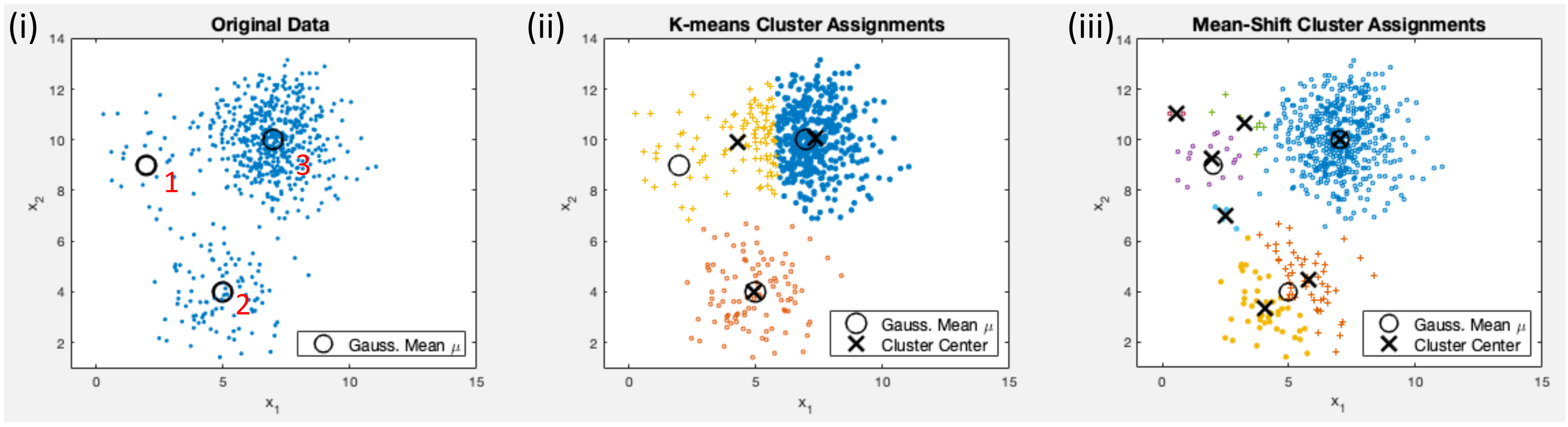
CS4243 Quiz 1

Sem 1 AY 2021/22

Question 1: Clustering	(11 points)
Question 2: Keypoint Detection & Matching	(9 points)
Question 3: Feature Invariances	(8 points)
Question 4: System Design	(12 points)
TOTAL	40 points

Q1: Clustering (11 points)



Consider the scatterplot in (i). The underlying data points are sampled from a mixture of three Gaussians $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_3$ with 25, 100 and 500 points sampled from the three respectively. The Gaussians have respective means $\boldsymbol{\mu}_1 = (2,9)$, $\boldsymbol{\mu}_2 = (5,4)$, and $\boldsymbol{\mu}_3 = (7,10)$, visualized as circles in each of the plots. Each Gaussian has the same covariance, i.e. $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}_3 = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$. Note this covariance matrix is diagonal so the Gaussians are "circular" with the same standard deviation in the x_1- and x_2-dimension.

(a) (Essay) You apply *k*-means clustering with k=3 and a random initialization and get the results of (ii). Each cluster's membership is indicated by a different marker colour and type. The cluster centers, denoted by X in (ii) approximately match to the underlying $\boldsymbol{\mu}_2$ and $\boldsymbol{\mu}_3$ but is incorrect for $\boldsymbol{\mu}_1$. Explain why this occurs and justify your solution based on the workings of the k-means algorithm.

1.5 pts

Fewer points are sampled from $\mathcal{G}_1$ compared to $\mathcal{G}_2$ and $\mathcal{G}_3$. ***(0.5 points for fewer points)***
K-means minimizes the sum of the distances from every point to the cluster center so the larger distances from the fewer points of $\mathcal{G}_1$ have less impact even if the cluster center is not correct. ***(0.5 points for considering each point equally / sum over all points, 0.5 points for wrong large distance having less impact)***

(b) (MC + Essay) Keeping all other settings and variables the same, which of the following modified $\boldsymbol{\mu}'_1$ is most likely to lead to the "correct" clustering, i.e. each cluster corresponds to one of the underlying Gaussians.

