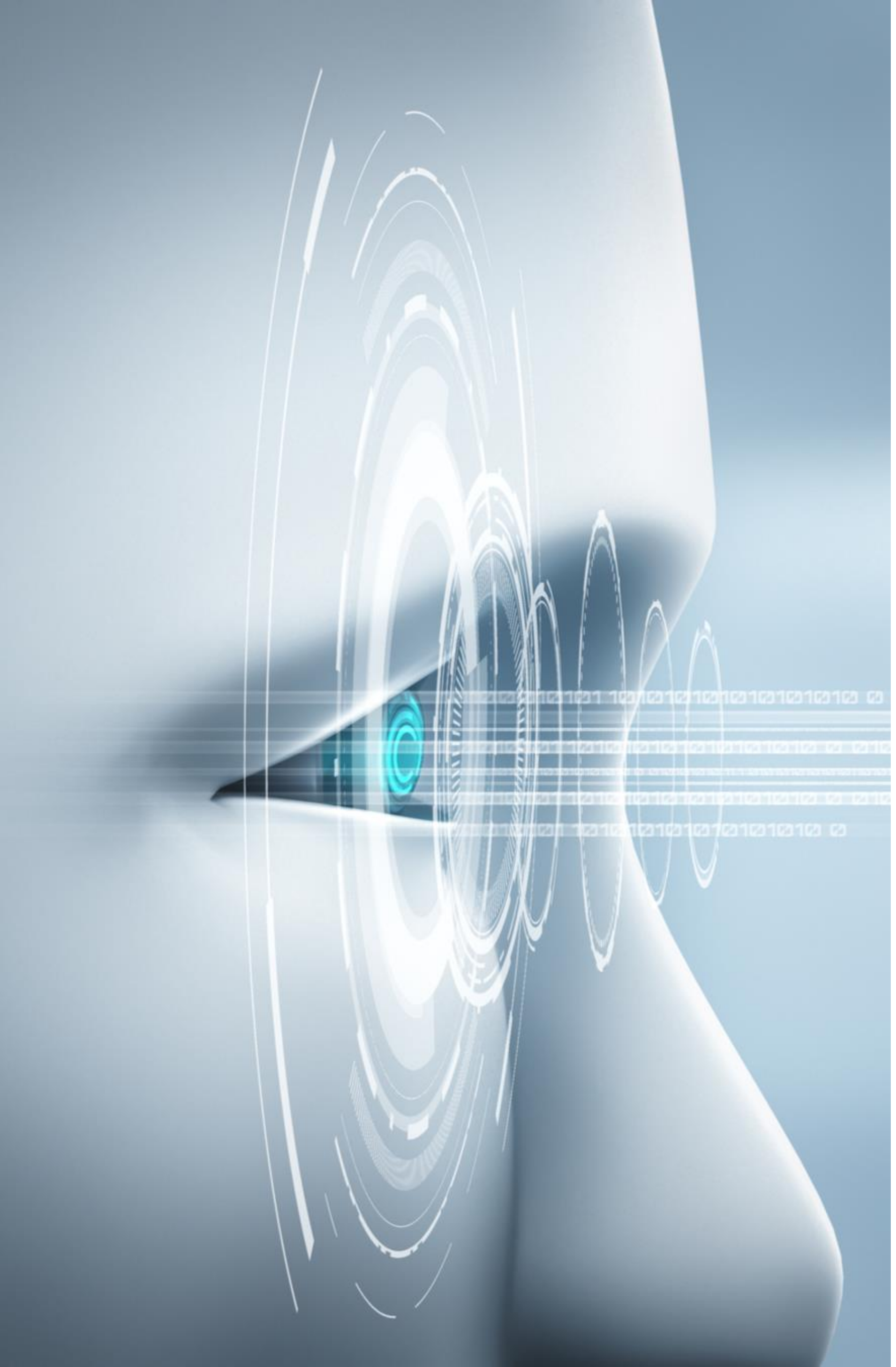


Computer Vision Applications

BY QUADEER SHAIKH

About me

Work Experience

- Risk Analyst
  - Morgan Stanley (Jan 2023 – Present)
- Data Science Intern
  - AkzoNobel Coatings International B.V. Netherlands (Feb 2022 – Dec 2022)
- Data Science Intern
  - EzeRx Health Tech Pvt. Ltd. (Jan 2022 – July 2022)
- Associate Engineer
  - Tata Communications Ltd. (July 2019 – Aug 2020)
- Network Automation and Analysis Engineer Intern
  - Cisco (June 2018 – July 2018)

Education

- M.Tech – Artificial Intelligence
  - NMIMS (2021 - 2023, currently pursuing)
- B.E. – Computer Engineering
  - Mumbai University (2015 - 2019)

Image Operations

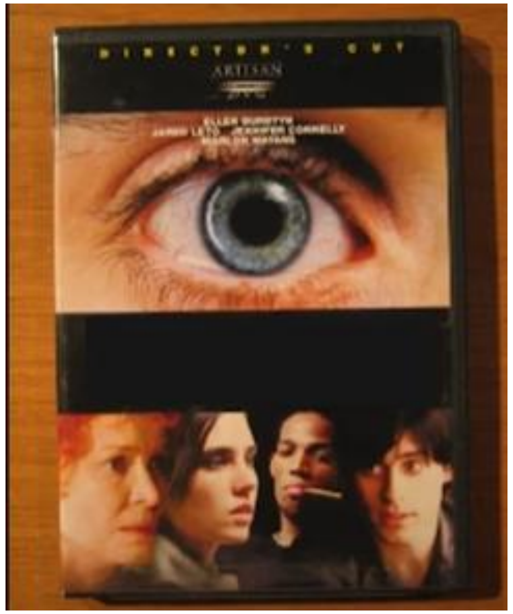
1. Pixel based operations
    1. Contrast Stretching
    2. Thresholding
    3. Inverting/Negative Images
    4. Erosion and Dilation
    5. Bitwise operations, etc.
2. Regions based operations
    1. Contour Detection
    2. **Edge Detection**
    3. **Corner Detection (not included)**
    4. **Line Detection, etc.**

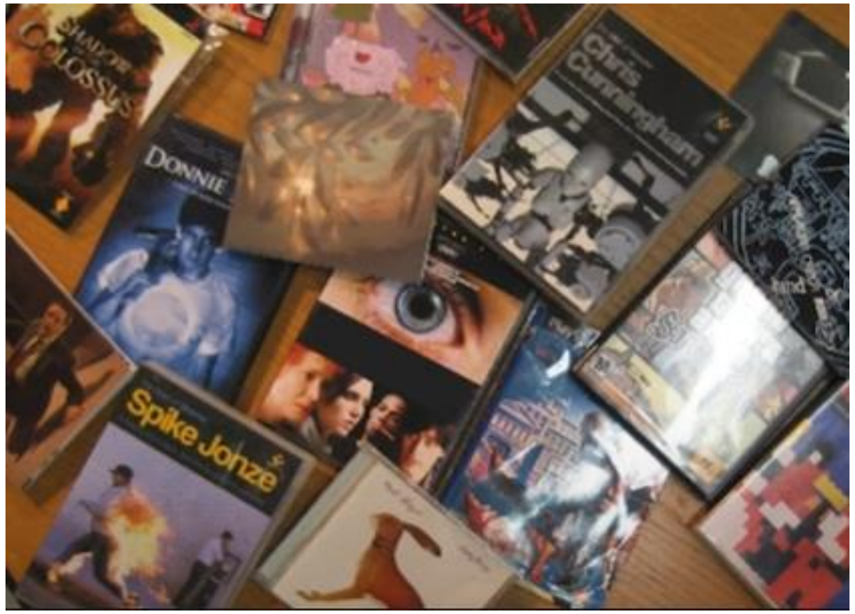
Features in Images

Features in an Image

Lines, edges, ridges, corners, blobs, etc

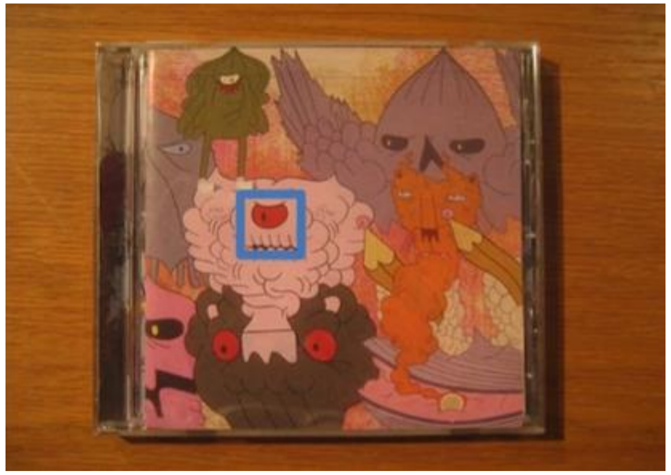
Which of these can be considered to be a good feature for a **recognition** task ?

