Szegedi Tudományegyetem

Informatikai Intézet

SZAKDOLGOZAT

Szelepcsényi Dávid

2025

Szegedi Tudományegyetem

Informatikai Intézet

**Kézzel rajzolt E-K diagram digitalizálása**

Szakdolgozat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Készítette: |  | Témavezető: |  |
|  | Szelepcsényi Dávid |  | Dr. Kardos Péter |  |
|  | Programtervező Informatikus BSc szakos hallgató |  | adjunktus |  |

Szeged

2025

## Feladatkiírás

A feladat egy olyan program készítése, ami egy raszteres képet kap, és azon végzett műveletekkel meghatározza a diagram elemeit, majd egy XML nyelvű, vektoros rajzolóprogram fájlformátumába átkonvertálni. Ez az OpenCV Python csomag segítségével érjük el. A program képes kell, hogy legyen felismerni az egyed kapcsolat diagram elemeit, azok közt a kapcsolatokat meghatározni. A program, a PyTesseract csomag segítségével képes a szövegek felismerésére. A program futása fél-automatikus, a felhasználó képes átírni a talált szöveget, illetve a felhasználó határozza meg, hogy melyik elem számít gyenge elemnek vagy gyenge kapcsolatnak.

## Tartalmi összefoglaló

* A téma megnevezése:

E-K diagram raszteres képének beolvasása, majd a diagrammot leíró, vektorgrafikus diagram szerkesztő program által szerkeszthető fájl készítése.

* A megadott feladat megfogalmazása:

Egy olyan program készítése, amely képes E-K diagram elemeit, és a köztük lévő kapcsolatokat felismerni egy, a programnak megadott kép alapján, majd az így összegyűjtött információ alapján egy XML forráskódú fájlt állít elő

* A megoldási mód:

Python programozási nyelv használatával valósítottam meg a feladatot. Egy kép megadása után, amennyiben az megfelel bizonyos előkövetelményeknek, a program elvéges a képen előkészítési lépéseket. Majd elsőnek leolvassa az elemek adatait, ezek segítségével, illetve az eredeti kép alapján meghatározza a köztük lévő kapcsolatokat, azokat eltárolja. Ezen adatok alapján, illetve a felhasználó által megadott adatok alapján előállít egy drawio kiterjedésű fájlt.

* Alkalmazott eszközök, módszerek:

Az implementációhoz Python 3.12.3.-as verziót, és OpenCV, kép kezelésre kitalált Python könyvtárat használta. A vizuális felülethez PyQt6 grafikus csomagot, a szövegek felismerésére PyTesseract függvény könyvtáratat használta. Az XML fájlok a Python beépített, erre a célra szolgáló xml csomag segítségével készültek. A szükséges számítások elvégzéséhez a program NumPy függvény könyvtárat, és a beépített math csomagokat alkalmaz.

* Elért eredmények:

A program képes felismerni az E-K diagram különböző elemeit, és az azok között fennálló kapcsolatokat. A szövegfelismerés hibája esetén a felhasználó képes javítani a hibákat, illetve képes meghatározni gyenge elemeket, kapcsolatokat, valamint megadni a diagram kulcsait.

* Kulcsszavak:

E-K diagram, Python, OpenCV, XML, PyTesseract,

## Tartalomjegyzék

[Feladatkiírás 2](#_Toc80443487)

[Tartalmi összefoglaló 3](#_Toc80443488)

[Tartalomjegyzék 4](#_Toc80443489)

[Bevezetés 5](#_Toc80443490)

[1. Téma háttere 5](#_Toc80443491)

[1.1. Digitális képfeldolgozá 5](#_Toc80443492)

[1.2. Diagrammok 5](#_Toc80443493)

[1.2.1. Diagramok, mik azok, miért van rájuk szükség 5](#_Toc80443498)

[1.2.2. E - K diagram 5](#_Toc80443499)

