دانشگاه
خواجه نصیرالدین طوسی
K. N. Toosi University
of Technology

Computer Vision

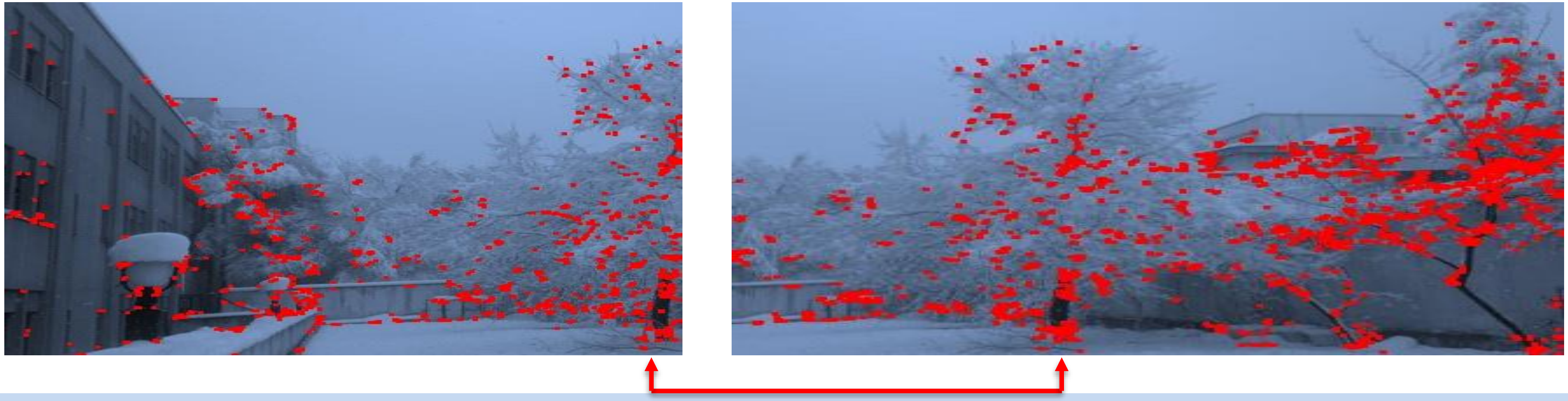
Lecture 7: Image Transformation

Dr. Esmaeil Najafi

MSc. Javad Khoramdel

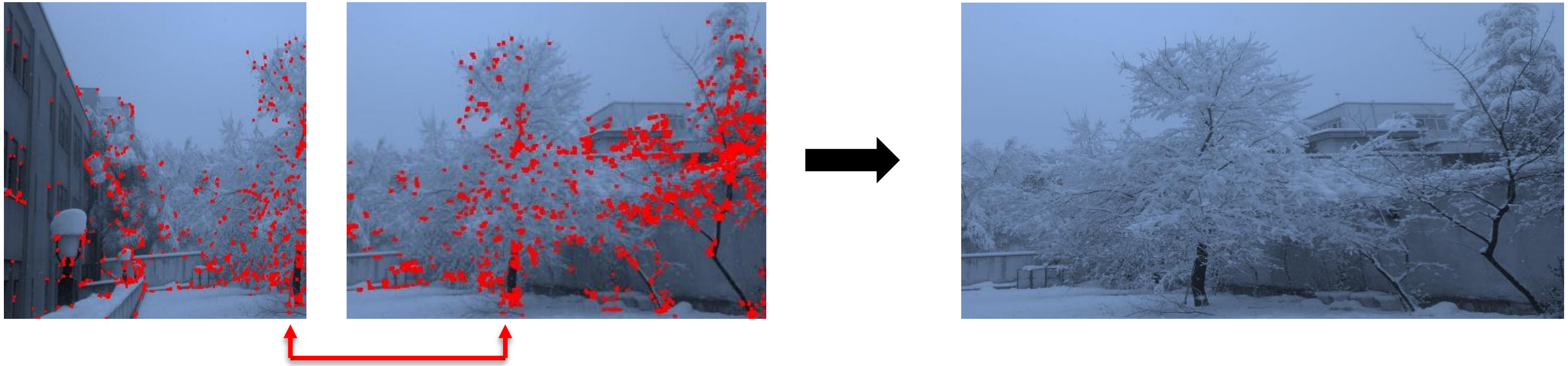
How to panoroma

- So far
  - We found “**unique patches**” in each images
    - ✓ Used Harris corner detector to find the corner points.
  - “ **Matched”** each patch from the first image to the corresponding patch to the second image based on their similarities
    - ✓ Compared each patch from the first image to all the patches in the second image to find the similar patch.



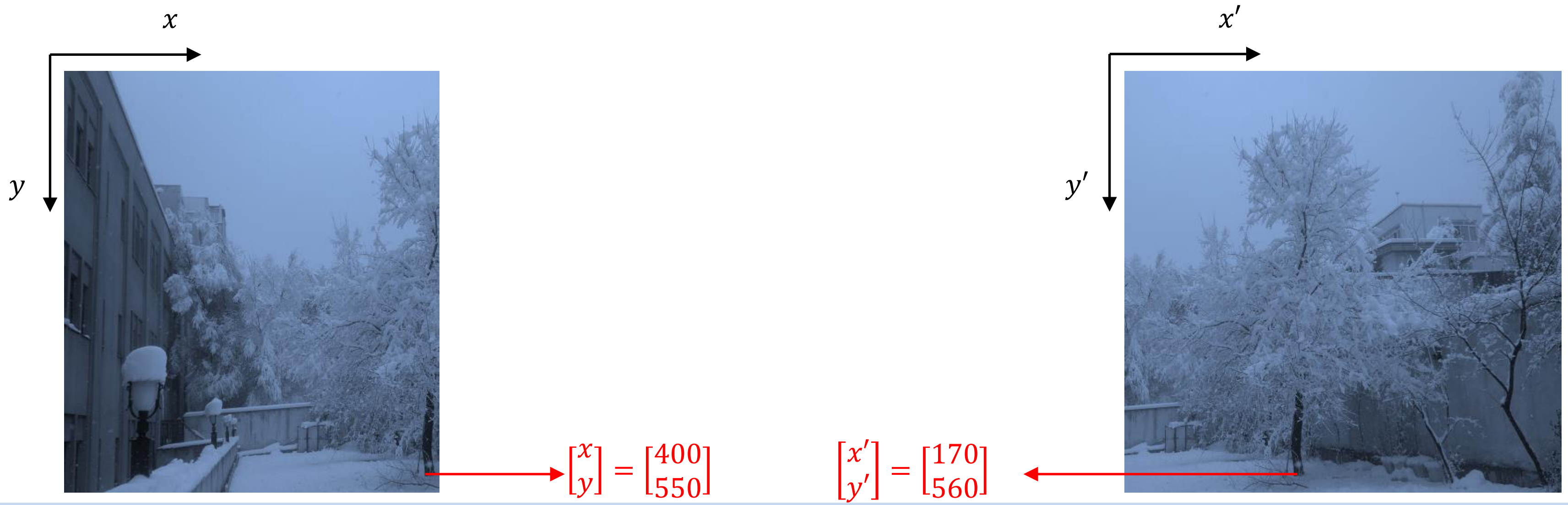
How to panoroma

- Now
  - “Transform” the images to form the panorama image.



Why do we need transformation?

- Each image has its own frame.
- The coordinates of similar points are represented in different frames.
- If we want to merge the images, all points should be presented in a single frame.
  - In other words, we need to find the coordinates of the points in the second image in $x-y$ frame



$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 400 \\ 550 \end{bmatrix}$

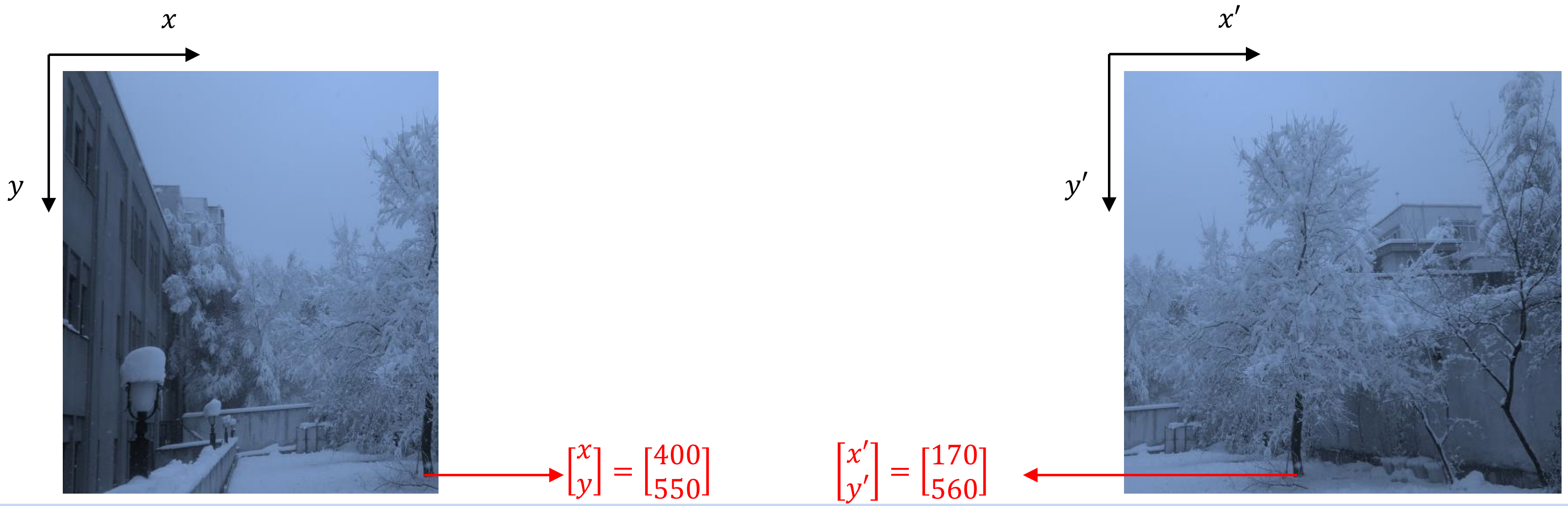
$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 170 \\ 560 \end{bmatrix}$

Let's take a detour!

