Gépi látás - GKLB\_INTM038

Contents

[Bevezetés, megoldandó feladat kifejtése (5%) 1](#_Toc105850203)

[Elméleti háttér (30%) 2](#_Toc105850204)

[Step#1 – Pozitív minta 2](#_Toc105850205)

[Step#2 – Negatív minta 2](#_Toc105850206)

[Step#3 – Pixelenkénti HOG jellemzők 2](#_Toc105850207)

[Step#4 – Cellánkénti HOG jellemzők, hisztogram 2](#_Toc105850208)

[Step#5 – Blokk normálás 3](#_Toc105850209)

[Step#6 – HOG jellemzők összegyűjtése és SVM modell tanítása 3](#_Toc105850210)

[Megvalósítási terv és kivitelezés (40%) 3](#_Toc105850211)

[Github repository 3](#_Toc105850212)

[Adatok, minták 4](#_Toc105850213)

[Alkalmazások 4](#_Toc105850214)

[Build instrukciók 4](#_Toc105850215)

[Alkalmazások leírása 4](#_Toc105850216)

[Teszt (20%) 6](#_Toc105850217)

[Tesztelési terv 6](#_Toc105850218)

[Teszteredmények 6](#_Toc105850219)

[Képek 6](#_Toc105850220)

[Vizualizált eredmények 6](#_Toc105850221)

[Videók 8](#_Toc105850222)

[Teszteredmények kiértékelése 8](#_Toc105850223)

[Felhasználói leírás (5%) 8](#_Toc105850224)

[References 8](#_Toc105850225)

# Bevezetés, megoldandó feladat kifejtése (5%)

A választott feladat típusát tekintve objektumdetektálásként sorolható be. A fő feladat olyan alkalmazás készítése, mely drónfelvétel alapján képes a felvételen előforduló embereket detektálni, illetve az előfordulásokat összeszámolni. Elsődleges use case egy olyan program készítése, mely képes egy utcai demonstráció képei alapján összeszámolni a résztvevők számát.

Az objektumdetektáláshoz a HOG (histogram of oriented gradients) + SVM (support vector machine) módszer került alkalmazásra, melynek fő előnye a viszonylagos egyszerűsége, módszertani eleganciája. A modell tanításához saját képi adatbázis készült, illetve ez kiegészült az internetről ingyenesen elérhető képekkel, melynek célja az alkalmazás robosztusságának javítása.

A feladat OpenCV könyvtárt használja és C++-ban van implementálva, Linux operációs rendszer alatt. Fontos, hogy feladatban a valós embereket LEGO™ figurák helyettesítik, a tanuló algoritmus is ezekkel lett tanítva. A programnak képesnek kell lennie egyaránt videóstream és képek alapján is dolgozni, ezeket inputként kezelni. A feladat tekinthető egyfajta járókelő detektálási problémának is (pedestrian/human detection), melynek kiterjett irodalma található az interneten, illetve az OpenCV könyvtár is nyújt beépített detektorokat, melyek a getDefaultPeopleDetector() és getDaimlerPeopleDetector() függvényekkel hívhatók. Ezek szándékosan nem kerültek alkalmazásra, mivel egyrészt ezek valós emberek detektálására lettek készítve, másrészt cél volt a kész modulok számának alacsonyan tartása.

# Elméleti háttér (30%)

A HOG (histogram of oriented gradients) módszertan kiegészítve SVM modellel, nalal Navneet és Bill Triggs 2005-ös munkája után lett népszerű és vált széles körben alkalmazottá. A módszer lényege a következő pontokban foglalható össze:

## Step#1 – Pozitív minta

Pozitív minta alatt értjük azokat a képeket, melyeket a tanuló algoritmus használ és melyeken szerepel az objektum, melynek detektálására a feladat irányul. Ezekből a képekből nyerhető ki a HOG jellemzője a detektálandó objektumnak, melyeket az SVM modell használ a detektáláshoz.

## Step#2 – Negatív minta

Azon képek halmaza, melyek nem tartalmazzák a detektálandó objektumot, így az SVM modell ezek alapján tanulja meg, hogy melyek azok a HOG jellemzők, melyek nem a detektálandó objektumhoz tartoznak.

## Step#3 – Pixelenkénti HOG jellemzők

A mintákban található képekből a HOG jellemzők kinyerése. A mintát alkotó képek 64x128-as méretben vannak elmentve. Az eredeti tanulmányban is ez a méret van, mivel ebben kényelmesen elfér egy ember és a valódi embereket szimuláló LEGO™ figurák esetében is megfelelő.

