Feature descriptors and matching

Basic correspondence

- Image patch as descriptor, NCC as similarity
- Invariant to?
  - Photometric transformations?
  - Translation?
  - Rotation?
  - Scaling?

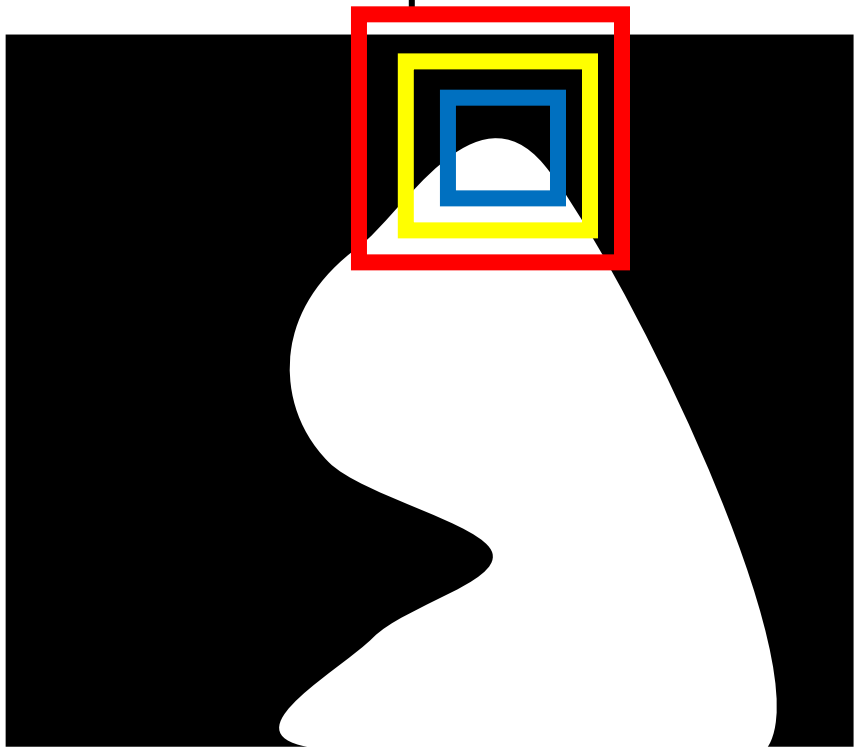
Rotation invariance for feature descriptors

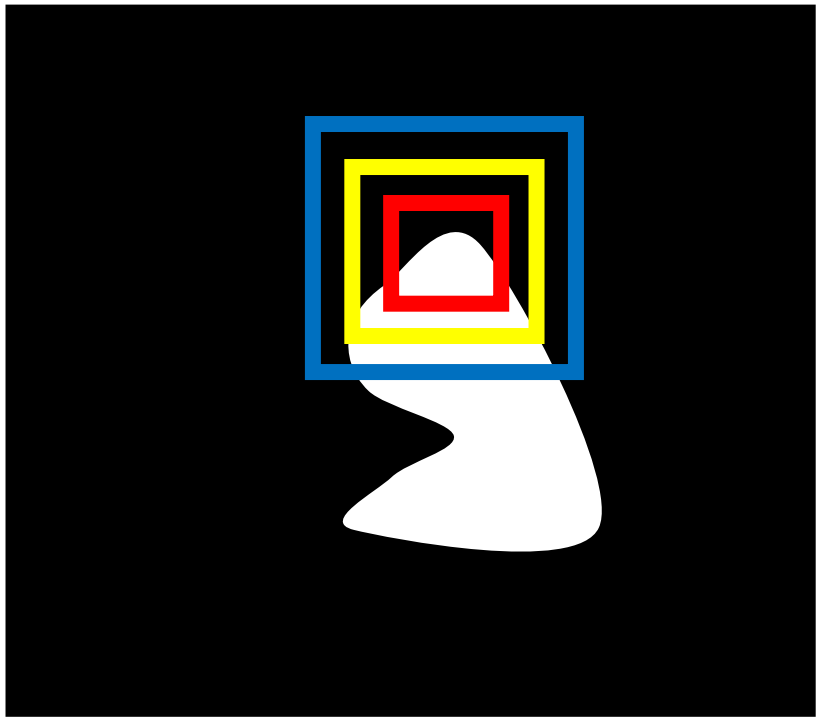
- Find dominant orientation of the image patch
  - This is given by $\mathbf{x}_{max}$, the eigenvector of $\mathbf{M}$ corresponding to λ_{max} (the *larger* eigenvalue)
  - Rotate the patch according to this angle
  - Figure: line represents x_{max}, box represents patch we take as feature descriptor

Figure by Matthew Brown

Scale invariance for feature descriptors

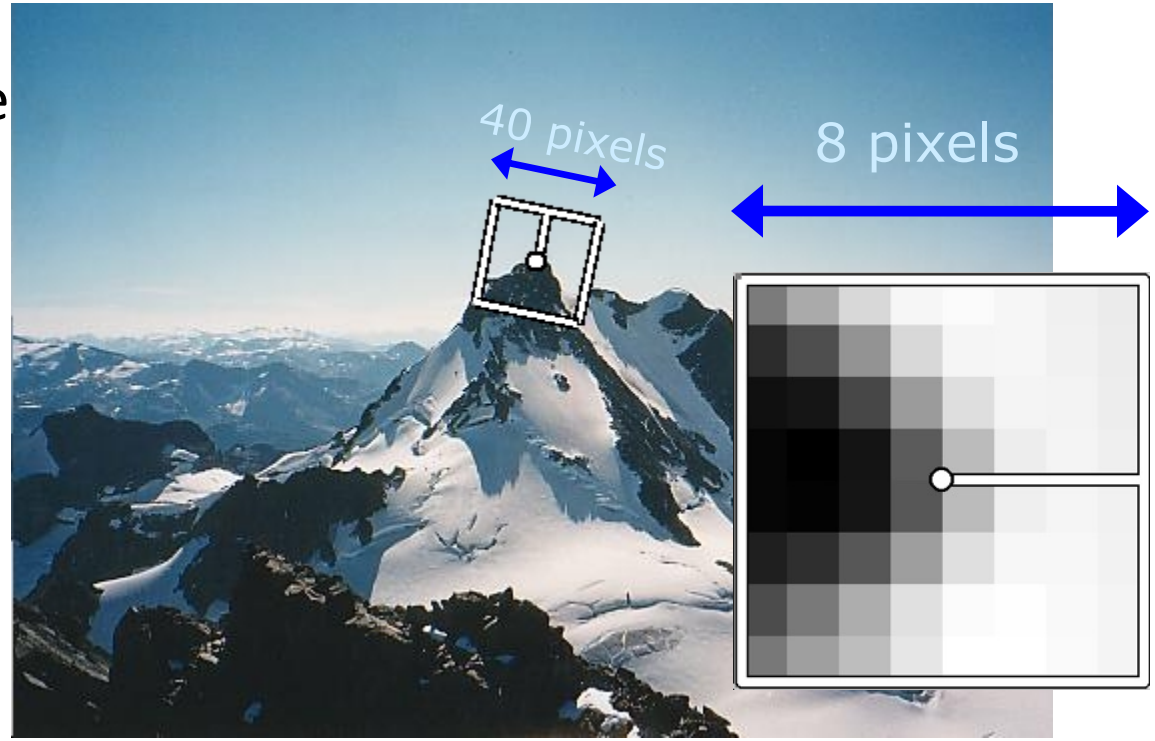
- Recall that corner detector searches over scales for maximum response: record scale which gives maximum response
- Use a patch of the same scale, then resize it to a fixed size





Multiscale Oriented PatcheS descriptor

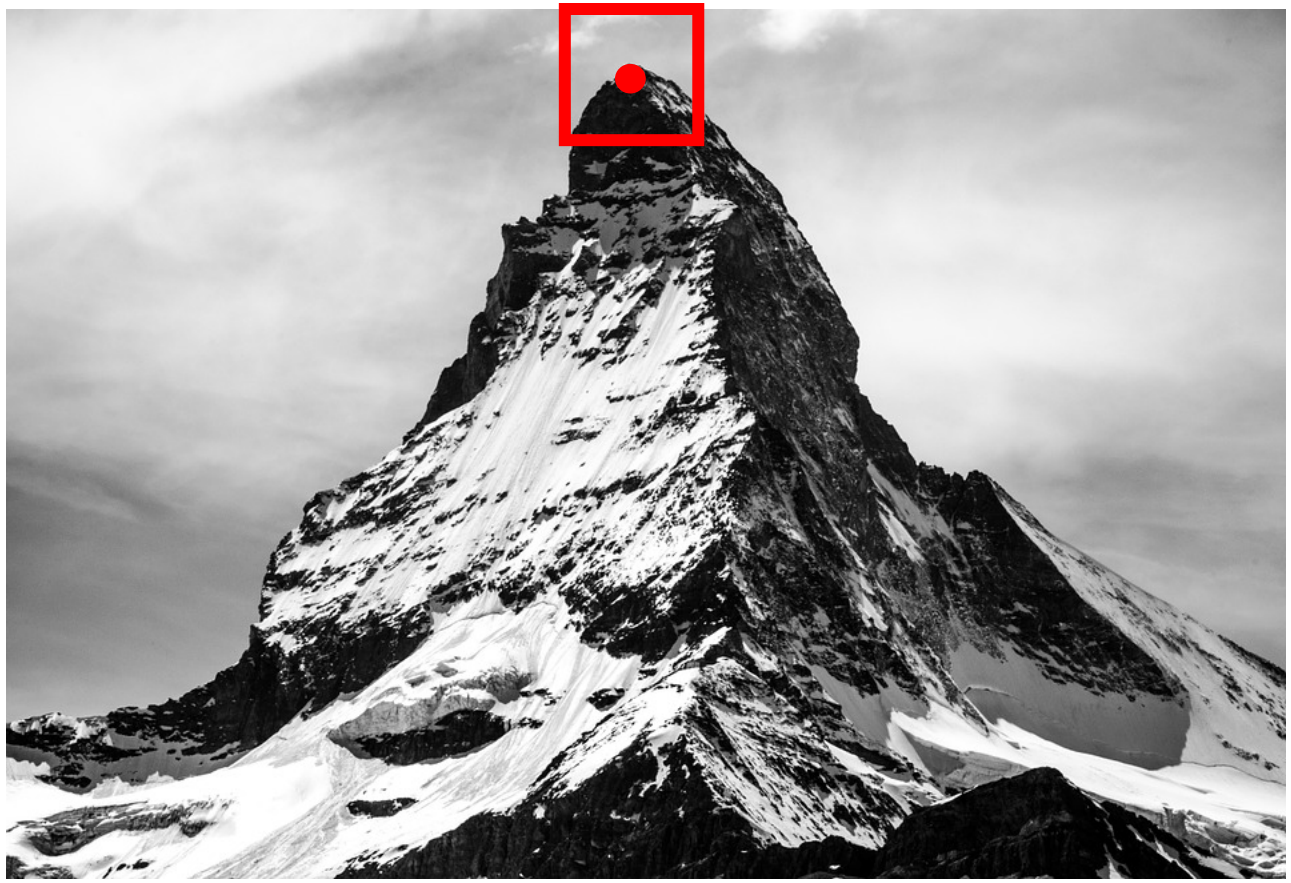
- Take 40x40 *oriented* square window around detected feature at *appropriate scale*
- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Adapted from slide by Matthew Brown