[2.Használt eszközök 6](#_Toc80443505)

[2.1 Python. 6](#_Toc80443506)

[2.2. OpenCV 6](#_Toc80443507)

[2.3. Numpy 6](#_Toc80443508)

[2.4. PyTesseract 6](#_Toc80443507)

[2.5. PyQt6 6](#_Toc80443508)

[2.6. Drawio 6](#_Toc80443508)

[3. Diagram felismerés és Xml generálás 5](#_Toc80443495)

[3.1. Adattárolás 5](#_Toc80443496)

[3.2. Előfeldolgozás 5](#_Toc80443497)

[3.2.1. Képek tisztítása 5](#_Toc80443498)

[3.2.2. Hibajavítás 5](#_Toc80443499)

[3.3. Alakzatfelismerés 5](#_Toc80443497)

[3.3.1. Első alakzat felismerés 5](#_Toc80443498)

[3.3.2. Hibaellenőrzés 5](#_Toc80443499)

[3.4. Kapcsolatok keresése 5](#_Toc80443497)

[3.4.1. Elemek eltávolítás és képtisztítás 5](#_Toc80443498)

[3.4.2. Vonalak keresése 5](#_Toc80443499)

[3.4.3. Vonalak elemekhez kapcsolása 5](#_Toc80443498)

[3.4.4. Validáció 5](#_Toc80443499)

[3.4.5. Komplex vonalak 5](#_Toc80443498)

[3.4.6. Nyilak definiálása 5](#_Toc80443499)

[3.5. XML fájl készítése 5](#_Toc80443497)

[4. Vizuális felület és szövegfelismerés 5](#_Toc80443501)

[4.1 A vizuális felület alapjai. 5](#_Toc80443502)

[4.2.Szövegkeresés 5](#_Toc80443503)

[4.3 Gyenge elemek és kulcsok. 5](#_Toc80443504)

[5. Eredmények és hibák 5](#_Toc80443501)

[5.1 Alkalmatás használata. 5](#_Toc80443502)

[5.1.1. Főablak és szabályok 5](#_Toc80443498)

[5.1.2. Szövegfelismerő ablak és felhasználói javítás 5](#_Toc80443499)

[5.2.Elért eredmények 5](#_Toc80443503)

[5.3.Hibák és megoldások 5](#_Toc80443503)

[6. Összegzés 5](#_Toc80443501)

[Irodalomjegyzék 7](#_Toc80443509)

[Nyilatkozat 8](#_Toc80443510)

[Köszönetnyilvánítás 9](#_Toc80443511)

# BEVEZETÉS

Miért fontosak a diagramok? Programozóként az első lépes a kódoláshoz, vagy egy projekthez, egy terv kialakítása. Itt jönnek elő a diagramok. Használatuk segít vizualizálni a folyamat elemeit, azokat kialakítani, megtervezni, és javítani. A diagrammok arra is szolgálnak, hogy egyszerű leírást adjanak akár a velünk dolgozóknak, akár olyanoknak, akik nincsenek tisztában az egész eddigi munkafolyamattal. Természetesen ehhez kell egy bizonyos megértés a diagram felépítéséhez, annak elemeihez és azok jelentéséhez. Mielőtt elkezdnénk készíteni egy diagramot, fennállhat a kérdés, hogy ezt milyen eszközzel is tegyük. Csináljuk kézzel esetleg, vagy erre kifejlesztett rajzoló programmal? Ha rajzoló programmal készítjük, onnan kinyomtatni egyszerű már, illetve a későbbi szerkesztés is sokkal könnyebb, de fordítva nem ez a helyzet. Szakdolgozatom célja ennek a folyamatnak a megkönnyítése, lehetővé tevése.

