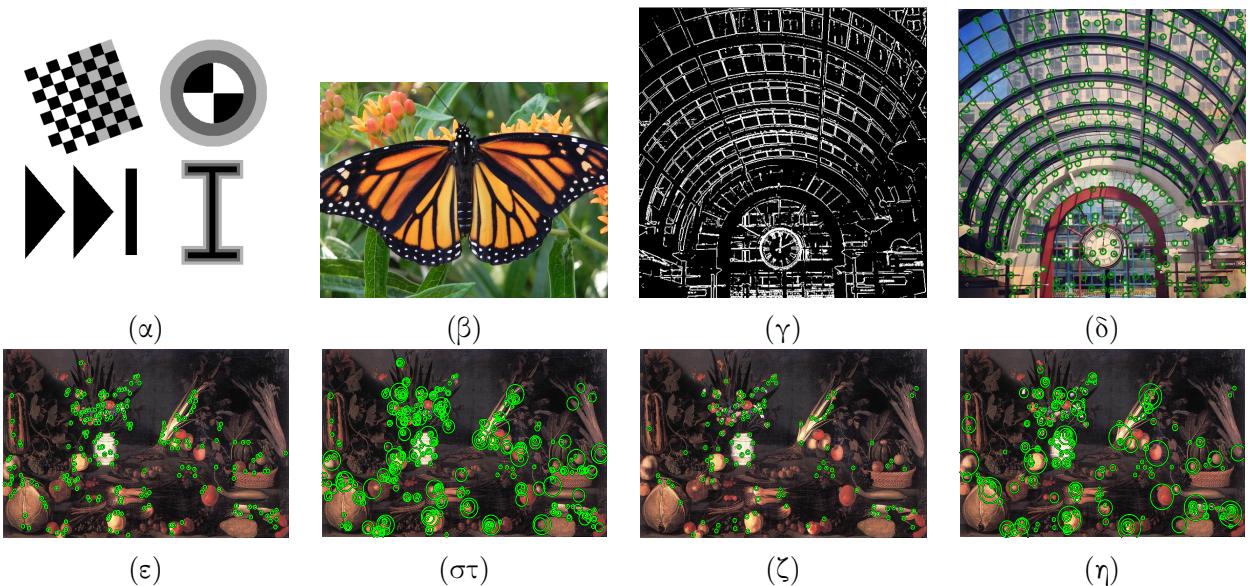


Εξηγήστε περιεκτικά και επαρκώς την εργασία σας. Επιτρέπεται για τους προπτυχιακούς φοιτητές η συνεργασία εντός ομάδων των 2 ατόμων. Κάθε ομάδα 2 ατόμων υποβάλλει μια κοινή αναφορά που αντιπροσωπεύει μόνο την προσωπική εργασία των μελών της. Για τους μεταπτυχιακούς φοιτητές η εργασία είναι ατομική. Αν χρησιμοποιήσετε κάποια άλλη πηγή εκτός του βιβλίου και του εκπαιδευτικού υλικού του μαθήματος, πρέπει να το αναφέρετε. Η παράδοση της αναφοράς και του κώδικα της εργασίας θα γίνει ηλεκτρονικά στο helios του μαθήματος (<https://helios.ntua.gr/course/view.php?id=964>). Στη σελίδα αυτή, στην ενότητα "Απορίες Εργαστηρίων", μπορείτε επίσης να υποβάλετε απορίες και ερωτήσεις δημιουργώντας issues.

Επισημαίνεται ότι απαγορεύεται η ανάρτηση των λύσεων των εργαστηριακών ασκήσεων στο github, ή σε άλλες ιστοσελίδες. Η σχεδίαση και το περιεχόμενο των εργαστηριακών projects αποτελούν αντικείμενο πνευματικής ιδιοκτησίας της διδακτικής ομάδας του μαθήματος.

Θέμα: Εντοπισμός Σημείων Ενδιαφέροντος και Εξαγωγή Χαρακτηριστικών σε Εικόνες



Σχήμα 1: (α) Συνθετική εικόνα για ανίχνευση ακμών, (β) Πραγματική εικόνα για ανίχνευση ακμών, (γ) Παράδειγμα ανίχνευσης ακμών, (δ) Παράδειγμα ανίχνευσης γωνιών, (ε) Παράδειγμα ανίχνευσης blobs, (στ) Παράδειγμα πολυχλιμακωτής ανίχνευσης blobs, (ζ) Παράδειγμα ανίχνευσης blobs, (η) Παράδειγμα πολυχλιμακωτής ανίχνευσης blobs.

Μέρος 1: Ανίχνευση Ακμών σε Γκρίζες Εικόνες

1.1. Δημιουργία Εικόνων Εισόδου

- 1.1.1 Η αρχική γκρίζα εικόνα I_0 = “edgetest_24.png” φαίνεται στο σχήμα 1(a) (βρίσκεται στο επιπρόσθετο υλικό στο helios). ‘Διαβάστε’ την εικόνα αυτή.
- Βοήθεια για Python: συναρτήσεις (cv2) imread, (matplotlib.pyplot) imshow, show.
- 1.1.2 Προσθήκη θορύβου στην I_0 , για την δημιουργία θορυβωδών εικόνων $I(x, y) = I_0(x, y) + n(x, y)$ που θα χρησιμοποιηθούν σαν είσοδοι στον αλγόριθμο ανίχνευσης ακμών. Ο θόρυβος $n(x, y)$ είναι λευκός gaussian, με μέση τιμή 0 και τυπική απόκλιση σ_n τέτοια ώστε ο PSNR (Peak-to-peak Signal to Noise Ratio) να έχει συγκεκριμένες τιμές. Χρησιμοποιήστε 2 διαφορετικές τιμές για τον PSNR του θορύβου:
- i) PSNR=20 dB, ii) PSNR=10 dB.

Ο PSNR ορίζεται ως εξής:

$$\text{PSNR} = 20 \log_{10} \left(\frac{I_{\max} - I_{\min}}{\sigma_n} \right) (\text{dB}), \quad (1)$$
$$I_{\max} = \max_{x,y} I(x, y), \quad I_{\min} = \min_{x,y} I(x, y),$$

από όπου μπορεί να βρεθεί η σ_n που επιφέρει συγκεκριμένο PSNR. Ελάττωση του PSNR αντιστοιχεί σε αύξηση των επιπέδων θορύβου της εικόνας.

► Βοήθεια για Python: (numpy) random.normal

1.2. Υλοποίηση Αλγορίθμων Ανίχνευσης Ακμών

Κατασκευάστε μία συνάρτηση EdgeDetect η οποία να υλοποιεί τον αλγορίθμο ανίχνευσης ακμών που περιγράφεται στα ακόλουθα βήματα:

- 1.2.1 Δημιουργία των κρουστικών αποκρίσεων δύο διακριτών γραμμικών φίλτρων που προσεγγίζουν τα συνεχή φίλτρα με τις εξής κρουστικές αποκρίσεις:
- i) Διδιάστατη Gaussian $G_\sigma(x, y)$,
 - ii) Laplacian-of-Gaussian (LoG) $h(x, y) = \nabla^2 G_\sigma(x, y)$.