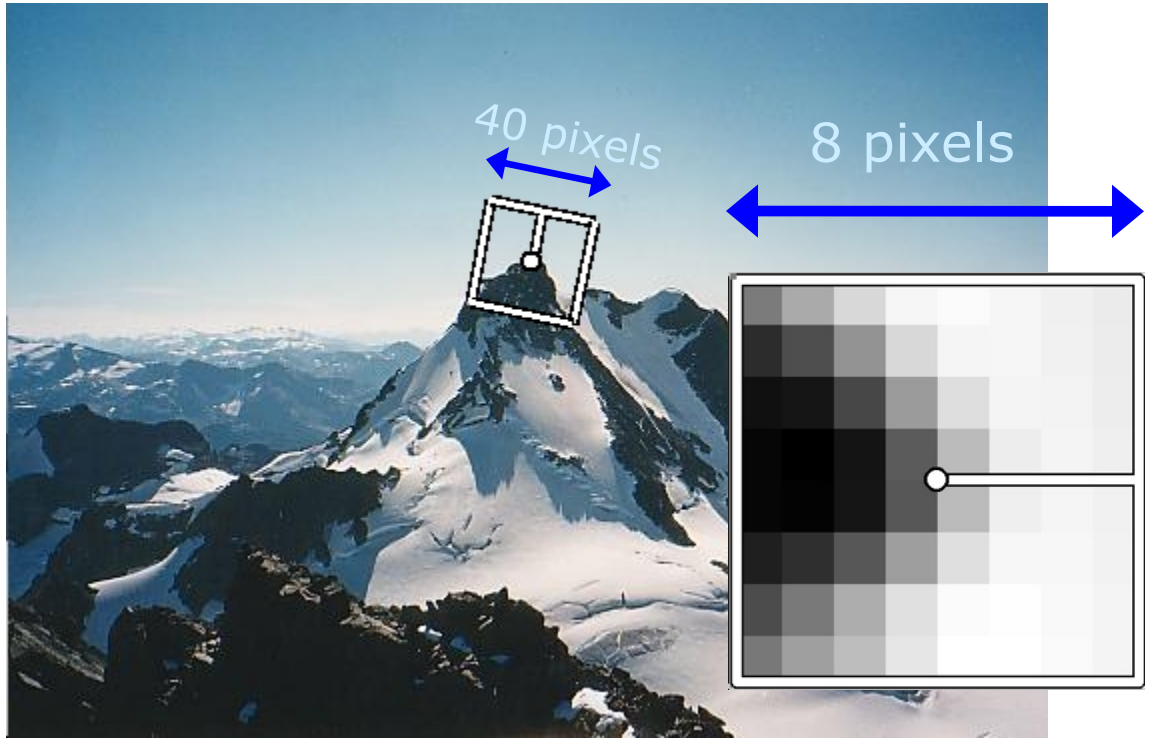
Multiscale Oriented PatcheS descriptor

- 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



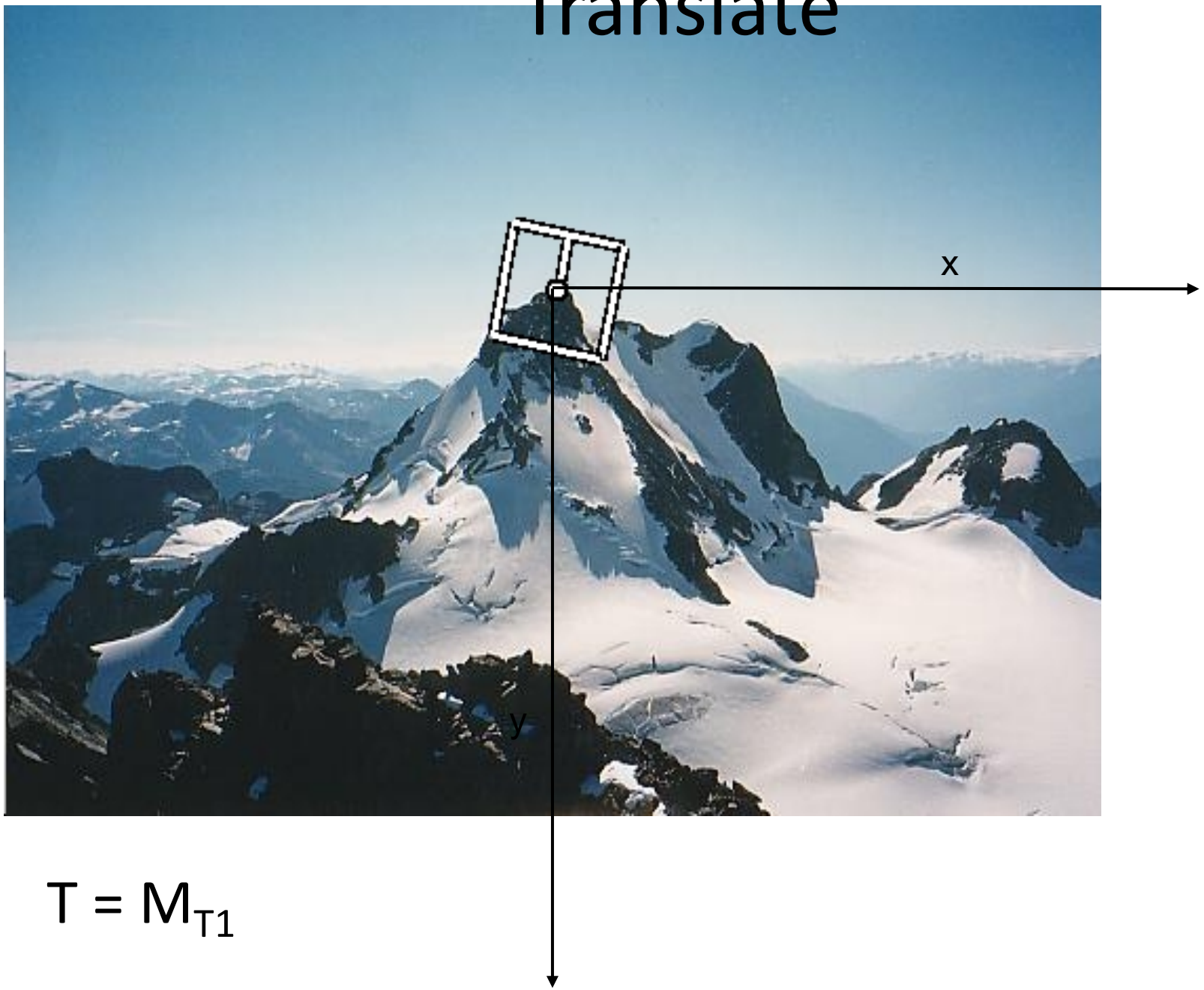
Adapted from slide by Matthew Brown

MOPS descriptor

•You can combine transformations together to get the final transformation

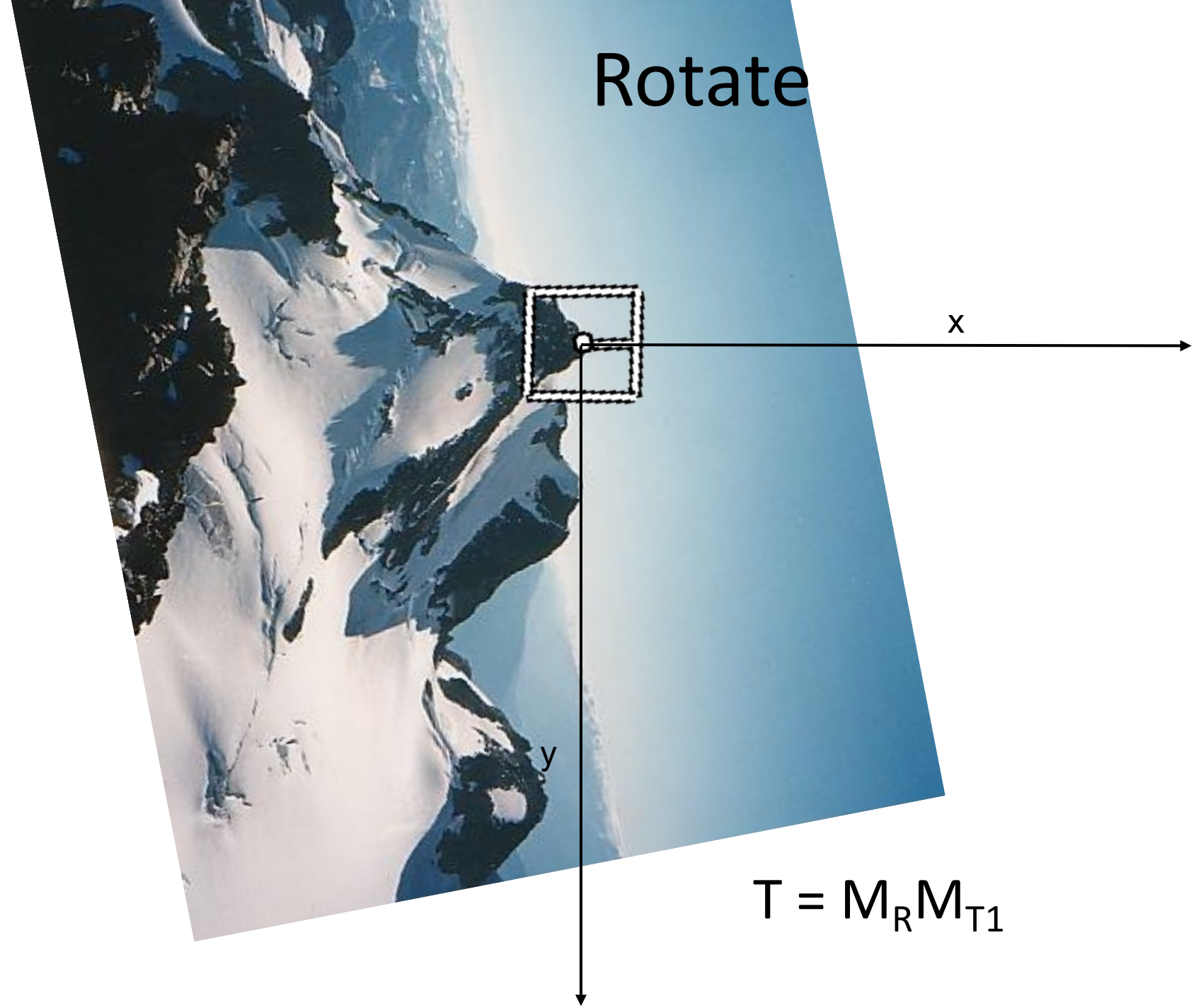
T = ?

