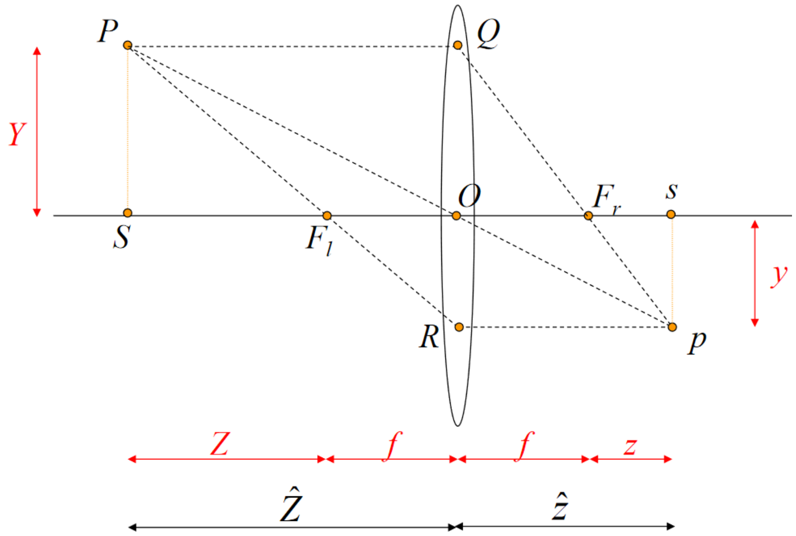
Nanyang Technological University, Singapore Autumn 2017

Notes for

CZ4003 Computer Vision

Written by Nicklas Hansen

Thin Lens Fundamental Equations



$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{f}$$

$$\frac{Y}{Z} = \frac{y}{z}$$

Small Aperture – increases depth of field, but less light -> longer exposure time

Pinhole Camera – infinitesimal aperture -> only one ray from each scene point -> lens not needed

Relationship between a point in world frame and a point image frame:

$$\begin{bmatrix} kx_{im} \\ ky_{im} \\ k \end{bmatrix} = \underbrace{\begin{bmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{M_{\text{int}}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & T_X \\ r_{21} & r_{22} & r_{23} & T_Y \\ r_{31} & r_{32} & r_{33} & T_Z \end{bmatrix}}_{M_{ext}} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Examples of intrinsic parameters M_{int} are camera properties, e.g. focal length and sensor size, and extrinsic parameters are translation and rotation vectors from world frame to image frame.

Blur – occurs when object is in front or behind the focal plane; blur can be negated by low aperture

diameter of aperture

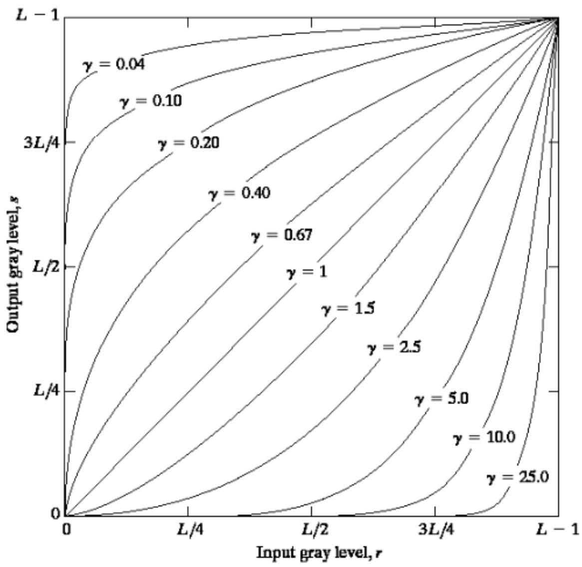
diameter of blur $b = \frac{df}{\hat{Z}}$ focal length

distance of object point

Image Negative – for image with gray-level in $[0,255]$ the transform is $s = 255 - r$ (r = intensity)

