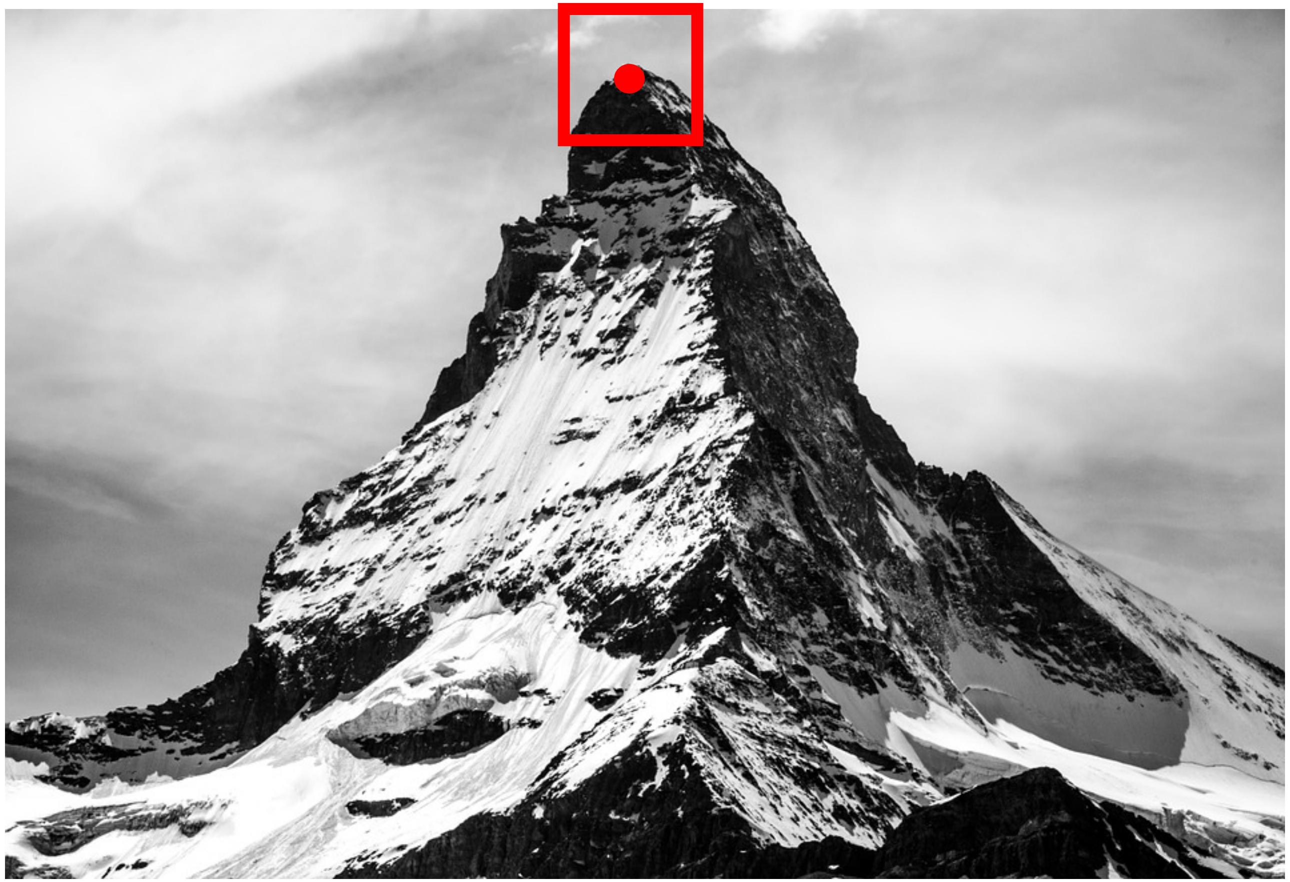
Feature descriptors and matching

Feature detection and description

- Harris corner detection gives:
  - Location of each detected corner
  - Orientation of the corner (given by $\mathbf{x}_{max}$)
  - Scale of the corner (the image scale which gives the maximum response at this location)
- Want feature descriptor that is
  - Invariant to photometric transformations, translation, rotation, scaling
  - Discriminative
- P

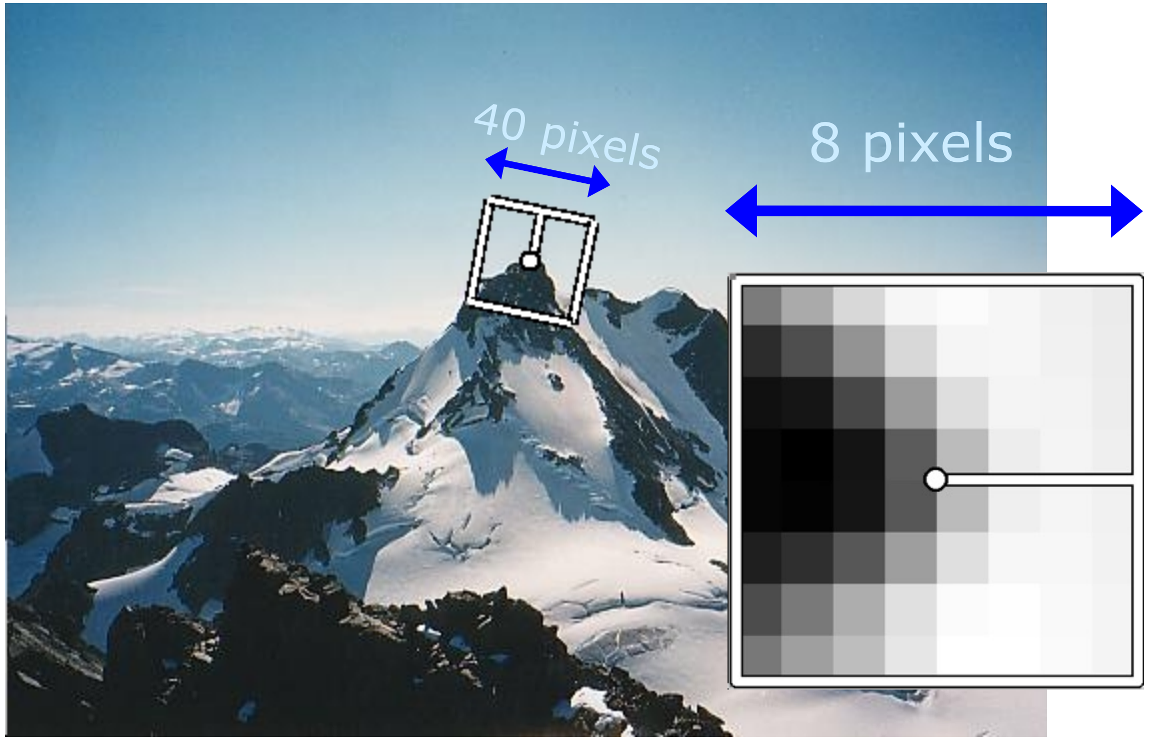
Multiscale Oriented PatcheS descriptor

- Describe a corner by the patch around that pixel
- Scale invariance by using scale identified by corner detector
- Rotation invariance by using orientation identified by corner detector
- Photometric invariance by subtracting mean and dividing by standard deviation



Multiscale Oriented PatcheS descriptor

- Take 40x40 square window around detected feature at the right scale
- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Adapted from slide by Matthew Brown

MOPS

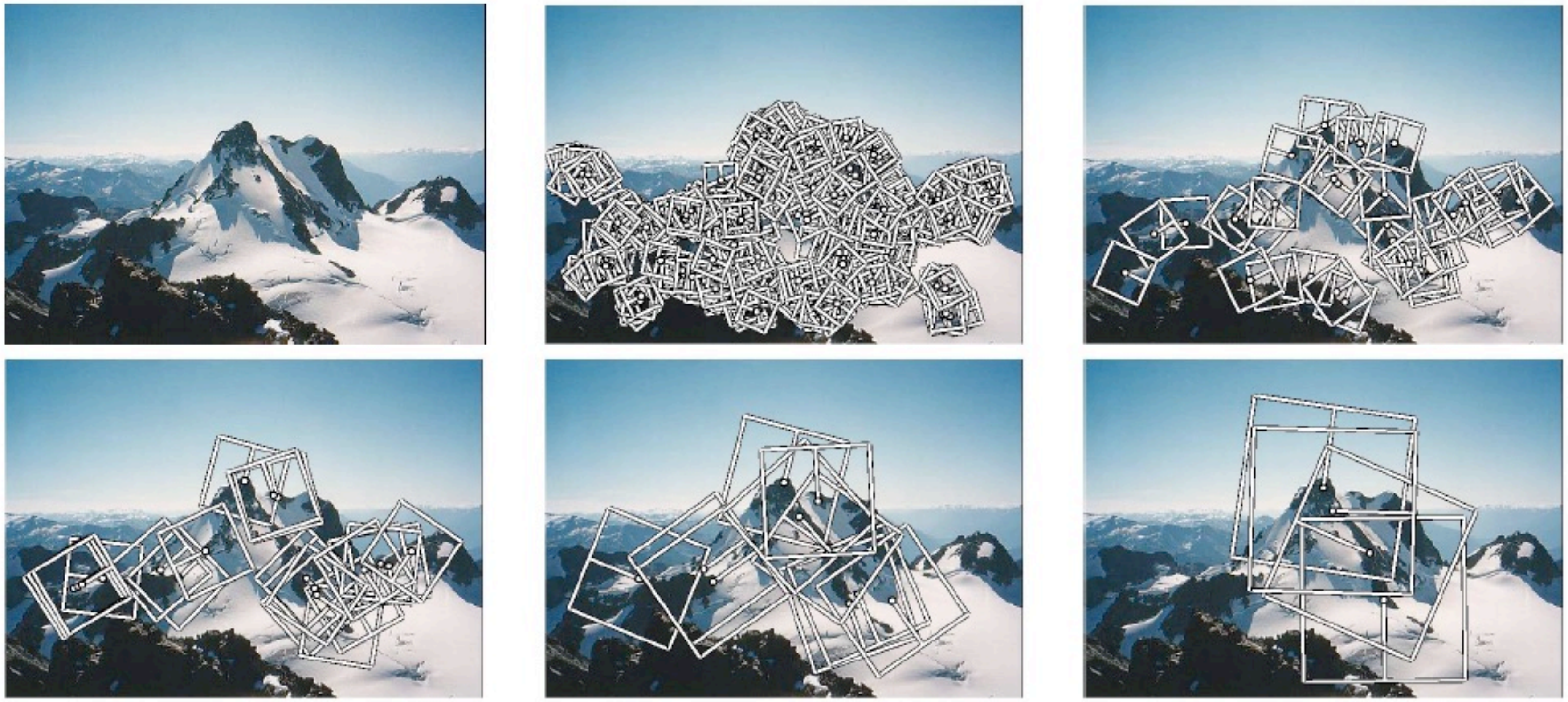


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Detour: Image transformations

