SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

AIS ID: 92320

HĽADANIE OBJEKTOV NA OBRAZE I

ZADANIE

Predmet:	I-BIOM – Biometria
Prednášajúci:	prof. Dr. Ing. Miloš Oravec
Cvičiaci:	Ing. Marián Šebeňa

Bratislava 2025 **Bc. Lukáš Patrnčiak**

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek

CLAHE	Contrast Limited Adaptive Histogram Equalization
FN	False Negatives
FP	False Positives
IoU	Intersection over Union
RGB	Red, Green, Blue
TP	True Positives

Úvod

Cieľom tohto zadania je v programovacom jazyky Python implementovať program, ktorý bude vedieť nájsť dúhovku oka na obraze. V rámci tohto zadania máme k dispozícii databázu fotografii (s .csv súborom s Ground Triths lokalizačnými kruhmi - *iris_annotation.csv*).

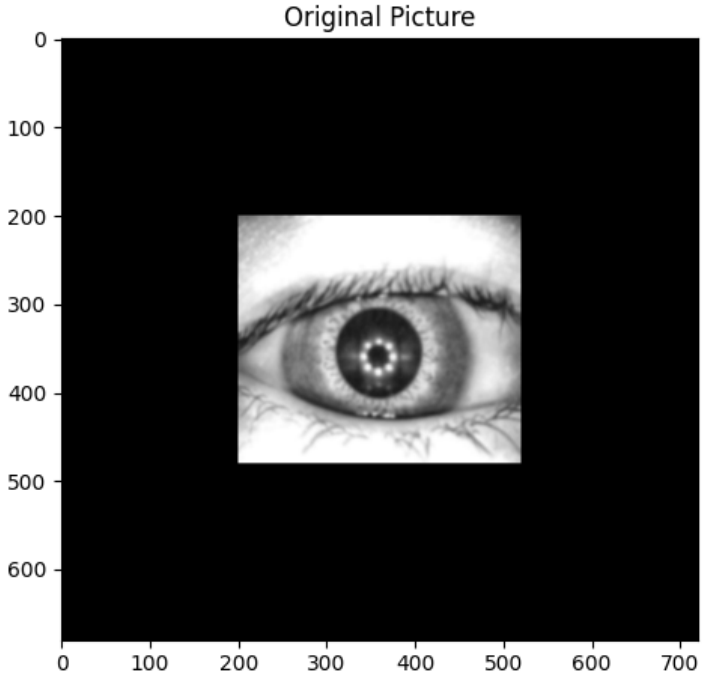
Úlohou je na tieto fotografie aplikovať metódy lokalizácie geometrických objektov a vyhodnotiť ich úspešnosť oproti 'pravdivím' pozíciam.

Najprv je potrebné načítať a zobraziť ľubovoľný obrázok z poskytnutého datasetu a potom bude potrebné nájsť pomocou Houghovej transformácie kružnice ohraničujúce zreničku, dúhovku, horné a spodné viečko. Následne bude do programu pridané vyhodnotenie úspešnosti tejto detekcie. Ďalší krok sa týka prehľadania celej databázy obrázkov, pričom optimálne parametre budeme hľadať pomocou mriežkového vyhľadávania. Posledný krok sa bude týkať segmentácie dúhovky podľa kružníc.

1 Spracovanie obrázka

Pre spracovanie dát - detekciu objektov, úpravu obrázkov, aplikovanie filtrov, detekciu hrán na obrázku a podobne bola použitá knižnica **OpenCV**[1], pre numerické výpočty bola použitá knižnica **NumPy**[2] a pre vizuálne zobrazenie kružníc na obrázku bola použitá knižnica **MatPlotLib**[3]. Pre spracované datasetu vo formáte .csv bola použitá knižnica **Pandas**[4] a pre spracovanie cesty k obrázkom v databáze bola použitá knižnica **Python OS**[5]

Prvým krokom bolo načítanie ľubovoľne vybraného obrázku oka z poskytnutého datasetu (bol vybraný obrázok *duhovky/029/R/S1029R03.jpg*). Tento obrázok bol načítaný funkciou *load_image()*. Aby sa zmestili všetky sterdy kružníc do obrázka, bol pridaný padding (200) v každom smere.



