PONTFELHŐ REGISZTRÁCIÓ
ITERATIVE CLOSEST POINT

Cserteg Tamás, URLGNI, 2018.11.22.

TARTALOM
Röviden

- Alakzatrekonstrukció áttekintés
- ICP algoritmusok
- Projektfeladat
- Demó

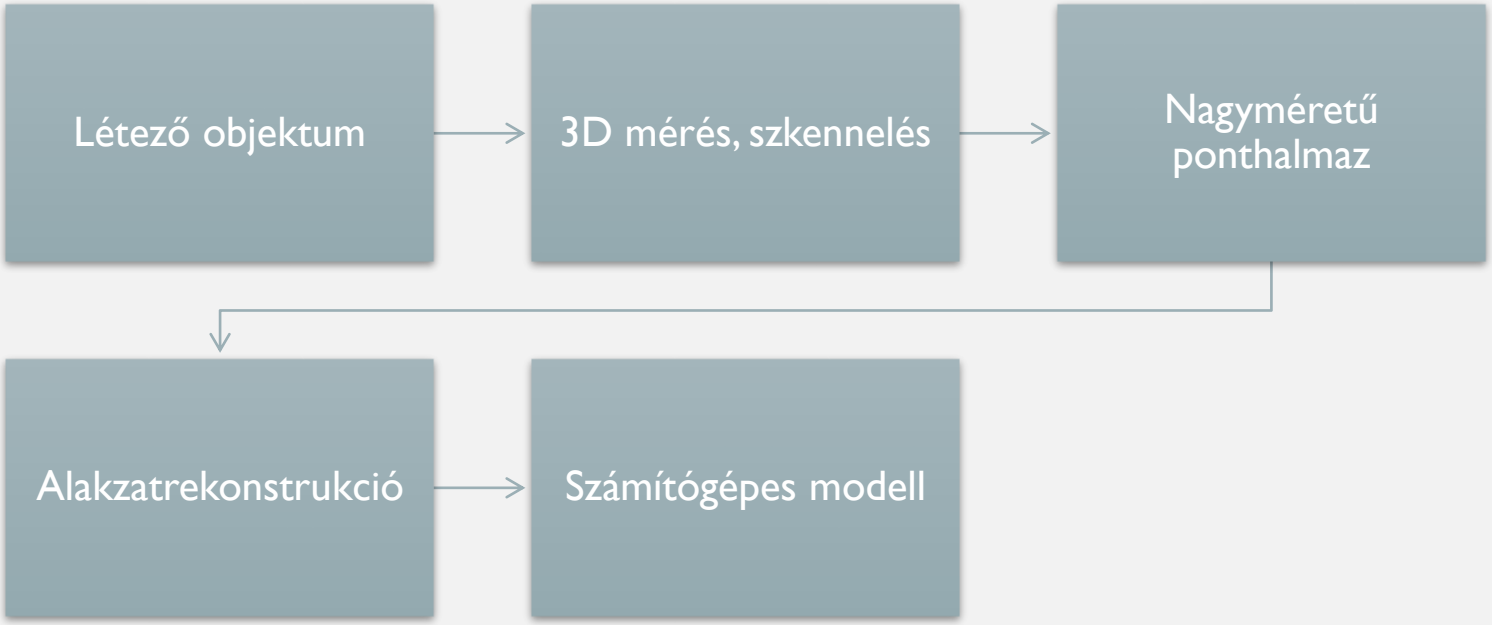
FORRÁSOK

- Cikkek
  - Efficient Variants of the ICP Algorithm [Rusinkiewicz 01]
  - A Method for Registration of 3-D Shapes [Besl 92]
- Órai diák, jegyzetek
- GitHub
  - Julia
  - MeshCat
  - NearestNeighbors
- Modellek
  - https://graphics.stanford.edu/data/3Dscanrep/
  - Salvi Péter
  - Universal Robots

ALAKZATREKONSTRUKCIÓ
Motiváció

- Nincs digitális modell (pedig kéne)
- Nem CAD technológiával készülő alkatrész
- Orvosi alkalmazások (fogászat, protézisek, robotizált sebészet)
- Kulturális örökség megőrzése
- Minőségellenőrzés

