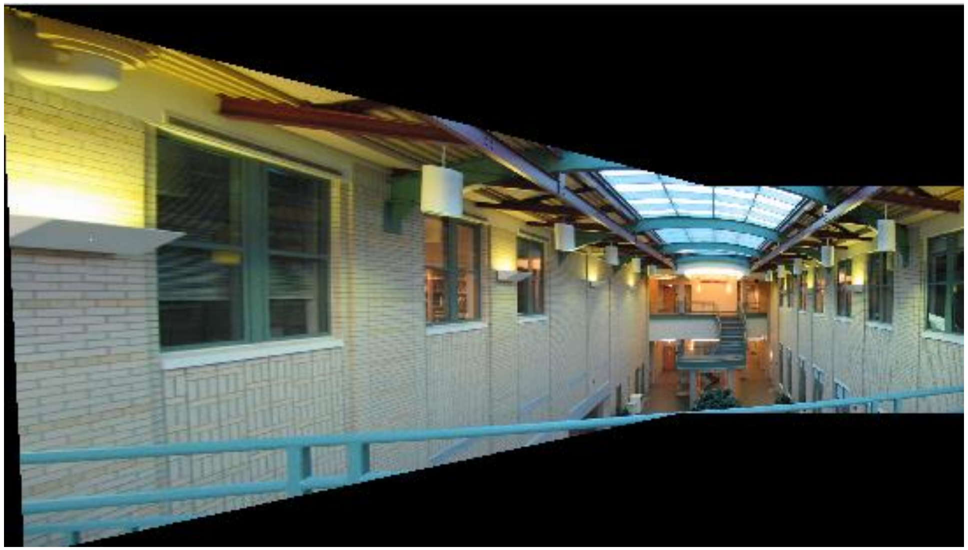
RANSAC for Estimating Homography

RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute *inliers* where $SSD(p_i', \boldsymbol{H}\, p_{i)} < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares H estimate on all of the inliers

RANSAC



Why "Recognising Panoramas"?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

Computational Photography, Alexei Efros, CMU, Fall 2005

Why “Recognising Panoramas”?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

Why "Recognising Panoramas"?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

- 2D Rotations (θ, ϕ)
  - Ordering $\nRightarrow$ matching images

Why "Recognising Panoramas"?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

- 2D Rotations (θ, ϕ)
  - Ordering $\nRightarrow$ matching images

Why "Recognising Panoramas"?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

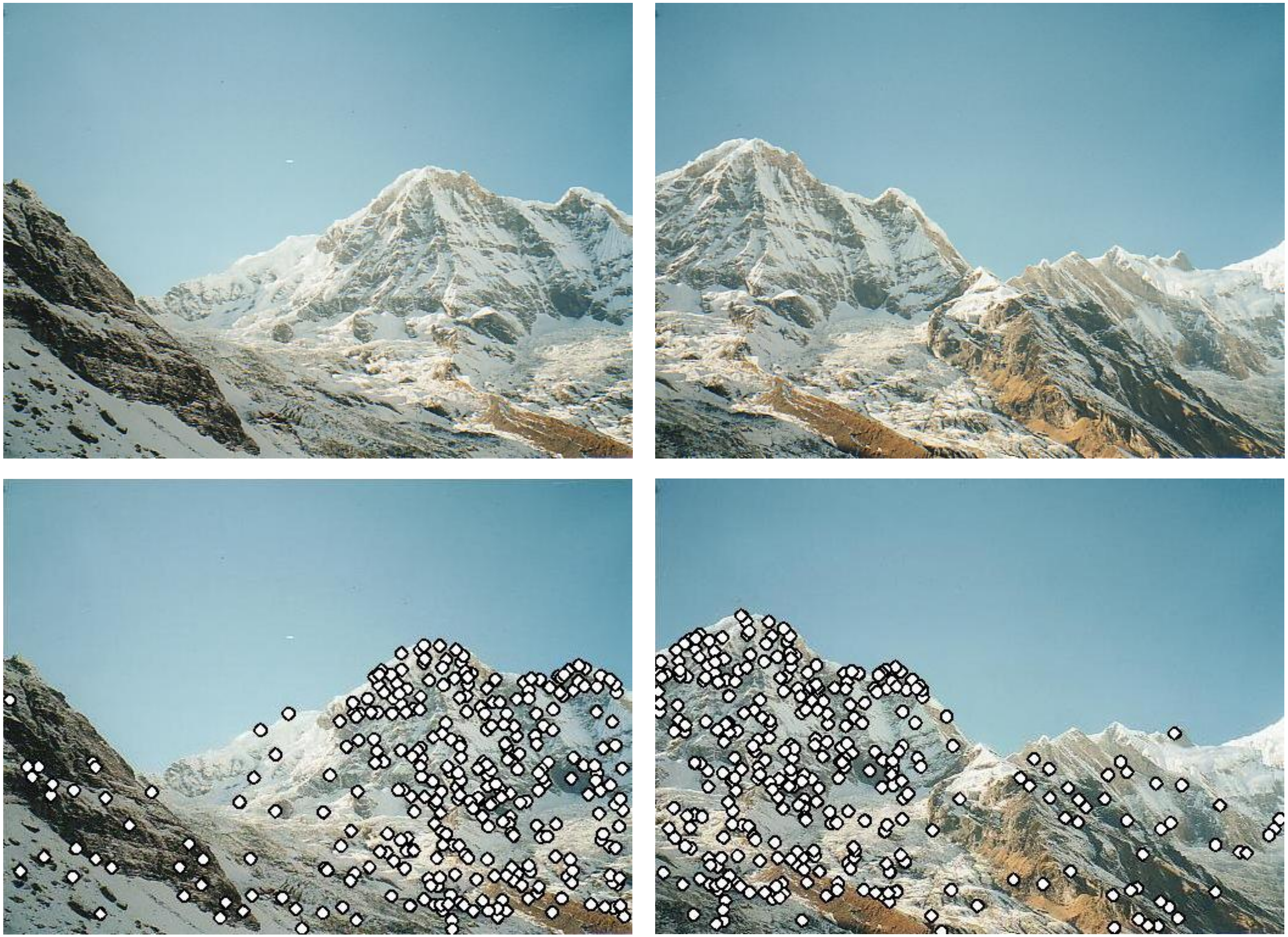
- 2D Rotations (θ, ϕ)
  - Ordering $\nRightarrow$ matching images

