Benjamin Scialom
A20430263

Computer Vision
CS-512
Homework 3

---

1) Corner detection

a.

A corner is easily detectable with an orientation histogram. Indeed, if there is a corner, we can see two peaks in the orientation histogram of a local window.

The basic algorithm is :

1. Find a correlation matrix of gradients in a local window (C is the gradient correlation matrix)
2. Find eigenvalues of the correlation matrix related to this window

$$\begin{cases} E(v) = v^T C v \\ v^* = argmin_v(E_v) \end{cases}$$

To find V that minimize $E(v)$, we have to solve : $\nabla E(v) = 0$.
By using gradient rules, it means resolve this equation : $2Cv = 0$.

Thus, the solution is the eigenvector to the smallest eigenvalue. The eigenvalue stands for the variance in the corresponding direction.

3. Corner detected if eigenvalues are sufficiently large

b.

PCA aims to detect the correlation between variables. If a strong correlation between variables exists, the attempt to reduce the dimensionality only makes sense. This is what PCA is all about: finding the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional subspace while retaining most of the information.

The basic idea is :

1. Find direction V which is the strict projection of a group of point {g(i)} such as V is minimized (minimization of the projection span). It is given by the following formula :

$$E(v) = \sum_i (g_i * v)^2 = \sum_i (v^T g_i)(g_i^T v) = v^T C v$$

Where C is an outer product (2x2 matrix) defined by :

$$C = \sum_i g_i g_i^T$$

2. Then, we look for additional direction which minimize projection and being orthogonal to the previous one.

c.

$$C = \sum_{i=1}^{q} g_i g_i^T = D^T D = \begin{bmatrix} 4 & 6 \\ 6 & 44 \end{bmatrix}$$

With : $D^T = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$

d.

Let take λ_1 the eigenvalue of the direction V_1 and λ_2 the eigenvalue of the direction V_2.
The condition on the eigenvalues of the gradient correlation matrix that is used for corner detection is :

-> if $\lambda_1 . \lambda_2 > \tau$ then it is a corner

That is to say if the two eigenvalues are small there is no corner. However if one is small and the other is big there is a possibility of corner if the product of the eigenvalues is superior to the threshold.

e.

The non-maximum suppression algorithm for corner detection has the 4 following steps :

1. We compute λ_1 and λ_2 for all window (overlapping windows)
2. We Select windows with $\lambda_1 . \lambda_2 > \tau$ and sort in decreasing order
3. We select the top of the list as corner and delete all other corners in its neighborhood from the list
4. We Stop once detecting $\alpha\%$ of the points as corner

f.

The Harris corner algorithm for corner detection had the 3 following steps :

1. We have to compute correlation matrix C for windows (overlapping windows)
2. We compute « cornerness » measure :

   $$G(c) = det(C) - K . tr^2(C) = \lambda_1 . \lambda_2 - k . (\lambda_1 + \lambda_2)^2$$

   Where, $\lambda_1 . \lambda_2$ stands for the possibility of having corner and $(\lambda_1 + \lambda_2)^2$ stands for the possibility of having an edge.

3. We detect corner when $G(c)$ is high

K is a user parameter generally between 0.04 and 0.15 but it could go to 0.5 maximum.
If K = 0 then G(c) detects corner and if K=0.5 then G(c) detects edge.

g.

Once we know that there is a corner in the window, we determine the localization of this corner. To determine if P is a corner, we connect each point of the edge x_i to p. The idea is to minimize this sum : $\sum_i n_i . (x_i - p)$.

$$E(p) = \sum_i (\nabla I(x_i) \,.\, (x_i - p))^2$$
$$= \sum_i (xi - p)^T \nabla I(x_i) \nabla I(x_i)^T (xi - p)$$

Let consider $p* = argmin(E(p))$. To find $p*$, we have to solve : $\nabla E(p) = 0$

$$E(p) = 0 \equiv -2 \sum_i \nabla I(x_i) \nabla I(x_i)^T \,.\, (x_i - p) = 0$$
$$\equiv \sum_i \nabla I(x_i) \nabla I(x_i)^T \,.\, p = \sum_i \nabla I(x_i) \nabla I(x_i)^T \,.\, x_i$$
$$\equiv C \,.\, p = \sum_i \nabla I(x_i) \nabla I(x_i)^T \,.\, x_i$$

The location of the corner written $p*$ is given thanks to the correlation matrix C.

$$p* = C^{-1} \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

We know that C is invertible since we detected a corner before looking for its location.

h.

