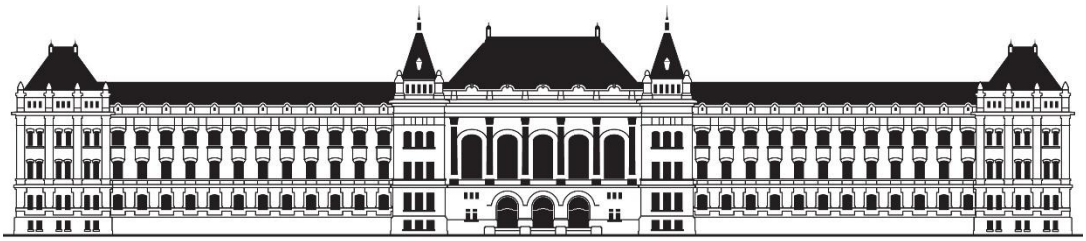


M Ű E G Y E T E M 1 7 8 2

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

Mechatronika, Optika és Gépészeti Informatika Tanszék

Kognitív robotika (BMEGEMINMKR)

# 2D to 3D

## Kognitív robotika házi feladat

Kuti Nikolett – IY8SA0

Hargitai Ádám – AGMTJO

2021.05.12.

# Tartalomjegyzék

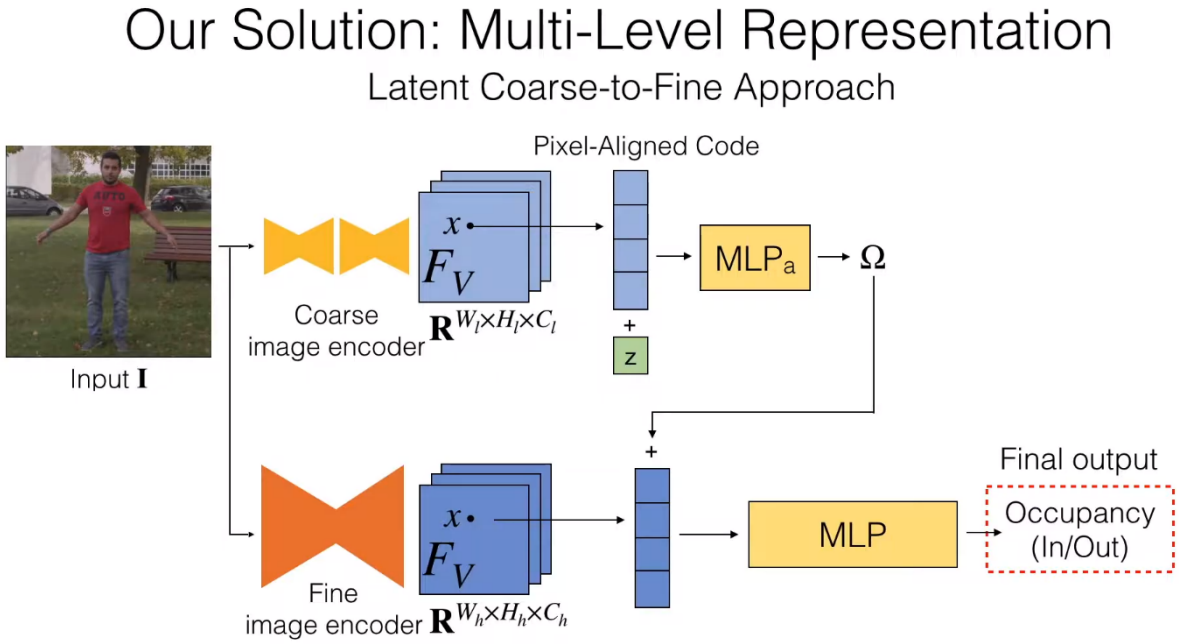
# 1. Feladat:

A feladatunk elsősorban olyan technológiák megismerése amelyek 2D architektúrából 3D architektúrát képesek előállítani, legyen ez absztrakt módon, tehát különböző bemeneti változók függvényében, vagy közvetlenül képből, esetleg képsorozatokból.

A feladat második részében szegmentálással kapcsolatos algoritmusok, illetve képjavítással kapcsolatos algoritmusokat vizsgálunk. A diplomamunkánk kapcsán mindketten napi szinten használunk szegmentálást ér áramlástani vizsgálatok során, azonban ennél mélyebben még nem vizsgáltuk meg ezen funkciókat. A célunk az, hogy képet kapjunk arról, pontosan hogyan is működhet egy ilyen szegmentációs modul és ez hogyan képes az orvostudományt segíteni.

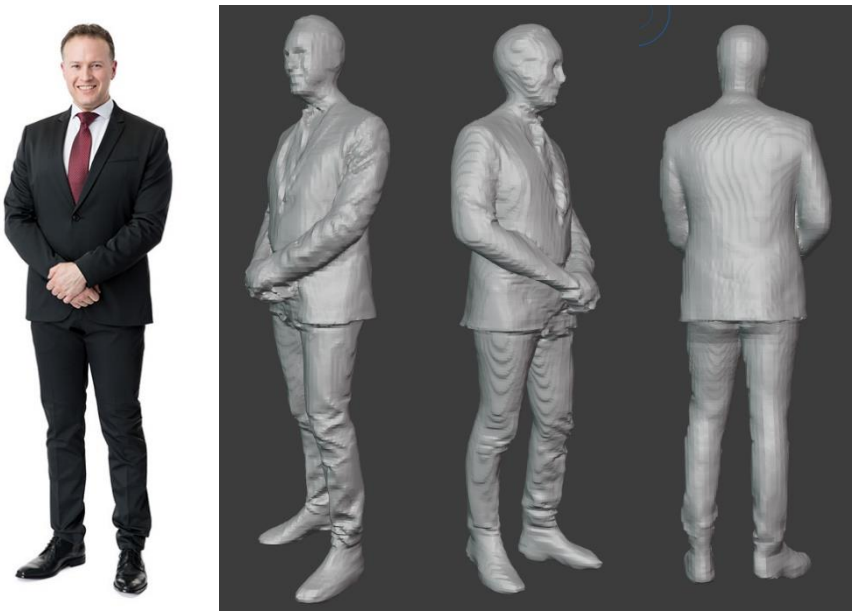
## 1.1. PifuHD-Facebook:

A 2D to 3D szoftverek keresése során legelsőnek a facebook AI csapata által fejlesztett PifuHD-t találtuk meg, mely egy 2D képből képes egy 3D modellt kialakítani. A módszer a névben is említett, a csapat által fejlesztett PIFU – Pixel-Aligned Implicit Function metódus során, melyben egy több szintű kapcsolt neurális háló dolgozik, ez látható az alsó ábrán. Amíg a felső szálon egy közelítő mesh készítése zajlik, addig az alsó szálon a felső szál végső outjából egy finomított modelt készítenek, amely felel a modell finomhangolásáért.