Obrázok 1: Načítaný obrázok oka aj s pridaným paddingom

Následne bolo potrebné pristúpiť k spracovaniu daného obrázka, pričom na toto spracovanie boli použité a implementované nasledovné techniky:

[1]https://opencv.org/
[2]https://numpy.org/
[3]https://matplotlib.org/
[4]https://pandas.pydata.org/
[5]https://docs.python.org/3/library/os.html

1. **Zvýšenie kontrastu obrázka**: Na tento účel bolo použité Contrast Limited Adaptive Histogram Equalization (CLAHE). Ide o techniku, ktorá upravuje kontrast obrazu dynamicky v malých blokoch (dlaždiciach), aby sa predišlo príliš silnému zlepšeniu kontrastu v homogénnych oblastiach, čo by mohlo spôsobiť šum. Na tento účet slúži funkcia *enhance_contrast()*, ktorej parametre sú:

    - *image*: Vstupný obrázok (v odtieňoch šedej, ktorý bol načítaný funkciou popísanou vyššie), na ktorý sa aplikuje zlepšenie kontrastu.
    - *clip_limit*: Parametrizuje hranicu (limit) pre clipping (odrezávanie) histográmu. Čím vyššia hodnota, tým menší je účinok clippingu (môže to viesť k agresívnejšiemu zvýšeniu kontrastu). V našom prípade bola zvolená hodnota 2.5.
    - *tile_size*: Veľkosť "dlaždíc"(tileGridSize) pre rozdelenie obrázka na malé bloky. V každom bloku sa vykoná histogramová ekvalizácia. V tomto prípade bola použitá veľkosť (8,8).

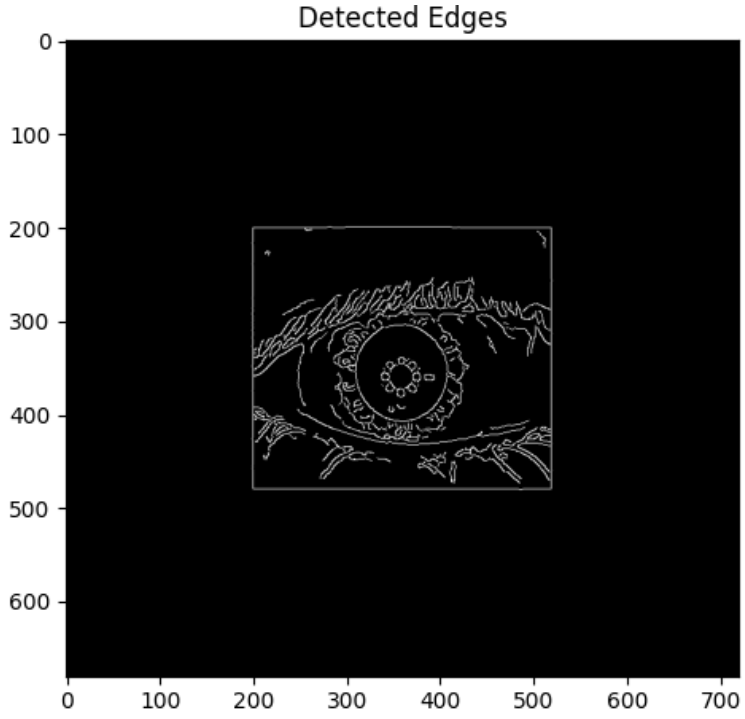
2. **Odstránenie šumu**: Prebehlo z obrázka pomocou Gaussovského rozmazania. Gaussovské rozmazanie je technika, ktorá priemeruje hodnoty pixelov okolo každého pixelu podľa Gaussovho jadra, čím pomáha zjemniť obraz a odstrániť šum.Na tento účet slúži funkcia *denoise_image()*, ktorej parametre sú:

    - *image*: Vstupný obrázok (v odtieňoch šedej, ktorý bol načítaný funkciou popísanou vyššie), z ktorého sa odstráni šum.
    - *kernel_size*: Veľkosť Gaussovského jadra (v tomto prípade (9, 9)). Väčšia veľkosť jadra spôsobí väčšie rozmazanie a odstránenie viac šumu, ale môže znížiť ostrosť obrazu.
    - *sigma*: Šírka Gaussovej krivky. Väčšia hodnota sigma vedie k väčšiemu rozmazaniu, čo môže byť užitočné pri silnejšom šume, ale môže tiež znížiť detaily v obraze V našom prípade bola použitá sigma 1.5.

3. **Detekcia hrán obrázku**: Prebieha v obrázku pomocou Cannyho detektora hrán. Cannyho detektor je jedným z najbežnejších algoritmov na detekciu hrán v obrázku, ktorý vyhľadáva rýchle zmeny intenzity a identifikuje okraje. Na tento účet slúži funkcia *detect_edges()*, ktorá vráti obrázok, kde sú hrany znázornené bielou farbou a zvyšok čiernou a ktorej parametre sú

    - *image*: Vstupný obrázok, na ktorom sa detegujú hrany.