Και στις δύο περιπτώσεις, να χρησιμοποιηθεί η ίδια τυπική απόκλιση (χωρική κλίμακα) σε για την Gaussian και οι πυρήνες να έχουν έκταση $n \times n$ pixels με $n = \text{ceil}(3 \cdot \sigma) \cdot 2 + 1$.

► Βοήθεια για Python: συναρτήσεις (numpy) ceil, (cv2) getGaussianKernel. Δημιουργήστε μια συνάρτηση που να υλοποιεί τα παραπάνω φίλτρα. Για την LoG, συνίσταται η άμεση υλοποίηση του πυρήνα (βοήθεια: (numpy) meshgrid). Αντ' αυτού, μπορεί να γίνει υλοποίηση μέσω εφαρμογής Laplacian στον Γκαουσιανό πυρήνα (βοήθεια: (cv2) Laplacian), κάτι που μπορεί όμως να οδηγήσει σε σφάλμα προσέγγισης.

- 1.2.2 Προσέγγιση της Laplacian L της εξομαλυμένης εικόνας $I_\sigma(x, y) = G_\sigma * I(x, y)$. Χρησιμοποιήστε 2 διαφορετικές εναλλακτικές:

- i) Γραμμική (L_1): συνέλιξη της I με την LoG h :

$$L_1 = \nabla^2(G_\sigma * I) = (\nabla^2 G_\sigma) * I = h * I \quad (2)$$

- ii) Μη-γραμμική (L_2): μη-γραμμική εκτίμηση της Laplacian της I_σ με μορφολογικά φίλτρα:

$$L_2 = I_\sigma \oplus B + I_\sigma \ominus B - 2I_\sigma, \quad B = \begin{array}{|c|c|c|} \hline & \bullet & \\ \hline \bullet & \bullet & \bullet \\ \hline & \bullet & \\ \hline \end{array}. \quad (3)$$

► Βοήθεια για Python: συναρτήσεις (cv2) `dilate`, `erode`, `filter2D`.

- 1.2.3 Προσέγγιση των zerocrossings (σημεία μηδενισμού) της L . Ένας πιθανός αλγόριθμος για το στάδιο αυτό είναι ο εξής :

- (α) Δημιουργία της Δυαδικής Εικόνας Προσήμου X της L . $X = (L \geq 0)$
(β) Εύρεση του περιγράμματος της X (συμμετρικό ως προς X και X^c):

$$Y = (X \oplus B) - (X \ominus B) \approx \partial X. \quad (4)$$

Τα zerocrossings αντιστοιχούν στα σημεία στα οποία η δυαδική εικόνα Y έχει την τιμή 1.

- 1.2.4 Απόρριψη των zerocrossings σε σχετικά ομαλές περιοχές. Τελικά επιλέγονται ως ακμές τα zerocrossings Y στα οποία η εξομαλυμένη εικόνα I_σ έχει σχετικά μεγάλη κλίση, δηλ. τα pixels (i, j) για τα οποία:

$$Y[i, j] = 1 \text{ και } \|\nabla I_\sigma[i, j]\| > \theta_{edge} \cdot \max_{x, y} \|\nabla I_\sigma\| \quad (5)$$

όπου θ_{edge} μία παράμετρος κατωφλίου.

► Βοήθεια για Python: συνάρτηση (numpy) `gradient`. Επίσης, τελεστής “&” για το λογικό ‘και’, ο οποίος εφαρμόζεται σε δύο πίνακες ίδιου μεγέθους και δρα στοιχείο - προς - στοιχείο.

Σημείωση Η συνάρτηση `EdgeDetect` που θα υλοποιήσετε για την ανίχνευση ακμών πρέπει να δέχεται σαν ορίσματα την εικόνα εισόδου I , την τυπική απόκλιση σ της Gaussian, την παράμετρο θ_{edge} και τον τύπο προσέγγισης της Laplacian (Γραμμική ή Μη-γραμμική). Επίσης, η συνάρτηση αυτή πρέπει να επιστρέψει στην έξοδο μία δυαδική εικόνα ακμών D , με τιμή 1 μόνο στα σημεία που επιλέγονται σαν ακμές.

1.3. Αξιολόγηση των Αποτελεσμάτων Ανίχνευσης Ακμών

- 1.3.1 Υπολογισμός των “αληθινών” ακμών, χρησιμοποιώντας την αρχική καθαρή εικόνα I_0 . Η δυαδική εικόνα αληθινών ακμών T μπορεί να βρεθεί υπολογιστικά εφαρμόζοντας τον απλό τελεστή ακμών (edge operator) στην I_0 :

$$M = (I_0 \oplus B) - (I_0 \ominus B) \quad (6)$$

και χρησιμοποιώντας κατωφλιοποίηση για μετατροπή της εξόδου σε δυαδική εικόνα:
 $T = M > \theta_{realeedge}$.

- 1.3.2 Ποσοτική αξιολόγηση των αποτελεσμάτων ανίχνευσης ακμών. Χρησιμοποιώντας τις δυαδικές εικόνες των αληθινών ακμών T και των ακμών D που ανίχνευσε ο αλγόριθμός σας στην θορυβώδη εικόνα εισόδου, υπολογίστε το ακόλουθο κριτήριο ποιότητας του αποτελέσματος ανίχνευσης:

$$C = [Pr(D|T) + Pr(T|D)] / 2 \quad (7)$$

όπου $Pr(D|T)$ είναι το ποσοστό των ανιχνευθεισών ακμών που είναι αληθινές (Precision) και $Pr(T|D)$ το ποσοστό των αληθινών ακμών που ανιχνεύθησαν (Recall). Αν τα D και T ειδωθούν σαν διαχριτά σύνολα (με στοιχεία τα pixels στα οποία η δυαδική εικόνα έχει τιμή 1), τότε ισχύει: $Pr(T|D) = \text{card}(D \cap T)/\text{card}(T)$, όπου το $\text{card}(\cdot)$ δίνει τον αριθμό των στοιχείων ενός συνόλου.

► Βοήθεια για Python: Το $\text{card}(A)$ για μία δυαδική εικόνα - σύνολο A μπορεί να υπολογιστεί ως $A.\text{sum}()$.

1.3.3 Στα παραπάνω, για κάθε διαφορετική είσοδο (διαφορετικό $PSNR$), πειραματιστείτε με τις τιμές των παραμέτρων σ και θ_{edge} ώστε να πετύχετε όσο το δυνατόν καλύτερα αποτελέσματα. Για κάθε περίπτωση, παρουσιάστε το 'καλύτερο' αποτέλεσμα που καταφέρατε να εξάγετε και σχολιάστε. Επίσης, σχολιάστε την επίδραση που έχει η μεταβόλη των παραμέτρων και του επιπέδου του θορύβου στο αποτελεσμα ανίχνευσης ακμών. Τέλος, συγκρίνετε τα αποτελέσματα για τις διαφορετικές προσεγγίσεις της Laplacian. Σε όλες τις περιπτώσεις, ο σχολιασμός σας να είναι οσο το δυνατόν πιο συνοπτικός.