2 pts

$\boldsymbol{\mu}'_1 = \boldsymbol{\mu}_1 + (\Delta u, \Delta v)$, where $\Delta u < 0, \Delta v < 0$
$\boldsymbol{\mu}'_1 = \boldsymbol{\mu}_1 + (\Delta u, \Delta v)$, where $\Delta u > 0, \Delta v > 0$
$\boldsymbol{\mu}'_1 = \boldsymbol{\mu}_1 + (\Delta u, \Delta v)$, where $\Delta u < 0, \Delta v > 0$ ***(0.5 points)***
$\boldsymbol{\mu}'_1 = \boldsymbol{\mu}_1 + (\Delta u, \Delta v)$, where $\Delta u > 0, \Delta v < 0$
no modification to $\boldsymbol{\mu}_1$ will result in the correct clustering

Justify your answer. Are there adjustments that you can make to any of the other $\boldsymbol{\mu}$ which would also lead to the "correct" clustering? Why or why not?
We need to shift $\mathcal{G}_1$ further away from $\mathcal{G}_3$ while avoiding $\mathcal{G}_2$ ***(0.5 points)***. This will increase the distance of the points in $\mathcal{G}_1$ wrt any false cluster centers that may arise from points in $\mathcal{G}_3$ ***(0.5 points for increasing distance)***. The same effect can be achieved by shifting $\mathcal{G}_3$ further away, i.e. $\boldsymbol{\mu}'_3 = \boldsymbol{\mu}_3 + (\Delta u, \Delta v)$ where $\Delta u > 0, \Delta v > 0$ ***(0.5 points)***

(c) (Essay) Keeping all other settings and variables the same, which of the following modified $\boldsymbol{\Sigma}_1'$ is most likely to lead to the "correct" clustering?

2 pts

$\boldsymbol{\Sigma}'_1 = \boldsymbol{\Sigma}_1 + \begin{bmatrix} \Delta u & 0 \\ 0 & \Delta u \end{bmatrix}$, where $\Delta u < 0$
$\boldsymbol{\Sigma}'_1 = \boldsymbol{\Sigma}_1 + \begin{bmatrix} \Delta u & 0 \\ 0 & \Delta u \end{bmatrix}$, where $\Delta u > 0$
no modification to $\boldsymbol{\Sigma}_1$ will result in the correct clustering ***(0.5 points)***

Justify your answer; if it is not possible, explain why. Are there adjustments that you can make to any of the other $\boldsymbol{\Sigma}$ which would achieve the same effect? Why or why not?
$\Delta u > 0$ causes $\mathcal{G}_1$ to grow in extent and merge with the other Gaussians. $\Delta u < 0$ shrinks the extent of $\mathcal{G}_1$ but this does not decrease the distance wrt the false centers that may arise from points in $\mathcal{G}_3$. ***(0.5 points for not decreasing the distances).***
$\boldsymbol{\Sigma}'_3 = \boldsymbol{\Sigma}_3 + \begin{bmatrix} \Delta u & 0 \\ 0 & \Delta u \end{bmatrix}$, where $\Delta u < 0$ ***(0.5 points)*** i.e. shrinking the extent of $\mathcal{G}_3$ reduces the distance to the (correct) cluster center and encourages all points of $\mathcal{G}_3$ to belong to one cluster. ***(0.5 points reducing distance)***

Q1: Clustering

(d) (MC + Essay) Consider the following three initializations for k-means clustering:
A: all three cluster centers randomly sampled from the range of [0,1] for x1 and x2,
B: all three cluster centers randomly sampled from the range of [0,15] for x1 and x2,
C: initialization of the cluster centers on $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$, and $\boldsymbol{\mu}_3$.

2 pts

All three initializations lead to clusters similar to that shown in (ii).

What can you determine about the number of iterations required for k-means to converge for initializations A vs. B?
A is more likely to need fewer iterations than B;
B is more likely to need fewer iterations than A;
A and B will converge at approximately the same rate;
not enough information to compare the convergence of A vs. B. ***(0.5 points)***

Explain your selection above. Explain how it is possible that even when we use initialization C, k-means still leads us to the "wrong" clusters as shown in (ii).
Initialization B is more likely to distribute the initial locations throughout the actual data distribution; this is more likely to partition the data equally into the clusters results in a lower objective. Initialization A starts all cluster centers concentrated in one corner, where one is likely to end up with the bulk of points belonging to one cluster and therefore requiring extra iterations of reassigning memberships to redistribute the centers. ***(0.5 points for commenting on B, 0.5 points for commenting on A)***
Initialization C may still lead to the "wrong" clusters since cluster centers at the true means does not minimize the objective function of k-means. ***(0.5 points)***

