



**Gépi látás**

GKNB\_INTM038

**Panoráma készítő**

**Gergye Patrik László**

RAF7BE

Győr, 2020/21/1

## Tartalomjegyzék

1	Bevezetés .....	2
2	Elméleti háttér.....	3
2.1	Mi is az a panoráma kép?.....	3
2.2	A panoráma készítés lépései leegyszerűsítve .....	3
2.3	A lehetséges problémák .....	3
2.4	SIFT .....	4
3	Tervezés és megvalósítás .....	5
3.1	Előkészületek, ötletek gyűjtése .....	5
3.2	Használt eszközök.....	5
3.3	A hagyományos megvalósítás.....	5
3.4	Az új egyszerűsített megoldás .....	8
4	Tesztelés.....	9
4.1	Bemenetek készítése és ezek hatása .....	9
4.2	A hagyományos megoldás eredményei és tapasztalatai .....	9
4.3	Az egyszerűsített megoldás eredményei és tapasztalatai.....	10
4.4	Összesítés és további fejlesztési, javítási lehetőségek .....	10
5	Felhasználói útmutatás.....	11
6	Felhasznált irodalom.....	12

## 1 Bevezetés

A tárgyra beadandó feladatként egy panoráma kép készítő program készítését választottam. A feladat célja egy olyan program elkészítése, amely képes két(vagy több) egymással átfedésben lévő kép összeolvásztására, ezáltal egy szélesebb “látószögű” képet létrehozva, mely a megnevezéséből adódóan hasonlít egy szemünk elé táruló panorámára.

Mivel szabadidőmben nagyon sokat foglalkozok a fotózás és képszerkesztés téma körével, ezért nagyon felkeltette érdeklődésemet a panoráma képek készítése. A modern eszközök, köztük okos telefonok, tabletok, fényképező gépek nagy része rendelkezik panoráma készítési funkcióval. Erre eleinte nem figyeltem fel, mert nem láttam túl nagy hasznát a minden napokban, azonban a program tesztelése során jöttem rá, hogy gyakorlatilag ez a lehetőség helyettesíteni képes bizonyos esetekben a nagyon széles látószögű lencsék használatát, ezáltal nagyon szűk területen is képesek vagyunk megfelelő mennyiségű részletet prezentálni.

Az elkészítés során a képzés más tárgyain tanult ismereteket is igyekeztem felhasználni ilyen például a szoftver életciklus modell, mely tiszta iránymutatást ad a lépések sorrendjét illetően, valamint ilyen például a Python nyelv is, melyet az implementálás során alkalmaztam kisebb nagyobb sikerekkel. A feladat megoldása során nem csak egy, hanem több panoráma készítési módszert is alkalmaztam, így az elkészítés és a tesztelés során is igyekeztem a különböző módszerek előnyeit, hátrányait, különbségeit elemezni, kifejteni annak érdekében, hogy tisztább képet kapunk a módszerek fejlődését, használhatóságát illetően.

## 2 Elméleti háttér

### 2.1 *Mi is az a panoráma kép?*

Amikor panoráma képekről beszélünk legtöbb esetben egy 3 dimenziós tér 2 dimenzióban való megjelenítésére gondolunk. Oldalirányú mozgással egybekötött képkészítéssel kezdődik a folyamat, melyet a képek valamelyen közös tulajdonsága alapján történő összefűzése követ. A képek készítése során akár 360 fokos fordulat is történhet, ezáltal a teljes körülöttünk lévő helyszín egy 2 dimenziós legalább 3:1 arányú vagy szélesebb képen jelenik meg. [1]

### 2.2 *A panoráma készítés lépései leegyszerűsítve*

A képek elkészítése után szükség lesz valamelyen algoritmus használatára, ami segít a képek pontos elhelyezésében. A képek elemzésre kerülnek, majd a különböző „kulcs tulajdonságaik” összegyűjtésre kerülnek. Ezek a tulajdonságok, pontok segítenek eldönteni, hogy a képek melyik részei lehetnek fedésben és, hogy valójában melyik képet érdemes a másikra ráfedni. A pontok összehasonlításakor lehetőség van egy előre definiált modell megadására is, ami nagyban segítheti a végeredmény hatékonyságát. A képek kulcspontjainak összehasonlításakor kiderül, hogy milyen a kép homográfiája, azaz, hogy mekkora az átfedés a két kép között. Ha nincs elég közös pont, akkor sok algoritmus nem képes megvalósítani az összeillesztést.

Ezt követően történik a képek megfelelő elhelyezése, úgy, hogy a lehető legjobb fedésben legyenek ezáltal az egységesség látszatát keltve, valamint a perspektívák megváltoztatása annak érdekében, hogy a végeredmény élethűbb legyen. Az összeillesztést követően lehetséges, hogy a képek nem tökéletes készítése és a perspektíva megváltoztatása miatt fekete rések keletkeznek, ezek eltüntetése azonban nem minden esetben egyszerű feladat. Az egyik lehetőség a kép további torzítása, ami látvány szempontjából nem túl előnyös, vagy a képek széleinél levágása, mely ugyan a képek méretéből elvezet, viszont nem okoz torzítást.