► Ενδεικτικές Τιμές Παραμέτρων: A) Για είσοδο με $PSNR = 20$ dB, $\sigma = 1.5$ και $\theta_{edge} = 0.2$, B) Για είσοδο με $PSNR = 10$ dB, $\sigma = 3$ και $\theta_{edge} = 0.2$.

1.4. Εφαρμογή των Αλγορίθμων Ανίχνευσης Ακμών σε Πραγματικές εικόνες

1.4.1 Εφαρμόστε τους αλγόριθμους ανίχνευσης ακμών που υλοποιήσατε στην πραγματική εικόνα "butterfly.jpg" χωρίς να προσθέσετε κάποιο θόρυβο.

1.4.2 Πειραματιστείτε και πάλι με τις τιμές των παραμέτρων σ και θ_{edge} ώστε να πετύχετε όσο το δυνατόν καλύτερα αποτελέσματα (ποιοτική αξιολόγηση). Παρουσιάστε το 'καλύτερο' αποτέλεσμα που καταφέρατε να εξάγετε και σχολιάστε το συνοπτικά. Επίσης, παρουσιάστε αποτελέσματα και για άλλες τιμές των παραμέτρων και σχολιάστε / ερμηνεύστε συνοπτικά τον τρόπο που η ανίχνευση ακμών μεταβάλλεται.

Μέρος 2: Ανίχνευση Σημείων Ενδιαφέροντος (Interest Point Detection)

Στο Μέρος 2 της εργαστηριακής άσκησης θα ασχοληθούμε με την ανίχνευση σημείων ενδιαφέροντος σε εικόνες. Θα κληθείτε να υλοποιήσετε ένα σύνολο από ανιχνευτές και να **σχολιάσετε** τις διαφορές που έχουν μεταξύ τους. Τους ζητούμενους ανιχνευτές θα τους υλοποιήσετε ως συναρτήσεις με είσοδο την εικόνα και τις απαιτούμενες παραμέτρους και ως έξοδο τις συντεταγμένες αλλά και την κλίμακα των σημείων ενδιαφέροντος που εντοπίστηκαν. Για τις ανάγκες του συγκεκριμένου μέρους θα σας δωθούν δύο εικόνες ("urban.edges.jpg" και "Caravaggio.png") πάνω στις οποίες θα πρέπει να σχεδιαστούν τα αποτελέσματα των ανιχνευτών που υλοποιήσατε για ενδεικτικές τιμές των παραμέτρων.

Για τις ανάγκες της οπτικοποίησης των σημείων ενδιαφέροντος που ανιχνεύσατε (ως κύκλος με κέντρο το σημείο αυτό και ακτίνα ανάλογη της κλίμακας που ανιχνεύθηκε) μπορείτε να χρησιμοποιήσετε τη συνάρτηση `interest_points_visualization` που βρίσκεται στο αρχείο `cv24_lab1_part2_utils.py`.

► Η συνάρτηση `interest_points_visualization` δέχεται σαν πρώτο όρισμα την έγχρωμη εικόνα και ως δεύτερο έναν πινάκα $N \times 3$ που αντιστοιχεί στα σημεία που ανιχνεύθηκαν με τις 2 πρώτες στήλες να αποτελούν τις συντεταγμένες τους και την τρίτη στην κλίμακα σ στην οποία ανιχνεύθηκαν.

► Ενδεικτικές Τιμές Παραμέτρων: $\sigma = 2$, $\rho = 2.5$, $k = 0.05$, $\theta_{corn} = 0.005$, $s = 1.5$, $N = 4$.

2.1. Ανίχνευση Γωνιών

Μια διαισθητικά απλή κατηγορία σημείων ενδιαφέροντος είναι η επιλογή σημείων που αντιστοιχούν σε γωνίες της εικόνας. Για την ανίχνευση των γωνιών όμως υλοποιήσετε την κλασσική μέθοδο των Harris-Stephens, όπως επεξηγείται παρακάτω. Για το ερώτημα αυτό και μόνο (2.1), εφαρμόστε την αναπτυχθείσα μέθοδο και στην εικόνα ακμών του προηγούμενου μέρους “butterfly.jpg”.

2.1.1 Υπολογίστε τα στοιχεία J_1 , J_2 και J_3 του δομικού τανυστή \mathbf{J} σε κάθε pixel (x, y) της εικόνας, σύμφωνα με τις σχέσεις:

$$\begin{aligned} J_1(x, y) &= G_\rho * \left(\frac{\partial I_\sigma}{\partial x} \cdot \frac{\partial I_\sigma}{\partial x} \right) (x, y) \\ J_2(x, y) &= G_\rho * \left(\frac{\partial I_\sigma}{\partial x} \cdot \frac{\partial I_\sigma}{\partial y} \right) (x, y) \\ J_3(x, y) &= G_\rho * \left(\frac{\partial I_\sigma}{\partial y} \cdot \frac{\partial I_\sigma}{\partial y} \right) (x, y) \end{aligned} \quad (8)$$

όπου $I_\sigma = G_\sigma * I$, και G_ρ , G_σ δισδιάστατοι Gaussian πυρήνες ομαλοποίησης με τυπικές αποκλίσεις σ (κλίμακα διαφόρισης) και ρ (κλίμακα ολοκλήρωσης) αντίστοιχα.

► Βοήθεια για Python: Τα στοιχεία J_1 , J_2 και J_3 μπορούν να ειδωθούν σαν γκρίζες εικόνες ίδιου μεγέθους με την εικόνα εισόδου I (συναρτήσεις `gradient` ή `filter2D`).

Σημείωση Ο τελεστής “.” στη σχέση (8) αντιστοιχεί σε πολλαπλασιασμό πινάκων σημείο προς σημείο.

2.1.2 Υπολογίστε τις ιδιοτιμές λ_- , λ_+ του τανυστή \mathbf{J} σε κάθε pixel, σύμφωνα με τις σχέσεις:

$$\lambda_{\pm}(x, y) = \frac{1}{2} \left(J_1 + J_3 \pm \sqrt{(J_1 - J_3)^2 + 4J_2^2} \right) \quad (9)$$

Σχεδιάστε σαν γκρίζες εικόνες τα $\lambda_+(x, y)$ και $\lambda_-(x, y)$. Αναφέρετε συνοπτικά τις παρατηρήσεις σας για τη σχέση των τιμών αυτών με τα χαρακτηριστικά της εικόνας (ακμές, γωνίες, κτλ.).

2.1.3 Με βάση τις υπολογισμένες ιδιοτιμές λ_- , λ_+ , να εξαχθεί το ακόλουθο “χριτήριο γωνιότητας” (cornerness criterion):

$$R(x, y) = \lambda_- \lambda_+ - k \cdot (\lambda_- + \lambda_+)^2, \quad (10)$$

όπου k είναι μία μικρή θετική σταθερά. Τελικά, επιλέξτε σαν γωνίες τα pixels (x, y) τα οποία:

- (Σ1) Είναι μέγιστα του R εντός τετραγωνικών παραθύρων που τα περιβάλλουν το μέγεθος των οποίων εξαρτάται από την κλίμακα σ ,
- (Σ2) Αντιστοιχούν σε τιμή του R μεγαλύτερη από ένα ποσοστό του ολικού μεγίστου του R , δηλαδή $R(x, y) > \theta_{corn} \cdot R_{max}$, όπου θ_{corn} ένα κατάλληλα επιλεγμένο κατώφλι.

