Project 2

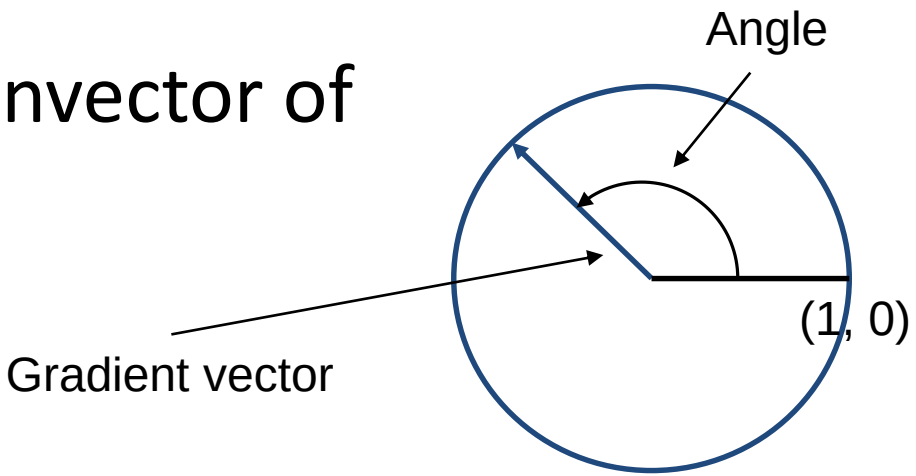
- Python, NumPy, SciPy
  - Tutorial link in writeup
- Main parts
  - Harris corner detection
  - MOPS feature descriptor
  - Simple feature matching

Images + NumPy

- Image coordinates: x, y
- Numpy array
  - Access like a matrix
  - Pixel at coordinate (x, y) is image[y, x]

Keypoint orientation

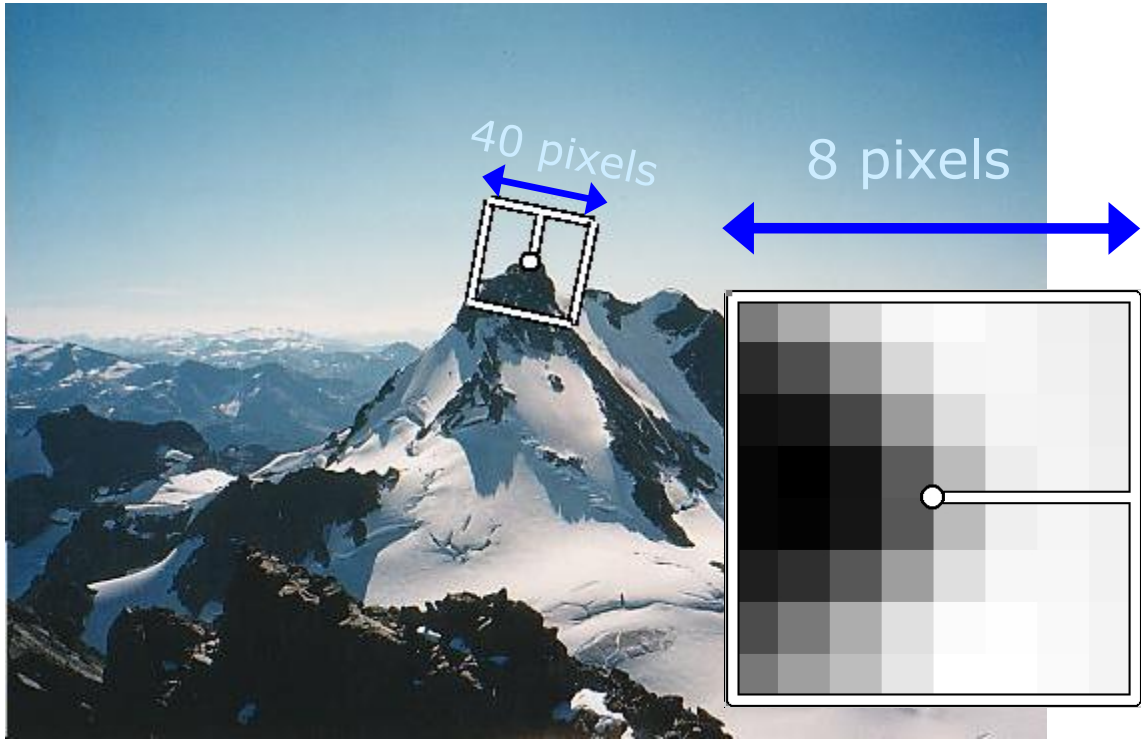
- Harris corner detector
  - Feature orientation angle *approximation*
    - *angle of gradient* computed on blurred image. 0 angle: counter clockwise from the point (1, 0)
  - **Not** given by an eigenvector of the structure tensor



Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Prefilter (because we are subsampling)
- Scale to 1/5 size
- 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
- You **don't** have to implement the multiscale part



Adapted from slide by Matthew Brown

Affine transformation

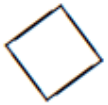
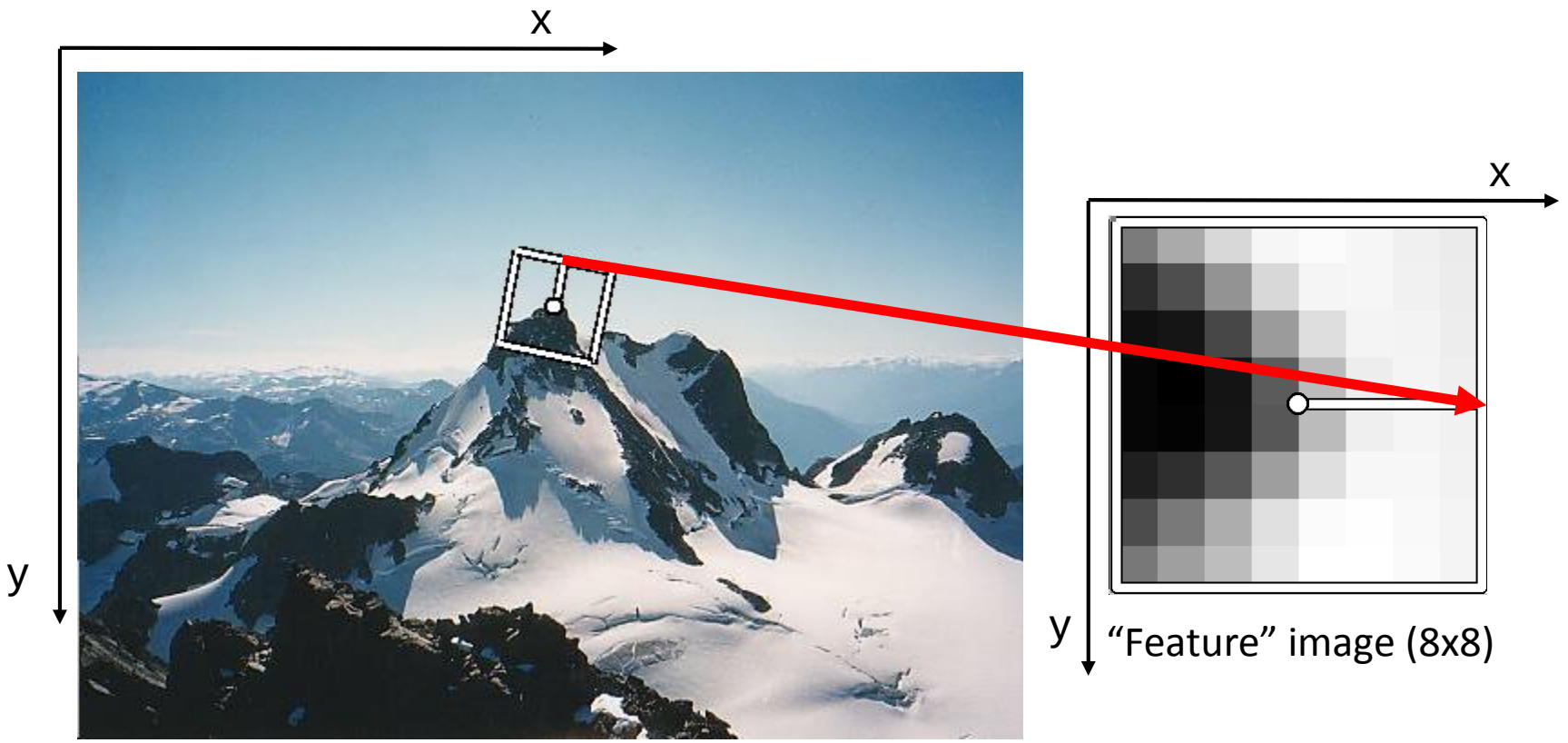
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

