CS 188-2
Discussion-Week 4

Ali Hatamizadeh
10/18/2019

Homogeneous Coordinates

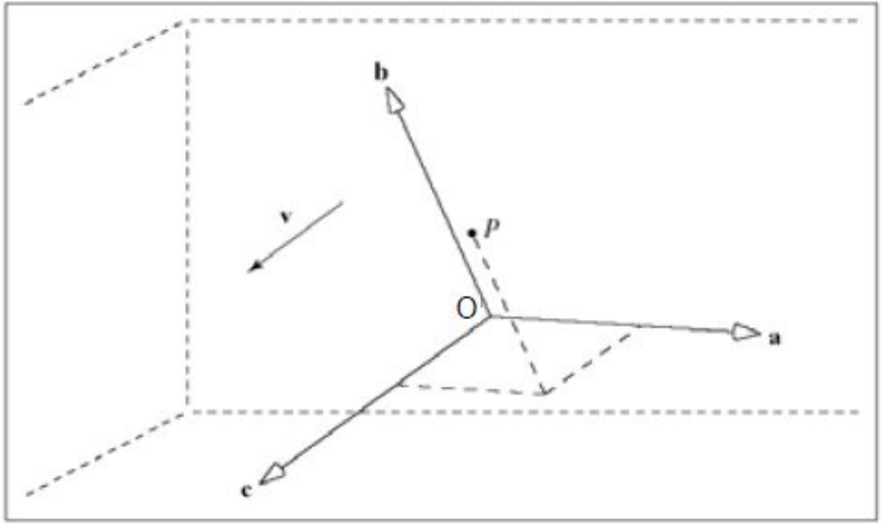
Homogeneous Coordinates

- Vectors and points are both presented as 4×1 column matrices:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$

Homogeneous Representation of Points and Vectors

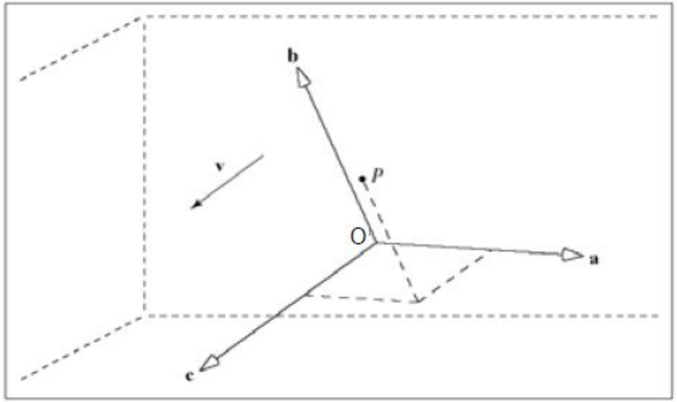
- Suppose we have a coordinate system represented by unit vectors of a,b and c as well as coordinate O for the origin



Homogeneous Representation of Points and Vectors

- Then a point $p = (p_1, p_2, p_3)$ can be represented as:

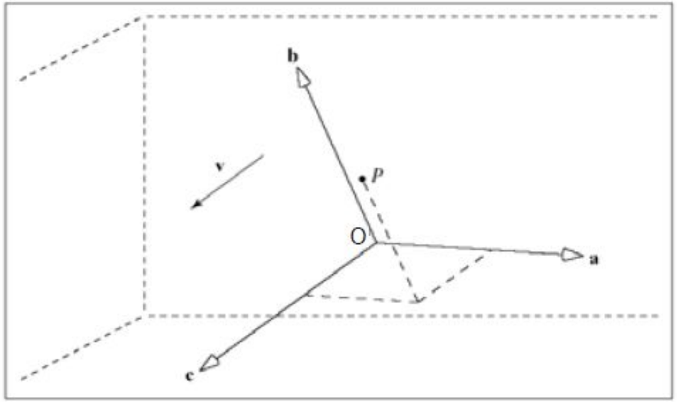
$$P = O + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$$



