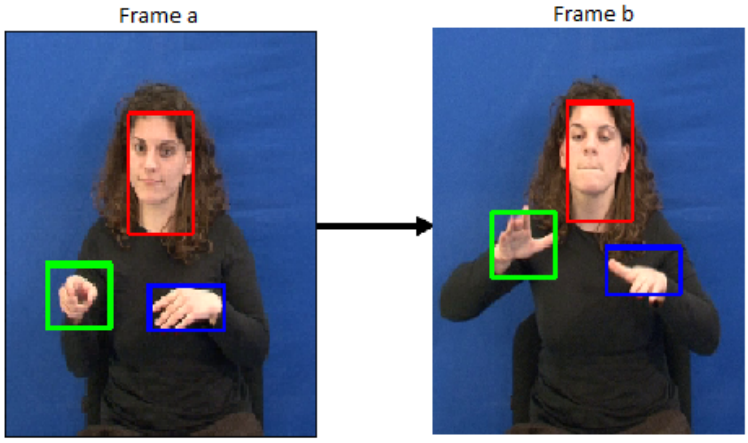
Ε.Μ.Π. – Σ.Η.Μ.Μ.Υ. **ΟΡΑΣΗ ΥΠΟΛΟΓΙΣΤΩΝ** Ακαδ.Έτος 2023–**2024**
Ημερ/νία: 10-04-2024 **2ο Εργαστήριο** Παράδοση: 13-05-2024

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 στη σελίδα του μαθήματος στο HELIOS (https://helios.ntua.gr/course/view.php?id=964). Στη σελίδα αυτή, στην ενότητα 'Απορίες Εργαστηρίων', μπορείτε επίσης να υποβάλετε 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: Εκτίμηση Οπτικής Ροής (Optical Flow), Εξαγωγή Χαρακτηριστικών σε Βίντεο για Αναγνώριση Δράσεων, Συνένωση Εικόνων (Image Stitching)

Μέρος 1: Παρακολούθηση Προσώπου και Χεριών με Χρήση της Μεθόδου Οπτικής Ροής των Lucas-Kanade

Σκοπός του Μέρους 1 του εργαστηρίου είναι η υλοποίηση ενός συστήματος παρακολούθησης προσώπου και χεριών (Face and Hands Tracking) σε μια ακολουθία βίντεο νοηματικής γλώσσας. Το σύστημα αρχικά θα ανιχνεύει στο πρώτο πλαίσιο την περιοχή του προσώπου και των χεριών (περιοχές ενδιαφέροντος) με χρήση ενός πιθανοτικού ανιχνευτή ανθρώπινου δέρματος. Στη συνέχεια θα μπορεί να παρακολουθεί τις περιοχές ενδιαφέροντος χρησιμοποιώντας τα εξαγόμενα διανύσματα οπτικής ροής, υπολογισμένα με τη μέθοδο των Lucas-Kanade.



Σχήμα 1: Παράδειγμα παρακολούθησης προσώπου-χεριών σε ακολουθία βίντεο νοηματικής γλώσσας.

1.1 Ανίχνευση Δέρματος Προσώπου και Χεριών

(α) (β) (γ)

Σχήμα 2: Παράδειγμα ανίχνευσης δέρματος σε βίντεο νοηματικής γλώσσας: (α) Ανίχνευση σημείων δέρματος. (β) Σχηματισμός συνεκτικών περιοχών. (γ) Τελική ανίχνευση περιοχής προσώπου.

Στο πρώτο ερώτημα ζητείται η ανίχνευση σημείων δέρματος στο πρώτο πλαίσιο της ακολουθίας και η τελική επιλογή της περιοχής του προσώπου και των χεριών. Για την ανίχνευση των σημείων δέρματος χρησιμοποιείται ο χρωματικός χώρος YCbCr, αφαιρώντας την πληροφορία της φωτεινότητας Y και διατηρώντας τα κανάλια Cb και Cr που περιγράφουν την ταυτότητα του χρώματος. Το χρώμα του δέρματος μοντελοποιείται με μια διδιάστατη Γκαουσιανή κατανομή:

$$P(\mathbf{c} = \text{skin}) = \frac{1}{\sqrt{|\Sigma|\,(2\pi)^2}} e^{-\frac{1}{2}(\mathbf{c}-\boldsymbol{\mu})\Sigma^{-1}(\mathbf{c}-\boldsymbol{\mu})'} \tag{1}$$

όπου $\mathbf{c}$ είναι το διάνυσμα τιμών Cb και Cr για κάθε σημείο (x, y) της εικόνας. Η Γκαουσιανή κατανομή εκπαιδεύεται υπολογίζοντας το 2×1 διάνυσμα μέσης τιμής $\boldsymbol{\mu} = [\mu_{C_b}\ \mu_{C_r}]^\top$ και τον 2×2 πίνακα συνδιακύμανσης Σ από τα δείγματα δέρματος που δίνονται στο αρχείο `skinSamplesRGB.mat` σε μορφή RGB. Η δυαδική εικόνα ανίχνευσης δέρματος προκύπτει από την εικόνα πιθανοτήτων $P(\mathbf{c}(x, y) = \text{skin})$, $\forall\,(x, y)$ με κατωφλιοποίηση. Ενδεικτικές τιμές κατωφλίου: στο διάστημα $[0.05, 0.25]$ (για τιμές πιθανοτήτων $[0, 1]$).

Για την τελική ανίχνευση των περιοχών δέρματος του προσώπου και των χεριών απαιτείται μια μορφολογική επεξεργασία της δυαδικής εικόνας δέρματος. Συγκεκριμένα, θα γίνει κάλυψη των τρυπών που εμφανίζονται, εφαρμόζοντας opening με ένα πολύ μικρό δομικό στοιχείο και closing με ένα μεγάλο δομικό στοιχείο. Έτσι θα εξαλειφθούν οι μικρές περιοχές και θα αποκτήσουν συνοχή οι περιοχές του προσώπου και των χεριών. Τέλος, θα δημιουργήσετε τρία ορθογώνια που θα περιβάλλουν τις περιοχές ενδιαφέροντος-δέρματος (bounding boxes) και θα χρησιμοποιηθούν στο 1.2 για τον υπολογισμό των διανυσμάτων Οπτικής Ροής και την τελική παρακολούθηση του προσώπου και των χεριών.

Υλοποιήστε την παραπάνω διαδικασία σε Python ως αυτόνομη συνάρτηση που να δέχεται ως εισόδους μια εικόνα (την πρώτη της ακολουθίας βίντεο), τη μέση τιμή $\boldsymbol{\mu}$ και τη συνδιακύμανση Σ της Γκαουσιανής κατανομής και να επιστρέφει το πλαίσιο οριοθέτησης της περιοχής ενδιαφέροντος στη μορφή `[x, y, width, height]`, όπου `x, y` οι συντεταγμένες του πάνω αριστερά σημείου, π.χ.

```
boundingBox = fd(I, mu, cov)
```

▶ Βοήθεια για Python: συναρτήσεις (scipy.stats) multivariate_normal, (cv2) morphologyEx, (scipy.ndimage) label.

1.2 Παρακολούθηση Προσώπου και Χεριών

Η αρχικοποίηση των Bounding Boxes που περιλαμβάνει το πρόσωπο και τα χέρια της νοηματίστριας στη μορφή `[x, y, width, height]` είναι η εξής:
Πρόσωπο: $[154, 102, 67, 115]$, Αριστερό χέρι: $[93, 272, 56, 83]$, Δεξί χέρι: $[201, 270, 56, 83]$
Τα Bounding Boxes είναι διευρυμένα, σχετικά με το αποτέλεσμα της ανίχνευσης του δέρματος, για να διευκολύνουν τη διαδικασία της παρακολούθησης. Οι παραπάνω διαστάσεις μπορούν να χρησιμοποιηθούν ανεξάρτητα από το αν έχει υλοποιηθεί το Μέρος 1.1 του εργαστηρίου.
Καλείστε να υλοποιήσετε τους αλγορίθμους, όπως περιγράφονται παρακάτω, και να τους εφαρμόσετε για το πρόσωπο και για τα χέρια της νοηματίστριας. Το αν θα τις εφαρμόσετε για κάποιες/όλες τις περιοχές μαζί ή ξεχωριστά, αφήνεται σε εσάς.