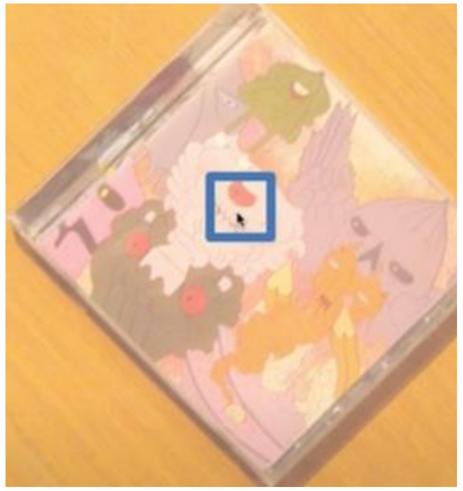




Challenges in matching images (recognition tasks)

1. Raw images are hard to match
2. Different size, orientation, lighting, brightness, etc



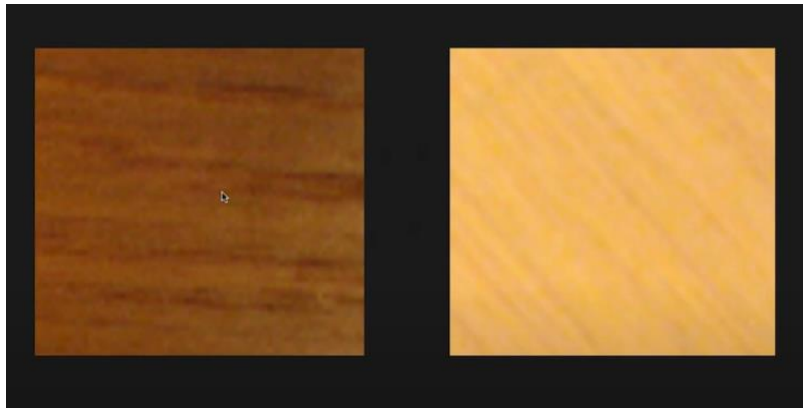


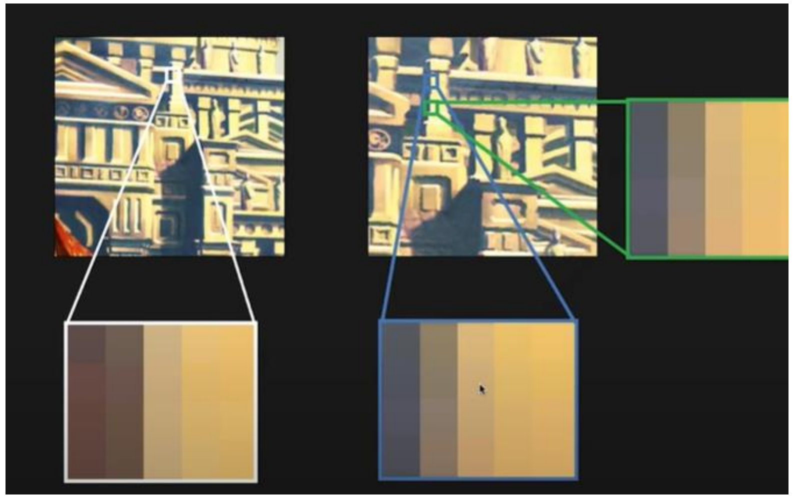
Solution to challenges in matching images (recognition tasks)

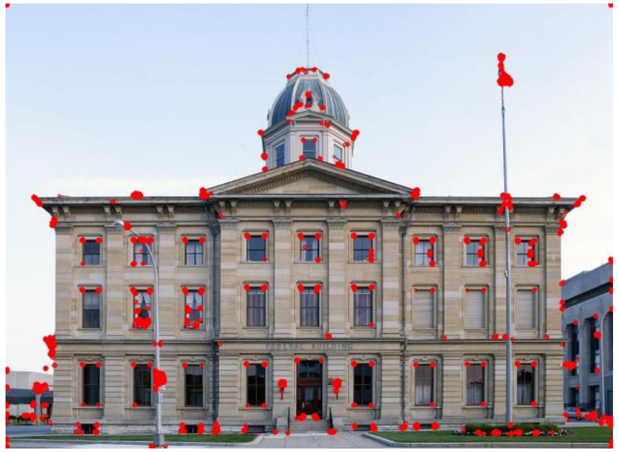
1. Removing sources of variation
    1. Matching becomes easier when we remove variations like change in size and orientation
2. Find some interesting points/features in images that can be matched
    1. Has rich image content (brightness variation, color variation, etc) within a local area
    2. Has well defined representation/signature for matching with other images
    3. Has well defined position in the image
    4. Should be invariant to image rotation and scaling

What are interesting points in an image ?

- Are random image patches interesting ?
- Are Lines interesting ?
- Are Corners interesting ?
- Are Edges interesting ?







What are interesting points in an image ?

- Are blobs interesting ? **Yes**
- Blobs are a group of pixel values that forms a somewhat colony or a large object that is distinguishable from its background
- They are considered to be interesting points/salient features

Blobs as Interesting Points

For a blob like feature to be useful we need to do the following

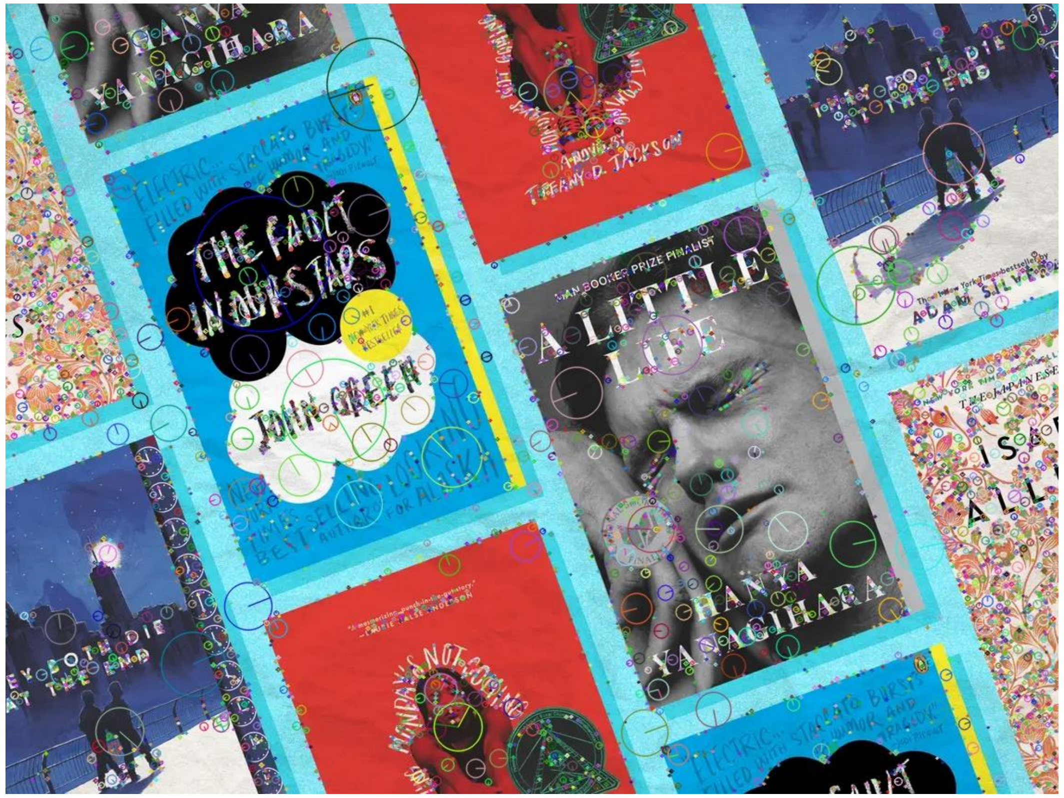
1. Locate the blob
2. Determine its size
3. Determine its orientation
4. Formulate a descriptor/signature

