M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem

Intelligens módszer párok keresésére képek nagy halmazában

Távközlési és Médiainformatikai Tanszék

Mogyorósi Ferenc

TDK dolgozat

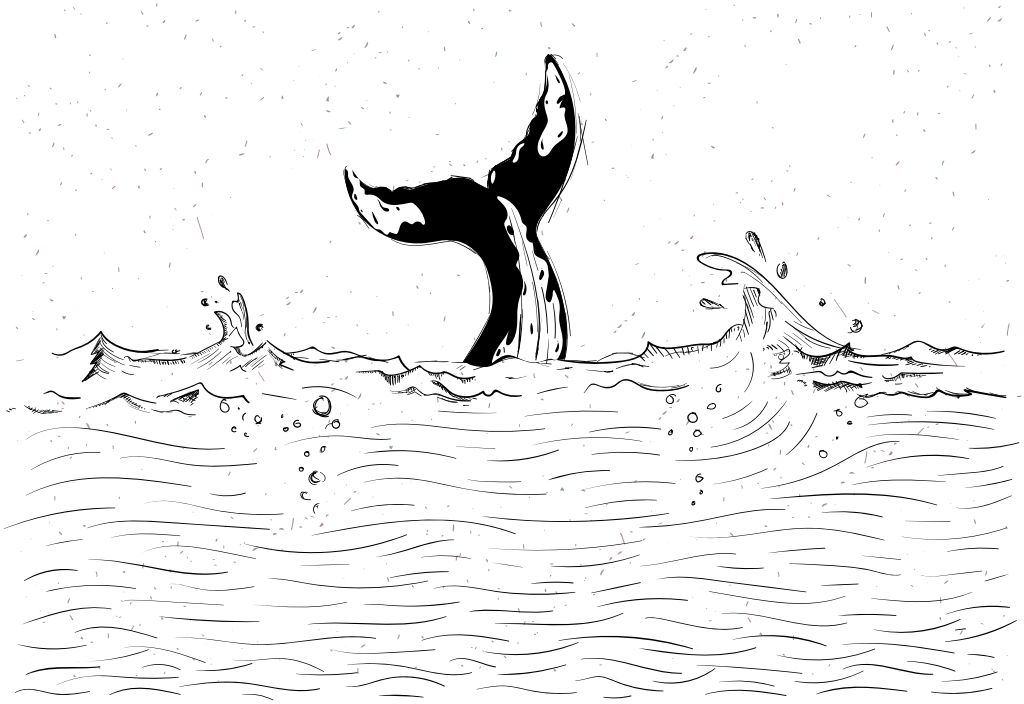
Konzulensek: dr. Szűcs Gábor, Papp Dávid

Budapest, 2017

Mogyorósi Ferenc: Intelligens módszer párok keresésére képek nagy halmazában

This document was typeset in LaTeX 2_ε.

Elérhetőségek:
Mogyorósi Ferenc – mogyi006@gmail.com
dr. Szűcs Gábor – szucs@tmit.bme.hu



Tartalomjegyzék

1. Bevezetés

Az okostelefonok elterjedésével párhuzamosan óriási népszerűségre tett szert a fényképezés. Egyre többen örökítik meg életük eseményeit okostelefonjukkal és töltik fel egy képmegosztó oldalra (pl. Instagram, Facebook) vagy a felhőbe. A megnövekedett képmennyiség miatt előtérbe került a kérdés, hogy hogyan lehet hatékonyan rendezni a képeket lehetőleg automatikus módon. Képeinket általában a rajtuk szereplő személyek, élőlények (pl. kiskutyánk) vagy tárgyak szerint szeretnénk rendszerezni. Ezek felismerésére a képfeldolgozás már kipróbált, hatékony eszközöket biztosít, melynek eredményeit tovább javíthatjuk gráfelméleti megfontolásokkal.

1.1. Cél és a tervezett megoldás

Tudományos diákköri munkám céljának egy olyan összetett rendszer létrehozását tűztem ki, amely képes egy nagy, címkézetlen képadathalmazban párokat/csoportokat megtalálni. Az általam megtervezett és implementált rendszer több részfeladatra bomlik, mindegyik egy-egy részproblémát old meg. A következőkben röviden bemutatom ezeket a részfeladatokat, a lehetséges módszereket, és azokat a saját megoldásokat, melyeket a munkám során megvalósítottam. A rendszerem először előfeldolgozza a képeket, majd egy tömörebb reprezentációba transzformálja őket, úgymond jellemzőket nyer ki belőlük. A jellemzőkinyerés keretében az egyes képeken kulcspontokat határoz meg, majd ezeknek a környezetét egy vektorral jellemzi. Két kép közötti hasonlóság meghatározásához először a képek kulcspontjainak párokba rendezése szükséges. Ezalapján a párosítás alapján kereshetünk egy transzformációt, ami a kulcspontokat egymásba viszi. Az ötletem az volt, hogy ha van olyan transzformáció ami sok kulcspontpárra illeszkedik, akkor a transzformáció mátrixát megvizsgálva következtethetünk a két kép hasonlóságára. Ha minden képpárra meghatározzuk a hasonlósági értéket egy hasonlósági mátrixot kapunk. Ha a képhalmazra súlyozott gráfként tekintünk, ahol a csúcsok az egyes képek, az élek súlya pedig a két kép közötti hasonlósági érték, akkor az elgondolásom szerint egy klaszterező algoritmussal klasztereket alakíthatunk ki a gráfban.
Minden egyes probléma megoldására is több kipróbált módszer állt a rendelkezésemre, ezért igyekeztem mindig a rendelkezésemre álló erőforrások és az adathalmaz méreteit és tulajdonságait fegyelembe vevő megoldást választani. Ehhez a módszerhez igazítottam a képek előfeldolgozását is, amivel a

képek azon tulajdonságait próbáltam erősíteni, melyek az általam választott jellemzőkinyerő módszerhez fontosak. Mivel a jellemzőleírás során minden kulcspont környezetét egy 128 dimenziós vektorral jellemezzük, ezért a tervezésem során úgy láttam, hogy a kulcspontok párosítása fogja igényelni a legnagyobb számítási kapacitást. Ennek csökkentése érdekében egy olyan algoritmust választottam ami nagy valószínűséggel találja meg az egyes kulcspontokhoz legjobban illő párt a másik képen. A klaszterező algoritmusok közül is egy kisebb számítási igényű és gyorsabb megoldást választottam, ami ennek ellenére nagyon jó eredményeket adott. A kompromisszumok ellenére jó eredmények elérését tűztem ki célul, amit sikerült is elérnem.

1.2. A dolgozat felépítése

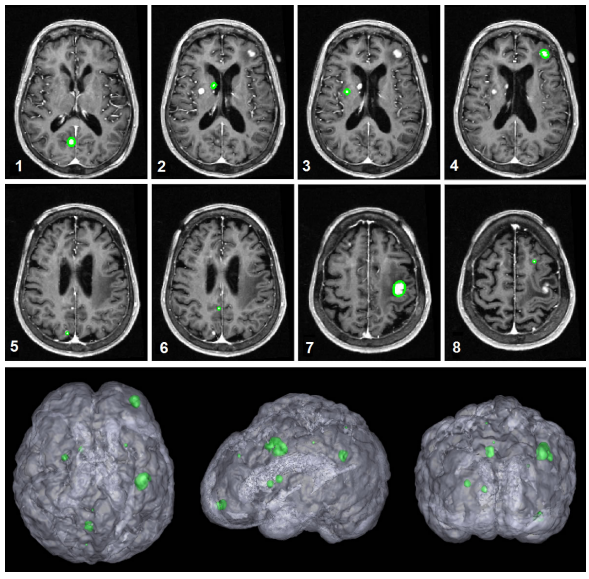
A következő fejezetben áttekintést nyújtok a képfeldolgozás alkalmazásairól és különféle problémákról. Bemutatom az általam használt algoritmusok elméleti hátterét és kiválasztásuk szempontjait. Áttekintem a klaszterezés matematikai hátterét és gyakorlati oldalát. Röviden összegzem azon szakirodalmi cikkeket, melyek hasonló problémára kerestek választ vagy hasonló módon oldottak meg egy problémát. A 3. fejezetben bemutatom a redszer felépítését, a használt eszközöket és a felmerült nehézségeket. A rendszert egy kisebb teszthalmazon saját magam is kiértékeltem, valamint egy nagyobb halmazon is kiértékelték annak a nemzetközi képfeldolgozási verseny szervezői, akiknek a versenyére jelentkeztünk többek között az én megoldásommal. Az utolsó fejezetben összegzem az eredményeket és felvázolom a lehetséges alkalmazási és továbbfejlesztési lehetőségeket.

2. Irodalmi áttekintés

A képfeldolgozás napjaink egyik legnagyobb kihívása. Rendkívül gyakran kerülünk szembe olyan helyzetekkel amik hosszú, monoton kategorizálást igényelnek. Gondoljunk csak a csekkbefizetésre vagy levélfeladásra, ahol rendkívül sok címet, irányítószámot és egyéb dolgot kell a dolgozónak felismerni és bevinni a rendszerbe. Ezek a problémák hozták előtérbe a képfeldolgozást ami lehetővé teszi, hogy a gép egy fotó alapján valamilyen karakterfelismerő algoritmus segítségével beolvassa ezeket a mezőket egy pillanat alatt. Az 1. táblázatban néhány példát látunk a képelemzés kihívásaira és felhasználásukra.