### 2.3 *A lehetséges problémák*

Mivel az eltérő algoritmusok a különböző bemeneteket nagyon változatosan kezelik, figyelni kell a képi tulajdonságokra.

Két kép expozícióját nagyon nehéz egyeztetni fényképezéskor, így ha ez nem kerül utólagos korrigálásra, látható nyoma maradhat a végeredményben elkészült kép fényességi szintjén. Egy másik jellemző probléma, amikor egy adott alanyra fókuszálunk és a mélységélesség következtében a háttér elmosódik, és az elmosódott háttérben mozgás történik. Ebben az esetben a kép ugyanazon területén aminek metszésben kellene lennie nem feltétlen fog megtörténni a várt kimenetel, mivel az egyik képen lehet, hogy rajt van a háttérben mozgó alany, a másikon viszont nincs.

További kamera beállításokból, mint például a záridő is származhatnak kellemetlenségek. Ha nincs megfelelően nagy záridő állítva, a kéz remegése, vagy bármilyen apró mozgás elmosódásokat okozhat a képen.

Érdemes még megemlíteni a lencse használatot is, mivel a szélesebb látószögű lenesék szélei mentén keletkezett torzítás nagyban megzavarhatja az algoritmusok működését.

Ezen problémák elkerülése érdekében nagyon fontos figyelni a megfelelő képi beállításokra, eszközhasználatra, valamint arra, hogy a készített bemeneti képek minimum 15-30%-a átfedésben legyen, hogy optimális végeredményt kaphassunk. [2]

## 2.4 SIFT

Az általab használt megoldások alapjakként a SIFT (Scale-invariant feature transform) algoritmus szolgál, mely egy nagyon elterjedt megoldás különböző képi tulajdonságok detektálására, elemzésére, feldolgozására. Felhasználási területei közé tartozik például az objektum felismerés, kép összeillesztés, 3 dimenziós modellezés, különböző mozdulatok észlelése és még sok más.

Jelen esetben főleg a detektálási és kép összeillesztési funkciókat használtam.

A SIFT algoritmus sok másik algoritmust használ, egyesít, ezáltal elérve kiemelkedő eredményeket és hatékonysságot. Az élek és a képen lévő foltok detektálására többféle algoritmus is elérhető. Ilyen például a LoG (Laplacian of Gaussian), ami helyett ennek megközelítőleges változata a DoG (Difference of Gaussian) került felhasználásra a SIFT algoritmusban. Ezen algoritmusok lényege a képek méretarányos terének megváltozásakor történő elmosódás vizsgálata. Ez a folyamat a kép különféle oktávjaira vonatkozik a Gauss-piramisban. A DoG megtalálása után az adott képeken a méretarányos térben kerülnek megkeresésre a szélsőértékek. A szélsőértékek jelen esetben potenciális kulcsPontokat jelentenek, melyek összehasonlításra kerülhetnek és később alapot biztosíthatnak az összeillesztési folyamathoz.

Taylor sorozatú méretarányos tér kiterjesztésével sor kerül a potenciális szélsőértékek pontosabb meghatározására, és ha ezek a szélsőértékek egy előre meghatározott kontraszt küszöbértéket nem érnek el, automatikusan elutasításra kerülnek. Mivel a DoG különösen érzékeny az élekre, így azok pontjai is eltávolításra kerülnek.

Ezt követően létrehozásra kerül a kulcsPontok alapján egy orientációs hisztogram, valamint szomszédsági mátrixok a kulcsPontok körül.

Az algoritmus egyik legfontosabb része a kulcsPontok összehasonlítása. Ez a folyamat a szomszédsági mátrixok összehasonlításával történik, ezen belül is a legközelebbi szomszédok összehasonlításával. Vannak viszont bizonyos esetek, amikor a legközelebbi szomszédok és az egyel távolabbi szomszédok nagyon közel vannak egymáshoz, zajosodás, vagy egyéb képi hibák miatt. Ilyen esetben a korábban említett küszöbértékekhez hasonlóan egy előre meghatározott érték dönti el az adott pontok megfelelőségét.

## 3 Tervezés és megvalósítás

### 3.1 Előkészületek, ötletek gyűjtése

Miután elfogadásra került az általam választott téma, elkezdtem megoldási lehetőségeket keresni a problémára. Hamar egyértelművé vált, hogy nincs olyan sok választási lehetőség, így a létező opciókat kezdtem tanulmányozni. A különböző videók és tudományos cikkek sokat segítettek a téma megértésében, hogy mélyebb ismereteket is szerezhessek a panoráma képek készítésével és a gépi látással kapcsolatban.

A szükséges eszközöket feltelepítettem, majd a teszteléshez előre készítettem néhány képet, hogy a későbbi folyamatok simábban menjenek.