1 pts

(e) (Essay) You apply mean-shift clustering and get the results shown in (iii). Explain why the number of resulting clusters associated to each Gaussian seems to be inversely proportional to the number of points sampled.
The many clusters suggests that the window or bandwidth is likely too **small** for the current setting. As such, the sparser the sampling, the fewer the number of points per mean shift window; when there are not enough points, the mean shift vector is less accurate or stops changing, hence more resulting clusters. ***(0.5 points for too few points in the mean shift window, 0.5 points for inaccurate / stopping mean shift vector)***

1 pts

(f) (Essay) Which parameter adjustment would you make in the mean shift algorithm to get the "correct" clustering, i.e. where each cluster corresponds to one of the underlying Gaussians? Explain why.
We need to increase the bandwidth parameter or window size *(0.5 points)* to allow for a sufficient points to be considered in the areas of $\mathcal{G}_1$ and $\mathcal{G}_2$. ***(0.5 points for more points added to mean shift window)***

1.5 pts

(g) (Essay) Keeping all other settings and variables the same, what modifications to any of the $\boldsymbol{\mu}$'s would reduce the number of clusters? Justify your answer; if it is not possible, explain why. What about the $\boldsymbol{\Sigma}$'s? Again, justify your answer; if it is not possible, explain why.
The number of clusters is due to the mis-match in bandwidth size wrt to the true density of the sampling. Changing the $\boldsymbol{\mu}$'s will not have any impact but reducing the values in the $\boldsymbol{\Sigma}$'s to shrink the extent of $\mathcal{G}_1$ and $\mathcal{G}_2$ will. ***(0.5 points μ, 0.5 points Σ, 0.5 points justification).***

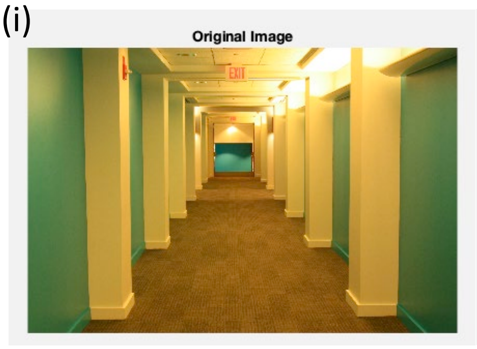
Question is ambiguous and does not state that we want the correct clustering wrt the underlying mixture of Gaussians, only the reduction in number of clusters. As such, we also accepted answers of moving the $\boldsymbol{\mu}$'s to be very close provided that this was sufficiently justified.

Q2: Keypoint Detection & Matching (9 pts)

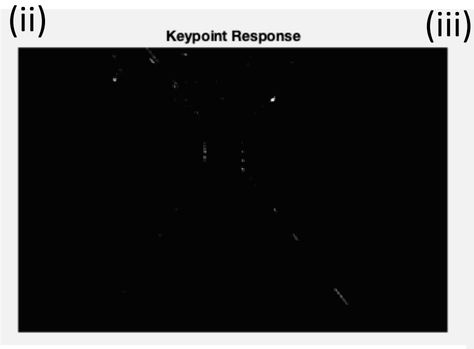
Since a corner is defined by having strong gradients in both the x- and y-direction, you design a custom keypoint detector by finding all the locations (x, y) in an image which have a sufficiently strong gradient response in both x- and y- direction.

You implement the keypoint response function by first ensuring that the absolute value of the image gradient in x- and y-direction, $|I_x|$ and$|I_y|$, surpass some threshold τ. You then sum the gradient magnitude G in a small window of $(2k +$

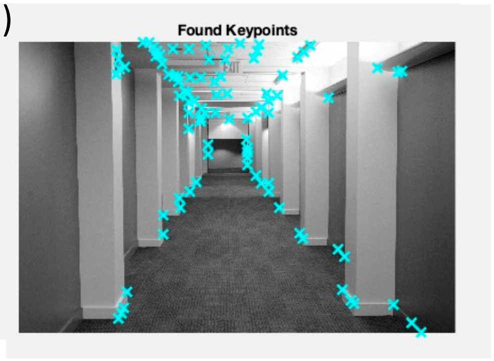
(i)



(ii)



(iii)



1 pts

(a) With the correctly tuned τ and k, will this customized keypoint detector be able to find all the "physical" corners in the hallway image, i.e. all the junctions where two straight edges meet? Why or why not?
In principle, yes. The junction points of physical corners are all characterized by strong gradients in both x- and y-direction and the above detector ensures this with thresholding. ***(0.5 points yes/no, 0.5 points justification).***