- We need to learn and study basics of image transformation

What do mean by “transformation”?

- We are looking for a matrix “T” which:
  - $\begin{bmatrix} x \\ y \end{bmatrix} = T \times \begin{bmatrix} x' \\ y' \end{bmatrix}$
  - Matrix “T” is a transformation matrix which transforms points from $x' - y'$ frame to $x - y$ frame.

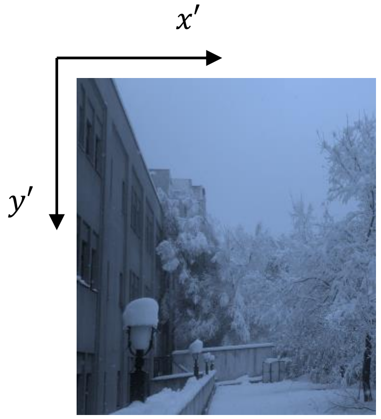


$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 400 \\ 550 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 170 \\ 560 \end{bmatrix}$$

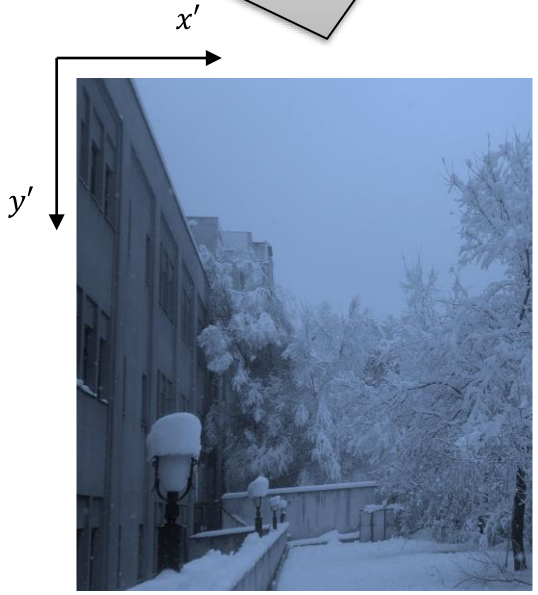
Scaling

- A scaling transformation alters size of objects in an image
  - $T_{scaling} = \begin{bmatrix} S & 0 \\ 0 & S \end{bmatrix}$
    - ✓ "S" is the scaling factor.

Scaling preserves orientations, parallelism, and angles.



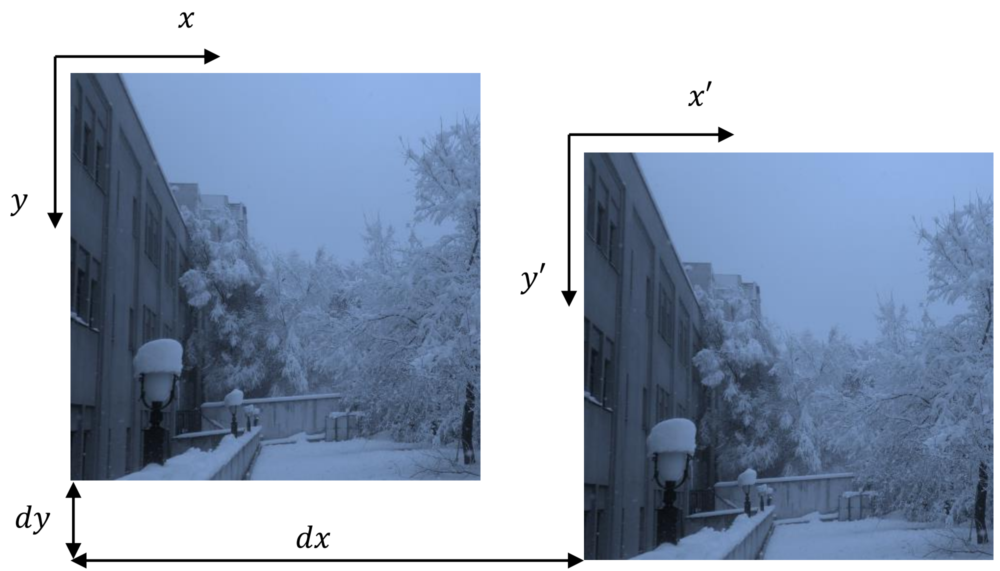
$S > 1$



$S < 1$

Translation is harder…

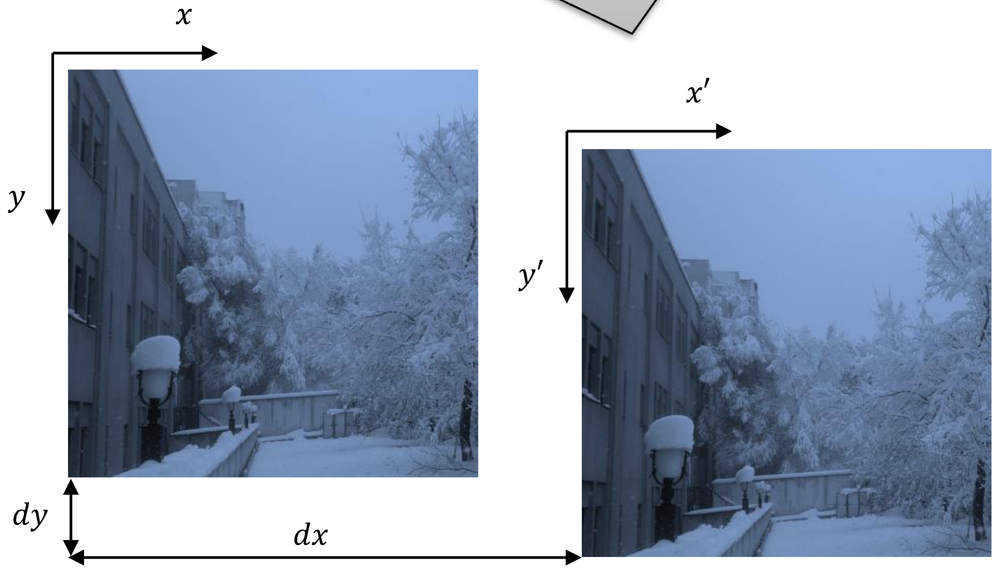
- Imagine that each point in the first image is shifted "dx" in the x-direction and "dy" in the y direction.
  - i.e. : $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' + dx \\ y' + dy \end{bmatrix}$
- Can you suggest a matrix T which satisfy the following conditions?
  - $\mathrm{T} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
  - And: $\begin{bmatrix} x \\ y \end{bmatrix} = T \times \begin{bmatrix} x' \\ y' \end{bmatrix}$



Translation: add another row

- Make the augmented vector $\bar{X}'$:
  - $\bar{X}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$
  - $\bar{X}'$ is just X' with an added 1
- Translation is easy now!
  - $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$
  - $T_{translation} = [I,\ t]$
    - ✓ I is identity matrix
    - ✓ $\bar{t}$ is translation vector (or offset)

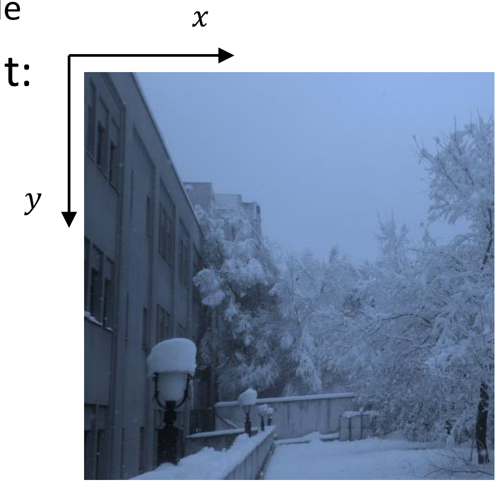
Translation preserves distances, orientations, parallelism, and angles.