- *low_threshold*: Dolná prahová hodnota pre detekciu hrán. Ak gradient intenzity je nižší ako tento prah, považuje sa to za šum a tieto hrany sa ignorujú. V tomtom prípade je dolná prahová hodnota nastavená na 60.
- *high_threshold*: Horná prahová hodnota pre detekciu hrán. Ak gradient intenzity je vyšší ako tento prah, hrany sú považované za silné a sú označené ako okraj. V tomtom prípade je dolná prahová hodnota nastavená na 120.



Obrázok 2: Detekované hrany na obrázku

4. **Houghova transformácia**: Pre detekciu kruhov v obrázku bola použitá Houghova transformácia, ktorá je presne na tento účet navrhnutá. Táto funkcia na základe nižšie popísaných parametrov vygeneruje niekoľko kružníc pre rôzne prvky oka (zrenička, dúhovka, horné a spodné viečko). Táto transformácia je súčasťou knižnice OpenCV (*cv2.HoughCircles*), ktorá je implementovaná v rámci funkcie *detect_circles()*, ktorej parametre sú:

   - *image*: Vstupný obrázok, v ktorom sa hľadajú kruhy.
   - *dp*: Rozlíšenie akumulátora. Hodnota určuje mieru rozlíšenia v akumulátore, ktorý sa používa na detekciu kruhov. Nižšie hodnoty zvyšujú presnosť detekcie, ale zároveň zvyšujú výpočtovú náročnosť. Bežná hodnota je 1 alebo 2.

- *min_dist*: Minimálna vzdialenosť medzi stredmi dvoch detekovaných kruhov. Toto zabraňuje detekcii viacerých kruhov, ktoré sú príliš blízko seba.
- *param1*: Maximálna hodnota pre detektor hrán (Cannyho detektor) použitý počas Houghovej transformácie. Určuje citlivosť pri detekcii hrán.
- *param2*: Minimálny prah pre detekciu stredov kruhu. Nižšia hodnota znamená väčšiu pravdepodobnosť detekcie, ale aj viac falošných pozitív.
- *min_radius*: Minimálny polomer detekovaných kruhov. To určuje minimálnu veľkosť kruhu, ktorý sa má detegovať.
- *max_radius*: Maximálny polomer detekovaných kruhov. To určuje maximálnu veľkosť kruhu, ktorý sa má detegovať.

Prehľad zvolených parametrov Houghovej transformácie pre jednotlivé elementy (zernička, dúhovka, horné a spodné viečko) v oku sú popísané s nasledujúcej tabuľke:

S1029R03.jpg	dp	min_dist	param1	param2	min_radius	max_radius
Zrenička	2.2	5	100	50	40	50
Dúhovka	2.2	5	100	30	95	120
Horné viečko	2.5	100	100	50	230	290
Spodné viečko	2.5	100	100	50	230	310

Tabuľka 2: Prehľad zvolených parametroch Houghovej transformácie

Parametre 'min_radius' a 'max_radiuspre' zreničku a dúhovku boli zámerne vybrané s nerovnosťou, že zrenička bude mať menší minimálny a maximálny polomer ako dúhovka, aby sme už pri detekovaní kružníc vybrali tie, ktoré spĺňajú reálny možný rozdiel veľkostí medzi týmito dvomi prvkami.

1.1 Filtrovanie pomocou pravidiel

Filtrovanie detekovaných kružníc Houghovou transformáciou je dôležité preto, aby boli vybrané čo najpresnejšie kružnice kopírujúce okraje zreničky, dúhovky, horného a spodného viečka oka (keďže množstvo z detekovaných nespĺňa požadované kritériá). Každý prvok (zrenica, dúhovka, viečka) má špecifické geometrické vlastnosti (veľkosť, umiestnenie), ktoré boli zachytené v pravidlách filtrácie kružníc. Tieto pravidlá sú implementované vo funkcii *filter_circles()*.

Vstupy funkcie:

- *detected_circles*: Slovník detekovaných kruhov, kde kľúče sú názvy prvkov ("pupil", ïris", üpper_eyelid", "lower_eyelid") a hodnoty sú zoznamy detekovaných kruhov vo formáte (x, y, r) (stred a polomer kruhu).
- *img_shape (tuple)*: Rozmery obrázka (výška, šírka).
- *tolerance (float)*: Povolená odchýlka od stredu obrázka (v rozsahu 0 - 1).

Najprv bolo potrebné získať rozmery obrázka v tvare (výška, šírka). Potom bolo potrebné nájsť stred obrázka (výška/2, šírka/2). Keďže nie vždy sa môže podariť detekovať kružnice presne v strede, bola zavedená tolerancia ($T \in < 0, 1 >$) (výška $\cdot T$, šírka $\cdot T$). Následne bol inicializovaný slovník pre nájdenie najlepších kruhov

Potom sme sa pokúsili nájsť strednú (priemernú) hodnotu y-ovej súradnice všetkých detekovaných kruhov zreničky (táto hodnota by mala byť ideálne rovnaká aj pre dúhovku, y-ová pozícia je kľúčová pri správnom umiestnení zreničky a dúhovky vzhľadom na ostatné časti oka a umožňuje správne vybrať najlepší detekovaný kruh), ktoré patria k dúhovke. Táto získaná hodnota je priemerná y-ová pozícia všetkých detegovaných kruhov pre dúhovku, čo nám dáva približnú vertikálnu polohu stredu dúhovky.