ALAKZATREKONSTRUKCIÓ
Folyamat



ALAKZATREKONSTRUKCIÓ
Szenzorika

ALAKZATREKONSTRUKCIÓ
Részletes folyamat

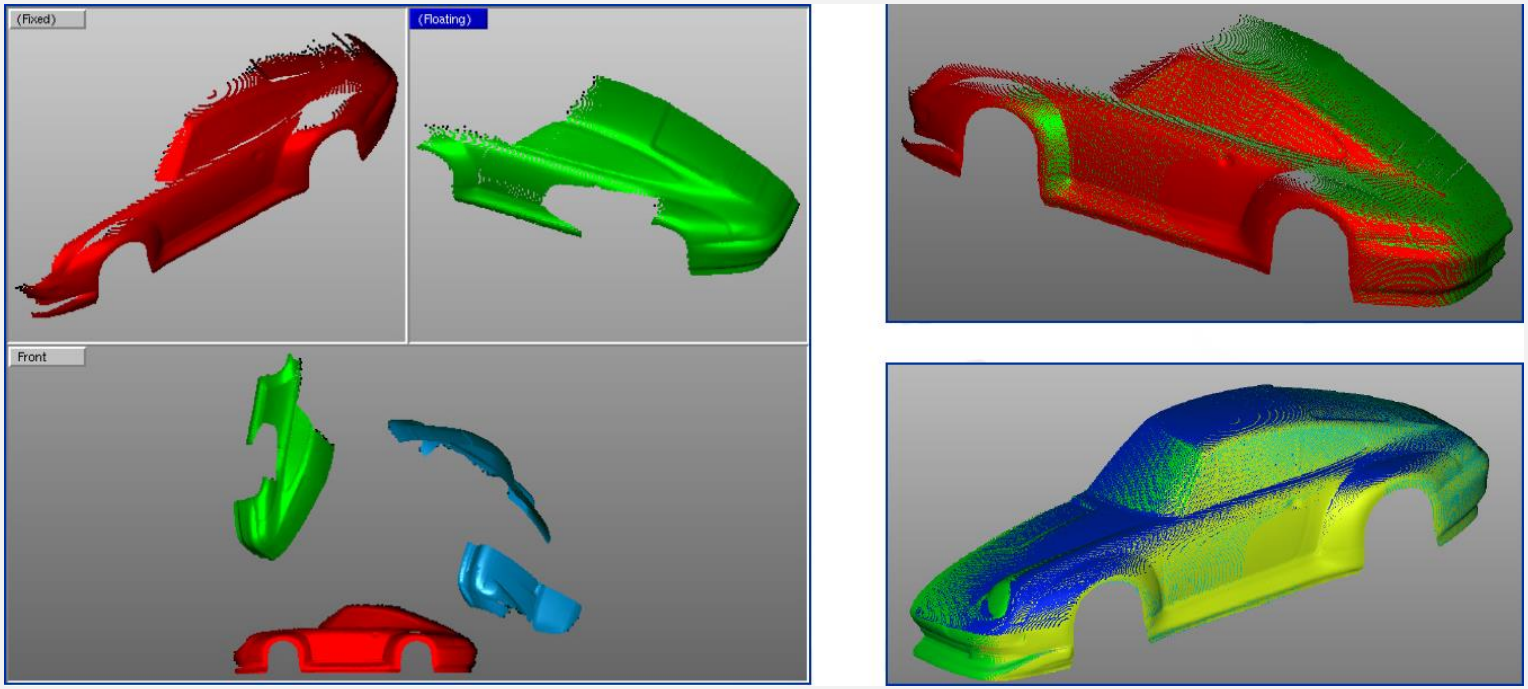
A digitális alakzatrekonstrukció folyamata

mért pontfelhők → számítógépes modell
fázisok:

1. 3D-s mérés..... → pontfelhő
2. ponthalmazok egyesítése és ritkítása pontfelhő → pontfelhő
3. háromszöghálók létrehozása..... pontfelhő → háromszögháló
   és javítása..... háromszögháló → háromszögháló
4. szegmentálás..... háromszögháló → tartományok
5. felületek osztályozása..... tartományok → attribútumok
6. elsődleges felületek illesztése..... tartományok → felületek
7. összekötő felületek illesztése..... felületek → felületek
8. modellek tökéletesítése, “fairing”..... felületek → felületek
   és kényszerek..... felületek → felületcsoportok
9. minőségellenőrzés ..... felületcsoportok → CAD modell
10. export CAD alkalmazások érdekében CAD modell →