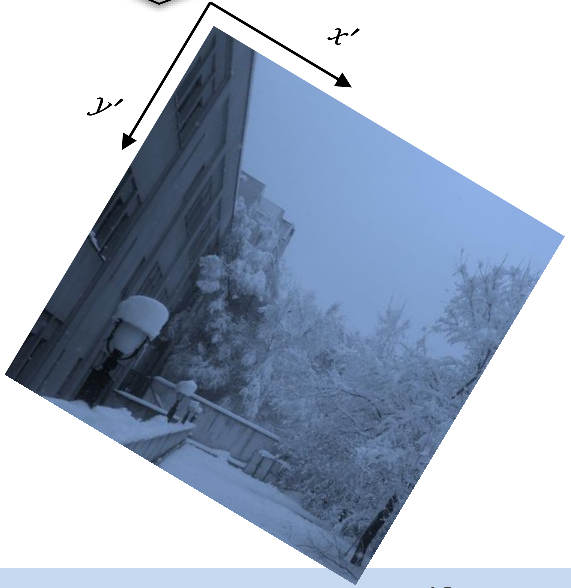


Euclidean: rotation + translation

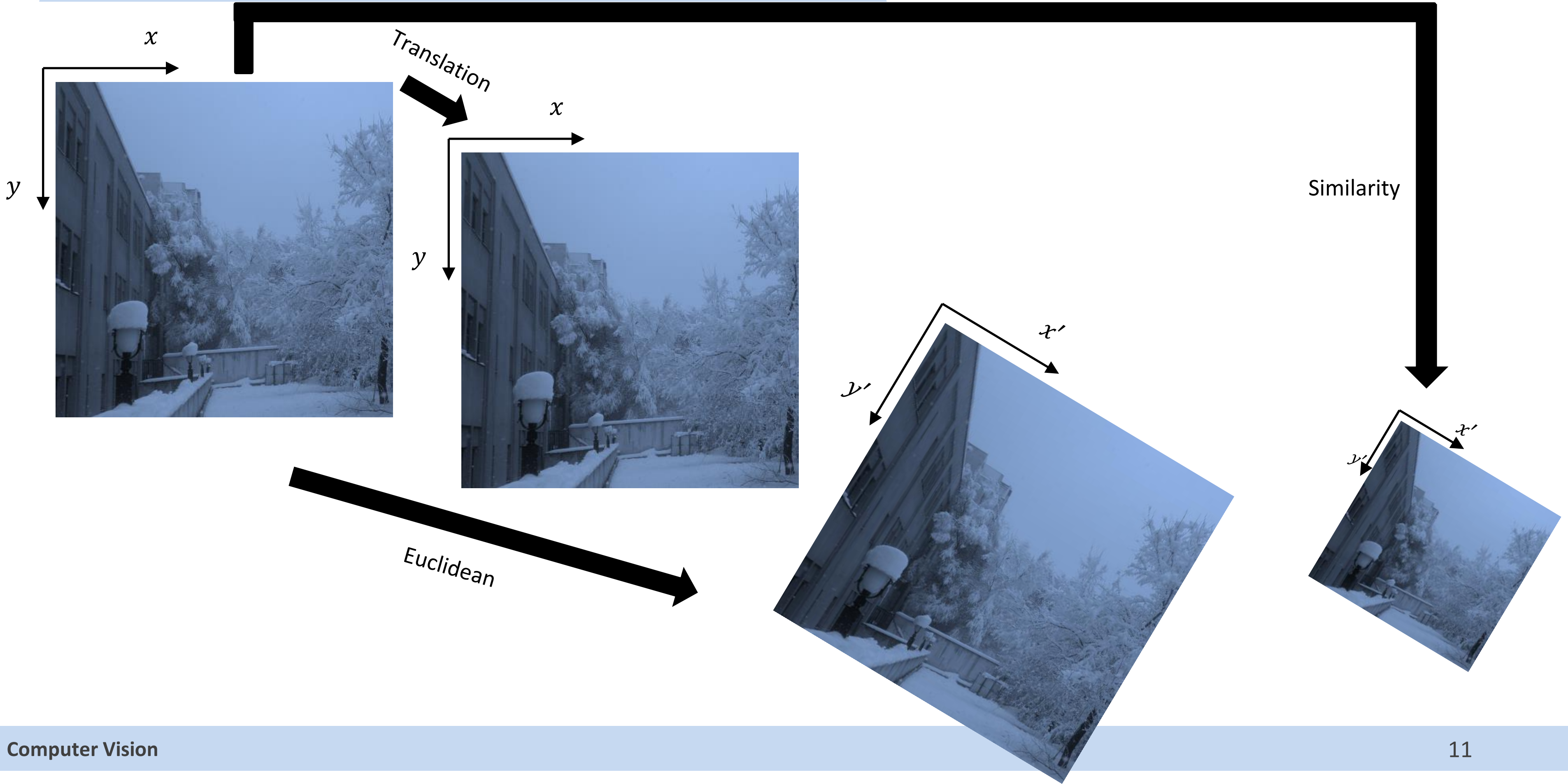
- In Euclidean transformation, we want to translate and rotate at same time.
- For rotation, we can use the rotation matrix:
  - $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
    - ✓ Where θ shows the rotation angle
- For translation, we can use vector t:
  - $t = \begin{bmatrix} dx \\ dy \end{bmatrix}$
- Transformation matrix:
  - $T_{Euclidean} = [R, t]$
  - Special case (only rotation)
    - ✓ $T_{rotation} = [R, 0_{2\times 1}]$

Euclidean preserves distances, parallelism, and angles.





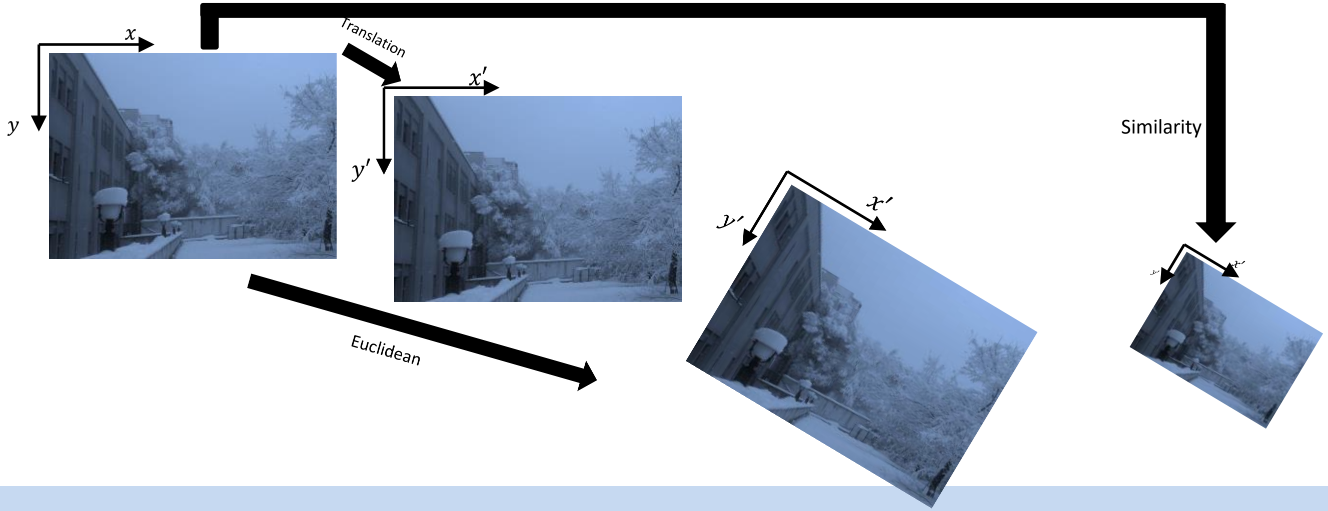
Similarity: scale + rotation + translation



Similarity: scale + rotation + translation

- Similarity transformation matrix:
  - $T_{similarity} = [sR, t]$
    - ✓ $T_{similarity} = \begin{bmatrix} s.\cos\theta & -s.\sin\theta & dx \\ s.\sin\theta & s.\cos\theta & dy \end{bmatrix}$

Similarity preserves parallelism and angles.