Contrast Stretching – $s = \frac{255\,(r - r_{min})}{r_{max} - r_{min}}$ for $r_{min} < r < r_{max}$, otherwise 0 or 255

Power-Law Transform – $s = c \cdot r^{\gamma}$ where c, γ are constants



1	2
3	4

Figure Q1b

2	8
18	32

Figure Q1c

Histogram Equalization – only approx. flat (not perfectly), approach is to shift gray-level bins

Reapplying the equalization does not render new results, as algorithm already produces best result

$$s_k = T(k) = \frac{255}{MN} \sum_{j=0}^{k} n_j \text{ for } k \text{ bins}$$

Spatial Filtering (Weighted Sum)

$w =$

-0.3	-0.2	0.1
-0.3	-0.4	0.1
0.4	0.3	0.3

As a correlation:

$$g(x, y) = \sum_{u=-a}^{a} \sum_{v-b}^{b} f(x+u, y+v) \cdot w(u, v)$$

As a convolution:

(kernel w is rotated)

-0.3	-0.2	0.1
-0.3	-0.4	0.1
0.4	0.3	0.3

0.3	0.3	0.4
0.1	-0.4	-0.3
0.1	-0.2	-0.3

$h(u, v) = w(-u, -v)$

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t-b}^{b} f(x-s, y-t) \cdot h(s,t)$$

Difference between correlation and convolution

Rotation makes no difference on a symmetric (e.g. gaussian) filter, but usually not the case

Convolution is associative, while correlation is not -> filters can be pre-computed -> computed fast

Averaging Filter (box) – different filter sizes can be used, output is average of neighbouring pixels

Gaussian Smoothing – 2D normal density function, variable filter sizes; further away -> less weight

$$h(x,y) = \frac{1}{2\,\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\,\sigma^2}\right)$$

Sum of filter coefficients should be normalised to 1

Median Filter – nonlinear, cannot be represented as convolution, useful for removing speckle noise

Frequency Domain – split signals into multiple layers, i.e. signal layer + interference layer (noise)

Low-pass filtering: image will blur, high-pass filtering: finding edges, band-pass: allows specific freqs.

Convolution Theorem – convolution of two spatial images equiv. to multiplying corresponding FTs

$$f(x,y) \otimes h(x,y) = IFT\left\{FT\left[f(x,y)\right] FT\left[h(x,y)\right]\right\}$$

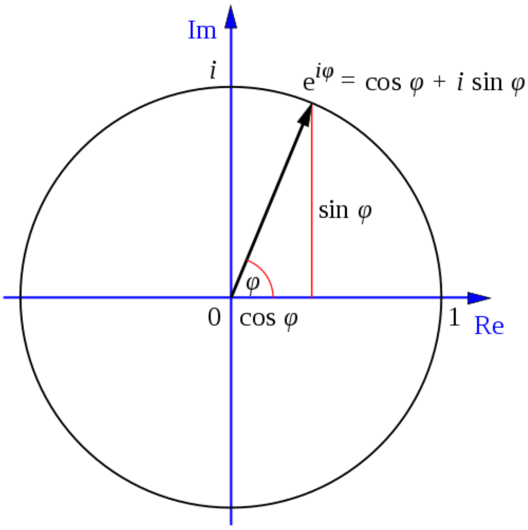
Fourier Transform – equation as below for function $f(x,y)$

$$F(u,v) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} f(x,y) \exp\left[-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)\right]$$

In case of $F(0,0)$, the exponent will be equal 0 and the transform is thus just a summation; $F(0,0)$ describes the lowest frequency in the Fourier spectrum and its value is the sum of all intensities of all pixels in the image

Relation between exp and cos/sin

$$\exp\left[j2\pi\left(\frac{xu}{M} + \frac{yv}{N}\right)\right] = \cos\left[2\pi\left(\frac{xu}{M} + \frac{yv}{N}\right)\right] + j\sin\left[2\pi\left(\frac{xu}{M} + \frac{yv}{N}\right)\right]$$