1.2.1 Υλοποίηση του Αλγόριθμου των Lucas-Kanade

Σε μια ακολουθία εικόνων N frames $I_n(\mathbf{x})$, όπου $n = 1, \ldots, N$ και $\mathbf{x} = (x, y)$, το πεδίο οπτικής ροής $-\mathbf{d}$, όπου $\mathbf{d}(\mathbf{x}) = (d_x, d_y)$, φέρνει σε αντιστοιχία δύο διαδοχικές εικόνες, έτσι ώστε

$$I_n(\mathbf{x}) \approx I_{n-1}(\mathbf{x} + \mathbf{d}) \tag{2}$$

Ο αλγόριθμος των Lucas-Kanade υπολογίζει την οπτική ροή σε κάθε σημείο της εικόνας $\mathbf{x}$ με τη μέθοδο των ελάχιστων τετραγώνων, θεωρώντας ότι το $\mathbf{d}$ είναι σταθερό σε ένα μικρό παράθυρο γύρω από το σημείο και ελαχιστοποιώντας το τετραγωνικό σφάλμα

$$J_{\mathbf{x}}(\mathbf{d}) = \int_{\mathbf{x}' \in \mathbb{R}^2} G_\rho(\mathbf{x} - \mathbf{x}') \left[I_n(\mathbf{x}') - I_{n-1}(\mathbf{x}' + \mathbf{d}) \right]^2 \mathrm{d}\mathbf{x}', \tag{3}$$

όπου $G_\rho(\mathbf{x})$ είναι μια συνάρτηση παραθύρωσης, π.χ. Γκαουσιανή με τυπική απόκλιση ρ.

Θεωρούμε ότι έχουμε μια εκτίμηση $\mathbf{d}_i$ για το $\mathbf{d}$ και προσπαθούμε να τη βελτιώσουμε κατά $\mathbf{u}$, δηλαδή $\mathbf{d}_{i+1} = \mathbf{d}_i + \mathbf{u}$. Αναπτύσσοντας κατά Taylor την έκφραση $I_{n-1}(\mathbf{x}+\mathbf{d}) = I_{n-1}(\mathbf{x}+\mathbf{d}_i+\mathbf{u})$ γύρω από το σημείο $\mathbf{x} + \mathbf{d}_i$, προκύπτει ότι

$$I_{n-1}(\mathbf{x} + \mathbf{d}) \approx I_{n-1}(\mathbf{x} + \mathbf{d}_i) + \nabla I_{n-1}(\mathbf{x} + \mathbf{d}_i)^T \mathbf{u} \tag{4}$$

Βάζοντας αυτήν την έκφραση στην Εξ. (3) μπορεί να δειχθεί ότι η λύση ελάχιστων τετραγώνων για τη βελτίωση της εκτίμησης της οπτικής ροής σε κάθε σημείο είναι

$$\mathbf{u}(\mathbf{x}) = \begin{bmatrix} (G_\rho * A_1^2)(\mathbf{x}) + \epsilon & (G_\rho * (A_1 A_2))(\mathbf{x}) \\ (G_\rho * (A_1 A_2))(\mathbf{x}) & (G_\rho * A_2^2)(\mathbf{x}) + \epsilon \end{bmatrix}^{-1} \cdot \begin{bmatrix} (G_\rho * (A_1 E))(\mathbf{x}) \\ (G_\rho * (A_2 E))(\mathbf{x}) \end{bmatrix} \tag{5}$$

όπου

$$A(\mathbf{x}) = \begin{bmatrix} A_1(\mathbf{x}) & A_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial I_{n-1}(\mathbf{x} + \mathbf{d}_i)}{\partial x} & \dfrac{\partial I_{n-1}(\mathbf{x} + \mathbf{d}_i)}{\partial y} \end{bmatrix} \tag{6}$$

$$E(\mathbf{x}) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{d}_i) \tag{7}$$

και το $*$ δηλώνει συνέλιξη. Η μικρή θετική σταθερά ϵ βελτιώνει το αποτέλεσμα σε επίπεδες περιοχές με μειωμένη υφή και άρα μειωμένη πληροφορία για τον υπολογισμό της οπτικής ροής. Η ανανέωση του διανύσματος οπτικής ροής $\mathbf{d}_{i+1} = \mathbf{d}_i + \mathbf{u}$, με το $\mathbf{u}$ να υπολογίζεται από την Εξ. (5), επαναλαμβάνεται αρκετές φορές ως τη σύγκλιση.

Υλοποιήστε τον αλγόριθμο των Lucas-Kanade σε Python. Ο αλγόριθμος να υλοποιηθεί ως αυτόνομη συνάρτηση, που να δέχεται ως εισόδους δύο εικόνες (κομμένα παράθυρα με βάση το bounding box από δύο διαδοχικά πλαίσια του βίντεο), ένα σύνολο από σημεία ενδιαφέροντος

(γωνίες) εντός του παραθύρου, το εύρος ρ του γκαουσιανού παραθύρου, τη θετική σταθερά κανονικοποίησης ϵ, και την αρχική εκτίμηση $\mathbf{d}_0$ για το πεδίο οπτικής ροής, και να επιστρέφει το $\mathbf{d}$, π.χ.

```
[d_x, d_y] = lk(I1, I2, features, rho, epsilon, d_x0, d_y0)
```

Αφού υλοποιήσετε τη δική σας συνάρτηση `lk()`, χρησιμοποιήστε την έτοιμη υλοποίηση που προσφέρει η βιβλιοθήκη της OpenCV για τον υπολογισμό του πυκνού πεδίου οπτικής ροής TV-L1 [9] (βλ. και ενότητα 2.2.2 για χρήσιμες οδηγίες εκτέλεσής της) ώστε να συγκρίνετε τα αποτελέσματα που δίνουν οι δύο μέθοδοι στα σημεία ενδιαφέροντος παρέχοντας κάποιες ενδεικτικές εικόνες, αποτελέσματα και σχολιασμό.
Χρήσιμες οδηγίες:

- Για τον υπολογισμό της $I_{n-1}(\mathbf{x}+\mathbf{d}_i)$ και των $\frac{\partial I_{n-1}}{\partial x}(\mathbf{x}+\mathbf{d}_i)$, $\frac{\partial I_{n-1}}{\partial y}(\mathbf{x}+\mathbf{d}_i)$ θα χρειαστείτε τις τιμές της I_{n-1} και των μερικών παραγώγων της σε ενδιάμεσα σημεία του πλέγματος. Για να το πετύχετε αυτό, θα μπορούσατε να χρησιμοποιήσετε εντολές όπως η map_coordinates(I1,[numpy.ravel(y_0+dy_i), numpy.ravel(x_0+dx_i)], order=1), όπου x_0,y_0 = numpy.meshgrid(I1.shape[1],I1.shape[0]), και όμοια για τις μερικές παραγώγους (προσοχή στη σειρά των ορισμάτων στις παραπάνω συναρτήσεις).

- Η διαδικασία $\mathbf{d}_{i+1} = \mathbf{d}_i + \mathbf{u}$ συγκλίνει μετά από αρκετές επαναλήψεις (συνήθως πάνω από 100 ανάλογα με το σημείο). Πειραματιστείτε με εναλλακτικά κριτήρια σύγκλισης (π.χ. η διαφορά δύο επαναλήψεων να πέφτει κάτω από ένα κατώφλι).

- Για την εξαγωγή των features μπορείτε να χρησιμοποιήσετε την `cv2.goodFeaturesToTrack`, που υλοποιεί την μέθοδο των Shi και Tomashi [10].

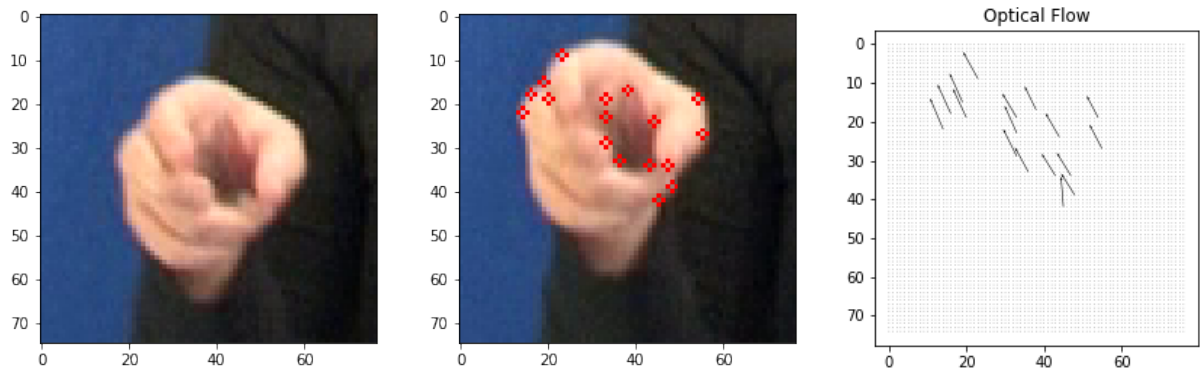
- Ενδεικτικές τιμές παραμέτρων: $\rho \in [1, 5]$ pixels και $\epsilon \in [0.01, 0.1]$ (για τιμές της εικόνας στο $[0, 1]$). Πειραματιστείτε με πολύ διαφορετικές τιμές των παραμέτρων ρ και ϵ και σχολιάστε πώς το αποτέλεσμα αλλάζει.

