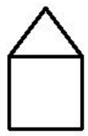
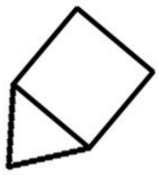
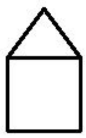
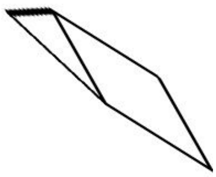
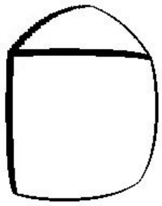
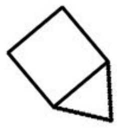
Fundamentals of Computer Vision - Final Exam	Dr. B. Nasihatkon	دانشگاه صنعتی خواجه نصیرالدین طوسی K. N. TOOSI UNIVERSITY OF TECHNOLOGY
Name:	**ID:**	**Tir 1396 - June 2017**

Image transformation (25 points)

A) Image A in the figure below undergoes different geometric transformations resulting in images B-G. Under each image B-G write down the type of the geometric transformation (translation, Euclidean, similarity, affine, perspective, or none of these). You have to write the most specific transformation (i.e. you will not get a point if you write Euclidean instead of translation) **(6 points)**

			
A	B: Similarity	C: Translation	D: Perspective
			
	E: Affine	F: None	G: Euclidean

B) Consider a point **(x,y) = (20,40)** in an image. We want to transform this point by 3 different homography matrices below. Notice that here the first and second coordinates are **x** and **y** respectively. For each matrix your final result must be a 2D vector **(x',y')** (i.e. not in homogeneous coordinates). **(9 points)**

$$\begin{bmatrix} 2 & 0.5 & -20 \\ 1 & 0 & -10 \\ 0.1 & 0.2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0.5 & 40 \\ 1 & 2 & -80 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -2 & -0.5 & 0 \\ 3 & 0.1 & 0 \\ -0.4 & 0.1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0.5 & -20 \\ 1 & 0 & -10 \\ 0.1 & 0.2 & 1 \end{bmatrix} \begin{pmatrix} 20 \\ 40 \\ 1 \end{pmatrix} = \begin{pmatrix} 40 \\ 10 \\ 11 \end{pmatrix} \Rightarrow (x', y') = (\frac{40}{11}, \frac{10}{11}) \approx (3.64, 0.91)$$

$$\begin{bmatrix} 1 & 0.5 & 40 \\ 1 & 2 & -80 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 20 \\ 40 \\ 1 \end{pmatrix} = \begin{pmatrix} 80 \\ 20 \\ 1 \end{pmatrix} \Rightarrow (x', y') = (80, 20)$$

$$\begin{bmatrix} -2 & -0.5 & 0 \\ 3 & 0.1 & 0 \\ -0.4 & 0.1 & 1 \end{bmatrix} \begin{pmatrix} 20 \\ 40 \\ 1 \end{pmatrix} = \begin{pmatrix} -60 \\ 64 \\ -3 \end{pmatrix} \Rightarrow (x', y') = (\frac{-60}{-3}, \frac{64}{-3}) = (20, -\frac{64}{3}) \approx (20, -21.33)$$

C) This question is about geometric properties preserved by different types of image transformations. Fill out the table below. **(10 points**, each wrong answer has **0.25 negative marks**, effective only for this question)

Note: self intersecting quadrilateral are still considered quadrilaterals.

	Translation	**Euclidean**	**Similarity**	**Affine**	**Perspective**
distance between a pair of points remains constant	**True**	**True**	**False**	**False**	**False**
angle between two lines remain constant	**True**	**True**	**True**	**False**	**False**
lines remain lines	**True**	**True**	**True**	**True**	**True**
angle between a line and the x-axis remains constant	**True**	**False**	**False**	**False**	**False**
quadrilaterals remain quadrilaterals	**True**	**True**	**True**	**True**	**True**
parallel lines remain parallel	**True**	**True**	**True**	**True**	**False**
circles remain circles	**True**	**True**	**True**	**False**	**False**
ratio between the areas of two shapes	**True**	**True**	**True**	**True**	**False**

stays constant					

RANSAC (15 points)

Assume we are to find an **affine** transformation which maps image **I** to image **J**. A point **x** in image **I** is mapped to its corresponding point **y** in image **J** with a map **y** = **A x** + **b**. Our task is to find **A** and **b**. We do this by finding point correspondences between the images.