Example:

1	0
0	1

Figure Q2a

???

$$F(1,1) = 0 + 0 + 0 + 1 \cdot \exp\left[-j2\pi\left(\frac{1 \cdot 1}{2} + \frac{1 \cdot 1}{2}\right)\right] = \exp(-j2\pi \cdot 2)$$

Edgels – pixel locations with locally maximum gray-level gradient in one direction and small change in perpendicular direction

Ridges – located between two parallel, near and sharp gradients in opposite directions

Binary Image – edges marked 1, otherwise 0 -> no explicit continuity information; can be solved using other representations: linked edges, polygonal lines, parametric curves

Path Encoding (Chain Codes) – each pixel is encoded as relative direction to the previous pixel, requires only 2-bit or 3-bit number; drawback is that it is expensive to access edgel coordinates

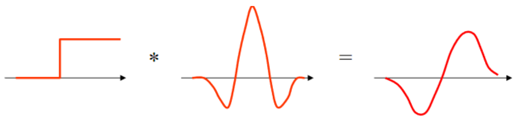
Edge Detection – consists of three steps: edge filtering (generate gray-scale image where pixels contain estimates of gradient magnitudes; edgel detection: binary image containing edgels from edge filtered image; edgel linking: produces a set of edges consisting of connected edgels

Edge Filters (Sobel) – estimate horizontal and vertical gradient magnitudes G_x and G_y separately

$$\nabla f \approx |G_x| + |G_y|$$

Edge Filters (Laplacian of Gaussian) – emphasizes curvature; edgels detected from zero-crossings; filter is less sensitive to noise because of Gaussian smoothing; adjustable σ for different thresholds

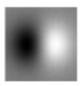
$$f \cdot \nabla^2 h = \nabla^2 (f \cdot h)$$



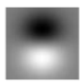
Canny Edge Detector – commonly used advanced algorithm; based on gaussian edge filtering; best

Gaussian Edge Filtering – input image is filtered twice by x and y derivatives of Gaussian

$$h_x(x,y) = \frac{\partial h}{\partial x} = -\frac{x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

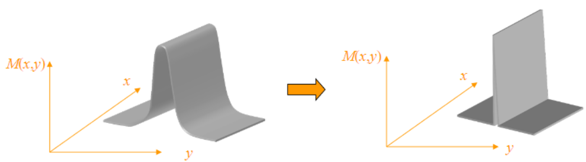


$$h_y(x,y) = \frac{\partial h}{\partial y} = -\frac{y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$G_x = f * h_x, \quad G_y = f * h_y$$

Non-maximal Suppression – reduce broad filtered edges to single-pixel-wide paths; set all gradient magnitudes to zero if they are not local maxima



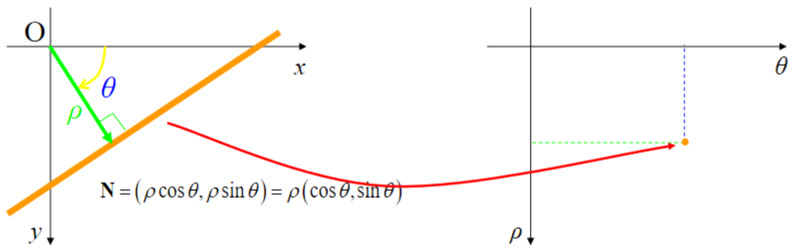
Hysteresis Thresholding – uses high threshold t_H and low threshold t_L + condition: a pixel has neighbouring pixels perpendicular to the edge gradient which have already been set to 1

Removes tiny noisy edges from the image

$$f_t = \begin{cases} 1, & f \geq t_H \\ 0, & t_L \leq f < t_H \ \& \ Condition \end{cases}$$

Hough Transform – useful for grouping straight image edges that belong to same physical edge in the world, but are disconnected e.g. due to noise or occlusion