Én az egyed – kapcsolat diagrammon dolgoztam, aminek a főbb célja az az elkészítéséhez adatbázis modelljének egy vizuális ábrát adni. A program ennek a diagram típusnak az elemeinek felismerésére lett beállítva. Amennyiben bizonyos feltételek teljesülnek, a program egy egyszerű Python grafikus felületen keresztül képes a diagramról készült képet feldolgozni, azt vektorgrafikus rajzoló és szerkesztő program által kezelhető formába hozni. Ebben a felhasználónak is szerepe van, bizonyos beállításokat neki kell megtennie.

A dolgozatom első részében szeretnék egy bevezetést nyújtani a téma hátterébe, a digitális képfeldolgozásba és a E-K diagram felépítésébe. Ezek szükségesek ahhoz, hogy a témát megértsük. Ezt követően bemutatom a használt eszközöket, amikre szükségem volt a program megírásához, így meglesz minden háttértudás a folyamat megértéséhez. A harmadik részben magáról a kódról beszélnék, azon belül is a képfeldolgozásról és annak segédeszközeiről. Itt elsőnek az adattárolás, majd a folyamat maga következik.

A képfeldolgozás három főbb részből áll. Előfeldolgozás, alakzatfelismerés és a kapcsolatkeresés. Végül az így megszerzett adatokból felépül a vektorgrafikus kép. A negyedik rész szól a felhasználó által kezelt részről. Ez magába foglalja a szövegfelismerést, a speciális kapcsolatok és elemek beállítását, illetve itt szeretném magának a vizuális felületnek a felépítését, működését ismertetni. Az ötödik rész az használati útmutatóról, az eredményekről és a hibákról szól. Érdekesnek tartom még azt is, hogy milyen hibák jöttek elő a szakdolgozatom kidolgozása alatt. Szerintem az eredmények megértése ezek nélkül nem teljes. Végül a hatodik fejezeben összefoglalnám a teljes folyamatot.

**1 A TÉMA HÁTTERE**

Ebben a fejezetben a szakdolgozatom témájának két háttérterületébe, a digitális képfeldolgozásba és a diagrammok, azon belül is az egyed – kapcsolat diagram megértésébe nyújtok betekintést. A digitális képfeldolgozással kezdünk, ami egy elég tágas téma és vizuális és absztrakt jellege miatt igen változatos. Majd a diagrammok szükségessége, szerepe és felépítésük majd maga az E-K diagram ismertetése következik.

* 1. **Digitális képfeldolgozás**

Ha látunk egy képet magunk elött, az agyunk megpróbálja azt értelmezni. Mit ábrázol, milyen színek jelennek meg benne, az egyes részeket külön és akár egy nagyobb egész részeként is felismerni, mondhatni információt nyerni a képből. A digitális képfeldolgozás ennek a folyamatnak a számítógép által végzett megfelelője.

A folyamat első lépése az előfeldolgozás. Miért van erre szükség? Ameddig az emberek képesek logikai következtetéseket vonni, elemetek akár rossz minőség ellenében is értelmezni, ugyan ez nehezen megfogalmazható kód formájában. Kiindulási pontunk egy képpel, avagy másnéven képmátrix-val, egy kettő vagy több dimenziós tömbbel kezdődik, amin különböző műveleteket végzünk. Ezeknek a célja a kép javítása, mondjuk zajszűrés, vagy a kontrasztok kiegyenlítése, hogy a kép értelmezhetőbb legyen, a folyamat eredménye ne legyen minőség béli hibák miatt rossz.  
 Következő lépés maga az elemzés. Attól függően, hogy mi a cél, különböző módon haladhatunk tovább. Képesek vagyunk a kép elemeit csoportosítani valamilyen feltétel alapján (pl.: szín vagy forma), vagy éleket, összefüggő elemeket keresni, akár részekre osztani, szegmentálni a kiindulási képet. Ezek az eljárások kombinálhatóak, sorban elvégezhetőek.  
 Ezek implementációjára több példa is van, több programozási nyelvnek megvan a saját modulja hozzá, mindegyik saját előnyeivel és hátrányaival. Én az OpenCV csomagot használtam, hogy elkészítsem a programomat.