Následne je potrebné skontrolovať celý slovník uložených dúhoviek (pričom pokiaľ nebola nájdená žiadna kružnica pre daný prvok oka, prehľadávanie preskočíme a prejdeme na ďalšie). Samotná filtrácia prebieha potom nasledovne:

1. **Najbližší kruh k stredu obrázka pre zrenicu a dúhovku**: V tejto časti kontrolujeme, či sa kruh nachádza v okolí stredu obrázka, pričom k dispozícii máme súradnice stredu obrázka a taktiež aj k ním prislúchajúce tolerancie, ktoré určujú, aký veľký priestor okolo stredu sa považuje za 'okolie'.

   Ak sa kruh nachádza v okolí stredu obrázka, vypočíta sa vzdialenosť medzi stredom obrázka a stredom kruhu. Táto vzdialenosť sa používa na vyhodnotenie, ktorý kruh je najbližšie k stredu obrázka.

   Ak je táto vzdialenosť menšia než predchádzajúca najmenšia vzdialenosť, tento kruh je považovaný za najbližší a je uložený do vytvoreného slovníka.

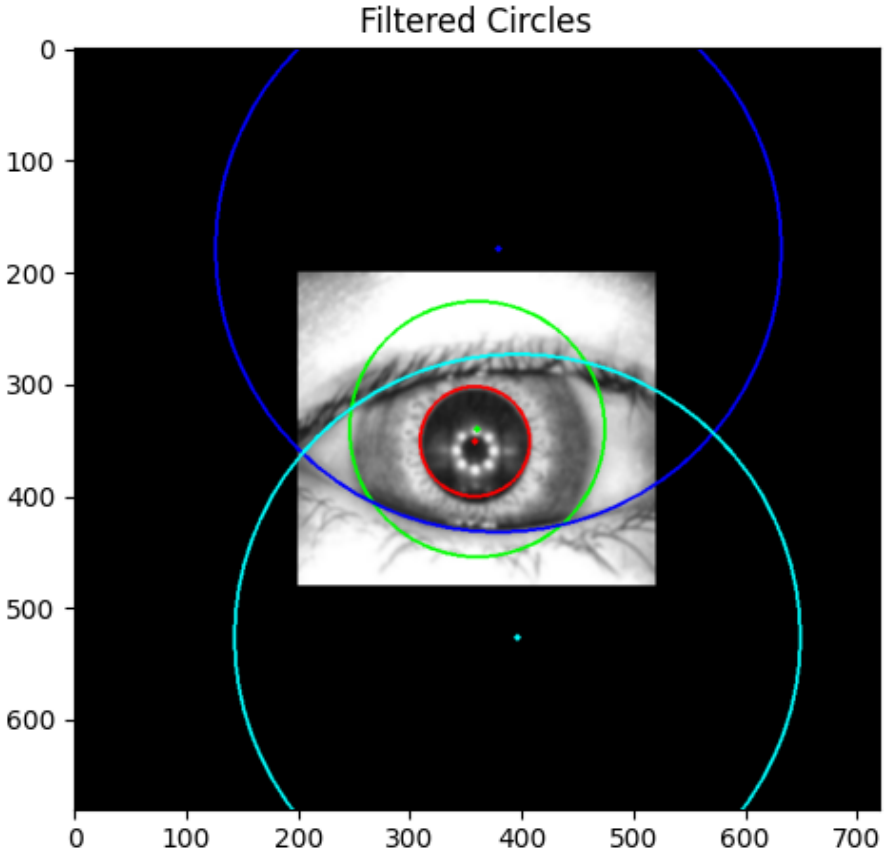
   Po prehľadaní všetkých detegovaných kruhov sa do slovníka uloží najbližší kruh, alebo ak žiadny kruh nebol nájdený, uloží sa prvý kruh zo zoznamu kruhov ako predvolený.

2. **Výber kruhu pre horné viečko medzi zreničkou a horným okrajom**: V tomto prípade spracovávame horné viečko, pričom kľúčovou podmienkou je, aby

bol detekovaný stred zreničky oka. Hľadáme kruh, ktorý sa nachádza najbližšie k hornému okraju obrázka a následne vypočítame hodnotu y-ovej súradnice, ktorá sa nachádza medzi zreničkou a najvyšším bodom pri hornom okraji obrázka. Následne vyberáme kruh, ktorý je najbližšie k požadovanej y-ovej hodnote (výber kruhu s najmenšou vertikálnou vzdialenosťou od tejto vypočítanej y-ovej hodnoty). Tento kruh sa následne uloží ako najlepší pre horné viečko.