Given a corner, we would like to compare its corner in another image or use it to characterize the image. That's what we call « feature point characterization ». In order to have good characterization some properties should be respected :

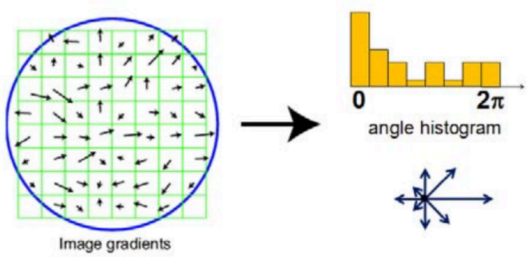
- Translation invariance (local window)
- Rotation invariance (histograms) -> shift the strongest peak to be at zero
- Scale invariance (pyramid)
- Illumination invariance (gradients)

What's an histograms of oriented gradients : (HOG)

1. Split into cell that are possibly overlapping
2. Create orientation histogram in each cells (using edge or gradient directions, possibly weighted by distance from center of gradient magnitude)
3. Concatenate these orientation histograms

i.

A dominant orientation estimate can be computed by creating a histogram of all the gradient orientations (weighted by their magnitudes or after thresholding out small gradients) and then finding the significant peaks in the distribution.



2) Line detection

a.

y = a.x + b : a line in image correspond to a point in parameter space
b = y - a.x : line in parameter space corresponds to point in image

Hough transform :

- Each point in the image defines a line in parameters space
- Multiple points define multiple lines in parameters space
- Intersection at lines in parameter space indicate parameters common to multiple points on the same line

Problem :

For the model y = a.x + b, it is difficult to determine **the possible range of « a »**. For instance, how can we represent **vertical lines** ?

b.

If we draw the slope into a a graph, we can notice that the value of c in the equation b.y + a.x + c =0 can be given by taking x=0 and using the angle of the slope :

$$tan(45) = \frac{-c}{10} \implies c = -tan(45).10 = -10$$

Moreover, since when y=0 the x-axis value is -10 and we now know c, we can determine that :

$$a = \frac{-c}{10} = 1$$

Therefore, the equation is : $y = -x + 10 \equiv y + x - 10 = 0$ (a=1,b=1 c=-10)

c.

The explicit polar equation of lines is given by :

$$(n_x, n_y) \,.\, (x, y) = d \equiv \cos(\theta)x + \sin(\theta)y - d = 0$$

After voting for a point, a point in the image defines a line in parameter space (describing the parameters at all possible lines through two points).

d.

- If we are using polar line equation, for each θ_i there is a vote of d_i given (x, y).
- If we are using basic line equation, given (x, y) we cacao the parameter a and compute b. Then, we cast votes at the location (a, b) to have a line.

e.

Bigger bins are more efficient but provide les localization. Indeed, big cells make the localization faster but small cells make the localization more accurate.

f.

We have to estimate a normal vector at each pixel. Each pixel provide a vote which is a small curve section (accounting for inaccurate normal direction).

A point (x, y) with normal θ votes for d :

$$d = x \,.\, \cos(\theta + \alpha\Delta\theta) + y \,.\, \sin(\theta + \alpha\Delta\theta)$$, where α is between -1 and 1.

g.

The number of dimension of the parameter space is 3.

---

3) Model Fitting

a.

Given the parameter $(x_i, y_i)_{i=1}^{n}$, we have to find the two parameter a and b of the line y= a.x + b. So, we have to minimize the difference between the vote and the model :

Let take a^* and b^* the two parameters that minimize the difference.

$$a^*, b^* = argmin_{a,b}(E(a,b))$$

where,

$$E(a,b) = \sum_{i=1}^{n} (y_i - (a \,.\, x_i + b))^2$$

To find a^* and b^*, we have to solve: $\nabla E(a,b) = 0$. But this is a bad model (y=ax+b) since **the parameter a is very difficult to determine if the slope is steep.**

Therefore, **vertical line** can't be fitted accurately with this equation.

b.

$$l^T x = 0 \equiv ax + by + c = 0 \text{ where } l^T = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \text{ and } x = [x, y, 1].$$

In the last equation a and b stand for the normal coordinate while c stands for the distance from the origin. Therefore, $l = [a = 1, b = 2, c = -2]$.

c.