Translate



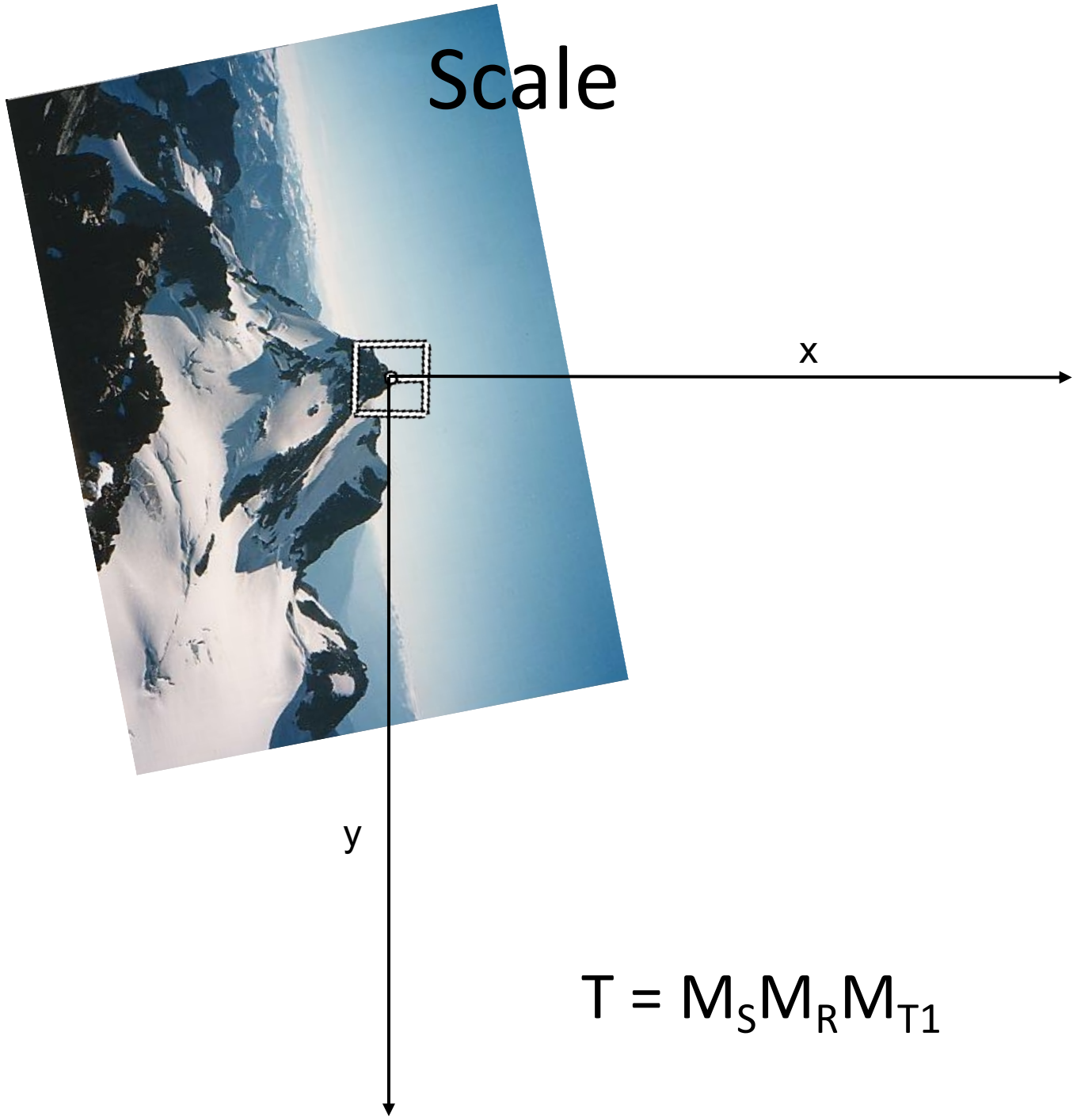
$T = M_{T1}$

Rotate



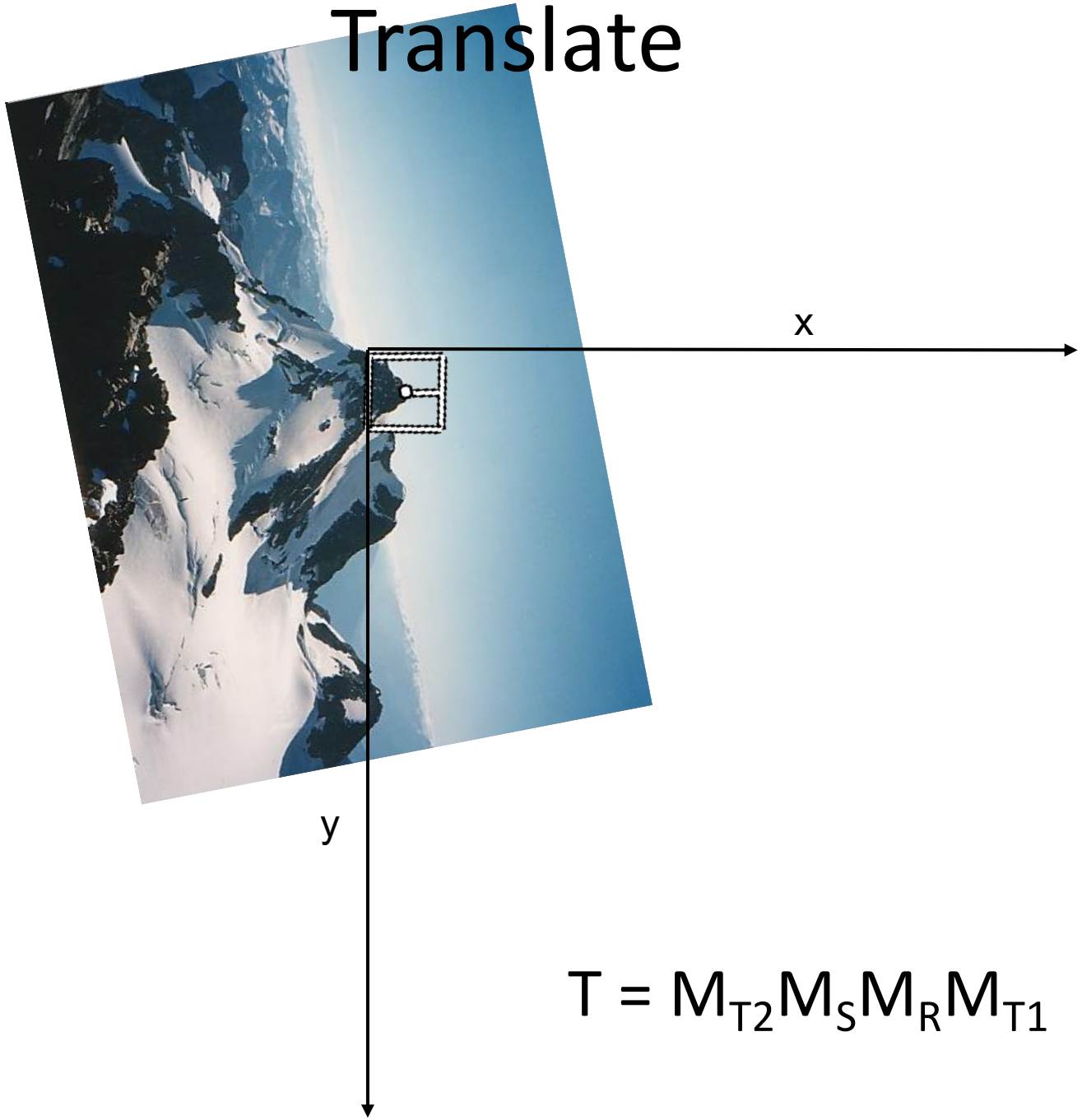
$$T = M_R M_{T1}$$

Scale



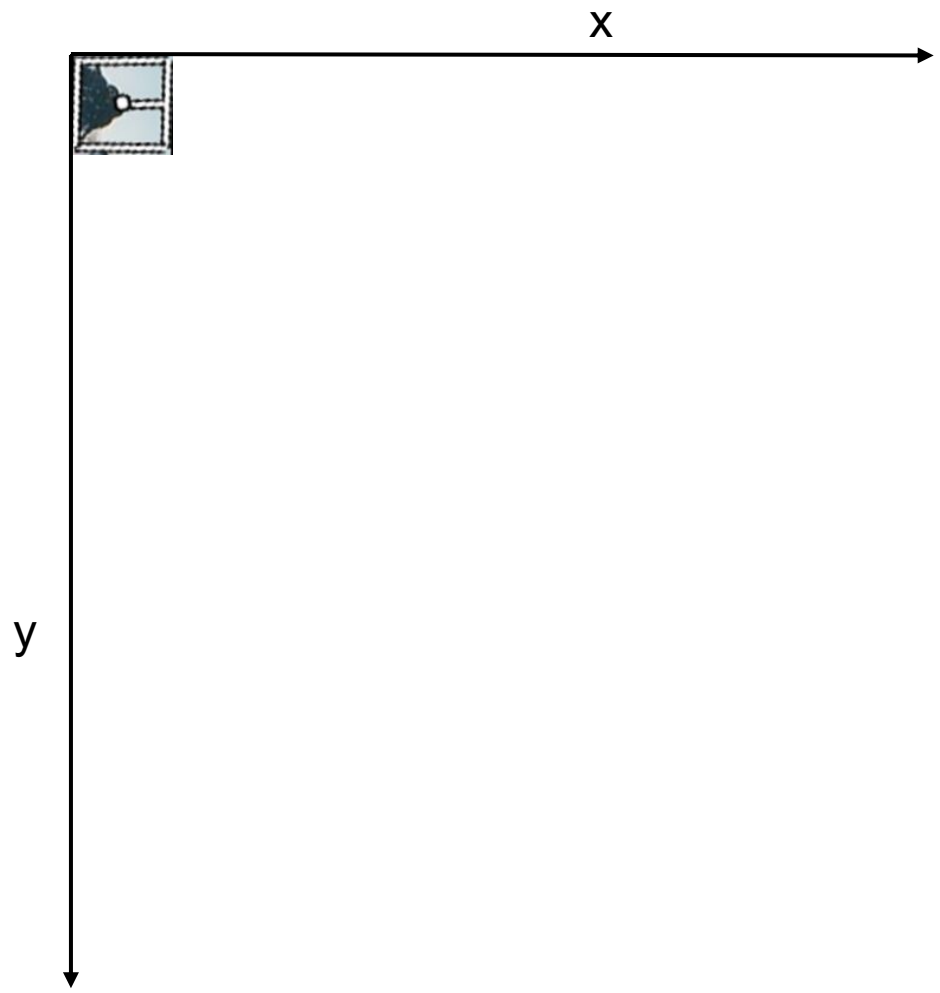