We will achieve this using the SIFT Algorithm (Scale Invariant Feature Transformation)

SIFT (Scale Invariant Feature Transform)

1. It is a feature detection algorithm in Computer Vision
2. SIFT helps locate the local features in an image, commonly known as the '*keypoints*' of the image. These keypoints are scale & rotation invariant that can be used for various computer vision applications, like image matching, object detection, scene detection, etc.
3. **The major advantage of SIFT features, over edge features or other features, is that they are not affected by the size or orientation of the image.**

SIFT (Scale Invariant Feature Transform)



SIFT Algorithm

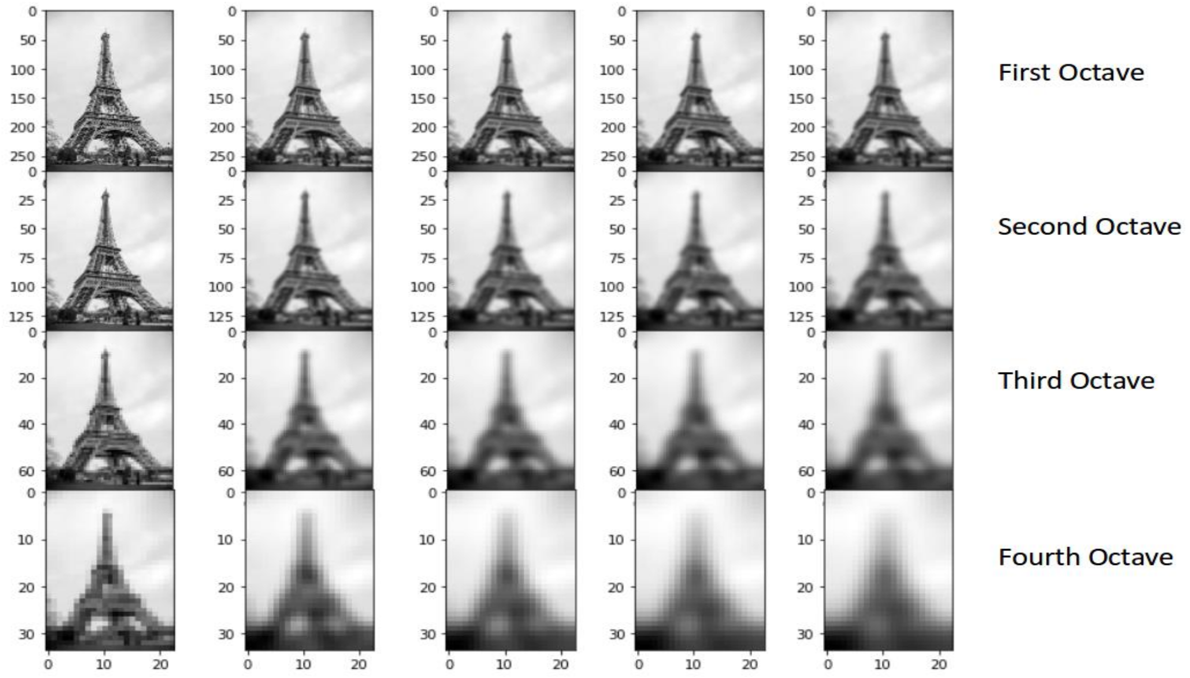
1. Construct a Scale Space – To make feature scale independent
2. Keypoint Localisation – Identifying the locations of suitable features/keypoints
3. Orientation Assignment – Ensuring that the keypoints are rotation invariant
4. Keypoint Descriptor – Creation of a unique descriptor/signature of detected keypoints

Construction of Scale Space

- Identification of most distinct features that are not scale dependent and ignoring any noise present
- Convert the image into grayscale and apply gaussian blur to remove noise
- Since we need to ensure that the features must be scale independent we need to search for the features on multiple scales by creating a scale space
- To create new sets of images of different scales we take the original image and keep on reducing its scale by half and create different blur versions of it

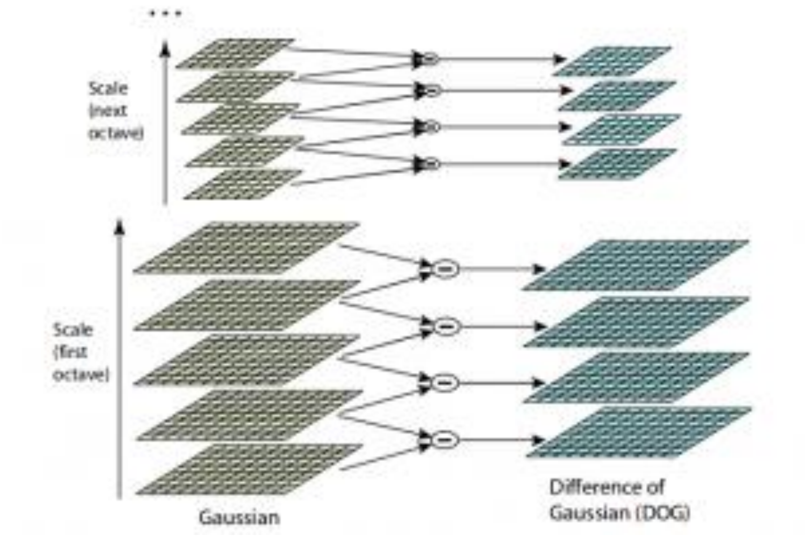
Construction of Scale Space

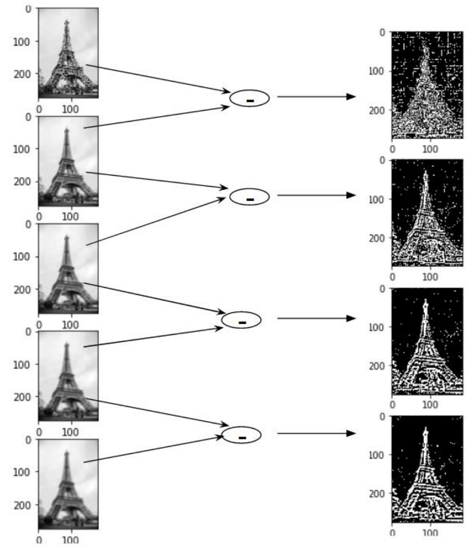
The ideal number of octaves should be four, and for each octave, the number of blur images should be five.



Construction of Scale Space

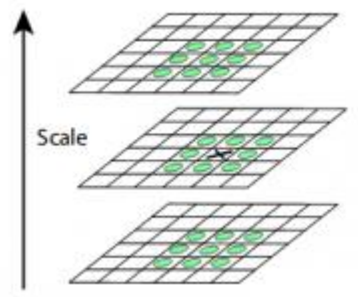
We have created images of multiple scales and used Gaussian blur for each of them to reduce the noise in the image. Now, we will try to enhance the features using a technique called Difference of Gaussians or DoG. DoG creates another set of images, for each octave, by subtracting every image from the previous image in the same scale.