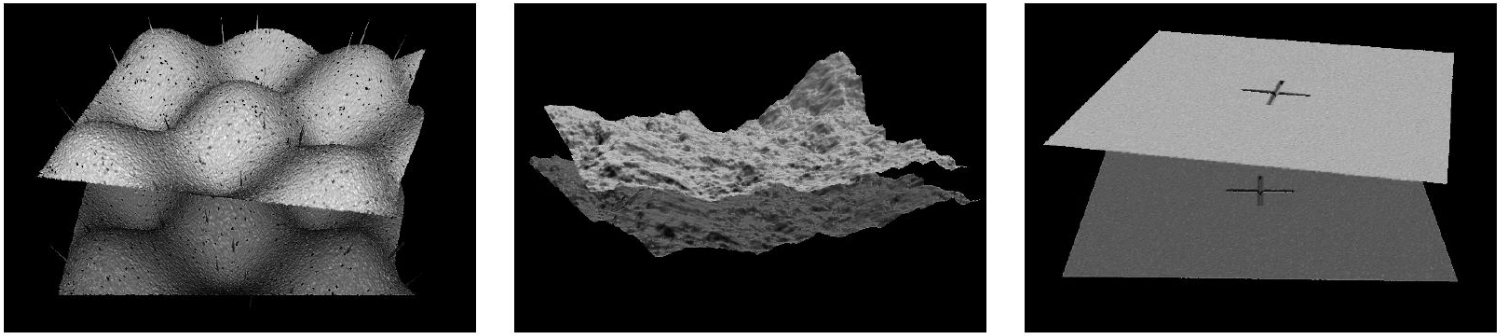
ALAKZATREKONSTRUKCIÓ
Pontfelhő regisztráció

Cél: különböző szögből felvett pontfelhők egyesítése



ITERATIVE CLOSEST POINT
Algoritmus

- Pontfelhők vagy háromszöghálók
- Párokra értelmezett: reading, reference
- Kettő közti transzformáció keresése
- Iteratív minimalizálás
- Pontok + normálisok, színek, intenzitások



(a) Wave (b) Fractal landscape (c) Incised plane

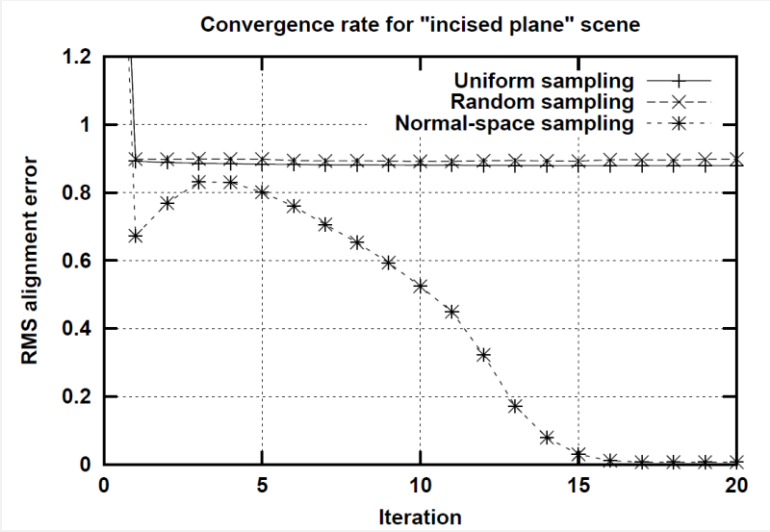
Figure 1: Test scenes used throughout this paper.

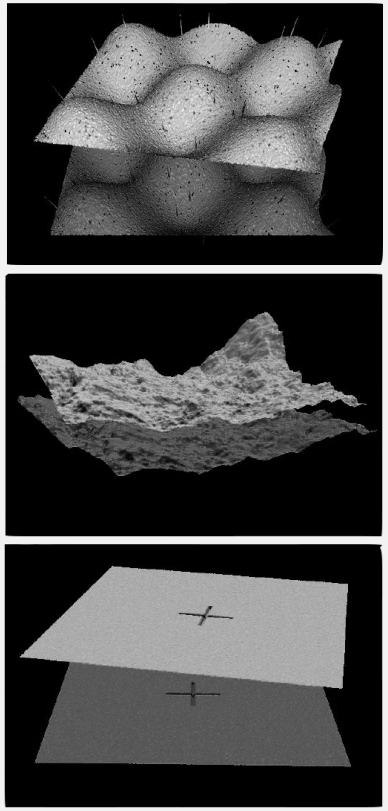
ITERATIVE CLOSEST POINT
Iteráció

1. Pontok kiválasztása (Selection)
2. Pontok párosítása (Matching)
3. Párok súlyozása (Weighting)
4. Párok kidobása (Rejection)
5. Hiba számítása és minimalizálása (Error metric, minimization)
   - Kilépési feltétel