$$T = M_S M_R M_{T1}$$

Translate



$$T = M_{T2}M_SM_RM_{T1}$$

Crop



MOPS

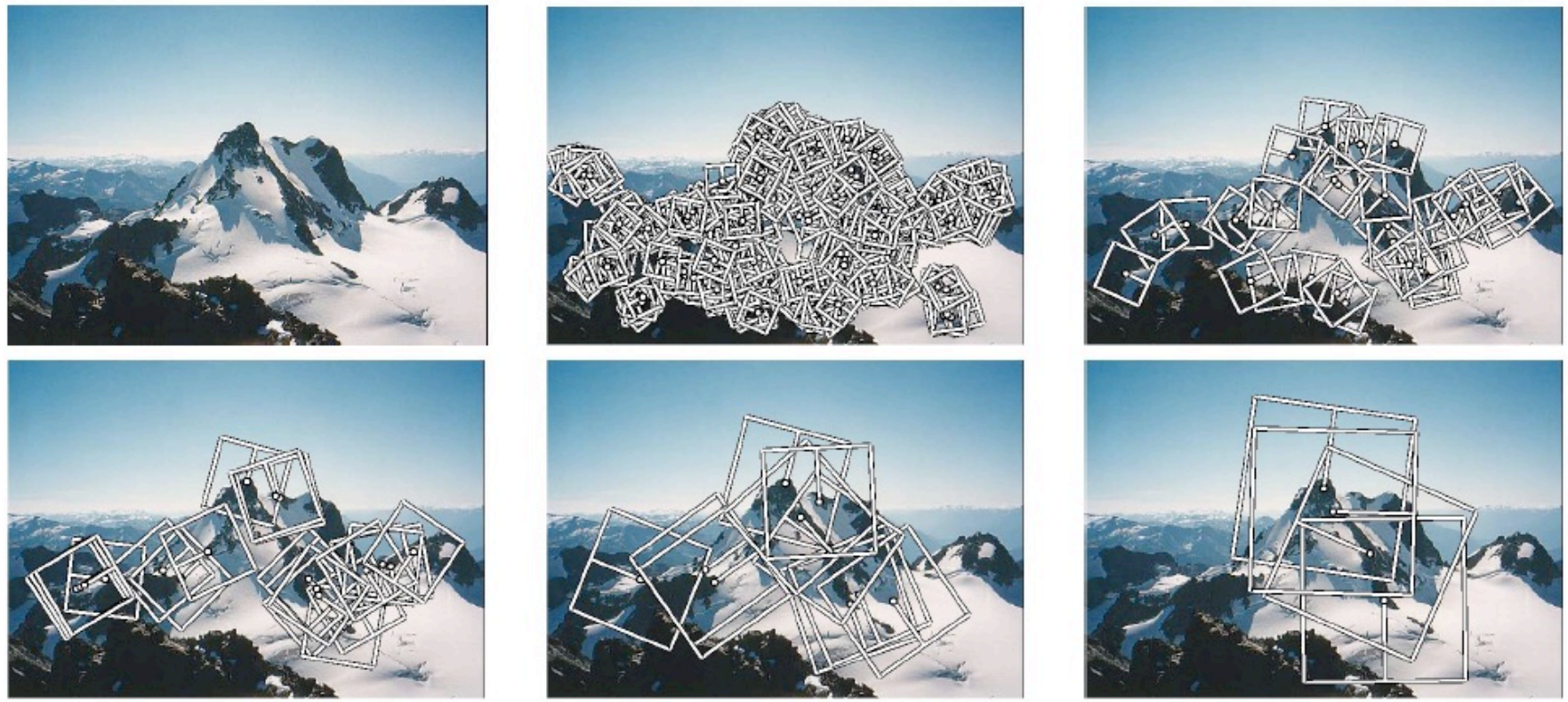
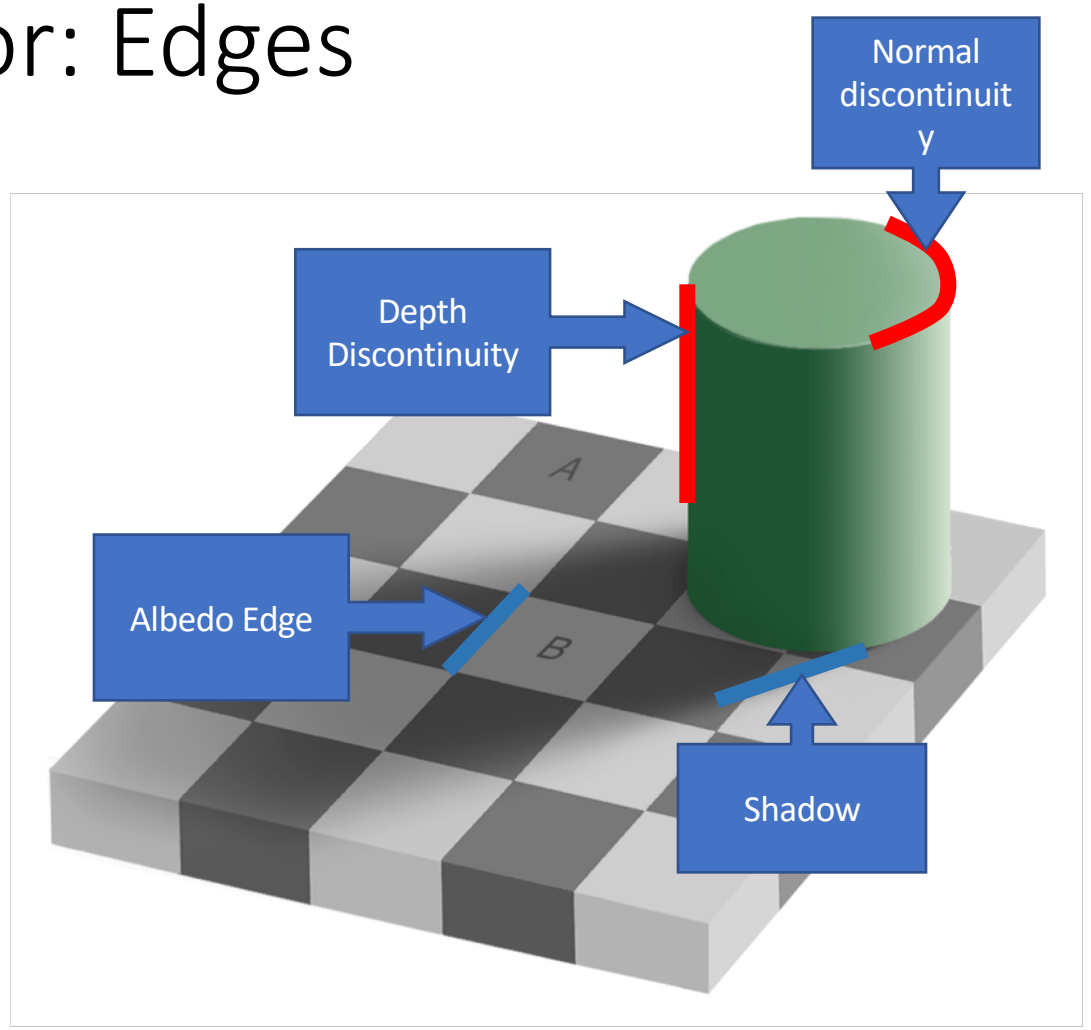