KeyPoint Localization: Local Maxima and Local Minima

- Find the important key points from the image that can be used for feature matching by looking for maxima and minima in the result of DoG step.
- To locate the local maxima and minima, we go through every pixel in the image and compare it with its neighboring pixels
- Neighboring pixels not only includes the surrounding pixels of that image (in which the pixel lies), but also the nine pixels for the previous and next image in the octave. (compared with 26 other pixel values)
- One more check is applied to select the most accurate key points



Keypoint Localization: Selection

- We have now generate scale invariant keypoints, but some of these keypoints may not be robust to noise
- We will eliminate the keypoints that have low contrast, or lie very close to the edge
- To deal with low contrast keypoints
  - a second-order Taylor expansion is computed for each keypoint. If the resulting value is less than 0.03 (in magnitude), we reject the keypoint.
- To deal with keypoints that are too close to the edge
  - A second-order Hessian matrix is used to identify such keypoints and they are removed
- Now we have the most accurate keypoints which need to be assigned an orientation value so that they are rotation invariant

Orientation Assignment

- Now that we have the stable and accurate keypoints we will assign orientation to them so that they are rotation invariant
  - Calculate the orientation – Calculate the orientation by using gradient operators for all the pixel points that fall within the keypoint blob
  - Calculate the histogram of orientation – Create a histogram by taking the angles of all the pixels inside the blob, the angle bin with the peak value becomes the **principle orientation of the blob**

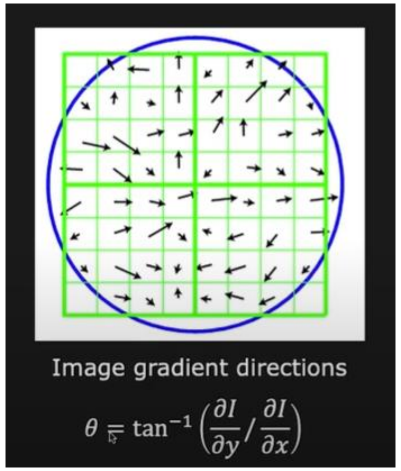
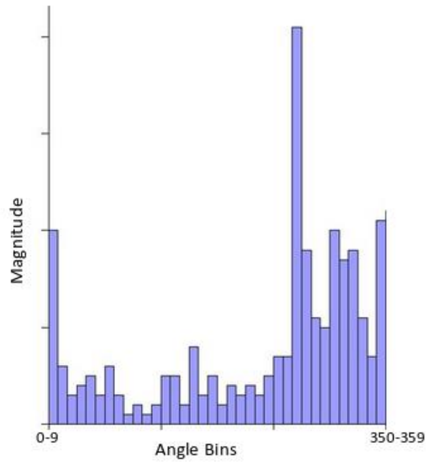
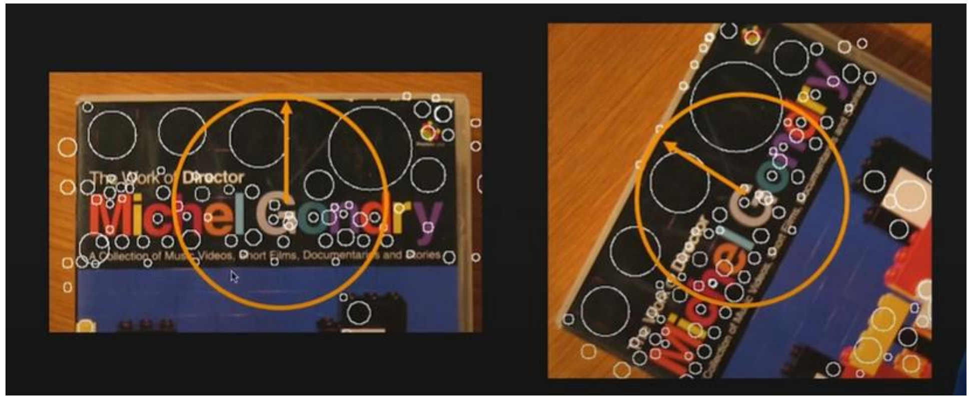
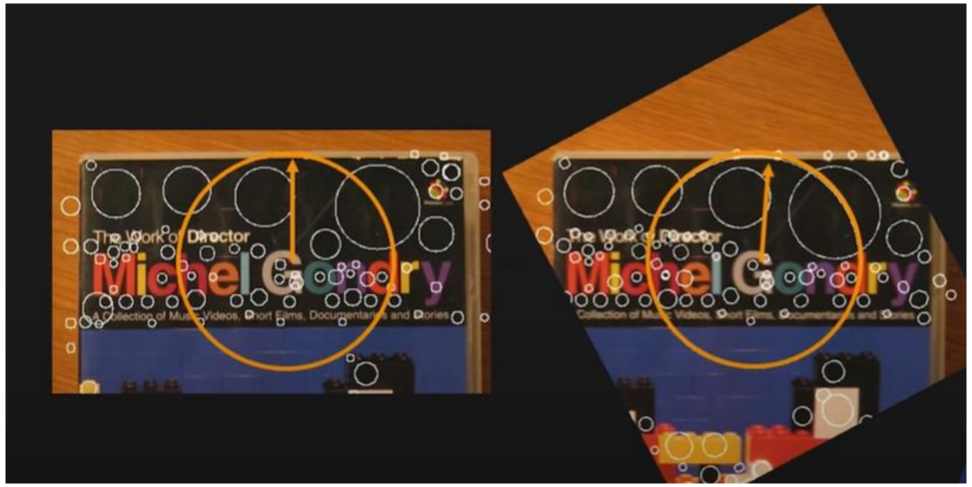


Image gradient directions
$\theta = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$



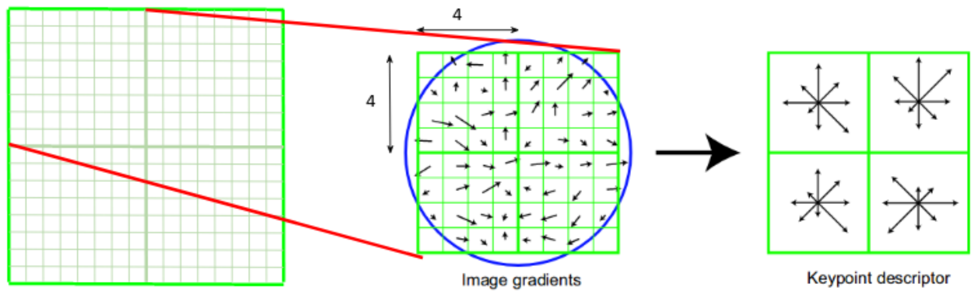
Why finding orientation is important ?

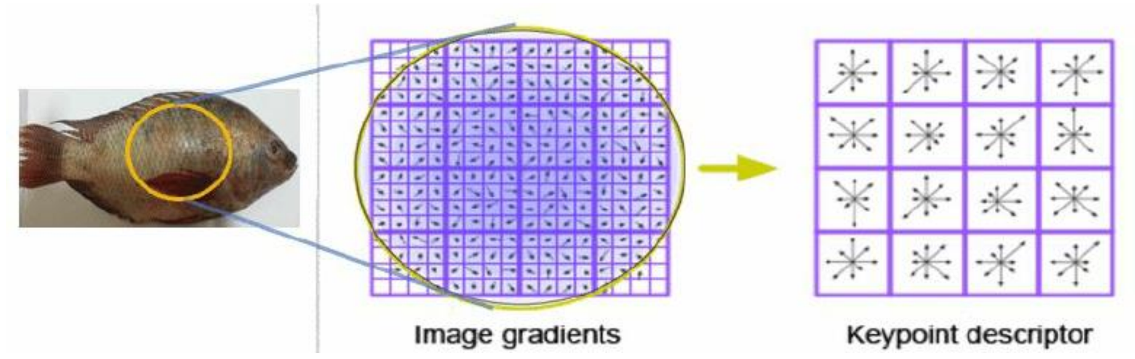

The Work of Director
Michel Gondry
A Collection of Music Videos, Short Films, Documentaries and Stories


The Work of Director
Michel Gondry
A Collection of Music Videos, Short Films, Documentaries and Stories