- What does it mean to rotate a patch?
- Each pixel has coordinates (x,y)
- Rotation represented by a matrix R
- Pixel's new coordinates:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R \begin{bmatrix} x \\ y \end{bmatrix}$$

- I'(x',y') = I(x,y)

Detour: Image transformations

- What if destination pixel is fractional?
- Flip computation: for every destination pixel figure out source pixel
  - Use interpolation if source location is fractional

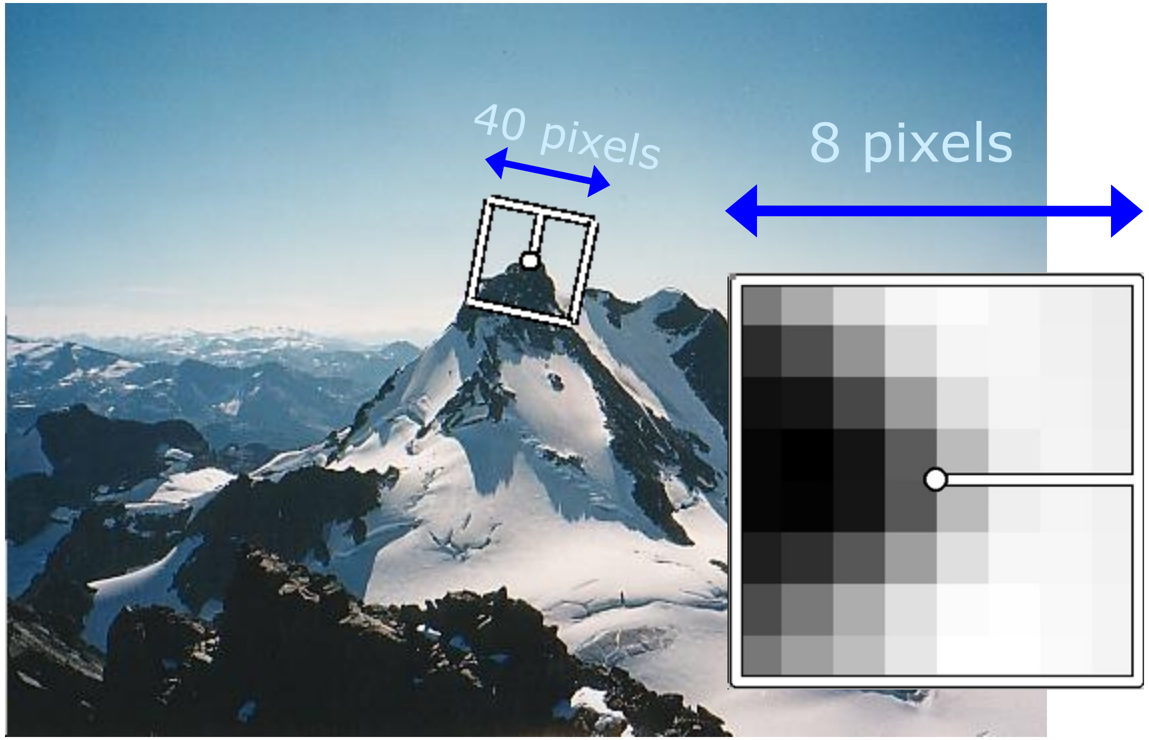
- I'(x', y') = I(x,y)

$$\begin{bmatrix} x \\ y \end{bmatrix} = R^{-1} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