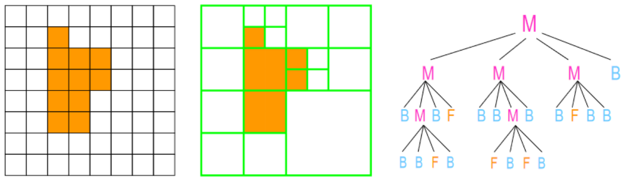
Line Equation – $x\cos\theta + y\sin\theta = \rho$ -> (ρ, θ) equals a unique line in the image



Take all edge points in image and transform them into sinusoidal curves in parameter space (ρ, θ); edge points having common point in parameter space lie on a common straight line in image -> points with high intensity means the corresponding line in image has strong edge support

Image Regions (Labelled Mask Overlay) – representation of image regions by mapping label values to different colours to display as overlay on image

Quadtree – recursively divides a square region into 4 quadrant sub-regions and codes them in a tree structure; division stops when all pixels in a sub-region has common label



Region Segmentation – involves dividing an image into separate homogeneous (uniform) regions (regions with pixels sharing similar properties, e.g. intensity, variance of neighbourhood)

Gray-Level Thresholding – simplest method; divides into two regions based on intensity

$$s = \begin{cases} 0, & r \le t \\ 1, & r > t \end{cases}$$

For automatic thresholding, t can be set to average pixel value in image, but Otsu is better

Otsu Method – minimize the pixel gray-level variance within each group; find threshold t

$$SS_W = \sum_{j=i}^{k} \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2$$

$$\sigma_L^2(t) = \frac{1}{q_L(t)} \sum_{r=0}^{t} (r - \mu_L(t))^2 p(r), \quad \sigma_H^2(t) = \frac{1}{q_H(t)} \sum_{r=t+1}^{255} (r - \mu_H(t))^2 p(r)$$

Application of Otsu is to compute $\sigma_w^2(t)$ for $t = [1; 254]$ and select the t with smallest σ_w^2

Texture Detection – hard to define, but has two approaches: structural or statistical

Structural – a texture is a set of texture elements (texels) repeated in spatial relationship; hard to automate -> what is a texel? Natural textures with inherent randomness not covered

Statistical (Co-occurrence Matrices) – matrix C_d is to discover distributions of relative positions of pixels; how often to pixels with same intensity appear together in a particular spatial relationship in an image patch?
$C(1,0)$ means co-occurrence 1 down, while $C(0,1)$ means 1 right

127	127	0	0
127	127	0	0
0	0	255	255
0	0	255	255

$\mathbf{C}_{(1,0)}$

	0	127	255
0	4	0	2
127	2	2	0
255	0	0	2

$\mathbf{C}_{(0,1)}$

	0	127	255
0	4	0	2
127	2	2	0
255	0	0	2

$\mathbf{C}_{(1,1)}$

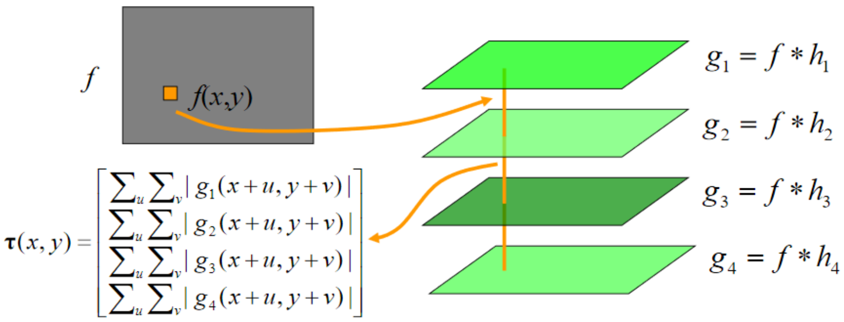
	0	127	255
0	2	0	2
127	2	1	1
255	0	0	1

To compare co-occurrence matrices for image patches of different sizes, they are normalised

$$\mathbf{N}_{\mathbf{d}}(a,b) = \frac{\mathbf{C}_{\mathbf{d}}(a,b)}{\sum_r \sum_c \mathbf{C}_{\mathbf{d}}(a,b)}$$

