**Eötvös Loránd Tudományegyetem**

**Informatikai Kar**

**A kép külső részén lévő információt nem tartalmazó területek meghatározása**

**Szerző:**

Székely Dorina

Programtervező informatikus BSc.

Budapest, 2022

Tartalomjegyzék

[2 Bevezetés 3](#_Toc104406546)

[2.1 Orvostechnológia 4](#_Toc104406547)

[3 Felhasználói dokumentáció 6](#_Toc104406548)

[3.1 Általános leírás 6](#_Toc104406549)

[3.2 Technológiai háttár 7](#_Toc104406550)

[3.2.1 Programozási nyelvek 7](#_Toc104406551)

[3.2.2 Fejlesztőkörnyezet 7](#_Toc104406552)

[3.2.3 Felhasznált könyvtárak 8](#_Toc104406553)

[3.3 Felhasználói felület 9](#_Toc104406554)

[3.3.1 Use Case diagram 9](#_Toc104406555)

[3.3.2 A program használata 10](#_Toc104406556)

[4 Fejlesztői dokumentáció 12](#_Toc104406557)

[4.1 Képfeldolgozási szűrések, algoritmusok 12](#_Toc104406558)

[4.1.1 Morfológiai képfeldolgozások 12](#_Toc104406559)

[4.1.2 Szűrés 12](#_Toc104406560)

[4.1.3 Canny 14](#_Toc104406561)

[4.2 A program részei 14](#_Toc104406562)

[4.3 A képfeldolgozás lépései a programomban 14](#_Toc104406563)

[4.4 A felhasználói felület – Windows Forms 16](#_Toc104406564)

[4.5 OpenCV 16](#_Toc104406565)

[5 Tesztelés 17](#_Toc104406566)

[6 Irodalomjegyzék 18](#_Toc104406567)

A felhasználói dokumentációban általánosabb leírás kell a szűrése, algoritmusok részbe, a fejlesztőibe OpenCV specifikusan leírva.

# Bevezetés

**A képfeldolgozást, ahogy számos más területet is elősegítette a digitalizáció.** A 70-es években kezdtek el ezek rohamosan fejlődni. Folyamatosan nőtt a számítógépek memóriája és a processzorok gyorsasága, melynek köszönhetően összetettebb szoftverek készülhettek, lehetővé téve az egyre komplexebb és pontosabb számításokat. Az elmúlt években jelentősen megnőtt az érdeklődés a képmorfológia, a neurális hálózatok, a színes képfeldolgozás, a képadatok tömörítése, a képfelismerés és a tudásalapú képelemző rendszerek iránt.

A digitális képfeldolgozó rendszerben a folyamat első lépése a kép beszerzése. A digitalizálási eljárások hibával járnak, ez rontja a kép minőségét, hamis információt adhat, zajok keletkezhetnek. A kép digitalizálása után a következő lépés az előfeldolgozás, mely irányulhat a digitalizálás során keletkezett hibák javítására, visszaállítására, de mivel az eredeti kép minőségének mértékét nem ismerjük, ezért ezek a javítások szubjektívek.

Az átalakítások történhetnek annak érdekében, hogy jobban megfeleljen céljainknak.

Funkciója a kép olyan javítása, amely növeli a többi folyamat sikerének esélyét, a következő lépés a szegmentálással, a bemeneti kép felosztásával foglalkozik. alkotórészeibe vagy tárgyaiba az Ábrázolás és Leírás foglalkozik a gyártmányadatok számítógépes feldolgozásra alkalmas formában történő rögzítésével, és ezt követően a felismerés az, amely címkét rendel egy objektumhoz, és az utolsó értelmezés magában foglalja a felismert objektumok halmazának jelentését.

Egy kép feldolgozása azt jelenti, hogy valamilyen műveletet kell alkalmazni a képen.