1 pts

(b) With this customized keypoint detector, where are you likely to get erroneous or false-positive corners and why do these errors occur? Would this occur with Harris corners? Why or why not?
False-positives are likely to occur on slanted edges in the image with high contrast as in these locations there are also strong gradients in both x and y. *This does not happen with Harris corners → eigenvalues of H are aligned with dominant edges* ***(0.5 points slanted edges w/ justification, 0.5 points harris w/ justification)***

1.5 pts

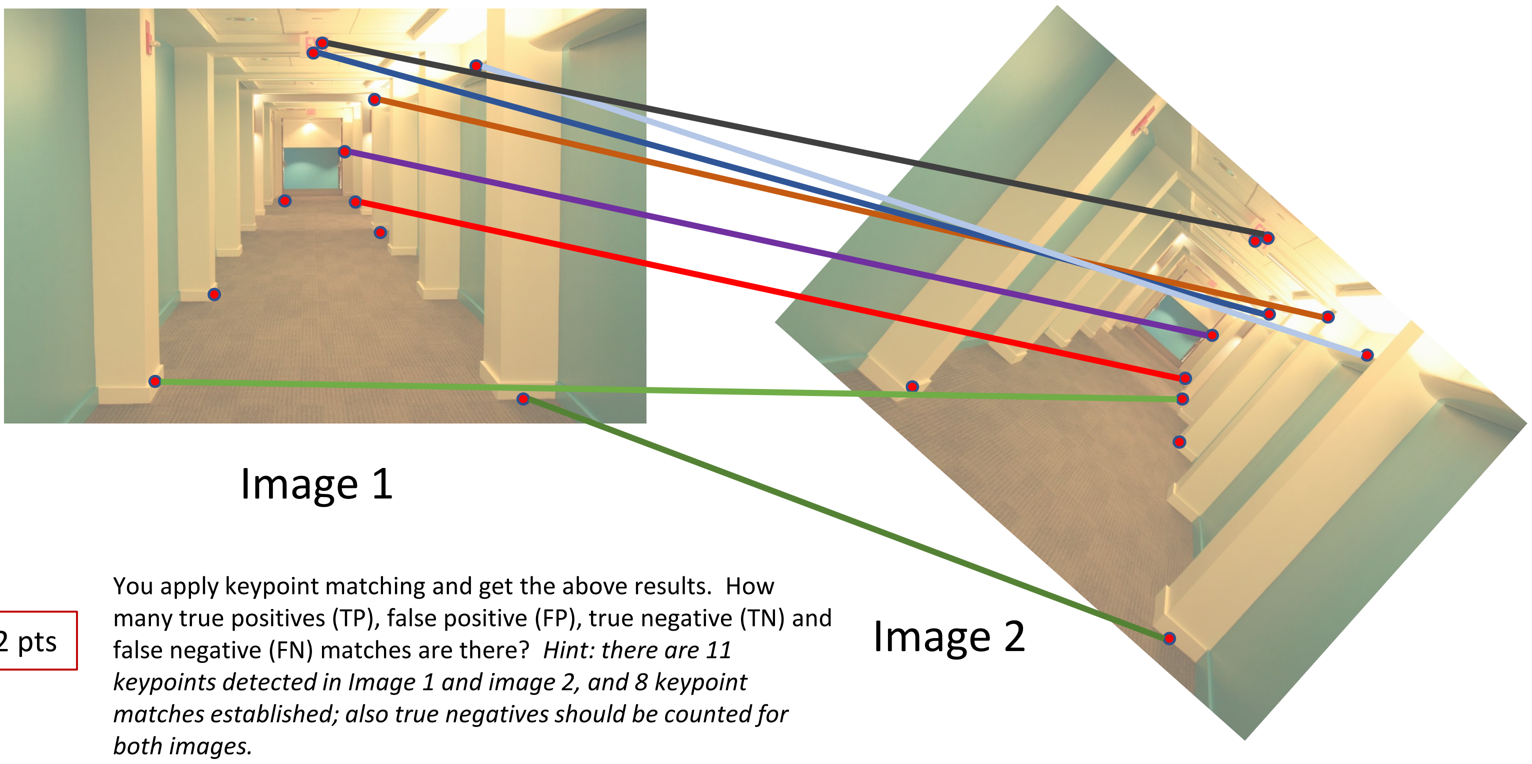
(c) Outline one modification you can make to the non-maximum suppression algorithm to reduce the number of false positive keypoints. Explain why your modification should help.
The FPs come from strong responses on slanted lines. In this case, one should adjust the non-maximum suppression to return only the two end points of any response "lines". The endpoints on the lines are indicative of true corners, while everywhere else along the response line arises from the strong gradients in x and y on the slant. ***(1 point modify NMS, 0.5 justification)***