Table 2.1 Hierarchy of 2D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The 2×3 matrices are extended with a third $[\mathbf{0}^T\ 1]$ row to form a full 3×3 matrix for homogeneous coordinate transformations.

MOPS descriptor



Source image

•We provide you a routine, **cv2.warpAffine**, that can perform the resampling, transformation and cropping

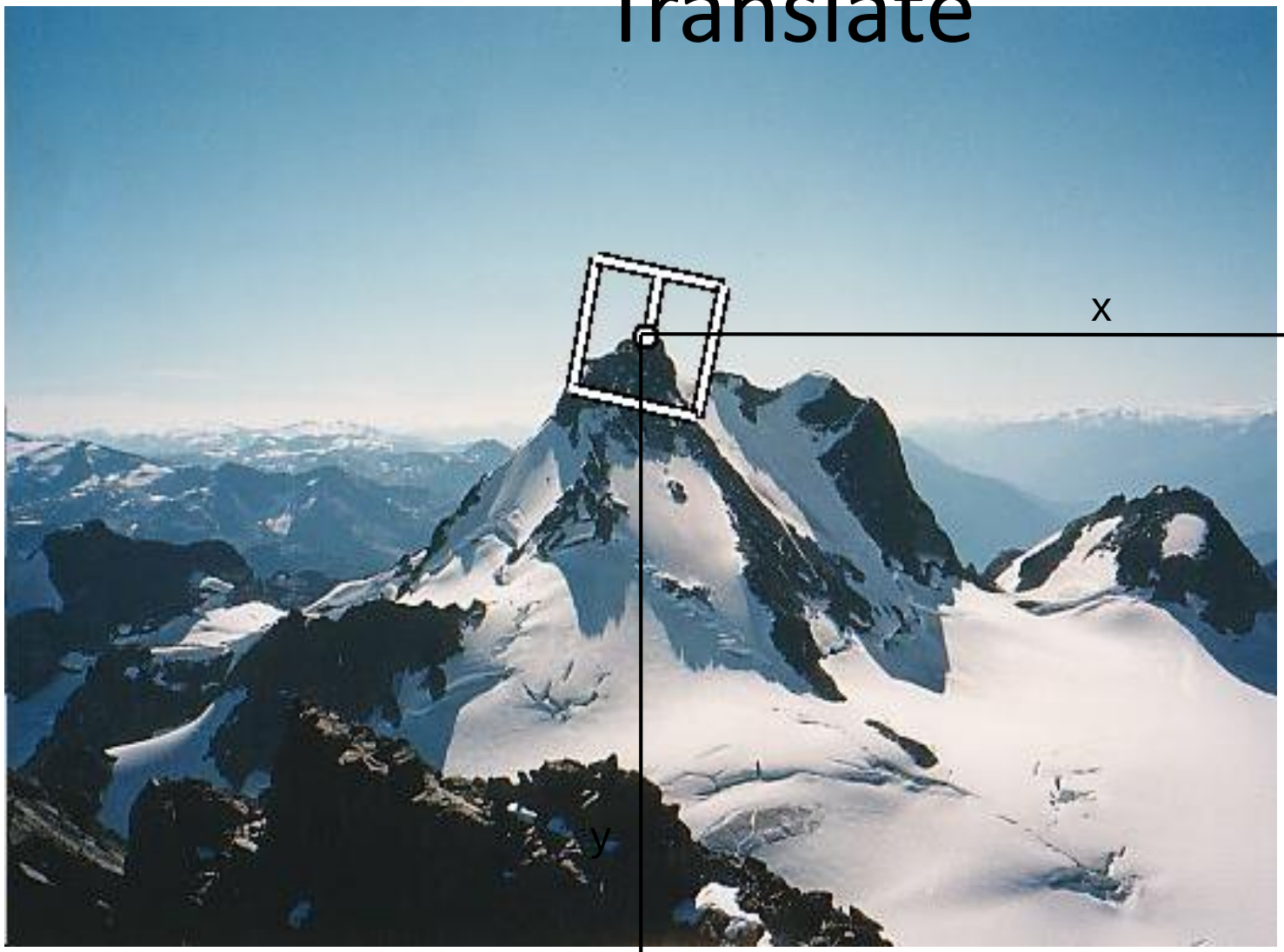
•You have to pass a **forward** warping affine transformation matrix (2x3), multiplied from the **left,** the coordinates are represented as a **column vector**