- Για να απεικονίσετε τα διανύσματα $\mathbf{d}$ όπως στο Σχ. 3(γ) χρησιμοποιήστε τη συνάρτηση `quiver(-d_x_r,-d_y_r,angles='xy',scale=100)`.

▶ Βοήθεια για Python: συναρτήσεις (numpy) `meshgrid`, (cv2) `getGaussianKernel`, `filter2D`, (scipy.ndimage) `map_coordinates`, (matplotlib.pyplot) `quiver`.

1.2.2 Υπολογισμός της Μετατόπισης των Παραθύρων από τα Διανύσματα Οπτικής Ροής

Έχοντας υπολογίσει την οπτική ροή της εικόνας I_n στα σημεία που ορίζουν τα σημεία ενδιαφέροντος εντός του bounding box της εικόνας I_{n-1}, απομένει να βρούμε το συνολικό διάνυσμα μετατόπισης του bounding box ορθογωνίου, με όσο το δυνατόν μεγαλύτερη ακρίβεια. Γνωρίζουμε ότι τα διανύσματα οπτικής ροής έχουν κατά κανόνα μεγαλύτερο μήκος σε σημεία που ανήκουν σε περιοχές με έντονη πληροφορία υφής (π.χ. ακμές, κορυφές) και σχεδόν μηδενικό μήκος σε σημεία που ανήκουν σε περιοχές με ομοιόμορφη και επίπεδη υφή. Έτσι, καθώς η πλειονότητα των σημείων ενδιαφέροντος (γωνίες) βρίσκονται σε σημεία με έντονη υφή, θα μπορούσαμε απλώς να χρησιμοποιήσουμε τη μέση τιμή των διανυσμάτων μετατόπισης. Όμως, για να πετύχουμε καλύτερη ακρίβεια ή για να απορρίψουμε outliers, θα μπορούσαμε να εφαρμόσουμε εναλλακτικά κριτήρια, όπως για παράδειγμα να υπολογίσουμε τη μέση τιμή των διανυσμάτων μετατόπισης που έχουν ενέργεια μεγαλύτερη από μια τιμή κατωφλίου. Ως ενέργεια διανύσματος ταχύτητας ορίζουμε $\|\mathbf{d}\|^2 = {d_x}^2 + {d_y}^2$.
Υλοποιήστε συνάρτηση που θα δέχεται σαν είσοδο τα διανύσματα της οπτικής ροής και θα υπολογίζει το τελικό διάνυσμα μετατόπισης του ορθογωνίου, π.χ.



(α) (β) (γ)

Σχήμα 3: (α,β) Δύο διαδοχικά καρέ της ακολουθίας βίντεο νοηματικής γλώσσας. Στο δεύτερο καρέ φαίνονται και επισημειωμένα τα σημεία ενδιαφέροντος. (γ) Διάγραμμα των διανυσμάτων οπτικής ροής στα σημεία ενδιαφέροντος, απ΄ όπου φαίνεται η κίνηση του χεριού προς τα πάνω και αριστερά.

```
[displ_x, displ_y] = displ(d_x, d_y)
```

Εκτελέστε το συνολικό σύστημα παρακολούθησης προσώπου και χεριών για την ακολουθία εικόνων βίντεο που περιέχονται στον φάκελο `Part 1` του συμπληρωματικού υλικού της άσκησης. Πρόκειται για βίντεο ελληνικής νοηματικής γλώσσας γυρισμένο σε στούντιο με ελεγχόμενο φωτισμό. Πειραματιστείτε με διαφορετικές τιμές των παραμέτρων ρ, ϵ και κατωφλίου ενέργειας οπτικής ροής και παρατηρήστε πώς επηρεάζεται το τελικό αποτέλεσμα. Πειραματιστείτε με εναλλακτικά κριτήρια υπολογισμού της μετατόπισης από τα διανύσματα οπτικής ροής. Πειραματιστείτε με εναλλακτική μέθοδο οπτικής ροής (π.χ. TV-L1) και σχολιάστε αν βελτιώνεται το τελικό αποτέλεσμα.

1.2.3 Πολυ-Κλιμακωτός Υπολογισμός Οπτικής Ροής

Υλοποιήστε την πολυ-κλιμακωτή εκδοχή του αλγόριθμου των Lucas-Kanade. Ο αλγόριθμος θα αναλύει τις αρχικές εικόνες σε γκαουσιανές πυραμίδες και θα υπολογίζει την οπτική ροή από τις πιο μικρές (τραχείς) στις πιο μεγάλες (λεπτομερείς) κλίμακες, χρησιμοποιώντας τη λύση της μικρής κλίμακας ως αρχική συνθήκη για τη μεγάλη κλίμακα. Ο αλγόριθμος να υλοποιηθεί ως αυτόνομη συνάρτηση, παρόμοια με πριν, αλλά να δέχεται επίσης ως είσοδο τον αριθμό των κλιμάκων της πυραμίδας, και να χρησιμοποιεί τον αλγόριθμο των Lucas-Kanade μονής κλίμακας ως υπο-ρουτίνα. Χρήσιμες οδηγίες:

- Για τη μετάβαση από μεγάλες σε μικρές κλίμακες κατά την κατασκευή της γκαουσιανής πυραμίδας φιλτράρετε την εικόνα με βαθυπερατό φίλτρο (π.χ. γκαουσιανή τυπικής απόκλισης 3 pixels) πριν την υποδειγματοληψία για να μετριάσετε τη φασματική αναδίπλωση (aliasing) της εικόνας.
- Κατά τη μεταφορά του $\mathbf{d}$ από μικρές σε μεγάλες κλίμακες μην ξεχάσετε να το διπλασιάσετε.

Τρέξτε στη συνέχεια τον πολυ-κλιμακωτό αλγόριθμό σας στην ίδια ακολουθία εικόνων και σχολιάστε τις διαφορές που παρατηρείτε στην ταχύτητα σύγκλισης και στην ποιότητα του αποτελέσματος σε σχέση με τον αλγόριθμο μονής κλίμακας. Το ολικό displacement $\mathbf{d}$ για το bounding box υπολογίζεται όμοια με το 1.2.2.

ΠΑΡΑΔΟΤΕΑ: Ζητείται να παραδώσετε τα εξής:

- Image plots (σχήματα εικόνων) για όλα τα βήματα της ανίχνευσης προσώπου και χεριών με ανίχνευση δέρματος, όπως την επιφάνεια της Γκαουσιανής κατανομής, την εικόνα

πιθανότητας δέρματος, τις δυαδικές εικόνες δέρματος πριν και μετά τη μορφολογική επεξεργασία και την τελική ανίχνευση προσώπου και χεριών (προαιρετικό για όσους επιλέξουν να κάνουν και το Μέρος 1.1).

- Τελικές εικόνες της παρακολούθησης του προσώπου και των χεριών με το ορθογώνιο παρακολούθησης σχεδιασμένο σε καθεμιά.
- Εικόνες με την οπτικοποίηση των διανυσμάτων οπτικής ροής για όλα τα πλαίσια της ακολουθίας εικόνων που σας δίνεται.
- Python scripts (λίστες εντολών) με επαρκή σχόλια στα σημαντικά σημεία.
- Τελική αναφορά που να περιλαμβάνει τις πιο σημαντικές εικόνες αποτελεσμάτων από τις παραπάνω, επεξήγηση των αλγορίθμων και σχολιασμό των αποτελεσμάτων.