Alkalmazások	Területek
Levél szortírozás, címkeolvasás	Dokumentumfeldolgozás
Tumordetektálás, kromoszómaelemzés	Orvosi képelemzés
Minőségellenőrzés	Ipari automatizálás
Tárgy- és környezetfelismerés	Robotika
Állatok és növények felismerése	Tengeri populációfigyelés

1. táblázat. A képelemzés alkalmazásai



1. ábra. Tumor szegmentáció képek alapján

A képelemzés rendkívül sokféle módon és eszközzel történhet. A legalapvetőbb klasszifikáció, amikor a képeinket osztályokba akarjuk besorolni aszerint, hogy mi látható rajtuk. Ennek a legjobb példája a számjegyfelismerés, ahol rendívül nagy és jó minőségű adathalmazok állnak rendelkezésre. A legismertebb az MNIST adatbázis ami 60.000 tanító- és 10.000 tesztképet tartalmaz, ebből néhányat a 2. ábrán láthatunk. Ezt a mai redszerek 99,9 %-os pontossággal képesek osztályozni, ami akár az emberi hatékonyságot is meghaladhatja, ami 99,7 % körül van.

2. ábra. Mintaképek az MNIST adatbázisból

A feladat jellegét a rendelkezésre álló információ mennyisége, minősége és címkézése határozza meg. Ha sok, jó minőségű kép áll rendelkezésünkre, melyek megfelelően címkézve vannak, akkor kiválasztjuk, hogy milyen osztályokat akarunk megkülönböztetni és mindegyikhez hozzárendeljük a pozitív példáit. Ezeket az összerendelt (kép-osztály) párokat tanítóhalmaznak nevezzük. Ezután már csak azt kell elérnünk, hogy a rendszerünk megtanulja mely képekre milyen osztálycímkét kell adnia. A tanulás általában úgy történik, hogy a tanítóhalmazon végigmenve valamilyen hibát képzünk, ezt a hibát visszavezetjük az egyes paraméterekre, amiket ez alapján módosítunk. Ezt a megoldást felügyelt tanulásnak nevezzük.

Nehezebb a probléma, ha csak kevés felcímkézett kép áll rendelkezésre. Ekkor gyakran használt eszköz, hogy különféle képmanipulációkkal egy képből többet csinálunk. Ilyen például a kép eltolása x-y irányban, forgatása és nyújtása. Ezen kívül megváltoztathatjuk az egyes pixelek értékét vagy ugyanannyival vagy más-más mértékben például valamilyen Gaussi zaj hozzáadásával. Ezek a módszerek nagy tanítóhalmaz esetén is jól alkalmazható-

ak, mivel segíti a rendszer általánosító képességét, ami segít jó választ adni ha új elem érkezik. Nyilván a tanítással a célunk az, hogy a látott minták alapján a rendszer olyan általános jellemzőket tanuljon meg, amik a valós helyzetekben is jó választ váltanak ki.

Egy másik probléma az lehet, ha sok adatunk van, viszont azok egyáltalán nincsenek felcímkézve. Ekkor is szeretnénk információt kinyerni a képekből viszont nem tudjuk, hogy mit keressünk. Ezt felügyelet nélküli tanulásnak nevezzük, mivel nincsenek elvárt bemenet-kimenet párjaink. Itt is nagyon sok lehetőségünk van, hogy a képek közötti összefüggéseket megkeressük és megértsük. Ilyen probléma például a képpárosítás. Tegyük fel, hogy nagy mennyiségű képünk van melyek címkézetlenek, de tudjuk, hogy egyes képeken ugyanaz az objektum szerepel. Ekkor úgy szeretnénk szétválasztani a képeket klaszterekre, hogy az egyes klaszterekben szereplő képeken megtalálható legyen ugyanaz az objektum.

2.1. A jellemzőkinyerés lehetőségei

Az előfeldolgozás során alacsony szintű képmódosító műveleteket hajtunk végre [1]. A bemenet és kimenet is egy-egy intenzitás kép, általában egy mátrixban megadva. Az előfeldolgozás célja a kép minőségének javítása azzal, hogy elnyomja a nemkívánt torzításokat és kiemel néhány jellemzőt a későbbi feldolgozási lépésekhez. Az itt használt eljárások a kép pixeleinek redundanciáját veszik figyelembe, vagyis hogy az egy objektumhoz tartozó szomszédos pixelek értékei ugyanakkora vagy hasonló intenzitásértékekkel rendelkeznek [2]. Így ha egy kilógó értéket találunk az helyettesíthető a szomszédos értékek átlagával. Általában elmondható, hogy az előfeldolgozás során matemetikai normalizációt hajtunk végre az adathalmazon. Az előfeldolgozáshoz hozzátartozik a képjellemzők kinyerése a kép tömörebb leírásának céljából. Ilyen például a Scale-Invariant Feature Transform (SIFT) [3], mely lokális régiók gradiensén alapszik és a Speeded-Up Robust Features (SURF) [4], mely előre meghatározott jellemzőket használ a lokális régiókban. Ezek a metódusok úgymond kép-piramisokat használnak (többféle nagyságban használják fel a képet), hogy egy skálatérbeli reprezentációt érjenek el. Az egyes jellemzőkinyerő eljárások előtt más és más előfeldolgozó eljárások a hasznosak. A zaj nehezíti a lokális gradiensek kiszámítását valamint megbízhatatlanná teszi őket, ezen zajszűrőkkel segíthetünk. Ha a kontraszt nem elég nagy, az hátráltatja a gradiens számolást. Ezt például lokális hisztogram kiegyenlítéssel orvosolhatjuk.

2.2. Scale-Invariant Feature Transform (SIFT)

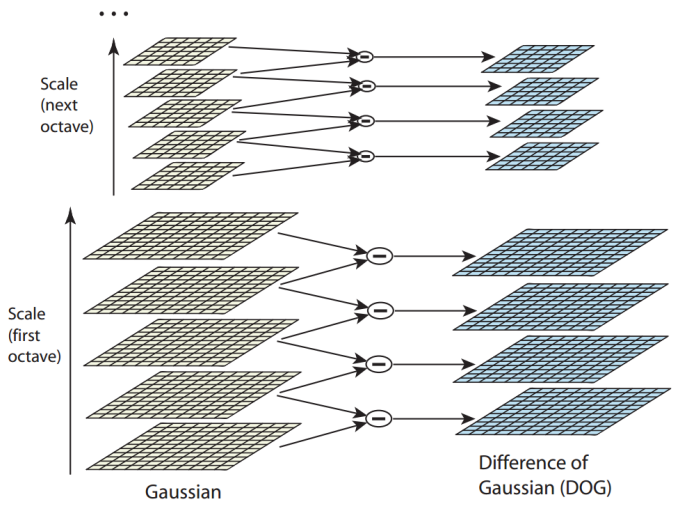
A SIFT-et [3] arra találták ki, hogy képekből olyan különböző, jellemzőket vonjon ki, amelyek megbízható párosítást tesznek lehetővé egy adott objektum különböző látószögből készült képei között. A jellemzők invariánsak a skálázásra és elforgatásra, emellett robosztus párosítást tesznek lehetővé különböző affin transzformációk, 3D-s látószögváltozás, zaj és fényviszony-megváltozás esetén. A következő lépésekben történik a jellemzők előállítása:

1. Extrém pontok keresése a skálatérben
2. Kulcspontok lokalizálása
3. Orientáció kijelölése
4. Kulcspontok leírása

Az elnevezése onnan fakad, hogy a képi adatot a helyi jellemzőkkel kapcsolatos skála-invariáns koordinátákká transzformálja. Képpárosításkor és felismeréskor a képekhez tartozó SIFT leírókat egy adatbázisban tároljuk. A képek jellemzőit ezután egyesével hasonlítjuk össze az adatbázisunk elemeivel, például a jellemzővektorok Euklédeszi távolságát vizsgálva. Mivel a kulcspontok eléggé különbözőek, ezért egy jellemző jó eséllyel megtalálja a párját az adatbázisban. Sajnos nem csak a keresett objektumot leíró jellemzőket kapjuk meg, hanem a háttér jellemzőit is, ami téves találatokat is eredményezhet. A jó találatokat úgy választhatjuk ki, hogy olyan kulcspont-halmazokat keresünk amik egyetértenek az objektum helyében, a nagyságában és az orientációjában is. Ezeket figyelembevéve már elég jó eredményt kaphatunk.

Extrém pontok keresése a skálatérben

Az extrém pont olyan szélsőértékkel rendelkező pont, amely kicsinyítésnél és nagyításnál is szélsőértékkel is rendelkezik. A kulcspont detekció első lépéseként olyan helyeket és skálákat keresünk, amelyek ugyanahhoz az objektumhoz rendelhető eltérő nézetekben. A skálainvariáns pontok keresése úgy végezhető, hogy stabil jellemzőket keresünk minden lehetséges skálán a skálatér folytonos függvényét használva. Megmutatták [5], hogy a skálatér egyetlen lehetséges kernelfüggvénye a Gauss függvény, így a skálatere egy képnek a következőképpen definiálható. A 3. ábrán láthatjuk, hogy a ská-



3. ábra. A skálatér és műveletei [3]

latér minden oktávjához az eredeti képet sorozatosan konvolváljuk a Gaussi szűrővel, hogy az ábra bal oldalán lévő skálatérbeli képeket megkapjuk. Egy kép skálatérbeli reprezentációja egy $L(x, y, \sigma)$ függvényként megadható, ami a Gaussi szűrő, $G(x, y, \sigma)$ és a kép, $I(x, y)$ konvolúciója.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

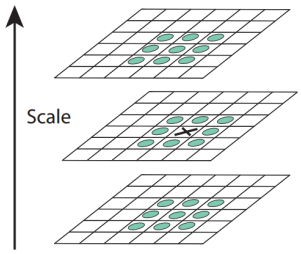
ahol $*$ a konvolúciós operátor x-en és y-on, és

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

A hatékonyabb detektálás érdekében a Gaussi függvények különbségét vesszük és konvolváljuk a képpel, ami két, egy k konstans szorzóval elválasztott skála különbsége:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

Ezután a szomszédosakat kivonjuk egymásból, hogy a jobb oldali különbségi Gauss képeket megkapjuk. Minden oktáv után a Gaussi képeket fele méretűre skálázzuk és megismételjük a kivonást. A 4. ábrán látható, hogy a különbségi Gauss képeken az extrém pontokat (maximum és minimum) az



4. ábra. Az extrém pont kiválasztása [3]

egyes pixelek (X) szomszédaikkal (O) történő összehasonlításával keressük meg. Csak akkor kerül egy pixel kiválasztásra, ha minden szomszédjánál nagyobb vagy minden szomszédjánál kisebb.