* 1. **Diagrammok**

Most, hogy megismerkedtünk a digitális képfeldolgozással, térjünk rá a másik témára amire épül a szakdolgozatom, a diagrammokra. Itt szeretnék adni egy rövid ismertetőt a diagrammokról, mik azok, miért használjuk őket, valamint azon belül is magát az E – K diagrammokat, amikkel én dolgoztam.

**1.2.1. Diagrammok, mik azok, miért van rájuk szükség**

A diagrammok fő célja az adatok, információk vizuális reprezentációja, hogy az értelmezhetőbb legyen, akár olyanok, akik nincsnek otthon az adott témában is megérthessék az adott pontokat.

Vegyünk példának mondjuk egy Excel táblázatot, amiben fel van jegyezve egy cég kiadásai, és bevételei, azok lebontva részekre. Akik ezt a táblázatot készítették átlátják a sok nyers adatot, de lehet mások, akik nem jártassak ezen a területen elvesznek benne. Ha viszont van egy diagram, ami egy tömörebb, egyszerűbb képet ad a rengetek információról, akkor könnyebben értelmezhető lesz az adathalmaz.

Rengetek féle diagram típus létezik, oszlop és kőr diagrammok, vagy programozó környezetben inkább használt példák, mondjuk osztály, csomag és Egyed – kapcsolat diagrammok.

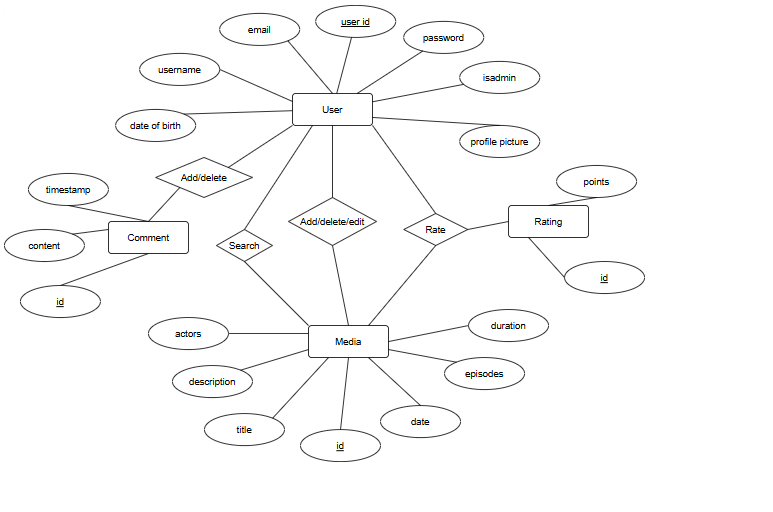
1.2.2 E – K diagram

Egyed – kapcsolat, vagy E – K diagram röviden, ennek a diagram típusnak a fő célja az adatbázisok logikai modelljének elkészítése, reprezentációja. Ez segít az adatbázisok elkészítésében, annak javításában, ha módosítani kell, annak megtervezésében. Illetve, ha meg akarjuk osztani valakivel az adatbázisunkat, egyszerűbb megosztani egy diagrammot is, hogy ne megából az adattáblákból keljen kitalálni, mi és hogyan kapcsolódik más elemekhez.

Elsőnek is, vannak *Egyedek* (entitások). Ezeket téglalappal jelöljük, és bele írjuk a nevüket. Egyedek rendelkeznek jellemzőkkel, tulajdonsággokkal. Például, egyed mondjuk egy könyve, aminek van írója, fejezetei, ezek tulajdonságok.