► Βοήθεια για Python: Για την εύρεση των σημείων που ικανοποιούν την συνθήκη (Σ1), μπορείτε να χρησιμοποιήσετε τις εντολές:

```

ns = np.ceil(3*sigma)*2+1
B_sq = disk_strel(ns)
Cond1 = ( R==cv2.dilate(R,B_sq) )

```

Η συνάρτηση `disk_strel` δίνεται στο αρχείο `cv24_lab1_part2_utils.py`

2.2. Πολυκλιμακωτή Ανίχνευση Γωνιών

Στο προηγούμενο ερώτημα η ανίχνευση των γωνιών γινόταν σε μία μόνο κλίμακα η οποία οριζόταν από τις παραμέτρους σ και ρ . Στο ερώτημα αυτό ωστε να ανιχνεύει γωνίες σε πολλαπλές κλίμακες και να μας επιστρέψει εκτός από τα σημεία που αντιστοιχούν σε γωνίες και την κλίμακα στην οποία ανιχνεύτηκαν. Η μέθοδος που ωστε να ανιχνεύει γωνίες και την κλίμακα στην οποία ανιχνεύτηκαν. Η μέθοδος που χρησιμοποιείται (Harris-Laplacian) αποτελείται από δύο στάδια, ένα για την επιλογή σημείων σε κάθε κλίμακα και ένα για την τελική επιλογή των σημείων που παρουσιάζουν μέγιστο σε κάποια μετρική αμετάβλητη ως προς την κλίμακα.

2.2.1 Το πρώτο στάδιο αποτελείται από την εφαρμογή του αλγορίθμου εύρεσης γωνιών μονής κλίμακας για διαφορετικές κλίμακες ολοκλήρωσης και διαφόρισης:

$$\sigma_0, \sigma_1, \dots, \sigma_{N-1} = s^0 \sigma_0, s^1 \sigma_0, \dots, s^{N-1} \sigma_0 \quad (11)$$

$$\rho_0, \rho_1, \dots, \rho_{N-1} = s^0 \rho_0, s^1 \rho_0, \dots, s^{N-1} \rho_0 \quad (12)$$

όπου σ_0, ρ_0 οι αρχικές κλίμακες διαφόρισης και ολοκλήρωσης, s ο παράγοντας κλιμάκωσης και N ο αριθμός των κλιμάκων.

2.2.2 Το δεύτερο στάδιο αφορά την αυτόματη επιλογή της χαρακτηριστικής κλίμακας για κάθε σημείο που ανιχνεύτηκε στο προηγούμενο βήμα. Αρχικά υπολογίζουμε την κανονικοποιημένη Λαπλασιανή της Γκαουσιανής (Laplacian of Gaussian LoG) για τις διαφορετικές κλίμακες διαφόρισης του προηγούμενου βήματος:

$$|LoG(\mathbf{x}, \sigma_i)| = \sigma_i^2 |L_{xx}(\mathbf{x}, \sigma_i) + L_{yy}(\mathbf{x}, \sigma_i)|, \quad i = 0, \dots, N - 1 \quad (13)$$

Στη συνέχεια απορρίπτουμε τα σημεία για τα οποία η κλίμακα που ανιχνεύθηκαν δεν μεγιστοποιεί την *LoG* μετρική σε μια γειτονιά 2 διαδοχικών κλιμάκων.

2.3. Ανίχνευση Blobs

Μια από τις σημαντικότερες κατηγορίες σημείων ενδιαφέροντος βασίζονται στην ανίχνευση ‘blobs’, που ορίζονται ως περιοχές με κάποια ομοιογένεια που διαφέρουν σημαντικά από την γειτονιά τους. Για την εύρεση τέτοιων περιοχών, σε αντιστοιχία με το κριτήριο γωνιότητας της μεθόδου Harris, γίνεται χρήση των μερικών παραγώγων δεύτερης τάξης της εικόνας και συγκεκριμένα η ορίζουσα του πίνακα Hessian:

$$H(x, y) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix} \quad (14)$$

όπου $L_{xx}(x, y, \sigma) = \frac{\partial^2}{\partial x^2}\{I_\sigma(x, y)\}$, $L_{yy}(x, y, \sigma) = \frac{\partial^2}{\partial y^2}\{I_\sigma(x, y)\}$ και $L_{xy}(x, y, \sigma) = \frac{\partial^2}{\partial x \partial y}\{I_\sigma(x, y)\}$.

2.3.1. Υπολογίστε τις μερικές παραγώγους δεύτερης τάξης της εικόνας L_{xx}, L_{xy}, L_{yy} επιλέγοντας μια τιμή για την κλίμακα σ και κατασκευάστε το κριτήριο $R(x, y) = \det(H(x, y))$ για κάθε pixel.

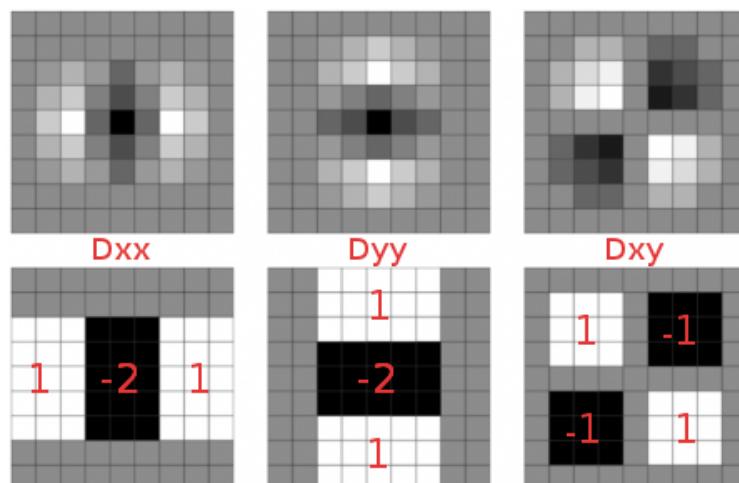
- 2.3.2. Επιλέξτε ως σημεία ενδιαφέροντος, τα σημεία που αποτελούν τοπικά μέγιστα και έχουν τιμή μεγαλύτερη από ένα κατάλληλα ορισμένο κατώφλι, ακριβώς όπως και στην ανίχνευση γωνιών με την μέθοδο Harris.

2.4. Πολυκλιμακωτή Ανίχνευση Blobs

- 2.4.1. Επαναλάβετε την διαδικασία της πολυκλιμακωτής ανίχνευσης γωνιών, προσθέτοντας ένα δεύτερο στάδιο για την επιλογή σημείων ενδιαφέροντος που παρουσιάζουν μέγιστο σε γειτονικές κλίμακες (Hessian-Laplace). Τα αρχικά σημεία ενδιαφέροντος για κάθε κλίμακα θα τα επιλέξετε σύμφωνα με την μέθοδο του προηγούμενου ερωτήματος.