ITERATIVE CLOSEST POINT
Selection

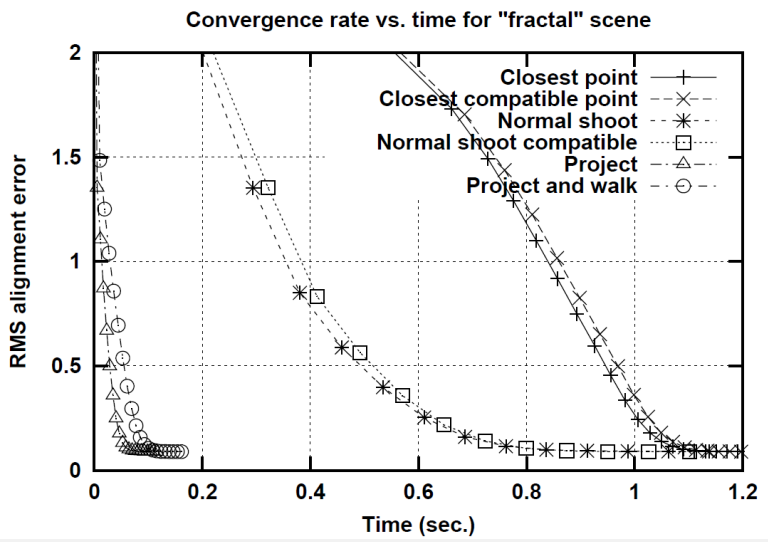
- Összes pont használata
- Véletlenszerű válogatás
- Nagy gradiensű pontok (szín, intenzitás)
- Normálisok eloszlása minél egyenletesebb legyen
- Egyik/mindkét mintából válogatás

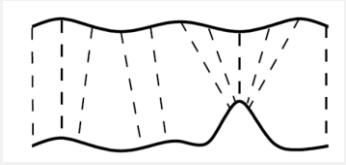


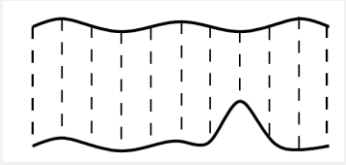


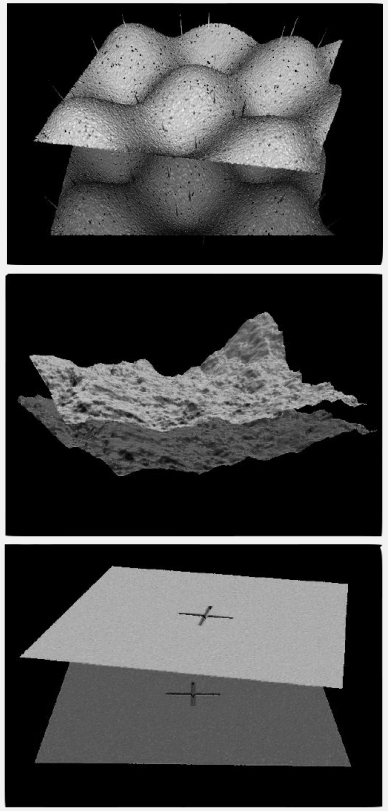
ITERATIVE CLOSEST POINT
Matching

- Legközelebbi pont
- Normál irányú metszés
- Vetítés a „célfelhő” nézőpontjából
- Előbbiek kompatibilitás ellenőrzésével
- Vetítés és keresés