### 3.2 Használt eszközök

A program Python nyelven íródott, ezen belül is a Python 3.9 verzióval, mely támogat minden olyan kiegészítő csomagot, melyet használni akartam a készítés során. [3]

A kód írásához a PyCharm fejlesztői környezetet használtam párhuzamosan Windows 10 és MacOS rendszereken, ezzel biztosítva, hogy a rendszerek között megfelelően fut a program. A PyCharm lehetőséget biztosított arra, hogy a kívánt kiegészítő csomagokat a fejlesztőkörnyezetből telepítsem, ezzel elkerülve az esetleges telepítési hibákat amik a parancssori felületen gyakran előfordulnak. [4]

A használt csomagok közül érdemes megemlíteni az os csomagot, mely lehetőséget biztosít arra, hogy a fájlokat a fájlrendszerből be tudja olvasni a program. A központi csomag azonban az opencv-python volt, ami egyszerű, azonban nagyon sokrétű gépi látás eszköztárával biztosítja, hogy a kívánt eredményt elérjük. Az opencv csomag a numpy csomagra épül, ami különböző matematikai eszközöket biztosít, melyek nagyon fontosak a képfeldolgozás területén is. [5][6]

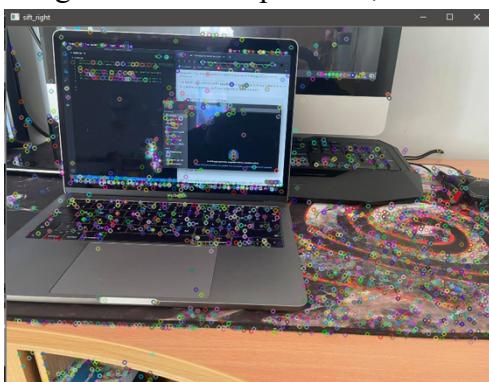
Két különböző programkódot is teszteltem, azért mert az opencv eltérő verziói eltérő eszközöket biztosítanak a feldolgozandó témára, ennek a legnagyobb oka bizonyos licensek (főleg a SIFT algoritmussal kapcsolatban) időközbeni megváltozása. A használt verziók a 4.4.0.46, illetve a jóval régebbi 3.4, valamint a numpy verziói közül is egy kicsit régebbi 1.19.4-es verziót használtam.

### 3.3 A hagyományos megvalósítás

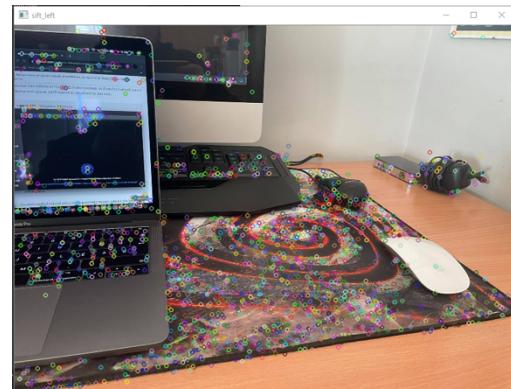
A hagyományos megoldás az opencv régebbi verziójával került megvalósításra, amiben még kisebb lépésekben kellett minden elvégezni.

Első lépésként a képek beolvasására, 40%-os átméretezésre, valamint szürkeárnyalatossá tételere került sor.

Ezt követte a SIFT algoritmussal való elemzés és számítás. Ebben a lépésben az algoritmus minden képen megkereste a kulcspontokat, ezek az alábbi képeken láthatók:



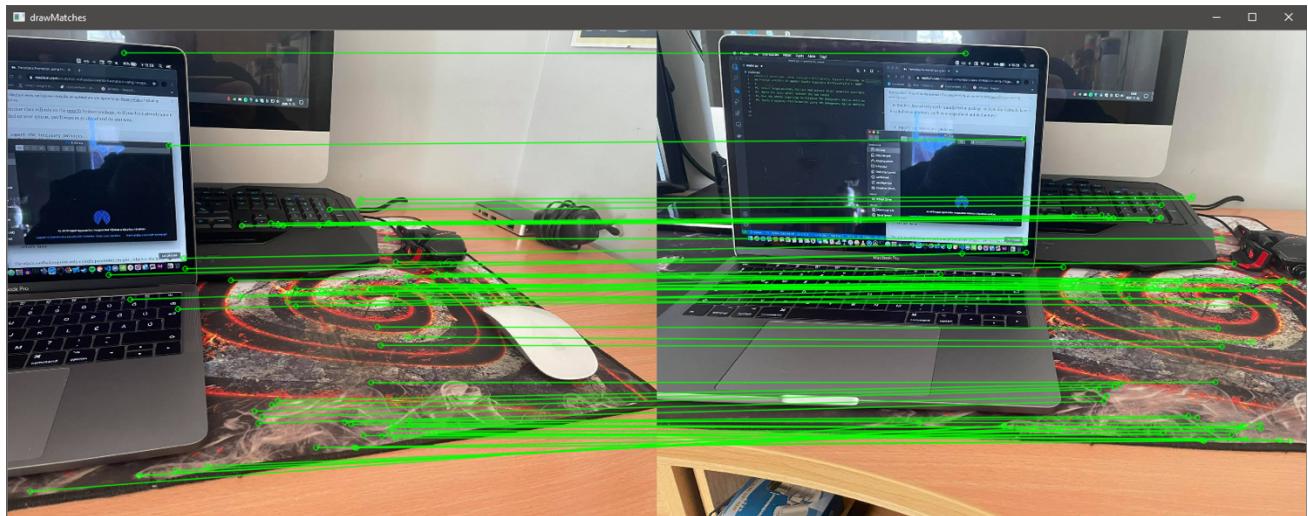
1. ábra: KulcsPontok



2. ábra: kulcsPontok

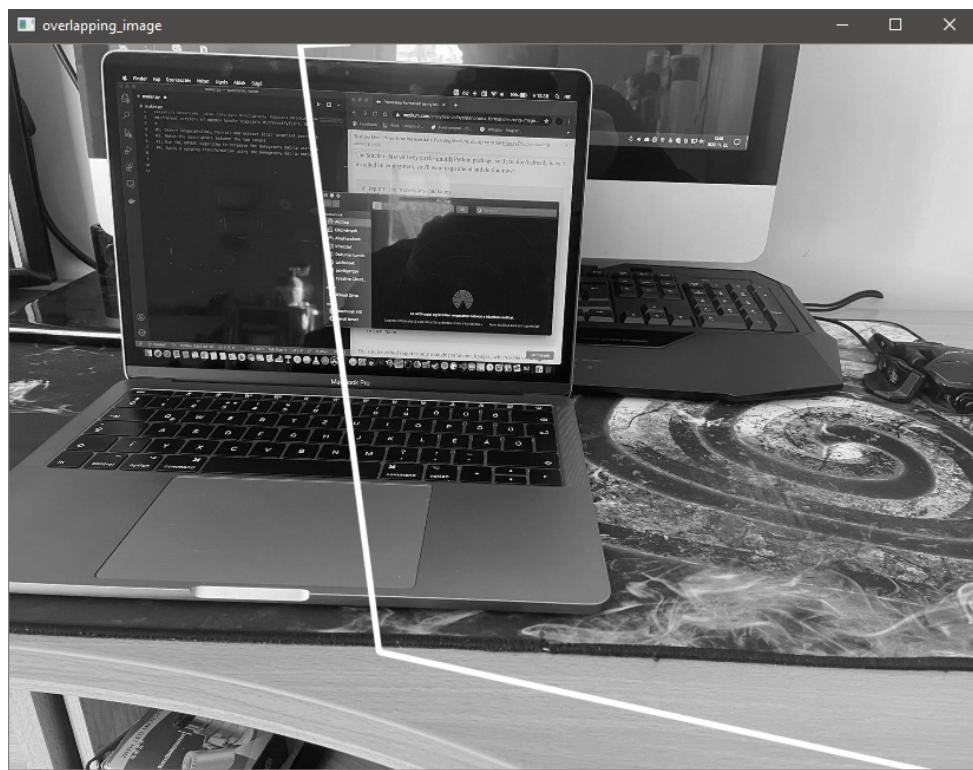
A kulcspontok megtalálását követően, ezek összehasonlításra kerülnek a FLANN(Fast Library for Approximate Nearest Neighbour) segítségével, ami egy nagyon hatékony könyvtár a legközelebbi szomszédok megtalálására. A legjobb egyezések az opencv brute force metódusával kerülnek felderítésre, majd a metódusban megadott mennyiségű további legjobb egyezések is meghatározásra, tárolásra kerülnek.

A program által elég jónak bizonyult egyezések kigyűjtésre kerülnek, majd a szemléltetés érdekében megjelenítésre is kerülnek vonalas összeköttetésekkel.



**3. ábra: Egyezések a képek között**

Ha a jónak bizonyult értékek száma elér egy előre meghatározott értéket, meghatározásra kerül a képek homográfiája, azaz közös metszete. Ezt követően megváltoztatásra kerül a képek perspektívája annak függvényében, hogy hogyan illenek össze legjobban. Az összeillesztési pontokon végbemegy egy élsimítási folyamat is, hogy kevésbé legyen feltűni az összeillesztés helye.