2.5. Επιτάχυνση με την χρήση Box Filters και Ολοκληρωτικών Εικόνων (Integral Images)

Ο υπολογισμός της Hessian για κάθε κλίμακα αντιστοιχεί στην συνέλιξη της εικόνας με αυξανόμενο μεγέθους φίλτρα, που είναι μια υπολογιστικά ακριβή διαδικασία. Για την επιτάχυνση της μεθόδου αυτής, στο [1] (Speeded Up Robust Features) προτάθηκε η προσέγγιση των φίλτρων 2ης παραγάγου με Box Filters, δηλαδή με φίλτρα που βασίζονται σε αυθορισμάτων ορθογώνιων περιοχών, όπως φαίνεται και στην εικόνα 2. Η υλοποίηση τέτοιων αυθορισμάτων γίνεται πολύ αποτελεσματικά με χρήση Ολοκληρωτικών Εικόνων (βλέπε Παράρτημα).



Σχήμα 2: Οπτικοποίηση της προσέγγισης των φίλτρων 2ης παραγάγου με Box Filters

2.5.1 Υπολογίστε την Ολοκληρωτική Εικόνα.

► Βοήθεια για Python: συνάρτηση (numpy) `cumsum`

- 2.5.2 Υπολογισμός των L_{xx}, L_{xy}, L_{yy} με χρήση της Ολοκληρωτικής Εικόνας σύμφωνα με τα Box Filters (D_{xx}, D_{xy}, D_{yy}) για το επιλεχθέν σ . Στο στάδιο αυτό θα υλοποιηθούν τα φίλτρα της εικόνας 2, όπου κάθε άυθροισμα των ορθογώνιων παραθύρων των φίλτρων σταθμίζεται με τον εικονιζόμενο συντελεστή. Για δεδομένο σ , το οποίο αντιστοιχεί σε φίλτρο μεγέθους $n \times n$ (με $n = 2\text{ceil}(3\sigma) + 1$), το μέγεθος των παραθύρων υπολογίζεται ως εξής:

- D_{xx} : ύψος παραθύρου $4 \times \text{floor}(n/6) + 1$, πλάτος παραθύρου $2 \times \text{floor}(n/6) + 1$
- D_{yy} : ύψος παραθύρου $2 \times \text{floor}(n/6) + 1$, πλάτος παραθύρου $4 \times \text{floor}(n/6) + 1$
- D_{xy} : ύψος παραθύρου $2 \times \text{floor}(n/6) + 1$, πλάτος παραθύρου $2 \times \text{floor}(n/6) + 1$

Παρατηρήστε ότι για κάθε φίλτρο επιθυμούμε τα τοπικά αθροίσματα για ένα σταθερού μεγέθους παράθυρο και το τελικό φίλτρο προκύπτει ως σταθμισμένος συνδυασμός αποχρίσεων για το παράθυρο αυτό.

2.5.3 Εύρεση των σημείων ενδιαφέροντος ως τα τοπικά μέγιστα του χριτηρίου $R(x, y)$ με τον ίδιο τρόπο με τα προηγούμενα ερωτήματα.

$$R(x, y) = L_{xx}(x, y)L_{yy}(x, y) - (0.9L_{xy}(x, y))^2 \quad (15)$$

Οπτικοποιήστε (ως εικόνα) το χριτήριο R της απλής Hessian μεθόδου και το παραπάνω προσεγγιστικό χριτήριο R για κάποια κλίμακα σ . Επιλέξτε ενδεικτικά 3-4 κλίμακες ($\sigma \in (2, 10)$) και σχολιάστε την ποιότητα προσέγγισης για αυξανόμενες κλίμακες.

2.5.4 Επαναλάβετε την διαδικασία της πολυκλιμακωτής ανίχνευσης σημείων ενδιαφέροντος όπως εξηγήθηκε σε προηγούμενα ερωτήματα.

Μέρος 3: Εφαρμογές σε Ταίριασμα και Κατηγοριοποίηση Εικόνων με Χρήση Τοπικών Περιγραφητών στα Σημεία Ενδιαφέροντος

Τα σημεία ενδιαφέροντος δίνουν μια εκτίμηση περιοχών, οι οποίες περιέχουν σημαντικά χαρακτηριστικά της εικόνας. Για τον λόγο αυτό από τις περιοχές αυτές εξάγουμε τοπικούς περιγραφητές (local descriptors), που κωδικοποιούν μια γειτονιά (με ακτίνα που εξαρτάται από την κλίμακα) γύρω από τα σημεία ενδιαφέροντος. Ως τοπικούς περιγραφητές υπάρχουν πολλοί, όπως ο SURF, ο HOG και ο SIFT.

- **SURF** (Speed Up Robust Features)
- **HOG** (Histogram of Oriented Gradients)

Και οι δύο περιγραφητές έχουν ως είσοδο μια γειτονιά ενός σημείου ενδιαφέροντος και βασίζονται στην κωδικοποίηση της πληροφορίας υποτυμημάτων της γειτονιάς αυτής με χρήση της πρώτης κατεύθυντικής παραγώγου. Η μέθοδος των SURF υπολογίζει πρώτα την γενική κατεύθυνση της γειτονίας, έτσι ώστε να εξαχθούν περιστροφικά ανεξάρτητοι περιγραφητές.

► **Βοήθεια για Python:** Οι υλοποιήσεις των περιγραφητών δεν υπάρχουν στην παρούσα άσκηση και υπάρχουν στην εικόνα `img` ως είσοδος στην `cv2` για την εκχριμένη με τις συναρτήσεις `featuresSURF(I, points)` και `featuresHOG(I, points)` για SURF και HOG αντίστοιχα. Οι συναρτήσεις αυτές έχουν ως είσοδο την εικόνα και τα σημεία ενδιαφέροντος ως πίνακας $N \times 3$ (συντεταγμένες και κλίμακα), όπως έχουμε ήδη περιγράψει. Το υλικό και τα δεδομένα για το Μέρος 3 της εργαστηριακής άσκησης βρίσκονται στην ηλεκτρονική διεύθυνση: http://cvsp.cs.ntua.gr/courses/vision/Material/cv24_lab1_part3_material.zip. Οι συναρτήσεις που σας δίνονται σε αυτό το μέρος βρίσκονται στο `cv24_lab1_part3_utils.py` αρχείο και μπορείτε να τις κάνετε `import` από αυτό.