Why “Recognising Panoramas”?

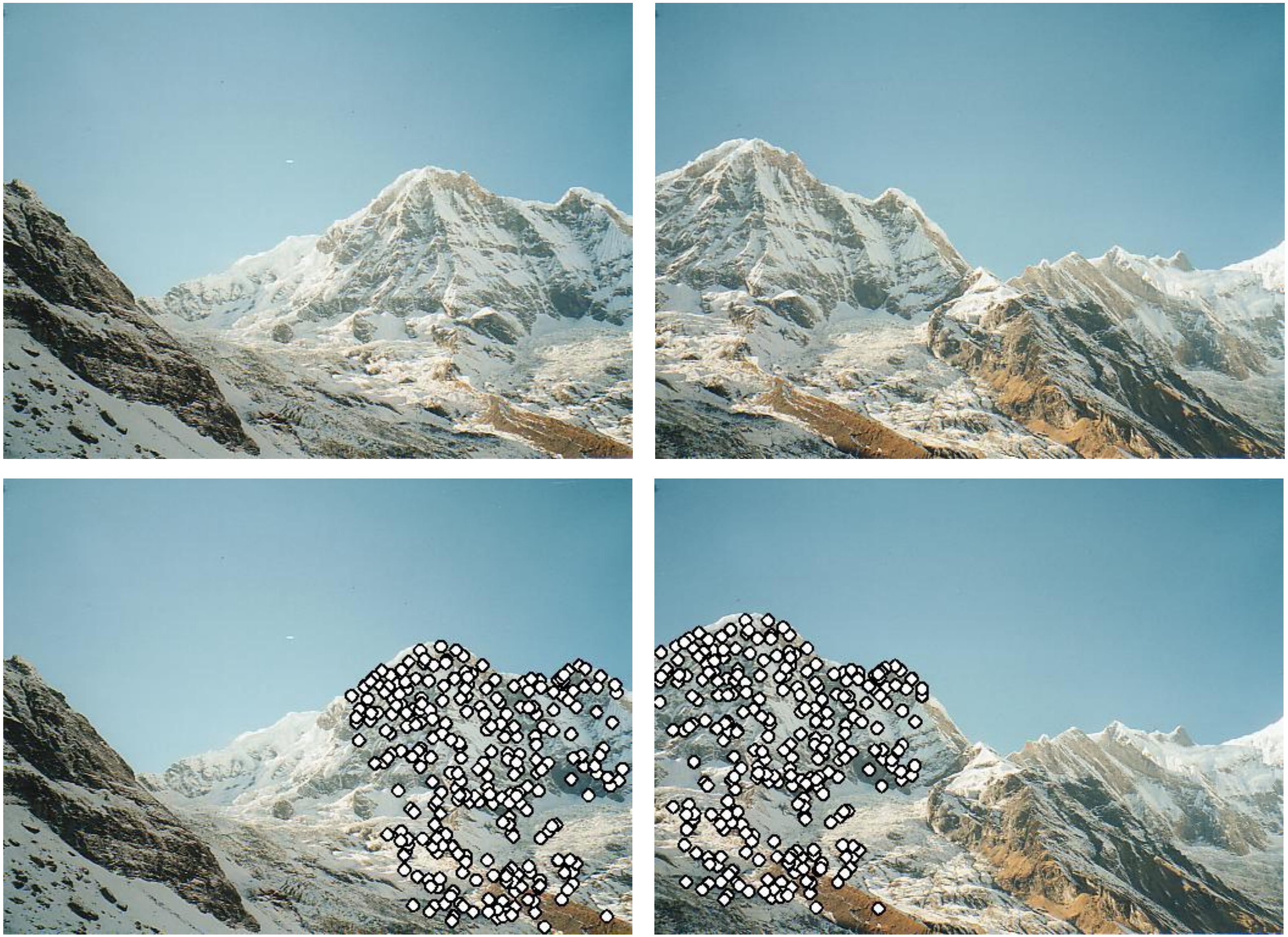
Overview of Image Alignment

- Feature Matching
- Image Matching
- Bundle Adjustment
- Multi-band Blending

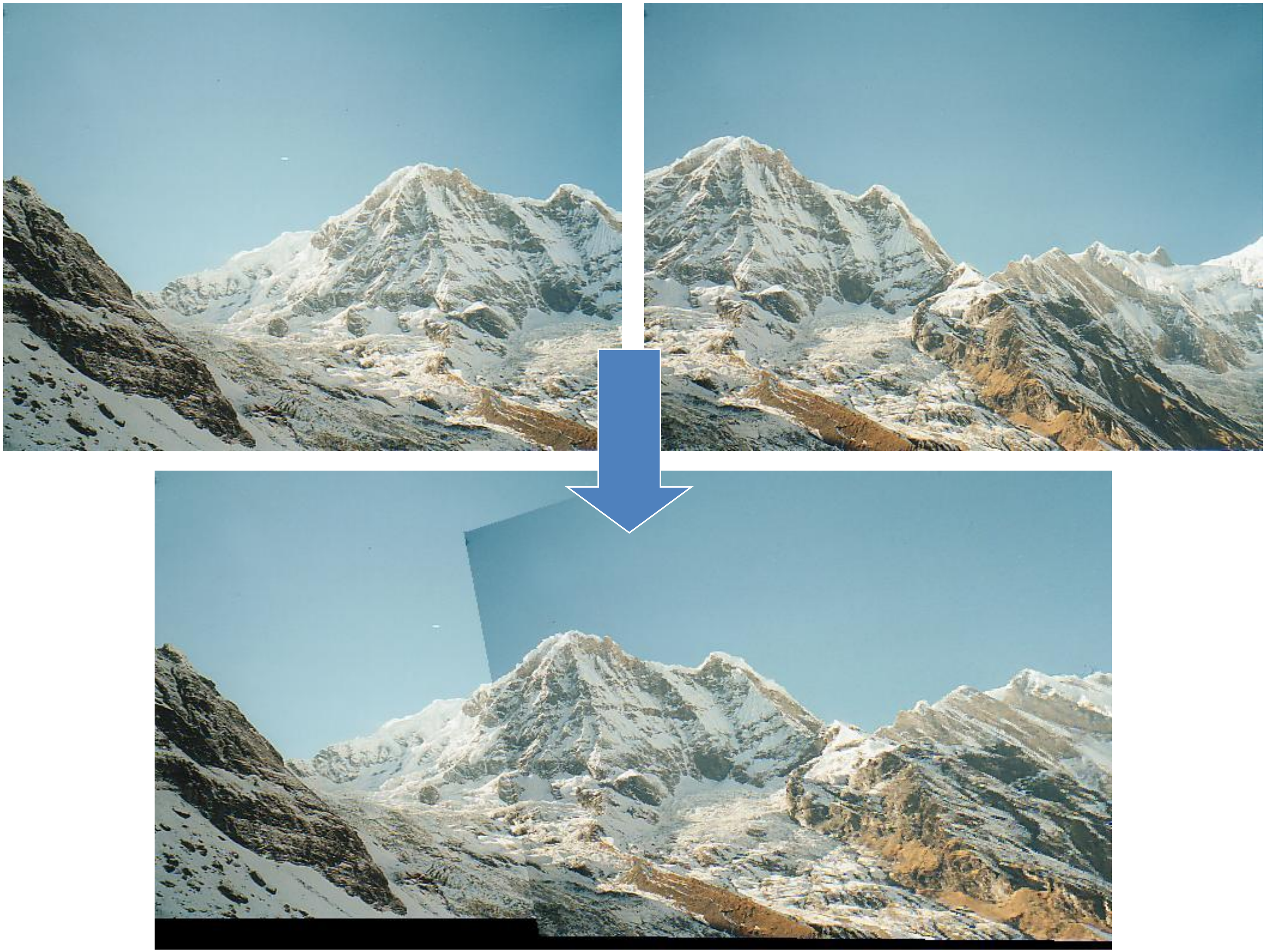