***Az alakfelismerés amikor az objektumokat a kép makrostruktúrájábol származtattuk le.***

Mára már a mindennapjainkban jelen van a képfeldolgozás. Életünk számos területén találkozhatunk vele. Kezdetben az űrkutatásban használták

**Milyen területeken jelenik még meg.**

A képfeldolgozást több különböző területen vált alkalmazhatóvá. Kezdetben a legrégebbi és legáltalánosabb területeken tudták sikeresen alkalmazni. Például a hadászatban, ahol a célpontok pontosabb követésében segítette a katonákat, akár a távolabb lévőket is. Emellett az első alkalmazási területek között van a régészet is, melynél hozzájárult a bonyolultabb lelőhelyek gyorsabb és részletesebb feltérképezéséhez. Az elmúlt két évtizedben újabb alkalmazási lehetőségek jelentek meg, melyek a mindennapokban is jelen vannak. Elég csak az újabb mobiltelefonokra gondolni, ahol már lehetséges az ujjlenyomatos és/vagy arcképes feloldás, mely a megnövekvő biztonságtechnikai elvárásoknak akart megfelelni.

Mindennapi életünket nagyban befolyásolják az orvostechnológiában használatos képalkotó gépek, melyek segítségével a különböző betegségek, rendellenességek könnyebben diagnosztizálhatóak. Ezek a digitális képalkotó gépek a tomográfok, közismertebb nevén CT, illetve a mágneses rezonanciaműszerek, másnéven MRI.

Dolgozatom témája az orvosi képalkotó műszer által készített képek feldolgozásán alapul, és ezt eljárásmódot kiterjesztettem sima egyszerű képekre, melyek megjeleníthetnek beszkennelt dokumentumokat, vagy egy rólunk készült fotót.

## Orvostechnológia

<https://hu.wikipedia.org/wiki/Orvosi_k%C3%A9palkot%C3%A1s>

A képalkotásnak fontos szerepe van a modernkori orvoslásban, egészségügyben. Mi is az orvosi képalkotás? Idesorolunk minden olyan technikát és eljárást, melynek célja, hogy az emberi testről minél pontosabb felvételek készülhessenek. Ezen fejlesztéseknek köszönhetően ma már lényegesen gyorsabban és pontosabban tudják megállapítani a különböző problémákat, felismerni a rendellenességeket és orvosolni az elváltozásokat, betegségeket, így segítve, hogy egészségesebben és tovább élhessünk. Az elmúlt időszakban lényegesen nőtt az ember átlagéletkora, és ebben nagy szerepet játszott a modern orvoslás, és a fejlett orvostechnológia, melynek egyik lényeges alapja az orvosi képalkotás. Az elmúlt két évtizedben is hatalmas áttöréseket értek el ezen a területen, mint például …

A jövőre tekintve nagyon hosszú a kitűzött célok listája, melyek csak időkérdése és megvalósításra kerülnek.

**Bevezetés**: Itt le kell írni, miről is fog szólni a szakdolgozatot, leírod mi a motiváció. Itt kell felkelteni az olvasó érdeklődését, hogy el akarja olvasni a dolgozatodat. Itt nem kell belemenni a részletekbe, itt lehet felületesen fogalmazni, hogy egy egyszerű halandó is megértse, miről szól a dolgozat. Itt veted fel a problémát, amivel elkezdtél foglalkozni. Bemutatod, hogyan akarod ezt megoldani. Lényegében egy előzetes áttekintést adsz a dolgozatról.

a témaválasztás indoklását,

és a megoldandó feladat rövid, közérthető leírását.

# Felhasználói dokumentáció

## Általános leírás

Programomban a már említett orvosi vonatkozású problémából indultam ki, mely az, hogy az elkészített felvételeken lehet olyan terület is, mely a vizsgálat szempontjából nem érdekelt részeket tartalmaz. Ezen részek detektálása azért fontos, mert lassíthatja, vagy akár gátolhatja a kép feldolgozását és helyes értelmezést. Hasonló probléma merülhet fel a szkenneléssel digitalizált képeknél vagy dokumentumoknál. Előfordulhat, hogy nem tudjuk pontosan beállítani a szkennelni kívánt fotó, vagy dokumentum méretét a digitalizáló eszközön, vagy az elmozdul. Ezzel az eljárással készült kép külső, érdektelen területe sötét színű, hogy minél jobban hasonlítson a röntgenképekre, ugyanis főként ezen orvosi vonatkozású problémának a megoldására szerettem volna koncentrálni. Tehát, a feldolgozás olyan képeket dolgoz fel / vizsgál, melyek egy viszonylag homogén, de zajt tartalmazó külső terület(ek)ből és egy konvex belső területből áll, mely a külsőtől jó kontraszttal elkülönül.