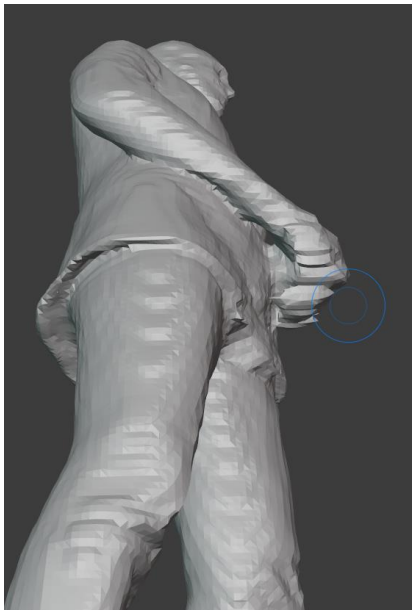

*1. ábra – PIFUHD működési diagram*

A szoftverről ingyenes DEMO is elérhető, így mi is ki tudtuk próbálni, hogy egy 2D kép alapján milyen minőségű modellt képes a kód előállítani.



2. *ábra – PIFUHD 2D képből 3D modell*

A kódon végigfuttatva a fenti ábrán bal oldalt látható képet a jobb oldalt látható modellt kapjuk. Ez a modell .obj kiterjesztéssel készül el, tehát pl. Blenderben tökéletesen megnyitható és megvizsgálható. Első ránézésre jó minőségű, a képet közelítő modellt kapunk, amely ugyan felbontásban a 2010-es évek számítógépes játékait idézi, azonban ezt Blenderben utómunkával jelentős mértékben fel lehet javítani.

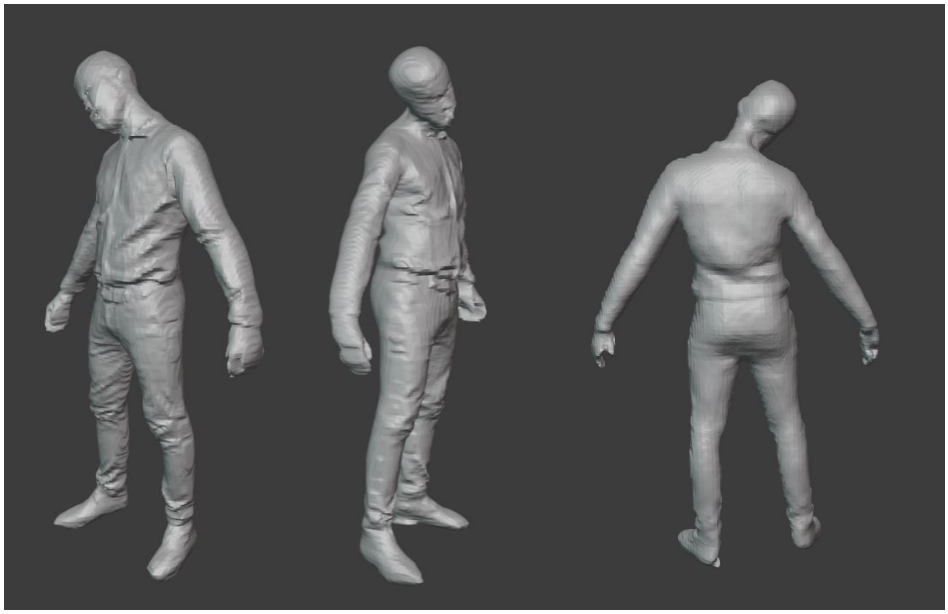


3. *ábra – A PIFUHD 3D modell kialakítása*

Az előző oldali ábrán részletesebben megnézve a modellt, azonban problémákba ütközhetünk, láthatóan a modell keze és teste egybe van olvadva, valamint a kéz kialakítása is problémásnak tűnik. Ez valószínűleg abból a zárt pózból ered melyet a képen is láthatunk, próbáljuk ki a kódot egy az animációban és modellkészítésben hasznosabb “T pózban” álló karakterrel.

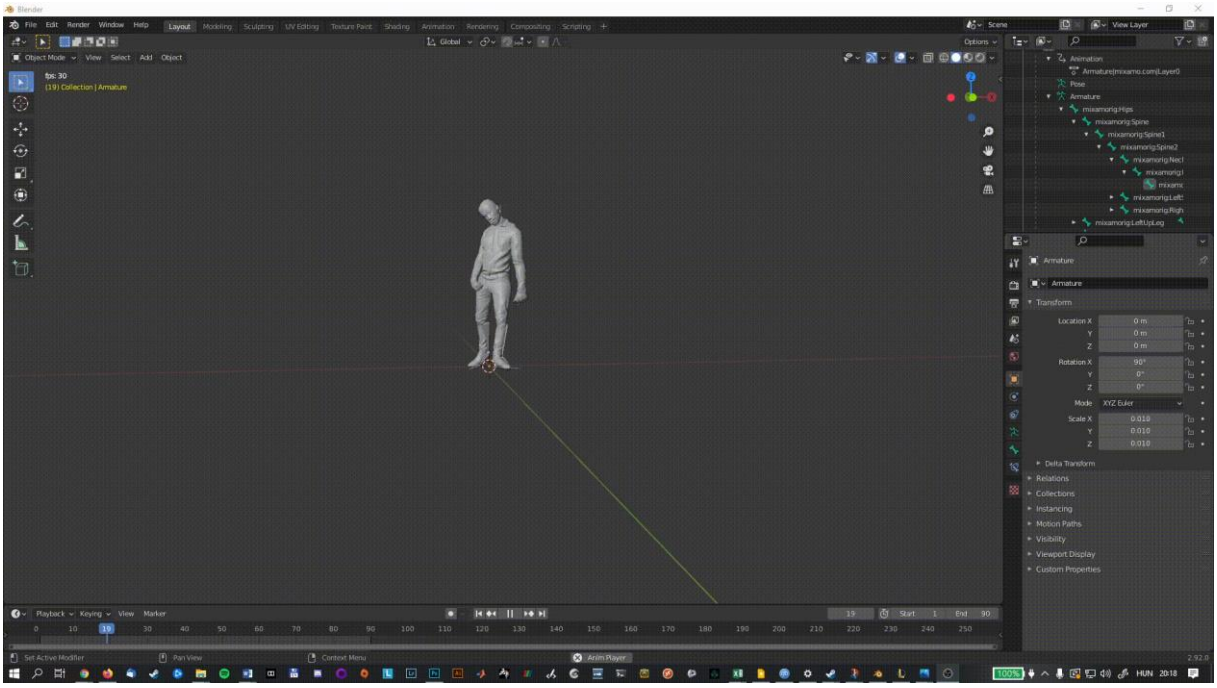
*4. ábra – T póz kép*

Egy rövid google keresés után a fenti ábra bal oldalán látható képet találtuk, amit egy kis photoshop munka után már fel is használhatunk a céljainkra. A jobb oldalon eltüntetve a vízjelet, valamint a hátteret kitörölve már készen is áll a képünk.