3. **Výber kruhu pre dolné viečko medzi zrenicou a dolným okrajom**: Tento prípad filtrácie je analogický k výberu kruhu pre horné viečko medzi zreničkou a horným okrajom, avšak v tomto prípade sa vyberá kruh, ktorý je medzi zreničkou a dolným okrajom obrázka. Zisťuje sa najnižší bod zo všetkých detegovaných kruhov (kruh, ktorý je najbližšie k dolnému okraju obrázka). Následne sa vypočíta stred medzi zrenicou a dolným okrajom a potom sa vyberie kruh, ktorý je najbližšie k tejto požadovanej stredovej výške. Vybraný kruh sa uloží ako najlepší kruh pre dolné viečko.

4. **Dodatočná kontrola veľkosti zreničky a dúhovky**: V tejto funkcii prebieha taktiež kontrola, či je zrenička menšia ako dúhovka. Pokiaľ nie je menšia, kružnica sa preskočí a pokračuje sa ďalej a pokiaľ je podmienka splnená, pridá sa do slovníka najúspešnejších kružníc (podľa zvolených pravidiel).



Obrázok 3: Detekované kružnice pre jednotlivé prvky v oku po filtrovaní pomocou pravidiel

1.2 Segmentácia dúhovky

Podľa detekovaných kružníc na vybranom obrázku bola vykonaná segmentácia dúhovky. Táto funkcionalita je implementovaná prosterdníctvom funkcie *segment_iris()*.

Najprv bolo potrebné získať dáta zreničky, dúhovky, horného a spodného viečka v tvare (výška, šírka, polomer). Pre následnú manipuláciu s kružnicami pre jednotlivé príznaky je používané *cv2.circle()*, pričom postup pozostáva z nasledovných krokoch:

1. Inicializácia čiernej masky (M_1).

2. Aktualizácia masky pre dúhovku - kružnica dúhovky, vrátane jej vnútra, bola nastavená na farbu (1,1,1) (biela farba v RGB rozsahu 0 - 255), pričom všetko ostatné ostáva nastavené na farbu 0.

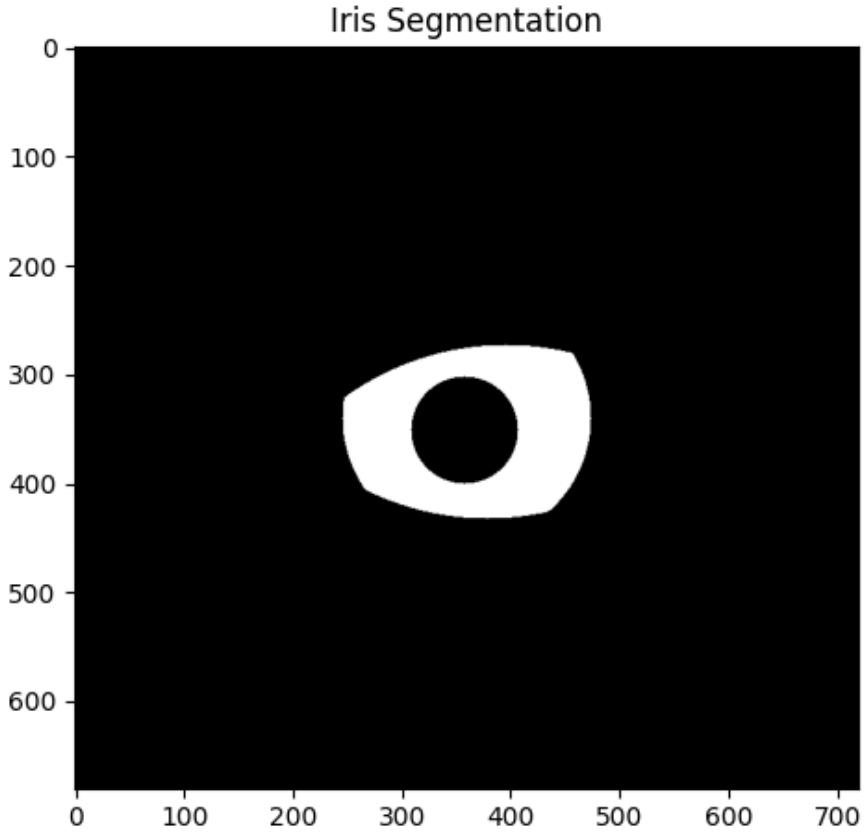
3. Aktualizácia masky pre zreničku - kružnica zreničky, vrátane jej vnútra, bola nastavená na farbu (0,0,0) (čierna farba v RGB rozsahu 0 - 255), pričom všetko ostatné ostáva nastavené podľa predošlých kritérii.