Homogeneous Representation of Points and Vectors

- Similarly, a vector $v = (v_1, v_2, v_3)$ can be represented as:

$$\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c}$$

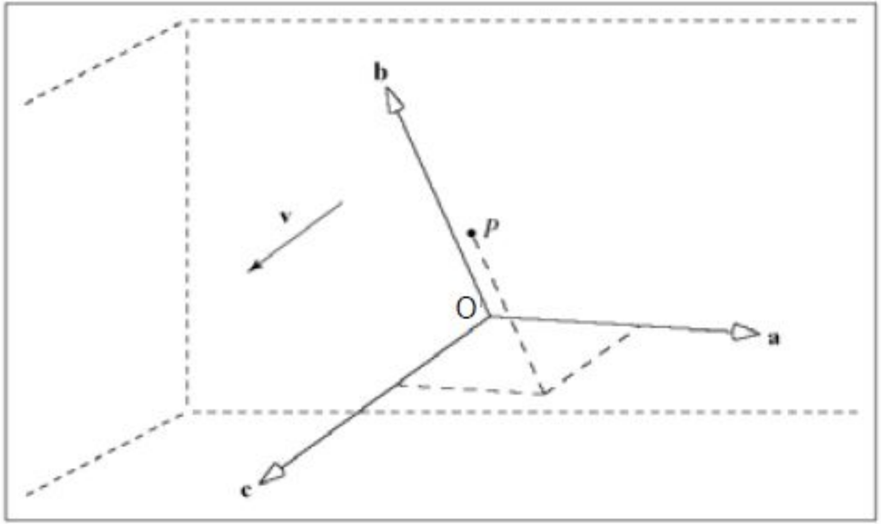


Homogeneous Representation of Points and Vectors

- In homogeneous coordinates, we can represent them as:

$$\mathbf{v} = [\mathbf{a}\ \mathbf{b}\ \mathbf{c}\ O]\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$$

$$P = [\mathbf{a}\ \mathbf{b}\ \mathbf{c}\ O]\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$



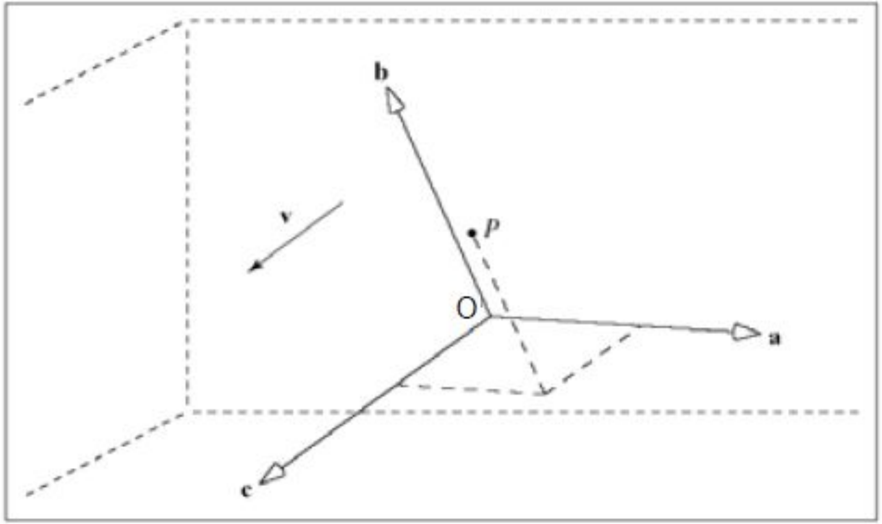
Homogeneous Representation of Points and Vectors

- Points and vectors are both represented as 4×1 column matrices. Does it make sense to just add them together ? what's the outcome ?

$$[p_1, p_2, p_3, 1]^T + [v_1, v_2, v_3, 0]^T = [p_1 + v_1,\ p_2 + v_2,\ p_3 + v_3, 1]^T$$

Homogeneous Representation of Points and Vectors

- Vector v and point P can be represented in terms of



Transformations

Transforms

- Linear Transformation: function T: $R^n \rightarrow R^m$ is a linear transform if it satisfies:

$$T(c_1\vec{u} + c_2\vec{v}) = c_1T(\vec{u}) + c_2T(\vec{v})$$

Can you spot the tow underlying conditions here ?

Transforms

- Linear Transformation: function T: $R^n \rightarrow R^m$ is a linear transform if it satisfies:

$$T(c_1\vec{u} + c_2\vec{v}) = c_1T(\vec{u}) + c_2T(\vec{v})$$

Can you spot the tow underlying conditions here ?