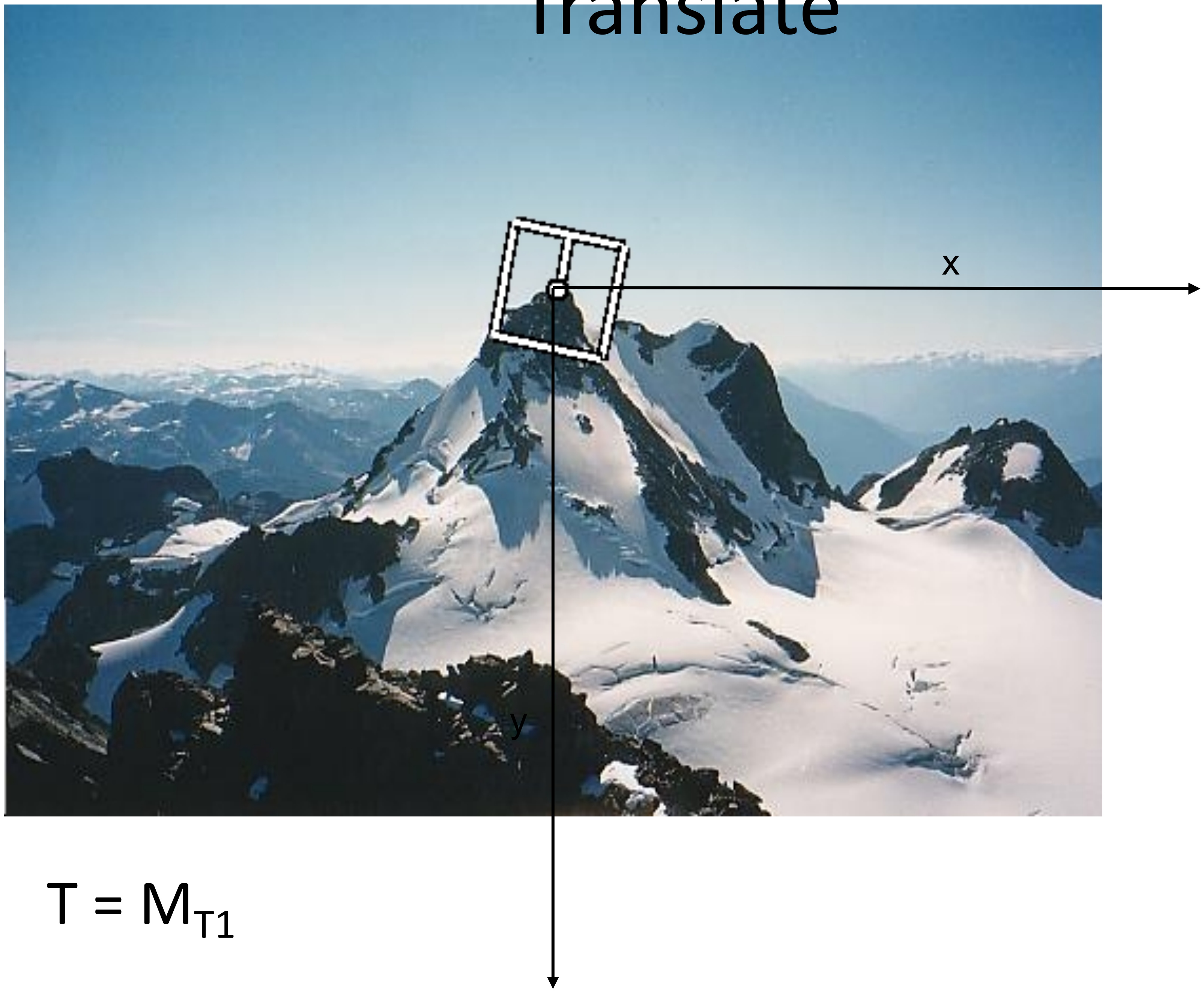
Adapted from slide by Matthew Brown

MOPS descriptor

- You can combine transformations together to get the final transformation

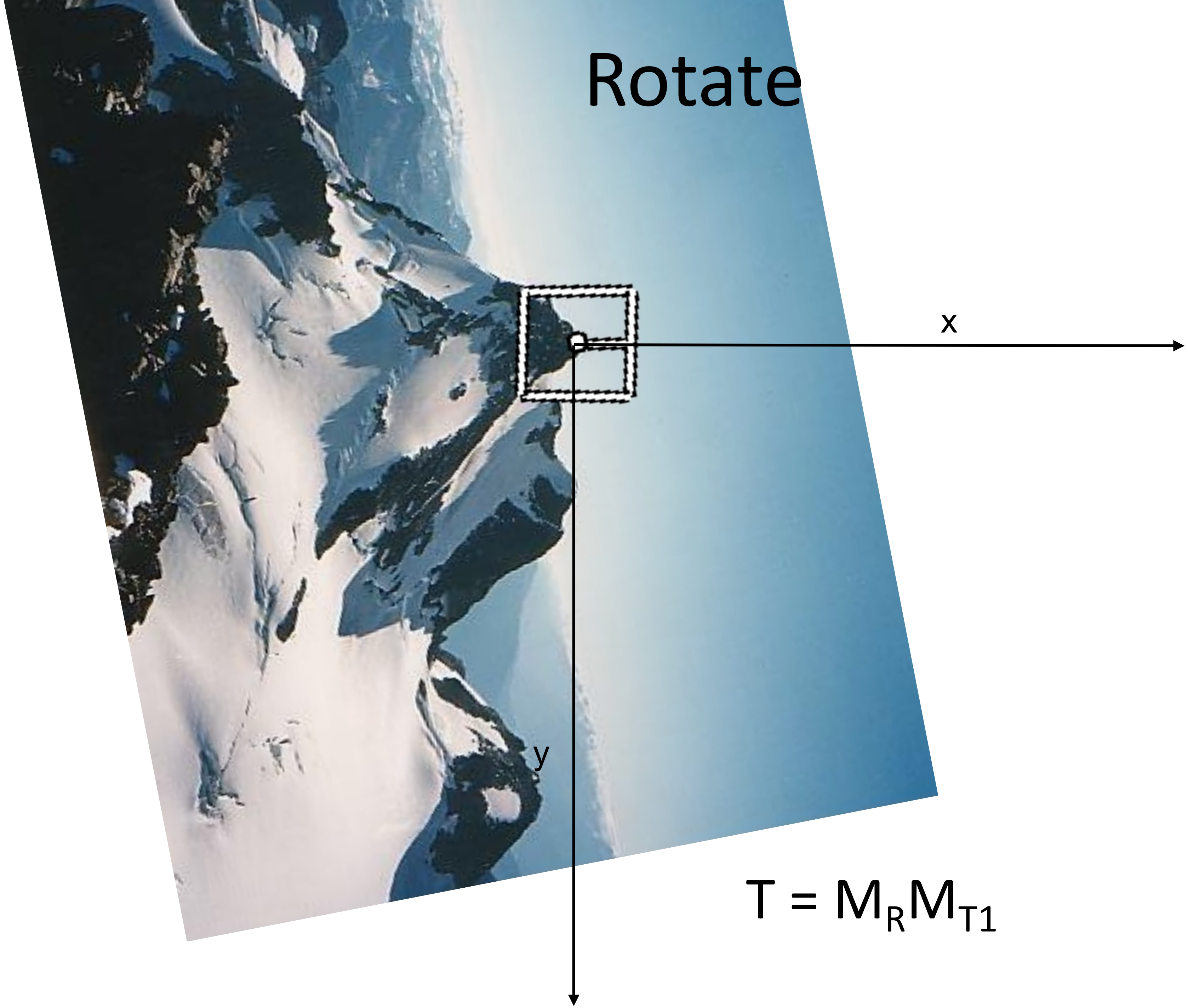
T = ?