4. Inicializácia čiernej masky pre horné (M_u) a spodné (M_d) viečko, pričom obidva kruhy boli nastavené na farbu 1, vrátane ich vnútra.

5. Kombináciou hodnoty pôvodnej masky a masky pre horné a spodné viečko dostaneme požadovanú oblasť: $M = M_1 \cdot M_u \cdot M_d$.

6. Aplikácia masky na pôvodný obrázok - tam, kde hodnota výslednej masky M nadobúda hodnotu 1 (True -pixel patrí do požadovanej výslednej oblasti dúhovky), nastaví sa hodnota pixelu na 1 (biela). V opačnom prípade sa nastaví na hodnotu 0 (čierna). Táto funkcionalita je zabezpečená pomocou funkcie *np.where()*, ktorá je v kóde pridaná kvôli zdôrazneniu princípu fungovania.

Je dôležité poznamenať, že v tomto prípade pracujeme s binárnou maskou, preto hodnoty farby pre požadované oblasti sú nastavené na vektor (1,1,1) - biela farba pre požadovanú oblasť, resp. (0,0,0) - čierna farba pre oblasti, ktoré chceme zanedbať. Z uvedeného dôvodu nie je možné brať do úvahy iné farby, ako tieto dve binárne hodnoty - biela a čierna.



Obrázok 4: Segmentovaná dúhovka

2 Vyhodnotenie úspešnosti detekcie

Pre vyhodnotenie úspešnosti detekcie bolo potrebné vypočítať hodnotu Intersection over Union (IoU) medzi dvomi kruhmi. Táto metrika je používaná na hodnotenie kvality detekcie tvarov, ktorá vyjadruje pomer prieniku oblastí dvoch kruhov k ich zjednotenej oblasti. Funkcia využíva geometrické výpočty, ktoré sa opierajú o vzdialenosť medzi stredmi dvoch kruhov a ich polomery. Výpočet pre prienik dvoch kružníc bol prebraný z [1], resp. z [2] pre výpočet IoU a vyhodnotenie úspešnosti.

Toto vyhodnotenie je implementované vo funkcii **iou()**, ktorá má 2 parametre - kružnice vo formáte (x, y, r), resp. (x-ová súranica stredu, y-ová súradnica stredu, polomer kruhu), pričom máme 2 kruhy - A a B.

- **Kružnica A**: stred (x_a, y_A) a polomer r_A
- **Kružnica B**: stred (x_B, y_B) a polomer r_B

Vzdialenosť d medzi stredmi kruhov A a B vypočítame podľa vzorca

$$d = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Plochy jednotlivých kruhov vypočítame ako $A = \pi \cdot r_a^2$ a $B = \pi \cdot r_B^2$. Ak sa kruhy nepretínajú $(r_A + r_B > d)$, IoU je rovné 0.

Ak je vzdialenosť medzi stredmi kruhov menšia alebo rovná absolútnej hodnote rozdielu ich polomerov $(d \leq r_A - r_B)$, znamená to, že jeden kruh je úplne vnútri druhého. Tento stav nastane, keď je menší kruh úplne obsiahnutý väčším kruhom. V tomto prípade je prienik oblasť menšieho kruhu, pretože celý menší kruh je obsiahnutý v tom väčšom. Takže oblasť prieniku je oblasť menšieho kruhu: $\pi \cdot min(r_a, r_B)^2$.

Plochu prieniku dvoch kružníc potom vypočítame

$$A \cap B = r_A^2 \cdot \arccos\left(\frac{d^2 + r_A^2 - r_B^2}{2 \cdot d \cdot r_A}\right) + r_B^2 \cdot \arccos\left(\frac{d^2 + r_B^2 - r_A^2}{2 \cdot d \cdot r_B}\right) - T,$$

$$T = \frac{1}{2}\sqrt{(-d + r_A + r_B)(d + r_A - r_B)(d - r_A + r_B)(d + r_A + r_B)}$$

kde:

- r_A a r_B sú polomery dvoch kruhov,
- d je vzdialenosť medzi stredmi týchto dvoch kruhov.
- arccos je inverzná funkcia kosínus, ktorá sa používa na výpočet uhla medzi bodmi, ktoré sa nachádzajú na obvodoch kruhov.

IoU je definované ako podiel oblasti prieniku k oblasti spojenia. Teda:

$$IoU = \frac{A \cap B}{A \cup B},$$

pričom $A \cup B = A + B - A \cap B$. Tento výpočet realizuje funkcia *iou()*, ktorá dostáva ako vstupné parametre 2 kružnice s rozmermi (x, y, r).