Μέρος 2: Εντοπισμός Χωρο-χρονικών Σημείων Ενδιαφέροντος και Εξαγωγή Χαρακτηριστικών σε Βίντεο Ανθρωπίνων Δράσεων

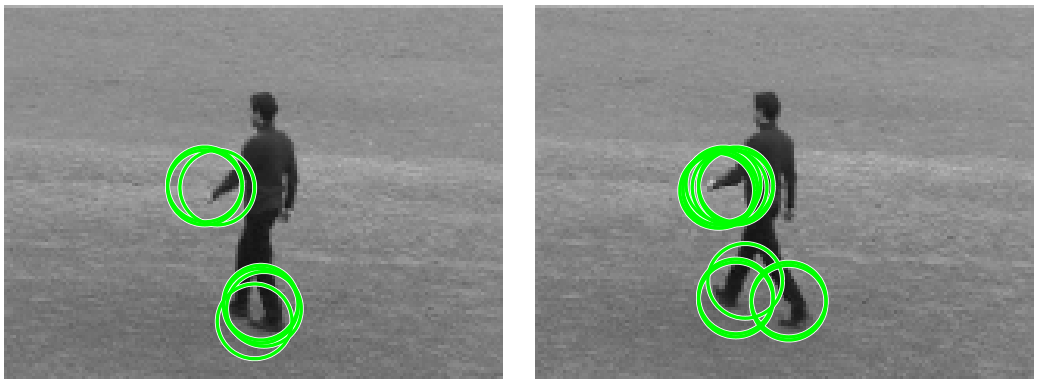
Στο 2ο Μέρος του εργαστηρίου θα ασχοληθούμε με την εξαγωγή χωρο-χρονικών χαρακτηριστικών με στόχο την εφαρμογή τους στο πρόβλημα κατηγοριοποίησης βίντεο που περιέχουν ανθρώπινες δράσεις. Όπως έχουμε ήδη δει, τα τοπικά χαρακτηριστικά (local features) έχουν δείξει τεράστια επιτυχία σε διάφορα προβλήματα αναγνώρισης της Όρασης Υπολογιστών, όπως η αναγνώριση αντικειμένων. Οι τοπικές αναπαραστάσεις περιγράφουν το προς παρατήρηση αντικείμενο με μια σειρά από τοπικούς περιγραφητές που υπολογίζονται σε γειτονιές ανιχνευθέντων σημείων ενδιαφέροντος. Τελικά, η συλλογή των τοπικών χαρακτηριστικών ενσωματώνεται σε μια τελική αναπαράσταση (global representation), όπως η γνωστή 'bag of visual words', ικανή να αναπαραστήσει τη στατιστική κατανομή τους και να προχωρήσει στα επόμενα στάδια της αναγνώρισης.

Η αναπαράσταση με χρήση τοπικών χαρακτηριστικών έχει επικρατήσει και στην αναγνώριση ανθρώπινων δράσεων, όπου γίνεται μια επιλογή από δεδομένα που αφ΄ ενός μειώνουν κατά πολύ τη διάσταση των βίντεο και αφ΄ ετέρου τα μετασχηματίζουν σε μια αναπαράσταση που τα κάνει διαχωρίσιμα. Στα πλαίσια της άσκησης θα σας δοθούν βίντεο από 3 κλάσεις δράσεων (walking, running, boxing) από τα οποία θα εξάγετε χωρο-χρονικούς περιγραφητές με σκοπό την κατηγοριοποίηση των δράσεων που απεικονίζουν.

Τα βίντεο, τα οποία βρίσκονται στον φάκελο `Part 2` του συμπληρωματικού υλικού της άσκησης, θα τα διαβάσετε καλώντας την συνάρτηση read_video(name,nframes,0) του ίδιου φακέλου, όπου name το πλήρες όνομα του βίντεο και nframes ο αριθμός των frames (π.χ. 200) που θέλετε να διαβάσετε. Το video θα αναπαρίσταται με έναν τρισδιάστατο πίνακα, του οποίου η 3η διάσταση αντιστοιχεί στον χρόνο και αποτελεί την ακολουθία των frames, τα οποία είναι grayscale εικόνες.

2.1 Χωρο-χρονικά Σημεία Ενδιαφέροντος

Οι ανιχνευτές τοπικών χαρακτηριστικών αναζητούν χωρο-χρονικά σημεία και κλίμακες ενδιαφέροντος, τα οποία αντιστοιχούν σε περιοχές που χαρακτηρίζονται από σύνθετη κίνηση ή απότομες μεταβολές στην εμφάνιση του video εισόδου. Αυτό επιτυγχάνεται μεγιστοποιώντας μια συνάρτηση 'οπτικής σημαντικότητας'. Πολλοί ανιχνευτές έχουν επινοηθεί τα τελευταία χρόνια αντλώντας αραιά αλλά εύρωστα σημεία [12]. Στην εργαστηριακή αυτή άσκηση θα ασχοληθούμε με 2 διαφορετικούς τέτοιους ανιχνευτές: 1) Harris detector [6] και 2) Gabor detector [3].



(α) (β)

Σχήμα 4: Παράδειγμα ανίχνευσης χωρο-χρονικών σημείων ενδιαφέροντος (Harris Detector).

2.1.1 Υλοποιήστε τον ανιχνευτή Harris ο οποίος αποτελεί μια επέκταση σε 3 διαστάσεις του ανιχνευτή γωνιών Harris-Stephens, που υλοποιήσατε στην 1η εργαστηριακή άσκηση. Για κάθε voxel του βίντεο υπολογίστε τον 3×3 πίνακα $M(x, y, t)$ προσθέτοντας στον 2Δ δομικό τανυστή και τη χρονική παράγωγο:

$$M(x, y, t; \sigma, \tau) = g(x, y, t; s\sigma, s\tau) * (\nabla L(x, y, t; \sigma, \tau)(\nabla L(x, y, t; \sigma, \tau))^T)$$

ή σε μορφή πικάνων:

$$M(x, y, t; \sigma, \tau) = g(x, y, t; s\sigma, s\tau) * \begin{pmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{pmatrix},$$

όπου $g(x, y, t; s\sigma, s\tau)$ ένας 3Δ γκαουσιανός πυρήνας ομαλοποίησης και $\nabla L(x, y, t; \sigma, \tau)$ οι χωρο-χρονικές παράγωγοι για την χωρική κλίμακα σ και τη χρονική κλίμακα τ. Τις παραγώγους (χωρικές και χρονικές) μπορείτε να τις υπολογίσετε εφαρμόζοντας συνέλιξη με τον πυρήνα κεντρικών διαφόρων $[-1\,0\,1]^T$ (προσαρμοσμένο στην κατάλληλη διάσταση).

Το 3Δ κριτήριο γωνιότητας ακολουθεί και αυτό την ίδια λογική:

$$H(x, y, t) = det(M(x, y, t)) - k \cdot trace^3(M(x, y, t))$$

2.1.2 Υλοποιήστε τον ανιχνευτή Gabor ο οποίος βασίζεται στο χρονικό φιλτράρισμα του βίντεο με ένα ζεύγος Gabor φίλτρων αφού πρώτα αυτό έχει υποστεί εξομάλυνση στις χωρικές διαστάσεις μέσω ενός 2Δ γκαουσιανού πυρήνα $g(x, y; \sigma)$ με τυπική απόκλιση σ. Τα Gabor φίλτρα ορίζονται ως:

$$h_{ev}(t; \tau, \omega) = \cos(2\pi t\omega) \exp(-t^2/2\tau^2) \text{ και } h_{od}(t; \tau, \omega) = \sin(2\pi t\omega) \exp(-t^2/2\tau^2)$$

Για τον υπολογισμό της κρουστικής απόκρισης των Gabor θεωρήστε μέγεθος παραθύρου $[-2\tau, 2\tau]$ και κανονικοποιήστε με την $L1$ νόρμα.

Η συχνότητα ω του Gabor φίλτρου συνδέεται με την χρονική κλίμακα τ (απόκλιση της γκαουσιανής συνιστώσας του) μέσω της σχέσης: $\omega = 4/\tau$. Το κριτήριο σημαντικότητας προκύπτει παίρνοντας την τετραγωνική ενέργεια της εξόδου για το ζεύγος Gabor φίλτρων:

$$H(x, y, t) = (I(x, y, t) * g * h_{ev})^2 + (I(x, y, t) * g * h_{od})^2$$

2.1.3 Για καθένα ανιχνευτή ακολουθήστε τα βήματα του Μέρους 2.1 του 1ου Εργαστηρίου (χωρίς τη συνθήκη Σ1) για να απορρίψετε σημεία που αντιστοιχούν σε σχετικά ομαλές περιοχές κάθε frame (περιοχές με χαμηλό ‘‘κριτήριο γωνιότητας’’).