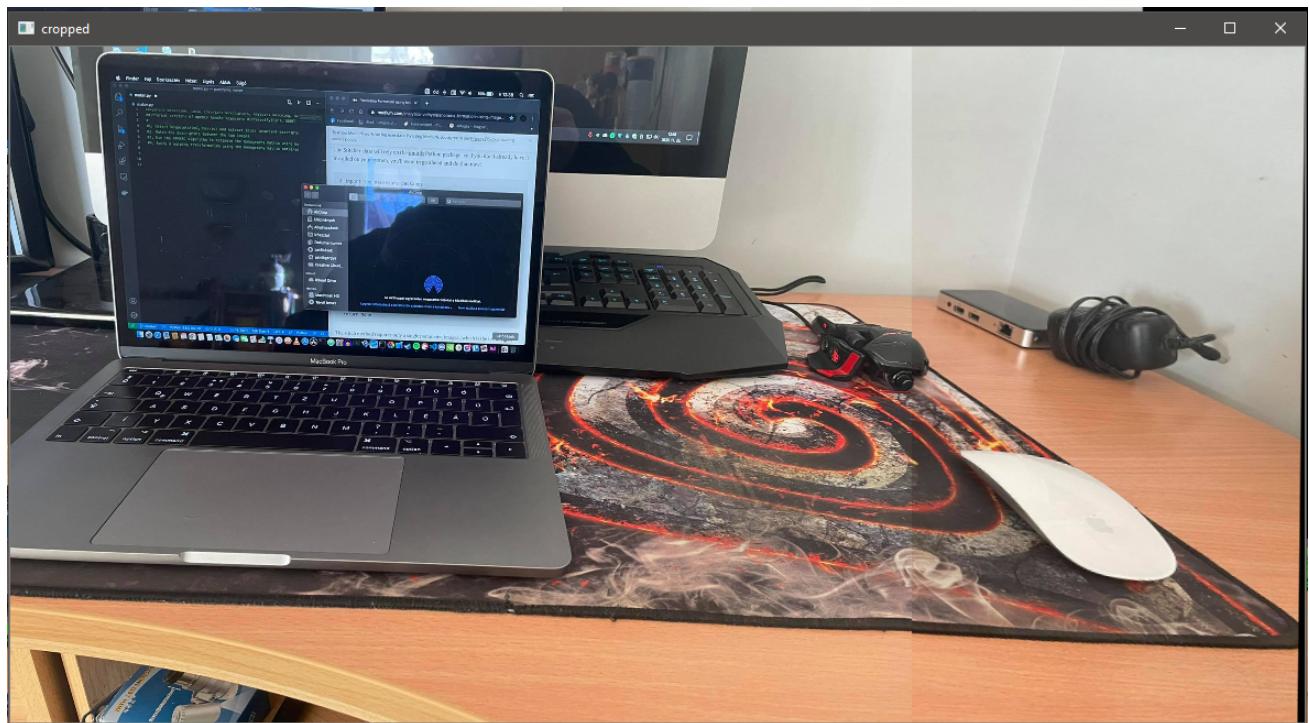
**4. ábra: A képek metszete**

Miután az összeillesztés megtörtént, a teljes kép perspektívája újra megváltoztatásra kerül, azáltal a végeredmény sokkal mutatósabb lesz, azonban megjelenhetnek fekete részek a képek transzformálása miatt.



5. ábra: A transzformálás eredménye

Az oldalakon megjelenő sávok eltüntetésére egy trim függvényt használok, mely a széleket jól levágja, azonban, ha a fekete rész alakzata nem szabályos, kisebb fekete részek továbbra is maradnak a képen.



6. ábra: Végeredmény

A kép ugyan nem tökéletes, az expozíciós eltérések nyoma jelen van, azonban kis korrigálással ez még javítható.

### 3.4 Az új egyszerűsített megoldás

A SIFT algoritmus licence szerződésének megváltozása miatt az opencv csomag működése valamilyen mértékben megváltozott. A képek összeillesztése egy sokkal egyszerűbb függvény sorozattal megvalósítható.

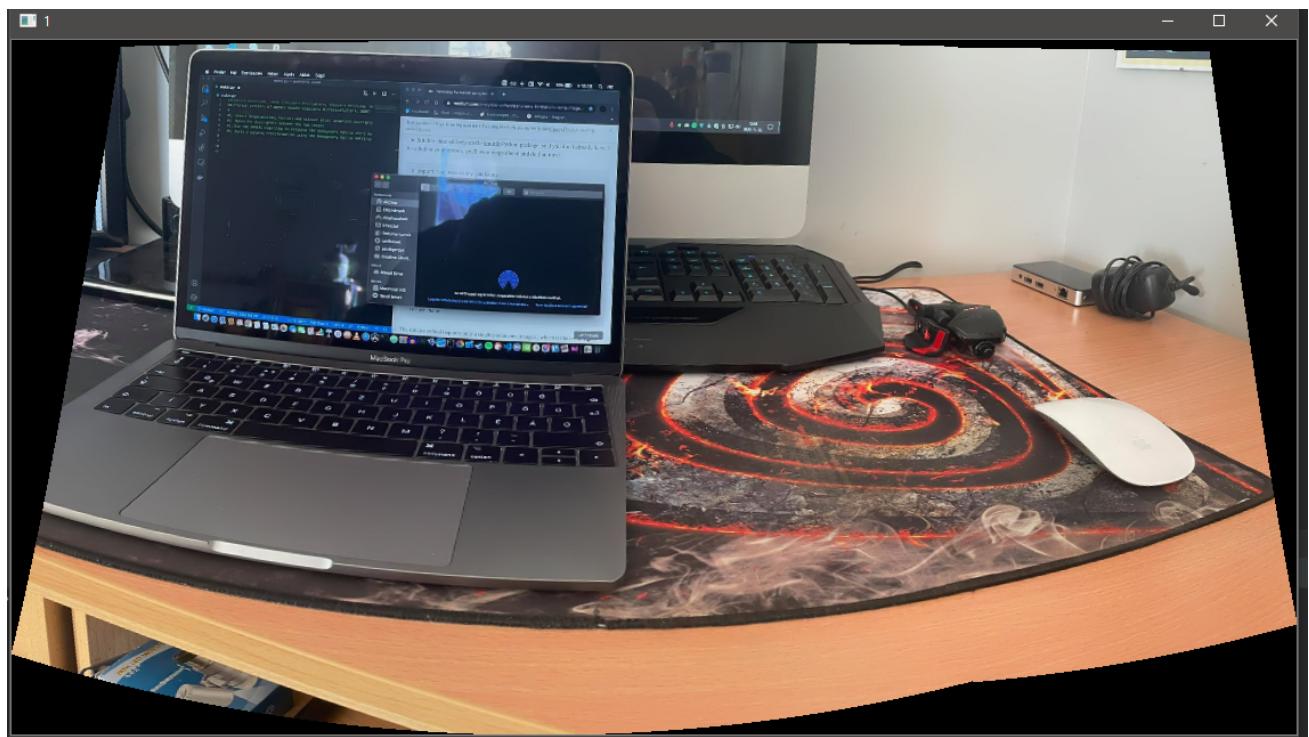
Az egyszerűsített program az os csomag segítségével beolvassa az Images mappa összes almappáját, melyek mindegyike egy-egy panoráma képhez szükséges elemeket tartalmaz.