Kulcspont lokalizáció

Miután kiválasztottuk a jelöltjeinket (az extrém pontokat) a szomszédaikkal való összehasonlítás után, a következő lépés a közelben lévő pontok megvizsgálása elhelyezkedés, skála valamint elsődleges deriváltak szempontjából. Így például az alacsony kontrasztú pontokat visszautasíthatjuk, mivel ezek jobban érzékenyek a zajra. A metódusban egy 3 dimenziós négyzetes függvényt illesztünk a pontokra, hogy meghatározzuk a maximum helyét. Ebben a megközelítésben a Taylor-sorba fejtett alakját használjuk a skálatér függvénynek, amelyet úgy tolunk el, hogy a középpont a mintapontba kerüljön:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

ahol D és a deriváltjai a mintapontban értékelődnek ki és $\mathbf{x} = (x, y, \sigma)^T$ az eltolás ettől a ponttól. Az $\hat{\mathbf{x}}$ szélsőérték helyét úgy kapjuk meg, ha deriváljuk ezt a függvényt x szerint majd egyenlővé tesszük 0-val:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1}\frac{\partial D}{\partial \mathbf{x}}$$

Ha a függvény értékét a szélsőértékhelyeken vizsgáljuk, akkor visszautasíthatjuk az instabil szélsőértékhelyeket, melyek kontrasztja alacsony. Visszahelyettesítve:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T\hat{\mathbf{x}}$$

A visszautasítási küszöb értéke egy paraméter, amely minden szélsőértéket visszautasít ha az $|D(\hat{\mathbf{x}})|$ értéke kisebb nála, ennek ajánlott értéke 0,03, ha az intenzitásértékeket a [0,1] intervallumban adjuk meg.

Orientáció kijelölés

Amennyiben mindegyik kulcsponthoz hozzárendelünk egy egyértelmű orientációt a kép helyi tulajdonságai alapján, akkor a leírót ennek megfelelően tudjuk reprezentálni, amivel elforgatás-invarianciát érhetünk el. A kulcspont körüli régió egyes irányokba eső gradienseit kiszámolva kiválaszthatjuk azt az egy orientációt, ami a kulcspont orientációja lesz. Ehhez a kulcspont körül lévő pontok lokális orientációját számoljuk ki:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

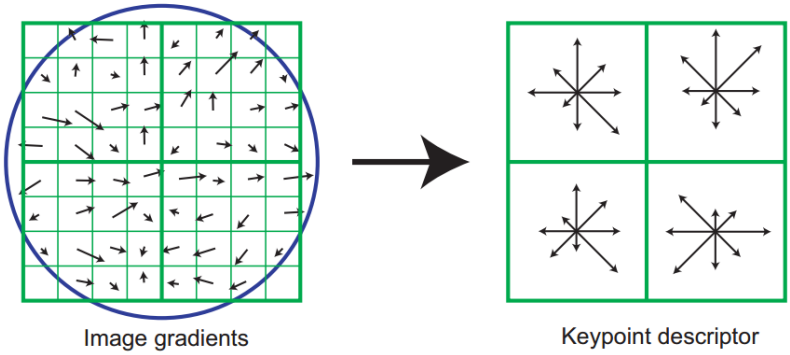
$$\theta(x,y) = \tan^{-1}(((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

A kulcspont orientációja a körülötte lévő pontok gradienseinek az orientációjainak a súlyozott összege. A súly a gradiens nagyságából és a kulcspont középpontú Gaussi körablak adott pontbeli értékéből tevődik össze. Ezután a legnagyobb nagyságó orientációval rendelkező pontot kiválasztjuk és a többi közül csak azokat tartjuk meg melyek a legnagyobb 80%-át elérik.

Lokális képleírás

Az előző lépésekkel pozíciót és orientációt rendeltünk minden kulcsponthoz. Ezek a paraméterek egy 2D koordinátarendszerben írják le a képet és ezekre a paraméterekre invariánsak. A következő lépés egy robosztus leíró megalkotása.

Az 5. ábrán látható a kulcspontleíró számítása. A leíró az orientációkat tartalmazó vektor. Itt 8 féle orientációt különböztetünk meg. Az ábra egy 2x2-es orientáció mátrixot ábrázol a kulcspont körül, de a gyakorlatban 4x4-es a legelterjedtebb. Ez azt jelenti, hogy $4 * 4 * 8 = 128$ elemű jellemzővektorokkal írjuk le a kulcspontokat, hisz minden mátrix térrészben levő 8 orientációs irányba eső vektorok hosszait jegyezzük fel. A megvilágítás változásának hatásait úgy csökkentjük, hogy egységnyi hosszra normalizáljuk. Így a kontrasztbeli változás (a pixelértékek egy konstans szorzót kapnak), ami megnöveli a gradienseket, eltörlődik a normalizációval. A megvilágításbeli



5. ábra. A kulcspont leíró készítése [3]

változás (a pixelértékekhez egy konstanst adunk) nem hat a gradiensértékekre, mivel a pixelértékek különbségét vesszük. Két paraméterünk van, amelyekkel a leíró komplexitását tudjuk szabályozni. Az orientációk száma, r és az orientáció mátrixok száma, $n \times n$. A leíróvektor mérete így rn^2. A komlexitás növelésével jobb eredményeket érhetünk el nagyobb adathalmazokon, de érzékenyebb lesz a torzításokra.

Gyakorlati alkalmazása

Kulcspont párosítás Minden egyes kulcspont párját a legközelebbi szomszéd módszerrel keressük az adathalmazban. Ez a leíró vektorok Euklédeszi távolságával tehető meg. A globális távolságküszöb sajnos nem alkalmazható, mivel egyes leírók jóval különbözőbbek mint mások, ezért másik módszerre van szükség, melyet a következő fejezetben mutatok meg a saját megoldásomnál. Sajnos a legközelebbi szomszéd keresére nagy dimenziós térben nincs pontosabb algoritmus mint a kimerítő keresés, amely lassú ugyan, de van néhány olyan adatstruktúra, amelyben gyorsabban is történhet a keresés, ilyen például a k-dimenziós fa is. Sok algoritmus van amely nagy gyorsítást nyújt az általunk használt 128 dimenziós vektor esetén.

k-dimenziós fák A k-dimenziós fa véges számú, k dimenziós pontok tárolására alkalmas adatstruktúra, melyet először Bentley javasolt [6]. A fa gyökérelemétől kiindulva minden csúcs egy elválasztósíkot reprezentál, minden ilyen elválasztás a csúcs mélységéhez tartozó koordináta szerint történik. Például a gyökérelem az első koordináta-tengelyre merőleges síkkal, a máso-

dik a második koordináta-tengelyre merőleges síkkal vágja két részre a teret. Az elválasztósíkok helyét a ponthalmaznak az adott részfába tartozó pontjainak a megfelelő koordinátára számolt mediánja adja. A pontokat úgy helyezzük a fába, hogy minden egyes csúcsnál ellenőrizzük az adott koordinátáját a pontnak, ha ez nagyobb mint az elválasztóérték, akkor a jobb oldali részfába, ha kisebb a bal oldali részfába kerül. Ha a fának több szintje lenne, mint a pontok dimenziója, akkor egyszerűen kezdjük elölről. Ezzel a módszerrel az a többdimenziós teret részekre daraboljuk elválasztóértékek szerint. Az elválasztóértékek megválasztása biztosítja, hogy kiegyensúlyozott fát kapjuk eredményül.

Legközelebbi szomszéd keresése A fának azon csúcsait szeretnénk megkeresni, amelyek a legközelebb vannak egy adott ponthoz. Ez a keresés hatékonyan megtehető a fa felépítésének tulajdonságait kihasználva. A következő lépéseken kell végigmenni:

1. Az új pont hozzáadásához hasonlóan végigmegyünk a csúcsokon az egyes dimenzió értékek szerint, amíg egy levélhez nem érünk. Ezt a levelet, mint "eddigi legjobb" elmentjük.

2. Minden csúcsnál ellenőrizzük a távolságot a pontunkhoz és ha közelebb vagyunk, új legjobbunk van, a többiek visszacsúsznak. Ezután ellenőrizzük, hogy az elválasztó hipersík másik oldalán lehet-e még közelebbi csúcs, ami egy egyszerű hipersíktól való távolság számolással megtehető.

3. Ha megvan a lehetősége a közelebbi pontoknak a hipersík másik oldalán, akkor átmegyünk a másik ágra és ott folytatjuk a keresést.

4. Ha nincs esély közelebbi pontokra, akkor folytatjuk a feljebb lépést és figyelmen kívül hagyjuk a másik ágat.

5. Az algoritmust a fa kezdőcsúcsánál fejezzük be, ekkor teljes a keresés.

A legközelebbi szomszéd keresése $O(\log n)$ komplexitású probléma k dimenziós fák esetén, ahol n a pontok száma.