A HOG jellemző tulajdonképpen a pixelenkénti gradienst jelenti. Ennek egy x és y (vízszintes és függőleges) irányú komponense (Gx és Gy gradiens) az adott pixel szomszédjainak intenzitásából számítható. A Gx és Gy meghatározzák a gradiens nagyságát (Magnitude) és szögét (θ):

, ahol r,c a sorokra és oszlopokra vonatkoznak

## Step#4 – Cellánkénti HOG jellemzők, hisztogram

A pixelenkénti HOG jellemzőket egy 8x8 (vagy 4x4) cellákba rendezve előállíthatók a cellára jellemző gradiens komponensenkénti (magnitude – nagyság és direction – szög) mátrixok. A mátrixokban a szögekre vonatkoztatott értékkészlet 0-360 fok, hossz/nagyság értékkészlet 0-255. Szögek esetében jellemző ugyanakkor, hogy csak a 0-180 fokos értéktartomány szerepel (unsigned gradients), mivel a 0-360 fokos tartomány nem javítja érdemben a modelleket.

A cellára vonatkozó 2 mátrix értékeiből előállítható a cellára számított hisztogram, mely a 0-180 fokos (szög)értékeket tartalmazza, jellemzően 20 fokos beosztásban, összesen 9 kategóriát meghatározva a hisztogramon:

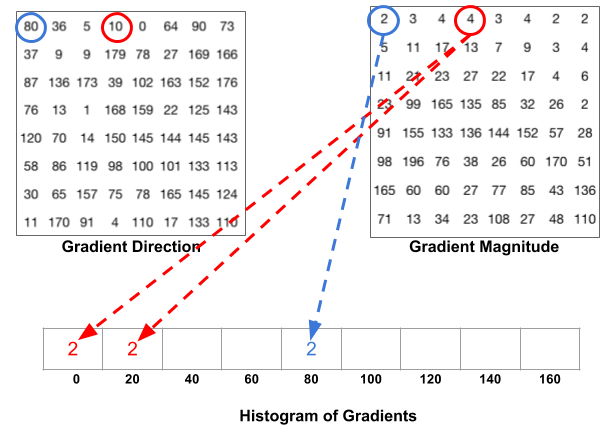


Figure – Gradiens mátrixok és hisztogram 8x8-as cellára számítva, Forrás: [Histogram of Oriented Gradients explained using OpenCV (learnopencv.com)](https://learnopencv.com/histogram-of-oriented-gradients/)

A kékkel jelölt értékek következő módon kerülnek besorolásra a hisztogramban: a szög értéke alapján kiválasztjuk a megfelelő hisztogram kategóriát. A szögérték 80, ami pontosan az 5. kategóriát (bucket-et) jelenti. A kategóriának megfelelő helyre beírható a gradiens mértékét mutató érték (Magnitude), ami 2.

A pirossal jelölt elemek esetében nem egyértelmű a besorolás, mivel a szögérték (10) pontosan 2 kategória (0 és 20) közé esik a hisztogramon. Ezért ennek az elemnek a nagysága a hisztogram osztályokba esés mértékének megfelelően két részre bontható. Mivel a 10-es szögérték pontosan a két hisztogram osztály határára esik, így fele-fele arányban tartozhat a 0 és 20 hisztogram osztályba. Tehát a 4-es méretből 2 kerül a 0-ás és 2 kerül a 20-as osztályba.

## Step#5 – Blokk normálás

A kialakított hisztogramok normálása annak érdekében, hogy az eredmény a fényviszonyokra kevésbé legyen érzékeny. A normáláshoz egy nagyobb blokk választható a stabilabb eredményért. Ezt az OpenCV algoritmusa automatikusan elvégzi, feltéve, hogy a paraméterezésnél figyelembe van véve a 16x16-os alapérték, és az előző lépésben használt cellák osztói ennek a 16x16-os alapértéknek (tehát 8x8 vagy 4x4 stb. megfelelő választás az előző lépésben).

## Step#6 – HOG jellemzők összegyűjtése és SVM modell tanítása

Az összes lépés elvégzése után lehetséges a HOG jellemzők összegyűjtése és az SVM modell illesztése.

