**HSV e HSL** -> Tinta° (Hue), Saturazione, Luminosità (value/lightness), intuitivo

**LUT**-> strut dati di supporto, salva i ris pre-calc per ogni possibile val di input dei pixel

**Binariz** -> soglia globale (manualmente, oss istogramma, metodo otsu) e locale

**Contr. Stretchin** -> Espans dei livelli di grigio per + contrasto(soglie 5º e 95º percentile)

**Convoluz**->operaz(commut e assoc) di media pesata locale, utilizza il filtro specchiato

sfocature per imperfezioni, evidenziare dettagli, estrazione dei contorni

**Istog** -> num pixel grigio (contrasto, lumin, evid oggg), equaliz distribuisce, HSL su L

**Trasf. Geom** -> alle coord dei pixel e non valore

**Filtri lin** -> val del pixel calcolato come somma pesata dei pixel nell’intorno

Separabili-> sono due vettori monodim al posto di una matrice

**Normalizz filtro** -> se i coeff >=0 si divide per la somma dei coeff per valori [0,255]

**Box Filter** -> attenua contrasto e rumore

**Filtro Gaussiano** -> pesi diminuiscono verso l’esterno (sfocatura)

**Sharpening** -> +contrasto e rumore: si calcola imm blurred con convol (filtro box o gaus)

e si sottrae all’originale; l’imm calc si somma all’iniziale moltiplicata per parametro k

**Bordi** -> si trovano in cambiamenti rapidi di f, sottraendo smooth dall’originale

**Gradiente** -> vett che ha per compon le due d parz. direz maggior variaz imm nel punto

**Canny edge detector** -> riconos contorni: smothing gaus, calc gradiente, soppres dei non-max

(elim pixel il cui modulo del gradiente non è max risp orient grad),

selez bordi con isteresi (si binarizza imm con soglie t1, t2 ( validi con modulo > t1 o adiac))

**Operatori(f**iltri derivativi) -> per calcolo del gradiente, estraggono contorni, calcolo dei max locali

in direz del gradiente (es.Canny). -**Roberts**: calc rapido, sensibi al rumore, assi orint modo tradiz.

-**Prewitt**: meno sensib a variaz, simmetrici. **Sobel**: come prewitt ma peso magg pixel centr

**Trasform distanza** di F risp F\* -> replica di F con pixel etichettati come val distanza da F\*

Con scansione diretta e inversa, usata per misuraz lung e spess, evid schelet, templ matc

**Etichet comp conn** ->scorre imm e per ogni pixel di foregr si consider i pixel già visitati

-Nessuno etic -> new etich -Uno etich -> stessa etich -Più di uno etich -> si annota equival.

**Morf Mat** -> tecnica derivata da teoria insiemi, per contorno e schelet o rimuovere particol

**Dilataz** -> new img è insieme d pixel t.c. traslando S almeno uno dei suoi elem è sovrapp

**Eros** -> new img è insieme d pixel t.c. traslando S l’intero elem strutturante è contenuto

**Apertura**-> eros-dilat, separa ogg debol connessi e rimuove regioni piccole

**Chiusura**-> dilat-eros, riempie buchi e piccole concavità, rafforza conn regioni deb conn

**Template Match**->ricerca di un templ in imm, sovrapposto in tutte le coord per trovare somigl

difficile gestione di pattern deformabili e computaz complesso.

Confronto Pixel a Pixel-> non efficace per traslaz, rotaz, scala, prosp, luminosità,

deform, diverse tecn di acquisiz.

ARRAY - RIEMPI CON VAL: np.full((15, 16), val) #\_like per passare dim di un array

ZEROS: np.zeros((15, 16), dtype=np.uint8)

DA A: np.arange(da, a, passo) | VUOTO: **np.empty**((2,7), np.int16)

RANDOM: np.random.randint(0, 256, (15, 16), dtype=np.uint8)

MATRICI: a = np.array( [[1,1], [0,1]] ) a.reshape(2, 6) a.T np.identity(3)

STRINGA LETT IMG: sprites = [cv.imread(f'movimento/{i}.png') for i in sprite\_names]

OR: cv.bitwise\_or/and/not(mask1, mask2)

SOMMA min delle righe: np.argmin(np.sum(img, 1))

MEDIA: media = np.mean(img[:Ym+1,...])