2.1.4 Για καθένα ανιχνευτή υπολογίστε τα σημεία ενδιαφέροντος σαν τα τοπικά μέγιστα του κριτηρίου σημαντικότητας. Για απλότητα, μπορείτε να επιστρέψετε τα N σημεία με τις μεγαλύτερες τιμές του κριτηρίου σημαντικότητας (π.χ. τα 500-600 πρώτα). Απεικονίστε για επιλεγμένα frames τα κριτήρια σημαντικότητας καθώς και τα σημεία που προκύπτουν χρησιμοποιώντας τη συνάρτηση `show_detection` απο το συμπληρωματικό υλικό. Μπορείτε να πειραματιστείτε με διαφορετικές χωρικές και χρονικές κλίμακες ή και με πολλαπλές κλίμακες. Τι παρατηρείτε ως προς τον τύπο των σημείων που ανιχνεύουν οι δύο μέθοδοι;

► *Η συνάρτηση* `show_detection` από το συμπληρωματικό υλικό δέχεται ως πρώτο όρισμα τον 3Δ πίνακα που αντιστοιχεί στο βίντεο και ως δεύτερο όρισμα τα N σημεία που εντόπισαν οι ανιχνευτές. Τα σημεία αυτά πρέπει να είναι στην μορφή ενός `numpy` πινάκα $N \times 4$. Οι τρεις πρώτες στήλες αντιστοιχούν στις συντεταγμένες τους (x, y, t), όπου t το frame στο οποίο ανιχνεύθηκαν και η τέταρτη στην κλίμακα σ στην οποία ανιχνεύθηκαν.

► *Ενδεικτικές τιμές παραμέτρων:* $\sigma = 4$, $s = 2$, $\tau = 1.5$, $k = 0.005$

► Βοήθεια για Python: συναρτήσεις `(cv2) getGaussianKernel, (scipy.ndimage) convolve1d, (numpy) argsort, unravel_index`.

2.2 Χωρο-χρονικοί Ιστογραφικοί Περιγραφητές

Οι χωρο-χρονικοί περιγραφητές που θα χρησιμοποιηθούν βασίζονται στον υπολογισμό ιστογραμμάτων της κατευθυντικής παραγώγου (HOG) και της οπτικής ροής (HOF - Histogram of Oriented Flow) [6] γύρω από τα σημεία ενδιαφέροντος που υπολογίσατε.

2.2.1 Για κάθε frame του βίντεο υπολογίστε το διάνυσμα κλίσης (gradient) και την TV-L1 οπτική ροή σε κάθε pixel.

2.2.2 Στη συνέχεια χρησιμοποιήστε τη συνάρτηση `orientation_histogram` από το συμπληρωματικό υλικό του μέρους αυτού προκειμένου να υπολογίσετε τους 2 ιστογραφικούς περιγραφητές. Η συνάρτηση αυτή δέχεται ως είσοδο το διανυσματικό πεδίο (κατευθυντικές παραγώγους είτε κατεύθυνση ροής), το μέγεθος του grid και το πλήθος των bins και επιστρέφει την ιστογραμματική περιγραφή της αντίστοιχης περιοχής. Εσείς καλείστε να εξάγετε τα διανυσματικά πεδία που απαιτούνται για μια (τετραγωνική) περιοχή $4 \times \sigma$ γύρω από το εκάστοτε σημείο ενδιαφέροντος (από την εικόνα που αντιστοιχεί στο frame που ανιχνεύσατε σημεία ενδιαφέροντος). Να δώσετε προσοχή στα όρια της εικόνας. Για τη δημιουργία του HOG/HOF περιγραφητή συνενώστε τους δύο επιμέρους περιγραφητές.

► *Η συνάρτηση* `orientation_histogram` *για την εξαγωγή τοπικών περιγραφητών καλείται ως εξής:* `desc = orientation_histogram(Gx,Gy,nbins,np.array([n,m]))`, όπου `Gx,Gy` ορίζουν ένα διανυσματικό πεδίο, `nbins` ειναι το πλήθος των bins και $n \times m$ είναι το μέγεθος του grid. Η έξοδος `desc` είναι ένα ιστόγραμμα (περιγραφητής) για την περιοχή του διανυσματικού πεδίου εισόδου μεγέθους $n \times m \times nbins$.

► *Για τον υπολογισμό του περιγραφητή HOF χρησιμοποιήστε ένα πυκνό πεδίο οπτικής ροής, όπως αυτό της TV-L1 [9]. Μπορείτε να χρησιμοποιήσετε τις συναρτήσεις:*

```
oflow = cv2.DualTVL1OpticalFlow_create(nscales=1)
flow = oflow.calc(patch1, patch2, None)
```

Προσέξτε πως οι εικόνες εισόδου για αυτό μόνο το βήμα πρέπει να είναι τύπου `uint8` *με εύρος* $0 - 255$.

2.3: Κατασκευή Bag of Visual Words και χρήση Support Vector Machines για την ταξινόμηση δράσεων

Στο ερώτημα αυτό θα γίνει κατηγοριοποίηση των βίντεο με τις ανθρώπινες δράσεις σε 3 κατηγορίες/κλάσεις (που η κάθε μία θα αντιπροσωπεύει ένα διαφορετικό είδος δράσης) με χρήση

των BoVW αναπαραστάσεων βασισμένων στα HOG / HOF χαρακτηριστικά, που εξάγατε στα προηγούμενα ερωτήματα. Το τελικό αποτέλεσμα θα είναι το ποσοστό επιτυχούς ταξινόμησης δράσεων, χρησιμοποιώντας SVM (Support Vector Machine) ταξινομητή.

2.3.1 Διαχωρίστε το σύνολο των βίντεο σε σύνολο εκπαίδευσης (train set) και σύνολο δοκιμής (test set) με βάση το αρχείο που σας δίνεται στο συμπληρωματικό υλικό.

2.3.2 Υπολογίστε την τελική αναπαράσταση (global representation) για κάθε βίντεο με την bag of visual words (BoVW) τεχνική που περιγράφεται στην 1η εργαστηριακή άσκηση, χρησιμοποιώντας μόνο τα βίντεο εκπαίδευσης. Για τον υπολογισμό των BoVW ιστογραμμάτων χρησιμοποιήστε τη συνάρτηση `bag_of_words` από το συμπληρωματικό υλικό αυτού του μέρους.

► *Η συνάρτηση* `bag_of_words` *καλείται ως εξής:* `bow_train, bow_test = bag_of_words( desc_train, desc_test, num_centers=D)`, *όπου* `desc_train, desc_test` το σύνολο των περιγραφητών των train και test βίντεο και `D` ο αριθμός των οπτικών λέξεων (δηλαδή των κέντροειδών του K-means). Τα `desc_train, desc_test` είναι λίστες μήκους N_{train}, N_{test}, δηλαδή έχουν μέγεθος ίσο με τον αριθμό των βίντεο εκπαίδευσης και δοκιμής αντίστοιχα. Κάθε στοιχείο κάθε μιας λίστας είναι ένας 2Δ numpy πίνακας του οποίου κάθε γραμμή περιέχει κι από έναν descriptor.

2.3.3 Το τελικό στάδιο συνισταται στην τελική κατηγοριοποίηση των εικόνων με βάση την BoVW αναπαράσταση. Για την κατηγοριοποίηση χρησιμοποιείται ενάς SVM ταξινομητής κατάλληλα προσαρμοσμένος για πολλαπλές κλάσεις. Η όλη διαδικασία υλοποιείται με τη συνάρτηση `svm_train_test` από το συμπληρωματικό υλικό, η οποία δέχεται 2 numpy πίνακες διαστάσεων $N_{train} \times D$ και $N_{test} \times D$, όπου N_{train}, N_{test} ο αριθμός των βίντεο εκπαίδευσης και δοκιμής αντίστοιχα και D η διάσταση του κάθε BoVW διανύσματος. Η συνάρτηση επιστρέφει το αποτέλεσμα της αναγνώρισης καθώς και το συνολικό ποσοστό επιτυχίας.

► *Η συνάρτηση* `svm_train_test` *για την εκπαίδευση του SVM και τη χρήση του για την κατηγριοποίηση όλων των βίντεο δοκιμής καλείται ως εξής:* `accuracy, pred = svm_train_test(bow_train, train_labels, bow_test, test_labels)`, *όπου τα ορίσματα* `bow_train, bow_test` *είναι* numpy *πίνακες διαστάσεων $N_{train} \times D$, $N_{test} \times D$, ενώ τα* `train_labels, test_labels` *είναι λίστες μήκους N_{train}, N_{test} αντίστοιχα. Η συνάρτηση επιστρέφει το αποτέλεσμα της αναγνώρισης (*`pred`*) καθώς και το συνολικό ποσοστό επιτυχίας (*`accuracy`*).*

2.3.4 Πειραματιστείτε με τους διαφορετικούς συνδυασμούς ανιχνευτών/περιγραφητών και παρατηρήστε τις μεταβολές στην κατηγοριοποίηση. Αναφέρετε τον καλύτερο συνδυασμό που χρησιμοποιήσατε και σχολιάστε.