2.3. Klaszterezés

Klaszterezésen egy adathalmaz pontjainak hasonlóságon alapuló csoportosítását értjük, amely az adatbányászat egyik legrégebbi területe. A klaszterezés célja az adathalmaz elemeit csoportokba sorolni úgy, hogy az azonos csoportba tartozó elemek lehetőség szerint minél hasonlóbbak, a különböző csoportokba esők pedig minél inkább eltérőek legyenek [7]. A klaszterezés úgynevezett felügyelet nélküli (unsupervised) tanulás, az osztályozással ellentétben, amely a felügyelt (supervised) tanuláshoz tartozik. Az osztályozás és a klaszterezés között az alapvető különbség az, hogy míg az előbbi esetében az elemeket előre rögzített csoportokba kell besorolni, addig az utóbbinál a csoportok meghatározása is része az eljárásnak [8].
Ha egy adott képhalmazban kell párokat keresnünk, akkor jogosan felvetődik a gondolat, hogy esetleg több kép is ábrázolhatja ugyanazt. Az egyes képfajták ekkor egy csoportot alkotnak. Ez a csoportosítás sokféle logika mentén történhet, szereplő tárgyak, emberek és állatok szerint is. Azt szeretnénk, hogy az egy csoportba tartozó képekre magas legyen a hasonlóság, míg a különböző csoportba tartozó képekre alacsony, ehhez szükségünk van egy eszközre amely felfedi a hasonlósági függvényünk hibáit és kijavítja azokat.

2.3.1. Gráfelméleti leírás

Most áttérünk egy gráfelméleti leírásra, amelyben ez a probléma vizsgálható. A képhalmazunkra, mint egy gráfra (G) tekinthetünk, melynek csúcsai (nódusai) a képek és közöttük súlyozott élek ($w \in [0, 1]$) futnak. Az A és B csúcs közötti élt w_{AB}-vel jelöljük ($w_{AB} = w_{BA}$). Ennek értéke az általunk kiszámolt $h(A, B)$ hasonlósági érték. Ezekre a hasonlósági értékekre a következő feltételek teljesülnek:

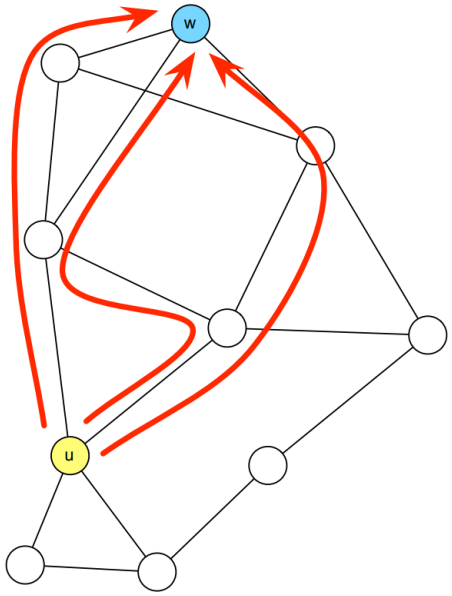
1. $\forall x : h(x, x) = 1$
2. $\forall x \forall y : h(x, y) = h(y, x)$

A gráf élek súlyait egy úgynevezett hasonlósági mátrixban tároljuk el, a már bevezetett jelölések szerint. Tehát ha A az a. kép és B a b., akkor a mátrix a. sorának b. oszlopában w_{AB} található. A feltételeknek megfelelően a kapott mátrix szimmetrikus lesz.

A k hosszúságú utak száma Legyen A a G gráf összekapcsoltsági mátrixa. G egyszerű és súlyozatlan n csúcsú gráf. Az $A^k(i,j)$-edik eleme (jelöljük $(A^k)_{ij}$-vel) az olyan k hosszúságú utak száma, melyik i-ben kezdődnek és j-ben végződnek [9]. Ezt indukcióval bizonyítjuk. $k = 1$ esetén, A éppen az 1 hosszú utakat adja meg. $k > 1$ esetén :

$$(A^k)_{ij} = (A^{k-1}A)_{ij} = \sum_{r=1}^{n} A_{ir}^{k-1} A_{rj}$$

A 6. ábrán az összes 3 hosszúságú utat láthatjuk két csomópont között.



6. ábra. Az u csúcsból w csúcsba vezető 3 hosszúságú utak

A k utas klaszterezés Legyen u és v egy gráf két csúcsa. Ekkor általánosságban elmondható, hogy a k hosszúságú utak száma nagyobb, ha u és v egy klaszterhez tartoznak és kisebb ha különbözőhöz. Az összekapcsoltsági mátrixból megkapott k hosszúságú utak számát felhasználhatjuk klaszterezésre, hiszen következtethetünk belőle a gráf srtuktúrális felépítésére. Legyen $Z_k(u,v) = (A^k)_{uv}$ hasonlósági mátrix, ahol k paraméter. Adott

k valamilyen értéke mellett egy $Z_k(u, v)$, tekintsünk rá hasonlósági mátrixként és végezzünk rajta single-link klaszterezést. A single-link klaszterezés a mátrix minden olyan elemét kidobja, amely egy adott t küszöb alatt van, az összekapcsolt komponenseit adja vissza a maradék éleknek.

Véletlen utazás **Súlyozatlan gráfban** egy csomópontból kiindulva egyenletes eloszlással választunk egyet a csúcs élei közül és átmegyünk rajta egy szomszédos csúcsig, majd az új csomópont élei közül választunk egyet és átmegyünk rajta, ezt ismételjük. **Súlyozott gráfban** nem egyenletes eloszlással, hanem adott valószínűséggel választunk új élt. Legyen a kiindulócsomópont u, éleinek súlya $w_1, \ldots, w_n$, ekkor az i. él választásának a valószínűsége:

$$P(i) = \frac{w_i}{\sum_{j=1}^{n} w_j}$$

Összekapcsoltsági mátrix esetén normalizáljuk az oszlopait, hogy az összegük 1 legyen az előző képlet szerint. Nevezzük ezt a mátrixot T_G-nek. Induljunk el egy u csúcsból. Mi a valószínűsége annak, hogy w-ben leszünk 3 lépés múlva? Legyen v_u oszlopvektor mindenhol 0 kivéve az u. helyen, ahol 1. A 0. lépésben $v_u[w]$ megadja a w-ben tartózkodás valószínűségét. Az első lépés után $(T_G v_u)[w]$ adja meg ezt. A k. lépés után, annak a valószínűsége, hogy w-ben vagyunk $(T_G^k v_u)[w]$. Más szavakkal a $(T_G^k v_u)[w]$ vektor minden csúcsra megadja nekünk annak a valószínűségét, hogy u-ból kiindulva k lépés után ott vagyunk [9].

Véletlen utazások használata klaszterezésre Mivel mi nem szeretnénk kiindulópontot választani, ezért v_u helyett használhatnánk egy olyan v vektort melynek minden eleme $1/n$, vagyis minden csúcsból egyforma valószínűséggel indulunk. Klaszterezési szempontból ez csak egy konstans szorzó így elhagyhatjuk és így csak $T_G^k = T^k$ mátrixal kell foglalkoznunk. $T^k[i, j]$ megadja annak a valószínűségét, hogy az i. csúcsból kiindulva k lépést megtéve a j. csúcsba érünk. Ha i és j ugyanabban a sűrűn összekapcsolt régióban vannak, akkor ezt az értéket nagynak sejtük. A gyakorlatban sajnos nagyon gyorsan szétoszlik a valószínűség a csomópontok között.

Markov láncok A véletlen utazás egy véges Markov láncnak tekinthető, amely időben reverzibilis. A Markov lánc olyan sorozata az $X_1, X_2, X_3, \ldots$ változóknak (a mi esetünkben a valószínűségi mátrixok), ahol a jelenlegi állapot adott és a jövőbeli állapotok függetlenek a múltbeli állapotoktól. A valószínűségek a következő időlépésben, csak a jelenlegi valószínűségektől függenek. A véletlen utazások és a véges Markov láncok elméletében nincs nagy különbség. Minden Markov lánc tekinthető egy véletlen utazásnak egy súlyozott, irányított gráfon. Ahhoz, hogy általánosságban definiálni tudjuk a véletlen utazást a súlyozott gráfokon, meg kell változtatnunk a súlyfüggvényt úgy, hogy a kimenő élek súlyainak összege 1 legyen, tehát egy valószínűségi mátrixot kell létrehoznunk [9].

Markov mátrix Legyen G egy n csúcsú gráf és legyen M az összekapcsoltsági mátrixa. A G-hez tartozó Markov mátrix, melyet T_G-vel jelölünk, úgy képződik, hogy minden i-re az i-edik oszlopa megegyezik M i-edik oszlopának normalizáltjával. Ennek kiszámításához definiáljunk egy d diagonális mátrixot, melynek főátlójában az adott oszlophoz tartozó M-beli súlyok összege szerepel, vagyis

$$d_{kk} = \sum_i M_{ik}$$

ekkor T_G a következőképpen definiálható

$$T_G = M_G d^{-1}$$

A T_G Markov mátrix egy G' gráf összekapcsoltsági mátrixának felel meg, amit G asszociált Markov gráfjának hívunk. G' irányított súlyfüggvényét, amit T_G tartalmaz, G lokálisan kiértékelt súlyfüggvényének nevezzük. Ez a transzformáció megyegyezik a véletlen utazásokhoz használt transzformációval. Könnyen belátható, hogy egy irányítatlan gráf esetén a kapott Markov mátrix már nem szimmetrikus.

A szorzó effektus növelése A k utas klaszterezés esetén állandósult állapotban a valószínűségek szétoszlanak a csúcsok között, pedig a kezdeti fázisokban ott alakulnak ki nagyobb értékek, ahol a csúcsok ugyanabban a sűrűbb régióban vannak. A súlyfüggvény megválasztása miatt szükség van egy új lépésre. Az utazás könyebb a szorosan összecsatolt régiókban és nehezebb a ritkásan összekapcsolt régiók között, de ez a hatás hosszútávon eltűnik. Ezért szükséges egy olyan lépés, amely előnyben részesíti a közeli

szomszédokat a távoliakkal szemben. Ennek egy természetes módja minden oszlop minden elemét valamilyen $r > 0$ kitevővel hatványozzuk és újra normalizáljuk. Ezt inflációnak nevezzük.