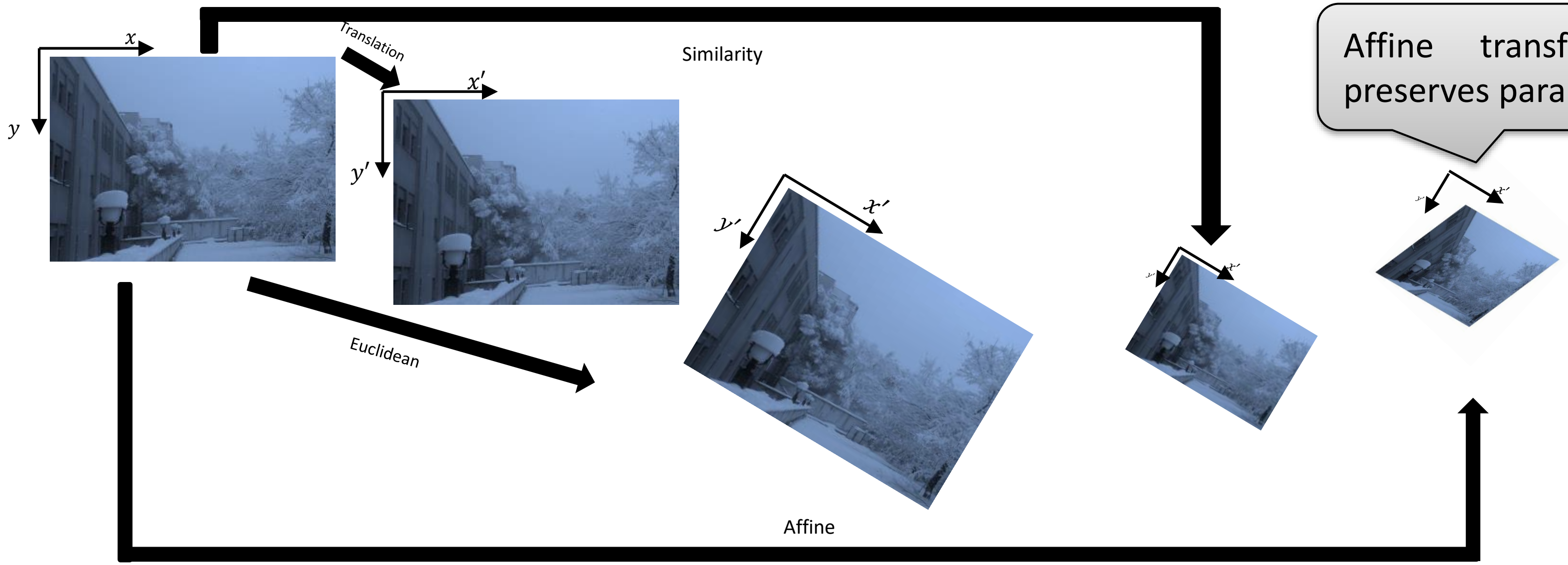


Affine: scale, rotate, translate, shear

- Affine transformation matrix is a general 2×3 matrix:
  - $T_{affine} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$

All this transformations are a special case of affine transformation.

Affine transform only preserves parallelism.



Combinations are still affine

- Say you want to translate, then rotate, then translate back, then scale.
  - $X = T_{scale} T_{translation_2} T_{rotation} T_{translation_1} X' = T . X'$
  - $T = \left(T_{scale} T_{translation_2} T_{rotation} T_{translation_1}\right)$
    - ✓ T is still affine transformation.

- Wait, but these are all 2x3, how to we multiply them together?

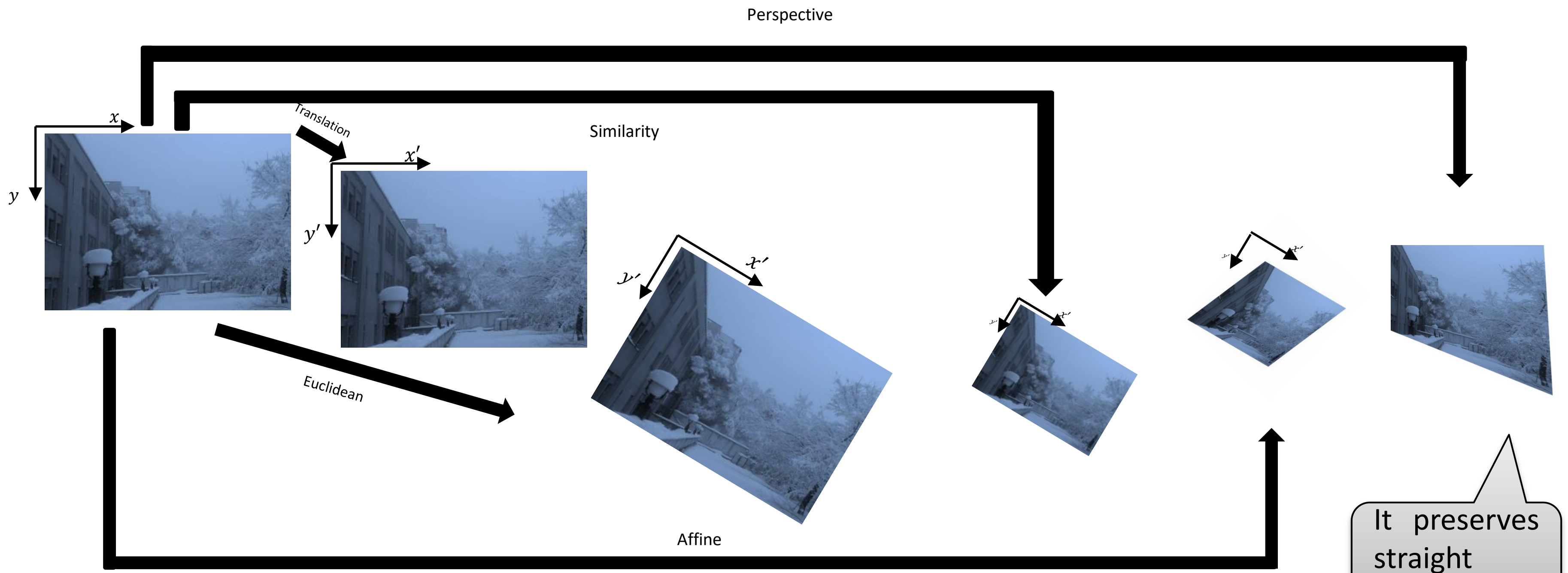
Added row to transforms

- Scaling:
  - $$\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}s&0&0\\0&s&0\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix}$$
    - ✓ $$T_{scaling}=\begin{bmatrix}s&0&0\\0&s&0\\0&0&1\end{bmatrix}$$
- Translation:
  - $$\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}1&0&dx\\0&1&dy\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix}$$
    - ✓ $$T_{translation}=\begin{bmatrix}1&0&dx\\0&1&dy\\0&0&1\end{bmatrix}$$
- Rotation:
  - $$\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix}$$
    - ✓ $$T_{rotation}=\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}$$

Now, we can multiply arbitrary number of transformations, and the result would be an affine transformation.

- Euclidean:
  - $$\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}\cos\theta&-\sin\theta&dx\\\sin\theta&\cos\theta&dy\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix}$$
    - ✓ $$T_{Euclidean}=\begin{bmatrix}\cos\theta&-\sin\theta&dx\\\sin\theta&\cos\theta&dy\\0&0&1\end{bmatrix}$$
- Similarity:
  - $$\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}s.\cos\theta&-s.\sin\theta&dx\\s.\sin\theta&s.\cos\theta&dy\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix}$$
    - ✓ $$T_{similarity}=\begin{bmatrix}s.\cos\theta&-s.\sin\theta&dx\\s.\sin\theta&s.\cos\theta&dy\\0&0&1\end{bmatrix}$$
- Affine:
  - $$\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}a&b&c\\d&e&f\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix}$$
    - ✓ $$T_{affine}=\begin{bmatrix}a&b&c\\d&e&f\\0&0&1\end{bmatrix}$$

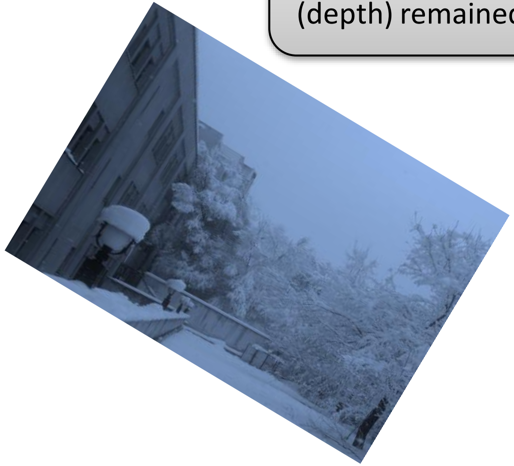
Projective transform



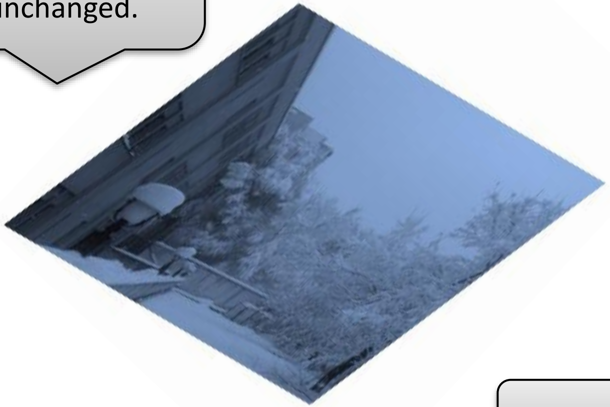
It preserves straight lines

Something is different

In these two transformations, it is like the distance to your eye (depth) remained unchanged.