*5. ábra – T póz modell*

A fenti ábrán már a kód által kapott modellt láthatjuk Blenderben szintén, ebben az esetben már nincsenek összenőtt részek, a végtagok könnyen kivehetők és elkülöníthetőek. Ha az lenne a célunk, hogy ezzel a modellel valamilyen animációt készítsünk akkor akár Blenderben kézzel vagy valamilyen online felületre, pl. Mixamo-ra feltöltve könnyen skeletont is adhatunk a modellünkhöz és elkezdhetünk valamilyen egyszerű animációt kialakítani vele.



*6. ábra – Gif egy egyszerű járásról*

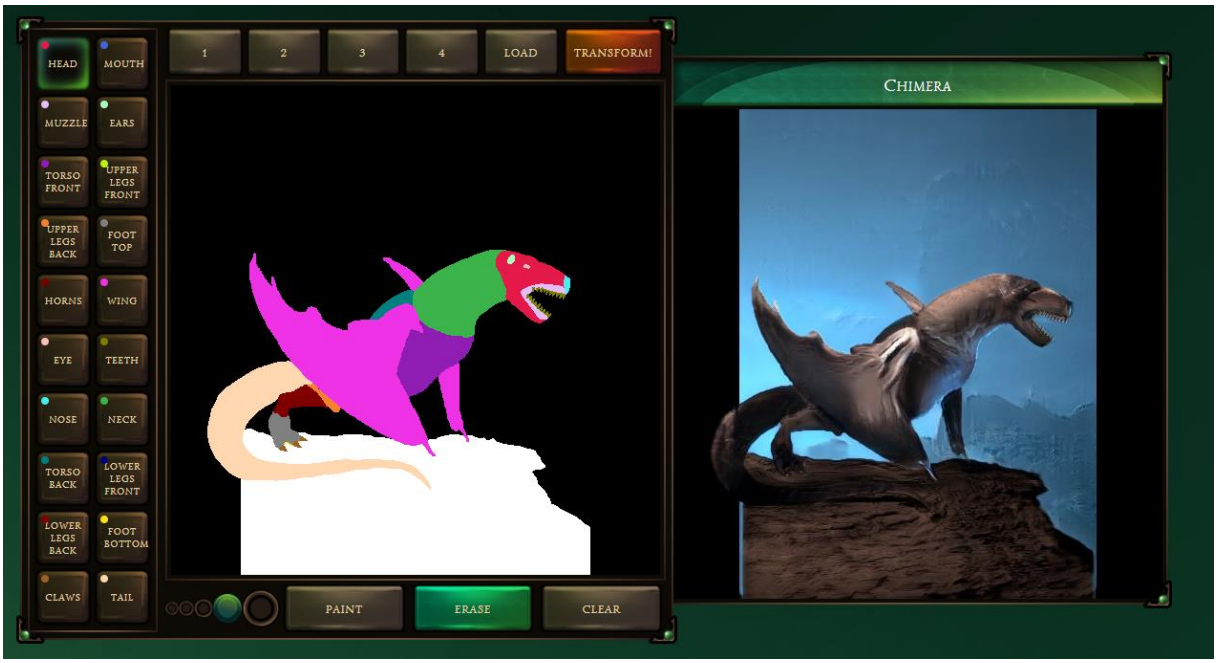
A fenti képen egy GIF formájában láthatjuk, hogy a Mixamo által kialakított csontvázhoz egy egyszerű járás animációt adva hogyan is viselkedik a modellünk. Ugyan a csontváz kissé pontatlan a pózból adódóan, így egy kicsit „zombi járást” kivitelez a karakterünk, de láthatjuk hogy milyen egyszerű így egy egyszerű képből modellt, majd utána animációt készíteni.

A modell nem annyira részletes, hogy egy komoly minőségű animációban is elfogadható legyen, azonban utómunkával kifejezetten jó minőséget lehet vele elérni, ezzel pedig egy animációs munka során a koncepcióból közelítő 3D modell kialakítás részhez szükséges időt jelentősen le lehet csökkenteni és a modellt készítő művésznek csupán finomhangolnia kell a modellt, ahelyett hogy teljesen nulláról blenderben szobrászati eszközökkel kialakítaná azt.

## 1.2. Chimera Painter-Google:

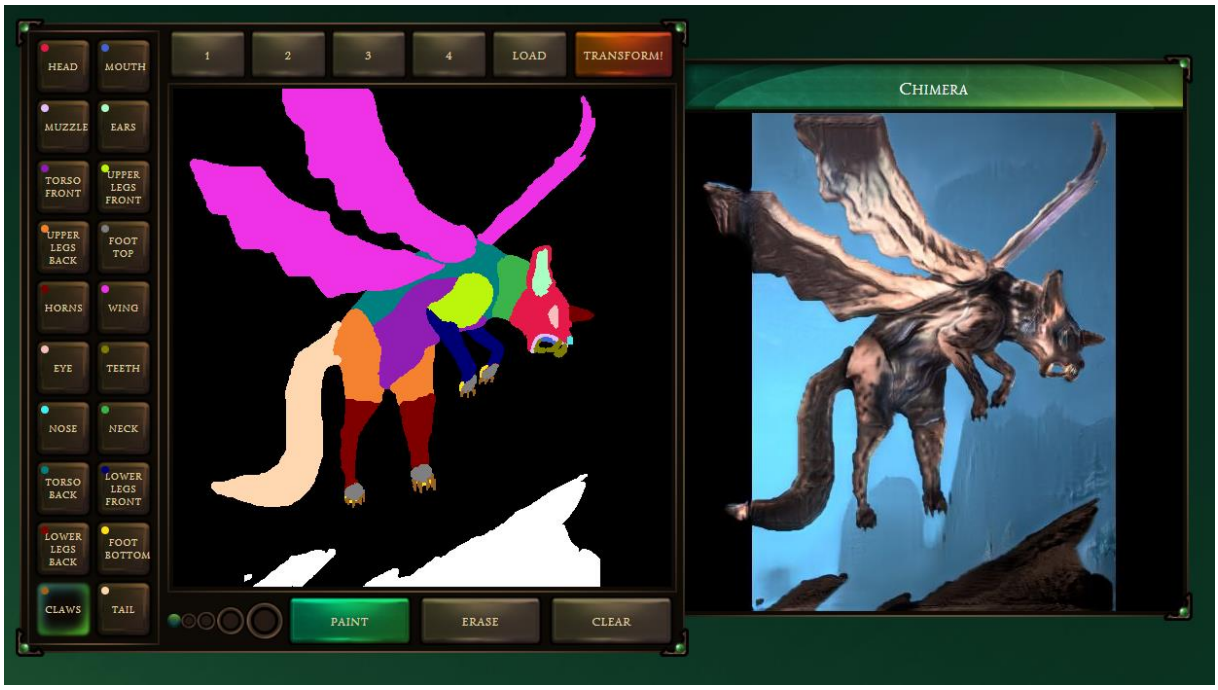
A Google fejlesztői egy másféle megközelítésben próbáltak gondolkodni. Ugyan hasonló módon két konvolúciós neurális háló állítja elő a „modellünket”, ebben az esetben a bemenet nem egy fotó, hanem valamilyen rajz, kép, amelyből az algoritmus valamilyen „lényt”/”kimérát” próbál elképzelni.