A) What is the minimum number of (**correct**) point correspondences (i.e. how many pairs of points) needed to estimate the affine transform. **(3 points)**

Three matches are needed. (An affine map has 6 degrees of freedom (6 unknowns), and each pair of points provides 2 equations, thus 3 pairs of points are needed to solve for the 6 unknowns)

B) Assume that we have found a number of **putative** point correspondences in the form of (X_i, Y_i) $i = 1, 2, \ldots, M$, where Y_i is the point in the second image matched to X_i in the first image, and M is relatively large. Among these only about **40 percent are true matches**. You are to implement a RANSAC algorithm to find the correct matches and estimate the transform. Write a pseudocode describing your algorithm by completing the code in the box below. Your algorithm should be tailored for the special case of affine transformation. **(6 points)**

Input:
- **point correspondences X_i, Y_i, i = 1,2,...,M**
- **a threshold δ**

Output: the set I of inlier indices and the true affine map A,b

$I_{best} = \{\}$ **# empty set**
for iteration = 1 to N:
  j,k,l ← choose 3 random numbers from 1,2,...,M without replacement
  $A, t \leftarrow$ **estimate_affine_map({(X_j, Y_j), (X_k, Y_k), (X_l, Y_l)})**
  $I_{in} \leftarrow \{i \mid \|AX_i + t - Y_i\| < \delta,\ i \in \{1, 2, ..., M\}\}$ **# indices of inliers**
  if size of I_{in} > size of I_{best} :
    $I_{best} \leftarrow I_{in}$
$A_{best}, t_{best} \leftarrow$ **estimate_affine_map(** $\{(X_i, Y_i) \mid i \in I_{best}\}$ **)**
return $A_{best}, t_{best}, I_{best}$

C) What is the minimum number of samples N we can choose in the above algorithm when we want the algorithm succeed with a probability of 0.9999? **(6 points)**

Remember the relation between the probability of getting at least one sample with all inliers (p), the number of point matches to compute the transformation (s), and the proportion of **outliers** (e): $(1-p) > (1-(1-e)^s)^N$

$log(1-p) > N\ log(1-(1-e)^s)$

(noticing $log(1-p)$ and $log(1-(1-e)^s)$ are negative numbers)

$\Rightarrow N > log(1-p)/log(1-(1-e)^s) = log(0.0001)/log(1-(0.40)^3) \approx 139.26 \Rightarrow N_{min} = 140$

Features / SIFT (25 points)

A) Describe each of the three major stages of the SIFT algorithm. You just have to state their purpose, not the steps taken in each of them. **(6 points)**

a) SIFT detection

Detects the locations of the keypoints (points that can be identified in different images) and their associated scale.

b) SIFT description

For each keypoint finds a descriptor (a feature vector of size 128) based on how image looks like in a local neighbourhood around the keypoint. The descriptor is sought to be scale- and rotation-invariant.

c) SIFT matching

Tries to find matches between the keypoints in two (or more) images based on their descriptors.

B) Which of the above steps can be done independently for each image? Which ones require both images? **(3 points)**

a, b) can be done independently for each image.

c) requires both images

C) RANSAC can be used to improve which of the above stages? How? **(3 points)**

RANSAC can improve the matching stage. It can be applied when there is a global model relating the keypoints in one image to their true matches in the other image. RANSAC improves matching by finding this model and removing the outliers at the same time.

D) **Orientation assignment** is done in which of the above stages? What is its purpose? **(3 points)**

It can be considered as the first step in the description stage, or alternatively, last part of the detection stage. Or it can be considered as a separate stage. All these answers are acceptable. The main purpose is to make the SIFT descriptor invariant to orientation. But it has other applications as well.

E) A Kd-tree can be used in which of the stages above? What issue with the algorithm makes us consider using a Kd-tree or similar data structures? **(3 points)**

It can be used in the matching stage. The issue is that for finding the nearest neighbours of each feature point in the reference (second) image the computational complexity is similar to performing an exhaustive search, that is comparing to all points in the second image. The KD-tree gives a way of approximately finding the nearest neighbours both efficiently and effectively.