A program a képfeldolgozás során szűréseket és algoritmusokat alkalmazva egyeneseket detektál, majd visszaadja azokat, melyek az érdekelt terület határvonalai.

Első sorban a programom az orvosokat célozza meg, mivel ők tudják az elkészült röntgenképeket megvizsgálva meghatározhatják a különböző elváltozásokat, melynek köszönhetően segíteni tudnak a betegeken. Mivel az egészségügyben, illetve egyéb területeken használt programok esetében a szinte tökéletes működés és a különböző algoritmusok, valamint számítások gyors lefutása a cél, ezért kezdetben ezek fejlesztésekor kevés hangsúlyt kap a felhasználói felület megjelenése. Ma már a szoftver ezen része is hangsúlyos, mivel a számítógépek fejlődésével gyorsult a lefutási idő, a memóriák mérete, és azok elérése.

A program hasznos lehet még bárki számára, aki rendelkezik a feltételeknek megfelelő paraméterekkel rendelkező digitalizált képpel, annak a program futása során vissza fogja adni az elvárt kimenetet.

## Technológiai háttár

### Programozási nyelvek

Programom magját C++ nyelven írtam, illetve az OpenCV könyvtár függvényeit használatam. Ez a leggyakrabban ajánlott programozási nyelvek között van a hasonló képfeldolgozási programok megvalósításához. A modern nyelveket figyelembe véve a Python lett volna a másik lehetőség, azonban ezen nyelvvel kevesebb tapasztalatom van. A C++ alapokat az egyetemi tanulmányaim során sajátítottam el, melyet a gyakornoki munkám során, illetve különböző online könyvek, fórumok segítségével bővítettem.

A grafikus felhasználói felület elkészítéséhez olyan nyelvet szerettem volna választani, mely könnyen implementálható, és metodikájában is közel áll a programom fő nyelvéhez, a C++-hoz. Ezért esett a választásom a C#-ra, melyben a Windows Forms osztálykönyvtárat használtam. Ajánlották még a WPF (Windows Presentation Foundation) keretrendszert, de mivel minél egyszerűbb megjelenésű UI-t szerettem volna, az előbbi keretrendszerrel is meg tudtam valósítani az elképzeléseimet. A C# programozási nyelv és a WinForms megfelelő alkalmazásához különböző online könyvek és fórumok voltak segítségemre. Ezekből a technológiákból nem volt korábban szerzett tudásom, így ezekhez nagyobb előtanulásra volt szükségem.

### Fejlesztőkörnyezet

Programomat Windows operációs rendszeren, a Visual Studio 2019 programban készítettem el. Ehhez szükséges alapokat az egyetemen, illetve a gyakornoki munkám során már elsajátítottam, így könnyebben ment a munka a program használatával.

#### A fejlesztőkörnyezet előállítása

A Visual Studio 2019 program a

<https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes> oldalról letölthető. A telepítés során szükséges c++, illetve a .NET desktop development kiegészítők letöltése is.

OpenCV könyvtár letölthető a <https://sourceforge.net/projects/opencvlibrary/files/opencv-win/> oldalról. A mappa kicsomagolása után szükséges a környezeti változók közé hozzáadni a opencv\build\x64\vc14\bin könyvtár teljes útvonalát.