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Color and Lighting

- MOPS invariant to additive and multiplicative changes to intensity
- But what about more sophisticated changes to intensity?

Better representation than color: Edges



Beyond translation and rotation

- MOPS invariant to translation and rotation
- But what about more sophisticated geometric transformations
  - World is 3D, so 2D translation and rotation not enough: *out-of-plane rotations*

patch

patch

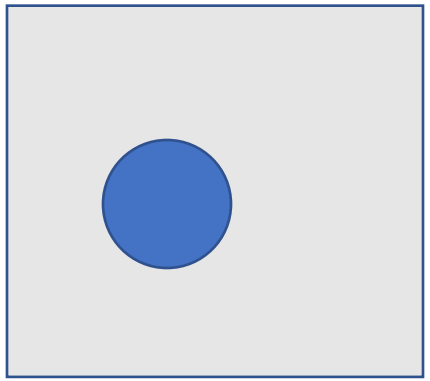
Towards a better feature descriptor

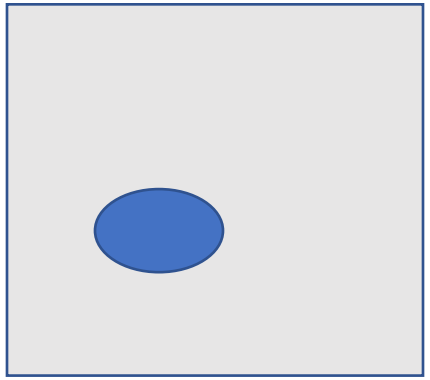
- Match *pattern of edges*
  - Edge orientation – clue to shape
  - Invariant to almost all photometric transformations

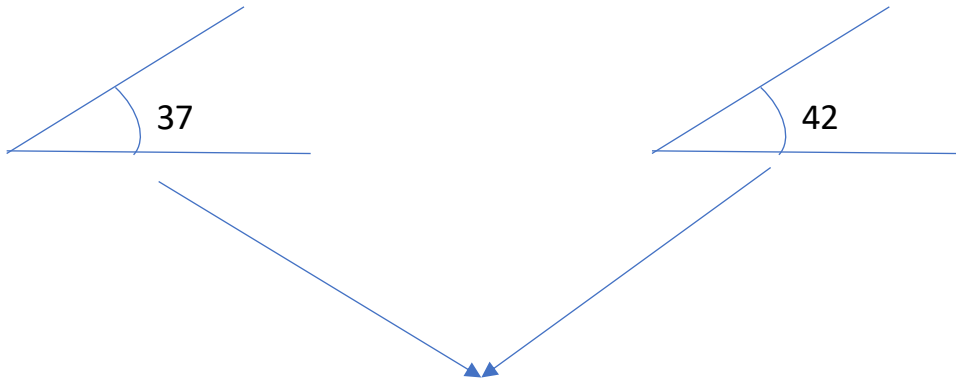
- Be resilient to *small out-of-plane rotations*
  - Small out-of-plane rotations might move pixels around in unpredictable ways
    - But only slightly
  - They might change edge orientations in unpredictable ways
    - But slightly
  - More generally, want invariance to *small deformations*

Invariance to small deformations

- Precise edge orientations are not resilient to out-of-plane rotations
- But we can *quantize* edge orientation: descriptor should only record rough orientation
  - Hope: rough orientation should stay the same!







Quantized orientation

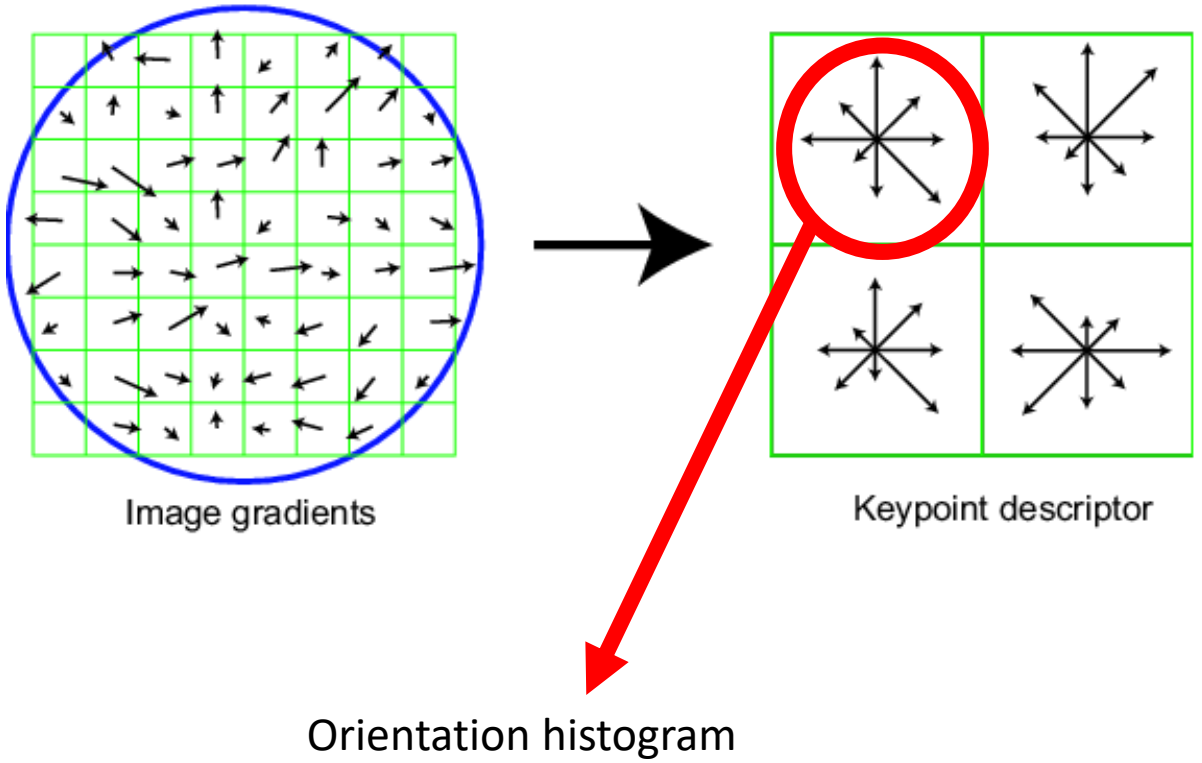
$$g(\theta) = \begin{cases} 0 & \text{if } 0 < \theta < 2\pi/N \\ 1 & \text{if } 2\pi/N < \theta < 4\pi/N \\ 2 & \text{if } 4\pi/N < \theta < 6\pi/N \\ & \cdots \\ N-1 & \text{if } 2(N-1)\pi/N \end{cases}$$

Quantization

- Quantization is a general trick
- Converts from continuous, real numbers to discrete choices

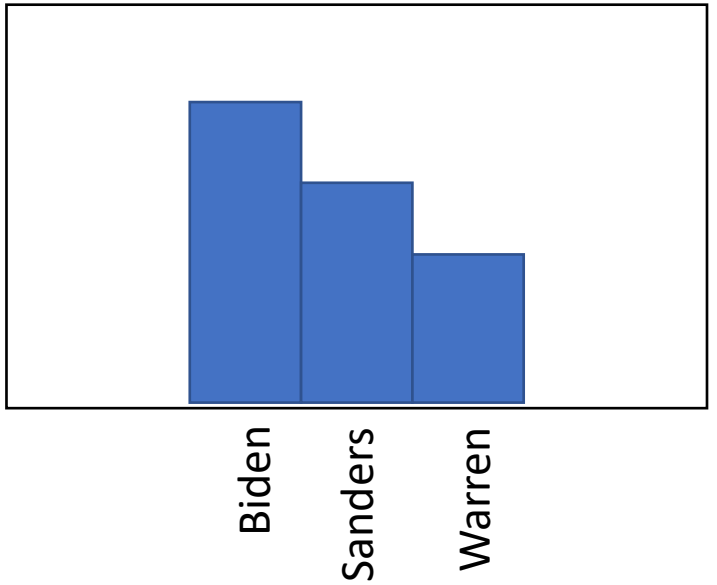
Invariance to small deformation

- 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 *total magnitude* of each orientation bin in each cell: *orientation histograms*