MEDIA pixel intorno: mean = cv.blur(img, (51,51), borderType= cv.BORDER\_CONSTANT)

BORDO: CONSTANT/REPLICATE/REFLECT/WARP/TRANSPARENT

**immagini**

LEGGI IMG: img = cv.imread('img.png', cv.IMREAD\_GRAYSCALE)

CONVERS IMG: gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY/BGR2HSV)

h,s,v=cv.split(img) cv.merge(h,s,v)

INGRANDIRE IMG: img\_bl = cv.resize(img, size, interpolation=cv.INTER\_LINEAR)

ISTOGRAMMA: hist = cv.calcHist([img], [0], None, [256], [0, 256])

EQUALIZ: res = cv.equalizeHist(img)

LUT: lut[img] #lut è un vettore da 256 elem da impostare

BINARIZ GLOBALE:\_, res = cv.threshold(img, 128, 255, cv.THRESH\_BINARY) #soglia a -1 per Otzu

LOCALE: res = cv.adaptiveThreshold(img, 255, cv.ADAPTIVE\_THRESH\_MEAN\_C,

cv.THRESH\_BINARY, dimIntorno, daSottrarre)

CONTR STRET: a, b = np.percentile(img, 5), np.percentile(img, 95)

n = 255\*(img.astype(float)-a)/(b-a) res = np.clip(n, 0, 255).astype(np.uint8)

TRASF AFF: M = cv.getAffineTransform(m\_pts,pts) #non so come si usino quindi boh

cv.warpAffine(img,M,(w,h),None,cv.INTER\_CUBIC,cv.BORDER\_CONSTANT,bc)

Mp = cv.getPerspectiveTransform(pts1, pts2)

res = cv.warpPerspective(img, Mp, (w,h), flags = cv.INTER\_CUBIC)

**Calibrazione**

cv.line(); cv.findChessboardCorners(); cv.getOptimalNewCameraMatrix(); cv.cornerSubPix(),

cv.drawChessboardCorners(); cv.calibrateCamera(); cv.undistort(),

cv.getOptimalNewCameraMatrix(); cv.RQDecomp3x3(), cv.Rodrigues().

**Filtri**

APPLICAZ FILTRO: cv.filter2D(img, cv.CV\_16S, filtro)

SEPARABILE: cv.sepFilter2D(img, cv.CV\_16S, filtrox, filtroy)

SFOCATURA: smoothed = cv.boxFilter(img, -1, (7,7), normalize = True)

GAUSSIANA: blurred = cv.GaussianBlur(img, (5,5), 0)

MEDIANA: med = cv.medianBlur(img, 9)

BORDI: edges = cv.Canny(blurred, 200, 100) #(img gaus, t1, t2)

GRADIENTE - DERIV: dx = cv.Sobel(img, cv.CV\_32F, 1, 0, scale=1/8)

dy = cv.Sobel(img, cv.CV\_32F, 0, 1, scale=1/8)

Oppure: dx, dy = cv.spatialGradient(InputImage)

mod = cv.magnitude(dx, dy) ang = cv.phase(dx, dy, angleInDegrees=True)

NORMALIZ: f/= f.sum() m = cv.normalize(m, None, 0, 255, cv.NORM\_MINMAX, cv.CV\_8U)

**Analisi**

TRASF DIST: dt4 = cv.distanceTransform(img, cv.DIST\_L1, 3) #d4

dt8 = cv.distanceTransform(img, cv.DIST\_C, 3) #d8

np.where(cv.distanceTransform(255-img, cv.DIST\_C, 3) > 9, 255, 0) #Pixel di background

con distanza minore di 9 da fore

CONTORNI: c, \_ = cv.findContours(img, cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_NONE)

res = cv.drawContours(img, c, -1, (255,0,0), 2) # (img, contorni, -1 → stampa tutti, colore, tratto linea)

COMP CONN: n, cc, stats, centroids = cv.connectedComponentsWithStats(bw)

ELIMINA PICCOLE: small = [i for i in range(1,n) if stats[i, cv.CC\_STAT\_AREA]<70]

mask[np.isin(cc, small)] = 0

ELEM STRUT: se = cv.getStructuringElement(cv.MORPH\_RECT/ELLIPSE, (15,15))