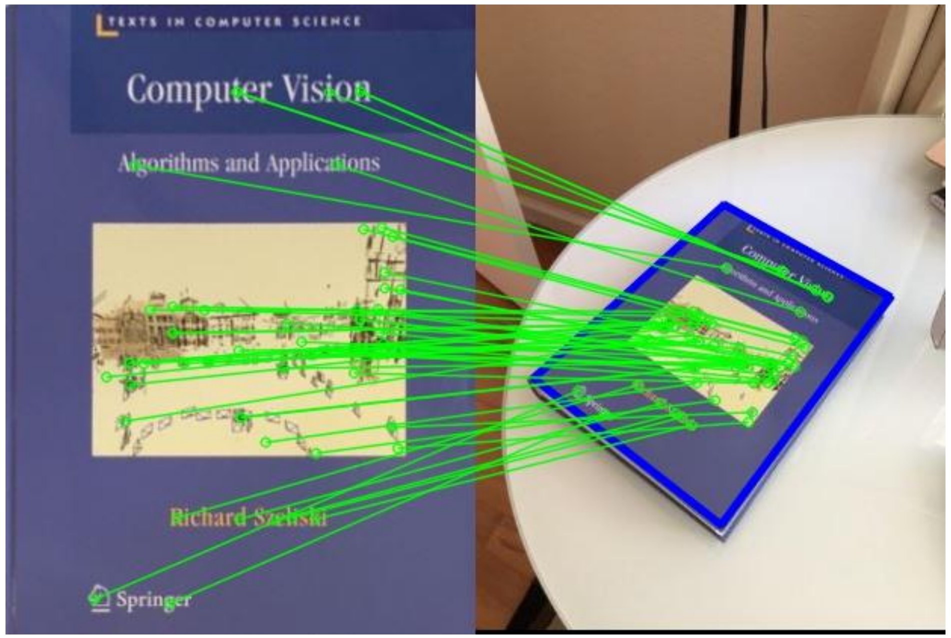
Keypoint Descriptor

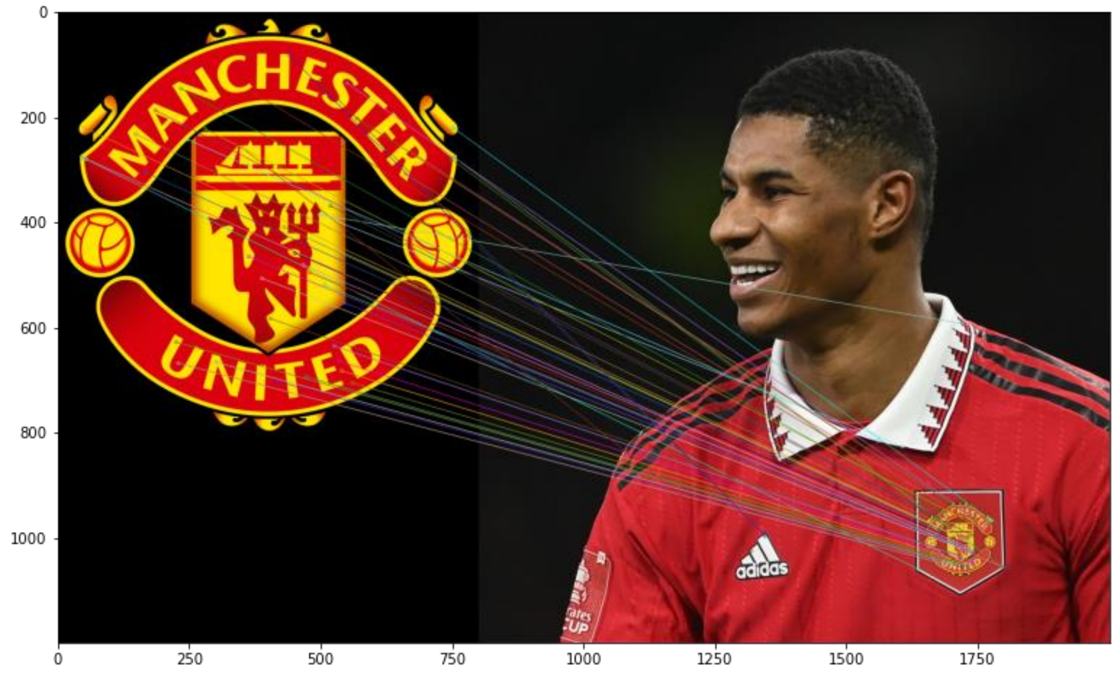
- Now that we have stable keypoints which are invariant to scale and rotation we just need to create a unique fingerprint for these keypoints called as descriptor which can be used for comparing the keypoints
- We will first take a 16×16 neighborhood around the keypoint. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available. It is represented as a vector to form keypoint descriptor.





SIFT Output

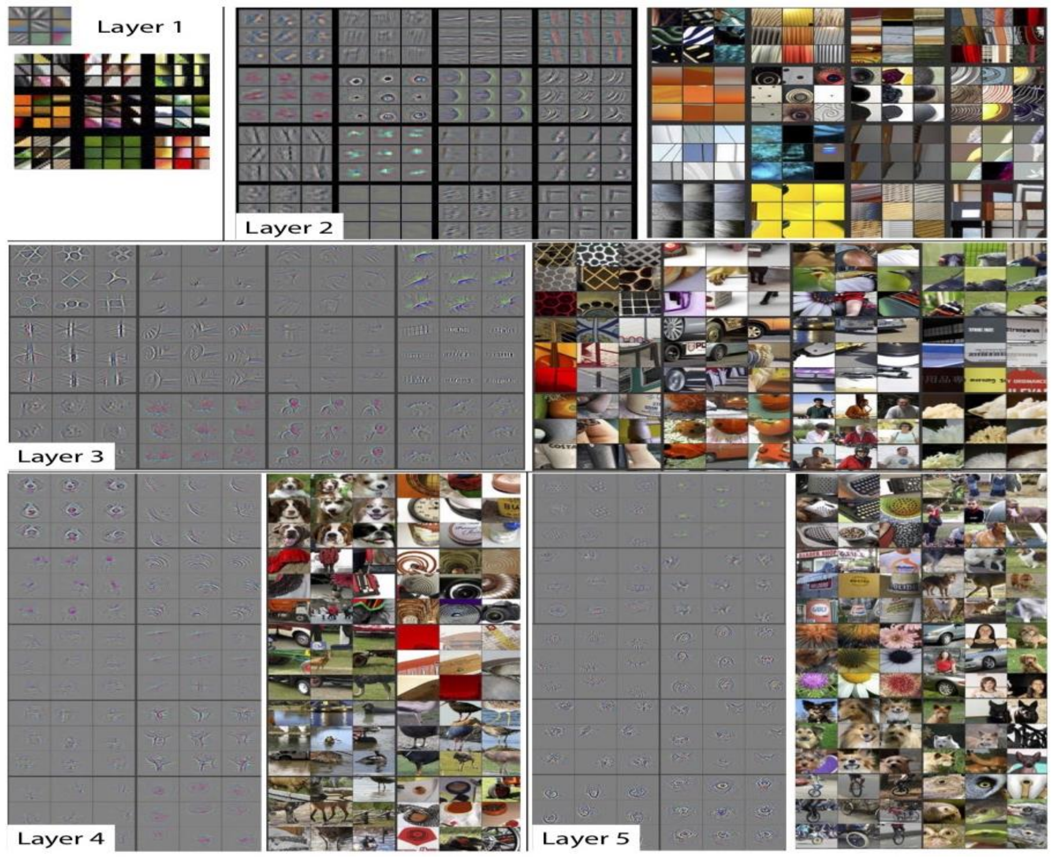




Other similar and improved algorithms: SURF, ORB, BRIEF

How CNNs learn ?