RANSAC for Homography



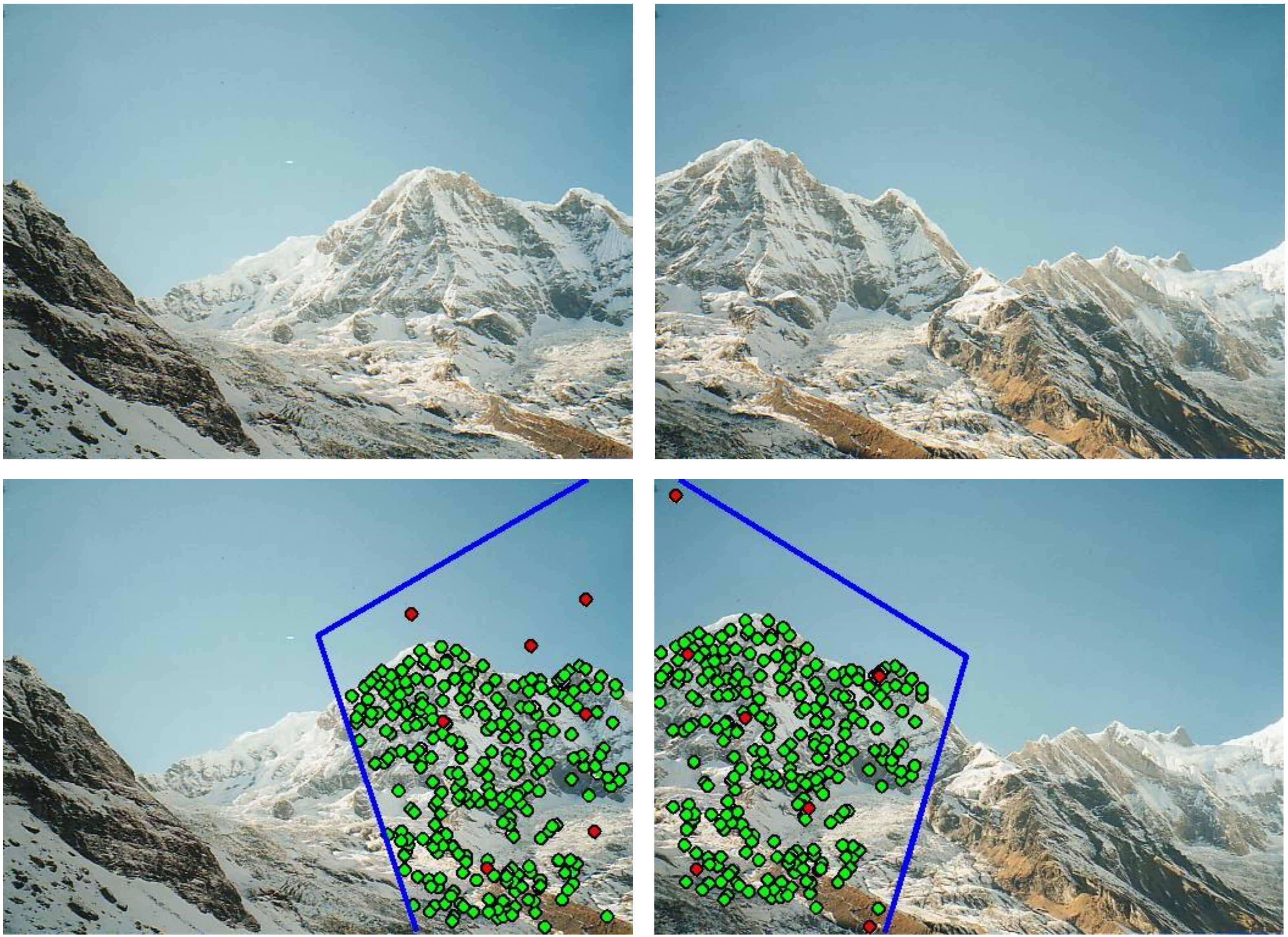
RANSAC for Homography



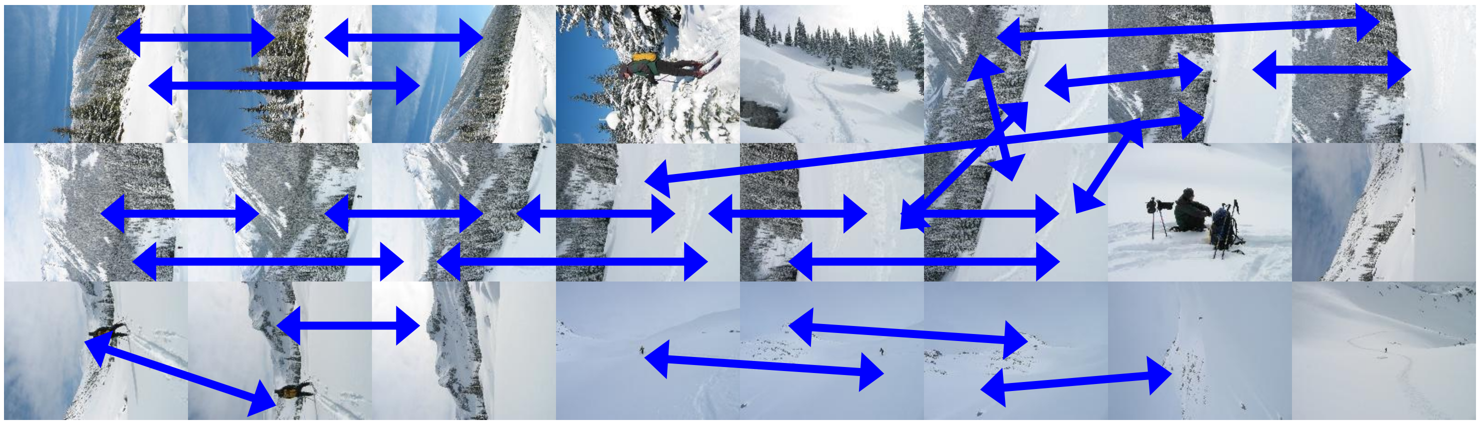
RANSAC for Homography



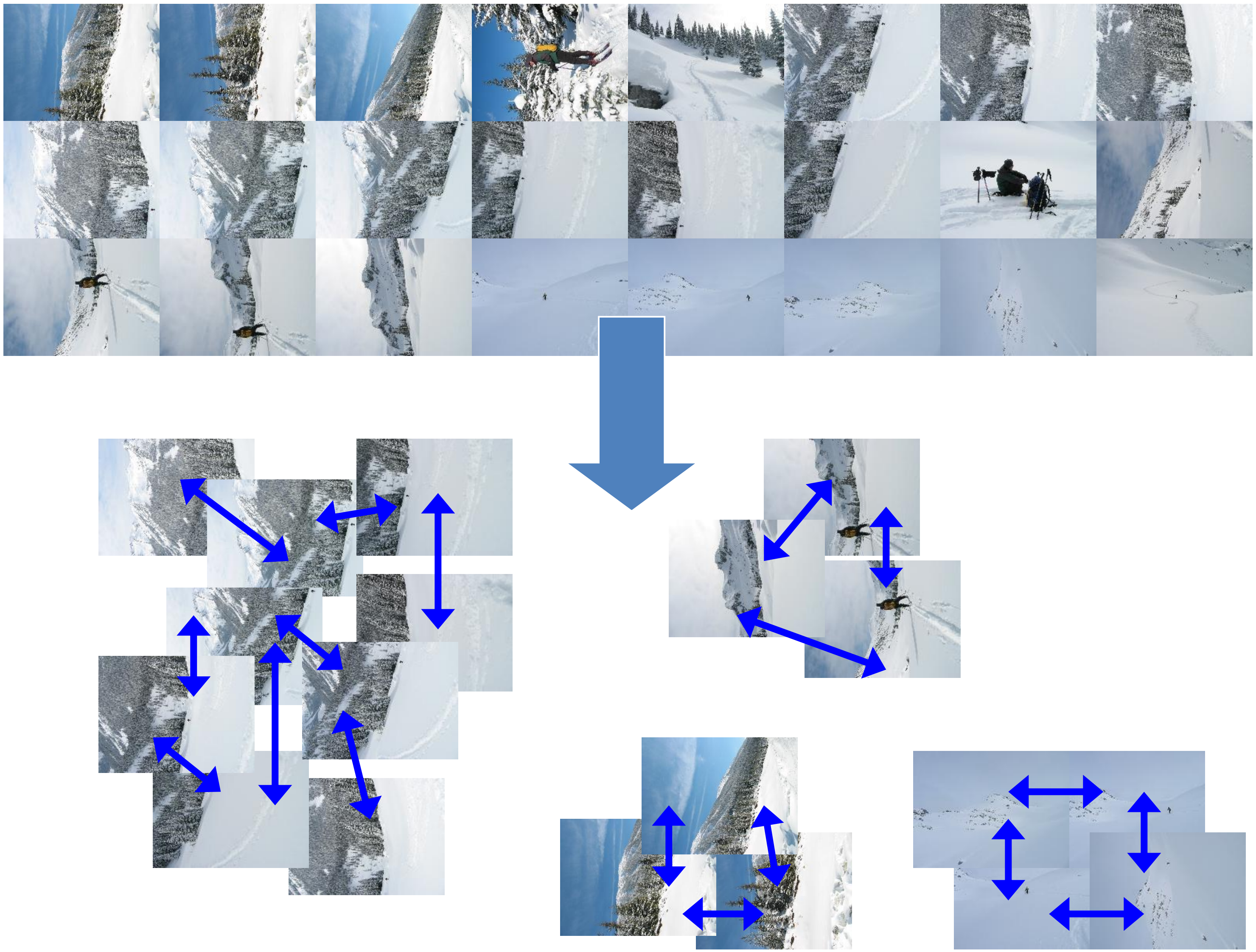