- T can be obviously represented by a matrix.
- Intuitively, linear transforms leave the origin untouched.

Linear Transforms

- Linear Transformation can be compactly written as matrix multiplications:

$$
\begin{aligned}
Q &= T(P) \\
&= \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix} \\
&= \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} \\
&= \mathbf{M}P
\end{aligned}
$$

Linear Transforms

- What kind of transformations can we get from the following ?

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix}$$

Linear Transforms

- How about scaling ?

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix}$$

Linear Transforms

- How about scaling ?

$$\begin{bmatrix} m_{1,1} & 0 \\ 0 & m_{2,2} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{1,1} P_x \\ m_{2,2} P_y \end{bmatrix}$$

Linear Transforms

- How about rotation ?

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix}$$

Linear Transforms

- How about rotation ?

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

Linear Transforms

- How about shearing ?

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix}$$

Linear Transforms

- How about shearing ?

$$\begin{bmatrix} 1 & -\alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

Linear Transforms

- How about translation?

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix}$$

Linear Transforms

- Let’s look at translation in more details. Translation can be formally described as:

$$Q = P + t$$

But this is not the same as :

$$Q = MP$$

Affine Transforms

- Translation is not a linear transformation.
- It's an affine transformation.
- In essence, we can represent

affine transformation = linear + translation

What are some properties of

affine transformation ?

Affine Transforms

- Properties of affine transformation:
  - Origin can be mapped to another location
  - Lines map to lines.
  - Parallel lines remain parallel

Affine Transforms

- Transforming points:

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{array}{cc} \text{Rotation/Scaling/Shearing} & \text{Translation} \\ \multicolumn{2}{c}{\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{bmatrix}} \end{array} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