MORF MAT: res2 = cv.morphologyEx(img\_bw, cv.MORPH\_DILATE/ERODE/OPEN/CLOSE, se)

res[res2==0]=0

BORDI: img4 = img3 - cv.morphologyEx(img3, cv.MORPH\_ERODE,

cv.getStructuringElement(cv.MORPH\_ELLIPSE, (5,5))) #2px

**Video**

ESTRAI FRAME: cap = cv.VideoCapture(‘filevideo’)

ret, frame = cap.read() #ret falso se non ci sono piu frame. cap.release() #alla fine

MOG: mog = cv.createBackgroundSubtractorMOG2(detectShadows=False) mask = mog.apply(f)

RESIZE: cv.resize(frame, None, fx = 1/2, fy = 1/2) #dimezza risoluzione

**Riconoscimento**

TEMPL MATC: R = cv.matchTemplate(img, imgT, cv.TM\_metodo) #confronta imgT in img

CNN: net = cv.dnn.readNet('minst/model.onnx')

net.setInput(cv.dnn.blobFromImage(img)) out = net.forward()

HSV e HSL -> Tinta° (Hue), Saturazione, Luminosità (value/lightness) asse verticale

Bin. Imm -> soglia globale (manualmente, oss istogramma, metodo otsu) e locale

Contr. Stretchin -> f( I [y,x] ) = 255 \* (I [y,x] – alfa ) / (beta – alfa)

Eq. Isto -> distrib liv grigio, per colori applica su canale L

Trasf. Geom -> alle coord dei pixel e non valore (mapping inverso)

Traf. Affini -> [tx,ty] ∶ è la traslazione, θ: è la rotazione, s ∶ la scala

Filtri lin -> val del pixel calcolato come somma pesata dei pixel nell’intorno

Normalizz filtro -> se i coeff >=0 si divide per la somma dei coeff per valori [0,255]

Correl -> appl filtro con formula

Conv -> appl filtro con formula

Notaz -> Correl I’ = I O F, Convol I’ = I \* F

Box Filter -> tutti coef a 1, effetto sfocatura (blur), probl produce artefatti

Filtro Gaussiano -> pesi diminuiscono verso l’esterno (sfocatura)

Sharpening -> messa a fuoco.

1. si calcola imm blurred con convol (filtro box o gaus) e si sottrae all’originale

2. l’immagine calcolata si somma all’iniziale moltiplicata per parametro k

Bordi -> si trovano in cambiamenti rapidi di f, sottraendo smooth dall’originale

Filtri derivativi: utilizzati per calcolo gradiente per estrarre contorni da imm

-Prewitt -> meno sensib a var di luce e rum, sim risporigine, assi orient modo tradiz

-Sobel -> danno peso maggiore al pixel al centro

Gradiente -> vett che ha per compon le due d parz. direz maggior variaz imm nel punto

Canny edge det -> riconoscimento contorni

1.Smooth gauss dell’imm, 2. Calc gradiente per ogni pixel

3. Soppress non massimi -> elim pixel il cui modulo del gradiente non è max risp orient grad

4. Selez bordi con isteresi -> si binarizza imm con soglie t1, t2 ( validi con modulo > t1 o adiac)

Traform distanza di F risp F\* -> replica di F con pixel etichettati come val distanza da F\*

Algoritmo distanza -> scansione diretta e indiretta (per misuraz lung e spess e evid schelet)

Etichet comp conn ->scorre imm e per ogni pixel di foregr si consider i pixel già visitati

-Nessuno etic -> new etich -Uno etich -> stessa etich -Più di uno etich -> si annota equival.

Morf Mat -> tecnica derivata da teoria insiemi, per contorno e schelet o rimuovere particol

Dilataz (F O S) -> new img è insieme d pixel t.c. traslando S almeno uno dei suoi elem è sovrapp

Eros (F O S) -> new img è insieme d pixel t.c. traslando S l’intero elem strutturante è contenuto

Apertura-> eros-dilat, separa ogg debol connessi e rimuove regioni piccole

Chiusura-> dilat-eros, riempie buchi e piccole concavità, rafforza conn regioni deb conn