Ezt követően két ciklussal végigmegy a mappákon, valamint a bennük lévő tartalmon.

A képek beolvasásra, 40%-os méretre újra méretezésre, valamint hozzáadásra kerülnek egy tömbhöz.

Egy létrehozott „stitcher” objektummal kerülnek a képek összeillesztésre. A előzőekben említett megoldáshoz hasonlóan itt is vizsgálatra kerül a képek megfelelősége, a kulcspontok számának megfelelő mennyisége, csak jelen esetben az opencv beépített függvénye minden automatikusan elvégez.

Ezt követően a program kiírja a sikeres és sikertelen próbálkozások számát. Fontos megemlíteni, hogy ez a megoldás közel sem olyan hatékonyságú, mint az előző, azonban erre jobban kitérek a tesztelés leírásánál.



7. ábra: Az egyszerűsített megoldás végeredménye

## 4 Tesztelés

### 4.1 Bemenetek készítése és ezek hatása

Mivel a panoráma képek készítésénél a végeredmény a legfontosabb, így elengedhetetlen a folyamatos tesztelés annak érdekében, hogy a végső kimenet valamennyire nézhető legyen.

Annak érdekében, hogy a lehető legváltozatosabb bemenetekkel próbálhassam ki a programot, saját kezűleg készítettem képeket.

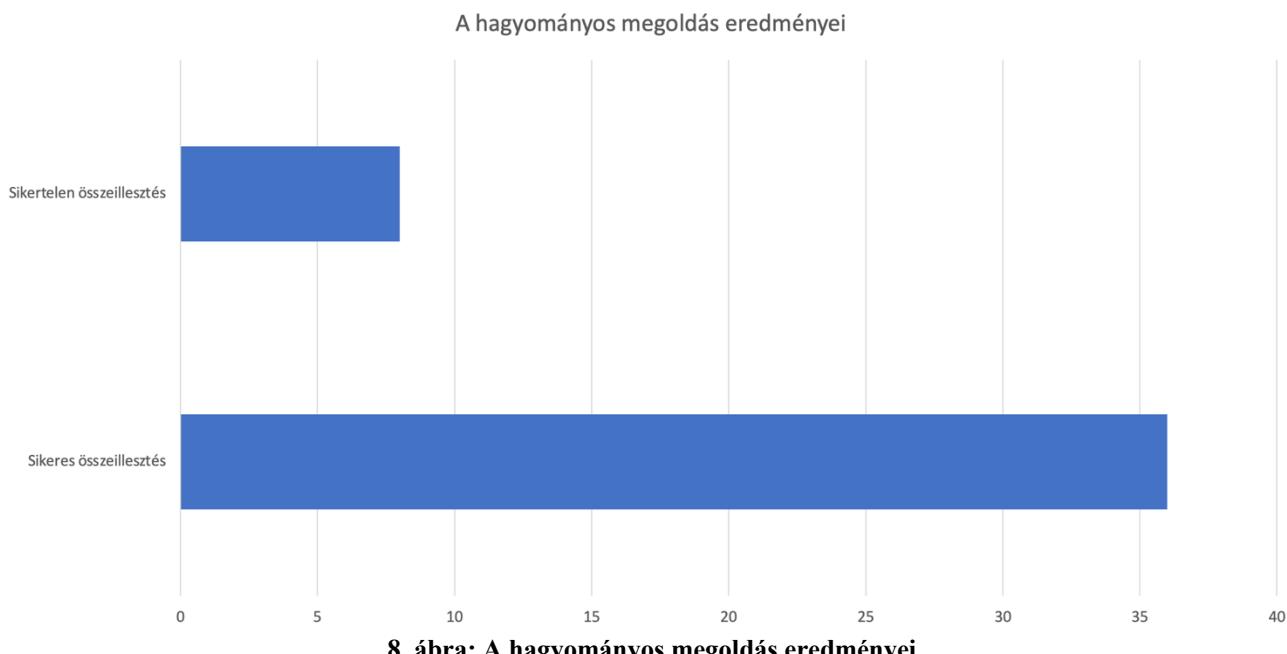
Figyeltem arra, hogy többféle eszköz képe, többféle beállítással megjelenjen a tesztképek között. A képek egy részét telefonnal készítettem automatikus beállítások mellett, ennek eredményeképpen két egymás után készített kép között is megfigyelhető volt expozíciós különbség, ami a panoráma elkészítése során egy látható elválasztási vonalat hagyott. Próbáltam közelebbi és távolabbi képeket is készíteni, hogy megnézzem mennyire lehet közel menni egy adott objektumhoz, úgy, hogy a kép még összeilleszthető legyen. A telefon mellett egy tükrök nélküli fényképezőgéppel is készítettem teszteket teljesen manuális beállítások mellett, különböző optikákkal. A manuális beállítások eredményeképpen megfigyelhető volt, hogy a fényképező képei sokkal könnyebben és jobb eredménnyel összeillesztésre kerültek, azonban jóval nagyobb mértékben csökkenteni kellett a képek méretét a magas felbontás miatt.