Translate



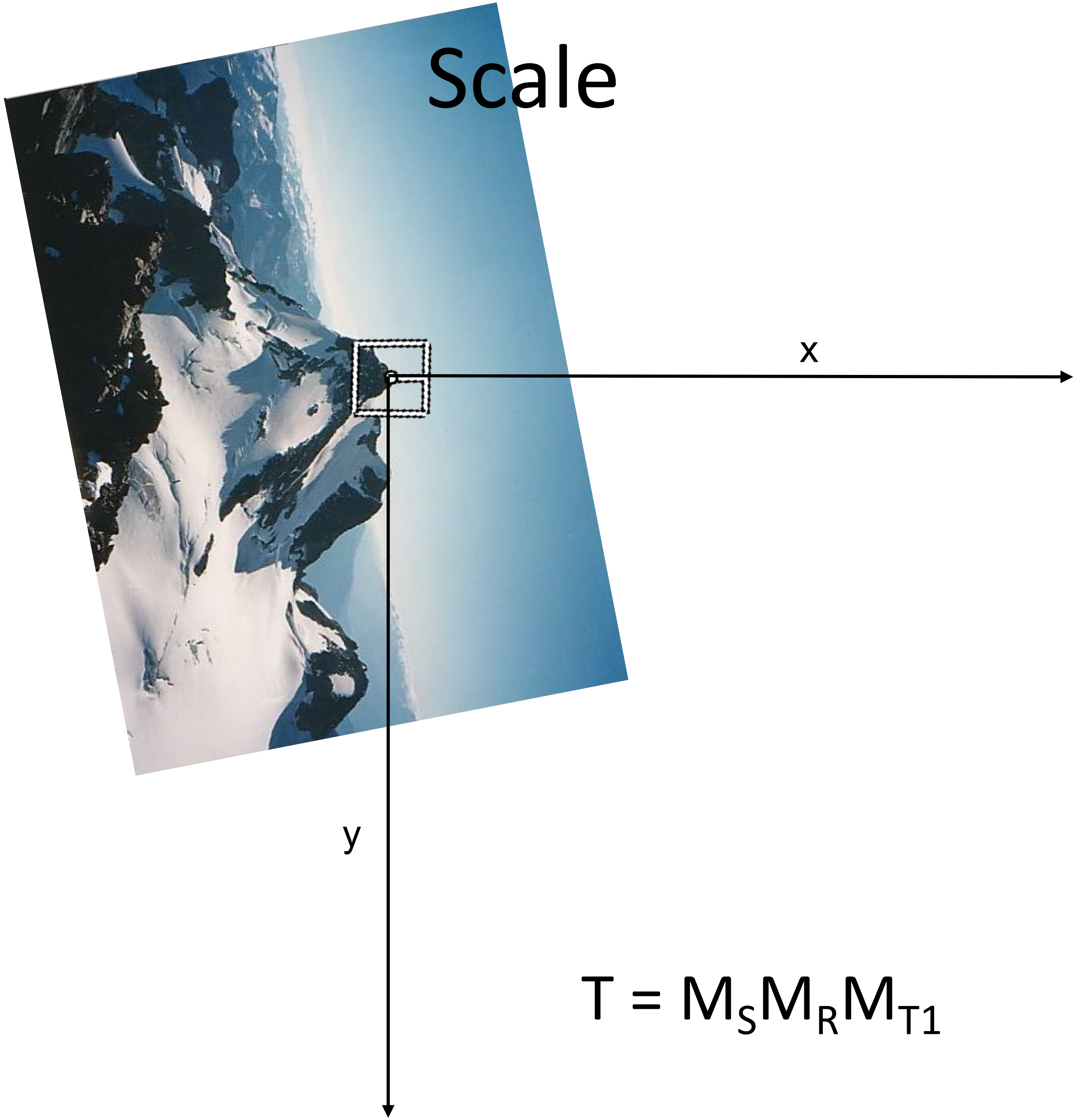
$T = M_{T1}$

Rotate



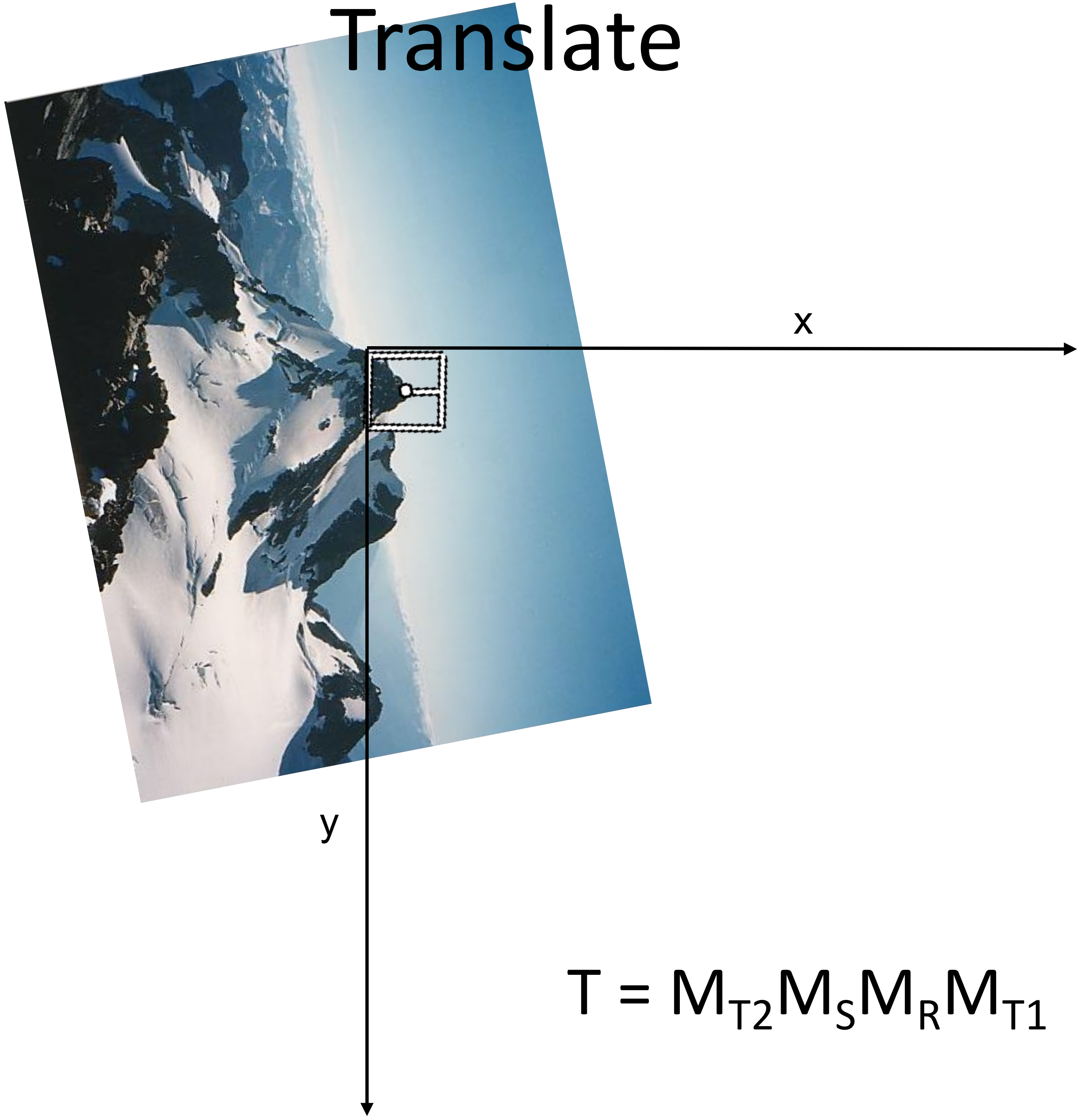
$$T = M_R M_{T1}$$

Scale



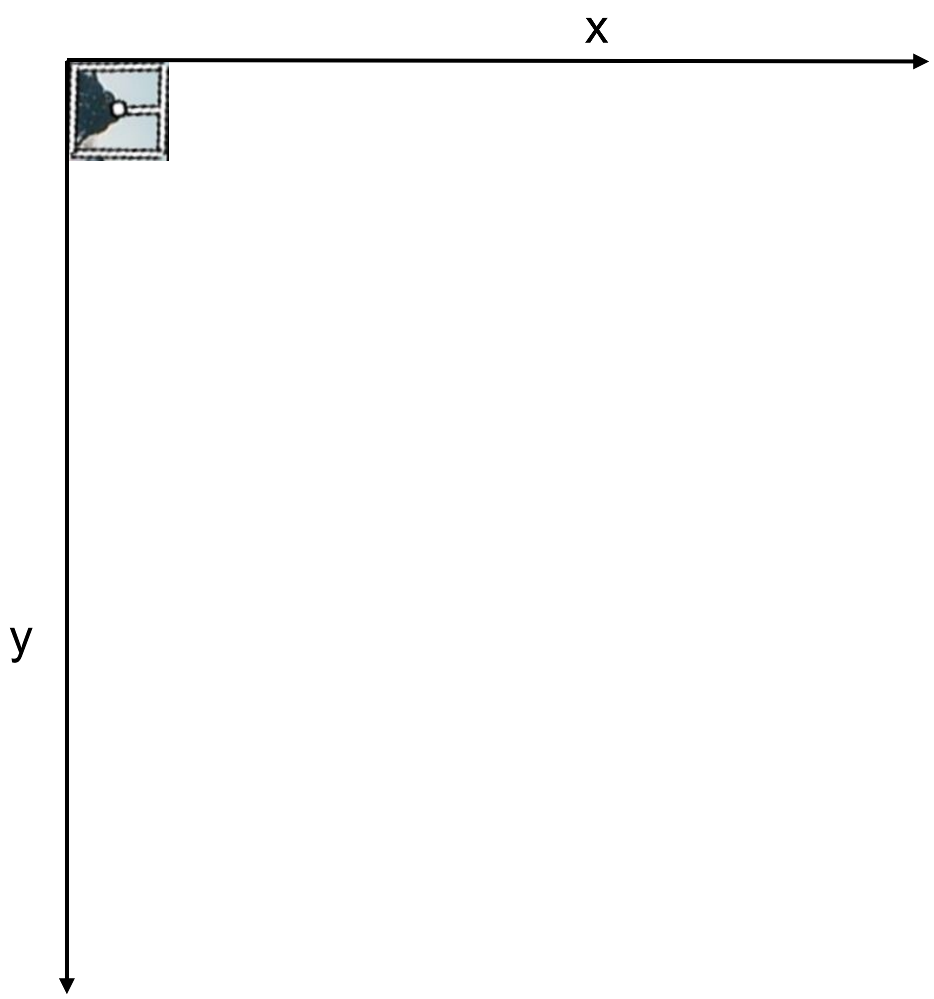
$$T = M_S M_R M_{T1}$$

Translate



$$T = M_{T2} M_S M_R M_{T1}$$

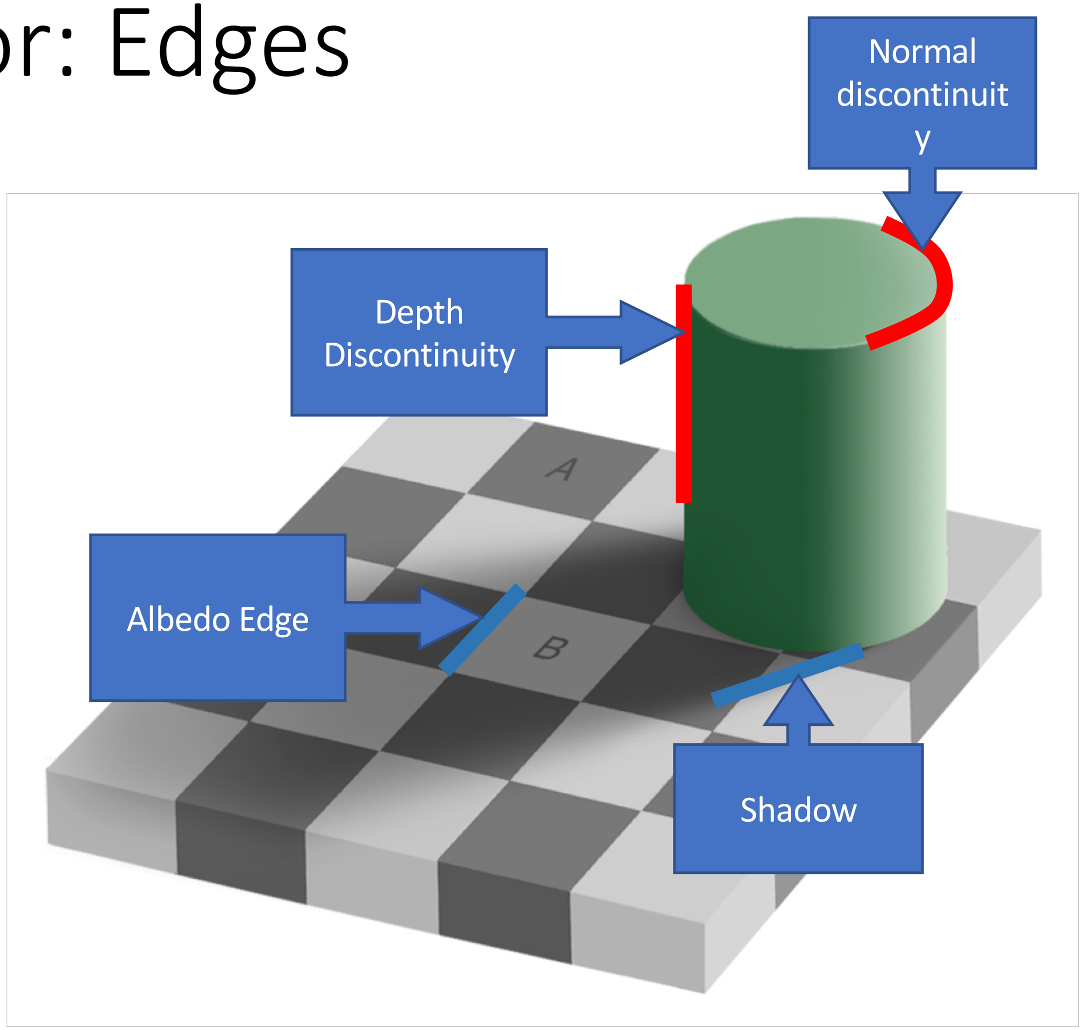
Crop



Color and Lighting

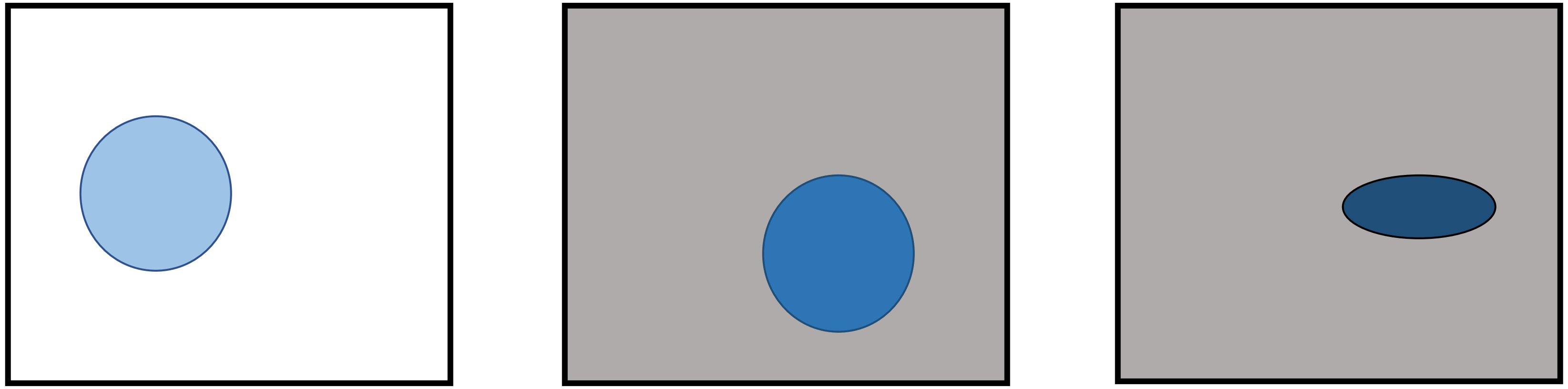
- Have invariance to additive and multiplicative changes to intensity
- But what about more sophisticated changes to intensity?