- They are neural networks that look at the spatial patterns and learn basic features like edges, lines, corners etc in the earlier layers and advanced features like blobs forming objects in the later layers
- The neural networks learn the patterns on their own during the training step itself
- **Note:** Topic will be covered in details during the deep learning based computer vision section



Geometric Image Transformations

Geometric Image Transformations

Filtering: Change the pixel values in the image

Transformation: Change the position/spatial coordinates of the image

Image Filtering

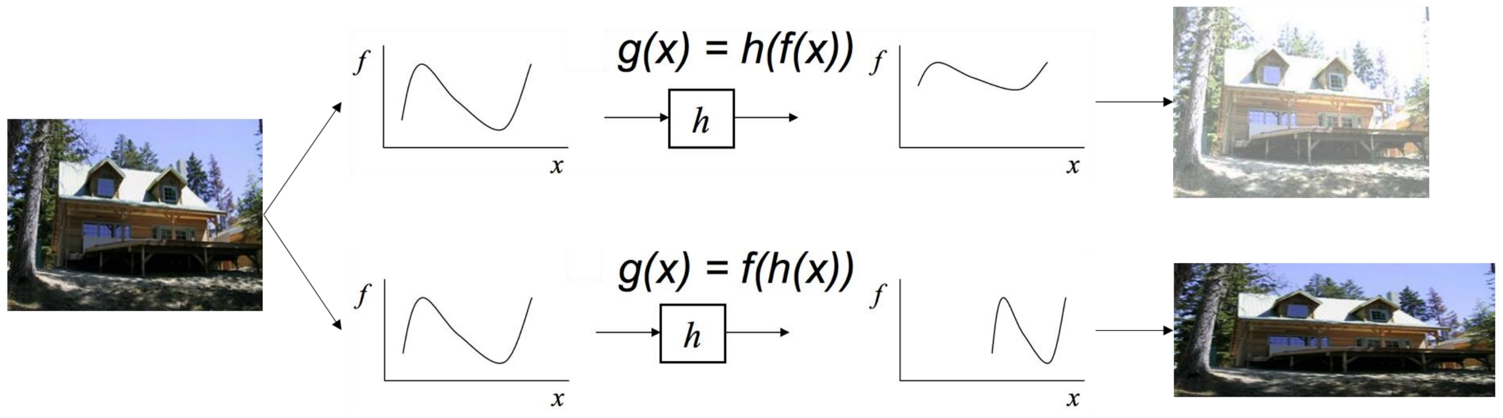


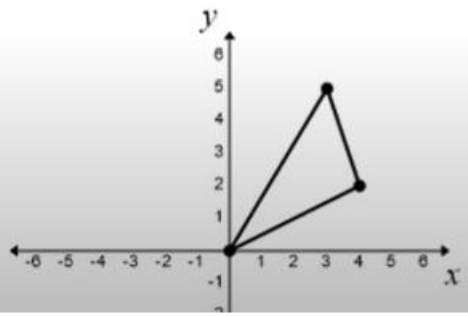
Image Warping

Example: How to perform a transformation on an image ?

Q. Consider the triangle represented by the matrix:

$$A = \begin{bmatrix} 0 & 3 & 4 \\ 0 & 5 & 2 \end{bmatrix} \begin{array}{l} \leftarrow x \text{ coordinates} \\ \leftarrow y \text{ coordinates} \end{array}$$

Use matrix operations to move the triangle <u>3 units to the left</u> and <u>1 unit down</u>.



Linear Transformations

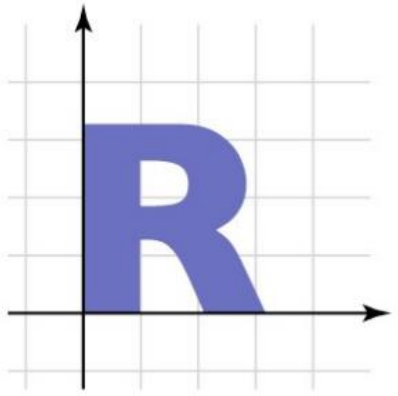
1. Scaling
2. Rotation
3. Reflection
4. Shearing
5. Translation

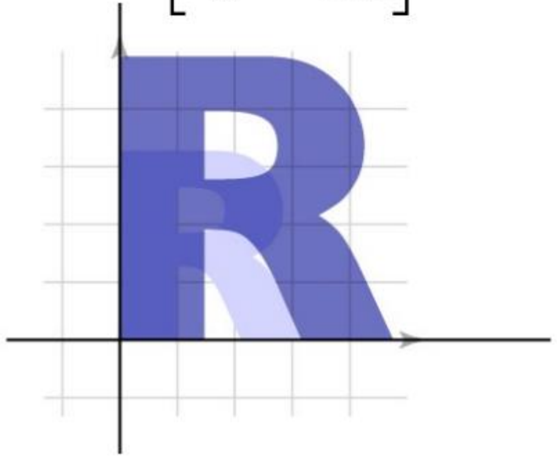
Scaling

- **Uniform scale**

$$\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} sx \\ sy \end{bmatrix}$$

$$\begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$



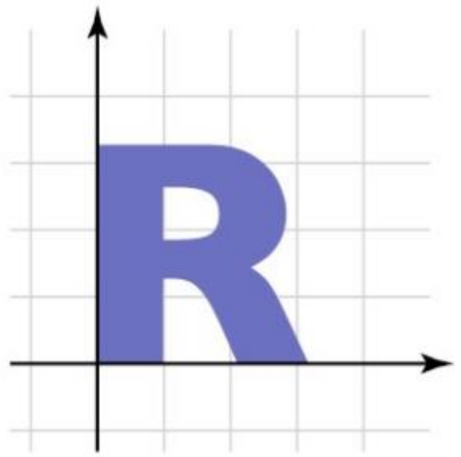


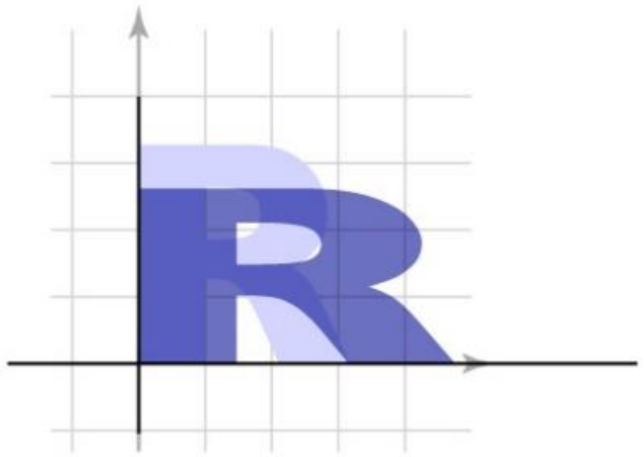
Scaling

- **Nonuniform scale**

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

$$\begin{bmatrix} 1.5 & 0 \\ 0 & 0.8 \end{bmatrix}$$

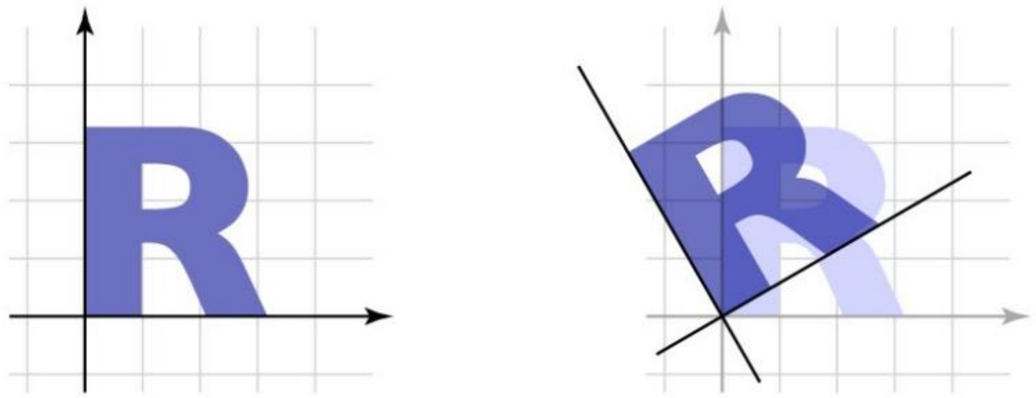




Rotation

- **Rotation** $$\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x \\ y\end{bmatrix} = \begin{bmatrix}x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta\end{bmatrix}$$