A neurális hálót különböző szörnyek és élőlények rajzaival tanították, melyeket 3D modellekből nyertek ki, az első szetben a teljes színekkel ellátott képek voltak a bemenetek, míg a másikban fekete fehér képek, melyeken különböző tónusokkal voltak a testrészek elkülönítve. Így tanították meg arra a neurális hálót, hogy akár paint skiccből képes legyen különböző végtagokat, szemeket, de akár szárnyakat vagy agyarokat társítani a rajz különböző részeihet és ezáltal egy épkézláb élölényt, egy „kimérát” kialakítani.




*7. ábra – A chimera painter demo interface-je*

Hasonlóan a facebook AI esetében, itt is található egy demo, amely a fenti ábrán látható interface-t hívja meg, itt különböző színek kiválasztásával jelöljük az adott részeit a képnek, természetesen teljesen nulláról kezdve is ki lehet alakítani valamilyen kimérát.

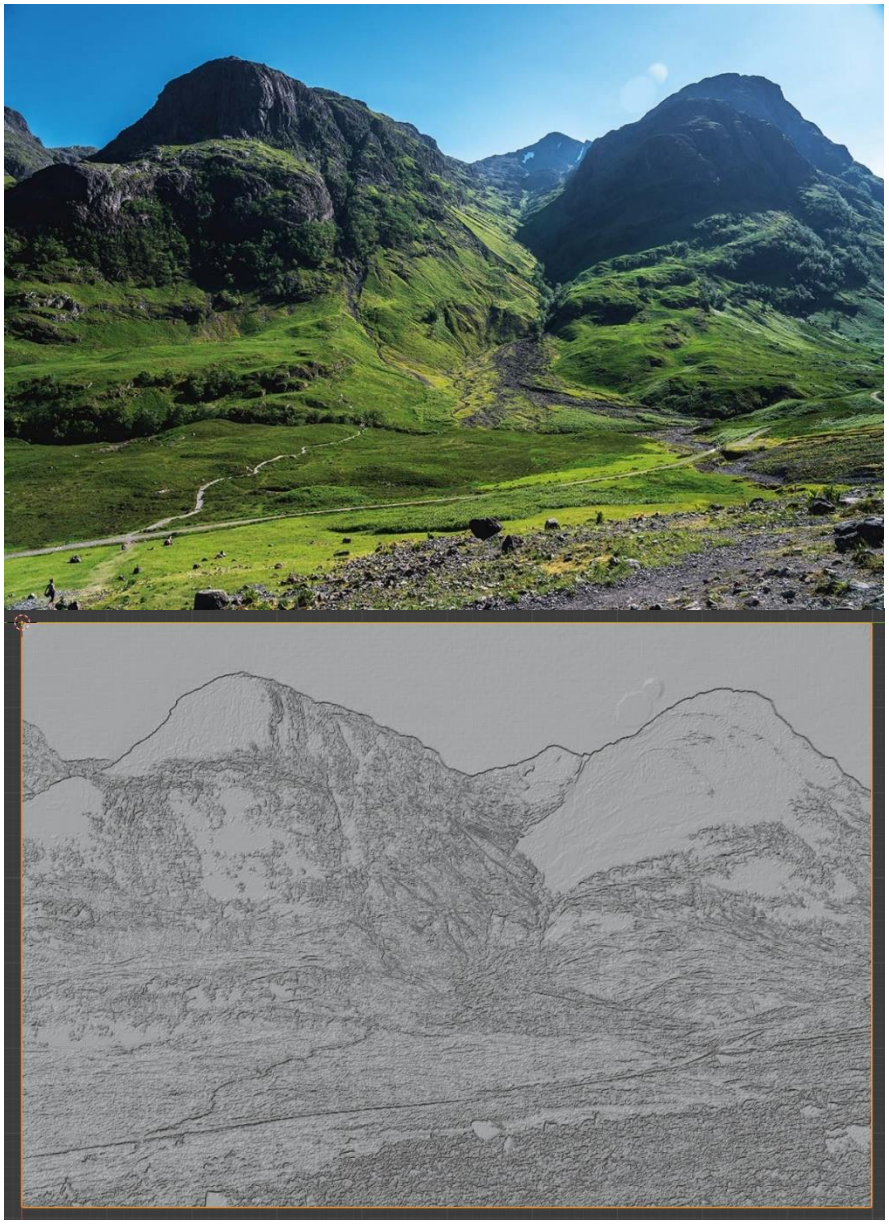



*8. ábra – A google általi Chimera az én rajzomból*

Ugyan modellt sajnos nem lehet róla letölteni, de egy kifejezetten frappáns kis szörnyet volt képes kialakítani az én rajzom alapján is a szoftver, úgy tűnik a textúra a lény közel minden testrészén közel azonos, ezért egy kicsit morbid benyomást tehet, de biztos vagyok benne, hogy ebben a demo funkció is szerepet játszik.

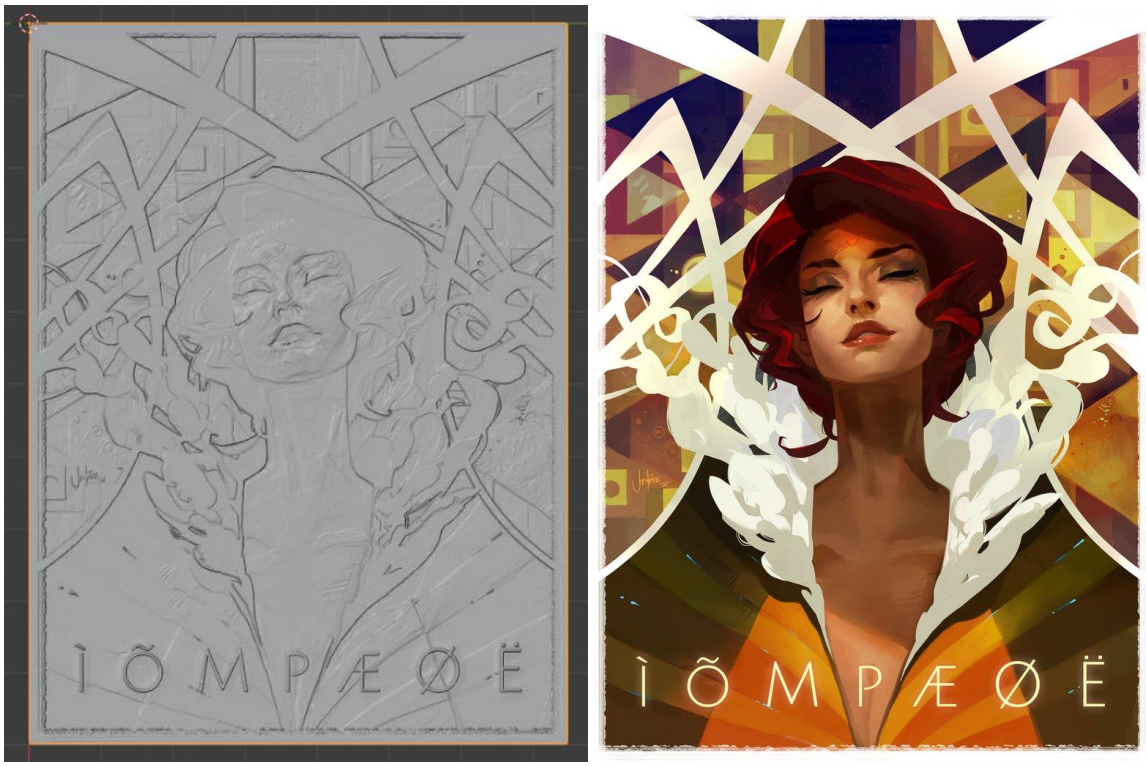
## 1.3. 2D to 3D:

Ezen felül még számos különböző open source kód található az interneten amik különböző inputokból valamilyen 3D outputot generálnak. Minimális python tudással mi is próbáltunk valami hasonlót alkotni. Az általunk készített kód 2D kép input mellett beolvassa a képet, majd azt pixel intenzitás szerint egy domború felületté alakítja, így alakítva ki egy kvázi 3D alakzatot. A kód a következő linken érhető el.



*9. ábra – A képből generált kvázi 3D domború felület*

A fenti ábrán láthatunk egy példát a kód működésére, a fenti bemeneti képből az alsó domború modellt alakította ki a kód. A hegyek, az égbolt, az alul menő út viszonylag jól kivehető a modellből, viszont ennél mélyebb részleteket már nem olyan egyszerű megfigyelni. A képet formailag jól írja le a modell, viszont a mélyebb részletek sajnos elvesznek benne.



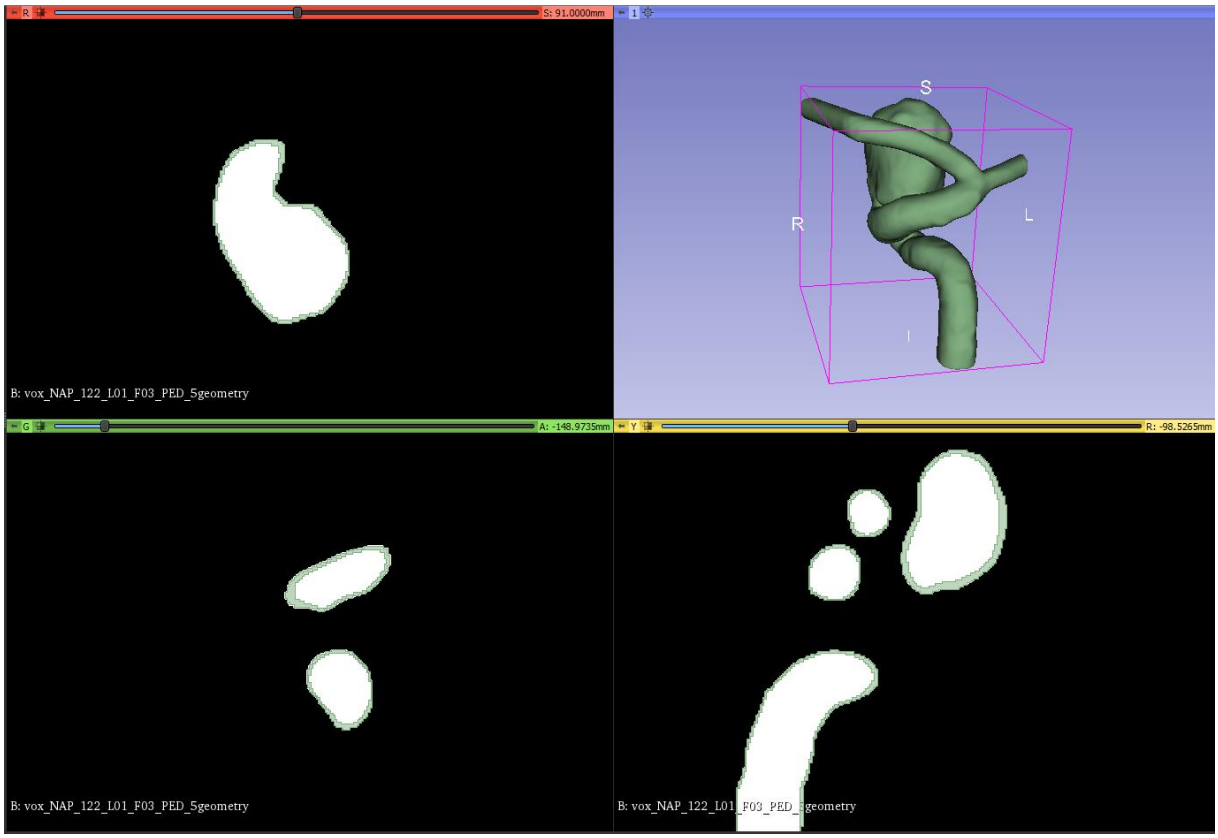
*10. ábra – Egy poszter leképezése*

A fenti ábrán egy másik poszter leképezését próbáltam meg, ebben az esetben a domború felület egészen jól leköveti az arc dinamikáját, kivehető az orr kiemelkedése, a szemgödrök, az állvonal, valamint a kulccsont és a nyak domborulatai is észrevehetőek. Ez a példa kevesebb mélységgel és részlettel bírt, mint a kép a hegyről, így a kód is jobban le tudta követni a változásokat.

## 1.4. Szegmentáció:

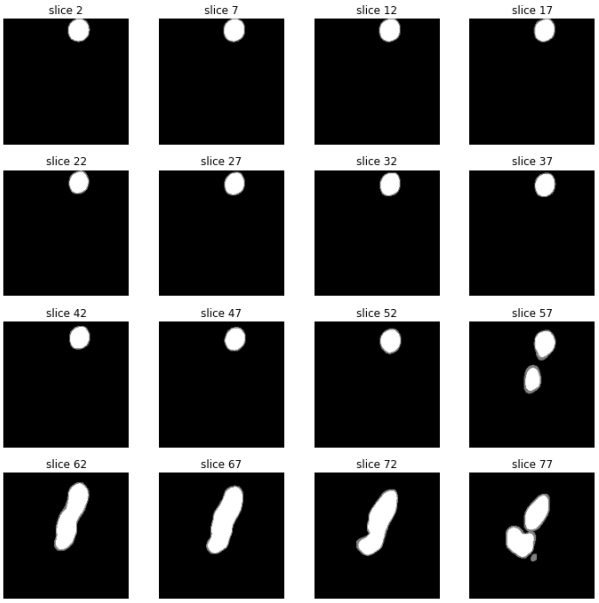
Ugyan a szegmentáció nem teljesen tartozik közvetlenül a 2D to 3D elnevezés alá, azonban a mi esetünkben érdekes volt megvizsgálni, hogy hogyan is lehet egy CT felvételt szegmentálni. Mindketten Biomechatronika szakirányosok vagyunk és erek áramlástani szimulációiból írjuk a szakdolgozatunkat, így napi szinten használunk szegmentációt CT képek feldolgozására és 3D meshek készítésére, azonban a hátterét nem ismerjük a szoftvernek. Így egy kicsit ebbe a témába is beletekintettünk.