A Visual Studio-ban megnyitott vagy létre hozott solution projektjeinek tulajdonságait kell még beállítani. A c++ -> General „Additional Include Directories” alatt hozzá kell adni az OpenCV include mappájának elérési útvonalát ($(OPENCV\_DIR)\..\..\include). Linker -> General menüponton belül, a "Additional Library Directories" alatt hozzá kell adni a libs könyvtárat ($(OPENCV\_DIR)\lib). Ezután meg kell adni azokat a könyvtárakat, amelyekbe a linker-nek be kell néznie. Ehhez a Linker -> Input menüpont, "Additional Dependencies" alatt hozzá kell adni az összes használni kívánt modul nevét. A programom esetén ezek a következő modulok:

A képen szöveg látható

Automatikusan generált leírás

Ezen beállítások után az osztálykönyvtár alkalmazhatóvá, futtathatóvá válik a Visual Studio-n belül.

### Felhasznált könyvtárak

#### OpenCV

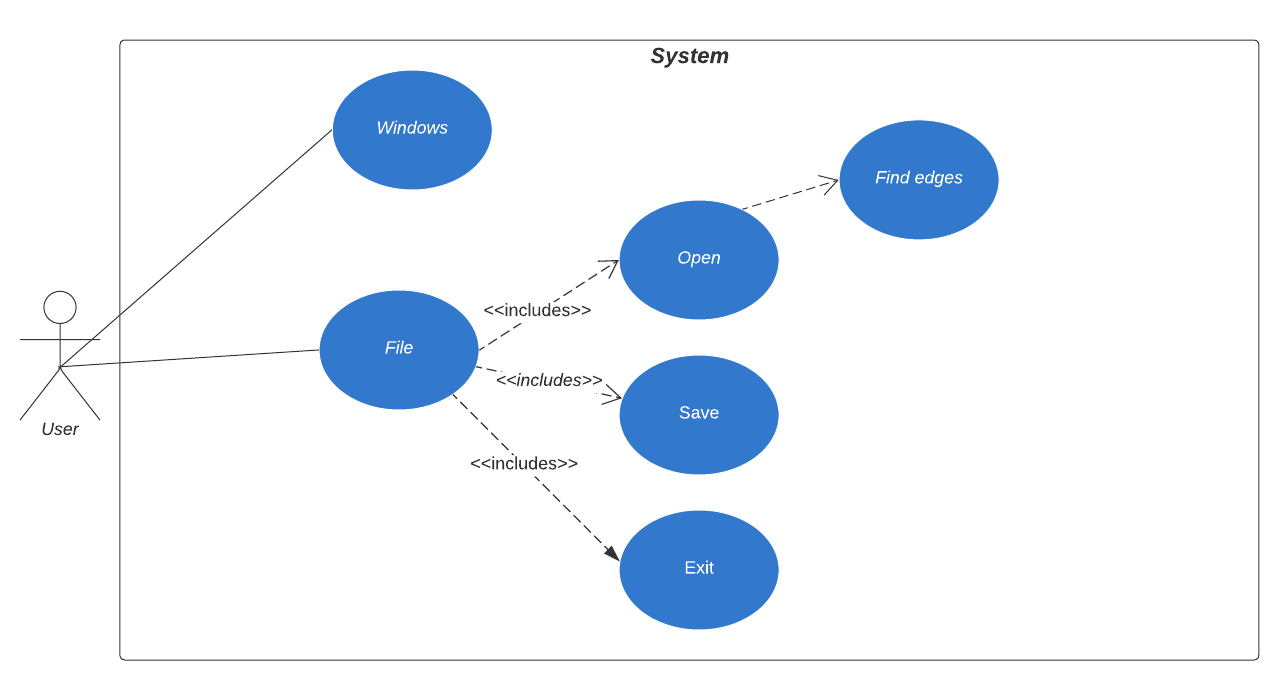
Az OpenCV (Open Source Computer Vision Library) egy nyílt forráskódú számítógépes látás- és gépi tanulási szoftverkönyvtár. Az OpenCV-t azért hozták slétre, hogy közös infrastruktúrát biztosítson a számítógépes látási alkalmazásokhoz, és felgyorsítsa a gépi érzékelés használatát a kereskedelmi termékekben. A könyvtár több mint 2500 optimalizált algoritmussal rendelkezik. Ezek az algoritmusok használhatók arcok észlelésére és felismerésére, tárgyak azonosítására, kameramozgások nyomon követésére, mozgó objektumok követésére, objektumok 3D-s modelljeinek kinyerésére, 3D pontfelhők előállítására sztereó kamerákból, képek összefűzésére, hasonló képeket megtalálására egy képadatbázisban, a vörös szemeket eltávolítására a vakuval készített képekről, szemmozgások követésére, a tájak felismerésére, stb.