Probabilistic model for verification



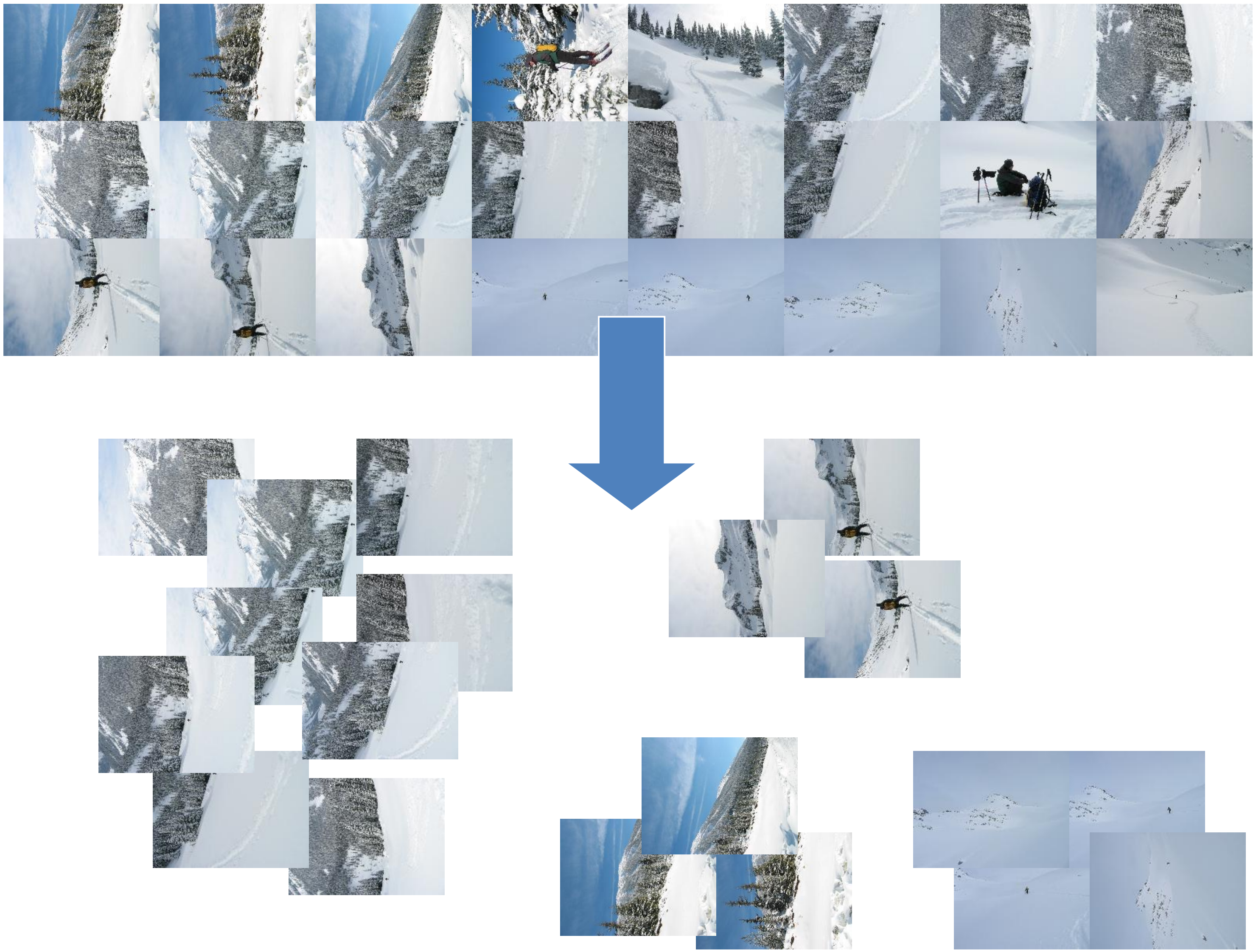
Finding the panoramas



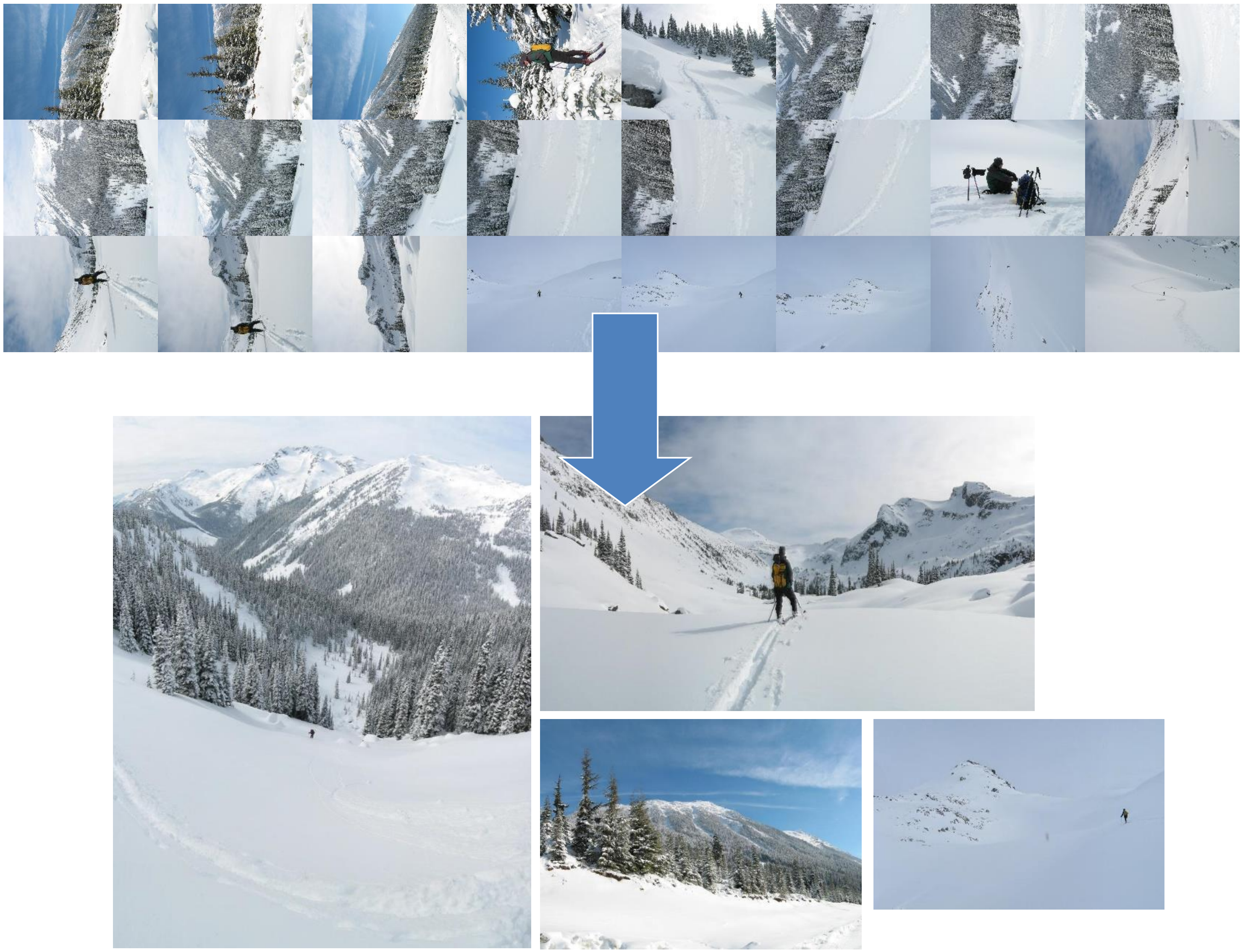
Finding the panoramas



Finding the panoramas