Különböző online források alapján próbáltunk egy python kódot írni, amely képes szegmentálásra. A kód elérhető online. Első próbálkozásként a diplomamunkámban használt érgeometriák egyikéből készítettünk egy egyszerű DICOM sorozatot, amelyet 3Dslicerben is szegmentáltunk és ezzel az adatsorral próbáltuk ki a kódot. Ez egy elég tiszta adatsor, hiszen itt utómunka után csupán az érszakasz részei láthatók.



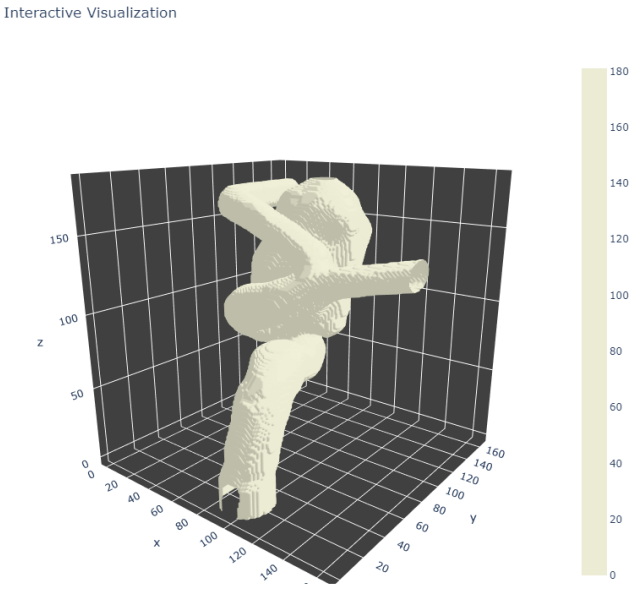

*11. ábra – 3D slicerben szegmentálva az említett érszakaszt*

A fenti ábrán látható, hogy itt csupán a hasznos geometria van jelen, így az ebből a Slicerrel készített dicom adatsorban is csak ez lesz majd látható.



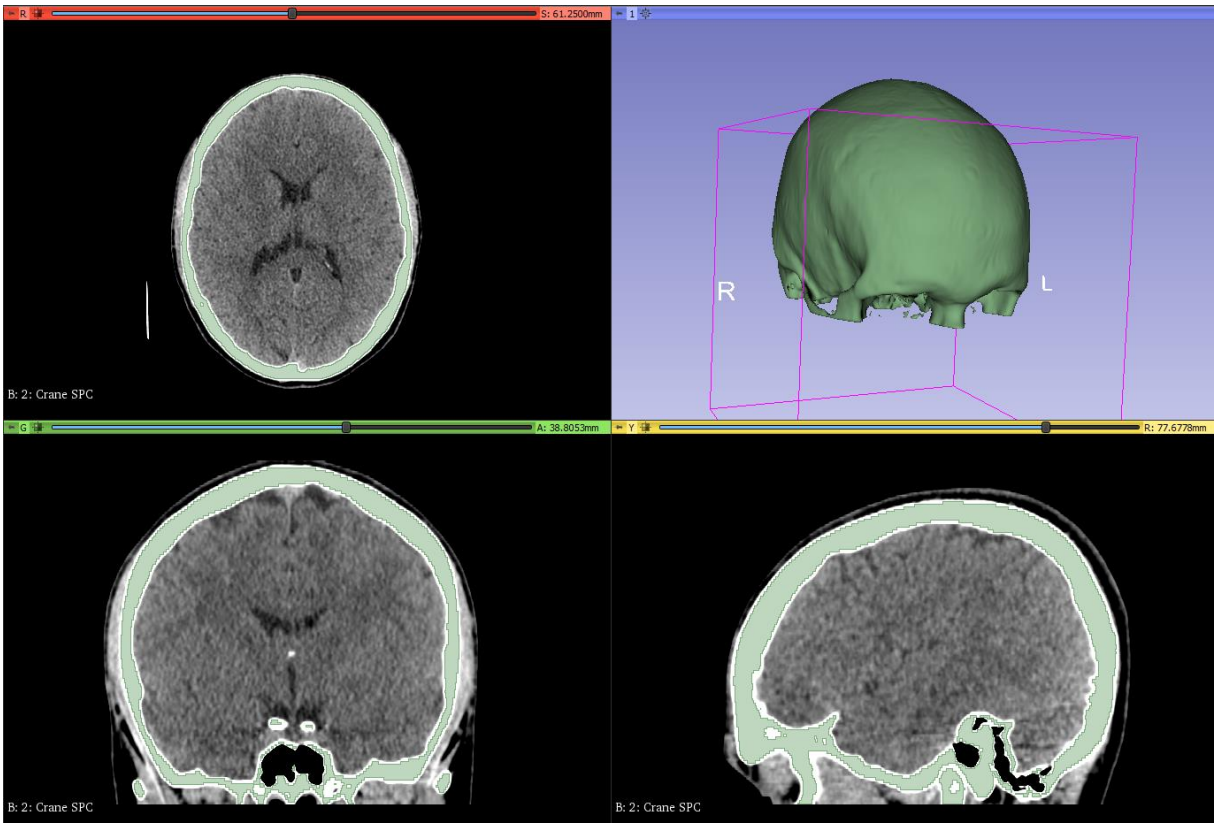

12. *ábra – Szeletek a dicom adatsorból*

Ezután kipróbáltuk ezzel a Dicom adatsorral a kódot. Első lépésben csupán betöltöttük az adatokat és bizonyos szeleteket néztünk meg, hogy sikerült-e beolvasni. Amint azt a fenti ábrán is láthatjuk a kód sikeresen beolvasta az adatokat. Ezután kipróbáltuk, hogy képes-e a kód egy mesht kialakítani a fájlból.