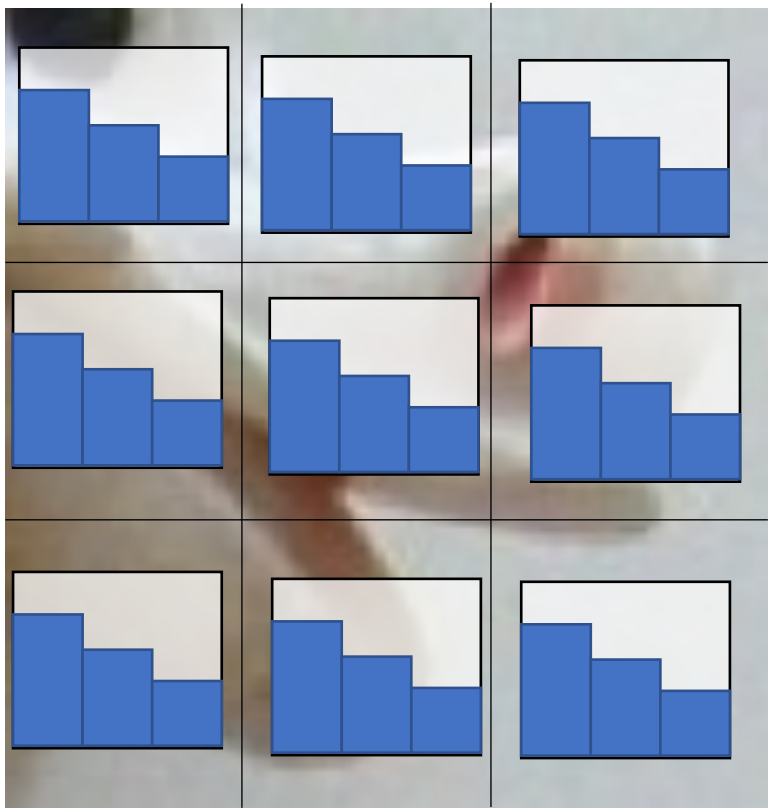
Histograms

- Consider a set of data points
- Each can be one of N choices
  - E.g Votes in the Democratic primary
- A histogram counts *how many votes each choice gets*
  - E.g. Biden: 54, Sanders: 49, Warren: 35…
- Can also be a *weighted histogram,* where each vote has some *weight*
  - Histogram is then *total weight each choice gets*



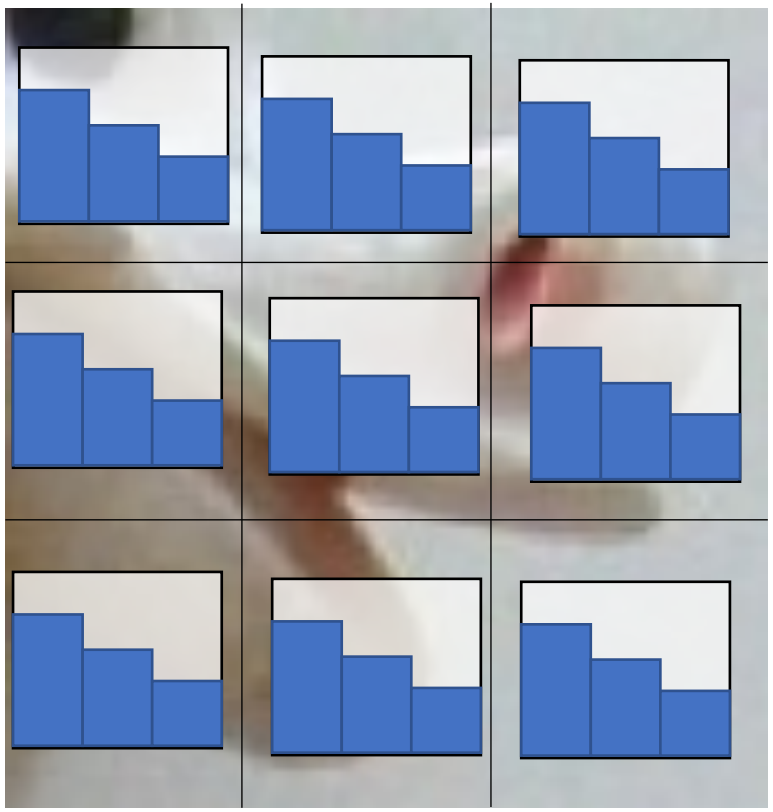
Orientation histograms

- Divide patch into (say) 9 cells
- Divide set of orientations (0-360) into (say) 3 bins
- Create one histogram for each cell
  - Weighted histogram of 3 “choices”
  - Each pixel “votes” into appropriate bin using gradient magnitude as “weight”



Orientation histograms

- Divide patch into (say) 9 cells
- Divide set of orientations (0-360) into (say) 3 bins
- Create one histogram for each cell
  - Weighted histogram of 3 “choices”
  - Each pixel “votes” into appropriate bin using gradient magnitude as “weight”
- Concatenate all histograms into descriptor



Orientation Histograms

- For every pixel p in the patch, let
  - $m(p)$ be the gradient magnitude,
  - $\theta(p)$ be the gradient orientation
  - $x(p), y(p)$ be the location
- Find cell c pixel falls in
- Find orientation bin o that $\theta(p)$ falls in
- Increment o-th total in c-th histogram by $m(p)$

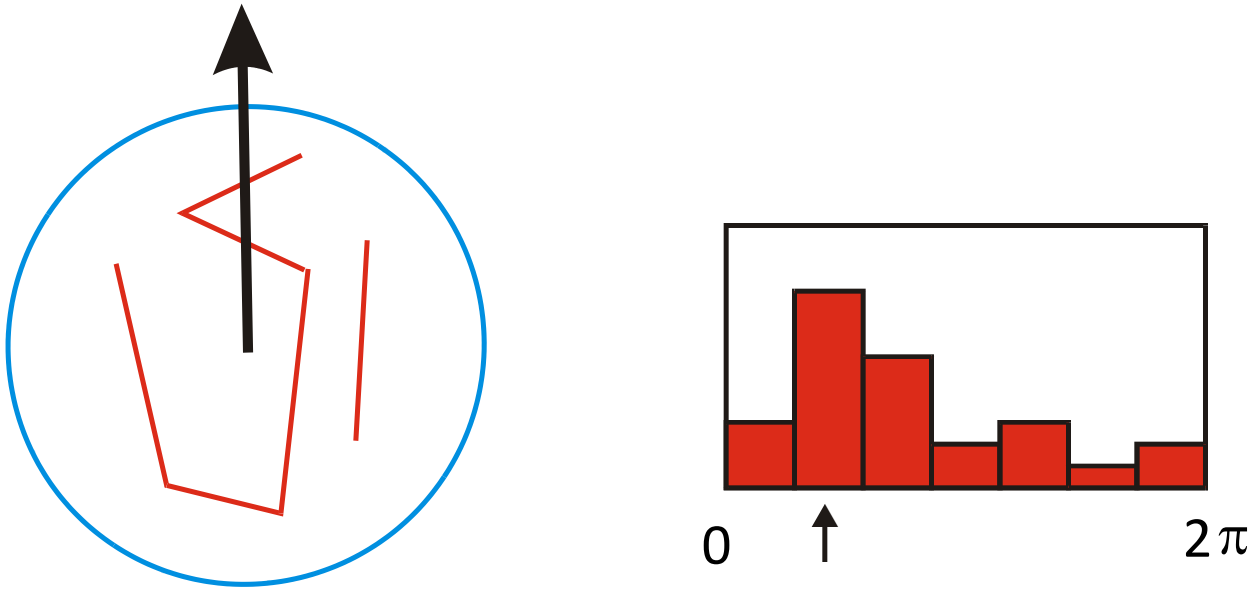
The SIFT Descriptor Pipeline

- Scale and rotation invariance as before:
  - Find dominant orientation and rotate till orientation is along X (say)
  - Use scale output by corner detector to crop patch of appropriate size
- Divide into cells and construct orientation histograms per cell

Rotation Invariance by Orientation Normalization

[Lowe, SIFT, 1999]