# Megvalósítási terv és kivitelezés (40%)

## Github repository

A GitHub repository az alábbi linken érhető el: <https://github.com/lefodor/peoplecounter>

A dokumentum írása pillanatában privát, de a 2021/22 tavaszi félév végén és az értékelés befejeztével publikusan is elérhető lesz.

## Adatok, minták

A mintákhoz saját, generált adatok kerültek alkalmazásra, nem történt külső forrásból származó adatok használata. A minta megoszlását az alábbi táblázat mutatja:

|  |  |  |
| --- | --- | --- |
| Pozitív minta | Negatív minta | Teljes minta |
| #339 (45%) | #415 (55%) | #754 |

A pozitív minta szétbontható abból a szempontból is, hogy milyen perspektívából láthatók a detektálandó figurák:

|  |  |  |
| --- | --- | --- |
| Perspektíva | Leírás | Darabszám # |
| Front + Fourtyfive | Teljesen szemből, vagy szemből, ~45 fokos magasságból | 282 (83.18%) |
| Above | Felülről készült kép | 21 (6.2%) |
| Side | Oldalnézetből készült kép | 36 (10.62%) |

A képek 62x128-as felbontással készültek (width x height).

## Alkalmazások

### Build instrukciók

Az applikációk build-je CMAKE-kel történik. A Github repository-ban minden applikációhoz megtalálható a CMakeLists.txt file, mely tartalmazza a build-hez szükséges beállításokat. Fontos, hogy a CMake.txt file alábbi sorában az installált OpenCV könyvtár elérési útvonala szerepeljen.

set(OpenCV\_DIR "~/opencv/build")

A könyvtárban, melyben a letöltött forrásfájlok vannak, létre kell hozni egy build alkönyvtárat, és innen végrehajtani a

cmake --build .

parancsot és a – például a hogpedestrians esetében – ./hogpedestrians parancssorba beírásával indítható a program.

### Alkalmazások leírása

#### Hogpedestrians

Megadott input pozitív és negatív minták alapján kiszámítja a HOG jellemzőket, SVM modellt illeszt és output file-ba menti a készített HOG struktúra elemeit.

##### Argumentumok:

* pos\_dir: pozitív minta abszolút v. relatív elérési útvonala
* neg\_dir: negatív minta abszolút v. relatív elérési útvonala
* obj\_det\_filename: output file (.yml) elérése, ahova elmenti
* detector\_width, detector\_height: mintákhoz használt képméret (64x128), jelenleg minden képnek azonosan 64x128 pixel méretűnek kell lennie.

##### Függvények

**void ResizeBox(cv::Rect& box):** Detektálások köré rajzolt téglalapot átméretezi, nincs használatban.

**std::vector< float > get\_svm\_detector(   
const cv::Ptr< cv::ml::SVM >& svm )**: Egy SVM modell segítségével inicializálja egy HOGDescriptor objektum detektorját.

**void load\_images(   
const cv::String & dirname,   
std::vector< cv::Mat > & img\_lst,   
bool showImages = false )**: Dirname könyvtárban megadott image file-okat betölti egy image vektor-ba.

**void sample\_neg(   
const std::vector< cv::Mat > & full\_neg\_lst,   
std::vector< cv::Mat > & neg\_lst,   
const cv::Size & size )**: negatív minta esetén megengedett, hogy a mintakép eltérjen a 64x128 pixeles képmérettől. Ebben az esetben ez a függvény véletlenszerűen kiválaszt egy ekkora méretű részletet az eredeti képből és azt használja a mintában.

**void computeHOGs(   
const cv::Size wsize,   
const std::vector< cv::Mat > & img\_lst,   
std::vector< cv::Mat > & gradient\_lst, bool use\_flip ):** előállítja a HOG jellemzőket a pozitív és negatív minták alapján

**void convert\_to\_ml(   
const std::vector< cv::Mat > & train\_samples,   
cv::Mat& trainData ):** előkészíti az adatokat, az SVM modellhez

#### Hogtestimg

Parancssorban megadott képfájlra lefuttatja a megadott objektumfelismerő modellt (detectionoutput.yml).

##### Argumentumok

* obj\_det\_filename: .yml file elérhetősége, melyben a hogpedestrians program által elmentett output szerepel
* parancssorban megadott képfájl elérési útvonala

Példa a kimenetre:

A screenshot of a video game

Description automatically generated

##### Hogteststream

Webkamerás stream-re lefuttatja a megadott objektumfelismerő modellt (detectionoutput.yml).