Teszteltem normál, ultraszéles, illetve telefotó optikákkal készített képeket is, különböző mélységélességi szintekkel. A tapasztalat az, hogy a mélységélesség kevésbé, azonban a látószög jóval nagyobb mértékben befolyásolja a végső kimeneti képet. A látószög változtatával a széleken megjelenő torzulást a programnak nehéz korrigálni, sok esetben pedig nem is képes ezt megvalósítani úgy, hogy az eredmény is kielégítő legyen.

### 4.2 A hagyományos megoldás eredményei és tapasztalatai

Összességében elmondható, hogy a hagyományos megoldás nagyobb százalékban volt képes az összeillesztésre, ennek oka lehet az is, hogy az új megoldásban kevésbé személyre szabhatók a különböző számértékek, mert ezeket a függvények automatikusan állítják be a háttérben.

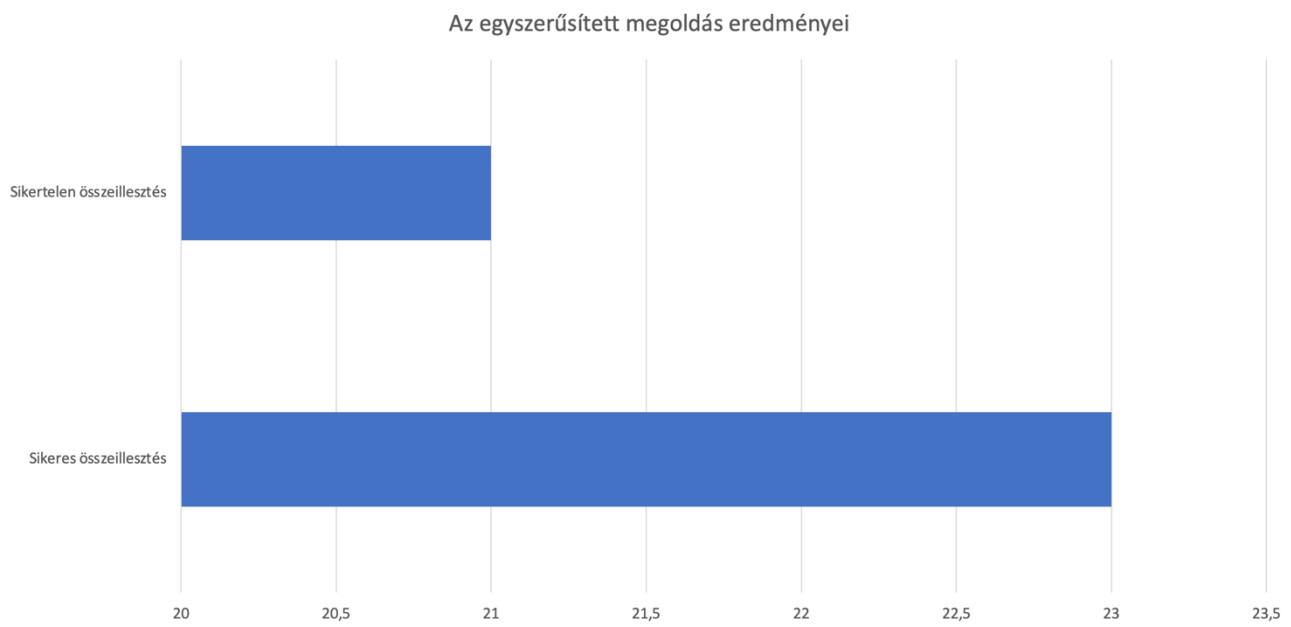
Az egyes összeillesztések azonban kevésbé voltak pontosak, illetve sokkal jobban érezhetők voltak az expozíciós, valamint látószögből eredő torzulások.