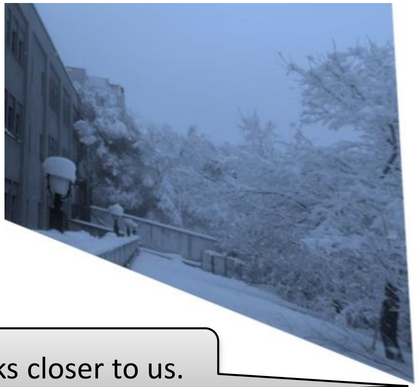


Rotation



Affine (rotation with shear)

This point looks far from us.



This point looks closer to us.

Perspective

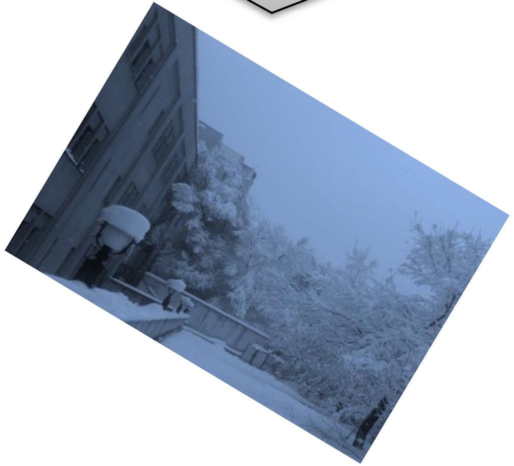
It feels like there is a change in the depth. But, how?

Something is different

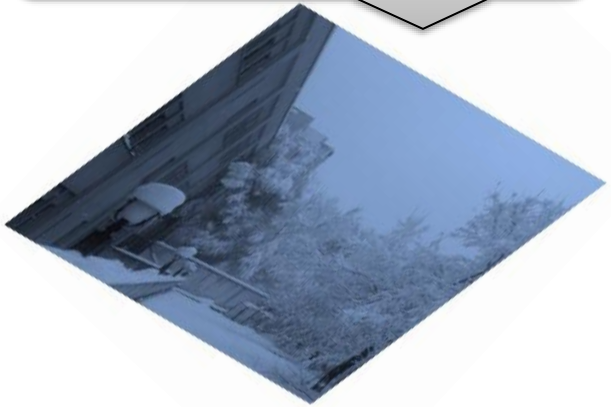
Rotation has been about the "**z**" axis (perpendicular to the image plane)

Rotation has been about the "**z**" axis (perpendicular to the image plane), and we have some kind of shear

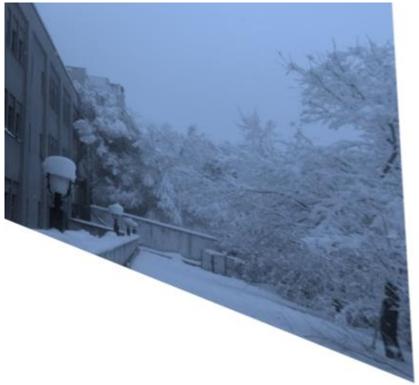
It seems we had some rotation around the "**y**" axis (axis inside the image plane).



Rotation



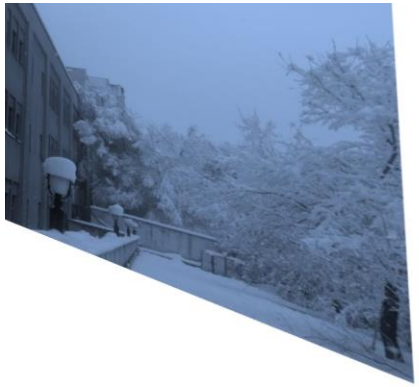
Affine (rotation with shear)



Perspective

Homography: A more generalized transformation

- Rotation about the "y" axis:
  - $R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$
- Rotation about the "x" axis:
  - $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$
- These transformations can not be represented by an affine transformation, where the last row is forced to be $[0, 0, 1]$:
  - $T_{affine} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$
- Homography:
  - $T_{homography} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$


Perspective

But, wait…

- Imagine that:
  - $T_{homography} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- What will happen to a point like $p = [1, 1]^T$ under this transformation?

But, wait…

- Imagine that:
  - $T_{homography} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- What will happen to a point like $p = [2\,,3]^T$ under this transformation?
  - First, we should make the augmented vector (added 1):
    - ✓ $\bar{p} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$
  - Then:
    - ✓ $\tilde{p}_{new} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 6 \end{bmatrix}$

We can not include the information of the "z" axis directly in a 2D image. We need to fix this.

Homogenous coordinates

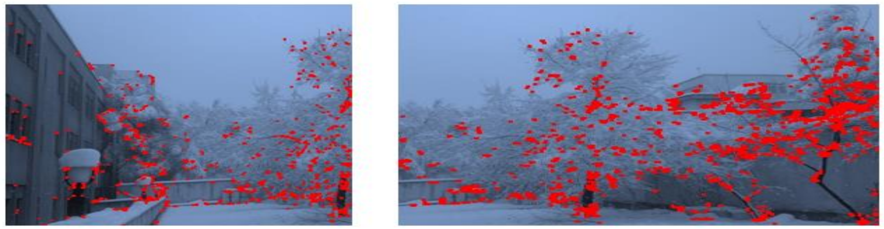
- Imagine that:
  - $T_{homography} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- What will happen to a point like $p = [2\,,3]^T$ under this transformation?
  - First, we should make the augmented vector (added 1):
    - ✓ $\bar{p} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$
  - Then:
    - ✓ $\tilde{p}_{new} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 6 \end{bmatrix} \rightarrow \bar{\boldsymbol{p}}_{\boldsymbol{new}} = \begin{bmatrix} \frac{\mathbf{2}}{\mathbf{6}} \\ \frac{\mathbf{3}}{\mathbf{6}} \\ \mathbf{1} \end{bmatrix}$

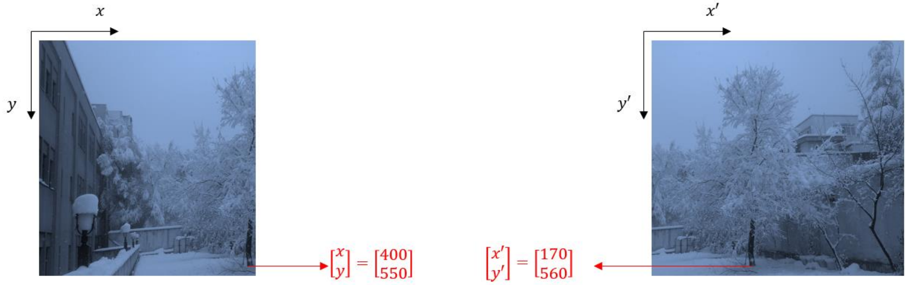
The actual $x - y$ components of the point is obtained by normalizing the point in homogenous coordinates with its 3rd element.

$\tilde{p}$ is presented in homogenous coordinates. The 3rd element is not necessarily equal to one.

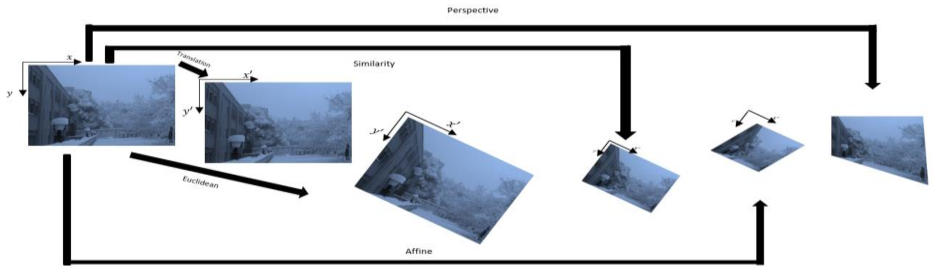
It has been a long journey, but not finished yet!

- We've found unique points in each of these two images.
  - We found corner points.
- We know which unique point in the first image is similar to another unique point in the second image.
  - We can compare features like gradients, edges, … .
- We studied different kinds of image transformations.
- Now, it's time to find the right transformation matrix to transform points from one coordinate to the other one.