Infláció Adott egy $M \in R^{k \times l}$ nemnegatív mátrix és egy valós nemnegatív r szám [9]. Az M mátrix minden oszlopának r kitevővel való hatványozásával kapott mátrix $\Gamma_r M$, ahol Γ_r az r kitevőjű inflációs operátor. Formálisan $\Gamma_r : R^{k \times l} \to R^{k \times l}$ definíciója:

$$(\Gamma_r M)_{pq} = \frac{(M_{pq})^r}{\sum_{i=1}^{k} (M_{iq})^r}$$

2.3.2. Markov Clustering Algorithm (MCL)

Az algoritmus rendkívül egyszerű lépésekből áll, vázát a mátrix hatványozásának és inflációjának ismétlése adja.

1. Markov gráf generálása az összekapcsoltsági gráfból
2. A kapott mátrixon a két műveletünket (hatványozás és infláció) ismételjük, amíg a mátrix idempotens nem lesz (négyzetre emelve önmagát eredményezi)
3. A végleges mátrixon ezután végrehajtunk egy klaszterezést

A k. iterációban 2 mátrixot számolunk ki T_{2k}-t és T_{2k+1}-et. T_{2k}-t az előző mátrix, T_{2k-1} e_k-adik hatványaként számoljuk ki. T_{2k+1}-et pedig T_{2k} Γ_r-el vett inflációjaként számoljuk ki.

Az algoritmus definíciója Az általános MCL algoritmust 2 paramétervektor határozza meg, az $e_{(i)}$ és $r_{(i)}$, ahol $e_i \in N, e_i > 1$ és $r_i \in R, r_i > 0$ [9]. Ha az algoritmus bemenete egy M valószínűségi mátrix, akkor a következő az MCL leírása

$$(M, e_{(i)}, r_{(i)}),$$

ami $T_{(i)}$ mátrixok végtelen sora, ahol $T_1 = M$, $T_{2i} = Exp_{e_i}(T_{2i-1})$ és $T_{2i+1} = \Gamma_{r_i}(T_{2i})$, ahol $i = 1, \ldots, \infty$.

Az MCL összefoglalása A Markov klaszterező algoritmus [9] (Markov Clustering Algorithm – MCL) egy gyors és skálázható felügyelet nélküli klaszterező algoritmus, ami egy gráfon belüli sztochasztikus áramlás-szimuláción alapszik. Ehhez a korábban ismertetett gráfklaszterezési paradigmákat használjuk fel, melyek szerint egy véletlen utazás alkalmával több élén is végigmegyünk egy klaszternek mielőtt elhagynánk. A véletlen utazást a súlyozott vagy súlyozatlan gráf összekapcsoltsági mátrixának normalizálásából kapott Markov mátrixon végeztük, egyszerű mátrixszorzás végrehajtásával, ezt expanziónak nevezzük. Mivel a valószínűségek nagy lépésszám esetén eloszlik az összes csúcs között, ezért szükség van egy lépésre amely megerősíti a klasztereket és előnyt ad a közelebbi csúcsoknak a távolibbakkal szemben. Ez a lépés az infláció, amely az egyes oszlopok hatványozását és normalizálását jelenti. Ez az erős kapcsolatokat megerősíti és a gyengéket tovább gyengíti. Az expanzió és infláció iterálását addig végezzük, amíg a mátrixunk idempotens nem lesz. Ekkor a mátrix már klaszterekre van bontva, melyek nehézségek nélkül kinyerhetőek belőle.

2.3.3. A klaszterezés pontosságának mérése

A klaszterezés jóságának megállapítása első ránézésre egyszerűnek tűnhet, de nem az. A jóság meghatározására több megoldás is született, melyek eltérő módon mérik a csoportosítás hatékonyságát. Ezek közül mutatom be a fontosabbakat a következőkben. Először vezessük be az elempárok besorolásának típusait:

- TN (true negative) : azon elempárok száma, amelyek két különböző osztályba tartoznak, és ennek megfelelően két különböző csoportba is kerültek besorolásra
- FP (false positive) : azon elempárok száma, amelyek két különböző osztályba tartoznak, mégis azonos csoportba kerültek besorolásra
- TP (true positive) : azon elempárok száma, amelyek azonos osztályba tartoznak, és ennek megfelelően azonos csoportba is kerültek besorolásra
- FN (false negative) : azon elempárok száma, amelyek azonos osztályba tartoznak, mégis két különböző csoportba kerültek besorolásra

Jelentse továbbá R a felidézést (recall) és P a pontosságot (precision), amelyeket az előbbi jelöléseket felhasználva az alábbiak szerint definiálhatunk:

$$R = \frac{TP}{TP + FN}, \quad \text{ill.} \quad P = \frac{TP}{TP + FP}$$

E két mennyiség függ egymástól, értékük egy adott algoritmus esetén csak egymás rovására javítható. A klaszterezés pontosságának ellenőrzésére gyakran használják az ún. F-mértéket [10], annak is legtöbbször a kiegyensúlyozott változatát ($\alpha = 0{,}5$) :

$$F = \frac{1}{\frac{\alpha}{P} + \frac{1-\alpha}{R}},$$

ahol az α paraméter a felidézés és a pontosság súlyát határozza meg. Szintén gyakran alkalmazott mérték az ún. accuracy, amely a következőképpen számítható ki:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

További lehetőséget ad a pontosság mérésére az ún. tisztaság (purity) mérték, valamint a csoportok átlagos entrópiájának meghatározása, de az előbbi csak akkor használható, ha a csoportok száma adott [10]. Emellett sajnos ezen módszerek egyike sem veszi figyelembe mindkét hibatípust (FP, FN), ezért általában az F-mértéket és az accuracy értéket alkalmazzák.

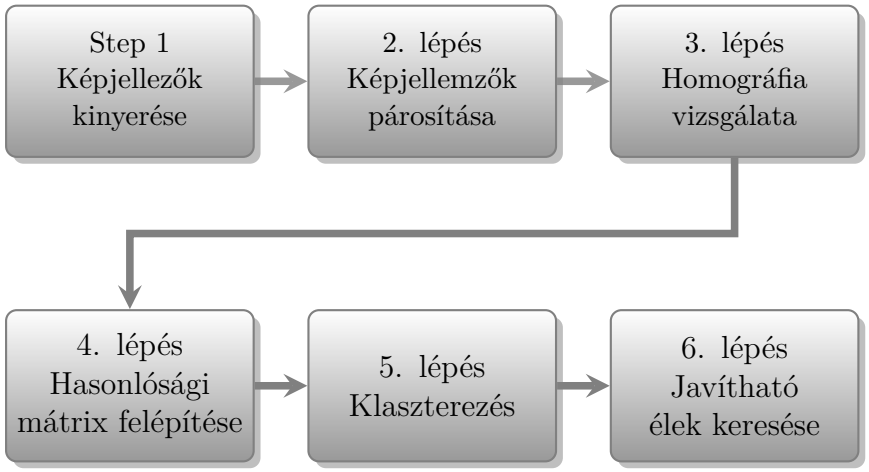
2.4. Hasonló problémák a szakirodalomban

A párkeresés, mint feladat minden tudományos területen jelen van, hiszen a párba állított elemekről hasonló kijelentéseket tehetünk, hasonló szabályokat alkalmazhatunk rájuk. A biológiában hasonló fehérje és DNS szekvenciákat keresnek nagy adatbázisokban, hogy a hasonlóságok alapján további következtetéseket vonhassanak le. Az informatikában is óriási szerepet tölt be pár/csoportkeresés, hiszen a felhasználókat minél pontosabban kell besorolni, hogy a megfelelő szolgáltatásokat/kedvezményeket lehessen a figyelmükbe ajánlani. A számítási kapacitás növekedtével a képfeldolgozásban is óriási adathalmazokkal kezdtek dolgozni, így megjelent az igény az automatikus rendezésre, rendszerezésre és felismerésre. A feldolgozás adattömörítéssel kezdődik, ez történhet valamilyen jellemzőkinyerő módszerrel [3, 4, 12] vagy egy előre betanított neurális hálózat által [13, 14, 15]. A jellemzőkinyerés

előnye a kisebb számítási kapacitás valamint nem szükséges hozzá egy nagy felcímkézett adathalmaz, amin hosszú hetek alatt betanítható. Ebben a témában hasonló megoldást alkalmaztak [16, 17] sikeresen, viszont a hasonlósági mátrix klaszterezése ezekben nem merült fel. Máshol már megjelent a klaszterezés gondolata is [18] A klaszterezés elterjedését késleltette a legtöbb algoritmushoz szükséges nagy számítási igény, szerencsére most már több, nagy adathalmaz esetén is jól skálázható megoldás van [9]. Az egyes megoldások előnyeit kihasználva próbáltam meg egy gyors és precíz rendszert alkotni, amely képes a hasonlósági értékek alapján összetartozó csoportokat meghatározni.