Better representation than color: Edges



Out-of-plane rotation

- Invariant to translation and rotation
- But what about more sophisticated geometric transformations



Out-of-plane rotation

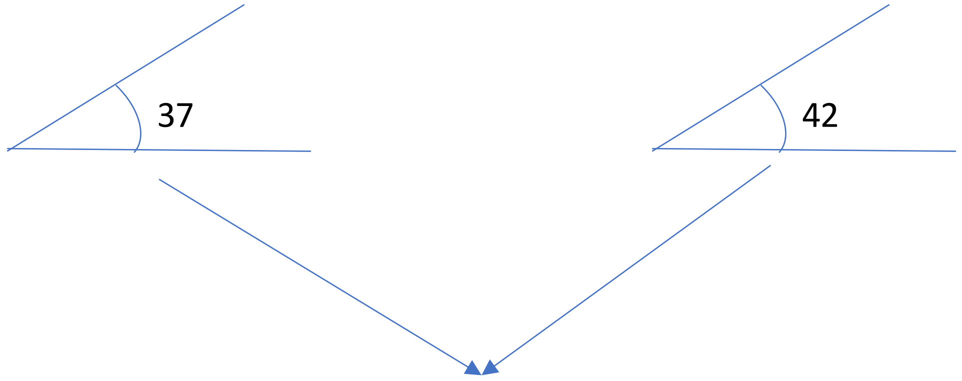
Towards a better feature descriptor

- Match *pattern of edges*
  - Edge orientation – clue to shape
  - Invariant to almost all photometric transformations

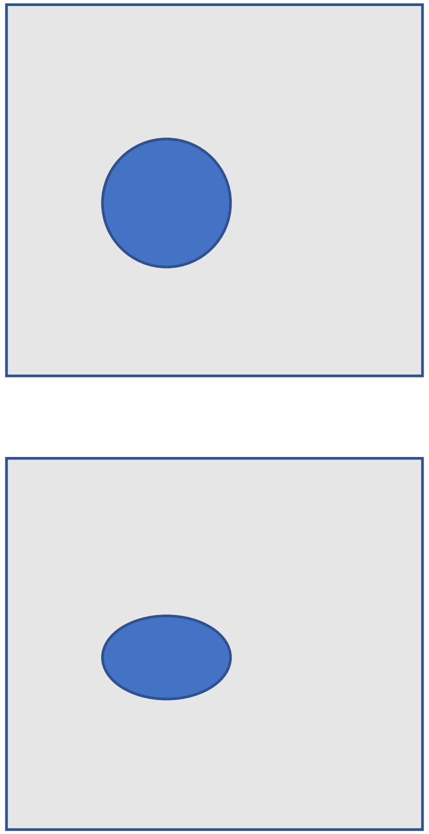
- Be resilient to *small deformations*
  - Deformations might move pixels around, but slightly
  - Deformations might change edge orientations, but slightly

Invariance to deformation

- Precise edge orientations are not resilient to out-of-plane rotations and deformations
- But we can *quantize* edge orientation: only record rough orientation



Between 30 and 45

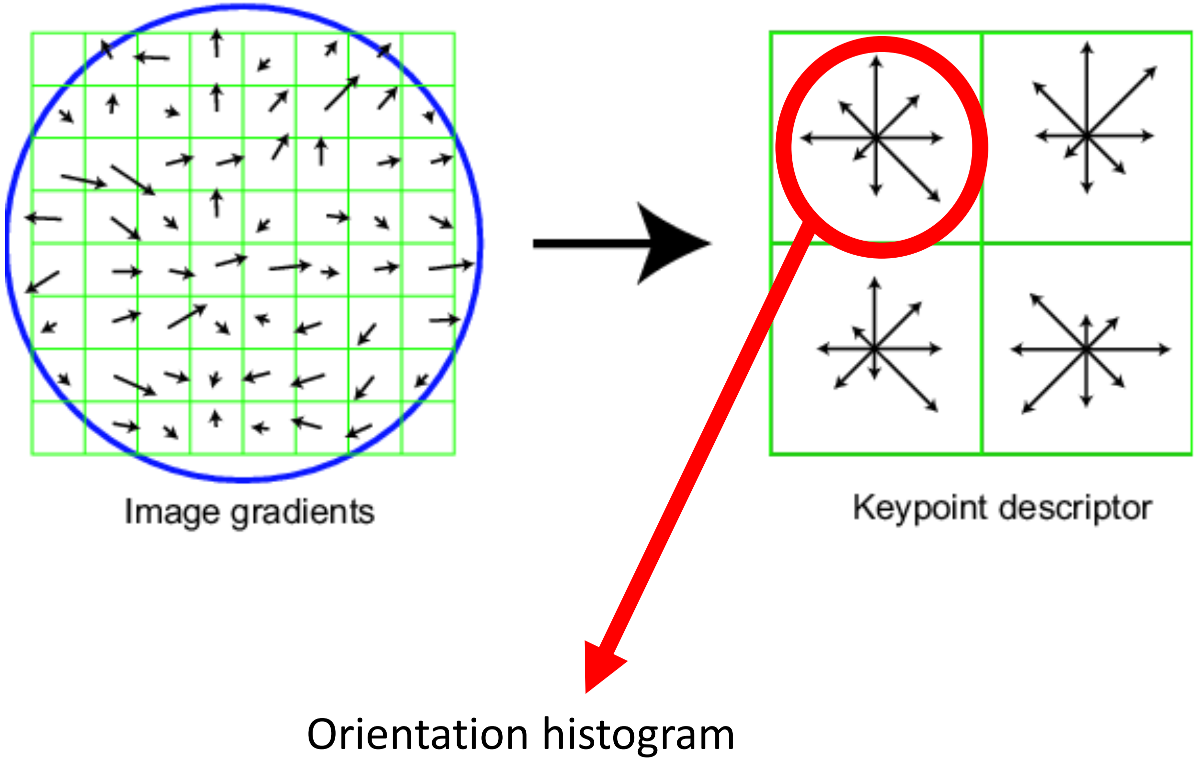


Invariance to deformation

$$g(\theta) = \begin{cases} 0 & \text{if } 0 < \theta < 2\pi/N \\ 1 & \text{if } 2\pi/N < \theta < 4\pi/N \\ 2 & \text{if } 4\pi/N < \theta < 6\pi/N \\ & \dots \\ N-1 & \text{if } 2(N-1)\pi/N \end{cases}$$

Invariance to deformation

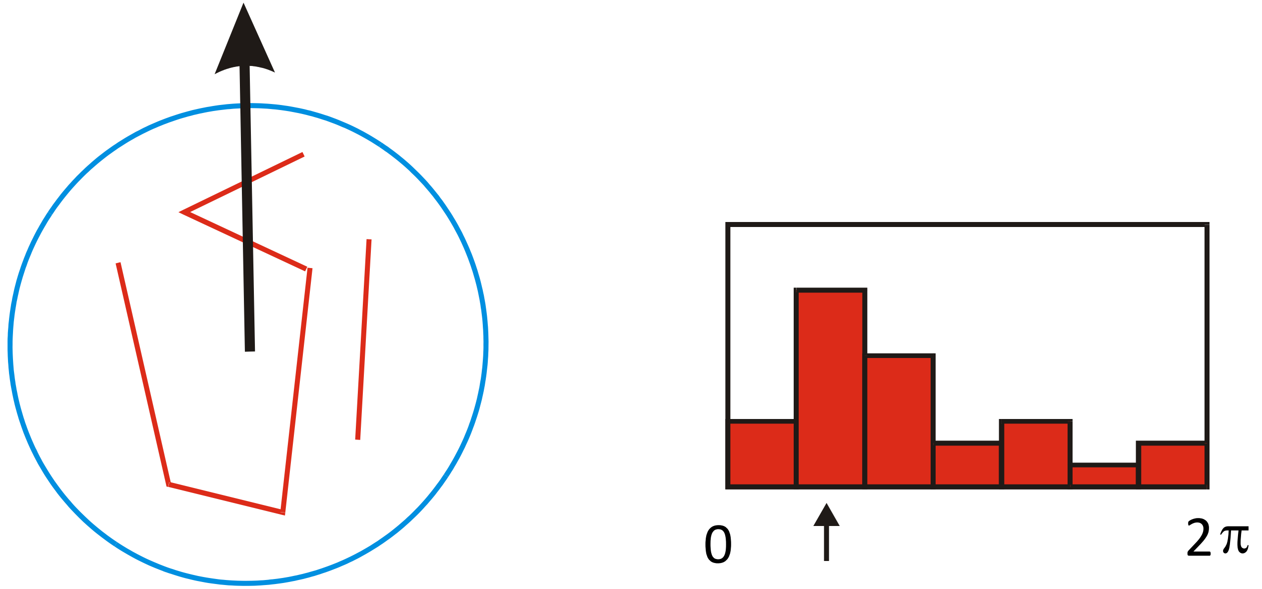
- Deformation can also move pixels around
- Again, instead of precise location of each pixel, only want to record rough location
- Divide patch into a grid of *cells*
- Record *counts* of each orientation in each cell: *orientation histograms*



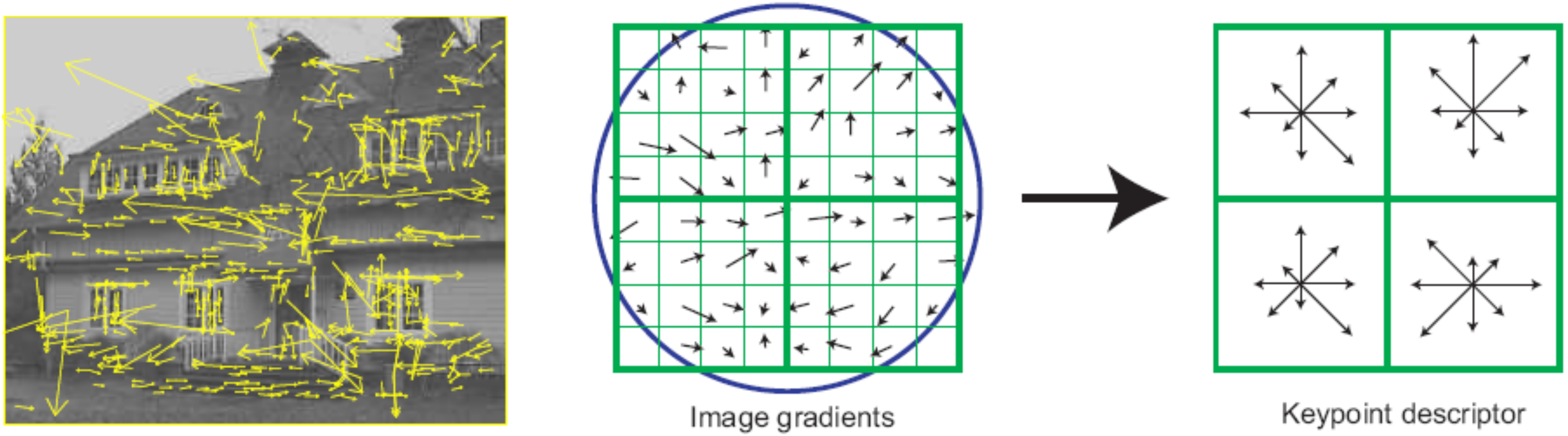
Rotation Invariance by Orientation Normalization