*Kapcsolat*, az a reláció, ami egy vagy több egyed között határoz meg valamilyen viszonyt. Ennek a jelölése egy vonal, az egyedek között, amin egy rombusz van, abba bele írva a kapcsolat neve. Amennyiben kapcsolat vissza mutat önmagára, akkor *rekurzív kapcsolatról* beszélünk, mondjuk egy könyve végén fel vannak jegyezve a források, akkor az egy rekurzív kapcsolat lehet, hiszen a források lehetnek ugyan úgy könyvek.

Mind, egyedeknek, mind kapcsolatoknak vannak *attribútumaik*, amik a tulajdonságok (könyve: cím, író, fejezet. Forrás: mikor lett utoljára használva az a forrás). Attribútumok jelölése egy ellipszis, amiben bele van írva a nevük. Ha ez a név alá van húzva, akkor az a *kulcs*, ami képes egyértelműen azonosítani az egyedet. Ezen felül lehetnek *összetett attribútumaink* is, olyan tulajdonságok, amik rendelkeznek további tulajdonságokkal (egy fejezetnek lehetnek paragrafusai, vagy alfejezetei).

Nem minden egyednek van önmagában értelme. Lehet, hogy nem létezik olyan adat, amely azt önmagában meghatározza az entitást, másszóval nincs kulcsa. Ezeket hívjuk *gyenge egyednek*, jelölésük egy dupla vonalas téglalap. Ilyenek esetén lennie kell a gyenge egyed és egy hozzá kapcsolható nem gyenge egyed között, egy *meghatározó kapcsolatnak*. Ennek jele az összekapcsoló vonalon lévő dupla vonalas rombusz, illetve egy nyíl a vonal végén, ami a nem gyenge entitásra mutat. 

**1.2.2.1. ábra** – E–K diagramra egy példa

Végül, habár nem olyan sokszor van rá szükség, léteznek *specializáló kapcsolatok* is. Ezeknek a szerepük a hierarchia jelölése egyedek között. Jele egy háromszög a vonalon, aminek a csúcsa a főtípus felé mutat. Példa: Egy könyve a főtípus, annak van írója, oldalszáma. Egy kisebb csoport ezen belül mondjuk egy tankönyv, ami az előbb említett attribútumok mellett rendelkezik saját, egyedi tulajdonsággal, mondjuk szómagyarázattal, de ugyan úgy könyv.

**2 Használt eszközök**

Most, hogy már ismerjük a téma hátterét, térjünk rá pontosan milyen eszközök kellettek a program elkészítéséhez.

**2.1 Python**

Miért *Python*? A személyes preferenciámat leszámítva, a Python-t egyszerűsége (egyik fő előnye az, hogy az angol nyelvre épül, és ezért könnyen értelmezhető) és sokoldalúsága miatt választottam. Mivel nem voltam benne eleinte biztos, milyen eszközökre lesz szükségem, így azt is figyelembe kellett vennem, hogy olyan programozási nyelvet válasszak, ahol nem fordul elő, hogy egy, a témához szükséges elemet, a választott nyelv nem támogat.

Rengeteg különböző területre kiterjedt mára a Python. Legyen szó chatbotok és AI programok készítéséről, grafikus felületek létrehozásáról, vagy képfeldolgozásról, ezekre mindre ad lehetőséget. Ezt programcsomagok, modulok és függvény könyvtárak formájában érhetjük el. Ezeket legtöbb esetben csak telepíteni kell, aztán importálni és kész is vagyunk, nagyobb csomagok esetén lehet bonyolultabb a folyamat. Egyetlen hátránya ezeknek, hogy egy részük nem Python-ban van leprogramozva, hanem mondjuk C vagy C++-ban, és nem lehet őket egyszerűen átírni.

Azt kell még tudnunk, hogy ez egy interpretált nyelv, azaz nincs szükség fordítóprogramra, viszont ezért cserépe a futásidő lassabb lehet a fordított nyelvekhez képest. Objektum orientált programozásra lett kitalálva, de nincs megszabva, hogy csak így lehessen kódolni benne.