3. Intelligens módszer megalkotása képpárok keresésére

A korábban ismertetett rendszert egy általános célú programozási nyelven, Python-ban valósítottam meg. A Python előnye a könnyen olvasható és írható kód viszont a legtöbb alkalmazásban a futási ideje többszöröse egy C++-beli megvalósításnak. Ennek oka, hogy a Python interpreteres nyelv, míg a C++ egy fordítóprogram segítségével hoz létre optimálisabb programot. A választásom mégis teljesen logikus, ha képfeldolgozást szeretnénk megvalósítani. A jelenleg legelterjedtebb függvénykönyvtár képfeldolgozásra az Open Source Computer Vision (röviden OpenCV) [19]. Ez egy C++-ban megírt függvénykönyvtár, amely a számítógépes látás minden témaköréhez nyújt megoldást. Több különböző nyelven is elérhető, használható C, C++, Python, Java, MATLAB és Octave környezetben is. Legnagyobb előnye, hogy ezek a kezelőnyelvek csak külső burokként funkcionálnak az eredeti C++ kód körül. Ez biztosítja, hogy minden platformon ugyanolyan gyorsan fut. Tehát a Python választásával a két nyelv legjobb tulajdonságait használjuk ki, mivel a programozása egyszerű és tiszta, valamint a futása is közel ugyanolyan gyors mint ha az egészet C++-ban írtuk volna meg. Az általam kidolgozott megoldás a 7. ábrán látható lépéseken megy végig.



7. ábra. A megoldásom lépései

3.1. Nemzetközi képfeldolgozási verseny képpárosításra

Mint említettem egy nemzetközi képfeldolgozási versenyen [20] teszteltem a megoldásomat, amely biztosított egy adathalmazt a megalkotott módszerem kiértékelésére. A feladat képpárok automatikus keresése volt egy nagy címkézetlen adathalmazban. A képeken bálnák szerepeltek és meg kellett találni az ugyanahhoz az egyedhez tartozó képpárokat, ami azért lehetséges, mert a bálnák farokuszonyai egyedi mintázattal rendelkeznek. Az ugyanahhoz az egyedhez tartozó képek automatikus összepárosítása fontos lépés a későbbi biológiai analízis felé, mivel ezt a feladatot jelenleg emberek végzik fáradságos munkával. A rendszerek összehasonlításához 2005 képből álló címkézetlen adathalmazt biztosítottak. A képek a legkülönfélébb szögekből, megvilágítási viszonyok mellett készültek. Több képen is hullámok nehezítették a felismerést. Méretben is jelentős eltérések voltak 100x200-tól egészen 600x800-ig. Nem tartalmaztak semmilyen metaadatot így csak a puszta képpixelekre hagyatkozhattunk.

3.2. Jellemzőleírás

Az első lépés a képek tömörebb formában való leírása volt, melyet már az Irodalmi áttekintésben tárgyaltam. Az OpenCV minden ismert jellemzőleíró módszert tartalmaz, így először összehasonlítottam az egyes módszerek sebességét és pontosságát a gyakorlatban, hogy igazoljam az általam választott algoritmus előnyösségét. Az összehasonlított algoritmusok a SIFT, a Speeded-Up Robust Features (SURF), a Binary Robust Independent Elementary Features (BRIEF) és az Oriented fast Rotated BRIEF (ORB) voltak. A SURF, BRIEF és ORB mind sebességre optimalizált megoldások, melyek kevésbé jó jellemzőket adnak mint a SIFT, viszont futási időben kedvezőbbek.

SIFT	$\sim$ 200 ms/kép
SURF	$\sim$ 40 ms/kép
BRIEF	$\sim$ 10 ms/kép
ORB	$\sim$ 10 ms/kép

2. táblázat. Jellemző futási idők

A 2. táblázaton az egy kép jellemzőinek kinyeréséhez szükséges átlagos idők összehasonlítását látjuk. Szembetűnő a SIFT jellemzők kinyerésének

lassúsága a másik kettőhöz képest, viszont nem csak a sebesség az egyetlen szempont a kiválasztásra. Az említett összehasonlító cikk szerint a SIFT adja a legjobb jellemzőket a legtöbb esetben. A feladatomban a SIFT leírók mellett döntöttem, mivel a leírókat elég egyszer kiszámolni minden képre, így a jellemzőkinyerési időveszteség nem annyira jelentős a leírók minőségével szemben. A feladatomban szereplő ~ 2000 képre ezek 400 s, 80 s és 20 s alatt futnának le, ami nem számottevő része a teljes rendszer lefutásának.

3.3. Jellemzőpárosítás

A SIFT minden kép esetén több száz kulcspontot talál, ez néhány száztól több ezerig is terjedhet a képtől függően. Ezek mindegyikét egy 128 elemű vektorral írjuk le, tehát minden képhez átlagosan 300 darab 128 elemű vektor tartozik. Minden kulcspontot tehát egy ponttal reprezentálunk egy 128 dimenziós térben. Amikor két képet összehasonlítunk a kulcspontokat állítjuk párba, mivel feltételezzük, hogy a párbaállított kulcspontok ugyanazt a képi mintázatot reprezentálják a különböző képeken. Tehát egymáshoz közeli pontokat keresünk a 128 dimenziós térben. Algoritmusnak a k-dimenziós fákat használó algoritmust használtam, melynek az elméletét már korábban kifejtettem. Már bevezettem, hogy nem csak a legközelebbi pontokat keressük meg hanem elvárjuk, hogy egy kulcspont párjának nemcsak a legközelebb, hanem szignifikánsan közelebb kell lennie, mint a második legközelebbi pontnak. Ezért a k-dimenziós fákban minden ponthoz megkeressük a hozzá legközelebb eső két pontot. Miután az egyik kép minden kulcspontjához meghatároztuk a másik azon két kulcspontját, melyek a legközelebb vannak hozzá, megvizsgáljuk a távolságaik arányát. A Lowe által bevezetett arány szerint, ha d_a a legközelebbi, d_b a második legközelebbi pont távolsága, akkor fogadjuk el a legközelebbi pontot párként, ha $d_a < 0.7d_b$. Ha ez nem teljesül akkor feltételezzük, hogy a kulcspont leírónk nem elég egyedi, mivel több másik leíróra is hasonlít. Ezzel a módszerrel kulcspontpárok egy halmazát kapjuk, melyből további következtetéseket vonhatunk le. Fontos, hogy meghatároztam egy minimális kulcspontpár számot, ami megadja, hogy legalább hány közös leíróra van szükség a két képen ahhoz, hogy nagyon hasonlónak (pár jelöltnek) tekintsük őket.

3.4. Homográfia

Ahhoz, hogy még többet megtudjunk a hasonlóságról, az ötletem az volt hogy nem csak a leírók hasonlóságát és számát vegyük figyelembe, hanem a transzformációt is amivel egymásba vihetőek. Az egymásba vihetőség mértéke segíthet abban, hogy kiszelektáljuk a hibás képpárosításokat és megerősítsük a jókat. A megadása egy H mátrixal történik, amit alkalmazva a kulcspontokra a leírójuk párjának kulcspontjába mennek át. Ha ez a kulcspontok nagy részére egy maximális hibaszinten belül teljesül, akkor jó transzformáció és ez megerősíti a párbaállítást. Amennyiben rossz a transzformáció, akkor következtethetünk arra, hogy a talált leírópárok, csak véletlenül találhatóak meg mindkét képen, de nem ugyanaz az alakzat szerepel rajtuk, így nem is vihetőek egymásba semmilyen transzformációs mátrixal, ez segít kiszűrni a hibás párosításokat. A mátrix jóságának vizsgálata sokféleképpen történhet. A legegyszerűbb azt vizsgálni, hogy a kép sarkaival mit csinál. Ha a téglalapból valamilyen konkáv négyszöget csinál, akkor feltételezhetjük, hogy nem jó. Vizsgálhatjuk a kapott négyszög területét is, például ha nagyon lecsökken vagy megnő, akkor gyanakodhatunk. A homográfia kinyeréséhez a RANSAC [21] (Random Sample Consensus) algoritmust használtam. Ez az algoritmus kétfelé osztja a pontokat, belső (amikre jó a modellünk) és külső (amikre nem jó a modellünk) pontokra. Az algoritmus véletlen módon mintavételezi a pontokat és ez alapján készít modelleket (esetünkben egy transzformációs mátrixot). A végső modellt egy szavazás után kapjuk meg, minden pont az olyan modellekre szavaz amelyek illeszkednek rá egy bizonyos hibahatáron belül. Azzal a feltételezéssel élünk, hogy a rossz (hibás leírópárokhoz tartozó) pontok egyenlően szavaznak az egyes modellekre, így nem befolyásolják nagy mértékben az eredményt. A modellek készítését addig folytatjuk amíg nem lesz elég belső pont amikre illeszkedik a modell. A modell fontos paramétere, hogy mekkora az a pixel-eltérés amit megengedünk. Tehát a belső pontokra mennyire pontosan kell működnie a transzformációnak. Ennek az értékét 5 pixelre állítottam, mert megfigyeléseim szerint ez adta legjobb eredményt. Ellenőrzésképp a transzformációt alkalmaztam az egyik kép sarkaira és a kapott pontokat vizsgáltam. Ha a kapott négyszög területe egy alsó határnál kisebb vagy egy felső határnál nagyobb volt akkor elvetettem a képet (ez az alsó határ a célkép területének a fele volt, míg a felső a célkép területének a kétszerese), emellett megvizsgáltam a kapott négyszög konvexitását is (egy konkáv négyszög nyilvánvalóan nem lehet jó).