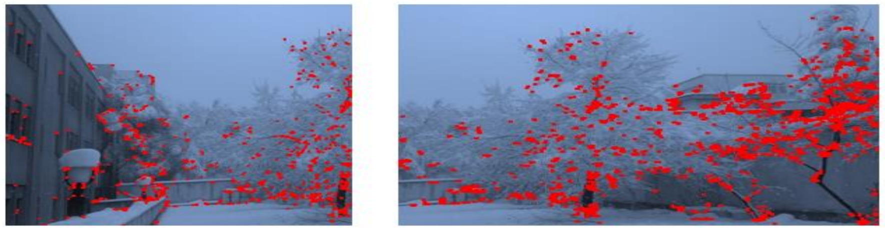
x
y
x'
y'
$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 400 \\ 550 \end{bmatrix}$
$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 170 \\ 560 \end{bmatrix}$

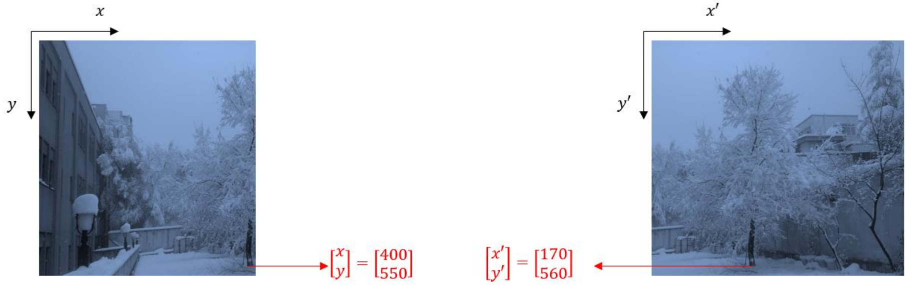


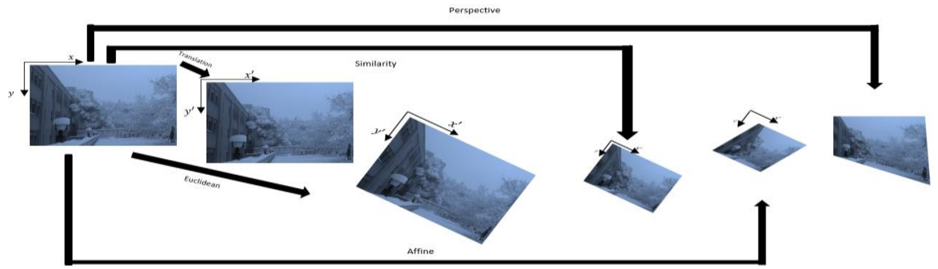
Finding the missing piece

- Now, it's time to find the right transformation matrix to transform points from one coordinate to the other one.

- In other words, the transformation matrix is unknown!
  - $T = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$
- For simplicity, imagine that the transformation is affine, so we have less unknown variables to worry about:
  - $T = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ 0 & 0 & 1 \end{bmatrix}$




x
y
x'
y'
$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 400 \\ 550 \end{bmatrix}$
$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 170 \\ 560 \end{bmatrix}$



Finding the missing piece

- We have an unknown affine transformation:
  - $T_{affine} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$
- Remember the equation:
  - $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$
  - Or:
    - ✓ $x = a.x' + b.y' + c$
    - ✓ $y = d.x' + e.y' + f$
- Let’s manipulate this equation a little bit:
  - $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' & y' & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x' & y' & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$

Finding the missing piece

- Let’s manipulate this equation a little bit:
  - $$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' & y' & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x' & y' & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$$
- We can write this relation for every points we found in the previous parts:
  - $$\begin{bmatrix} x_1 \\ y_1 \\ . \\ . \\ x_n \\ y_n \end{bmatrix}_{2n\times1} = \begin{bmatrix} x_1' & y_1' & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1' & y_1' & 1 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_n{}' & y_n{}' & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n{}' & y_n{}' & 1 \end{bmatrix}_{2n\times6} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}_{6\times1}$$

In what circumstances this system of equations has unique answers?

Finding the missing piece

- We can write this relation for every points we found in the previous parts:
  - $$\begin{bmatrix} x_1 \\ y_1 \\ . \\ . \\ x_n \\ y_n \end{bmatrix}_{2n\times1} = \begin{bmatrix} x'_1 & y'_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x'_1 & y'_1 & 1 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_n{}' & y_n{}' & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n{}' & y_n{}' & 1 \end{bmatrix}_{2n\times6} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}_{6\times1}$$

At least we need 3 points, but we want lots of points in practice! The reason is that there may be noise or incorrectly matched points. We want to use as many points as possible to reach a good estimation.

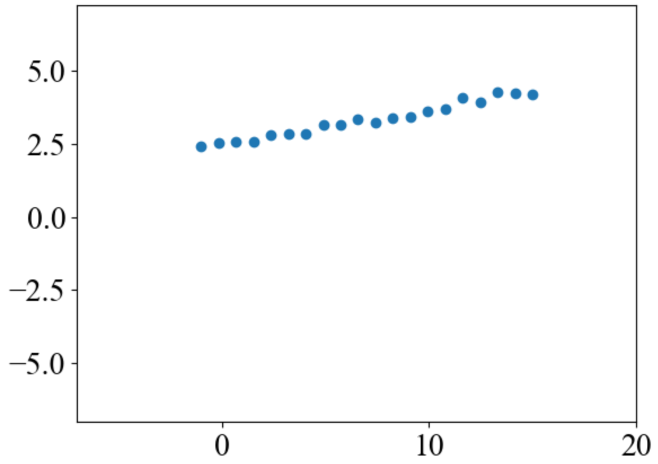
- Or equivalently we can write:
  - $Y_{2n\times1} = A_{2n\times6}.X_{6\times1}$

- Mathematically speaking, when $n > 3$, there are no solutions for the system of equations!

No solutions? But, why?

- Visualization helps to understand it better, but it is hard to visualize for six variables.
- Let’s simplify the problem for now:
  - Say we have a bunch of points:
    - ✓ $p_1 \;\;= (-1 \quad\;\;, \;2.14)$
    - ✓ $p_2 \;\;= (-0.16, \;2.54)$
    - ✓ …
    - ✓ $p_{20} = (15 \quad\;\;, \;4.17)$
  - We want to find the slope and intercept of a line which passes through all these points:
    - ✓ $y = w.x + b$
  - If the points were on a straight line, we knew how to find w and b but here, the points are “**almost**” on a straight line.
  - The relation between x and y is not strictly linear.

Can you graphically find the straight line?



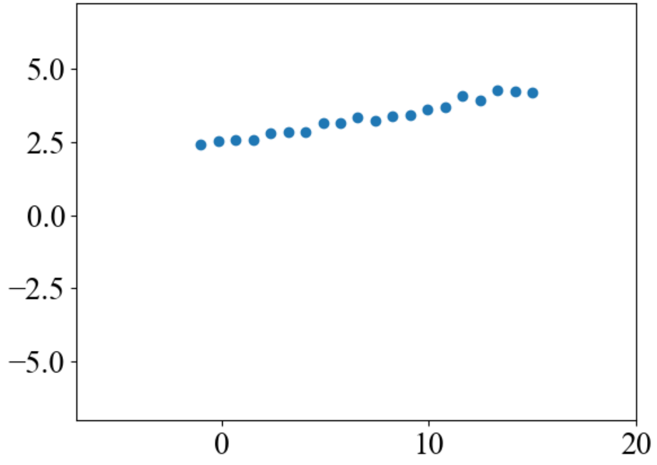
We need to compromise

- If we want the exact solution, it is not linear, we should look for a non-linear equation.

- Even in non-linear estimations, we do not find the exact relationship, and try to approximate.

- In the current case, a line looks fine enough.

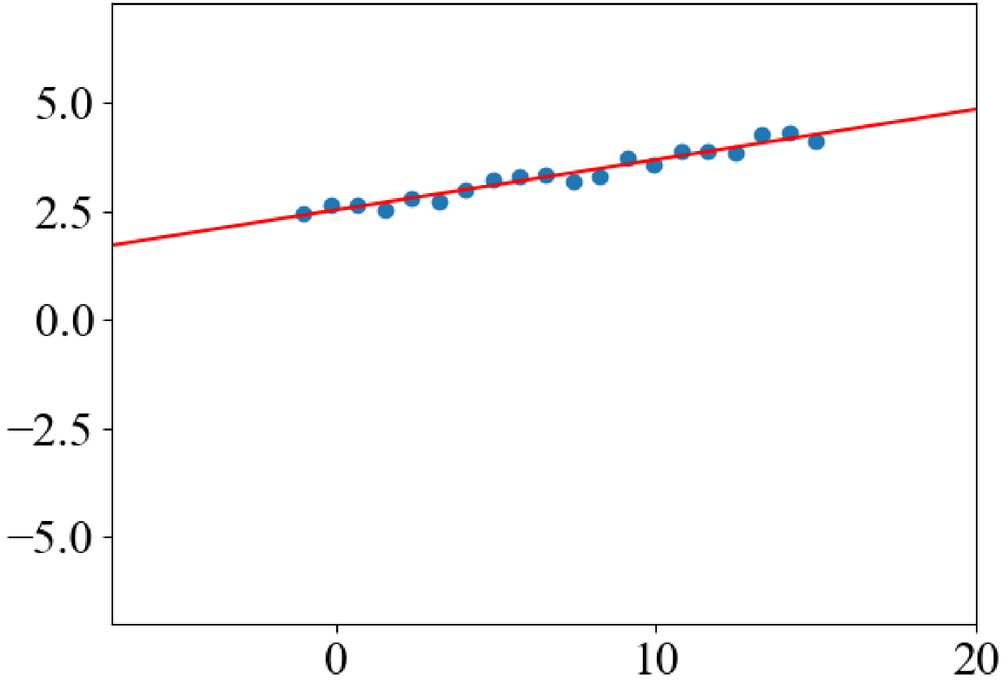
- Among all the lines we can choose, let us a line that has the least square error for all the points.