When p is not on the line l, $l^T p$ is the geometric distance of p from l.
So, the objective is given by :

$$E(l) = \sum_i (l^T p_i)^2 = \sum_i l^T p_i p_i^T l = l^T \sum_i P_i P_i^T l = l^T S l$$

Where S is the correlation matrix for the line parameters. Finally, we have to minimize l^* to find the unknown parameters :

$$l^* = argmin_l(E(l))$$

Thus,

$$\nabla E(l) = 0 \rightarrow sl = 0$$

We can conclude that l is the eigenvector of s belonging to 0 eigenvalue.

d.

The matrix that has to be formed to find to find the parameters of the line that fits the point is the correlation matrix S.

$S = D^T D$ where $D = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ . & . & . \\ x_n & y_n & 1 \end{bmatrix}$

In our case $D = \begin{bmatrix} O & 1 & 1 \\ 1 & 3 & 1 \\ 2 & 6 & 1 \end{bmatrix}$. Then, $S = \begin{bmatrix} 5 & 15 & 3 \\ 15 & 46 & 10 \\ 3 & 10 & 3 \end{bmatrix}$.

e.

The explicit equation for conic curves is :

$l^T p = 0$, where $p = (x^2, xy, y^2, x, y, 1)$ and $l^T = (a, b, c, d, e, f)$.

Which is equivalent to : $ax^2 + bxy + cy^2 + dx + ey + f = 0$

And, the constraint that guarantees that the model will be an ellipse is : $b^2 - 4ac$.

f.

The equation that needs to be solved for fitting a conic curve using geometric distance is :

$l^* = argmin_l(E(l))$ and where :

$$E(l) = \sum_{i}^{n} (l^T p_i)^2 = l^T S l \text{ with } S = \sum_{i} p_i p_i^T. \text{ } (p_i \text{ are points}).$$

However, we have to take into account the constraint for the ellipse :

So the objective function becomes :

$$E(l) = l^T S l + \lambda(l^T C l + 1)$$

If the constraint is satisfied, then $l^T C l + 1 = 0$.
If we want to find l^*, we have to solve the equation : $\nabla E(l) = 0$

The matrix C is : $C = \begin{bmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

The issue with algebraic distance is that points are not treated equally. That is to say, we give **priority to vertical post of the ellipse**.

g.

The geometric distance measure the distance from a point to arbitrary implicit curve f(x)=0.
Let define $d(p, f) = |p - x^*|$ as the euclidian distance where x^* is the closest point.

The solution is given by the following approximation :

$$d(p, f) = |p - x^*| = \frac{|f(p)|}{|\nabla f(x^*)|} \approx \frac{|f(p)|}{|\nabla f(p)|}$$

So, the objective is :

$$E(l) = \sum_{i} \frac{|f(p_i, l)|}{|\nabla f(p_i, l)|} \text{ where } f(p_i, l) = l^T p_i$$

The a additional complication is according to me to find the closest point. To do that, we have to find the tangent at x^*(compute the gradient and rotate by -90°).

h.

- what's active contour ?

We gradually deform initial contour to fit object boundaries by using parametric curve $\phi(s)$ to represent the contour.

- The objective function is given by :

$$E[\phi(s)] = \int_{\phi} (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image} ds \text{ (continuous)}$$

The first term in $E(p_i)$ stands for the **continuity energy**, the second one stands for **the curative energy** and the last one stands for **the image energy**.
The two first terms represent the **internal energy** while the last one represents the **external energy**.

continuity energy : $E_{cont} = |\frac{\partial \phi}{\partial s}|^2$

image energy : $E_{image} = -|\nabla I|^2$

curative energy : $E_{curv} = |\frac{\partial^2 \phi}{\partial^2 s}|^2$

i.

For a discrete curve the objective becomes :

$$E(p_i) = \sum_{i=1}^{n} \alpha_i(|p_{i+1} - p_i|)^2 + \sum_{i=1}^{n} \beta_i(|p_{i+1} - 2p_i + p_{i-1}|)^2 - \sum_{i=1}^{n} \gamma_i |\nabla I(p_i)|^2 \text{(discrete)}$$

α_i, β_i, γ_i are user selected parameters , p_i are the unknown model parameters and d is a computed parameter.
Each term stands for the exact same thing that has been describe just at the previous question.

j.

By putting the variable $\beta_i = 0$, we can allow discontinuity at corners.