Note: *adaptive non-maximum suppression does not allow you to reduce false-positive keypoints – it simply allows you to find more keypoints in areas of lower contrast that would otherwise not be considered because of the lower scores*

1.5 pts

(d) You are not satisfied with the found corners and decide to sharpen the input image before passing it through your detector. Where are you most likely to get false positives which currently do not occur in the image? Justify your solution. Would this occur with Harris corners? Why or why not?
Sharpening emphasizes gradients of edges; we will get false positives in any region with lots of strong random gradients since we are summing over some window. This is most problematic for the carpet. The same problem will occur with Harris corners as they are also susceptible to noise. ***(0.5 points carpet, 0.5 points justification, 0.5 points harris w/ justification).***

Q2: Keypoint Detection & Matching (cont'd)



2 pts

You apply keypoint matching and get the above results. How many true positives (TP), false positive (FP), true negative (TN) and false negative (FN) matches are there? *Hint: there are 11 keypoints detected in Image 1 and image 2, and 8 keypoint matches established; also true negatives should be counted for both images.*

Fill in blank: TP _, FP_, TN_, FN _. **(0.5 pts each)**

You are unhappy with the matches and decide to try again with a higher distance ratio threshold. What do you expect to happen to the number of TPs, FPs, TNs and FNs?
(MC: increase, stay the same or increase, decrease, stay the same or decrease, not enough information) ***(0.5 pts each)***

2 pts

TP : stay the same or increase
FP : stay the same or increase
TN : stay the same or decrease
FN : stay the same or decrease