After normalisation, $0 < N_d(a,b) \leq 1$ for all a and b

Statistical (Filter Banks) – bank of sinusoidal filters with gaussian window; after convolution with a given filter bank, the mean absolute value of the neighbourhood around each filtered output pixel is computed; corresponding mean absolute values across the filter outputs can be concatenated to form a texture feature vector for that pixel location



K-means Clustering – useful when there's more than two groups; each pixel is associated with a feature vector; goal of clustering is to classify feature vectors that are close to each other in feature vector space in the same group and separate those that are far apart; algorithm repeats two steps:

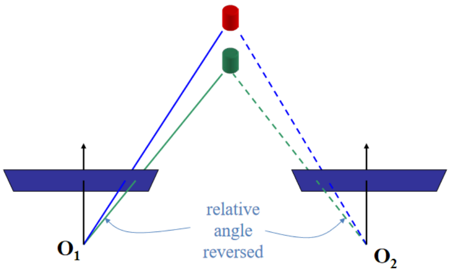
Assignment Step – assign each observation to nearest cluster

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \ \forall j, 1 \leq j \leq k\}$$

Update Step – calculate new means for all clusters

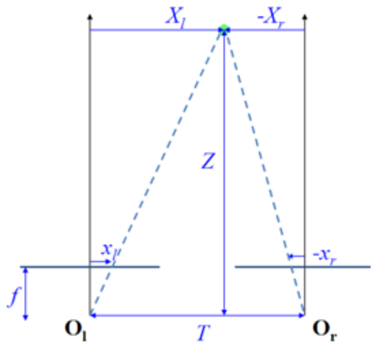
$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Stereo Vision – change in relative angular displacement of img points across different camera views



Camera Perspectives – assume 1D camera translated by T along x-direction in cam frame

$x_l = f\frac{X_l}{Z}$ $x_r = f\frac{X_r}{Z}$, $x_l - x_r = f\frac{X_l - X_r}{Z} = \frac{fT}{Z}$, depth: $Z = \frac{fT}{x_l - x_r} = \frac{fT}{d}$ where d = disparity



Triangulation Accuracy – if $T \to 0$, then $\delta Z \to \infty$ (cameras should not be too close)
$Z \to \infty, \delta Z \to \infty$ (lower accuracy for further points in world)

Correspondence Problem – for each important image point in one image, find the corresponding point in a second image; appearance-based: match points with similar look (can be used if cameras close); feature-based: match similar features (edges, corners, etc.)

Sum-of-Squares Difference (SSD) – $\arg\min_{(x,y)} \sum_{u=-N}^{N} \sum_{v=-N}^{N} [I(x+u, y+v) - g(u,v)]^2$
For small image patch g with center location (x, y), the image patch with smallest SSD in relation to any image patch in the second image is the best appearance-based matching; can in practice be computed using convolution

Trade-off in applications – appearance matching most accurate when cameras close, 3D estimation is most accurate when cameras are far apart, so something in-between is used

Feature-based Matching – relatively invariant to viewpoint; e.g. match corner points in two images with the same angle; corner points can have very different orientation; in practice, there may be multiple candidates to a given feature – a heuristic can be used to differentiate, but does not always guarantee the right solution

Proximity – match feature point to candidate with nearest (x, y) position in image