► *Για να μειώσετε τη διάρκεια των πειραμάτων σας, συστήνεται να αποθηκεύετε στο δίσκο τα ενδιάμεσα σποτελέσματα (όπως π.χ. τα εντοπισμένα σημεία ενδιαφέροντος), ώστε κάθε φορά που κάνετε κάποια αλλαγή απλά να τα φορτώνετε και να μην τα ξαναϋπολογίζετε. Αυτό μπορεί να γίνει με τη χρήση της βιβλιοθήκης* `pickle`.

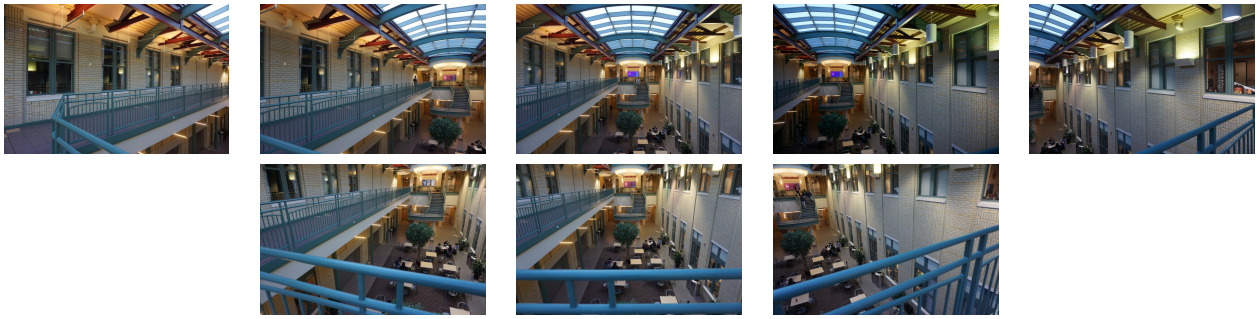
ΠΑΡΑΔΟΤΕΑ: Ζητείται να παραδώσετε τα εξής:

- Τελικές εικόνες ή βίντεο που απεικονίζουν τα ανιχνευθέντα σημεία ενδιαφέροντος (ένα παράδειγμα ανίχνευσης για καθεμιά από τις τρεις δράσεις).
- Ποσοστά αναγνώρισης για κάθε συνδυασμό ανιχνευτή σημείων ενδιαφέροντος - περιγραφητή, καθώς και για όλους τους πειραματισμούς σας.

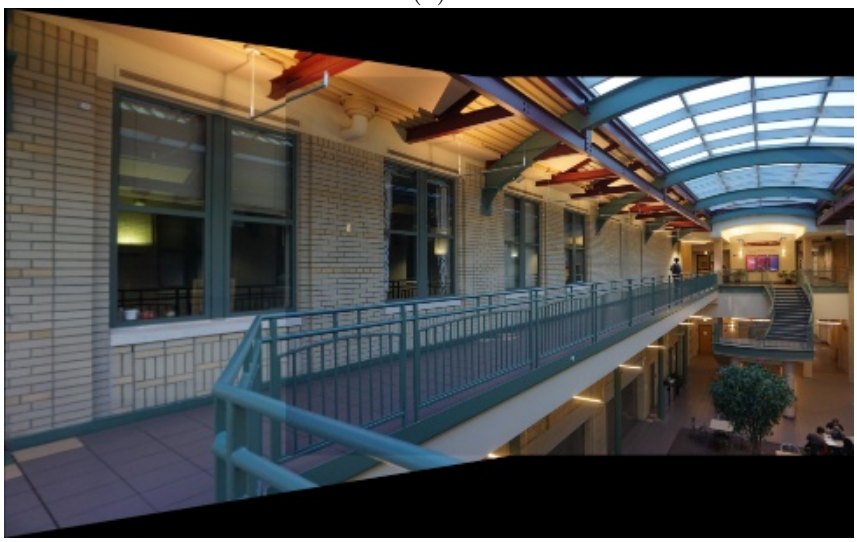
- Python scripts (λίστες εντολών) με επαρκή σχόλια στα σημαντικά σημεία.
- Τελική αναφορά που να περιλαμβάνει τις πιο σημαντικές εικόνες αποτελεσμάτων από τις παραπάνω,χαρακτηριστικές εικόνες ενδιάμεσων αποτελεσμάτων (π.χ. πίνακας $H(x, y, \cdot)$) επεξήγηση των αλγορίθμων και σχολιασμό των αποτελεσμάτων.

Μέρος 3: Συνένωση Εικόνων (Image Stitching) για Δημιουργία Πανοράματος

Το τρίτο μέρος του εργαστηρίου περιλαμβάνει τη δημιουργία μιας εφαρμογής για τη συνένωση διαδοχικών εικόνων που έχουν ληφθεί από ένα σημείο του πραγματικού κόσμου (κάμερα σε σταθερό σημείο) με απλή περιστροφή της κάμερας. Αυτό συμβαίνει διότι σε αυτή την ειδική περίπτωση είναι εφικτός ο υπολογισμός μιας ομογραφίας μεταξύ των διαδοχικών εικόνων, ούτως ώστε να μπορούν να τοποθετηθούν στο ίδιο πλαίσιο αναφοράς. Ο αναγνώστης μπορεί να απευθυνθεί για περισσότερες λεπτομέρειες στο βιβλίο [5], καθώς και στη δημοσίευση [2].
Ενδεικτικά, κάποιες από τις εικόνες που σας δίνονται ως το σύνολο δεδομένων που θα κληθείτε να χρησιμοποιήσετε παρουσιάζονται στο Σχήμα 5, καθώς και ένα ενδιάμεσο αποτέλεσμα.
Ο σκοπός της δημιουργίας τέτοιων πανοραμάτων είναι κατά κύριο λόγο η τεχνητή αύξηση του πεδίου ορατότητας (field of view) μέχρι και 360^o. Εναλλακτικά, μπορεί να χρησιμοποιηθεί και σε εφαρμογές αποθορυβοποίησης ή αύξησης της ανάλυσης εικόνων (super-resolution).



(α)



(β)

Σχήμα 5: (α) Οι οκτώ αρχικές εικόνες προς συνένωση. (β) Ενδιάμεσο ενδεικτικό αποτέλεσμα συνένωσης δύο εικόνων.

Περιγραφή του αλγόριθμου συνένωσης εικόνων

Η βασική ρουτίνα που καλείστε να υλοποιήσετε στο παρόν 3ο μέρος του εργαστηρίου για τη συνένωση των εικόνων `img1` και `img2` (στο εξής θα αναφέρονται ως 'Εικόνα Α' και 'Εικόνα Β') είναι η `stitchImages`:

```
stitched_result = stitchImages(img1, img2)
```

Στη συνέχεια αναλύουμε τα βασικά προτεινόμενα βήματα για την υλοποίησή της, τα οποία παρουσιάζονται συνοπτικά στον παρακάτω αλγόριθμο.

Algorithm 1 Image stitching algorithm pipeline

1: **procedure** IMAGE_STITCHING($img1, img2$)
2: **Step 1:** Extract SIFT features and descriptors from both images
3: **Step 2:** Match features by applying Brute-Force-based or FLANN-based matching
4: **Step 3:** Apply Lowe's criterion to speedily reject most outliers
5: **Step 4:** Compute RANSAC-based Homography H
6: **Step 5:** Apply inverse warping to $img1$
7: **Step 6:** Merge the warped version of $img1$ with $img2$ under the same coordinate system of $img2$

Βήμα 0: Διάβασμα εικόνων

Το σύνολο των εικόνων που καλείστε να συνενώσετε περιλαμβάνει οκτώ εικόνες που απεικονίζουν διαδοχικά και επικαλυπτόμενα τμήματα από το εσωτερικό ενός κτηρίου. Οι εικόνες αυτές περιλαμβάνονται στον φάκελο `Part 3` του συμπληρωματικού υλικού της άσκησης.

Βήμα 1: Εντοπισμός χαρακτηριστικών ενδιαφέροντος και εξαγωγή περιγραφητών

Στο Εργαστήριο 1 σας είχε ζητηθεί η υλοποίηση των χαρακτηριστικών ενδιαφέροντος (γωνίες), ενώ στη συνέχεια η υλοποίηση των τοπικών περιγραφητών SURF και HOG σας δινόταν έτοιμη. Χάριν συντομίας, σε αυτό το μέρος του εργαστηρίου έχετε τη δυνατότητα να χρησιμοποιήσετε τα χαρακτηριστικά ενδιαφέροντος και περιγραφητές SIFT έτοιμα, όπως έχουν υλοποιηθεί στη βιβλιοθήκη `OpenCV`. Στη συνέχεια, απεικονίστε τα σημεία που υπολογίσατε πάνω στις αρχικές εικόνες.