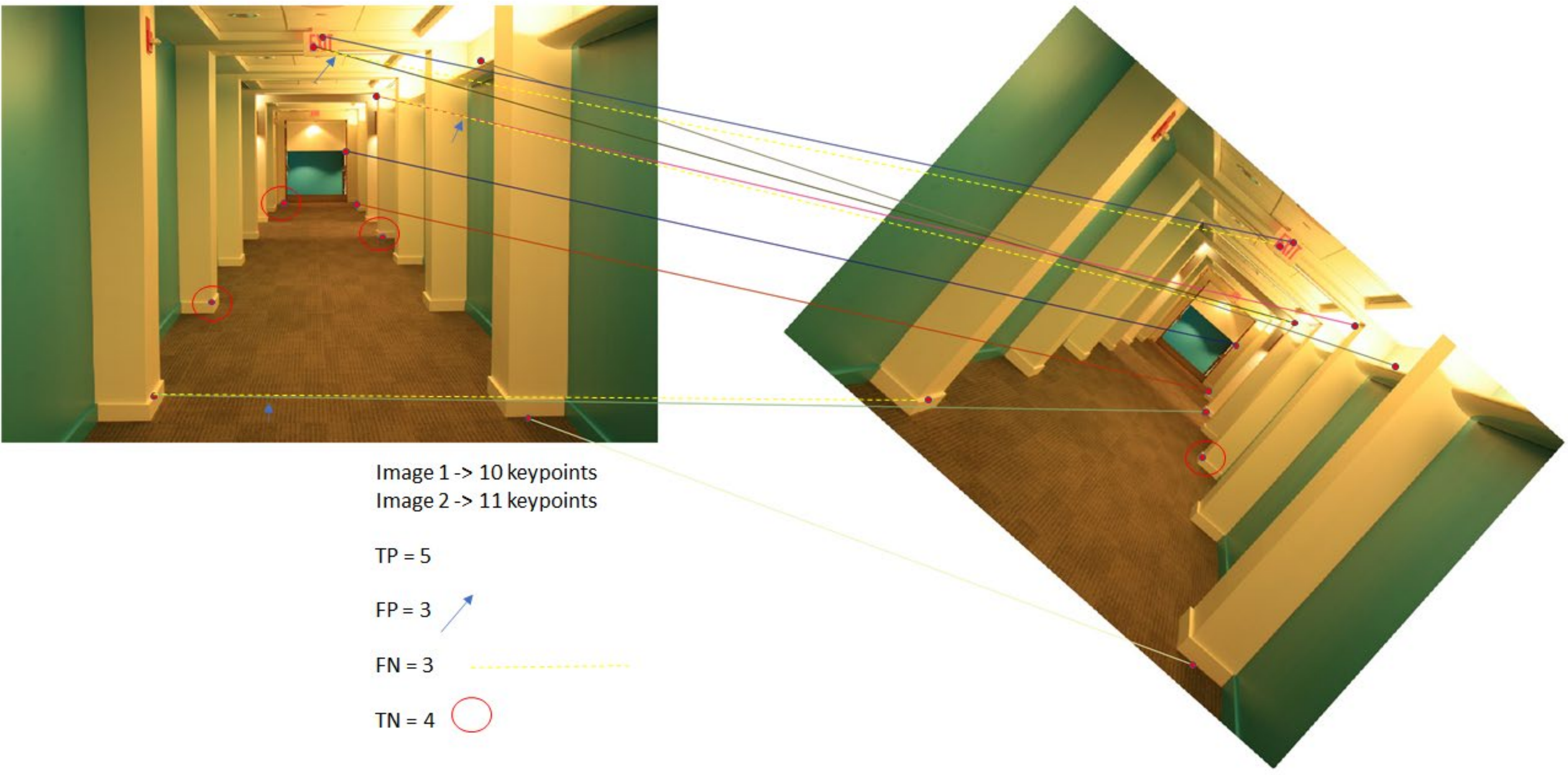


Image 1 -> 10 keypoints
Image 2 -> 11 keypoints

TP = 5

FP = 3 ↗

FN = 3 - - - - - - -

TN = 4 ○

Q3: Feature Invariances (8 points)

Consider four keypoint detectors:

- the custom designed one as described in Q3
- a Harris corner detector
- a Laplacian of Gaussians with a fixed σ
- the SIFT keypoint detector.

Additionally, consider four descriptors:

- Weighted intensity: a local descriptor that sums a circular Gaussian-weighting of the greyscale intensities in a 5x5 window around the keypoint
- Weighted gradients: a local descriptor that sums a circular Gaussian-weighting of the gradient magnitudes in a 5x5 window around the keypoint
- the GIST descriptor
- the SIFT keypoint descriptor.

For a given image $I(x, y)$ and a transformed version $I'(x, y)$, select the relevant keypoint detector and descriptors for which the statement is true. You may disregard any quantization effects, i.e. assume that there is no intensity clipping and suitable interpolation and rounding operations are applied.

(*multiple selection)*

The keypoint locations after transformation are equivariant:
Custom, Harris Corners, Laplacian of Gaussian, SIFT.

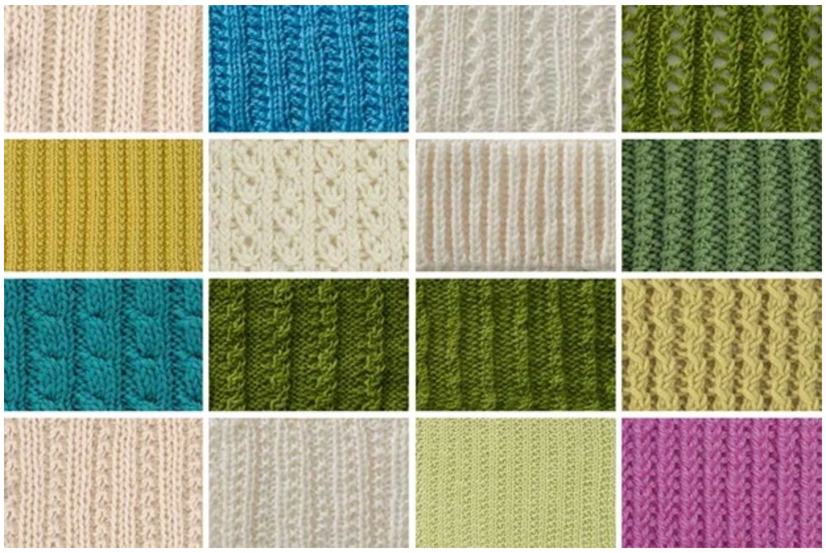
The keypoint descriptors after transformation are invariant:
weighted intensity, weighted gradients, GIST, SIFT