Následné vyhodnotenie presnosti kruhov prebieha na základe porovnaní medzi vybranými (vyfiltrovanými) najlepšími detekovanými kruhmi a kruhmi obsahujúce **Ground Truth** hodnoty (originálne ohdnoty) kruhov pre dané príznaky. Taktiež je tu zadaná prahová hodnota 0.75, čiže odchýlka, resp, tolerancia nepresnosti nájdených kruhov pre ich úspešnú klasifikáciu. Pre tento účel bolo potrebné vypočítať IoU pre jednotlivé príznaky.

Princíp vyhodnotenia presnosti jednotlivých kruhov prebieha nasledovne, pričom budeme pracovať s 3 množinami - True Positives (TP) - správne detekované kruhy, False Positives (FP) - chybné detekcie a False Negatives (FN) - kruhy, ktoré mali byť detekované, ale neboli. Je potrebné poznamenať, že tieto metriky sa vypočítavajú pre každý príznak (zrenička, dúhovka, horné a spodné viečko samostatne):

1. **Iterácia cez všetky ground truth kružnice**. V tejto časti pracujeme so slovníkom obsahujúci pár názov kružnice - (výška, šírka, polomer) kružnice, pričom pokiaľ sa názov kružnice zhoduje s tým, ktorý je aj v slovníku s vyfiltrovanými detekovanými kruhmi, vypočíta sa IoU kruhu.
   - Ak IoU presahuje prahovú hodnotu, započíta sa TP.
   - Ak IoU nedosiahne prahovú hodnotu, započíta sa FP.
   - Ak sa kruh nenachádza medzi vyfiltrovanými detekovanými kruhmi, započíta sa FN.
2. **Výpočet metrík**. Tento výpočet prebieha v percentách, resp. v rozsahu od 0 po 1.
   - **Precision**: (koľko z detekovaných kruhov bolo správnych).

     $$\text{Precision} = \frac{TP}{TP + FP}$$

   - **Recall**: (koľko správnych kruhov sa podarilo detekovať).

     $$\text{Recall} = \frac{TP}{TP + FN}$$

   - **F1-Score**: (harmonický priemer precision a recall).

     $$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

S1029R03.jpg	IoU	Precision	Recall	F-1 Skóre
Zrenička	0.8692	1.0	1.0	1.0
Dúhovka	0.79850	1.0	1.0	1.0
Horné viečko	0.8105	1.0	1.0	1.0
Spodné viečko	0.5777	0.75	1.0	0.8571428571428571

Tabuľka 3: Tabuľka vypočítaných hodnôt IoU, Precision, Recall a F1- Skóre pre jednotlivé príznaky

Tento výpočet realizuje funkcia *evaluate_detection()*, ktorá dostáva ako vstupné parametre odfiltrované a ground truth hodnoty kružníc v tvare (x, y, r) a taktiež povolený treshhold (odchýlku), ktorá je zadaná na 0.75.

3 Mriežkové vyhľadávanie

V tomto bode je potrebné prejsť celú datbabázu obrázkov mriežkovým vyhľadávaním (Grid Search), ktorá má tvar

/duhovky

· koreňový adresár

/N

· poradie jednotlivých obrázkov

/L

· adresár obsahujúci ľavú stranu oka
/S1{N}{L}{M}.jpg

· snímka ľavého oka

/P

· adresár obsahujúci pravú stranu oka
/S1{N}{P}{M}.jpg

· snímka pravého oka

kde N je číslo v tvare 'XXX', L, R znamená ľavé, resp. pravé oko a M je poradové číslo obrázka v tvare 'X' (príklad - štruktúra cesty k uvedenému obrázku, s ktorým pracujeme - 'duhovky/029/R/S1029R03.jpg').

Toto vyhľadávanie je realizované funkciou *grid_search()*, ktorá ako parameter dostáva obrázok, detekované hrany a rozsahy testovania minimálneho a maximálneho polomeru pre jednotlivé kružnice (kvôli náročnosti výpočtov neboli ostatné parametre menené - boli použité z predchádzajúcej úlohy). Rozsahy sú v tvare (minimum, maximum, krok iterácie).

Obrázok	minimum	maximum	krok iterácie
Zrenička	20	60	5
Dúhovka	70	120	5
Horné viečko	180	250	20
Spodné viečko	180	270	20

Tabuľka 4: Tabuľka rozsahu pre min. a max. polomer pre Grid Search

Funkcia cyklicky testuje všetky možné kombinácie minimálneho a maximálneho polomeru, s daným iteračným krokom, pričom pokiaľ sú detekované nejaké kružnice, uložia sa do slovníka s názvom príznaku a taktiež aj aktuálnom konfiguráciou parametrov. Potom prebehne filtrácia kružníc (podľa vyššie spomínanej funkcie (*filter_circles()*), ktoré

sa následne rozdelia na 2 skupiny - parametre kružnice v tvare (x, y, r) a parametre konfigurácie Houghovej transformácie.