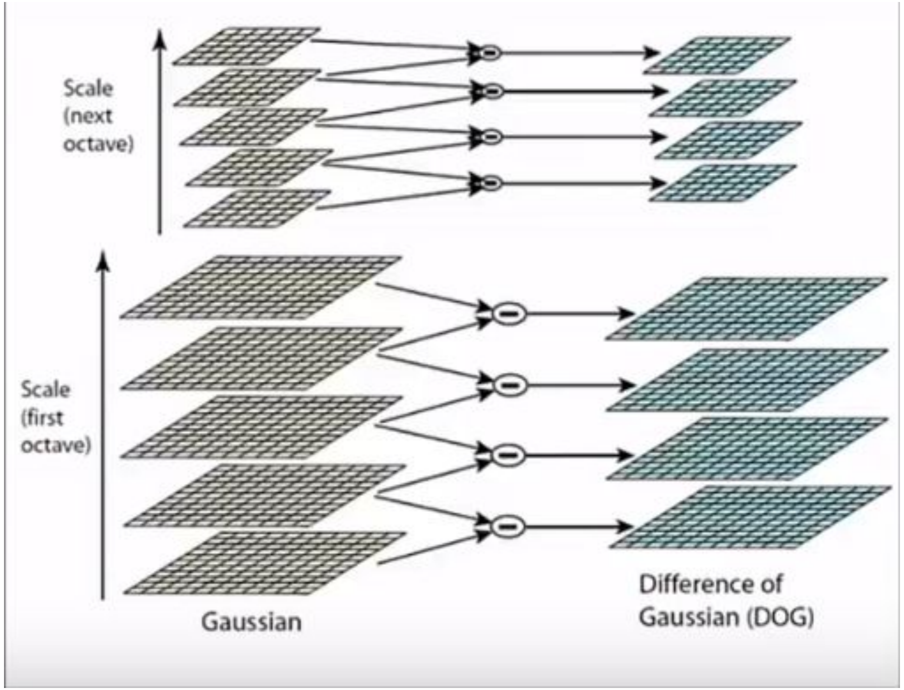
Affine Transforms

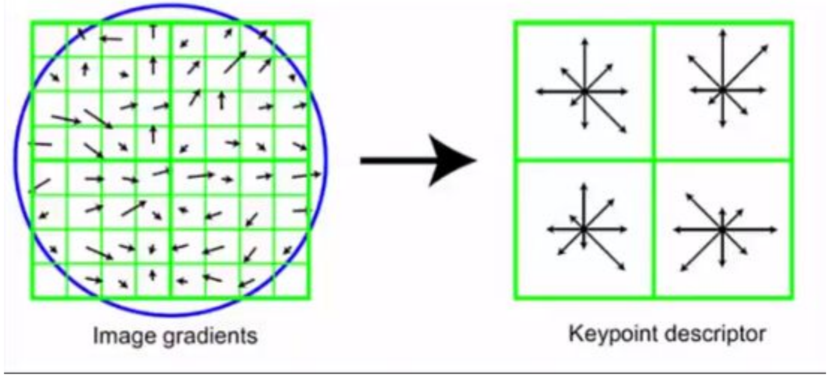
- Transforming vectors:

$$\begin{array}{cc} & \begin{array}{ccc} \text{Rotation/Scaling/Shearing} & & \text{Translation} \end{array} \\ \begin{bmatrix} W_x \\ W_y \\ 0 \end{bmatrix} = & \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ 0 \end{bmatrix} \end{array}$$

Transformations

Let’s talk about SIFT..





https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html

Let's talk about SIFT..

```
import cv2
from time import time
import numpy as np
features=[]

image = cv2.imread('./sift-scene.jpg', cv2.IMREAD_GRAYSCALE)

t1=time()
sift = cv2.xfeatures2d.SIFT_create()
kp, descriptors = sift.detectAndCompute(image, None)

for des in descriptors:
    features.append(des)

print('Compute time is {} '.format(time()-t1))

print('Shape of descriptors is {}'.format(np.shape(descriptors)))
```

Homework 1

- Try to take advantage of different packages such cv2 and sklearn to implement functionalities such as KNN, bilinear interpolation , SVM, SIFT, ORB,SURB etc.
- Remember to save results as numpy arrays
- Remember to use this version of cv2:

```
pip install opencv-python==3.4.2.16

pip install opencv-contrib-python==3.4.2.16
```

Homework 1

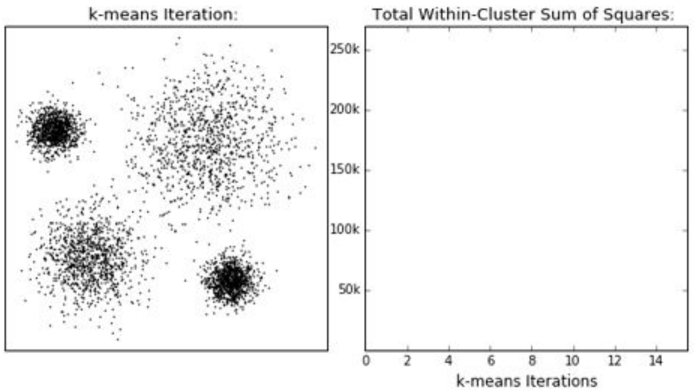
- For KNN classifier:
    - from sklearn import neighbors
    - model = neighbors.KNeighborsClassifier(n_neighbors=num_neighbors, algorithm='kd_tree', metric='euclidean')

  https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

- In task 1, you simply use a KNN with a pixel-wise distance function as the feature for scene recognition
    - It is not a good representative !
    - Since it is not invariant to viewpoint, illumination and scale.

Homework 1

- In task 2, we use local descriptors such as SIFT, ORB and SURF :
    - We use a collection of features for each image
    - We group the derived features by similarity and build a vocabulary of features
    - We use a clustering algorithm for grouping.
        - K-means (in short but you need to review in details) :
            - Input: K set of points
            - Place k centroids randomly at different locations
            - Repeat until convergence
            - For every point x_i:
                - Find the nearest centroid j
                - Assign the category to class j
            - For every cluster j:
                - Recompute their centroid
            - Stop when no cluster changes occur



Homework 1

- In task 2, the cluster centroids are the words in vocabulary
  - Example: 220 features detected, 50 words
- In task 3, now we use the BOW made in task 2 in the test set.
  - Calculate BOW
  - Use KNN to classify images (use 9 neighbors)
  - Do this process for dictionary sizes of 20 and 50.
  - Consider SIFT, ORB and SURF and K-Means and Hierarchical-Clustering
- **Hierarchical agglomerative clustering**: repeatedly combine the two nearest cluster into a larger cluster.
- In task 4, you improve the classifier and use BOW+SVM.

Thank you!