A Google, a Microsoft és még számos ismert nagyvállalat, és startup cég is alkalmazza az OpenCV könyvtárban található algoritmusokat, függvényeket. Különböző területeken jelenik meg a használata, például segítséget nyújt a robotok navigálásába, és abban, hogy tárgyakat vehessenek fel, bányászati eszközök monitorozásában, termékek címkéjének ellenőrzésében, arcfelismerésben, és még sok más területen.

C++, Python, Java és MATLAB felületekkel rendelkezik, támogatja a Windows, Linux, Android és Mac OS operációs rendszereket. Az OpenCV leginkább a valós idejű képalkalmazások (…). Az OpenCV nyelvezete C++.

## Felhasználói felület

### Use Case diagram



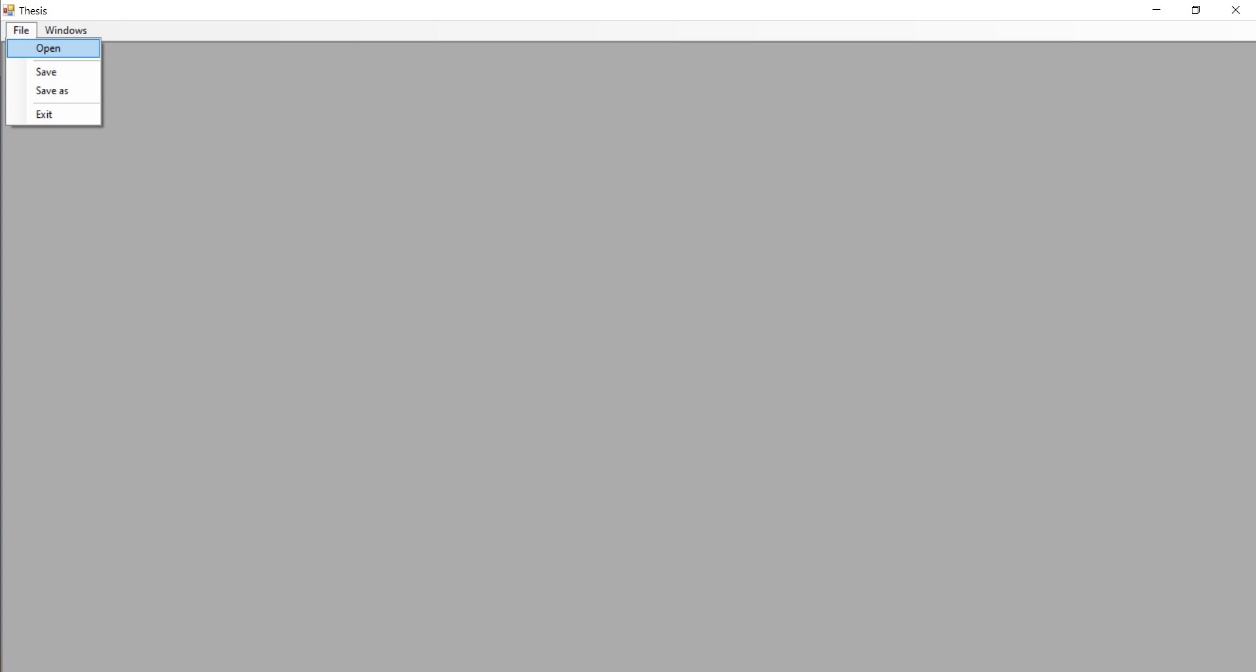
### A program használata

#### Ebben a részben a programom interaktív grafikus felhasználói felületét (Graphical User Interface - GUI) ismertetem.

Minden szoftvernél fontos tartozék az általános felhasználói tájékoztató, más néven használati útmutató. Ennek célja, hogy a *user* használat előtt megkapja a szoftverhez szükséges alap információkat.