MOPS descriptor

- You can combine transformations together to get the final transformation
- Pass this transformation matrix to *cv2.warpAffine*

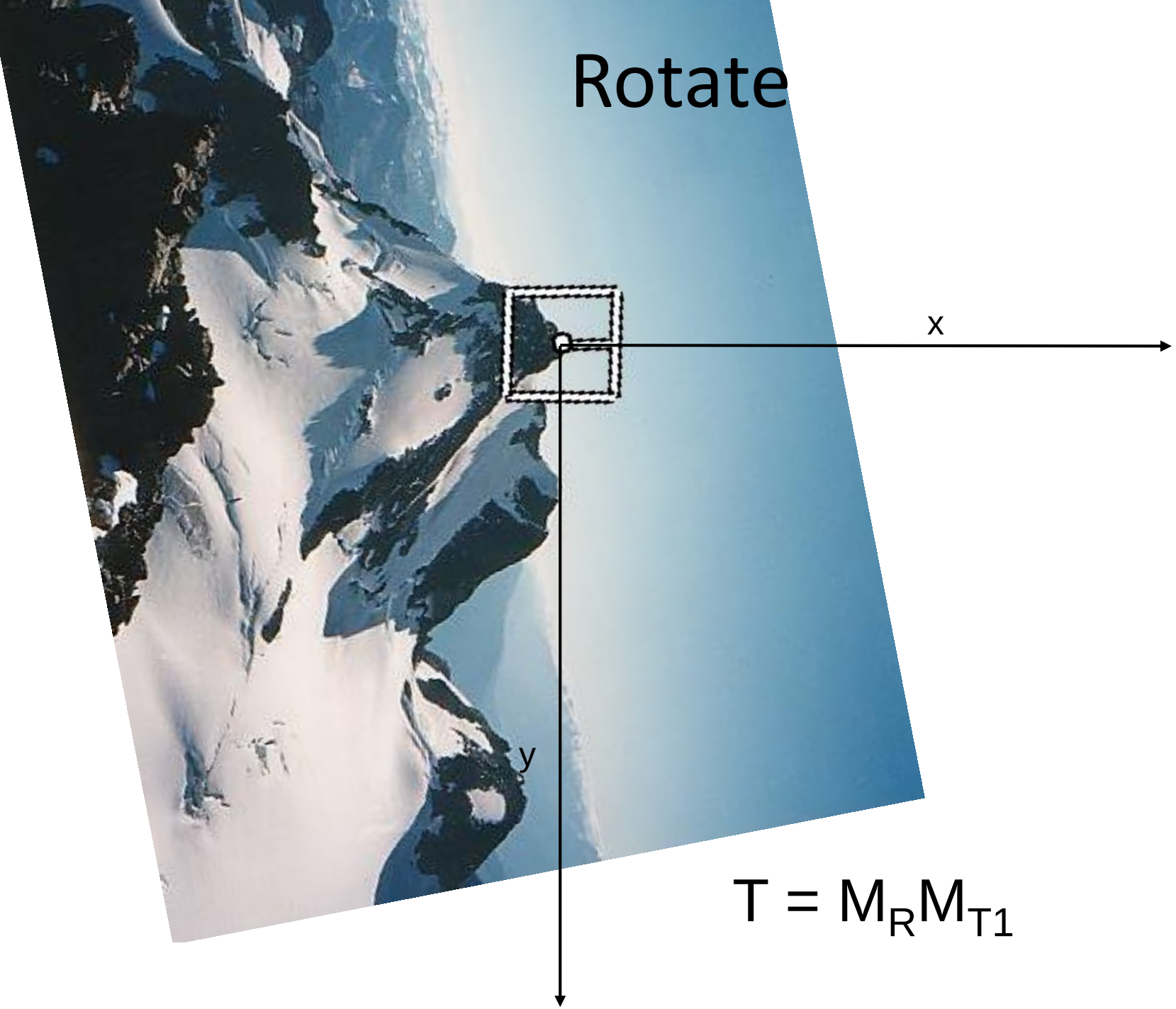
T = ?