F) The matrices below show the output of the **absolute value** of the **Difference of Gaussians** operator applied on images with different bandwidths in one octave of in the SIFT algorithm. Identify the keypoint locations in the **scale-space** chosen by the SIFT algorithm, **before** removing the edges and low contrast points, and sub-pixel localization. For each keypoint report **x, y and scale.**

Note: Pixel indices start from **(x=0,y=0)**. As a reference, in the highlighted cell at scale=2, the scale-space location is (x=3, y=1, scale=2). **(7 points)**

Note: You will get **negative marks** for reporting wrong keypoint locations (only effective for this question).

Hint: The keypoints do not exist in the boundary pixels of the images, nor they can be in the first and last scales.

(x=2, y = 2, scale = 1)
(x=1, y = 2, scale = 3)

scale = 0

1	3	2	4
1	6	3	2
5	7	2	3
6	1	2	1

scale = 1

4	5	1	2
5	3	7	3
4	5	8	2
3	2	1	2

scale = 2

1	1	3	1
3	4	5	4
2	7	6	2
1	1	4	2

scale = 3

1	2	3	1
3	3	4	2
2	10	7	3
1	3	2	1

scale = 4

3	1	1	1
3	5	7	1
3	9	8	1

2	2	2	1

Generative Classification (11 points)

A) What is the difference between generative and discriminative models for classification? **(4 points)**

Generative models create a probabilistic (likelihood) model independently for each class. The classification is done based on the posterior probabilities computed from the likelihood models.

Discriminative models learn a way to discriminate between different classes, without directly building a separate model for each class. This means that is they mostly focus on modeling an explicit or implicit boundary between different classes.

B) Assume that we want to classify images in 3 categories of **A**pple**, O**range and **B**anana for which the prior probabilities are **P(A) = 0.3, P(O) = 0.5, P(B) = 0.2**. We see a new image `I` from which we extract a feature vector **x**. The likelihoods are as:

P(x | A) = 0.24
P(x | O) = 0.15
P(x | B) = 0.38

What should image `I` be classified as? An apple, an orange, or a banana? Why? Write down the complete derivations. **(7 points)**

$$P(A \mid x) = p(x \mid A)\, p(A) / p(x) = 0.24 * 0.3 / p(x) = 0.72 / p(x)$$
$$P(O \mid x) = p(x \mid O)\, p(O) / p(x) = 0.15 * 0.5 / p(x) = 0.75 / p(x)$$
$$P(B \mid x) = p(x \mid B)\, p(B) / p(x) = 0.38 * 0.2 / p(x) = 0.76 / p(x)$$

$$\Rightarrow P(B \mid x) > P(O \mid x) > P(A \mid x)$$

⇒ It has to be classified as **Banana**

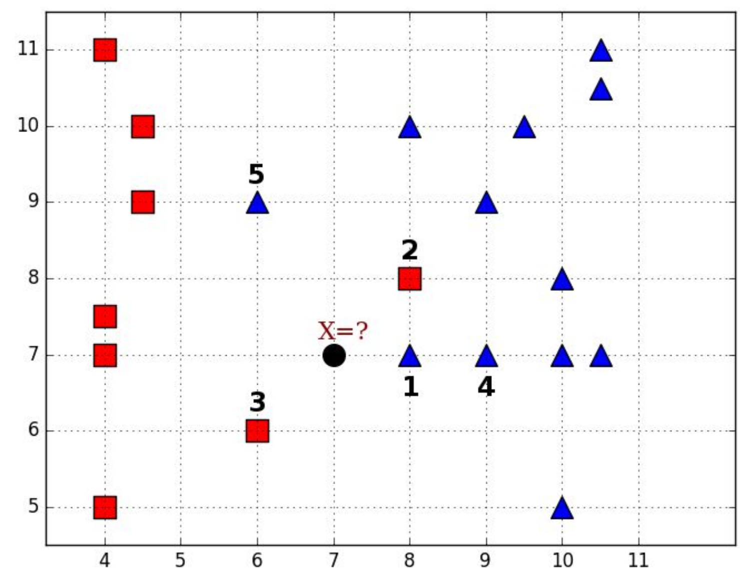
k-Nearest Neighbours Classifier (11 points)

A) Briefly describe how Nearest Neighbour (NN) and k-Nearest Neighbour (kNN) classifiers work. **(3 points)**

NN: A new sample is assigned to the class of its closest training data point in the feature space.

k-NN: For a new sample its **k** closest neighbours among the training data is found in the feature space. The assigned class is the one which is in the majority among the k neighbours.