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



Tips and tricks 1: Weak edges

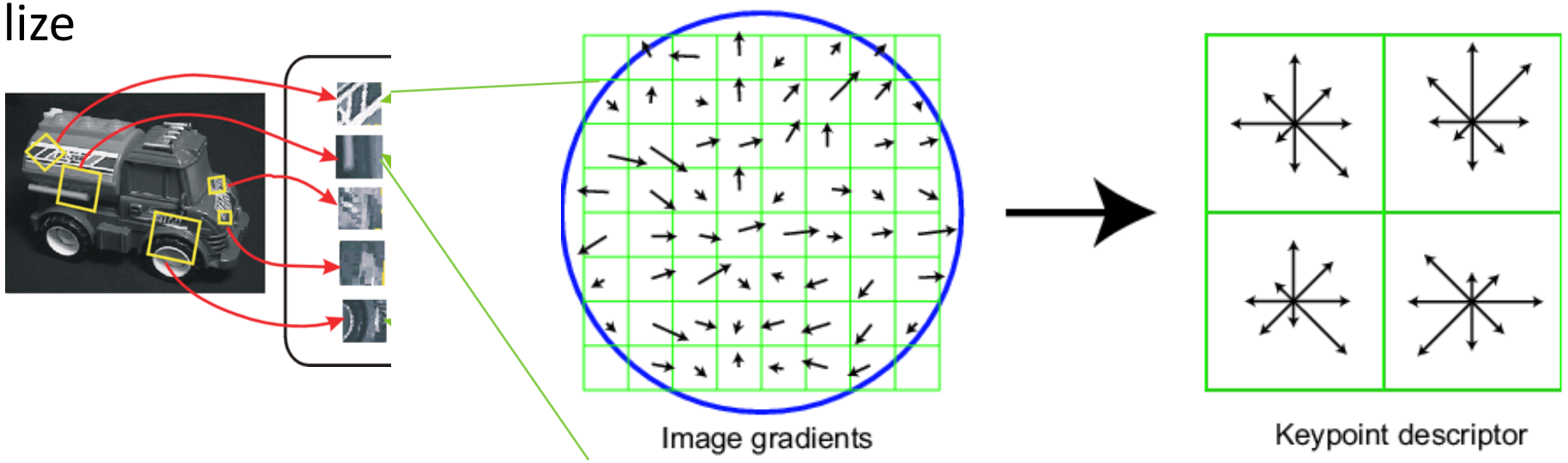
- Lots of weak edge pixels with low gradient magnitude might dominate histogram
- Orientation is unreliable for low gradient magnitude
- Idea: Threshold gradient magnitude: only keep edges of high gradient magnitude

Tips and tricks 2: Soft voting

- Small deformations can cause pixels on edge of cells to switch to new cell
  - If edge pixel, drastically different descriptor
- Idea: soft voting
  - Pixel votes into 4 nearest cells based on distance
  - Similar to bilinear interpolation
- Use same idea for orientation bins

Tips and tricks 3: Reduce effect of illumination

- Gradient magnitudes affected by multiplicative intensity changes
- Normalize descriptor to unit norm
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - after normalization, clamp gradients >0.2
  - renormalize



Detail: Feature detection

- We have talked about Harris corner detector
- Other feature detectors available too
- SIFT implementations usually use *Difference-of-Gaussians filters*
  - Peaks in DoG response

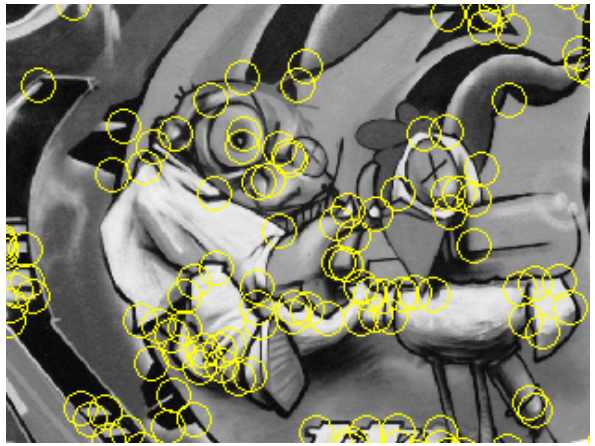
Properties of SIFT

Extraordinarily robust matching technique

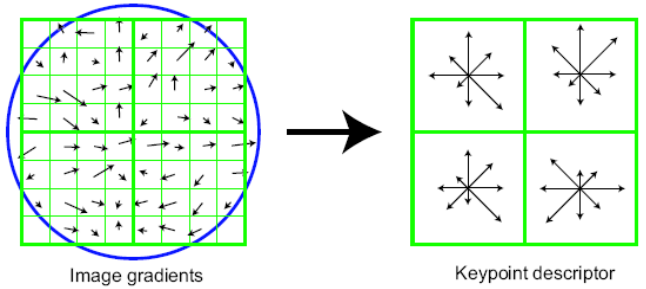
- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available:
  http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT

Summary

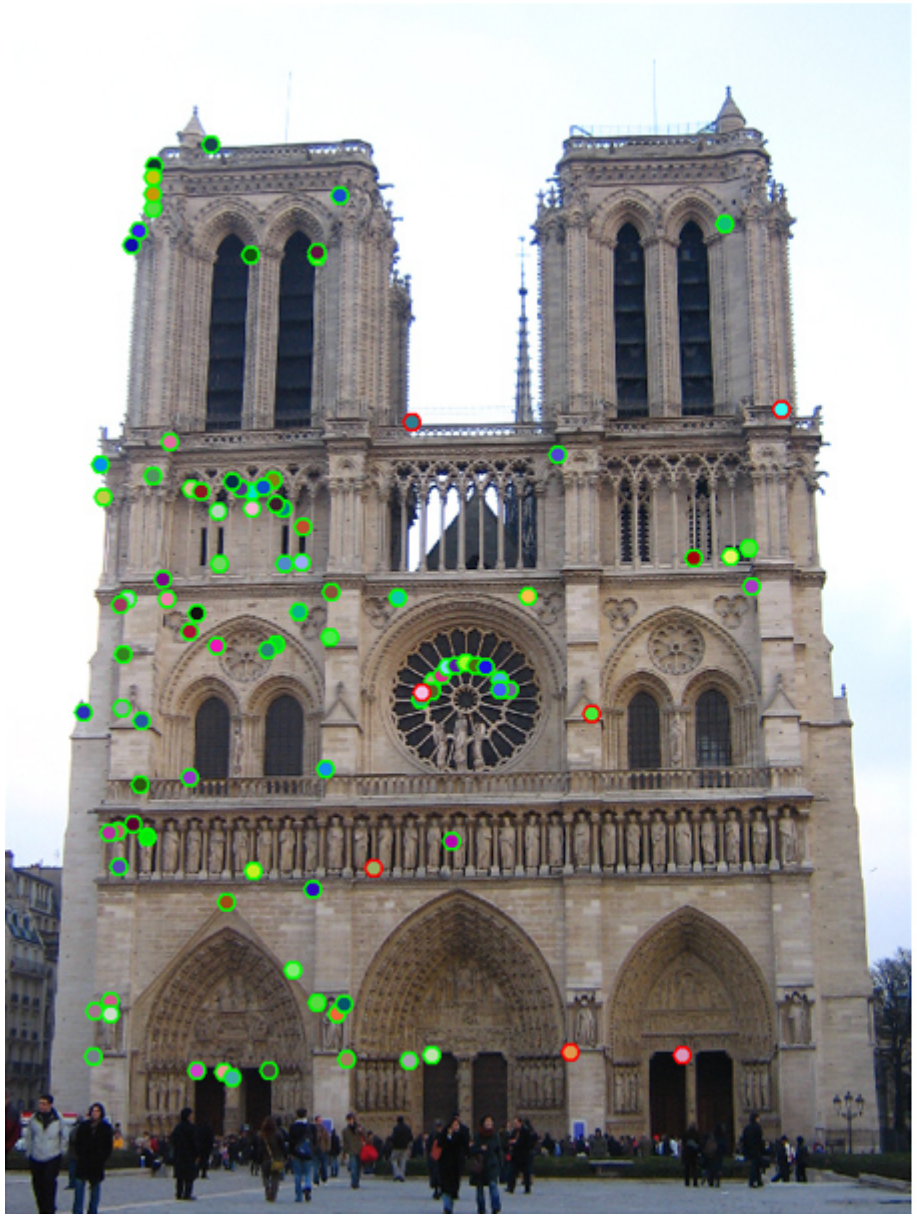
- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Harris, DoG

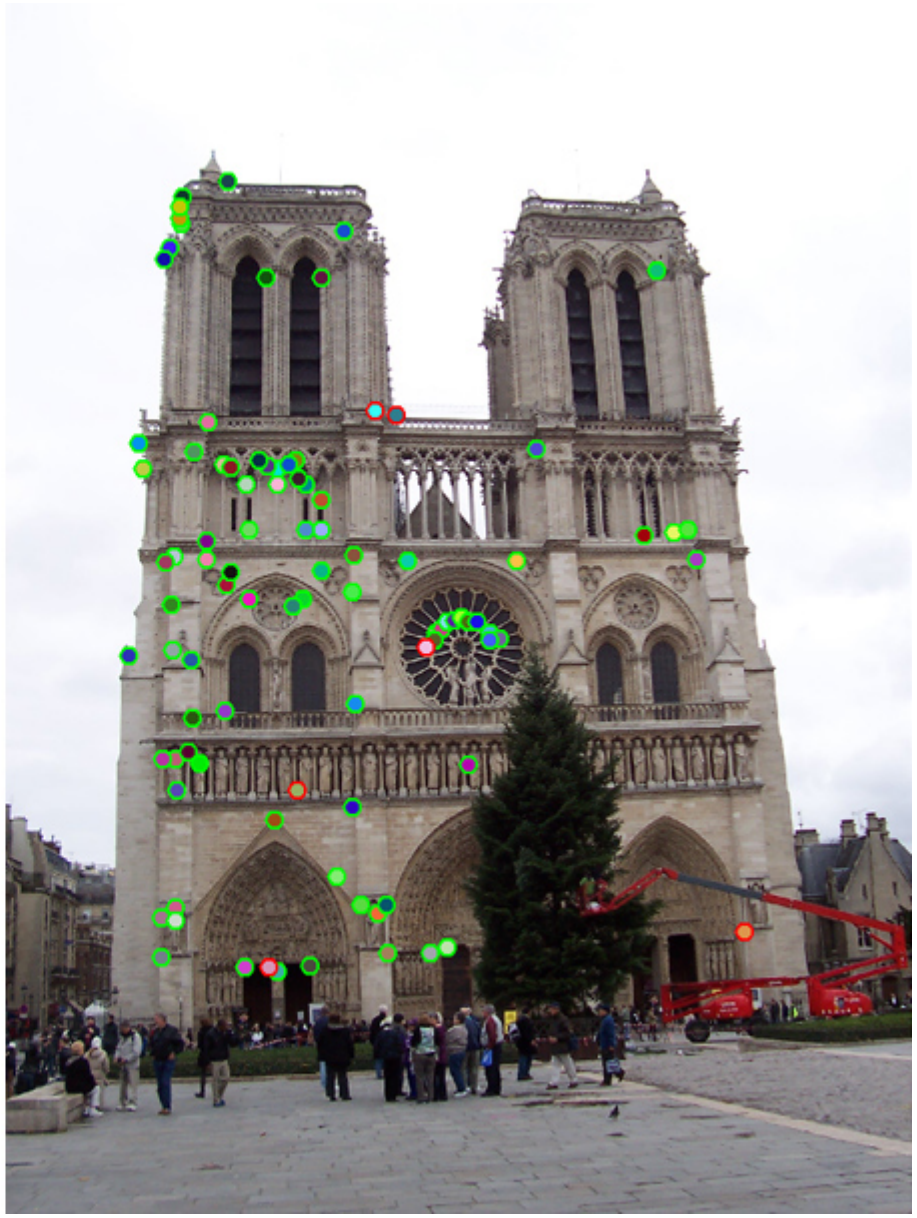


- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT and variants are typically good for stitching and recognition
  - But, need not stick to one



Which features match?