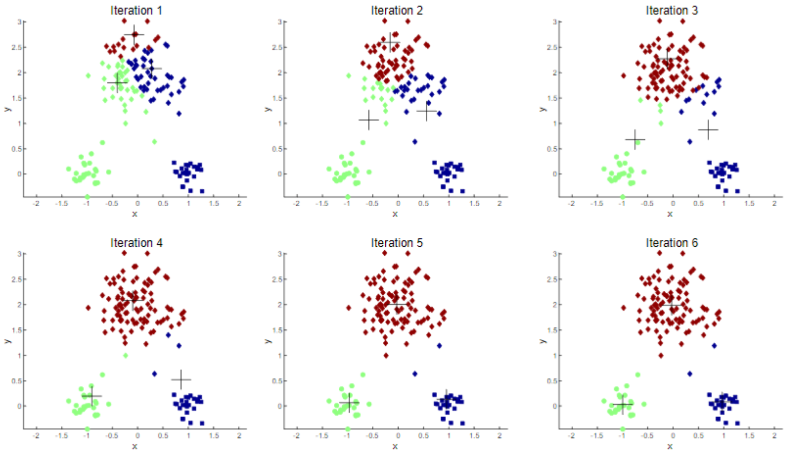
Ordering – feature points found on a contour in the first image must still be on a contour in same order in the second image, otherwise not correct candidate

3D Reconstruction – assume projection matrices from camera calibration are known; then 3D world coordinates can be reconstructed from a L and R camera, where $[X\ Y\ Z]^T$ is the unknowns

$$\underbrace{\begin{bmatrix} a_9 x_l - a_1 & a_{10} x_l - a_2 & a_{11} x_l - a_3 \\ a_9 y_l - a_5 & a_{10} y_l - a_6 & a_{11} y_l - a_7 \\ b_9 x_r - b_1 & b_{10} x_r - b_2 & b_{11} x_r - b_3 \\ b_9 y_r - b_5 & b_{10} y_r - b_6 & b_{11} y_r - b_7 \end{bmatrix}}_{\mathbf{W}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \underbrace{\begin{bmatrix} a_4 - x_l \\ a_8 - y_l \\ b_4 - x_r \\ b_8 - y_r \end{bmatrix}}_{\mathbf{q}}, \qquad \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{W}^{+}\mathbf{q} = (\mathbf{W}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{q}$$

Object Recognition – challenges are scaling, deformation, background, clutter and in-class variation

Clustering



Within-cluster sum of squares

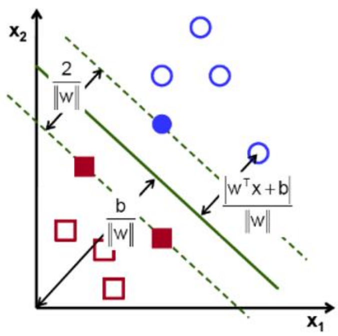
$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

Between-class sum of squares

$$BSS = \sum_i |C_i| (m_i - m)^2$$

$BSS + WSS = \text{const}$

Support Vector Machine – given linearly separable dataset, find a separating hyperplane; there are infinitely many hyperplanes, but optimal plane will be the one with the largest margin



$$\text{minimize} \quad J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$$

$$\alpha_i \left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right] = 0, \quad \forall i = 1 \ldots N$$
$$\Rightarrow \alpha_i = 0 \text{ or } y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$$

Any vector in the dataset either serve as support vectors ($a_i = 0$) to define hyperplane or they have no contribution ($a_i \neq 0$); as such, the entire dataset can be replaced by only support vectors

Precision, Recall and F-measure – if a cut-off threshold of 0.8 is chosen, any instance with posterior probability greater than 0.8 is classified as positive

		PREDICTED CLASS	
		Class =Yes	Class= No
ACTUAL CLASS	Class =Yes	(TP) 3	(FN) 2
	Class =No	(FP) 3	(TN) 2

Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

p = TP/(TP+FP)= 3/(3+3) = 1/2
r = TP/(TP+FN) =3/(3+2) = 3/5
F-measure = 2pr/(p+r)=6/11

ROC Curve

Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

Bag of Features – quantize features using visual vocabulary; represent images of frequencies of "visual words", e.g. using histograms; use SVM (if binary) or other supervised learning to classify