3.1. Ταίριασμα Εικόνων υπό Περιστροφή και Αλλαγή Κλίμακας

Στο μέρος αυτό υπάρχει εξετάσουμε την ικανότητα εύρεσης της περιστροφής και της κλίμακας με τη χρήση των ανιχνευτών σημείων ενδιαφέροντος που υλοποιήσατε και τους παραπάνω τοπικούς περιγραφητές. Συγκεκριμένα για το πείραμα αυτό έχουν χρησιμοποιηθεί 3 εικόνες, τις οποίες έχουμε παραμορφώσει περιστρέφοντάς τις (5 περιστροφές: $-20^\circ, -10^\circ, 0^\circ, 10^\circ, 20^\circ$) και

αλλάζοντας το μέγεθός τους (4 κλίμακες: 0.6, 0.8, 1.0, 1.2). Συνεπώς έχουμε συνολικά 20 παραμορφώσεις (μετασχηματισμοί ομοιότητας) και χρησιμοποιώντας μια από τις εικόνες ως εικόνα αναφοράς, θα προσπαθήσουμε να αντιστοιχίσουμε τους τοπικούς περιγραφητές της με αυτούς των υπόλοιπων παραμορφωμένων εικόνων. Η διαδικασία αυτή ονομάζεται ταίριασμα (matching). Επιλέγοντας ένα σύνολο αντιστοίχησης σημείων - τοπικών περιγραφητών, μπορούμε να εκτιμήσουμε τον μετασχηματισμό ομοιότητας μεταξύ των εικόνων και κατ' επέκταση την περιστροφή και διάφορα κλίμακας.

- 3.1.1 Η εκτίμηση της περιστροφής και της κλίμακας των εικόνων και η αποτίμησή τους γίνεται αυτόματα από την συνάρτηση `matching_evaluation`. Η συνάρτηση αποτίμησης `matching_evaluation` δέχεται ως ορίσματα μια συνάρτηση για την εξαγωγή σημείων ενδιαφέροντος και μια συνάρτηση για την εξαγωγή τοπικών περιγραφητών και επιστρέφει το μέσο σφάλμα εκτίμησης κλίμακας και το μέσο σφάλμα εκτίμησης γωνίας περιστροφής για κάθε μια από τις 6 εικόνες που χρησιμοποιήθηκαν. Παράδειγμα χρήσης της συνάρτησης αποτίμησης `matching_evaluation` μπορείτε να βρείτε στο αρχείο με όνομα `test_matching_evaluation.py`.
- 3.1.2 Για κάθε έναν από τους 5 ανιχνευτές σημείων ενδιαφέροντος που υλοποιήσατε (χρησιμοποιήστε μόνο την πολυχλιμακωτή έκδοση της μεθόδου των Box Filters) και για τους δύο τοπικούς περιγραφητές (συνολικά 10 συνδυασμοί), να υπολογίσετε και να σχολιάσετε την ικανότητα εκτίμησης της περιστροφής και κλίμακας των εικόνων για κάθε συνδυασμό.

3.2. Κατηγοριοποίηση Εικόνων

Στο μέρος αυτό θα αξιολογηθεί η επίδοση και η καταλληλότητα των διαφόρων ανιχνευτών και περιγραφητών σε ένα τυπικό πρόβλημα κατηγοριοποίησης εικόνων. Συγκεκριμένα θα σας δοθεί ένα σύνολο εικόνων από τη βάση Pascal VOC2005. Κάθε εικόνα ανήκει σε μια από τις τρεις ακόλουθες κλάσεις: αυτοκίνητο, άνθρωπος και ποδήλατο. Σκοπός είναι η κατηγοριοποίηση της κάθε εικόνας στη σωστή κλάση χρησιμοποιώντας σαν χαρακτηριστικά αναγνώρισης τους περιγραφητές που θα κατασκευάσετε.

- 3.2.1 Αρχικά για κάθε εικόνα και από κάθε κλάση θα πρέπει να εξάγετε τα χαρακτηριστικά με βάση τα οποία θα γίνει η κατηγοριοποίηση. Χρησιμοποιήστε τη `FeatureExtraction` συνάρτηση που σας δίνεται προκειμένου να εξάγετε τα χαρακτηριστικά για όλη τη βάση. Για την χρήση των διαφορετικών ανιχνευτών/ περιγραφητών χρησιμοποιείστε ανώνυμες συναρτήσεις όπως και στο προηγούμενο μέρος. Πειραματιστείτε μόνο με τις πολυχλιμακωτές εκδόσεις των περιγραφητών.
- 3.2.2 Στη συνέχεια θα πρέπει να γίνει ο διαχωρισμός των εικόνων στα σύνολα `train` και `test` καθώς και η δημιουργεί ετικετών (`label`) που θα δείχνουν την κλάση στην οποία θα ανήκει κάθε εικόνα. Χρησιμοποιήστε τη συνάρτηση `createTrainTest` που σας δίνεται προκειμένου να κάνετε αυτόν τον διαχωρισμό χρησιμοποιώντας σαν όρισμα τα χαρακτηριστικά που υπολογίσατε στο προηγούμενο ερώτημα.
- 3.2.3 Κατασκευή αναπαράστασης Bag of Visual Words (**Προαιρετικό για τους προπτυχιακούς, υποχρεωτικό για τους μεταπτυχιακούς¹**)

¹Οσοι προπτυχιακοί επιλέξουν να υλοποιήσουν μόνοι τους το Μέρος 3.2.3, θα έχουν bonus 10% επί αυτής της εργαστηριακής άσκησης.

Μέχρι τώρα είδαμε πως μπορούμε να εξάγουμε τοπικούς περιγραφητές γύρω από σημεία ενδιαφέροντος. Ωστόσο για το πρόβλημα της αναγνώρισης χρειαζόμαστε ένα συνολικό (global) διάνυσμα χαρακτηριστικών με την ίδια διάσταση για κάθε εικόνα. Για το λόγο αυτό αντί να χρησιμοποιήσουμε απευθείας τους τοπικούς περιγραφητές που εξάγαμε θα υπολογίσουμε ιστογράμματα εμφάνισης τους στο σύνολο της εικόνας. Τα βήματα για την κατασκευή της BoVW αναπαράστασης συνοψίζονται παρακάτω:

- Δημιουργία του οπτικού λεξικού βάση του οποίου θα υπολογιστούν τα ιστογράμματα.
Για τον προσδιορισμό των οπτικών λέξεων του λεξικού θα εφαρμοστεί ο αλγόριθμος συσταδοποίησης kmeans για το σύνολο των περιγραφητών που αντιστοιχούν στο σύνολο εκπαίδευσης. Αρχικά θα πρέπει να γίνει η συνένωση όλων των περιγραφητών του συνόλου εκπαίδευσης σε ένα διάνυσμα χαρακτηριστικών ανεξάρτητα από την κλάση στην οποία ανήκουν. Στη συνέχεια εφαρμόστε τον kmeans αλγόριθμο για ένα τυχαίο υποσύνολο του παραπάνω διανύσματος (της τάξης του 50% του συνολικού μεγέθους) και για αριθμό κέντρων 500-2000.
- Για κάθε εικόνα του συνόλου train και test υπολογίστε την ελάχιστη ευκλείδια απόσταση του κάθε τοπικού περιγραφητή από τα κέντρα που υπολογίστηκαν στο προηγούμενο ερώτημα. Κατασκευάστε για κάθε εικόνα το ιστόγραμμα που προκύπτει από τη συχνότητα εμφάνισης των οπτικών λέξεων του λεξικού με βάση την παραπάνω απόσταση (Συνάρτηση histc).
- Κανονικοποιήστε κάθε ιστόγραμμα που προκύπτει με βάση την L2 νόρμα.

Για τους προπτυχιακούς φοιτήτες που δεν θα επιλέξουν να υλοποιήσουν την BoVW αναπαράσταση η όλη διαδικασία μπορεί να γίνει με τη συνάρτηση BagOfWords που σας δίνεται.

