# Jellemzőkinyerés

## SIFT

A David Lowe által publikált SIFT (Scale Invariant Feature Transform) lényege, hogy minden képből leírókat, úgynevezett kulcspontokat nyer ki, különböző matematikai módszerek segítségével. Az algoritmus olyan módon készíti el ezeket a leírókat, hogy a kép különböző torzításai (nagyítás-kicsinyítés, nyújtás, elforgatás) esetén is nagy eséllyel megtalálhatóak legyenek a módosított képen. Ezen kulcspontok mindegyike egy 128 dimenziós vektor, és egy kép akár több ezer ilyen vektort is tartalmazhat. A SIFT egy igen jó hatásfokú megoldás képi egyezések keresésére, ám számítási igénye nagy/bonyolult képek esetén nagy lehet, amit optimalizálásokkal lehet javítani. Megemlítendő még, hogy a SIFT algoritmus csak a leírók generálását, annak matematikai módszereit definiálja. Az azon túli, ezekre épülő módszerek, akár csak a leírók egyezésének vizsgálata, többféleképpen megoldhatóak, ezekre a szerző csak ajánlást tesz.

## SURF

A SURF (Speeded Up Robust Feature) egy új skála-invariáns, jellemző-kinyerő módszer képekre. Számítási igénye alacsonyabb a legtöbb módszernél, mégis nagy hatásfokkal működik. Ezt úgy éri el, hogy a képek integráltját használja fel a konvolúciós lépés során, építve az eddigi módszerekre és egyszerűsíti azokat. Alapötletét a SIFT szolgáltatta, hasonlóan sok másik algoritmushoz, azonban Haar-like leírókat használ a képek jellemzésére.

## GLOH

A GLOH (Gradient Location and Orientation Histogram) módszer szintén egy robosztus képleíró. Hasonló a SIFT-hez, azonban inkább a területekre, régiókra koncentrál a hisztogramok előállításánál. A leírók itt 64 dimenziósak.

# SIFT részletesen

A SIFT minőségét tekintve kiemelkedő módszer, az objektumok azonosítására is alkalmas leíróvektort állít elő. Hátránya, hogy nagyon erős előfeldolgozó, és ablakozó processzortömböket igényel.

A SIFT, mint módszer már jó néhány éve publikálásra került, ennek köszönhetően elég elterjedt, és sok helyen használt. A legújabb algoritmusok is ezen alapulnak, továbbfejlesztve az ötletet, így a SIFT a maga nemében mérföldkőnek számít. Választásunk főként a robusztussága, és a képtorzításokra való érzéketlensége miatt esett rá. A futási ideje nem a legjobb, de a prototípus programhoz bőven elegendő, későbbiekben lecserélhető, például az újabb SURF algoritmusra. Nem utolsó sorban, a SIFT-nek elérhető a szerző, Lowe által publikált C nyelvű programja is. Ezt a parancssoros programot használjuk fel a kulcspontok generálására, Python nyelvű és parancsokkal és megfelelő paraméterezéssel.

Fontos megjegyezni, hogy a SIFT nem tartalmazza a konkrét objektumok osztályokba sorolását. Ez azt jelenti, hogy a megtalált, két kép közötti egyezésről nem tudjuk, hogy valójában micsoda is, csak azt, hogy mindkét képen megtalálható. Konkrét objektumok (pl. cipő, labda, ház) felismeréséhez (osztályozásához) tanuló algoritmusok, tanító halmazok és módszerek szükségesek. A generált kulcspontok az elforgatott, torzított képen is megtalálhatóak, vagy egy másik képen lévő képrészleten is. További számítások, például az első lépésként elvégzendő kulcspont-egyezések vizsgálata már nem képezi a SIFT részét. A kulcspont-egyezéseket első lépésben az NNS (Nearest Neighbour Search) módszerrel célszerű vizsgálni, azonban a későbbiekben célszerűbb áttérni a hatékonyabb Best-bin-first algoritmusra.

A SIFT algoritmus térbeli Gauss szűrők (aluláteresztő, a képet simító szűrő) sorozatával, és a szűrt képek különbségének kiszámításával a bementeti kép olyan transzformáltjait hozza létre, melyeken a foltszerű képrészletek robusztusan lokalizálhatók. A szűrő σ=2 paraméter melletti futtatásával az effektív képfelbontás felére csökken, illetve a folyamatos skála térben 1 oktávval. Lowe javaslata szerint ahhoz, hogy optimális detekciót érjünk el, szükség van a kép dupla méretű reprezentációjára, és oktávonként három további finom felosztásra. A dedikált 2D celluláris processzortömbön hatékonyan kiszámítható a Gauss szűrőknek megfelelő diffúzió, de a tömb méretének a szenzorfelbontás kétszeresére történő emelése legalább négyszeres hardver komplexitást igényelne, míg az oktávonkénti finom felbontások lineáris növekedést jelentenek.