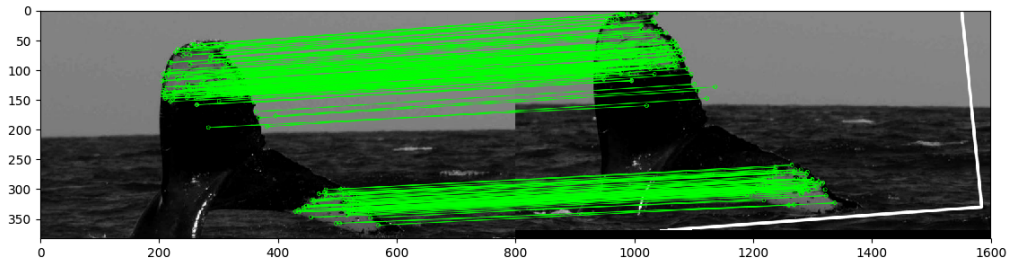
3.5. Saját teszt halmaz létrehozása és értékelése

A feladatom képfeldolgozási részének értékelésére szükségem volt egy olyan megbízható teszt halmazra amely elég sok képpárra tartalmazza, hogy azok párok-e vagy nem. Mivel a versenyben ilyet nem biztosítottak, ezért létrehoztam egy sajátot. Az eredeti képhalmaz ~ 2000 képből állt és sok különböző egyedet tartalmazott, így véletlen választás esetén viszonylag kicsi lett volna az esélyem valódi párok találására a majdnem 2 millió lehetőség közül. Szerencsémre rendelkezésemre állt egy lista egy tavalyi eredményről, amelyben jó eséllyel találhattam helyes párokat. Ennek a listának az első 1000 párját ellenőriztem kézzel. Erre egy külön programot írtam, amely végigmegy a lista képpárjainak általam megadott szakaszán, megjeleníti őket, majd az általam lenyomott billentyű szerint (0 vagy 1) menti az eredményt. Ez a teszthalmaz 251 valódi és 749 hamis párt tartalmazott. Ez kiválóan alkalmas volt az algoritmusom főbb tulajdonságainak vizsgálatára, mivel a valódi párokból is eleget tartalmazott.
Az értékeléshez az algoritmust az általam átnézett képpár halmazon futtattam le. A következő lépéseken ment végig az algoritmus:

- A képpár halmazban szereplő képek beolvasása és kulcspontjaik kinyerése
- A lista képpárjain végigmenve a kulcspontok összepárosítása a k-dimenziós fák által visszaadott két legközelebbi szomszéd és a Lowe arány vizsgálata alapján. A 20-nál kevesebb kulcspont párral rendelkező képpárok visszautasítása.
- A kulcspont párosítás által létrejött transzformáció leírása a RANSAC algoritmussal és vizsgálata. Az áttranszformált kép vizsgálata konvexitás és méret szempontjából. A nem megfelelő képek visszautasítása.

A 8. ábrán a végeredményre láthatunk egy példát. A módszerem megtalálja a fehér foltokat a farkon, melyek az egyediségét jelentik a bálnának és ezáltal legjobban jellemzik. A transzformált sarkok alkotta téglalap fehérrel van berajzolva, nem esik az egész a képre mivel a farok elfordult helyzetben van az első képhez képest.

Az ilyen képpárosító feladatoknál jól bevált módszer a 4 féle eredmény (true-positive, false-positive, false-negative, true-negative) arányának vizsgálata. Egy true-pozitiv pár az amely a valóságban és a programom szerint is



8. ábra. Példa egy jó párosításra

pár, egy false-negative pedig a programom szerint nem pár, de a valóságban igen. A nagy és ritka (kevés valós pár van az összes lehetőséghez képest) adatkészlet miatt kulcsfontosságú, hogy a false-pozitive párok száma minimális legyen. Összehasonlítást végeztem a homográfia vizsgálatával és anélkül, hogy a homográfia felhasználásának hatásáról is pontos képet kapjak. Ezt az 5. táblázatban láthatjuk. Először csak a leíró párok darabszáma alapján kerestem a párokat, a második esetben viszont a homográfiát is vizsgáltam.

Módszer	True-pozitive	False-pozitive	False-negative	True-negative
Homográfia nélkül	192	158	59	591
Homográfiával	182	38	69	711

3. táblázat. Eredmények az első ezer párra

Szembetűnő a false-pozitive párok számának jelentős csökkenése, ami a homográfia vizsgálatának köszönhető. Sajnos ennek hátrányai is vannak, mivel a true-pozitive párok száma is csökkent, igaz csak kis mértékben. Az így visszaadott párok recall (R), precision (P) és accuracy (ACC) értékei:

$$R = \frac{TP}{TP + FN} \quad , \quad P = \frac{TP}{TP + FP} \quad \text{ill.} \quad ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

ahol TP–true-positive és FP–false-pozitive. A recall 76,5%-ról 72,5%-re csökkent, viszont a precision 54,86%-ról 82,73%-ra nőtt, ami hatalmas javulás. Az accuracy 78,3%-ról 89,3%-ra nőtt a nem valós párok hatékonyabb kiszűrésének köszönhetően, a false-pozitive párok aránya 5%-ra csökkent 21,1%-ról. Ez nagy előny a nagy halmazra tekintve, ahol csak kevés valódi pár van. Érdemes azt is figyelembe venni, hogy a teszthalmaz nem véletlenszerű

képpárokból áll az adathalmazból, hanem már válogatott képekből. Tehát egy algoritmus már hasonlóként jelölte meg őket, ez még jobb színben tünteti fel a 5%-os false-positive arányt.

3.6. Klaszterezés

Miután a párosításra egy meglehetősen pontos algoritmust sikerült készítenem, megvizsgáltam a klaszterezés hatását. Ehhez létrehoztam egy gráfot a képhalmazból melyben az 1300 párt (itt már egy nagyobb halmazt használtam, amely 262 valós párt és 1038 nem párt tartalmazott), mint éleket tudtam felhasználni. Ez elhanyagolhtó rész a gráf összes éléhez képest, viszont nagyon sok új információt tudtam kinyerni belőle. Mint már bevezettem a gráfot egy hasonlósági mátrixként adtam meg melynek (i, j) és (j, i)-edik elemei az i. és j. kép közti hasonlósági értéket tartalmazzák. Az élek súlyozását a két kép leírópárjainak száma ($feature\_pairs$) alapján végeztem a következő szabály szerint:

$$w_{ij} = w_{ji} = \begin{cases} 0 & feature\_pairs < 20 \\ 0.1 + (feature\_pairs - 20) * 0.009 & 20 \leq feature\_pairs \leq 120 \\ 1.0 & 120 < feature\_pairs \end{cases}$$

20 közös leíró jelentett 10% hasonlóságot, míg 120 100%-ot (120 felett ugyanúgy 100%). Normalizálás után a (i, j) és (j, i)-edik elemek már különbözni fognak, mivel oszloponként 1-re normalizáljuk a hasonlóságokat, hogy minden csúcsra a kimenő élek összege 1 legyen. Ezzel a hasonlósági értékekből valószínűségeket kaptunk, amelyek az egyes csúcsokból a többi csúcsba való átjutás valószínűségét tartalmazzák véletlen utazás esetén. A gráfot először a kézzel válogatott párok által megadott élekből építettem fel, normalizáltam a hasonlósági mátrixot és meghívtam rá az MCL algoritmust, amely megadta a talált klasztereket. Mivel a gráf éleinek nagy részéről semmit nem tudtunk ezért kiindulási értéknek egy heurisztikus valószínűséget adtam meg. Az első 1000 között megtalált ~ 250 pár és a versenyen elért 0.25-ös average precision azt a következtetést sugallta, hogy nagyjából 1000 pár lehet az adathalmazban. Így adhatunk egy durva becslést egy véletlenszerűen választott pár esetén a valódiságának valószínűségére: $p = \frac{1000}{2000000} = 0.0005$.

A klaszterekből kétféle élre tudtam következtetni. Az egyik a rosszul vagy még nem azonosított (nem vizsgáltuk a két kép hasonlóságát) belső élek, ezek azok a belső élek, amelyekre a programom alacsony valószínűséget adott

vagy még nem adott rájuk valószínűséget. A klaszterezés ezzel szemben egy csoportba sorolta őket valószínűsítve, hogy ezek az élek párokat jelölnek. A másik a rosszul vagy még nem azonosított külső élek. Ezt érdemes külön vizsgálni a kézzel válogatott és a program által kiszámolt valószínűségű párokon. A kézzel válogatott párok által megadott klaszterek esetében, amennyiben két klaszter között megy egy biztosan 0 értékű él, vagyis az él egy biztosan nem párt jelöl, akkor két klaszer más egyedhez tartozik. Ennek értelmében minden köztük menő élet nullázhatunk, hiszen egyik sem jelenthet párt. Ezzel a módszerrel sok párjelölt kizárható, így kevés képpár kézi válogatásával is sok továbbira következtethetünk. Ez valamennyire megkönnyíti a teszthalmaz létrehozását. A program által visszaadott valószínűségek esetén már nehezebb a helyzet. Itt a hiányos halmaz miatt két klasztert akkor tekintünk különbözőnek, ha több él is van, melyek ezt valószínűsítik. A többségi döntés elvét alkalmazzuk, a két klaszter között levő élek legalább felét ismernünk kell és ezeknek legalább felének a különbözőségre kell szavazniuk. A 0.25%-os hasonlóságot jelöltem meg mint felső határt arra, hogy a különbözőségre szavaz egy él. Ha tehát van egy n és egy m csúcsból álló klaszterünk, akkor összesen $n \times m$ él fut a két klaszter között. Ezek élek legalább felét ismernünk kell és az ismertek legalább felének legfeljebb 25%-os hasonlósággal kell rendelkeznie. Ebben az esetben különbözőként jelöljük meg a klasztereket és következtethetünk a közöttük húzódó ismeretlen élekre.