Nagy hátránya ennek a megoldásnak, hogy nehezebb a kódot kezelní, több opció van, kevésbé sem olyan átlátható, mint a modern megoldás, viszont ezzel együtt több beállítási lehetőséget biztosít, melyek a végeredményt befolyásolhatják.

#### **4.3 Az egyszerűsített megoldás eredményei és tapasztalatai**

A sok egyszerű beépített funkció miatt ez a megoldás sokkal felhasználó közelibb, nagyon egyszerű megcsinálni és végeredményben az elkészült képek összeillesztései sokkal kevésbé láthatóak. Megfigyelhető, hogy ezzel a megoldással sokkal valósághűbb az elkészült összhatás, viszont ez azért lehetséges, mert rendkívül sok torzítási folyamatot végez el a program. Ennek eredménye, hogy sokkal nagyobb fekete töltő terület marad a képeken, és sokkal szabálytalanabb alakzatokban, mint az előző megoldás esetén, így sokkal nehezebb a kép széleit levágni. Az összeillesztés sikerességének százalékos értékei is rosszabbak, mint a másik megoldásnak, viszont a lefutási sebessége és a képi eredmények ezt valamelyen szinten ellensúlyozzák.



**9. ábra: Az egyszerűsített megoldás eredményei**

#### **4.4 Összesítés és további fejlesztési, javítási lehetőségek**

Mindkét megoldásnak megvannak az előnyei, illetve hátrányai. Az opencv folyamatos fejlődése miatt néha nehéz követni, hogy mi éppen az optimális megoldás az egyes témakörökben.

Bizonyos megoldások jobb hatékonyسágot biztosítanak, mások pedig szebb képi végeredményt. A szépsége ennek az, hogy a felhasználó maga döntheti el mit preferál.

Az elkészített rövid programok nagyon távol állnak az optimális megoldástól, azonban jól prezentálják a technológia rövid idő alatti változását. Mindkét megoldást lehetett volna tovább csiszolni, például dinamikusabb, hatékonyabb szél levágással, dinamikusan változó útra méretezéssel, képjavító filterekkel. A jövőben jó lenne egy olyan „panoráma” készítő megoldás kivitelezése is, ami nem csak vízszintesen, hanem függőlegesen is képes képeket összeilleszteni, ezáltal hatalmas felbontású, akár vászonra nyomtatható képeket biztosítana.

## 5 Felhasználói útmutató

A program letöltése és futtatása nagyon egyszerű. Első lépésként szükséges a különböző függőségek telepítése, ilyen például a Python 3.9-es verziója. A Python telepítése után operációs rendszertől függően lehetőségünk van parancssori csomag telepítésre, vagy IDE (Prografejlesztői környezet) felületen kereszttüli telepítésre. Az általam javasolt megoldás a PyCharm fejlesztői környezet telepítése.

Ezt követően szükség van a program letöltésére a githubon keresztül. Ez megoldható webes felületről, vagy a parancssori `git clone https://github.com/halicssharp/panorama\_maker.git` parancs kiadásával.

Következő lépésként érdemes új projektet indítani a PyCharm környezetben, és ebbe az új projektbe beilleszteni a korábban leklónozott repository tartalmát. Annak függvényében, hogy melyik programot szeretnénk futtatni, a fájl nevét át kell írni. A PyCharm automatikusan csak a `main.py` fájlt futtatja le.

Az Images mappában találhatóak a futtatáshoz/teszteléshez használható képek. Az egyszerűsített megoldáshoz(rövidebb kód) nincs szükség változtatásra, automatikusan beolvassa a teljes Images mappa tartalmát. A másik megoldáshoz a gyökér mappába szükséges két kép bemásolása, melyek neve 1.jpg, illetve 2.jpg kell, hogy legyen, azonban a forráskódban természetesen ez személyre szabható.

Ezt követően a PyCharm környezetből futtatható minden program. Az eredmények, valamint a hagyományos program esetében a részeredmények is megjelenítésre kerülnek.

## 6 Felhasznált irodalom

- [1] Sirish Kumar: Panorama image stitching(2010)
- [2] [https://en.wikipedia.org/wiki/Image\\_stitching](https://en.wikipedia.org/wiki/Image_stitching)
- [3] <https://docs.python.org/3/whatsnew/3.9.html>
- [4] <https://www.jetbrains.com/pycharm/documentation/>
- [5] [https://docs.opencv.org/master/d6/d00/tutorial\\_py\\_root.html](https://docs.opencv.org/master/d6/d00/tutorial_py_root.html)
- [6] <https://numpy.org/doc/>