Összegezve, ez egy igen rugalmas, egyszerűen tanulható és használható programozási nyelv, ami számomra tökéletes volt a szakdolgozatomhoz.

**2.2 OpenCV**

Az *OpenCV* (Open Source Computer Vision Library) egy nyílt forráskódú könyvtár, ami magába foglal több száz számítógépes látás algoritmust. [4] Eredetileg C-ben írták meg, de később áttértek C++ nyelvre. Ezt a függvény könyvtárat mind Java, mind Python környezetben tudjuk használni, nagyjából ugyan olyan módon, de vannak minimális eltérések a nyelvek sajátosságai miatt, én az utóbbit választottam.

A képek OpenCV-ben képmátrixként, egy kettő vagy több dimenziós tömbként vannak kezelve. (0,0) koordinátájú pont a bal felső sarok, első szám az adott pont x (vízszintes) tengelyen lévő koordinátája, második az y (függőleges) tengelyé. Mindegyik képpont rendelkezik *intenzitással.* Ha a kép szürke árnyalatos akkor ez egy szám 0 és 255 között, ahol 0 a fekete, 255 a fehér. Amennyiben színes képről beszélünk, akkor mindegyik színcsatornának saját intenzitása van kék, zöld, piros sorrendben, hasonló értékek között. Minél nagyobb az érték, annál világosabb a szín.

Egy számomra fontos funkció, a *szegmentálás* volt. Ez a kép részekre bontása, egy objektumot leíró képpontok csoportjának meghatározására szolgáló folyamat. [1] Több szempont szerint el lehet végezni, mondjuk egy területen lévő hasonló színek szerint, vagy, amit én is használtam, képen látható élek szerint. Ezek a részek alapján aztán további műveleteket lehet végezni, de egyesesetekben ehhez nem elég az OpenCV.

**2.3 NumPy**

A *NumPy* a Python tudományos számítsa technikai alapcsomagja. [5] Egy nyílt forráskódú, ingyenes, és egyszerűen használható programcsomag, amit segít a bonyolultabb számítások és tömb operációk gyorsabb elvégzésében.

Amennyiben OpenCV-t használunk, előbb-utóbb szükségünk lesz NumPy-ra is. Ez azért van, mert, mert a képmátrixokon végzett folyamatok időigénye igen nagy lehet más esetben, illetve segít, hogy ne kézzel keljen megírni egyes részműveleteket, elemeket. Például használatával nem kell kézzel maszkokat, egyfajta mátrixot a képműveletekhez, kézzel megírni, hanem elég, ha megadjuk a mátrix dimenzióit, és generál nekünk egyet.

## Irodalomjegyzék

1. Tanács Attila: Digitális képfeldolgozás jegyzet: <https://www.inf.u-szeged.hu/~tanacs/pyocv/>

Utolsó megtekintés dátuma: 2025. 03. 31.

1. Németh Gábor: Adatbázisok jegyzet, Egyed – kapcsolat diagram: <https://inf.u-szeged.hu/~gnemeth/adatbgyak/exe/EK_diagram/>

Utolsó megtekintés dátuma: 2025. 03. 29.

1. Python 3.12 dokumentáció: <https://docs.python.org/3.12/faq/general.html#what-is-python>

Utolsó megtekintés dátuma: 2025. 03. 30.

1. OpenCV 4.11 dokumentáció: <https://docs.opencv.org/4.11.0/>

Utolsó megtekintés dátuma: 2025.03.31.

1. NumPy dokumentáció: <https://numpy.org/doc/stable/user/index.html#user>

Utolsó megtekintés dátuma: 2025.04.01.

## Nyilatkozat

Alulírott ………………..………… szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet ……………………….. Tanszékén készítettem, …………………….…….…… diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Diplomamunka Repozitóriumában tárolja.

Dátum

Aláírás

## Köszönetnyilvánítás

Mellékletek

A.

B.

C.