##### Argumentumok

* obj\_det\_filename: .yml file elérhetősége, melyben a hogpedestrians program által elmentett output szerepel

Példa a kimenetre:



# Teszt (20%)

## Tesztelési terv

Az alkalmazás tesztelése képekre lett végezve a Github repo-ban található hogtestimg és hogeststream programokkal. Alapvetően az alkalmazás használata az alábbi szcenáriókban kerültek tesztelésre:

* Front: a detektálandó objektum a kamerával szemben helyezkedik el – 4 tesztadat
* Fourtyfive: a detektálandó objektum a kamera alatt ~45 fokos szögben helyezkedik el - 7 tesztadat
* Above: a detektálandó objektum a kamera alatt helyezkedik el (nem pontosan 90 fokos szög, de megközelíti a derékszögű beesést) – 4 tesztadat
* Structure: a detektálandó objektumok valamilyen struktúrán vannak elhelyezve (épület szimulálása) – 3 tesztadat

A front és fourtyfive csoportban levő teszteseteknél szerepelnek átfedéses esetek is, ezek kiértékelő táblázatban külön (o) szimbolummal vannak jelezve.

## Teszteredmények

Az eredmények a következő részekben található táblázatokban kerülnek bemutatásra. A táblázat oszlopai:

* Actual: ténylegesen mennyi LEGO figura helyezkedik el a képen, ezeket kell detektálni
* Detected#: korrekt detektálások száma
* Detected, False+: hibás pozitív.
* Result: százalékos érték, mutatja hány százalékos detektálási arány az adott tesztadat esetében. Számítás: Detected# / Actual.

A False+ oszlopban a (\*) jelölés mutatja, ha a hibás pozitív észlelés elkerülhető lenne a non-maxima suppression módszer alkalmazásával, ami az átfedő detektálások esetében összevonja ezeket. Ez nem került implementálásra az alkalmazásban. Ebben az esetben 1 korrekt detektálás átfed 1 fals pozitív észleléssel, amit a módszer kiszűrne. A csillagok száma azt jelzi, hány fals pozitív lenne elkerülhető, ezek a fals pozitív detektálások tulajdonképpen 1 korrekt észlelés duplikációinak tekinthetők.

### Képek

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Group | Name | Actual # | Detected (#, False+) | | Result (%) |
| Front | front\_12\_37\_17 | 3 | 2 | 0 | 66.67% |
| Front | front\_12\_37\_42 | 3 | 2 | 1(\*) | 66.67% |
| Front | front\_12\_38\_06 (o) | 3 | 2 | 1(\*) | 66.67% |
| Front | front\_12\_38\_30 (o) | 2 | 0 | 0 | 0% |
| Fourtyfive | fourtyfive\_12\_39\_00 | 3 | 3 | 1(\*) | 100% |
| Fourtyfive | fourtyfive\_12\_39\_14 | 2 | 2 | 1(\*) | 100% |
| Fourtyfive | fourtyfive\_12\_40\_05 | 5 | 4 | 1 | 80% |
| Fourtyfive | fourtyfive\_12\_40\_37 (o) | 3 | 3 | 0 | 100% |
| Fourtyfive | fourtyfive\_12\_40\_42 (o) | 2 | 1 | 0 | 50% |
| Fourtyfive | fourtyfive\_12\_41\_04 (o) | 3 | 1 | 0 | 50% |
| Fourtyfive | fourtyfive\_12\_41\_22 (o) | 2 | 2 | 0 | 100% |
| Above | above\_12\_45\_37 | 5 | 0 | 0 | 0% |
| Above | above\_12\_45\_40 | 3 | 1 | 0 | 33.33% |
| Above | above\_12\_45\_45 | 4 | 0 | 0 | 0% |
| Above | above\_12\_45\_49 | 4 | 0 | 0 | 0% |
| Structure | structure\_12\_57\_40 | 5 | 3 | 3(\*) | 60% |
| Structure | structure\_12\_58\_18 | 5 | 4 | 2 | 80% |
| Structure | structure\_12\_58\_41 | 5 | 3 | 4(\*\*) | 60% |

### Vizualizált eredmények

#### Group: Front

|  |  |
| --- | --- |
| Name | Image output |
| front\_12\_37\_17 | A picture containing text, indoor, toy  Description automatically generated |
| front\_12\_37\_42 | A picture containing indoor, toy  Description automatically generated |
| front\_12\_38\_06 (o) | A picture containing text, wall, indoor  Description automatically generated |
| front\_12\_38\_30 (o) | A picture containing indoor  Description automatically generated |