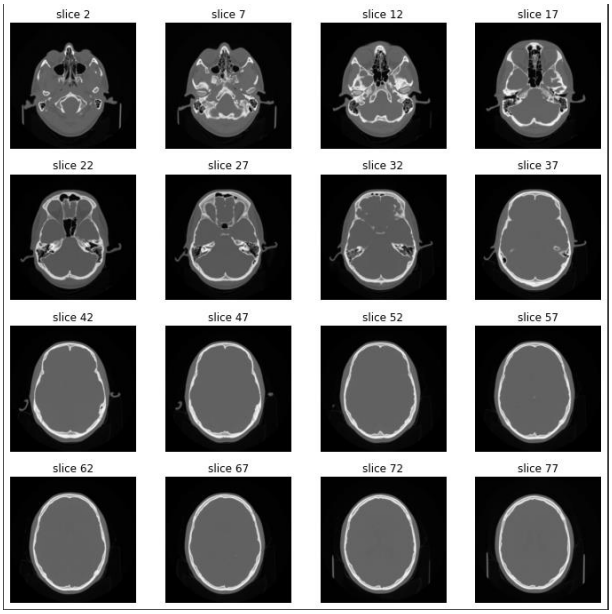

*13. ábra – A kód által kialakított mesh*

Az ábrán látható mesh közel azonos a 3D Slicerben készített meshünkkel, az érszakasz alsó részén a 0. pozícióban valószínűleg valahol elvesztettünk egy voxelnyi adatot, hiszen a Slicerben látható modellen itt zárt volt a geometria. A másik nagyobb különbség a simítás hiánya, aminek köszönhetően a Slicerben látható modell nem „rücskös” egyáltalán, hanem egy sima felületet láthatunk.



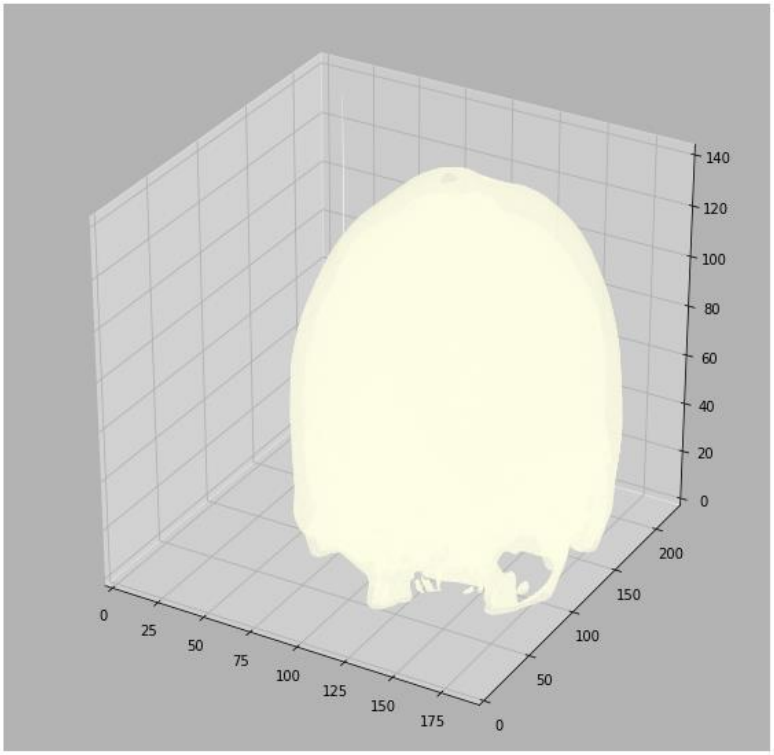

14. *ábra – Koponya CT slicerben*

Ezután kipróbáltuk a kódot egy igazi dicom lista segítségével. A pcir.org weboldalon publikusan elérhető CT képek közül töltöttünk le egyet és néztük meg elsőként Slicerben, ahol a fenti képet láthattuk. Ez alapján a vizsgált adat egy koponya CT felvétele, amelyet ezután átadtunk a kódnak kipróbálásra.




*15. ábra – Koponya CT dicom szeletei*

A kód által beolvasott dicom szeletek jónak tűnnek, így futtathatunk tovább.



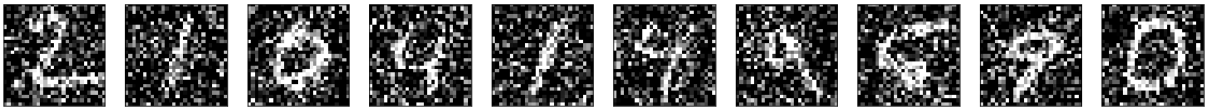

*16. ábra – A kód által készített szegmentáció*

A fenti ábrán láthatjuk, hogy hasonló módon a Slicerhez a kód megtalálta csontszövetet és képes volt azt szegmentálni a kép többi részétől.

Egy másik érdekes alkalmazás a Hasib Zunair által írt neurális háló, amely tüdő CT felvételek alapján próbálja meg eldönteni, hogy az adott páciens átesett-e már COVID-19 víruson vagy sem. A kód egy publikus kutatás CT felvételeit használja, amelyben két külön csoportban COVID-on átesett és COVID mentes tüdők CT felvételei vannak kategorizálva, a neurális háló ezeken az adatokon tanul. Ebben a kódban egy hasonló CT felvétel feldolgozási és szegmentálási kódrészlet után, egy 3D konvolúciós neurális hálós tanítással készít predikciókat a CT felvételekről.

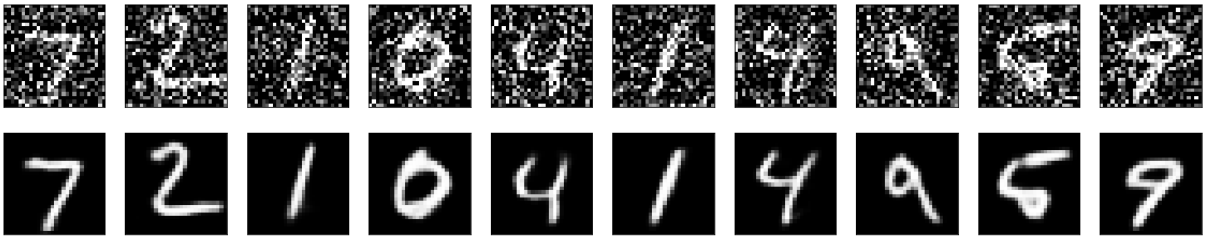
## 1.5. Képjavítás/zajcsökkentés:

Ugyan nem feltétlenül tartozik a 2D to 3D témaköréhez szintén, de képfeldolgozási technikáknál jelentős szerepet tölt be a képjavítás, felskálázás annak érdekében, hogy a feldolgozott kép alapján jobb végterméket tudjunk előállítani. Erre a problémára próbáltunk meg egy autoencodert készíteni a keras tutorialjai alapján, amely képes zajt eltávolítani képekből, egy többrétegű konvolúciós neurális háló segítségével. Ezt a hálót a keras MNIST adatcsomagjában levő 24x24 pixeles fekete alapon fehér számok képeivel tanítottuk, majd egy kicsit nagyobb 64x64 pixeles mintán vizsgáltuk is a zajcsökkentés képességeit is.