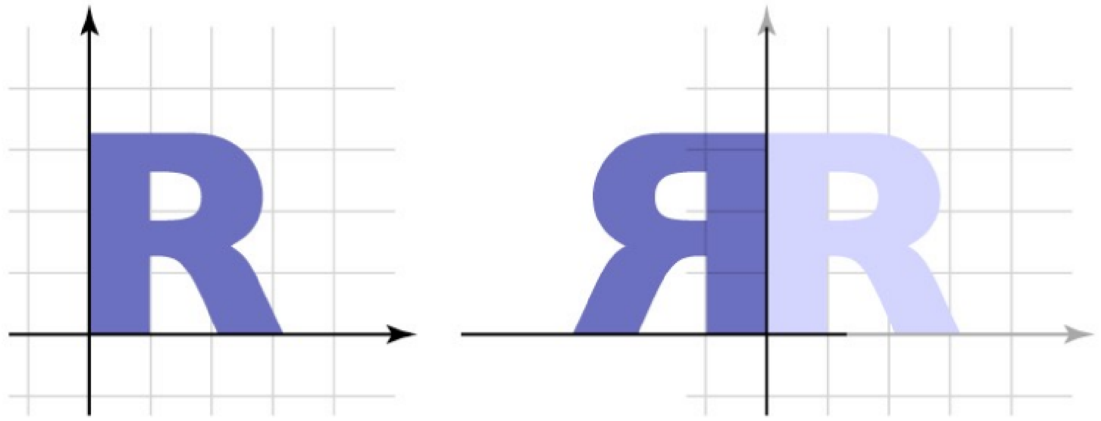
$$\begin{bmatrix}0.866 & -0.5 \\ 0.5 & 0.866\end{bmatrix}$$



Reflection

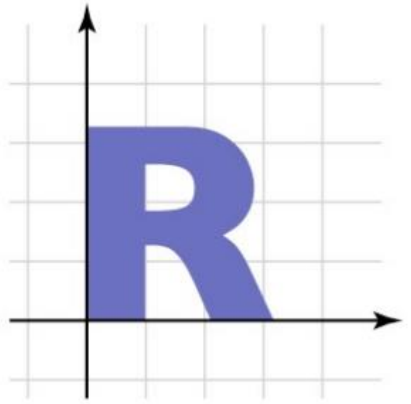
- **Reflection**
  - can consider it a special case of nonuniform scale

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

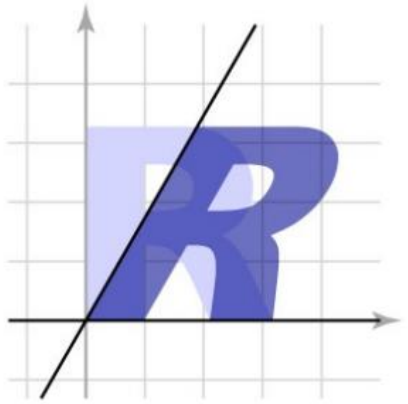


Shearing

- **Shear** $\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + ay \\ y \end{bmatrix}$

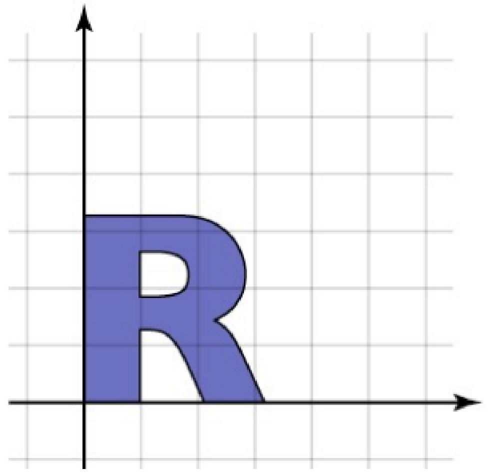


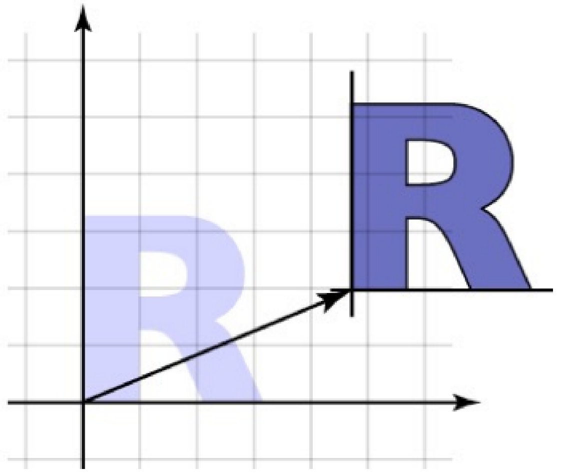
$$\begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$$



Translation

Can we represent the transformation matrix of translation using a 2x2 matrix ? **Ans: NO**





$$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} X \\ Y \\ 1 \end{matrix}$$

X*1 + Y*0 + tx = translated X
X*0 + Y*1 + ty = translated Y
X*0 + Y*0 + 1 = 1

Collectively known as Affine Transformation

Translate

$$\begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \cos(\Theta) & -\sin(\Theta) & 0 \\ \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

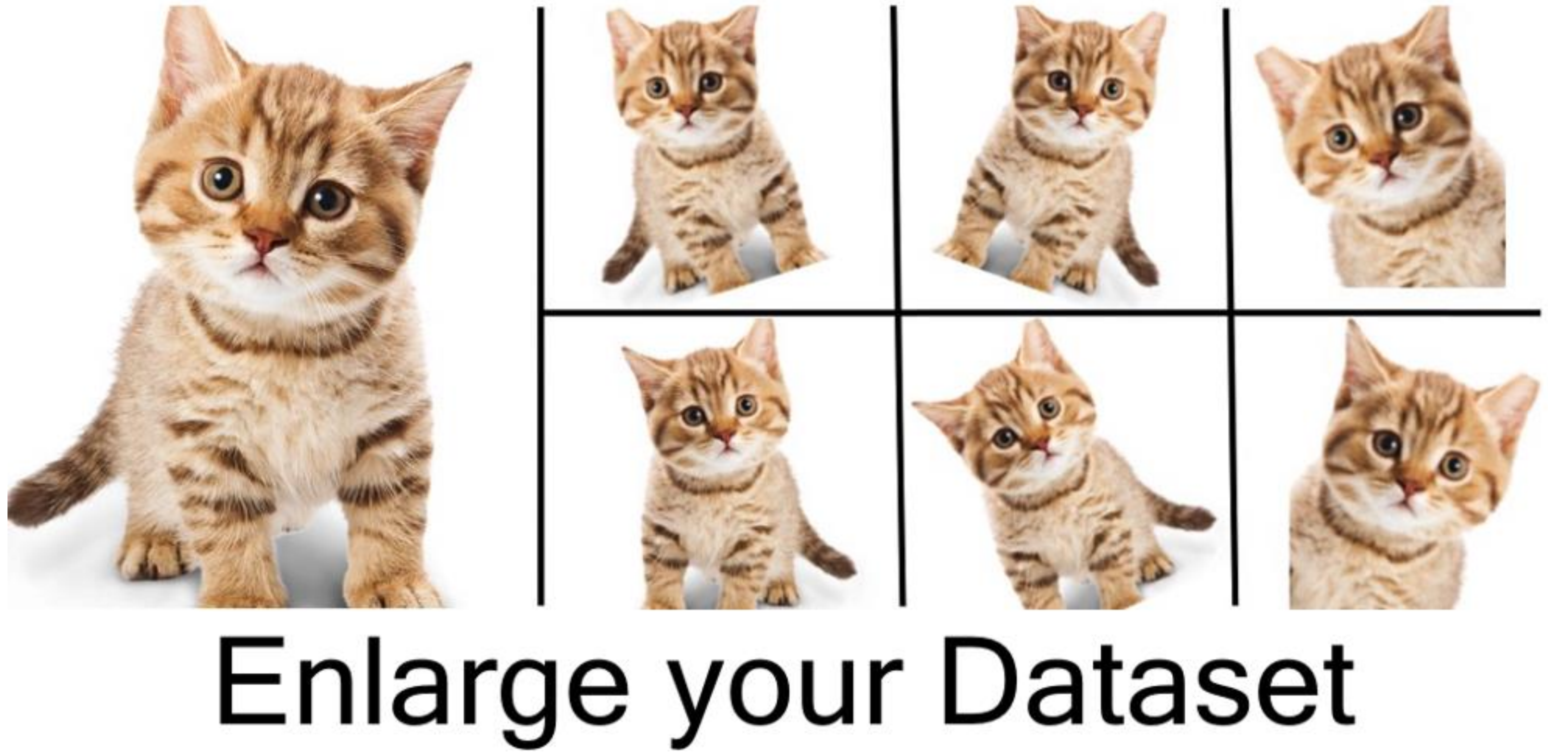
Shear

$$\begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{tx} \\ a_{21} & a_{22} & a_{ty} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}\times x + a_{12}\times y + a_{tx}\times 1 \\ a_{21}\times x + a_{22}\times y + a_{ty}\times 1 \\ 0\times x + 0\times y + 1\times 1 \end{bmatrix}$$