A program indításakor megjelenik a fő ablak, melyen keresztül lehet a képeket betölteni. Fontos megjegyezni, hogy itt csak a támogatott fájlformátumú fájlok választhatók ki, melyek a .png, .jpg, .jpeg kiterjesztések. Fontos megjegyezni, hogy a program nem fut hibára, és nem enged semmit csinálni, míg a kép nincse betöltve a rendszerbe. Ezen fő funkción kívül más lehetőségek is elérhetők, úgy, mint a már betöltött képek visszatöltése. Ezeket a windows menüpont alatt listázza ki a program, bezárása után, új munkamenetnél már nem fog emlékezni rá. *A visszatölteni kívánt képre kattintva, Mentése a felhasználó által kiválasztott helyre.* A hasznos területet és az információt nem tartalmazó területeket határoló élek detektálását követően megtekinthető az eredeti kép a határoló élekkel, illetve nyomon követheti a felhasználó milyen változásokon ment végig a kép a végeredmény eléréséig. Ezt követően jön képbe az a hasznos funkció mellyel el lehet menteni ezt a képet bármelyik mappába a Save As funkcióval. A felsorolt funkciókon, lehetőségeken kívül még tudjuk a képeket forgatni, illetve emelni, vagy csökkenteni fényerőt.

*A program elindulása után, mikor még nincsnek képek betöltve, a File menüre kattintás után egy lehetőségünk van, az, hogy Open menüpontra kattintva betölteni egy általunk kiválasztott képet. Ezt a felugró Open File Dialog ablakban tudjuk kiválasztani. Semmilyen más menüpont nem választható ebben az állapotban, minden más le van tiltva. A megnyíló ablakban már ezek elérhetőek, megjelenik a kiválasztott képünk, illetve jobb oldalt három gomb és egy* ***trackbar. Forgathatjuk a képet, állíthatjuk a fényer***



# Fejlesztői dokumentáció

## Képfeldolgozási szűrések, algoritmusok

Programomban használtam OpenCV könyvtárbeli függvényeket,

### Morfológiai képfeldolgozások

A morfológiai képfeldolgozás alaktant jelent, mely a halmazokon elvégezhető műveletek jelenti. Ebben az eljárásban képet hasonlítunk össze egy csúsztatott strukturáló elemmel, melyet "megütköztetünk" képi objektumokkal. Ez a folyamat mintaillesztéshez hasonlítható. A strukturáló elem

### Szűrés

Az egydimenziós jelekhez hasonlóan a képek is szűrhetők különféle aluláteresztő szűrőkkel (LPF – low-pass filter), felüláteresztő szűrőkkel (HPF – high-pass filter). Az LPF segít a zaj eltávolításában, a képek elmosódásában stb. A HPF szűrők segítenek megtalálni a széleket.

A kép elmosódását úgy érik el, hogy a képet egy aluláteresztő szűrő kernellel vonják össze. Ez hasznos a zaj eltávolítására. Valójában eltávolítja a képről a magas frekvenciájú tartalmat (pl. zaj, élek). Tehát az élek egy kicsit elmosódnak ebben a műveletben (vannak olyan elmosási technikák is, amelyek nem homályosítják el a széleket). Az OpenCV az elmosási technikák négy fő típusát kínálja.

* Átlagolás

Ez úgy történik, hogy egy képet összevonunk egy normalizált dobozszűrővel. Egyszerűen leveszi a kernelterület alatti összes képpont átlagát, és lecseréli a központi elemet.