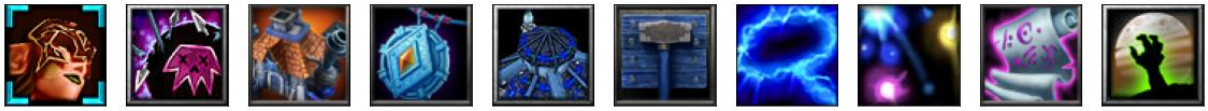
17. ábra – Az MNIST adatcsomag képei zajjal ellátva

A fenti ábrán látható módon miután betöltöttük a csomag képeit, ezeket zajosítottuk, majd ezeken a zajos képeken és az eredeti képeken tanítottuk a neurális hálót. Illetve vizsgáltuk, hogy milyen mértékben képes a zajt eltávolítani a képekről.



*18. ábra – A zajtalanított képek a zajossal összehasonlítva*

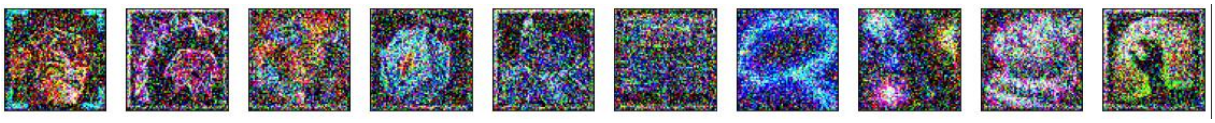
A fenti ábrán láthatjuk a zajtalanított képeket, ez alapján a neurális hálónk 50 epochos tanulás után jó munkát végzett, a képeken látható zaj közel teljes mértékben eltűnt. Vizsgáljuk meg azonban más, nem ebből a mintából vett képek alapján magát a neurális hálót és annak eredményeit.



19. ábra – Warcraft 3 ikonok 64x64 pixeles méretben

Annak érdekében, hogy a konfigurációs, valamint a futási idők ne nőljenek meg túlzottan kisméretű képeket kerestünk. Szerencsére a Warcraft 3 nevű RTS játék ikonjai annak következtében, hogy a játék a 2000-es évek elején jött ki hasonló méretben vannak, mint a korábbi mintánk. A fenti ábrán ezen ikonokból láthatunk párat, ezek 64x64 pixel méretű, .png kiterjesztésű RGB képek. A játékhoz összesen ~800 ikon tartozik, ez sajnos kicsi minta ahhoz, hogy minőségi tanítást tudjunk elérni, de azért így is megpróbálhatjuk. Ezt a ~800 ikont két részre bontjuk, ~600 tanítási ikon és ~200 validálási ikon.

Ezek után, ugyanúgy mint az előző mintaszettnél zajt rakunk a képekre és meg is van az adathalmazunk.



*20. ábra – Warcraft 3 ikonok zajjal*

A fenti ábrán láthatjuk a zajos képeket, a képeken akkora mértékű zaj található, hogy magát a képet torzítja, azonban a lényeges információ, körvonalak és maga a kép még kivehető belőle. Ezek után kezdhetjük is a neurális háló tanítását.

```
Epoch 47/50
469/469 [==============================] - 145s 309ms/step - loss: 0.0941 - val_loss: 0.0942
Epoch 48/50
469/469 [==============================] - 146s 311ms/step - loss: 0.0942 - val_loss: 0.0940
Epoch 49/50
469/469 [==============================] - 144s 308ms/step - loss: 0.0943 - val_loss: 0.0939
Epoch 50/50
469/469 [==============================] - 144s 308ms/step - loss: 0.0941 - val_loss: 0.0938
Epoch 97/100
6/6 [==============================] - 11s 2s/step - loss: 0.4744 - val_loss: 0.4878
Epoch 98/100
6/6 [==============================] - 11s 2s/step - loss: 0.4746 - val_loss: 0.4877
Epoch 99/100
6/6 [==============================] - 11s 2s/step - loss: 0.4747 - val_loss: 0.4882
Epoch 100/100
6/6 [==============================] - 11s 2s/step - loss: 0.4748 - val_loss: 0.4880
```

*21. ábra – A neurális hálók tanulása*

A fenti képen láthatjuk a két eset neurális hálóinak tanulását, míg az első esetben 50 epoch alatt is 0,1 alá konvergált a veszteségi függvény, addig az ikonok esetében 100 epoch alatt is csak 0,5 alá sikerült ezt az értéket vinni. Ez köszönhető annak, hogy az első mintaszettben 60000 tanulási mintánk volt, míg a másodikban csupán ~600 csak, ez két nagyságrend beli különbség, valamint sokkal egyszerűbb fekete fehér képekről volt szó, míg ebben az esetben színes, különböző stílusú és színvilágú képekről van szó.

22. ábra – Warcraft 3 ikonok zaj nélkül eredmény

Eredményünk azonban így is született, amelyet a fenti ábrán láthatunk, a zaj láthatólag eltűnt, azonban jelentős minőségi esést tapasztalhatunk a képekben, a képek körvonala, az általuk ábrázolt tárgy, jelenet vagy éppen mágikus képesség még így is kivehető.

A minőségvesztés természetesen része az ezen technológiával való munkának, a következő lépés az eredeti képek visszanyerésének érdekében egy képjavító/upscaling eljárás lenne, amely az elvesztett pixeleket próbálná pótolni. Sajnos azonban az időnk szűkössége miatt erre a részre már nem volt kapacitásunk.

## 2. Összegzés:

A képfeldolgozási eljárások az utóbbi évtizedben jelentős mértékben fejlődtek, azonban a 2D to 3D eljárások még nem képesek egy rutinos művészt, szobrászt, modellalkotót helyettesíteni, azonban lényegesen képesek megkönnyíteni a munkájukat, felgyorsítani azt a folyamatot, mely során egy koncepcióból egy kész modellhez jutnak. A mesterséges intelligenciák, neurális hálók ilyen téren hatalmas fejlődésen mentek át és segítik ezt a folyamatot több irányból.

Az orvosi adatfeldolgozásban, CT, illetve MRI felvételek alapján a mai világban már egyszerű felhasználói szoftverekkel képesek vagyunk modelleket alkotni. Ezen modellek alapján pedig neurális hálók segítségével megpróbálhatunk különböző diagnózisokat felállítani, azonban ezek tanításához lényeges méretű adatsor szükséges, amely képes megfelelő pontossággal tanítani a neurális hálót.

A képjavítás és zajcsökkentés egy igen fontos része a képfeldolgóz technológiáknak, különösen abban az esetben, ha az inputunkról érkező kép nagyon zajos vagy éppen régen készült és az akkori technológia engedte felbontás igencsak csekély. Az általunk végigvezetett példán láthatjuk, hogy már egy könnyen kezelhető, otthoni számítógépen futtatható és tanítható neurális háló esetében is érhetünk el komoly eredményeket a zajcsökkentés terén.