1. Combine all linear transformations into a single transformation matrix known as affine transformation matrix
2. It has 6 Degrees of Freedom (6 values where you can specify change/transformation)
3. I.e. you can perform translation+rotation+scaling using a single matrix
4. Lines map to lines, Parallel lines remain parallel
5. Origin may or may not always map to origin

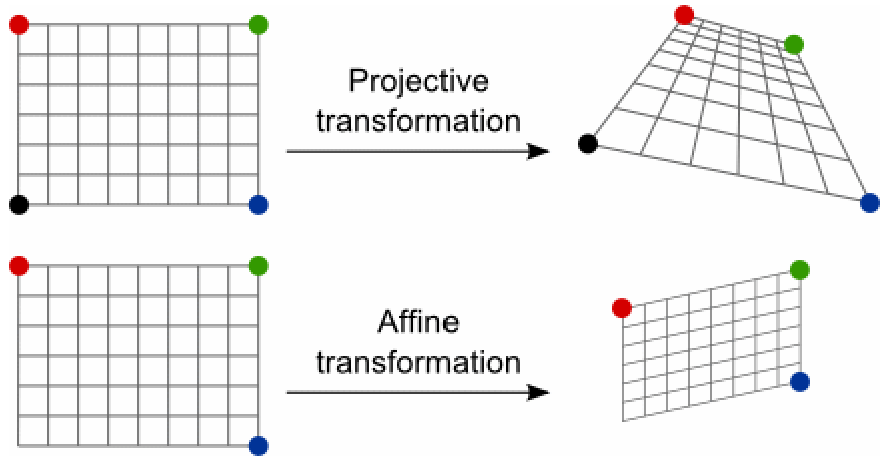
Affine Transformation Use Case

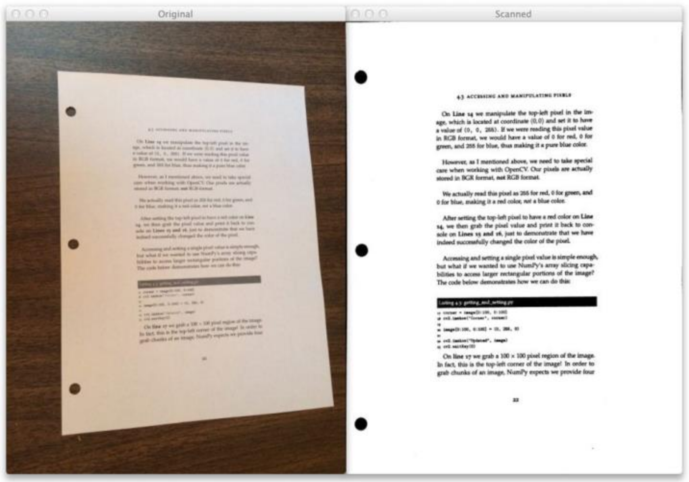


Projective Transformation

Mapping of one plane to another through points

E.g. Document scanners, tilted pictures of docs scanned and projected in an appropriate manner





4.3 ACCESSING AND MANIPULATING PIXELS

On **Line 14** we manipulate the top-left pixel in the image, which is located at coordinate (0,0) and set it to have a value of (0, 0, 255). If we were reading this pixel value in RGB format, we would have a value of 0 for red, 0 for green, and 255 for blue, thus making it a pure blue color.

However, as I mentioned above, we need to take special care when working with OpenCV. Our pixels are actually stored in BGR format, **not** RGB format.

We actually read this pixel as 255 for red, 0 for green, and 0 for blue, making it a red color, not a blue color.

After setting the top-left pixel to have a red color on **Line 14**, we then grab the pixel value and print it back to console on **Lines 15 and 16**, just to demonstrate that we have indeed successfully changed the color of the pixel.

Accessing and setting a single pixel value is simple enough, but what if we wanted to use NumPy's array slicing capabilities to access larger rectangular portions of the image? The code below demonstrates how we can do this:

Listing 4.3: getting_and_setting.py

```
17 corner = image[0:100, 0:100]
18 cv2.imshow("Corner", corner)
19
20 image[0:100, 0:100] = (0, 255, 0)
21
22 cv2.imshow("Updated", image)
23 cv2.waitKey(0)
```

On **line 17** we grab a 100 × 100 pixel region of the image. In fact, this is the top-left corner of the image! In order to grab chunks of an image, NumPy expects we provide four

22

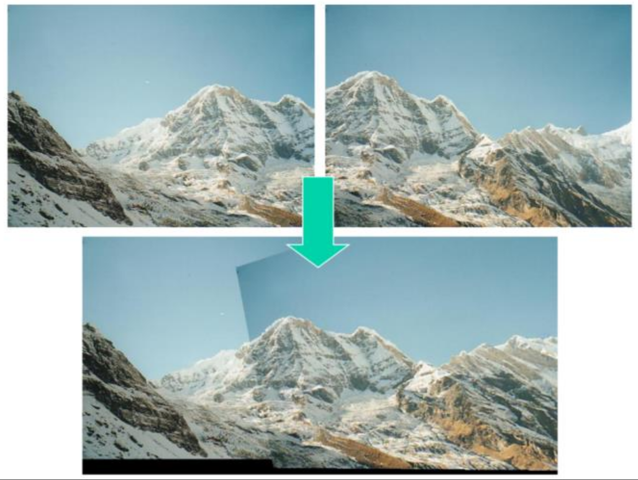
Projective Transformation

- Mapping of one plane to another through points
- E.g. Document scanners, tilted pictures of docs scanned and projected in an appropriate manner
- 3 x 3 Homography matrix is used to project one image from a plane onto another image plane
- All the values in the homography matrix should sum up to 1, therefore there are only 8 degrees of freedom and not 9
- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Image Stitching - Panorama

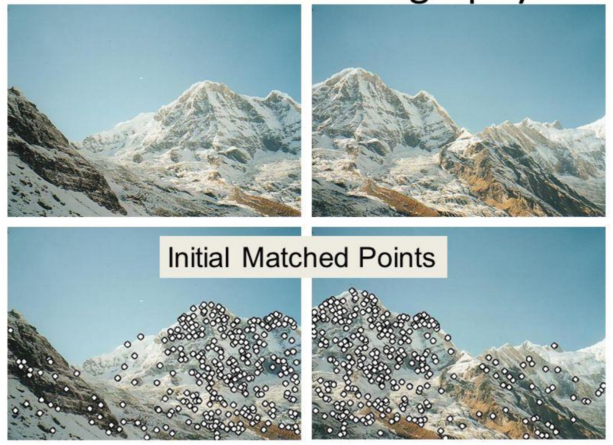
1. Detect Keypoints and extract local invariant descriptors from the two images – SIFT, ORB
2. Match the descriptors between two images
3. Use the RANSAC Algorithm to estimate a homography matrix using our matched feature vectors
4. Apply a warping/projective transformation using homography matrix



RAndom SAmple Consensus (optional)

1. Random choose **s** samples (typically 4). S is the min samples to fit a model
2. Fit the model to the randomly chosen samples
3. Count the number of data points/keypoints (inliers) that fit the model within a threshold error
4. Repeat steps 1 to 3 N times
5. Choose the model which has the larges number of inliers

RANSAC for Homography



Thank you

For any queries drop an email at: quadeershaikh15.8@gmail.com