[Lowe, SIFT, 1999]

- Compute orientation histogram
- Select dominant orientation (mode of the histogram; alternative to eigenvector of second moment matrix)
- Normalize: rotate to fixed orientation



The SIFT descriptor



SIFT – Lowe IJCV 2004

Scale Invariant Feature Transform

Basic idea:

- DoG for scale-space feature detection
- Take 16x16 square window around detected feature
    - Compute gradient orientation for each pixel
    - Throw out weak edges (threshold gradient magnitude)
    - Create histogram of surviving e

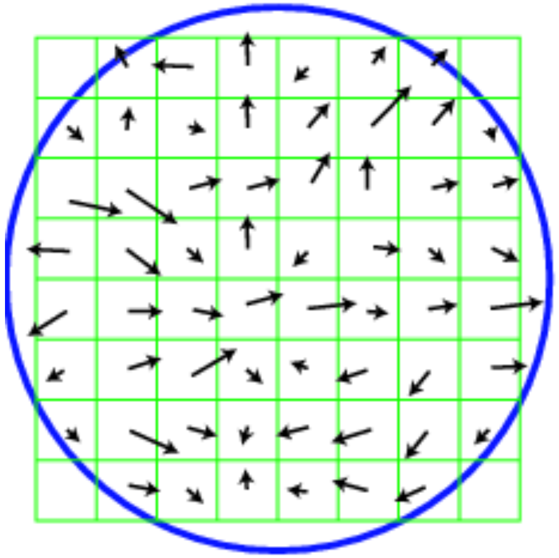
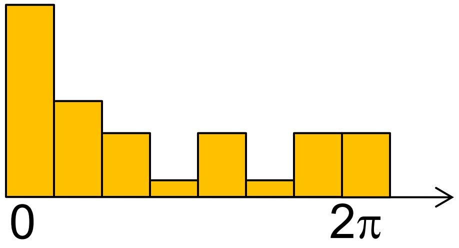
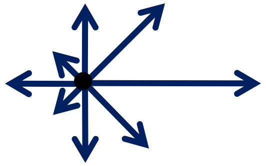


Image gradients



angle histogram



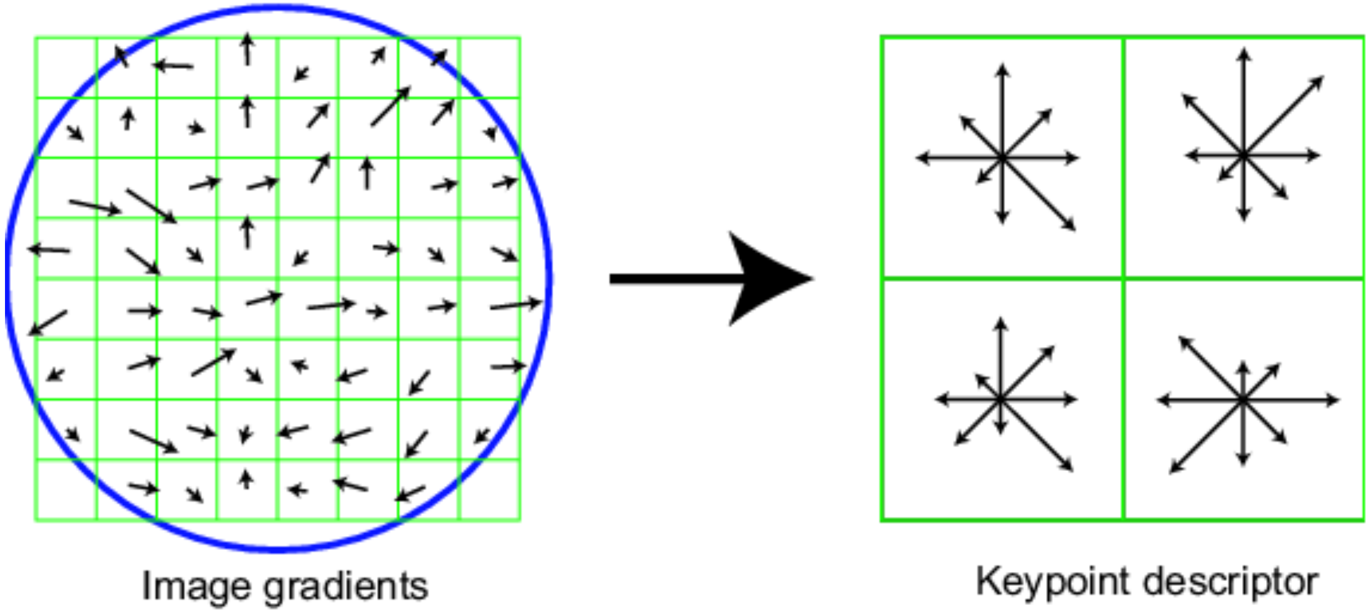
Keypoint descriptor

Adapted from slide by David Lowe

SIFT descriptor

Create histogram

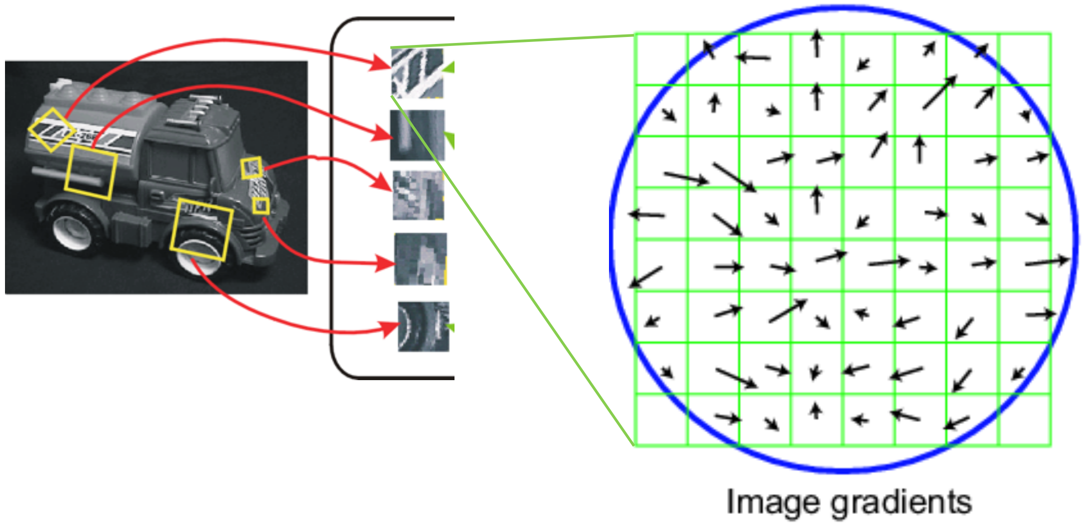
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Adapted from slide by David Lowe

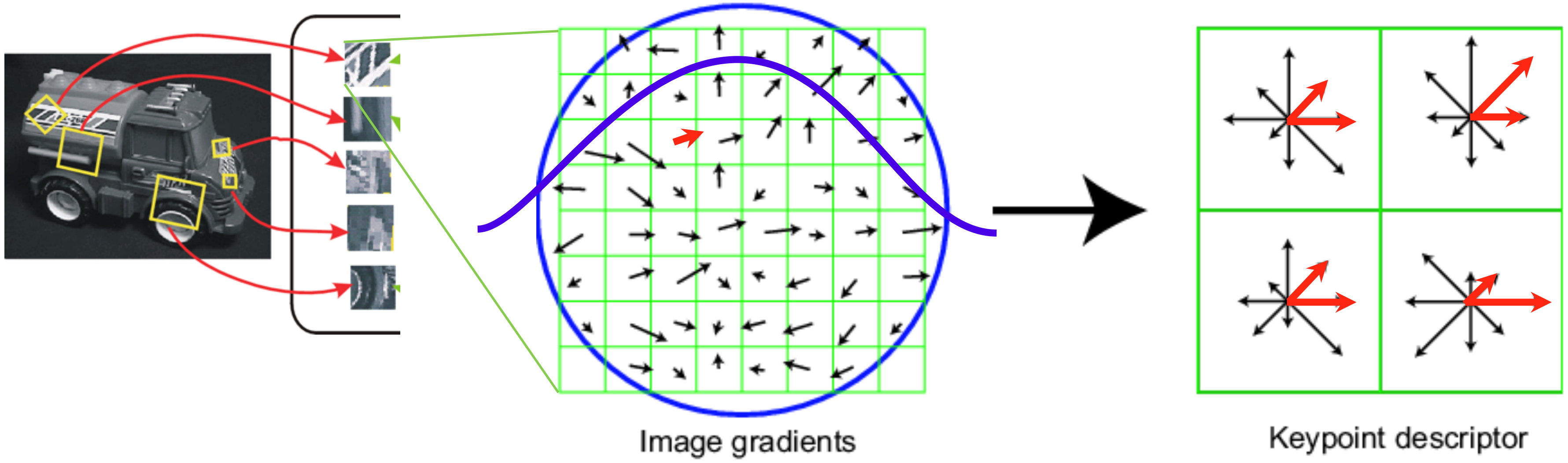
SIFT vector formation

- Computed on rotated and scaled version of window according to computed orientation & scale
  - resample the window
- Based on gradients weighted by a Gaussian