Ezzel a módszerrel az algoritmus az eredeti teszthalmaz 1300 éléből még 570-re tudott következtetni. 553 új külső élt talált az eddigi 1038-hoz és 17 belső élet talált a meglévő 262-höz. Az új párok nagyrészt hármas klaszterek egy hiányzó élét pótolták, de olyanra is volt példa, hogy egy négyelemű klaszter 4 meglévő élét egészítették ki 6-ra. A klaszterezett pontok száma 470 volt, tehát ennyi egyednek volt valós párja a mintahalmazban. A rendszerem által kiszámolt hasonlóságokból felépített gráf esetén kevesebb új élt talált az algoritmus, mivel itt szigorúbb feltételeket kellett teljesíteni a klaszterek különbözőségének bizonyításához valamint a kapott hasonlósági értékek sem voltak olyan erősek, mint a kézzel válogatott esetben. Ezután megvizsgáltam a homográfia vizsgálata után kapott és a klaszterezés után kapott párok helyességét. Mivel a teszthalmazomat 1870-re bővítette a klaszterezés így az rendszerem által kapott hasonlósági értékekből képzett klaszterek és az algoritmusom által talált élek nagy részét pontosan tudtam értékelni. Azokat az éleket, melyek a teszthalmazomban nem szerepeltek, de a hasonlósági értékekből képzett gráf klaszterezése és az új élek keresése után megjelentek, hibásnak tekintettem, mert nem állhatott rendelkezésre elég információ,

hogy ténylegesen biztosan párnak vagy nem párnak jelölhessük őket.

Módszer	True-pozitive	False-pozitive	False-negativ	True-negativ
Homográfiával kapott párok	182	38	69	711
Klaszterezéssel kapott párok	194	52	66	960

4. táblázat. Eredmények a klaszterezés után

A 4. táblázatban látható a klaszterezés hatása. A True-pozitive párok száma 12-vel nőtt, 182-ről 194-re. Ez két hatásból tevődik össze: a kijavított rossz besorolásokból (3 db, mivel a false-negativ 3-mal csökkent) és a még nem vizsgált, de a klaszterezéssel megtalált új párokból (a maradék 9 db). Sajnos a false-pozitive párok száma is megnőtt a hibásan osztályozott párok klaszterezére miatt. Mivel ezek egyenlő arányban történtek ezért ebben az esetben elfogadható, de figyelnünk kell arra, hogy az ilyen élek erősségét ezzel a bizonytalansággal megsúlyozzuk. Látható, hogy a true-negativ élek száma jelentősen elmarad a program által adott párok esetén a kézzel készített mintahalmazhoz képest. Ez a gyengébb hasonlósági értékek és a miattuk bevezetett szigorúbb feltételeknek köszönhető. A kapott eredményekre kiszámoltam a már bevezetett mennyiségeket, melyek segítenek képet alkotni a klaszterezés pontosságáról.

$$R = \frac{TP}{TP + FN}, \qquad P = \frac{TP}{TP + FP} \quad \text{ill.} \quad ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

Módszer	recall (R)	precision (P)	accuracy (ACC)
Homográfiával kapott párok	72,5 %	82,7 %	89,3 %
Klaszterezett párok	74,6 %	78,8 %	90,7 %

5. táblázat. A klaszterezés recall (R), precision (P) és accuracy (ACC) értékei

Látható, hogy a klaszterezés a három minőségi mutatóból kettőben javította az eredményt, még úgy is, hogy a gráf összes éléhez képest elhanyagolható számú állt rendelkezésre. Ennek a hatása a megadott élszám növelésével javulni fog.

3.7. Futtatás a verseny adathalmazra

A hosszú futtatási idő és számítási kapacitás igény miatt nem a saját számítógépemen futtattam le a módszerem képfeldolgozási részét. Ehhez a Microsoft Azure felhőszolgáltatásban béreltem egy virtuális gépet, amelyen futtattam az algoritmust. A környezet felállítása az egyik legidőigényesebb és legproblémásabb feladat volt a munkám során. A működőképes gépre SSH-n keresztül csatlakoztam fel, az adathalmaz és a programok letöltése után elindíthattam a teljes futtattást. A minimális futási idő érdekében a lehető legkevesebb kiírást engedélyeztem. Mivel $\binom{2005}{2}$ összehasonlítást kellett végeznünk így a becsült futási idő 128 órásra adódott, végül kicsit több 130 óra alatt futott le. Mivel a teljes adathalmazról nem állt rendelkezésünkre semmilyen párosítás így a már bevezetett mérőszámokat nem állt módomban kiszámolni, de a verseny szervezői ezt hivatalosan megtették és értékelték az eredményeket. A versenyre bizonyossági sorrendet kellett felállítani a párok között és egy szöveges fájt kellett beadni, mely <imageX.jpg imageY.jpg score> formában tartalmazta a kapott párjaink valószínűségét. A párokat sorba kellett állítani a legvalószínűbbtől a legkevésbé valószínűig. A szervezők ezen egy átlagos precizitást (average precision) mértek, mely minden helyes párnál számol egy precizitást az addigi párok alapján, majd a végén ezt átlagolja:

Az eredmények a következők lettek:

Run	Avergage Precision
BME_DCLab_whalerun_3	0.51
ZenithINRIA_whalerun_SiftGeo	0.39
bmemit_run1	0.36
BME_DCLab_whalerun_2	0.30
BME_DCLab_whalerun_1	0.30
MLRG_whalerun_2.txt	0.01

9. ábra. A verseny eredménye

Minden csapat 3 futtatás eredményét tölthette fel, így 3 módszer eredményét értékelhette. Az én futtattásom a 'BME_DCLab_whalerun_3' volt és ez lett az idei évben a legjobb átlagos pontosságú, jóval jobb eredményt nyújtott, mint a második legjobb megoldás. A versenyre elkészült megoldásunkat egy konferencia cikkben is megírtuk 'Image matching for individual recognition with SIFT, RANSAC and MCL' címmel [22].

4. Összefoglalás

Tudományos Diákköri munkám keretében létrehoztam egy képfeldolgozó-rendszert, amely egy címkézetlen képhalmazban képes párokat/csoportokat kialakítani úgy, hogy az egy csoportban lévő képek hasonlítsanak egymásra. Az algoritmusnak több lépése van, először minden képből jellemzőket nyer ki a tömörebb reprezentáció érdekében, ezután párosával megvizsgálja a képeket és a két kép jellemzőit megpróbálja párba állítani a megadott korlátozó feltételeknek megfelelően. Ha elegendő mennyiségű jellemzőpárt talált akkor még egy homográfiai vizsgálatot is végez a kulcspontok elhelyezkedésének egymásbavihetőségéről. Miután minden képpára elvégeztük ezt a vizsgálatot létrehozunk egy súlyozott gráfot a leírópárok száma alapján számolt valószínűségek szerint és ebből a Markov klaszterező algoritmus segítségével klasztereket nyerünk ki, majd a klaszterekből a valószínűségek alapján a párokat. Az implementált párkereső algoritmust egy magam által címkézett részhalmazon vizsgáltam, a különböző részek után, melyek eredményét összevetettem. Ez alapján láttam, hogy a klaszterezés és a homográfia is javítja az eredményeket. Ezen kívül részt vettünk egy nemzetközi kihíváson, ahol egy sokkal nagyobb képhalmaz állt rendelkezésre, ezen az adathalmazon kapott eredményeket a verseny szervezői értékelték egy átlagos pontossági mutatóval. A versenyen a legjobb eredményt sikerült elérnem $0,51$ átlagos pontossággal. A verseny eredményéről írtunk egy tudományos publikációt, mely idén ősszel egy nemzetközi konferencia kiadványában jelent meg [22], továbbá egy folyóirat cikk is készült, mely jelenleg elbírálás alatt van. Az eredmények alapján a rendszer alkalmas egy ismeretlen képhalmaz szétválogatására, mivel nagy pontossággal képes hasonló képekből klasztereket kialakítani. Emellett érdekes lehet félautomata üzemmódban való használata, ahol célzott párok megjelenítésével (melyeket a hasonlóságuk alapján tehet meg) és valósidejű klaszterezésével jelentősen felgyorsíthatja a munkát így véve le a terhet a felhasználó válláról.

Hivatkozások

[1] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image pre-processing*, pages 56–111. Springer US, Boston, MA, 1993.

[2] Scott Krig. *Image Pre-Processing*, pages 39–83. Apress, Berkeley, CA, 2014.

[3] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[5] Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. 21:224–270, 09 1994.

[6] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, Sept 1979.

[7] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[8] Tikk Domokos. *Szövegbányászat*. TYPOTeX, 2007.

[9] Stijn Van Dongen. A cluster algorithm for graphs. *Report-Information systems*, (10):1–40, 2000.

[10] Magnus Rosell, Viggo Kann, and Jan-Eric Litton. Comparing comparisons: Document clustering evaluation using two manual classifications. In *ICON 2004, India.*, 2004.

[11] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[12] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. *BRIEF: Binary Robust Independent Elementary Features*, pages 778–792. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[13] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59, 2011.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[15] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR 2016 Workshop*, 2016.

[16] Alexis Joly, Jean-Christophe Lombardo, Julien Champ, and Anjara Saloma. Unsupervised Individual Whales Identification: Spot the Difference in the Ocean. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum*, pages 469–480, Evora, Portugal, September 2016.

[17] Dávid Papp, Dániel Lovas, and Gábor Szücs. Object detection, classification, tracking and individual recognition for sea images and videos. In *CLEF (Working Notes)*, pages 525–533, 2016.

[18] Symeon Papadopoulos, Christos Zigkolis, Giorgos Tolias, Yannis Kalantidis, Phivos Mylonas, Yiannis Kompatsiaris, and Athena Vakali. Image clustering through community detection on hybrid image similarity graphs. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2353–2356. IEEE, 2010.

[19] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015. [accessed 3-February-2017].

[20] SeaCLEF 2017. `http://www.imageclef.org/lifeclef/2017/sea`. [accessed 2-March-2017].

[21] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[22] Dávid Papp, Ferenc Mogyorósi, and G Szücs. Image matching for individual recognition with sift, ransac and mcl. *Working Notes of CLEF*, 2017, 2017.