* Gauss-elmosódás

Ebben a módszerben a dobozszűrő helyett Gauss-kernelt használnak. Ez a cv.GaussianBlur() függvénnyel történik. Meg kell adnunk a kernel szélességét és magasságát, aminek pozitívnak és páratlannak kell lennie. Meg kell adnunk a szórást is X és Y irányban, sigmaX és sigmaY. Ha csak a sigmaX van megadva, a sigmaY megegyezik a sigmaX-szel. Ha mindkettő nulla, akkor a rendszer a kernel méretéből számítja ki. A Gauss-féle elmosódás rendkívül hatékonyan eltávolítja a képről a Gauss-zajt.

* Medián elmosódás

Itt a cv.medianBlur() függvény a kernelterület alatti összes pixel mediánját veszi fel, és a központi elemet ezzel a medián értékkel helyettesítjük. Ez rendkívül hatékony a képen megjelenő só-borszaj ellen. Érdekes módon a fenti szűrőkben a központi elem egy újonnan számított érték, amely lehet egy pixelérték a képen vagy egy új érték. Ám a medián elmosódásnál a központi elemet mindig valamilyen pixelérték váltja fel a képen. Hatékonyan csökkenti a zajt. A kernel méretének pozitív páratlan egész számnak kell lennie.

* Kétoldalú szűrés

A cv.bilateralFilter() rendkívül hatékony a zaj eltávolításában, miközben éles marad az élek között. De a működés lassabb a többi szűrőhöz képest. Már láttuk, hogy egy Gauss-szűrő veszi a pixel környékét, és megtalálja annak Gauss-féle súlyozott átlagát. Ez a Gauss-szűrő önmagában a tér függvénye, vagyis a szűrés során a közeli pixeleket veszik figyelembe. Nem veszi figyelembe, hogy a pixelek közel azonos intenzitásúak-e. Nem veszi figyelembe, hogy egy pixel élpixel-e vagy sem. Így a széleket is elmosja, amit nem akarunk.

A bilaterális szűrés szintén egy Gauss-szűrőt vesz igénybe a térben, de még egy Gauss-szűrőt, amely a pixelkülönbség függvénye. A tér Gauss-függvénye biztosítja, hogy csak a közeli pixeleket vegye figyelembe az elmosásnál, míg az intenzitáskülönbség Gauss-függvénye biztosítja, hogy csak a központi képponthoz hasonló intenzitású képpontokat vegye figyelembe az elmosásnál. Így megőrzi az éleket, mivel a széleken lévő pixelek intenzitása nagy eltérést mutat.

Gauss-szűrő

A Gauss-szűrőt képek élsimításánál alkalmazzák. Ilyenkor a képet Gauss függvénnyel konvolváljuk. Ilyenkor dióhéjban az történik, hogy a kiinduló pont intenzitását lecserélik a korábban már kiszámolt minden pont intenzitásának gauss függvénnyel vett szorzatára, ahol a gauss függvény bemenete a az adott pont koordinátáinak és a kiinduló pont koordinátáinak a különbsége. Természetesen az egész folyamat legalább két dimenzióra van értelmezve. A mellékelt linken részletesebb leírás található, az ábrán pedig a gyakorlati ereje.

### Canny

Programom során az OpenCV segítségével felhasználtam a Canny él detektáló függvényt, mely ebben a kategóriában az egyik legelterjedtebb, azért mert nagyon rugalmas és könnyen használható. Alapvetően négy fázisból áll, mellyel eléri a kívánt eredményt. Zajcsökkentés, a kép intenzitás gradiensének, hamis élek elnyomása és hiszterézis küszöbérték. Ezeknek is köszönhető, hogy az él detektáló algoritmusok közül a legjobb teljesítményt a Canny nyújtja.

## A program részei

Programom két projektből áll, magából a képfeldolgozásból, ami C++ nyelven íródott, illetve a C# - Windows Forms projektből, ami pedig a felhasználói felület. Az előbbi projektből DLL-t (Dynamic Link Library) készül, amit a C# projekt importál.