## Simítás

A transzformáció első lépésében Gauss-féle konvolúciós szűrővel hajtunk végre simítást. A simítást többször is elvégezzük, a kép méretét mindig felére csökkentve. Így egy képpiramist hozunk létre, mely a feldolgozás gyorsítását szolgálja. Ezt követően a szomszédos konvolúciós szintek különbségét képezzük, így egy különbség skála szintjei jönnek létre. Ennek leírására használjuk a D(x, y, ϭ) függvényt :

D(x, y, σ) = (G(x, y, kσ) − G(x, y, σ)) ∗ I(x, y) = L(x, y, kσ)−L(x, y, σ),

ahol ∗ a konvolúciós operátor, D(x, y, σ) a Gauss-féle konvolúciós maszk, I(x, y) a bemeneti kép, L a skálatér egy szintje, k pedig egy pozitív egész, mely a skálaszintet jelöli. σ az alkalmazott Gauss függvény szórását jelöli. Leegyszerűsítve D olyan különbség szinteket képez, melyben egyik szint éppen k-szor feljebb található a skála-térben, mint a kivonandó másik. Ezzel a lépéssel biztosítható, hogy hatékonyan találjunk „biztos” pontokat és megtartsuk a módszer „erejét” adó invarianciát.

## Szélsőértékek keresése

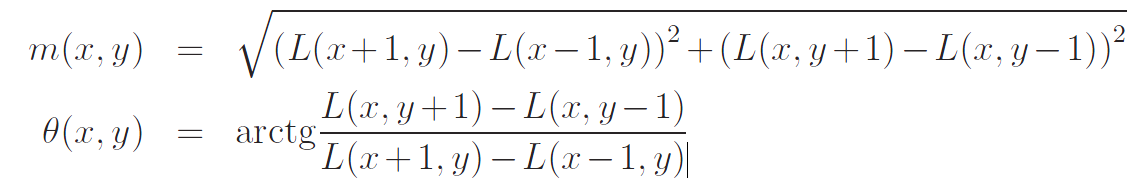
A következő lépésben szélsőértékeket kell keresni a D által létrehozott Gauss-különbség térben. A kulcspontok a lokális maximum, illetve minimum értékek lesznek, melyek az aktuális pixel saját szintjén elhelyezkedő nyolcas környezetével, illetve a szomszédos szinteken lévő 18 másik szomszédjával való összehasonlításból számítandóak ki. Ez így összesen huszonhat összehasonlítás. Ha ez a pont a többivel való összehasonlítások után minimális vagy maximális, akkor szélsőérték. Az így kapott pontokon további műveleteket kell végezni. Mivel többször is elmostuk a képeket és a méretüket is csökkentettük, ezért interpoláció segítségével a környező adatok alapján vissza kell keresnünk a kijelölt pontok eredeti helyét az eredeti képen.

## Kulcspontok számának csökkentése

Ha ez megvan, akkor a hatékonyság növelése érdekében csökkenteni kell a kulcspontok számát, ugyanis ezek közül még nem mind hordoz fontosnak mondható információt. Ez azért szükséges, hogy csak a stabil pontok maradjanak meg a további számításokhoz. Az alacsony kontrasztú pontokat és a gyenge élpontokat el kell távolítani egy küszöböléssel. Ezt megtehetjük könnyedén, ha kiszámítjuk a Laplace operátorral számított értékét az adott pontnak. Ha a kontraszt érték egy megadott küszöb alatt van, akkor kivesszük a pontot a kulcspont listából. A stabilitáshoz azonban nem elegendő a kis kontrasztú pontok kiszűrése. A Gauss-különbség függvény jól használható információkkal szolgál az élekről, azonban ha egy él elég gyenge, akkor érzékeny lesz a zajra. Ahhoz, hogy kiszűrjük a további gyenge pontokat, meg kell vizsgálnunk, hogy van-e fő görbület az éllel párhuzamosan, illetve gyenge görbület a merőleges irányban, D-ben az adott helyen. El kell vetni a pontot, ha ez a különbség a legnagyobb és legkisebb sajátvektor aránya alatt van, amit 2×2-es Hesse-féle mátrixból számíthatunk ki az adott helyre illetve skálaszintre nézve.

## Irányok meghatározása

A feldolgozás következő lépéseként a pontokhoz irányokat – akár többet is – kell rendelni, amelyeket a lokális gradiens jellemzők alapján határozunk meg. A gradiens m nagyságát és Φ irányát az alábbi módon határozhatjuk meg:



Ezt elvégezzük az adott kulcspont adott sugarú – általában 4 vagy 8 – környezetére is, majd ezeket az értékeket egy olyan σ paraméterű Gauss-maszk szerint súlyozzuk, amely rendszerint másfélszerese a kulcspont léptékének (lásd az alábbi). Ezt követően készítünk egy szöghisztogramot 36 vödörrel. Az így létrejött régiókat összegezzük 4×4-es felbontású részekre úgy, hogy közben csökkentjük a vödrök számát nyolcra. Nagyon fontos, hogy az egyes kulcspontokhoz tartozó hisztogramok a ponthoz előzőleg kiszámolt orientációhoz vannak igazítva.



. ábra. Szöghisztogram készítése.



. ábra. Irányított szöghisztogram.

A domináns irányok a hisztogram kiemelkedő értékei lesznek. Alapvetően a legnagyobb ilyen csúcsot tekintjük fő iránynak, de ha előfordul még olyan érték, mely a maximális érték 80%-án belül van, akkor létre kell hozni vele egy új kulcspontot ugyanazon a helyen.

Általánosságban a pontok 15%-a rendelkezik többirányú hozzárendeléssel, ezzel csak növelik az ilyen kulcspontok stabilitását. A leíró végül úgy jön létre, hogy minden kulcsponthoz és környezetéhez tartozik 4×4 darab hisztogram egyenként 8-8 értékkel. Ez összesen egy 128 elemű vektort eredményez. Ha a megvilágításból adódó változásokat szeretnénk kiküszöbölni, akkor normalizálnunk kell a vektort, így elérhetünk bizonyos fokú invarianciát e változásokkal szemben.