Translate



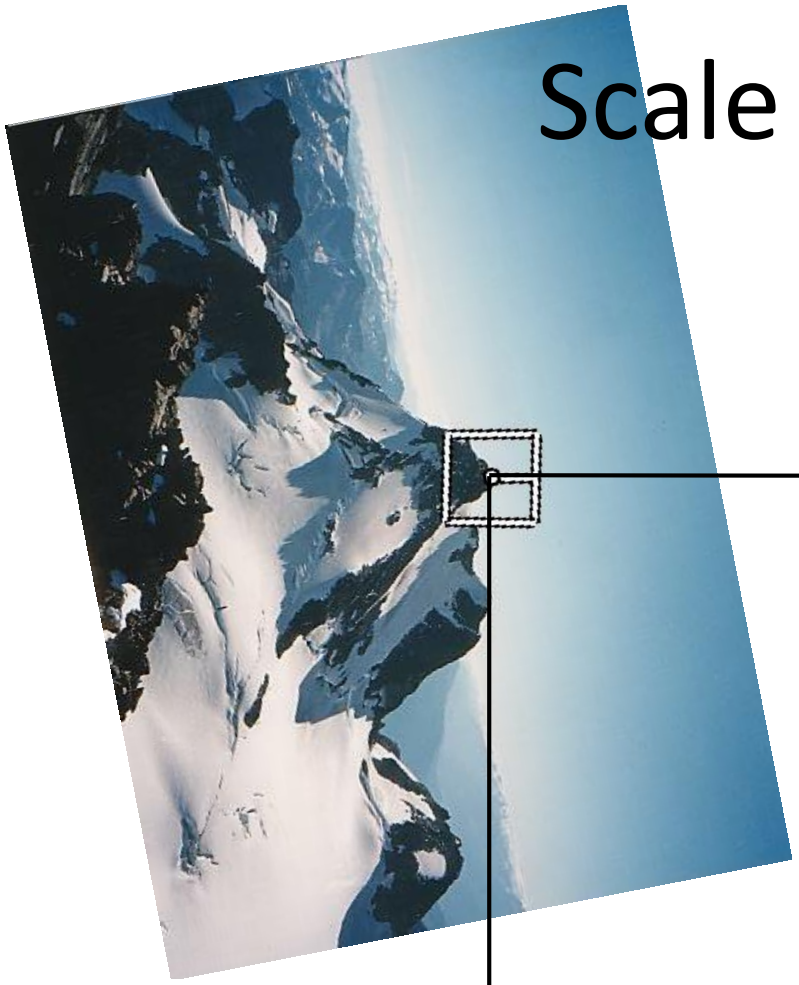
$T = M_{T1}$

Rotate



$$T = M_R M_{T1}$$

Scale

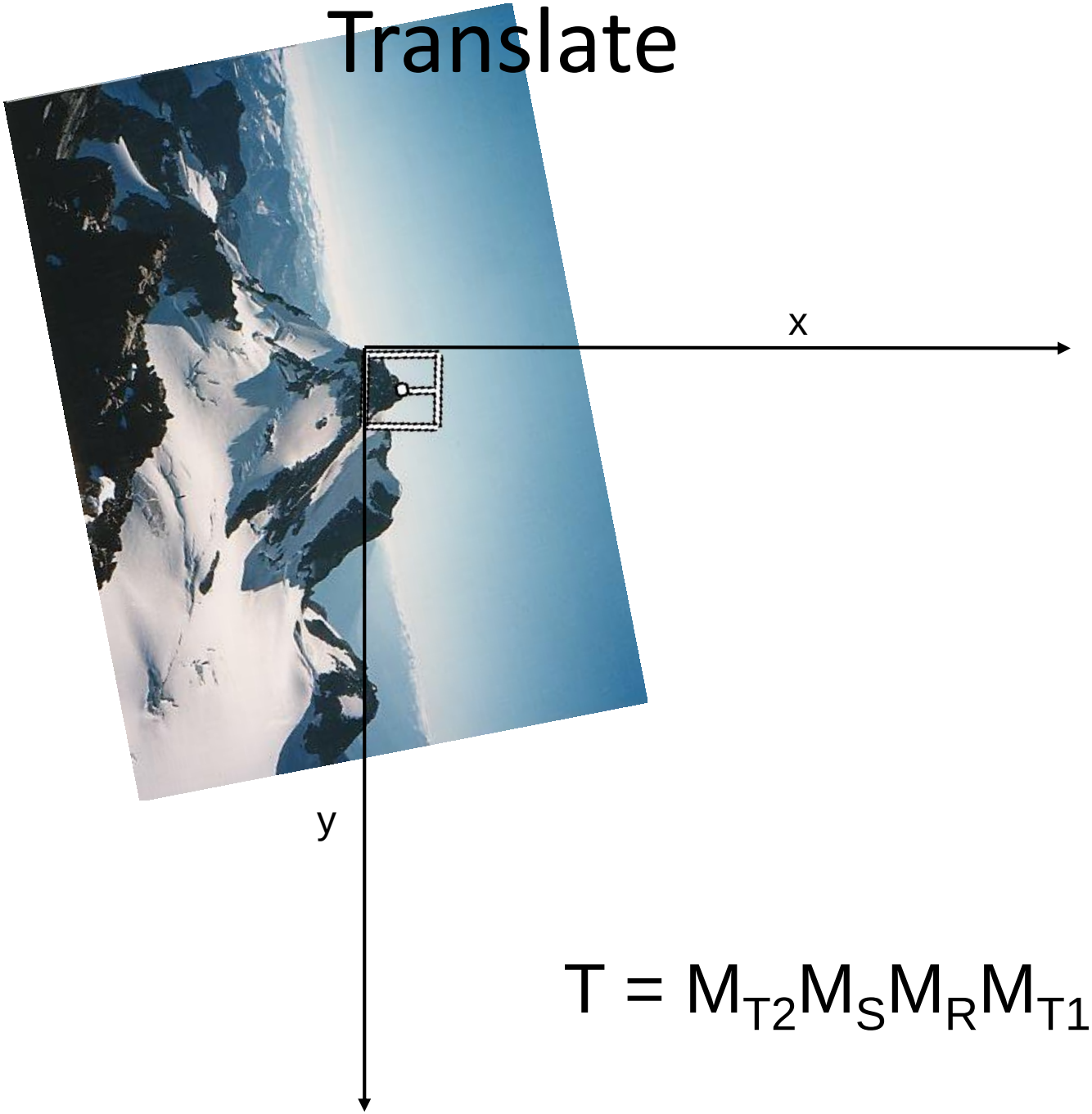