- Transformation: $I'(x, y) = I(x + \Delta u, y + \Delta v)$ where $\Delta u > 0, \Delta v > 0$
  custom, Harris, LoG, SIFT (all are invariant to position offsets)
  weighted intensity, weighted gradients, GIST, SIFT (all except GIST)
- Transformation: $I'(x, y) = I(x, y) + \Delta u$, where $\Delta u > 0$
  custom, Harris, LoG, SIFT (all are invariant to intensity pffset)
  weighted intensity, weighted gradients, GIST, SIFT (all except weighted intensity)
- Transformation: $I'(x, y)$ is a mirrored version of $I(x, y)$, i.e. flipped over the X axis.
  custom, Harris, LoG, SIFT (all are invariant to flipping)
  weighted intensity, weighted gradients, GIST, SIFT (GIST and SIFT are not mirror invariant)
- Transformation: $I'(x, y) = I(y, x)$
  custom, Harris, LoG, SIFT (all are invariant to transpose)
  weighted intensity, weighted gradients, GIST, SIFT (GIST and SIFT descriptors are not invariant)
- Transformation: $I'(x, y) = I(0.5 * x, y)$
  custom, Harris, LoG, SIFT (none)
  weighted intensity, weighted gradients, GIST, SIFT (none)
- Transformation: $I'(x, y) = I(0.5 * x, 0.5 * y)$
  custom, Harris, LoG, **SIFT** (scaling transformation)
  weighted intensity, weighted gradients, GIST, **SIFT** (only SIFT)
- Transformation: $I'(x, y)$ is a histogram stretched version of to $I(x, y)$.
  custom, Harris, LoG, SIFT (none are invariant)
  weighted intensity, weighted gradients, GIST, SIFT (none)
- Transformation: $I'(x, y)$ is 45 degree clock-wise rotated version of $I(x, y)$.
  custom, **Harris, LoG, SIFT** (all except custom)
  weighted intensity, weighted gradients, GIST, SIFT (all except GIST)

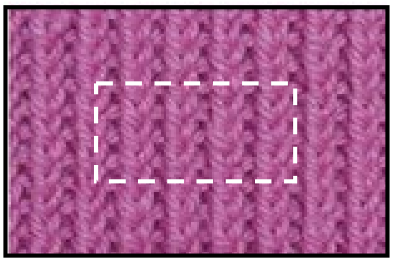
8 x (0.5 MC on detector, 0.5MC on descriptor)

Q4: System Design - Manufacturing of Knits (12 points)

(i)



(ii)



You are hired to design a visual inspection system for a textile factory that manufactures knits.

2 pts

(a) The inspection system is specified to classify the type of knitting pattern. Samples are shown in (i). You decide to design a filter bank with a series of Gabor filters based on the following definition:

$$f_{mn} = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{m^2+n^2}{2\sigma^2}\right] \sin\left[\frac{2\pi(\cos[\omega]m + \sin[\omega]n)}{\lambda} + \phi\right]$$

As you want to keep the system efficient, you want to minimize the number of filters in your filter bank. Which of the Gabor filter parameter(s) should you keep constant and which parameter(s) should you allow for more variations? Justify your answer.

MC: $(\sigma, \omega, \lambda, \phi)$ / open-form for reason
Keep constant: ω or ϕ *(1/2 mark each)*
Many variants: λ or σ *(1/2 mark each)*

Reason: these knits feature predominantly vertical patterns of varying thickness. Therefore, it is okay to keep ω at a constant value to be oriented to vertical patterns. ϕ is the translation offset; since we convolve the kernel, it simply shifts the maximum to occur at the same offset; will not have any effect. We want to vary λ to adjust the wavelength to distinguish the different vertical thicknesses. As λ often is set to be proportional to σ, we will accept σ as a solution if this is explicitly mentioned in the explanation. *(0.5 for each explanation)*

2 pts

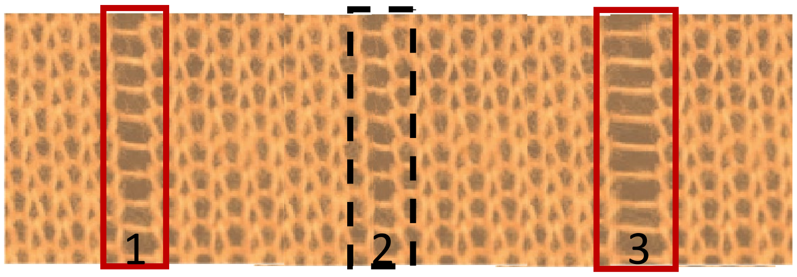
(b) You decide to use textons to classify the knitting pattern. You establish a database of reference samples of 600x800 pixel image patches and learn a texton dictionary to perform nearest neighbor classification based on the normalized texton histogram. Before you can demo the system to your boss, a colleague inadvertently changes the setting so that collected test samples are center-crops of 300x400 pixels. You can see an example of the original setting outlined by black and the new setting outlined by white dashed lines in (ii). Detail the cases in which your system will still work (if any) and the cases in which it will not (if any) and justify your solution.