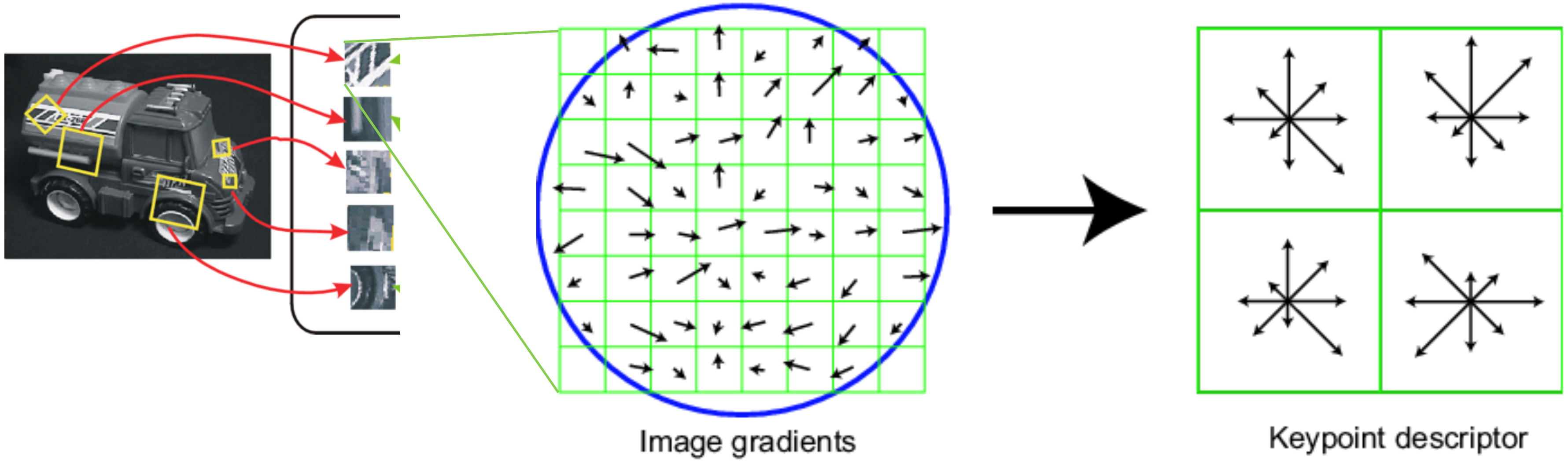
Ensure smoothness

- Trilinear interpolation
  - a given gradient contributes to 8 bins:
    4 in space times 2 in orientation



Reduce effect of illumination

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - after normalization, clamp gradients >0.2
  - renormalize



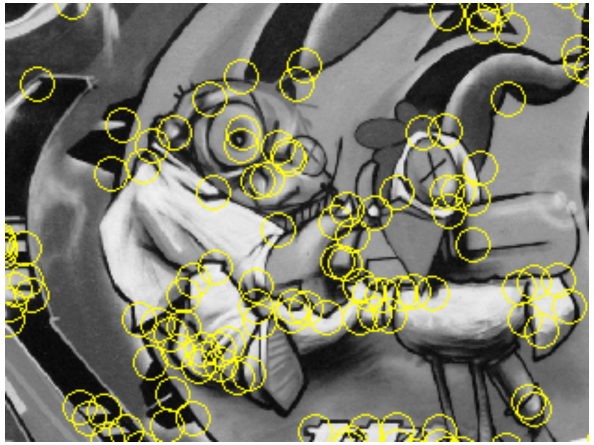
Properties of SIFT

Extraordinarily robust matching technique

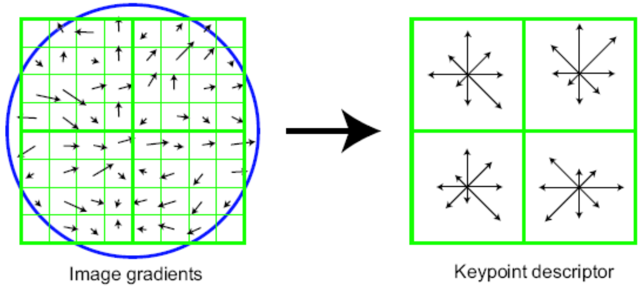
- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available: http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT


Summary

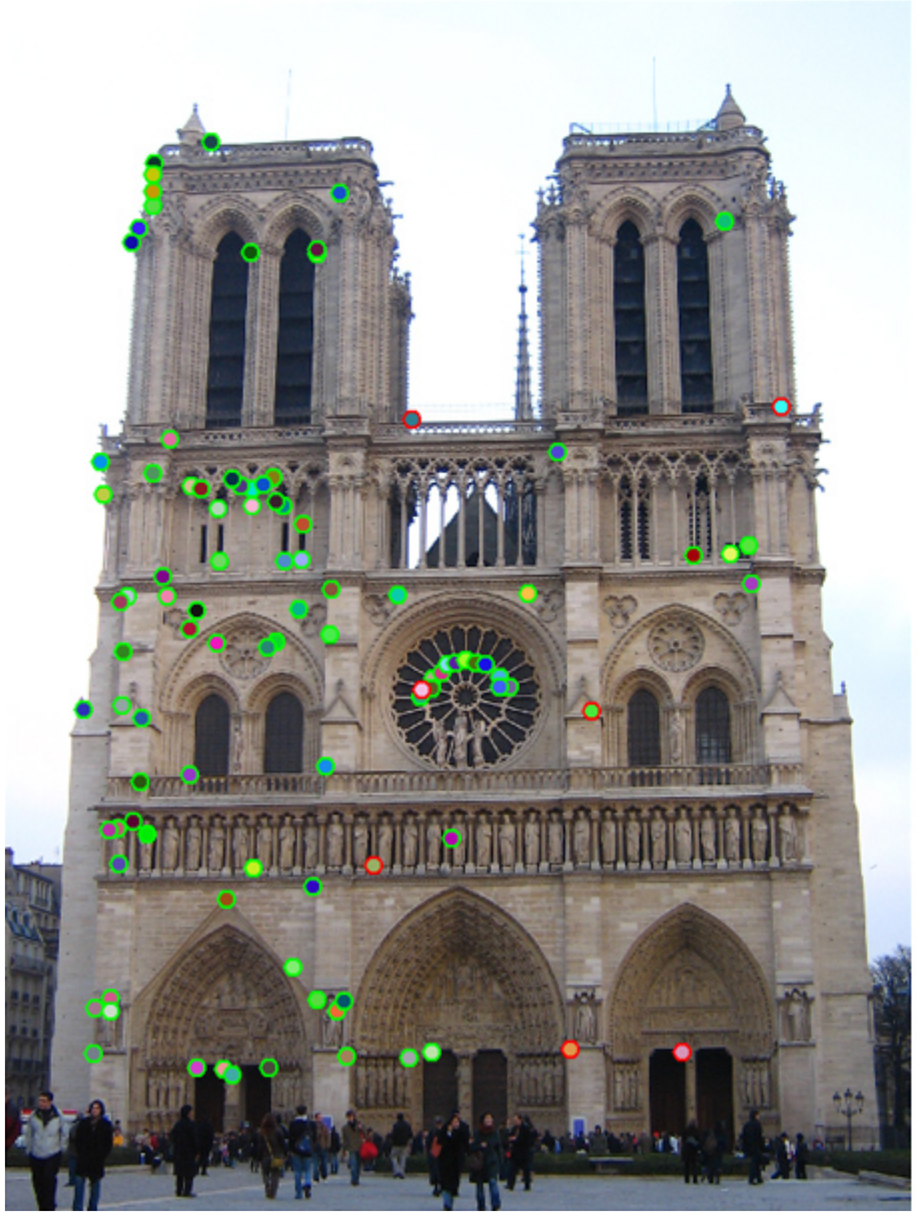
- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Harris, DoG

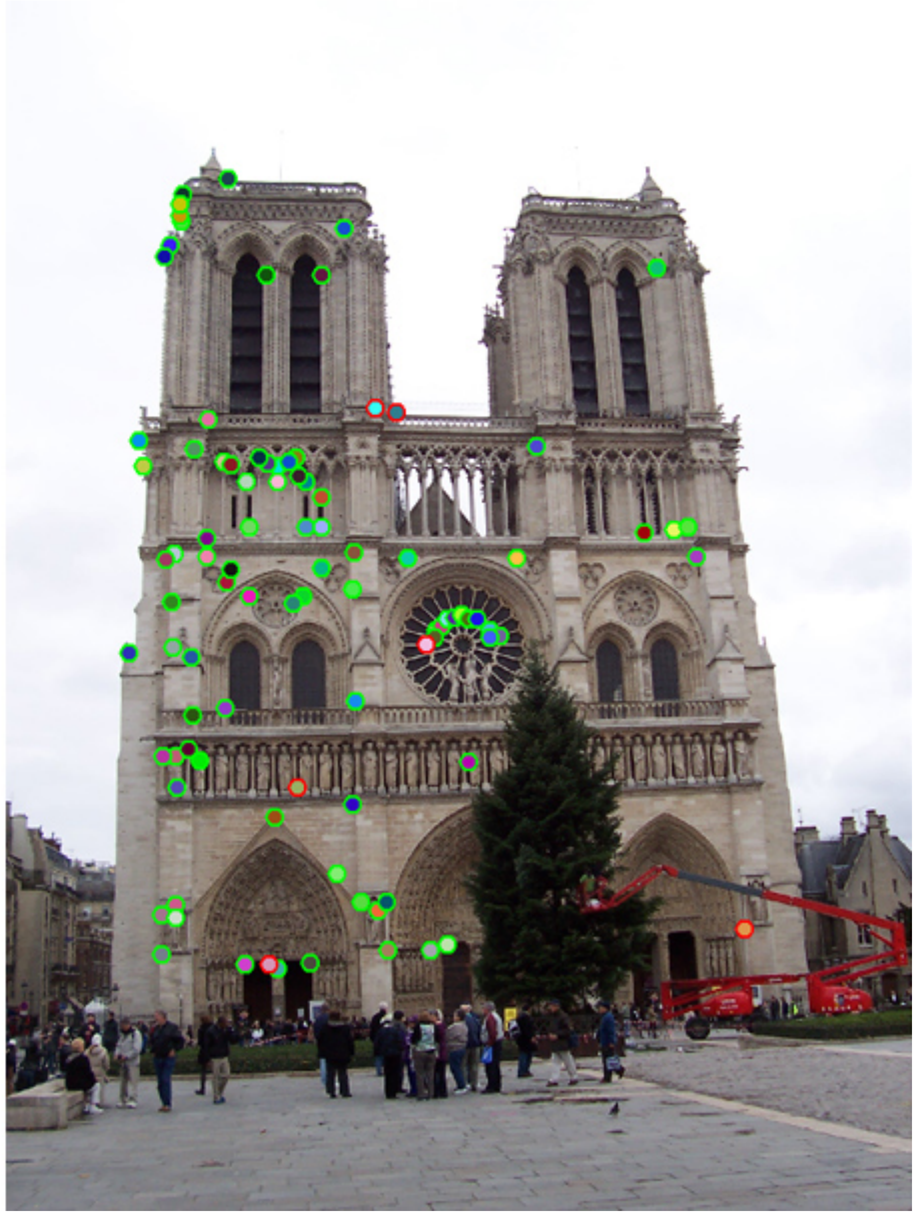


- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT and variants are typically good for stitching and recognition
  - But, need not stick to one



Which features match?