B) The figure below shows our training data of 2D features for two classes, name them Square and Triangle. We want to classify a new data point **X** shown by a circle in the figure. Mark the first, second, 3rd, 4th and 5th nearest neighbours (write numbers 1,2,3,4,5 beside each data point). **(3 points)**



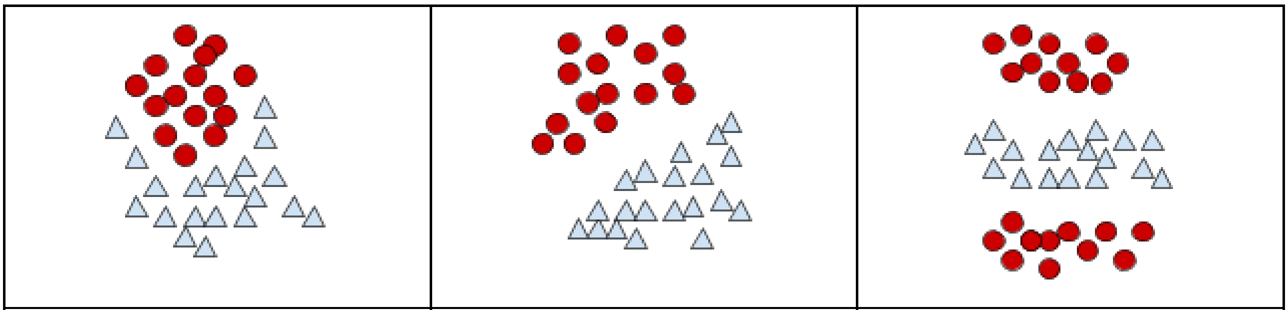
C) What is the result of classification with NN, 3-NN and 5-NN classifier? Why? **(5 points)**

T: Triangle
S: Square

The nearest neighbour is T, thus X is classified as Triangle
The 3 nearest neighbours are T,S,S (1 T and 2 S), thus X is classified as Square
The 5 nearest neighbours are T,S,S,T,T (3 T and 2 S), thus X is classified as Triangle

Support Vector Machines (SVM) (13 points)

Assume we want to classify our images into two classes **+1** and **-1**. We have extracted 3 different types of 2-dimensional features. The images below show our training data plotted in the three corresponding feature spaces.



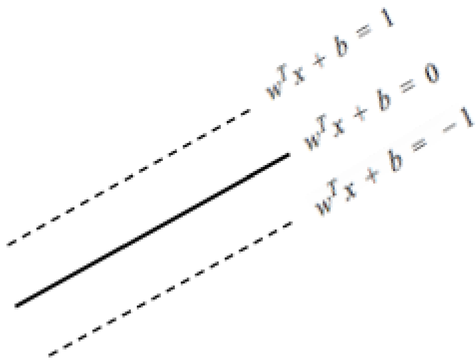
a	b	c

A) Which feature types among the above (a, b or c) are suitable to use with a linear SVM For each case explain **why or why not**. **(3 points)**
   a) No. Samples from different classes are not separable by a line.
   b) Yes. A line can partition the samples to different classes.
   c) No. No single line can split samples to different classes.

B) If we want to do SVM classification with the remaining feature type(s), what method we can use? **(3 points)**

One way is to apply a nonlinear transformation to increase the dimensionality of the feature points. This may increase the chance of the data points to be separated by a hyperplane. This can be done either explicitly, or implicitly using the kernel trick.

C) Your task in this question is to find **w** and **b** given a data set of 2D features. To do so, you need to solve a quadratic program. But, to make your job simpler, your good teacher tells you that the support vectors are **x=(-1, 2)** and **x=(-2,1)** for class **-1** and **x=(1, 3)** for class **+1**. Calculate the vector **w** and scalar **b**. Write down the derivations. (Assume that the data points are linearly separable.) **(7 points)**



Let $w^T = (u, v)$

The support vectors x=(-1, 2) and x=(-2,1) lie on the line $w^T x + b = -1$, thus:

$-u + 2v + b = -1$ and

$-2u + v + b = -1.$

The support vector x=(1,3) lies on the line $w^T x + b = 1$, thus

$u + 3v + b = 1.$

Solving these set of three equations gives $u = 2,\ v = -2,\ b = 5 \Rightarrow w = (2, -2),\ b = 5$