Finding the panoramas



Homography for Rotation

- Parameterise each camera by rotation and focal length

$$\mathbf{R}_i = e^{[\boldsymbol{\theta}_i]_\times}, \;\; [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- This gives pairwise homographies

$$\tilde{\mathbf{u}}_i = \mathbf{H}_{ij}\tilde{\mathbf{u}}_j, \;\; \mathbf{H}_{ij} = \mathbf{K}_i\mathbf{R}_i\mathbf{R}_j^T\mathbf{K}_j^{-1}$$

Bundle Adjustment

- New images initialised with rotation, focal length of best matching image

Bundle Adjustment

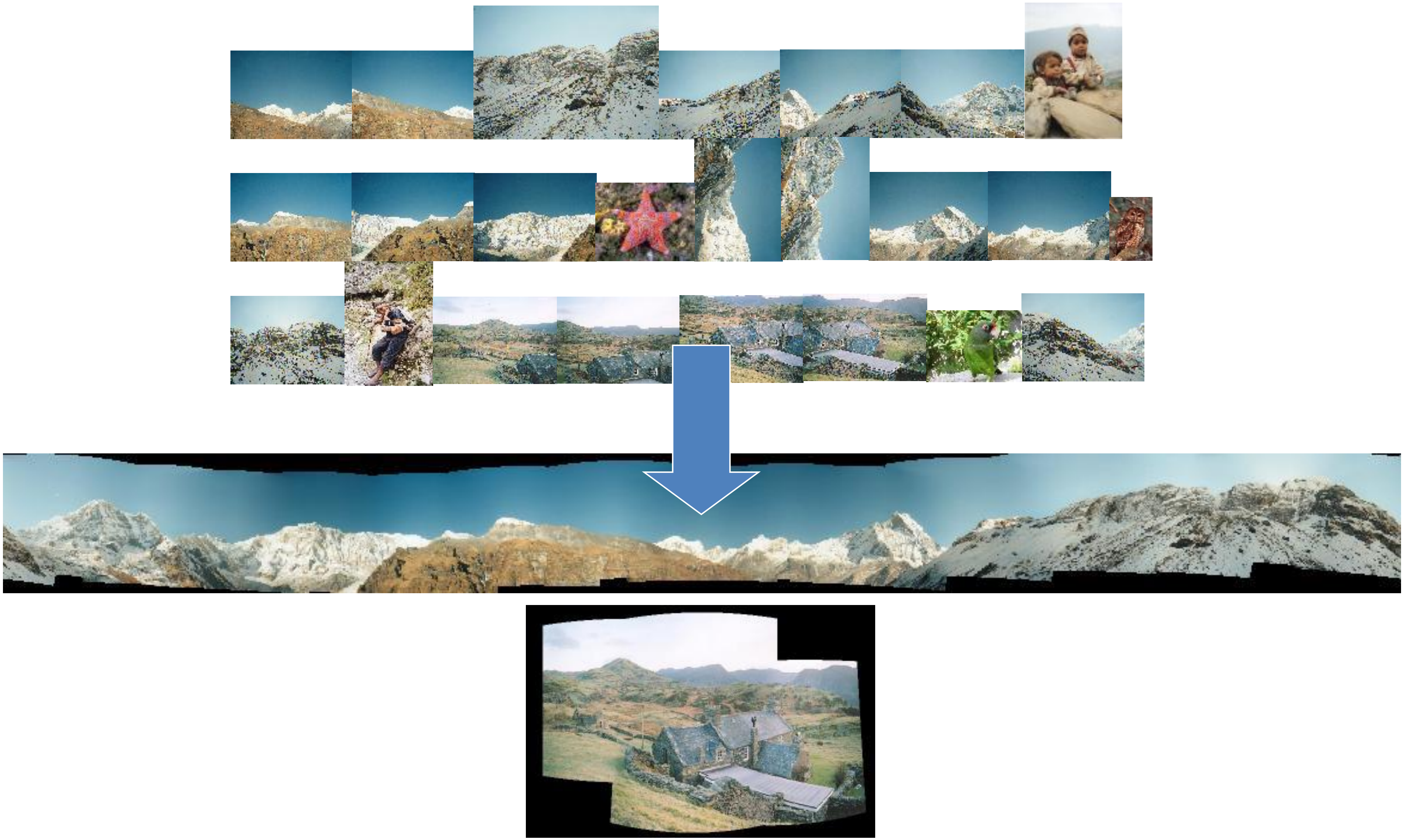
- New images initialised with rotation, focal length of best matching image