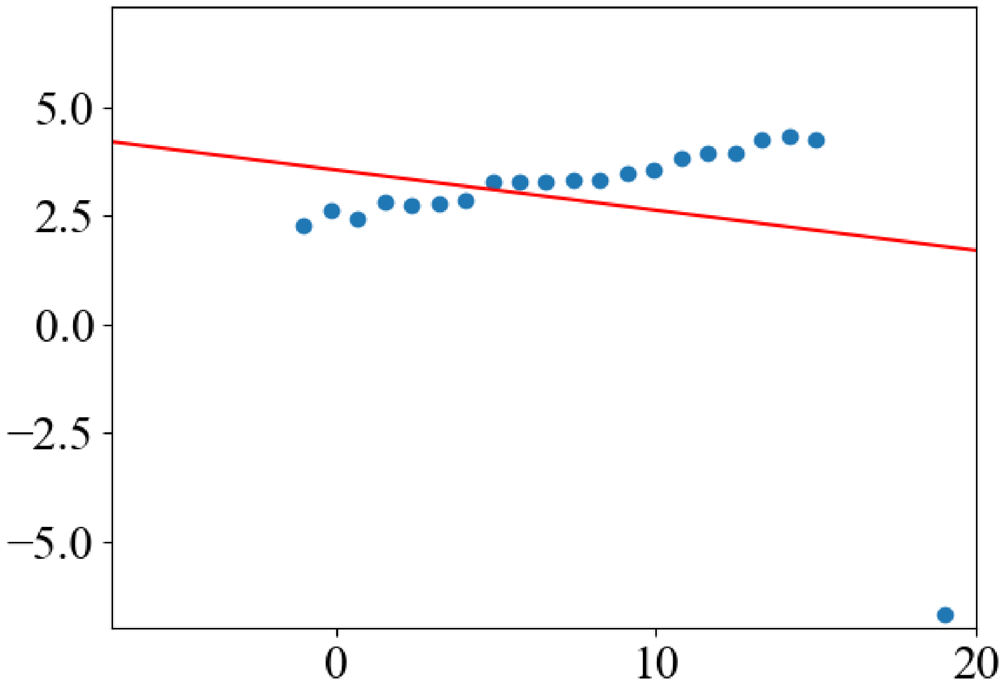
Least square

- We want X, which delivers the least squared error:
  - Squared Error $= |Y - AX|^2 = |Y - AX|^T |Y - A.X|$
  - Squared Error $= Y^T Y - Y^T AX - X^T A^T Y - X^T A^T AX$
  - $\frac{\partial Squared\ Error}{\partial X} = 0$
    - ✓ $\frac{\partial}{\partial X}[Y^T Y - Y^T AX - X^T A^T Y - X^T A^T AX\ ] = 0$
    - ✓ $-A^T Y - A^T Y + A^T AX + A^T AX = 0$
    - ✓ $\boldsymbol{X} = \left(\boldsymbol{A^T A}\right)^{\boldsymbol{-1}} \boldsymbol{A^T Y}$

The least square method is fine, but…

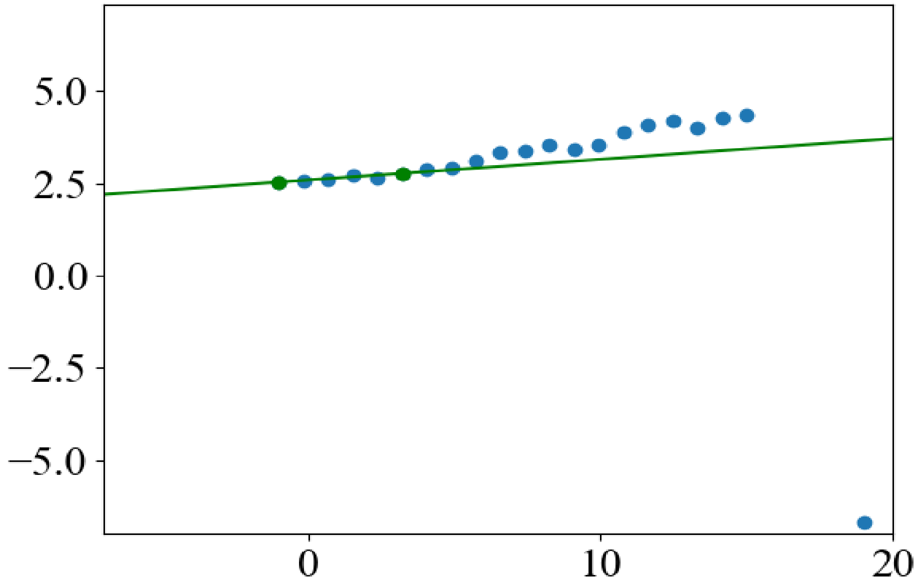


The least square is sensitive to the outliers



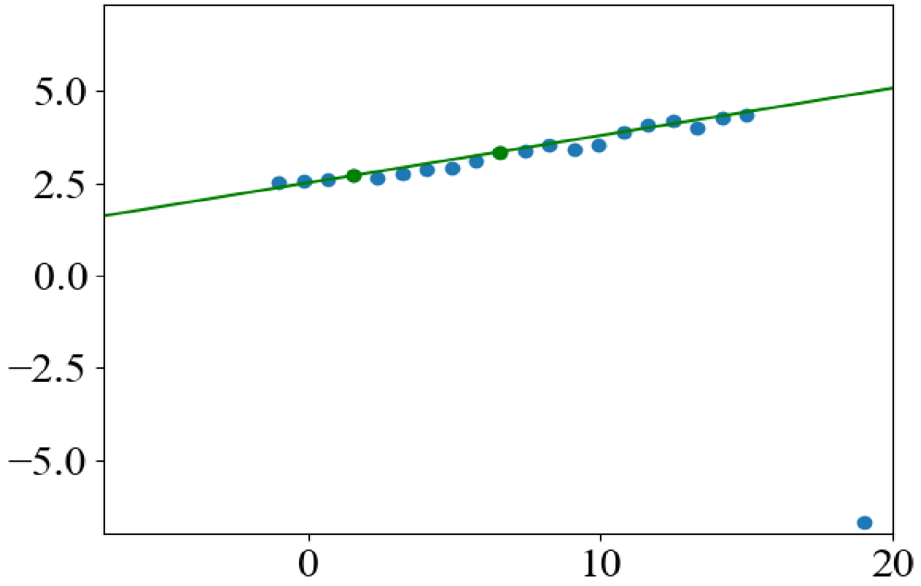
How can we neglect the outliers?

- What if we only use two points for finding the line?
- Is this line a good line?



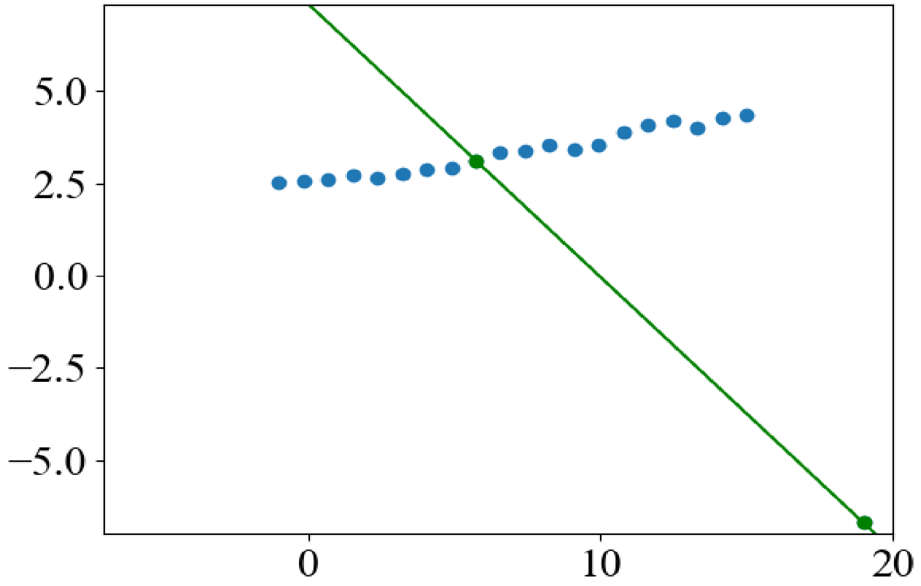
How can we neglect the outliers?

- How about this one?



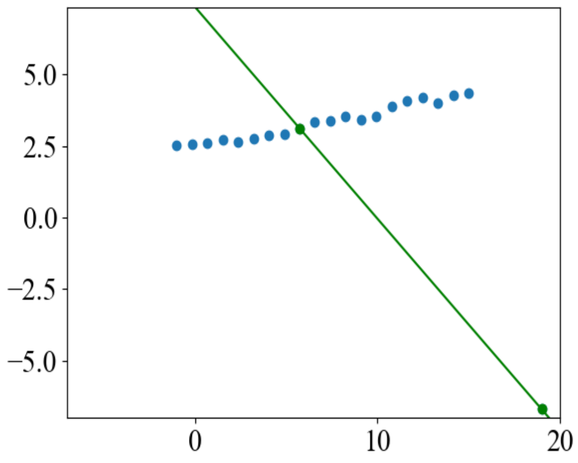
How can we neglect the outliers?