▶ Βοήθεια για Python: συναρτήσεις (cv2) `SIFT_create`, `detectAndCompute`, `drawKeypoints`.

Βήμα 2: Διαδικασία ταιριάσματος χαρακτηριστικών μεταξύ δύο εικόνων

Σε αυτό το βήμα ζητείται να βρεθεί η αντιστοίχιση κάθε χαρακτηριστικού της εικόνας Α με τα δύο 'κοντινότερα' χαρακτηριστικά της εικόνας Β (ο λόγος που ζητούνται τα δύο 'κοντινότερα' χαρακτηριστικά θα γίνει κατανοητός στο επόμενο Βήμα 3) με βάση ένα κριτήριο εγγύτητας που προκύπτει μέσω του υπολογισμού της απόστασης μεταξύ των αντίστοιχων περιγραφητών. Τα χαρακτηριστικά της εικόνας Α συναντώνται στη βιβλιογραφία και στο documentation ως query features, ενώ της εικόνας Β ως train features. Πιο συγκεκριμένα, θα μπορούσατε να επιλέξετε μεταξύ των εξής δύο τρόπων εύρεσης του ζητούμενου αποτελέσματος:

- Εύρεση των **πραγματικών** κοντινότερων γειτόνων μέσω της εφαρμογής Brute-Force αντιστοίχισης: Είναι ο πιο απλός (αλλά και ταυτόχρονα πιο αργός) τρόπος αντιστοίχισης, κατά τον οποίο για κάθε χαρακτηριστικό της εικόνας Α ελέγχονται όλα τα χαρακτηριστικά της εικόνας Β και επιλέγονται τα δύο που απέχουν το ελάχιστο.

  ▶ Βοήθεια για Python: συναρτήσεις (cv2) `BFMatcher`, `knnMatch`.

- Εύρεση των **προσεγγιστικών** κοντινότερων γειτόνων μέσω της αντιστοίχισης με χρήση της βιβλιοθήκης FLANN (Fast Library for Approximate Nearest Neighbors) (συνιστάται): Είναι ένας πολύ πιο γρήγορος τρόπος εύρεσης των αντιστοιχίσεων, καθώς βασίζεται στην εκπαίδευση ενός κατάλληλου δέντρου με βάση τα χαρακτηριστικά της εικόνας B (γι' αυτό ονομάζονται και train) και έπειτα σε απλή επιλογή της βέλτιστης διαδρομής μέσα στο δέντρο.

  ▶ Βοήθεια για Python: συναρτήσεις (cv2) `FlannBasedMatcher`, `knnMatch`. *Ο* `FlannBasedMatcher` *δέχεται δύο παραμέτρους* (`index_params` *και* `search_params`) *ως λεξικά της* Python. *Στο πρώτο λεξικό μπορείτε να χρησιμοποιήσετε ως αλγόριθμο τον* `FLANN_INDEX_KDTREE` *με αριθμό δέντρων ίσο με 5, ενώ το δεύτερο μπορεί να δοθεί ως το κενό. Περισσότερες σχετικές πληροφορίες υπάρχουν και στο documentation της FLANN:* https://usermanual.wiki/Document/flannmanual184.1650808969/view.

Βήμα 3: Διατήρηση μόνο των πιο έγκυρων αντιστοιχίσεων μέσω εφαρμογής του κριτηρίου Lowe

Από την εύρεση των δύο καλύτερων αντιστοιχίσεων για κάθε χαρακτηριστικό της εικόνας A ενδέχεται ο λόγος της απόστασης μεταξύ των δύο ζευγών περιγραφητών να είναι πολύ κοντά στη μονάδα. Ελέγχοντας αν είναι μεγαλύτερος από ένα κατώφλι (για παράδειγμα το 0.8) μπορούμε να απαλλαχθούμε από ένα μεγάλο πλήθος λάθος αντιστοιχίσεων, καθώς πιθανοτικά έχει δειχτεί στο seminal paper του Lowe για τα χαρακτηριστικά SIFT [7] ότι για τέτοιες τιμές του λόγου απορρίπτονται το 90% των λάθος ταιριασμάτων, ενώ χάνεται μόλις το 5% των σωστών. Μπορείτε να αποθηκεύσετε τα αποτελέσματα των ταιριασμάτων που θα διατηρηθούν σε ένα boolean διάνυσμα ως μάσκα.
▶ Βοήθεια για Python: συνάρτηση (cv2) `drawMatches`. *Μπορείτε έτσι να οπτικοποιήσετε τις αντιστοιχίσεις που προέκυψαν ύστερα από την εφαρμογή του κριτηρίου του Lowe.*

Βήμα 4: Υπολογισμός ομογραφίας με RANSAC

Με δεδομένες τις αντιστοιχίσεις σημείων μεταξύ της εικόνας A και B μπορεί πλέον να υπολογιστεί η ομογραφία H μεταξύ των δύο εικόνων. Πιο συγκεκριμένα, κάνουμε τη σύμβαση υπολογισμού της ομογραφίας H_{BA} που μεταφέρει τα σημεία από την εικόνα A στη B μέσω της σχέσης $\mathbf{x}_B = H_{BA}\mathbf{x}_A$, όπου $\mathbf{x}_A$ και $\mathbf{x}_B$ είναι τα σημεία του επιπέδου της εικόνας σε ομογενείς συντεταγμένες και H_{BA} ο πίνακας της ομογραφίας με διαστάσεις 3×3.
▶ Βοήθεια για Python: συνάρτηση (cv2) `findHomography`. *Στην έτοιμη υλοποίηση της* `OpenCV` *μπορείτε να εφαρμόσετε και τη διαδικασία απόρριψης outliers μέσω της παραμέτρου* `cv2.RANSAC` *με κάποιο κατώφλι απόρριψης που θα ορίσετε εσείς.*

Βήμα 5: Υπολογισμός μετασχηματισμένης εικόνας μέσω εφαρμογής Inverse warping

Σε αυτό το βήμα πραγματοποιείται η μεταφορά όλων των τιμών φωτεινότητας των πίξελ της εικόνας A στα σημεία που προσδιορίζονται μέσω της υπολογισθείσας ομογραφίας H. Σας προτείνεται να κατασκευάσετε την εξής συνάρτηση:

```
img1_warped, img1_topleft_coords = projectionImage(H, img1)
```

όπου `img1_warped` είναι η μετασχηματισμένη εικόνα (με διαφορετικές διαστάσεις από την αρχική, πρέπει να τις προσδιορίσετε) και `img1_topleft_coords` είναι οι συντεταγμένες στις οποίες πρέπει να τοποθετηθεί η πάνω-αριστερά γωνία της μετασχηματισμένης εικόνας A (εκπεφρασμένες ως προς το πλαίσιο αναφοράς συντεταγμένων της εικόνας B - ενδέχεται να βρίσκονται στα αρνητικά των αξόνων x, y).

▶ *Εάν μεταφέρετε τα σημεία της εικόνας A μέσω της ομογραφίας H, τότε θα συναντήσετε το εξής πρόβλημα: Οι ακέραιες τιμές των συντεταγμένων της εικόνας A μετασχηματίζονται σε δεκαδικές τιμές στο πλαίσιο αναφοράς της εικόνας B. Έτσι, εφαρμόζοντας κάποια συνάρτηση τύπου* `floor` *ή* `round`*, κάποια από τα πίξελ της μετασχηματισμένης εικόνας A θα προκύψουν κενά. Προκειμένου να το αποφύγουμε αυτό, εφαρμόζουμε* ***inverse warping****, δηλαδή υπολογίζουμε από πού προέρχονται οι ακέραιες τιμές συντεταγμένων στο πλαίσιο αναφοράς της εικόνας B μέσω της αντίστροφης ομογραφίας H^{-1}. Σε αυτή την περίπτωση, η εφαρμογή συναρτήσεων* `floor` *ή* `round` *(ή καλύτερα η εφαρμογή linear interpolation) προσδίδει τιμές φωτεινότητας σε όλα τα πίξελ.*

Βήμα 6: Συνένωση των εικόνων

Σε αυτό το βήμα πραγματοποιείται η συνένωση της μετασχηματισμένης εικόνας A που υπολογίστηκε στο προηγούμενο βήμα με την εικόνα B. Μπορείτε να πραγματοποιήσετε αυτή τη λειτουργία μέσω της εξής συνάρτησης που θα ορίσετε και θα υλοποιήσετε:

```
stitched_image = mergeWarpedImages(img1_warped, img2, img1_topleft_coords)
```

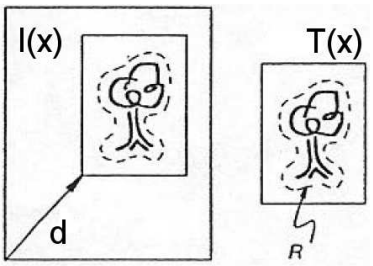
όπου `stitched_image` είναι η τελική εικόνα (με διαφορετικές διαστάσεις από τις υπόλοιπες εικόνες, θα χρειαστεί να τις προσδιορίσετε). Η λειτουργία που πραγματοποιεί η παραπάνω συνάρτηση είναι ιδιαίτερα απλή. Πρέπει να συνενώνει τις `img1_warped` και `img2` με τέτοιο τρόπο ώστε η πάνω-αριστερά γωνία της μετασχηματισμένης εικόνας A να βρίσκεται στις συντεταγμένες που προσδιορίζουν οι `img1_topleft_coords` ως προς το πλαίσιο αναφοράς συντεταγμένων της εικόνας B.