Computational Photography, Alexei Efros, CMU, Fall 2005

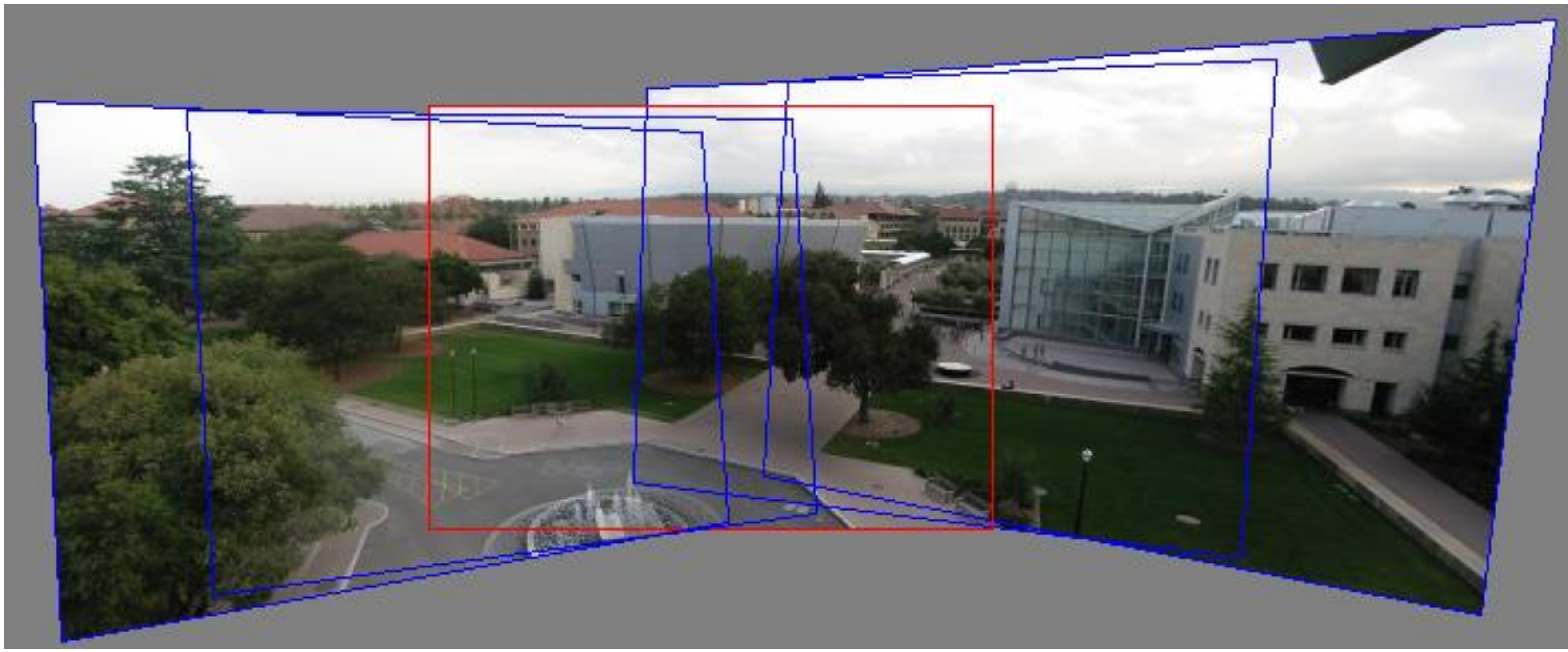
Multi-band Blending

- Burt & Adelson 1983
  - Blend frequency bands over range $\propto \lambda$

Results



Can we use homographies to create a 360 panorama?



- In order to figure this out, we need to learn what a **camera** is

360 panorama