3.2.4 Το τελικό στάδιο αποτελείται από την τελική κατηγοριοποίηση των εικόνων με βάση την BoVW αναπαράσταση. Για την κατηγοριοποίηση χρησιμοποιείται ενάς SVM Support Vector Machine ταξινομητής κατάλληλα προσαρμοσμένος για πολλαπλές κλάσεις. Η όλη διαδικασία υλοποιείται με τη συνάρτηση svm, η οποία επιστρέφει το αποτέλεσμα της αναγνώρισης καθώς και το συνολικό ποσοστό επιτυχίας. Πειραματιστείτε με τους διαφορετικούς ανιχνευτές/περιγραφητές και σχολιάστε πως αλλάζουν τα ποσοστά αναγνώρισης. Ποιο συνδυασμό τους θα προτείνατε τελικά για το πρόβλημα της κατηγοριοποίησης εικόνων;

ΠΑΡΑΔΟΤΕΑ: Image plots (σχήματα εικόνων) από όλα τα κύρια στάδια, Python κώδικας, αποτελέσματα συγκρίσεων, συνοπτική επεξήγηση των αλγορίθμων και σχολιασμός των αποτελεσμάτων. Έπειτα από την παράδοση της γραπτής εργασίας, θα ακολουθήσει μια προφορική συνοπτική εξέταση.

Σημείωση 1: Σας προτείναμε διάφορους αλγορίθμους για διάφορα στάδια της άσκησης με διάφορες τιμές των παραμέτρων. Αυτά είναι μόνο ενδεικτικά. Μπορείτε να πειραματισθείτε με άλλες τιμές και άλλους τρόπους λύσης.

Σημείωση 2: Αν η αρχική intensity εικόνα έχει τιμές $I(x) \in [0, 1]$, και f είναι η εικόνα εξόδου από ένα γραμμικό φίλτρο, υπάρχει περίπτωση το πεδίο τιμών της f να υπερβαίνει το αρχικό $[0, 1]$. Η εντολή “imshow(f)” της matplotlib.pyplot εξόρισμού απεικονίζει τη συνάρτηση f ως

εικόνα στην οθόνη με κανονικοποιημένο πεδίο τιμών $[0, 1]$. Με την εντολή “`imshow(f, vmin=0, vmax=1)`” μπορείτε πάλι να την απεικονίσετε στο πεδίο αυτό, αλλά αυτή τη φορά θέτοντας ένα όριο στις τιμές που μπορούν να αναπαρασταθούν.

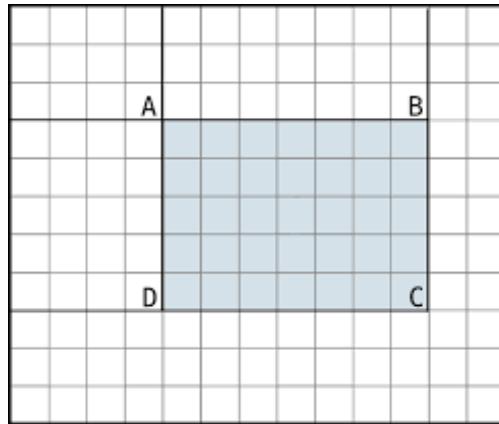
Σημείωση 3: Η ακολουθία Python εντολών

```
import matplotlib.pyplot as plt
...
plt.subplot(1,2,1)
plt.imshow(I1)
plt.title('title 1')
plt.subplot(2,1,2)
plt.imshow(I2)
plt.title('title 2')
plt.savefig('filename.jpg')
```

δείχνει 2 εικόνες σε 1 παράθυρο, προσθέτει τίτλους, και αποθηκεύει ολόκληρο το σχήμα στο αντίστοιχο .jpg αρχείο.

Σημείωση 4: Για την παράδοση της εργασίας σας, αποθηκεύστε σε έναν φάκελο όλα τα ζητούμενα και αφού τον συμπλέσετε, ανεβάστε το αρχείο με την ονομασία προσαρμοσμένη κατάλληλα στα στοιχεία σας `cv24_Lab1_{Part Or Total}_Name1_AM1_Name2_AM2.zip`.

Παράρτημα: Ολοκληρωτικές Εικόνες (Integral Images)



Σχήμα 3: Παράθυρο Υπολογισμού Αθροίσματος

Θεωρώντας την εικόνα $I(x, y)$, μεγέθους $N \times M$, η προκύπτουσα ολοκληρωτική εικόνα ορίζεται ως:

$$S(x, y) = \sum_{i \leq x} \sum_{j \leq y} I(i, j) \quad (16)$$

δηλαδή κάθε στοιχείο της ολοκληρωτικής εικόνας ισούται με το άθροισμα όλων των στοιχειών της αρχικής εικόνας I που βρίσκονται πάνω και αριστερά από το στοιχείο αυτό.

Η κατασκευή της ολοκληρωτικής εικόνας, ξεκινώντας από πάνω αριστερά, γίνεται με εφαρμογή δυναμικού προγραμματισμού, σύμφωνα με την παρατήρηση ότι κάθε νέο στοιχείο της ολοκληρωτικής εικόνας $S(i, j)$ μπορεί να υπολογιστεί απευθείας με την χρήση των γειτονικών του, ήδη

υπολογισθέντων, στοιχείων $S(i, j - 1)$, $S(i - 1, j)$ και $S(i - 1, j - 1)$:

$$S(i, j) = I(i, j) - S(i - 1, j - 1) + S(i, j - 1) + S(i - 1, j), \quad i \in [1, N], j \in [1, M] \quad (17)$$

Η Εξ. (17) σε συνδυασμό με την αρχικοποίηση $S(i, 0) = 0$, $i \in [0, N]$ και $S(0, j) = 0$, $j \in [0, M]$ συνθέτουν έναν αποδοτικό αλγόριθμο εύρεσης της ολοκληρωτικής εικόνας με πολυπλοκότητα $O(NM)$.

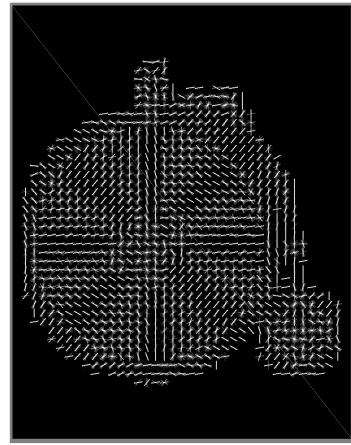
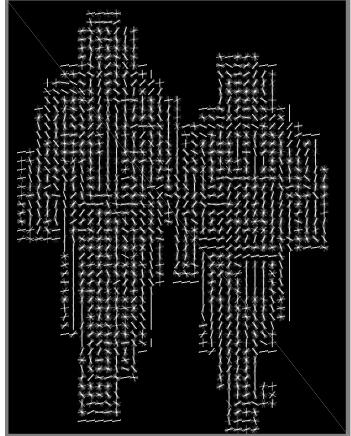
Με την χρήση της ολοκληρωτικής εικόνας οποιοδήποτε άθροισμα εντός ενός δοθέντος παραθύρου πάνω στην αρχική εικόνα μπορεί να υπολογιστεί άμεσα (με πολυπλοκότητα $O(1)$) με παρόμοια λογική με αυτή της κατασκευής της ολοκληρωτικής εικόνας. Συγκεκριμένα το άθροισμα των στοιχείων εντός του παραθύρου του Σχήματος 3, που οριοθετείται από τα σημεία A, B, C και D , υπολογίζεται με χρήση της ολοκληρωτικής εικόνας σύμφωνα με την Εξ. (18) και ισοδυναμεί με τον υπολογισμό του εμβαδού του σκιασμένου κομματιού της εικόνας.