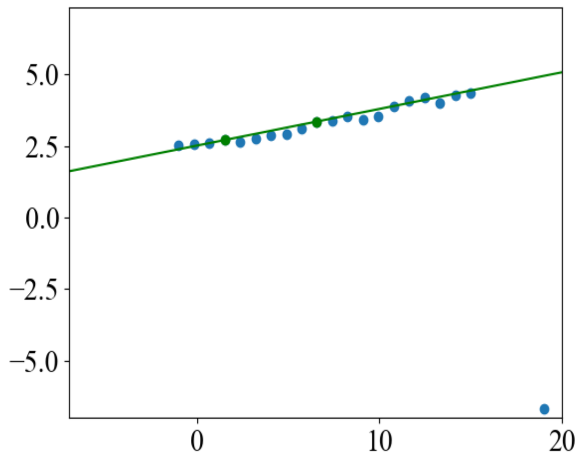
- And this one?

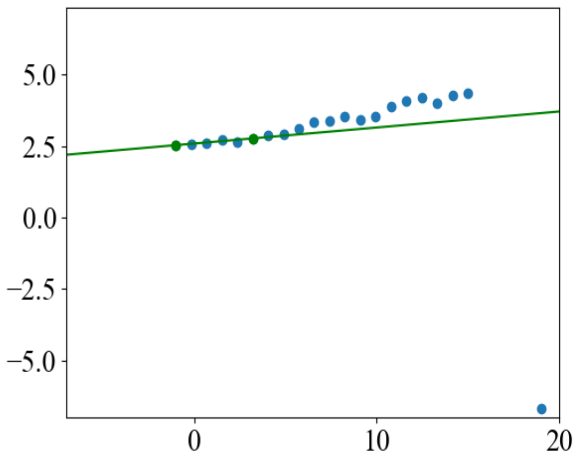


How can we neglect the outliers?

- Why is the middle line better than the two other lines?

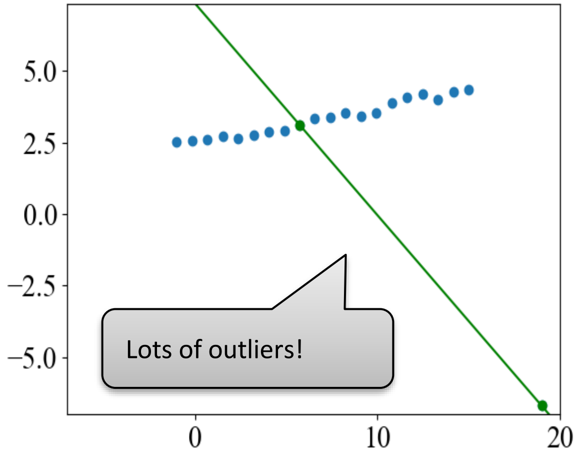


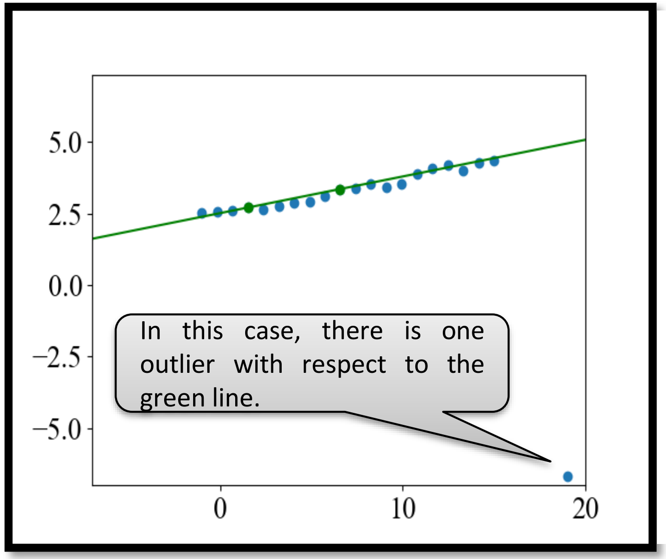


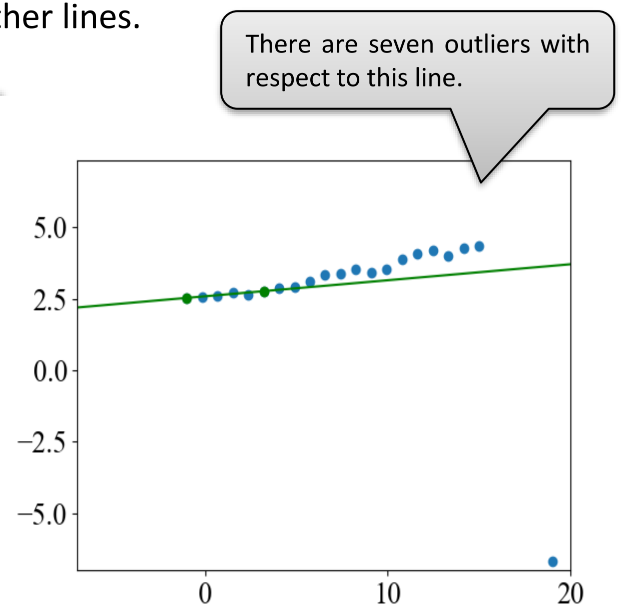


How can we neglect the outliers?

- Why is the middle line better than the two other lines?
  - It has less error than the two other lines.
    - ✓ In other words, the number of points which are outliers with respect to the middle line is less than the outlier points with respect to the other lines.

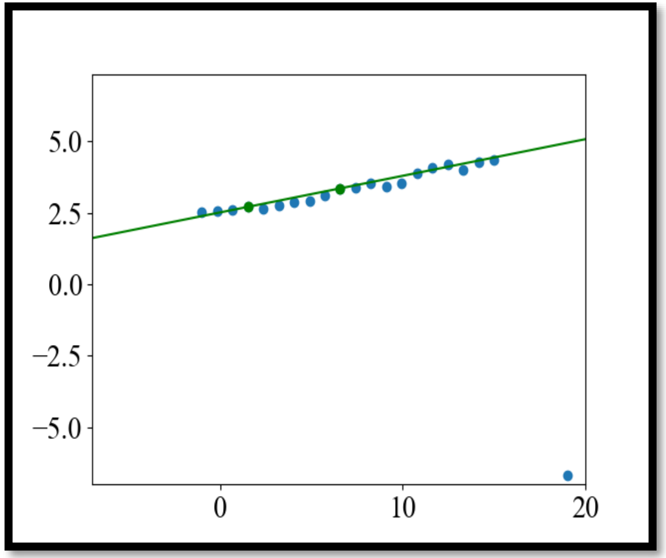






How can we neglect the outliers?

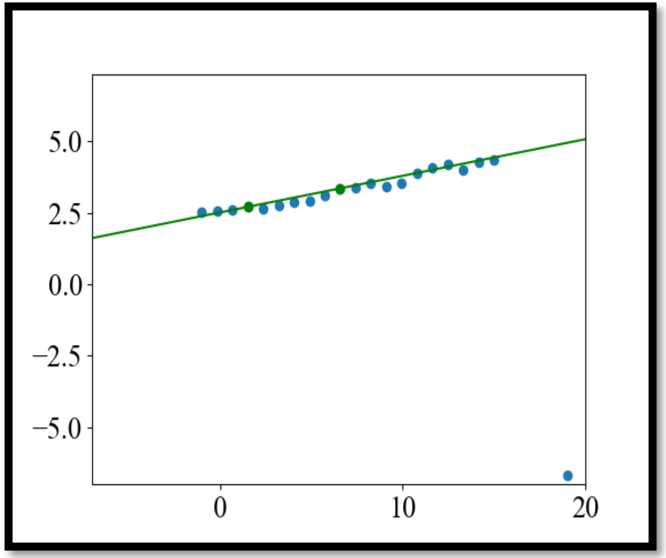
- Find the all possible combinations:
  - p_1 & p_2
  - p_1 & p_3
  - …
  - p_m & p_n
- For each combination:
  - find the line passing through those two lines.
  - Find the error for all the other points.
  - If the error is greater than a threshold, consider that point as an outlier; otherwise, it is an inlier.
- Choose the line with the least error (or the least number of inliers).



If the number of samples is very large, this method is computationally very expensive. There are lots of combinations to try.

RANSAC: RANdom SAmple Consensus

- Instead of trying all the possible combinations, repeat the following lines **k** times:
  - Best model = None
  - Best fit = INF
  - While iteration<k:
    - ✓ Choose n random samples
    - ✓ Fit model to these samples
    - ✓ If estimation error> threshold, it is outlier.
    - ✓ Find the number of outliers
      - ➢ If the no. of outliers < Best fit:
        - ❖ Best fit = newly fitted model
      - ➢ If outliers < desired value:
        - ❖ Best fit = newly fitted model
        - ❖ break


5.0
2.5
0.0
−2.5
−5.0
0
10
20