#### Group: Fourtyfive

|  |  |
| --- | --- |
| Name | Image output |
| fourtyfive\_12\_39\_00 | A picture containing text, indoor  Description automatically generated |
| fourtyfive\_12\_39\_14 | A screenshot of a video game  Description automatically generated with low confidence |
| fourtyfive\_12\_40\_05 | A screenshot of a video game  Description automatically generated |
| fourtyfive\_12\_40\_37 (o) | A screenshot of a video game  Description automatically generated with medium confidence |
| fourtyfive\_12\_40\_42 (o) | A picture containing text  Description automatically generated |
| fourtyfive\_12\_41\_04 (o) |  |
| fourtyfive\_12\_41\_22 (o) | Graphical user interface  Description automatically generated with medium confidence |

#### Group: Above

|  |  |
| --- | --- |
| Name | Image output |
| above\_12\_45\_37 | A picture containing indoor, toy  Description automatically generated |
| above\_12\_45\_40 | A screenshot of a video game  Description automatically generated with low confidence |
| above\_12\_45\_45 |  |
| above\_12\_45\_49 |  |

#### Group: Structure

|  |  |
| --- | --- |
| Name | Image output |
| structure\_12\_57\_40 | A picture containing indoor, wall, toy, cluttered  Description automatically generated |
| structure\_12\_58\_18 | A picture containing indoor, wall, toy  Description automatically generated |
| structure\_12\_58\_41 | A picture containing indoor, toy  Description automatically generated |

## Teszteredmények kiértékelése

### Mutatók csoport szintű bontásban

|  |  |  |
| --- | --- | --- |
| Group | Mean Result % | Mean False+ (%) |
| Front | 50% | 25% |
| Fourtyfive | 83% | 15.79% |
| Above | 8.33% | 0% |
| Structure | 66.67% | 54.54% |

Mean False+ (%) = ∑Detected, False+ / ∑(Detected# + Detected, False+), vagyis csoporton belül összes detektálás hány százaléka fals pozitív.

A csoportszintű aggregált mutatókól látható, hogy a LEGO felismerő alkalmazás a Front és Fourtyfive csoportokban teljesít jól ( a front csoportban a front\_12\_38\_30 jelzésű kép ad 0%-t, ami elrontja az egyébként 66.67%-os eredményt). A kapott számok visszatükrözik a tanulási minta jellemzőit, abban az értelemben, hogy a mintában szereplő legtöbb elem a tesztelésnél használt Front és Fourtyfive (a teljesen szemből és 45 fokos magasságból) csoportokból kerültek ki.

A Structure csoport is jól teljesít, ezt tulajdonképpen a Front és Fourtyfive csoportokba sorolt képek keveréke, megtoldva egyéb elemekkel, amelyek a magasabb, 54.54%-os fals pozitív találatokat okozzák. A részeredmény alapján elmondható, hogy szükséges lehet a negatív minta bővítése olyan adatokkal, melyekben a teszteseteken látható struktúrák szerepelnek, illetve értelemszerűen olyan környezeti elemekkel, melyek alkalmazás során szintén fals pozitív eredményeket hozhat.

A minta arányain is szükséges lehet módosítani, mivel a minta jelenlegi megoszlása pozitív/negatív részre 45/55%-os, tehát hozzávetőlegesen 1:1 az arány. A szakirodalomban általában 1:10, de minimum 1:4 szerepel a negatív mintaelemek javára.

További javítási lehetőségek:

* Átfedő detektálások összevonására non-maxima suppression módszer alkalmazása, ezzel csökkenthető a fals pozitív detektálások száma.
* Detektor kalibrációja, a detektor paraméterei jelenleg állandóak, de lehetséges ezek futásidőben történő dinamikus változtatása, a detektor körülményekhez / üzemmódhoz való kézi kalibrálása. (+) lehetséges mindez automatikus módon is.
* Külső adatok használata az adatbázis (tanulási és tesztelési, validációs) bővítésére, valamint pozitív / negatív mintaarány javítása (eltolása negatív irányba).

# Felhasználói leírás (5%)

# References

Dalal, N., & Triggs, B. (2005). *Histograms of Oriented Gradients for Human Detection.* Retrieved from INRIA: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

Mallick, S. (2016, December). *LearnOpenCV.* Retrieved from LearnOpenCV: https://learnopencv.com/histogram-of-oriented-gradients/