## A képfeldolgozás lépései a programomban

Feladatom fő része a képfeldolgozás, melyet a DetectEdges osztály végez el. Ezt az osztályhoz két állomány tartozik: detectEdges.h és a detectEdges.cpp. A projektben ezeken kívül még két fájl található, a thesis.h és a thesis.cpp. A cpp fájl azt a függvényben kerül meghívásra a képfeldolgozó függvény és egyéb számítások úgy, mint a feldolgozandó kép szükség szerinti átskálázása.

A DetectEdges osztály a következő függvényeket tartalmazza:

A képen szöveg látható

Automatikusan generált leírás

* DetectEdges():
* imageProcessing( Mat img )

A függvény bemenete az a kép, melyen a felhasználó végre szeretné hajtani a műveleteket. A következő műveleteket mindig az előző lépésben átalakult képen végzi. A függvényben első lépésként szürkeárnyalatossá alakítja a bemenetet. Ezután az esetlegesen keletkezett zajokat, illetve az erős éleket a Gauss-szűrés csökkenti, eltávolítja. A Canny éldetektáló algoritmus meghatározza ezen szűrt képen a fontosabb éleket.

**dilatáció**

**erodálás**

Az utóbbi négy lépés mindegyike után az eredmény mentésre kerül.

* + A képen szöveg, személy, képernyőkép, dokumentum látható

    Automatikusan generált leíráskimenet: Mat
* getContours( Mat image, Mat imgResized )
  + kimenet: vector<Point>
* drawPoints( vector<Point> points, Scalar color, Mat imgResized )
  + kimenet: void
* distanceBetweenPoints( Point a, Point b )
  + kimenet: double
* getLineTwoPoints( vector<Point> )
  + kimenet: pair<Point, Point>
* reorder( vector<Point> points )
  + kimenet: vector<Point>
* saveImage( Mat image, String process, String name )
  + kimenet: int
* setImageName( String path )
  + kimenet: void
* getImageName()
  + kimenet: String

Ebben az osztályban több OpenCV könyvtárbeli függvényt, illetve általam készített függvényt használtam, melyek működését később részletezek.

Egyéb felhasznált akármik: Mat, vector, pair<,>

## A felhasználói felület – Windows Forms

## OpenCV

# Tesztelés

# Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | The OpenCV Library [Online]. [Hozzáférés dátuma: 2022.04.15]  **https://opencv.org/**. |
| [2] | „ELTE EDIT - IK Szakdolgozatok,” [Online]. Available: http://edit.elte.hu/xmlui/handle/10831/27. [Hozzáférés dátuma: 09 07 2019]. |
| [3] | „Murphy törvénye - wikipédia,” [Online]. Available: https://unciklopedia.org/wiki/Murphy\_t%C3%B6rv%C3%A9nyei. [Hozzáférés dátuma: 09 07 2019]. |
| [4] | „ BSc pontrendszer,” [Online]. Available: https://www.inf.elte.hu/dstore/document/1667/Bsc\_b%C3%ADr%C3%A1lat.docx. [Hozzáférés dátuma: 19 11 2019]. |
| [5] | G. Gergő, „Szakdolgozat Word sablon,” 2019. [Online]. Available: http://ggombos.web.elte.hu/szakdoli/szakdoli\_how\_to.docx. [Hozzáférés dátuma: 19 11 2019]. |
| [6] | „BSc szakdolgozat követelmények,” [Online]. Available: https://www.inf.elte.hu/dstore/document/257/PTI\_BSc\_szakdoli\_uj.pdf. [Hozzáférés dátuma: 19 11 2019]. |
| [7] | „ELTE címer,” [Online]. Available: https://www.inf.elte.hu/dstore/document/203/cimerek.doc. [Hozzáférés dátuma: 19 11 2019]. |
| [8] | „Szakdolgozat leadáshoz szükséges dokumentumok,” [Online]. Available: https://www.inf.elte.hu/content/adatlapok-formanyomtatvanyok.t.1052?m=129. [Hozzáférés dátuma: 19 11 2019]. |
| [9] | C. Máté, „ELTE szakdolgozat template (Github),” [Online]. Available: https://github.com/mcserep/elteikthesis. [Hozzáférés dátuma: 10 07 2019]. |