ΠΑΡΑΔΟΤΕΑ: Ζητείται να παραδώσετε τα εξής:

- Ενδιάμεσες εικόνες από τη συνένωση ζευγαριών εικόνων καθώς αναπτύσσεται το πανόραμα, καθώς και το τελικό αποτέλεσμα συνένωσης των οκτώ εικόνων μαζί.
- Python scripts (λίστες εντολών) με επαρκή σχόλια στα σημαντικά σημεία.
- Τελική αναφορά που να περιλαμβάνει τις πιο σημαντικές εικόνες αποτελεσμάτων από τη διαδικασία που ακολουθήθηκε, χαρακτηριστικές εικόνες ενδιάμεσων αποτελεσμάτων (π.χ. πλαίσια αναφοράς συντεταγμένων των δύο εικόνων), επεξήγηση των αλγορίθμων (π.χ. με ποιο τρόπο αποφασίσατε να υλοποιήσετε τα Βήματα 5 και 6) και σχολιασμό των αποτελεσμάτων.

Παράρτημα: Αντιστοίχιση Εικόνων με τη Μέθοδο των Lucas και Kanade

Η εργασία των Lucas-Kanade [8] εισήγαγε ένα πλαίσιο στο οποίο εντάσσεται μια ευρεία οικογένεια αλγορίθμων για αντιστοίχιση εικόνων.
Θεωρούμε μια εικόνα I την οποία επιθυμούμε να αντιστοιχίσουμε με μια εικόνα αναφοράς (τεμπλέτα) T. Αυτές μπορεί για παράδειγμα να είναι ολόκληρα διαδοχικά πλαίσια βίντεο ή να έχουν ληφθεί από ζεύγος στέρεο καμερών. Εναλλακτικά, στην περίπτωση που στόχος είναι ο υπολογισμός του πεδίου οπτικής ροής τα I και T είναι μικρές υποπεριοχές (π.χ. μπλοκ διάστασης 8×8 pixel) εξαγμένες από τα πλαίσια του βίντεο.



Σχήμα 6: Μεταφορική αντιστοίχιση μεταξύ εικόνας I και τεμπλέτας T. Από το [8].

Υποθέτουμε επίσης ότι η σχετική κίνηση μεταξύ των αντίστοιχων περιοχών στις δυο εικόνες προσεγγίζεται ικανοποιητικά ως μεταφορική, δηλαδή $I(\mathbf{x}+\mathbf{d}) \approx T(\mathbf{x})$, με $\mathbf{d} = (d_x, d_y)^T$, όπως φαίνεται στο Σχ. 6. Αναζητούμε το διάνυσμα μετατόπισης $\mathbf{d}$ που ελαχιστοποιεί κάποιο κριτήριο σφάλματος αντιστοίχισης, π.χ. το μέσο τετραγωνικό σφάλμα

$$J(\mathbf{d}) = \sum_{\mathbf{x}\in R} \left(I(\mathbf{x}+\mathbf{d}) - T(\mathbf{x})\right)^2 . \tag{8}$$

Στον αλγόριθμο Lucas-Kanade θεωρούμε ότι έχουμε μια εκτίμηση $\mathbf{d}_i$ για το $\mathbf{d}$ και προσπαθούμε να τη βελτιώσουμε κατά $\mathbf{u} = (u_x, u_y)^T$, δηλαδή $\mathbf{d}_{i+1} = \mathbf{d}_i + \mathbf{u}$. Εάν αναπτύξουμε κατά Taylor την έκφραση $I(\mathbf{x}+\mathbf{d}) = I(\mathbf{x}+\mathbf{d}_i+\mathbf{u})$ γύρω από το σημείο $\mathbf{x}+\mathbf{d}_i$, προκύπτει η γραμμική προσέγγιση $I(\mathbf{x}+\mathbf{d}) \approx I(\mathbf{x}+\mathbf{d}_i) + A(\mathbf{x})\mathbf{u}$, όπου έχουμε ορίσει τον 1×2 πίνακα μερικών παραγώγων της εικόνας $A(\mathbf{x}) = (\frac{\partial I}{\partial x}(\mathbf{x}+\mathbf{d}_i), \frac{\partial I}{\partial y}(\mathbf{x}+\mathbf{d}_i))$. Αντικαθιστώντας αυτήν την έκφραση στην Εξ. (8) προκύπτει

$$J(\mathbf{u}) = \sum_{\mathbf{x}\in R} \left(E(\mathbf{x}) + A(\mathbf{x})\mathbf{u}\right)^2 , \tag{9}$$

όπου $E(\mathbf{x}) = I(\mathbf{x}+\mathbf{d}_i) - T(\mathbf{x})$ είναι η εικόνα με το σφάλμα αντιστοίχισης κατά την τρέχουσα επανάληψη. Για να βελτιστοποιήσουμε τη J μηδενίζουμε τον πίνακα μερικών παραγώγων ως προς $\mathbf{u}$, οπότε προκύπτει το 2×2 γραμμικό σύστημα

$$0 = \frac{dJ}{d\mathbf{u}} = 2\sum_{\mathbf{x}\in R} A(\mathbf{x})^T \left(E(\mathbf{x}) + A(\mathbf{x})\mathbf{u}\right), \tag{10}$$

$$\text{ή} \quad \left(\sum_{\mathbf{x}\in R} A(\mathbf{x})^T A(\mathbf{x})\right)\mathbf{u} = -\sum_{\mathbf{x}\in R} A(\mathbf{x})^T E(\mathbf{x}) \tag{11}$$

Η Εξ. (5) είναι παραλλαγή της (11) στην οποία έχουμε επίσης εφαρμόσει στο κριτήριο σφάλματος (8) γκαουσιανό παράθυρο στάθμισης, που δίνει μεγαλύτερη έμφαση στα κεντρικά pixel του μπλοκ, και επίσης έχουμε προσθέσει την κανονικοποιητική σταθερά μ στη διαγώνιο του πίνακα γραμμικού συστήματος.

Πέρα από το βασικό αλγόριθμο που περιγράφουμε, η τεχνική των Lucas-Kanade μπορεί να επεκταθεί για να καλύψει γενικότερα μοντέλα κίνησης (π.χ. περιστροφή ή αφινική κίνηση), καθώς και μοντέλα φωτομετρικής αλλοίωσης [4]. Μια πρόσφατη επισκόπηση της σχετικής βιβλιογραφίας μπορεί να βρεθεί στα [1, 11].

References

[1] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *Int. J. of Comp. Vis.*, 56(3):221–255, 2004.

[2] Brown and Lowe. Recognising panoramas. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1218–1225 vol.2, 2003.

[3] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proc. IEEE Int'l Workshop on VS-PETS*, 2005.

[4] C.S. Fuh and P. Maragos. Motion displacement estimation using an affine model for image matching. *Optical Engin.*, (30):7, July 1991.

[5] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2 edition, 2003.

[6] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Conf. CVPR*, 2008.

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, nov 2004.

[8] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Int. Joint Conf. on Artificial Intel.*, pages 674–679, 1981.

[9] T. Pock, M. Urschler, C. Zach, R. Beichel, and H. Bischof. A Duality Based Algorithm for TV-L1-Optical-Flow Image Registration. In *Proc. Medical Image Computing and Computer-Assisted Intervention – MICCAI*, pages 511–518. Springer Berlin Heidelberg, 2007.

[10] J. Shi and C. Tomasi. Good Features to Track. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1994.

[11] R. Szeliski. Image alignment and stitching: a tutorial. *Found. and Trends in Comp. Graph. and Vision*, 2(1):1–104, 2006.

[12] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proc. BMVC*, 2009.