Feature matching

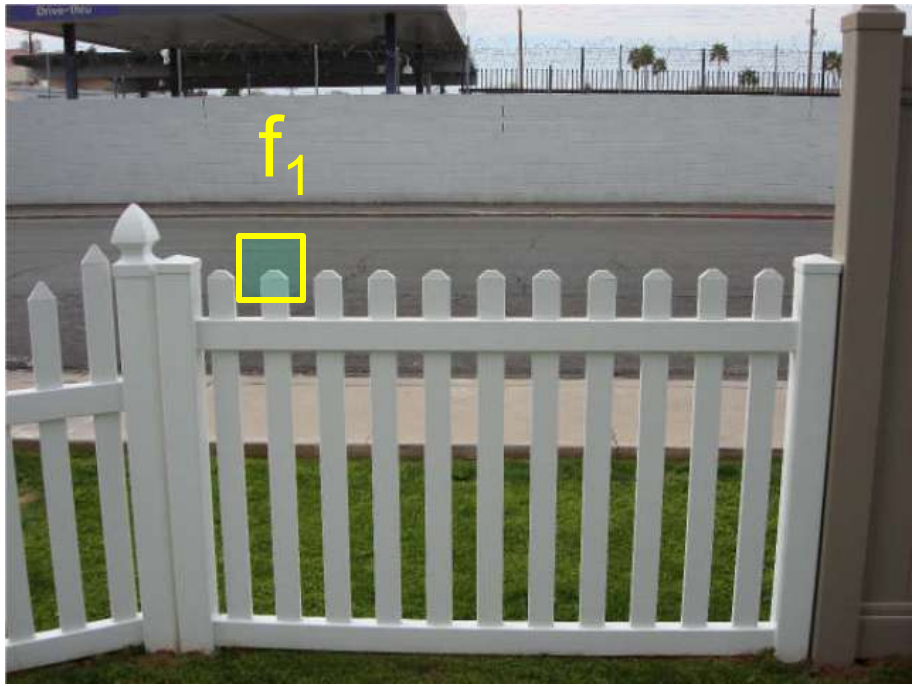
Given a feature in I_1, how to find the best match in I_2?

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

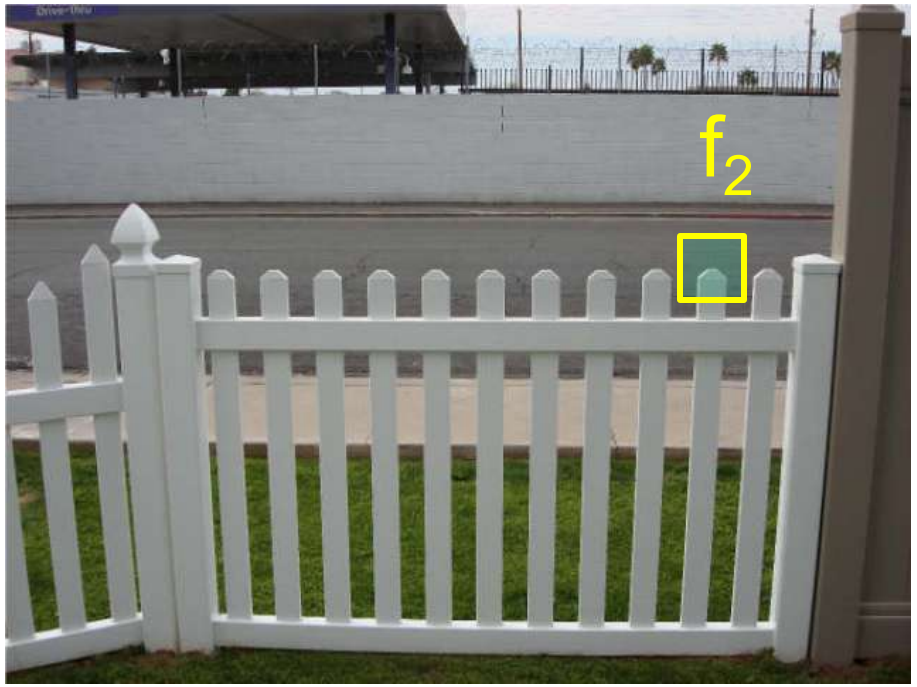
Feature distance

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

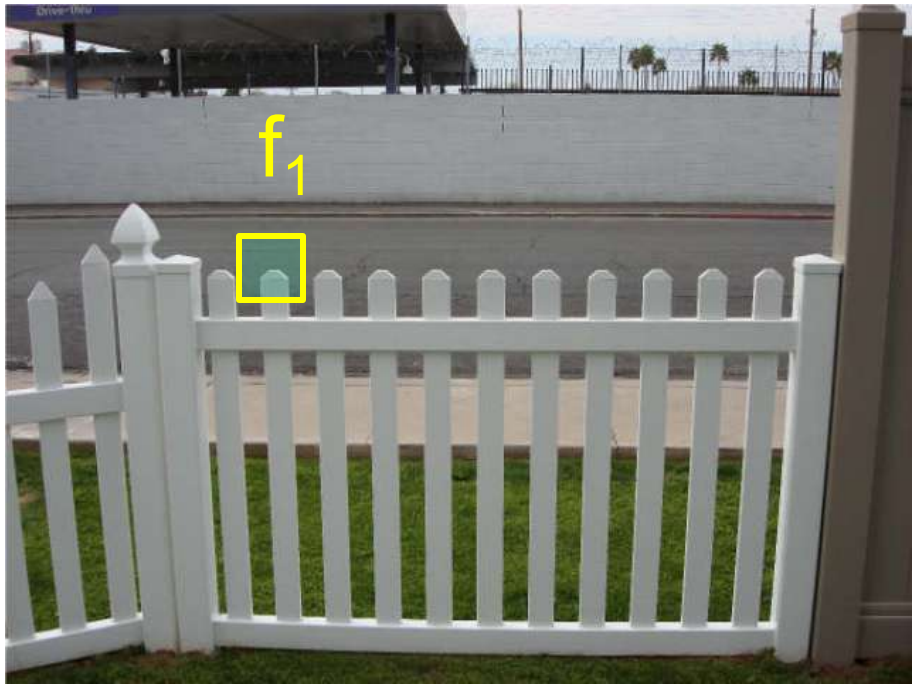


I_2

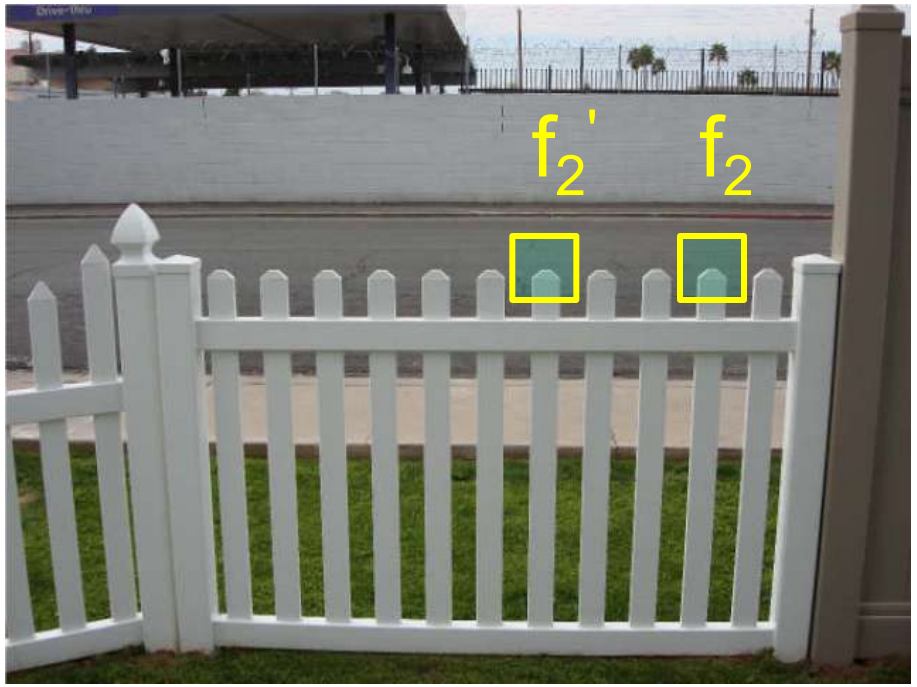
Feature distance

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