x

y

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

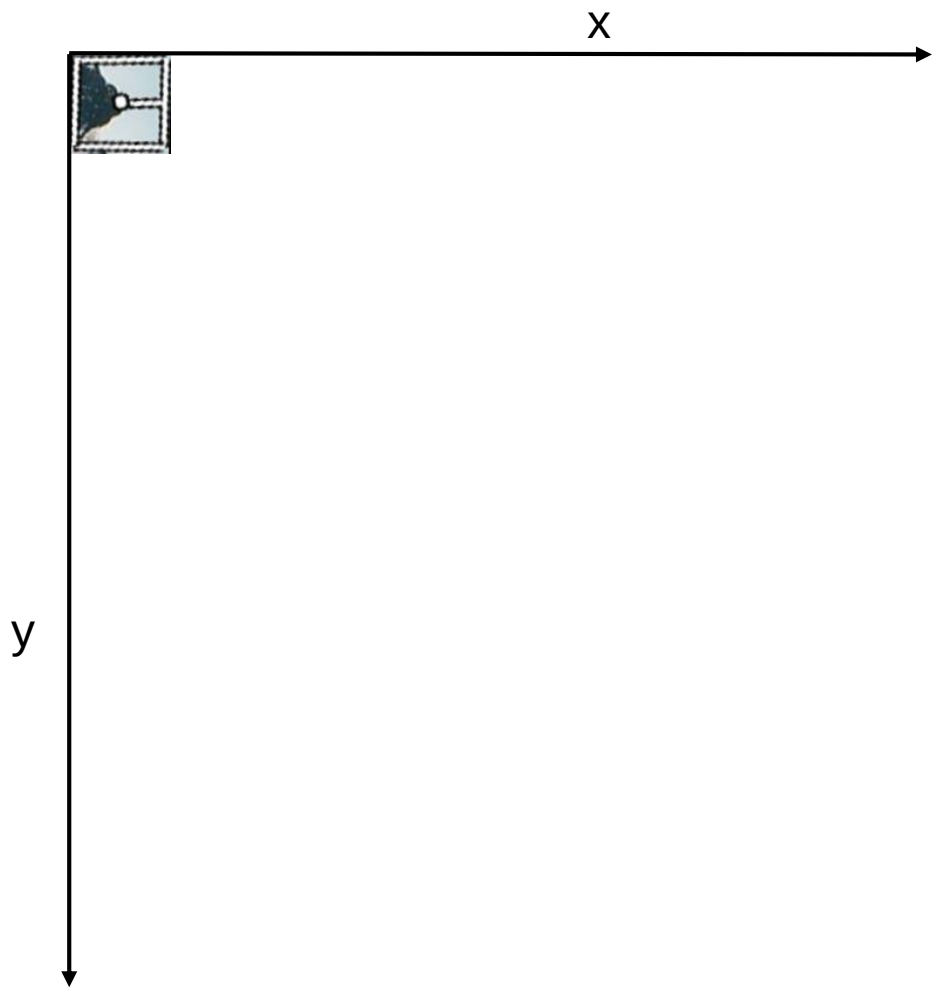
Translate



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

Crop

- *cv2.warpAffine* also takes care of the cropping



Demo