Feature matching

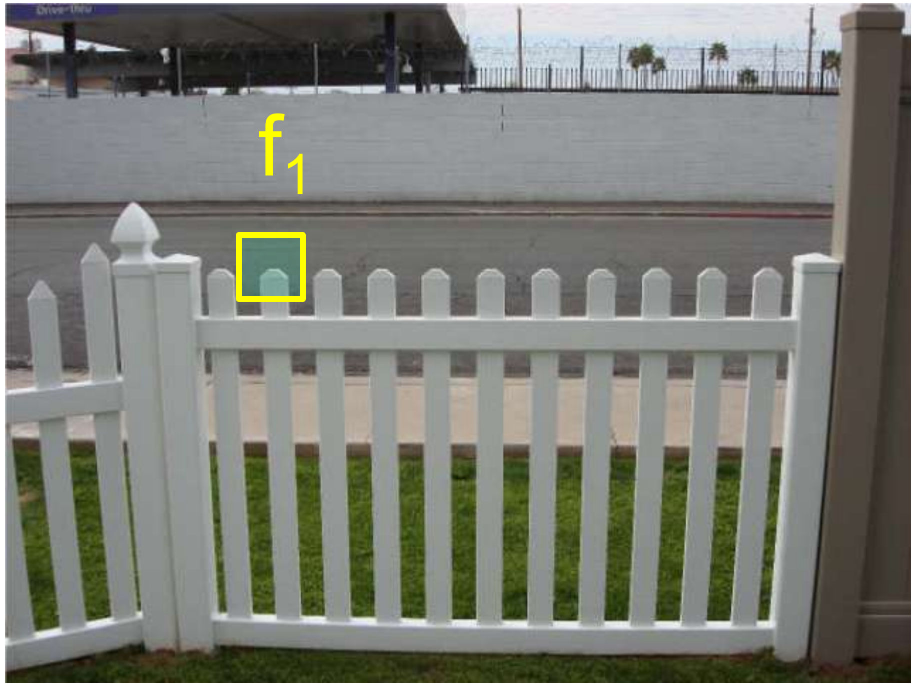
Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

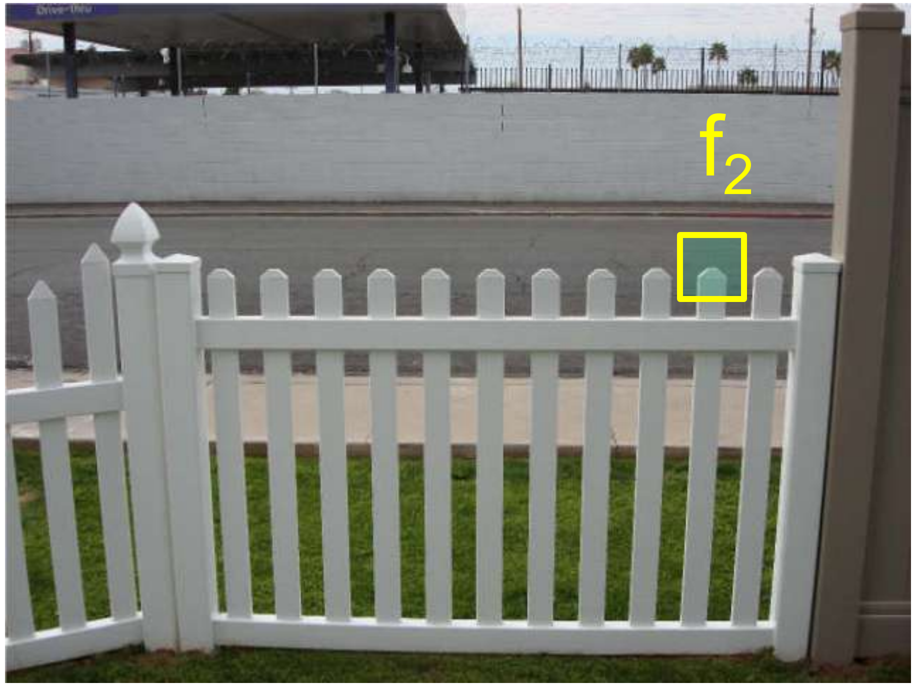
Feature distance

How to define the difference between two features f_1, f_2?

- Simple approach: L_2 distance, $||f_1 - f_2||$
- can give good scores to ambiguous (incorrect) matches



I_1

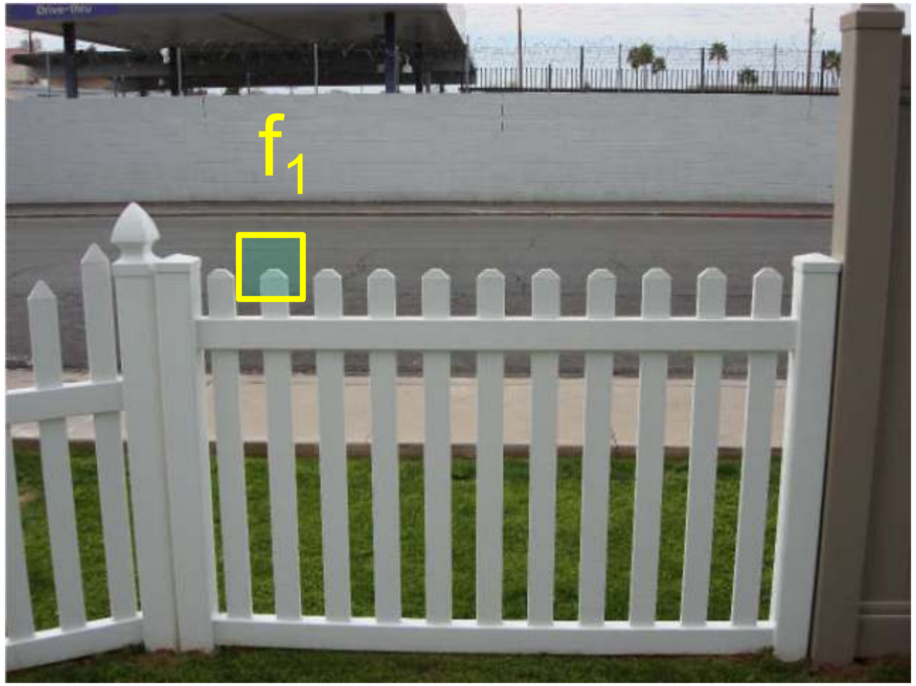


I_2

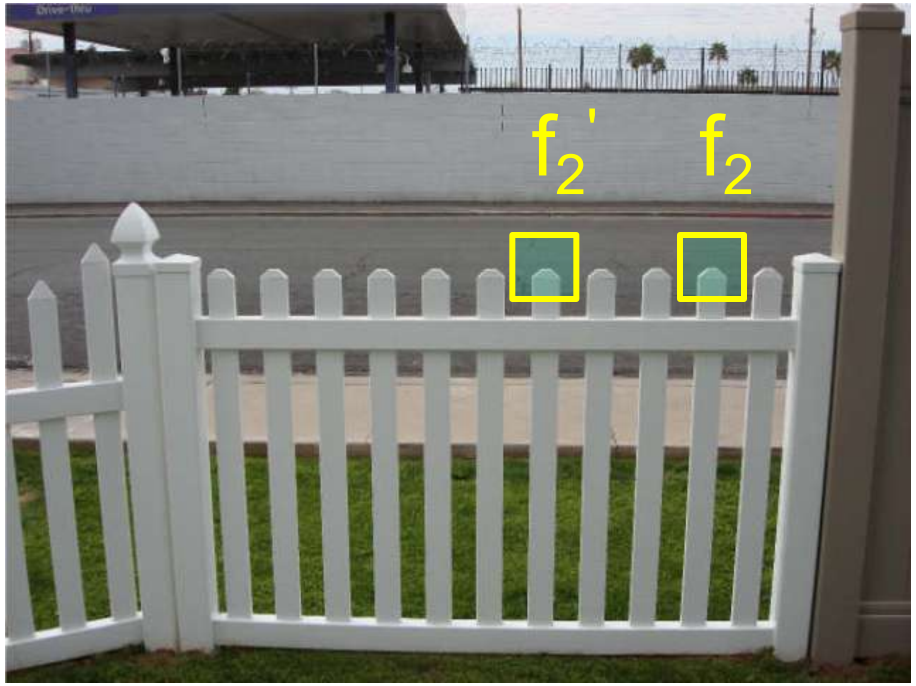
Feature distance

How to define the difference between two features f_1, f_2?

- Better approach: ratio distance = $||f_1 - f_2 || / || f_1 - f_2' ||$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2
  - gives large values for ambiguous matches



I_1



I_2