The system will still work, because the matching is done via a normalized texton histogram – the relative counts of the textons in the crop will not change. *(1 mark for correct answer & justification; 0.5 marks for wrong answer but plausible justification)*

Exceptional case is if the scale of the overall texture of the knit pattern exceeds the 300x400 pixels s.t. it can no longer be reliably captured. The closest example is the top left image, if the vertical bars are even thicker that we can no longer capture the pattern within the window, i.e .the underlying texture frequencies are too low. *(1 mark for correct answer & justification; 0.5 marks for wrong answer but plausible justification;)*

Q4: System Design - Manufacturing of Knits (cont'd)

(iii)



Rectangular filter examples

(c) In a haste to fix your system, you opt to re-scale the cropped test samples by a factor of two to get back 600x800 image patches. Detail the cases in which your system will still work (if any) and the cases in which it will not (if any) and justify your solution.

1 pts

Re-scaling the crops by a factor of 2 completely changes the scale of the texture. As the texton dictionary is not learned at this new scale, the system will break down completely. ***(1 point for correct answer and justification, no points for guessing simply "no" without justification)***

(d) Due to broken needles, the stitching pattern is uneven and results in columns of holes, as highlighted by the boxes in (ii). The holed columns occur randomly and vary in width. If the width of the holed column does not exceed some threshold, then it is not considered defective, e.g. column 2 marked by the dashed box is not a defect, while columns 1 and 3 are defects.

Outline a method using SLIC to find *all* the holed columns and count the number of defective columns. State any assumptions you make and outline possible draw-back(s) of your method.

2.5 pts

We apply SLIC to segment the sample into superpixels, where yarn form some superpixels, holes form others. Based on the average RGB value of each superpixel, we can identify the hole-based superpixels and based on the superpixel centers, match columns of holes. From these columns, count number of pixels within each hole column; if this crosses some threshold, assume that this is a defective column. ***(0.5 pts for identifying "hole" superpixels, 0.5 points for finding columns of holes, 0.5 points for exceeding width; any reasonable method for accomplishing the above will suffice)***

Assumption: Yarn and the holes in between can be cleanly separated into individual superpixels. ***(0.5 points for reasonable assumption related to method)***

Drawbacks: the holes in the holed columns are of a different shape than the holes in the regular stitches; the above-described method does not identify shape of superpixel, and there may be the case where the holed column is narrower in width than the normal columns. ***(0.5 points for reasonable drawback)***

(e) Consider as input parameters to the SLIC algorithm N, the number of superpixels, and c, a compactness parameter where the higher the value, the more regular the superpixels' shape. Outline how adjusting i.e. increasing or decreasing, N and c will affect the performance of your system in (d).

2 pts

N is too small, insufficient to separate the yarn and holes into distinct superpixels. N is too big, may separate the hole superpixels esp. the large ones into separate superpixels. In both cases, can no longer identify the columns correctly and or accurately estimate the width. ***(0.5 point for effect, 0.5 points impact).***

C is too small, very irregularly shaped superpixels; oversegments the holes; C too large, holes get merged with the yarn, leads to inaccurate estimate of the width. ***(0.5 point for effect, 0.5 points impact).***

Q4: System Design - Manufacturing of Knits (cont'd)

2.5 pts

- (f) Outline any alternative method to find *all* the holed columns and count the number of defective columns. State any assumptions you make and outline possible draw-back(s) of your method.

  Method 1: Learn a filter bank to perform segmentation. Since we are interested in identifying specific columns, can consider using non-square filters as studied in class, but long rectangular filters tuned specifically to the regular vs/ "holed" filter responses (see above). Take various filter outputs as the feature response and segment hole vs. non-holed columns to perform segmentation. Count contiguous "hole" columns / check exceed width.

  Assumption: can find filter kernels for filter bank which can sufficiently distinguish the patterns between regular vs. hole pixels; drawback: the responses may not be precise enough to localize around the edges of the hole columns and may not be so accurate.

  Same marking scheme as (d): (0.5 pts for identifying "hole" superpixels, 0.5 points for finding columns of holes, 0.5 points for exceeding width; any reasonable method for accomplishing the above will suffice) (0.5 points for reasonable assumption, 0.5 pts drawback)

  ** note, using texton dictionary + histogram is not a reasonable method together with segmentation → we discussed in lecture how histograms are poor at localization.

Descriptor

You design a new local descriptor in which you form histograms as follows