Funkcia *grid_search()* je potom zavolaná v cykle, v ktorom sa prehľadáva celá databáza, ciže zisťujú sa konkrétne cesty k jednotlivým obrázkom, pomocou ktorých, ako to bolo v kapitole 1 už vysvetlené, prebieha načítavanie obrázku podľa cesty, zvýšenie kontrastu, odstránenie šumu a detekcia hrán.

Potom boli prostredníctvom funkcií *grid_search()* a *evaluate_detection()* vypísané a vypočítané jednotlivé parametre, kružnice a metriky (kapitola 2) a následne bolo pomocou váženého priemeru F-1 skóre pre jednotlivé príznaky oka vypočítané celkové F-1 skóre:

$$F1_{total} = \frac{F1_{pupil} + F1_{iris} + F1_{upper\_eyelid} + F1_{lower\_eyelid}}{4}$$

. Posledný krok spočíval v nájdení názvu obrázku, resp. jeho cesty v datasete, ktorý dosahuje najlepšie priemerné F1-skóre. Do výsledného slovníka bol teda vložený kľúč s názvom, resp. cestou k danému najúspešnejšiemu obrázku a hodnoty, ktoré obsahujú parametre Houghovej transformácie, parametre kružníc v tvare (x, y, r) a dosiahnuté úspešnosti IoU, Precision, Recall a F1-skóre pre jednotlivé príznaky.

S1134L06.jpg	dp	min_dist	param1	param2	min_radius	max_radius
Zrenička	2.2	5	100	50	60	80
Dúhovka	2.2	5	100	30	120	170
Horné viečko	2.5	100	100	50	240	360
Spodné viečko	2.5	100	100	50	260	390

Tabuľka 5: Tabuľka parametrov Houghovej transformácie pre najlepší obrázok z databázy

S1134L06.jpg	X-ová súradnica	Y-ová súradnica	Polomer
Zrenička	160	138	79
Dúhovka	157	140	123
Horné viečko	306	136	271
Spodné viečko	24	261	310

Tabuľka 6: Tabuľka parametrov jednotlivých kružníc príznakov pre najlepší obrázok z databázy

S1134L06.jpg	IoU	Precision	Recall	F1-skóre
Zrenička	0.0	0.0	0	0.0
Dúhovka	0.0	0.0	0	0
Horné viečko	0.79	0.33	1.0	0.5
0 Spodné viečko	0.058	0.25	1.0	0.4

Tabuľka 7: Vypočítané metrické hodnoty pre najlepší obrázok z databázy

Ako je už z obrázkov zrejmé, neboli detekované presné kružnice pre najlepší obrázok z celej databázy - to môže byť spôsobené predovšetkým tým, že (predovšetkým kvôli potrebe vysokého výkonu a dlhej doby trvania behu cyklov) bol Grid Search obmedzený len na kombináciu minimálnych a maximálnych polomerov jednotlivých kružníc. Zmena všetkých parametrov Houghovej transformácie by mohla teoreticky kladne ovplyvniť výsledky. Taktiež by bolo možné zmeniť rozsahy pre minimálny, resp. maximálny polomer jednotlivých kružníc. Pre urýchlenie detekcie a výpočtov bola taktiež databáza zredukovaná na 155 párov obrázkov (L a R). Najlepší obrázok má cestu 'duhovky/134/L/S1134L06.jpg'.

Záver

Cieľom zadania bolo implementovať program v programovacom jazyku Python, ktorý detekuje kružnice pre zreničku, dúhovku, horné a spodné viečko oka. Ako bolo možné vidieť, detekcia v prípade jednej vzorky prebehla (v rámci istých odchýlok) úspešne, zatiaľ čo prehľadávanie celej databázy obrázkov oka (Grid Search)a následne detekciou týchto kružníc bolo o niečo zložitejšie a výsledky neboli príliš úspešné (vzhľadom na iteratívne 'skúšanie' kombinácie min. a max. polomeru pre jednotlivé príznaky a taktiež redukciu dataabázy obrázkov).

Zoznam použitej literatúry

1. WOLFRAM, MathWorld. *Circle-Circle Intersection*. Dostupné tiež z: `https://mathworld.wolfram.com/Circle-CircleIntersection.html`.
2. BOESCH, Gaudenz. *What is Intersection over Union (IoU)?* 2004. Dostupné tiež z: `https://viso.ai/computer-vision/intersection-over-union-iou/`.