$$\sum \sum_{ABCD} I(i, j) = S_A + S_C - S_B - S_D \quad (18)$$

Σύμφωνα με την παραπάνω ανάλυση οποιοδήποτε φίλτρο βασίζεται σε υπολογισμό άθροισμάτων εντός ορθογώνιων παραθύρων (π.χ. φίλτρο μέσης ή φίλτρο διακύμανσης) μπορεί να υπολογιστεί πολύ εύκολα και αποδοτικά με την χρήση των ολοκληρωτικών εικόνων με πολυπλοκότητα ανεξάρτητη του μεγέθους του παραθύρου του φίλτρου (σε αντίθεση με τεχνικές συνέλιξης).

Παράρτημα: Ιστογράμματα Προσανολισμένης Κλίσης Histogram of Oriented Gradients HOGs)

Τα Ιστογράμματα Προσανατολισμένης Κλίσης (Histograms of Oriented Gradients “HOGs”) αποτελούν ένα σύνολο χαρακτηριστικών για την περιγραφή της δομής του σχήματος σε μια εικόνα και έχουν χρησιμοποιηθεί με μεγάλη επιτυχία στην αναγνώριση των αντικειμένων, όπως το ανθρώπινο σώμα, που περιέχονται σε μια εικόνα [2]. Τα HOGs παρέχουν μια πυκνή επικαλυπτόμενη περιγραφή των περιοχών μια εικόνας και υπολογίζονται σε ένα πυκνό πλέγμα ομοιόμορφα κατανευμημένων κελιών. Με τον τρόπο αυτό αναδεικνύουν την πληροφορία για το τοπικό σχήμα κωδικοποιώντας τις κατευθύνσεις της κλίσης της εικόνας σε ιστογράμματα. Τέλος, επιτυγχάνουν να είναι αρκετά αμετάβλητα ως προς τις αλλαγές της φωτεινότητας εφαρμόζοντας μια τοπική κανονικοποίηση των ιστογραμμάτων σε επικαλυπτόμενες περιοχές. Υπάρχουν αρκετά είδη HOGs που χρησιμοποιούνται στην πράξη ανάλογα με τον τρόπο που γίνεται ο διαχωρισμός του πλέγματος κελιών και του βαθμού της επικάλυψης. Ωστόσο, τα βασικά στάδια του υπολογισμού της κλίσης στα σημεία της εικόνας και του ορισμού των διευθύνσεων βάσει των οπίων θα υπολογιστούν τα ιστογράμματα είναι τα ίδια για όλους τους τύπους HOGs. Το αρχικό βήμα αφορά τον υπολογισμό των gradients σε κάθε pixel της εικόνας. Για το σκοπό αυτό, συνήθως χρησιμοποιούνται μονοδιάστατες “μάσκες” διακριτών παραγώγων $g_x = [-1 \ 0 \ 1]$ και $g_y = [-1 \ 0 \ 1]^T$. Εναλλακτικά, μπορούν να χρησιμοποιηθούν και άλλες παραλλαγές για τις μάσκες των διακριτών παραγώγων, όπως cubic-corrected, Prewitt και Sobel. Φιλτράροντας την εικόνα με τους πυρήνες g_x και g_y προκύπτουν τα προσανατολισμένα gradients I_x και I_y της εικόνας, στην οριζόντια και κάθετη διεύθυνση αντίστοιχα.



Σχήμα 4: Παραδείγματα οπτικοποίησης του Ιστογραφικού Περιγραφητή HOG.

Στη συνέχεια, χρησιμοποιώντας τα προσανατολισμένα gradients, υπολογίζεται το μέτρο και την γωνία των gradients σε κάθε pixel της εικόνας, αξιοποιώντας τις παρακάτω σχέσεις:

- Μέτρο: $Magnitude(x, y) = \sqrt{I_x^2 + I_y^2}$
- Γωνία: $Angle(x, y) = \arctan \frac{I_y}{I_x}$

Στο επόμενο βήμα η εικόνα χωρίζεται σε μη επικαλυπτόμενες περιοχές π.χ. 5×5 pixels, τις οποίες θα ονομάζουμε κελιά (cells). Το ποσοστό επικάλυψης των γειτονικών κελιών επιλέγεται ως μια επιπλέον παραμετρό r , έτσι ώστε το κελί να επεκταθεί κατά $pad_x = r \times cell_size_x$ αριστερά και δεξιά του αρχικού κελιού και αντίστοιχα κατά $pad_y = r \times cell_size_y$ πάνω και κάτω. Το τελικό μέγεθος των κελιών θα είναι $(cell_size_x + 2 \cdot pad_x) \times (cell_size_y + 2 \cdot pad_y)$ pixels. Ο υπολογισμός του τοπικού ιστογράμματος περιγράφεται αναλυτικά στη συνέχεια.

Το εύρος γωνιών $0^\circ - 180^\circ$ χωρίζεται ομοιόμορφα σε 9 τμήματα, ώστε κάθε ένα από αυτά να έχει εύρος $180^\circ / 9 = 20^\circ$. Κάθε τμήμα αντιστοιχεί σε μία ράβδο (bin) του τοπικού ιστογράμματος που θα κατασκευάσετε. Κάθε pixel εντός του κελιού “ψηφίζει” για την κατασκευή του

ιστογράμματος ως εξής:

(Σ1) Η “ψήφος” δίνεται στην ράβδο στην οποία αντιστοιχεί η γωνία του gradient.

(Σ2) Η τιμή της “ψήφου” ισούται με το μέτρο του gradient.

Εφόσον όλα τα pixels έχουν “ψηφίσει”, η τιμή κάθε ράβδου του ιστογράμματος διαφείται με την $L2$ ή $L1$ νόρμα, ώστε να κανονικοποιηθούν οι τιμές του τοπικού ιστογράμματος. Ο τελικός περιγραφητής αντιστοιχεί στο καθολικό ιστόγραμμα που προκύπτει από την συνένωση (concatenation) όλων των τοπικών ιστογραμάτων.

References

- [1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.
- [2] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005.
- [3] Μαραγκός Π. *Ανάλυση Εικόνων και Όραση Υπολογιστών*. Ε.Μ.Π., 2014.
- [4] C. Harris and M. Stephens. A combined corner and edge detector. *Alvey Vision Conference*, 15:50, 1988.
- [5] B. Jähne. *Digital Image Processing*, chapter 13. Springer, 2002.
- [6] Tony Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.
- [7] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE, 2001.