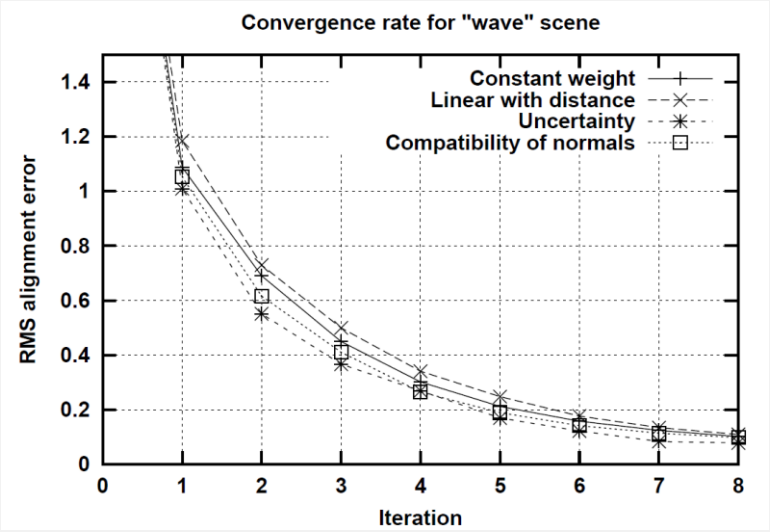


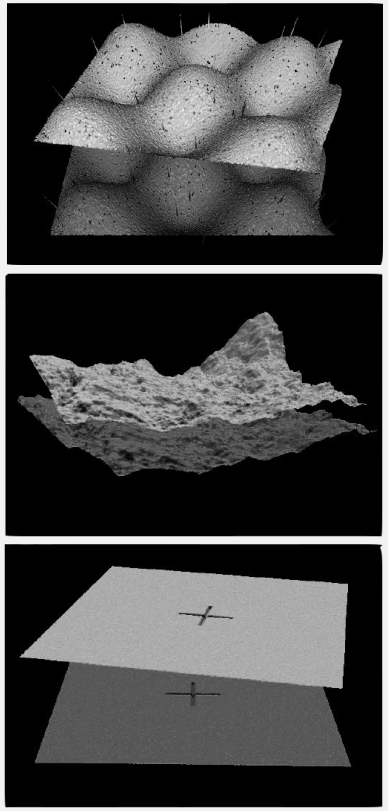




ITERATIVE CLOSEST POINT
Weighting

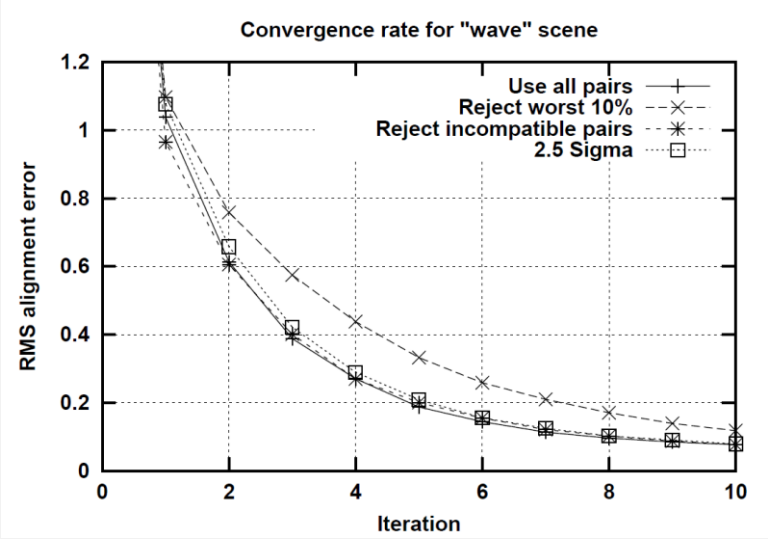
- Konstans súly
- Távolsággal fordítottan arányos súly
- Normálisok/színek szorzata
- Bizonytalanságon alapuló súlyok

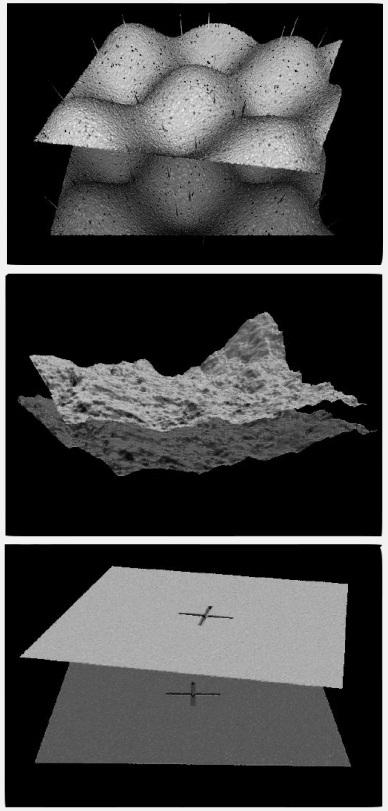




ITERATIVE CLOSEST POINT
Rejection

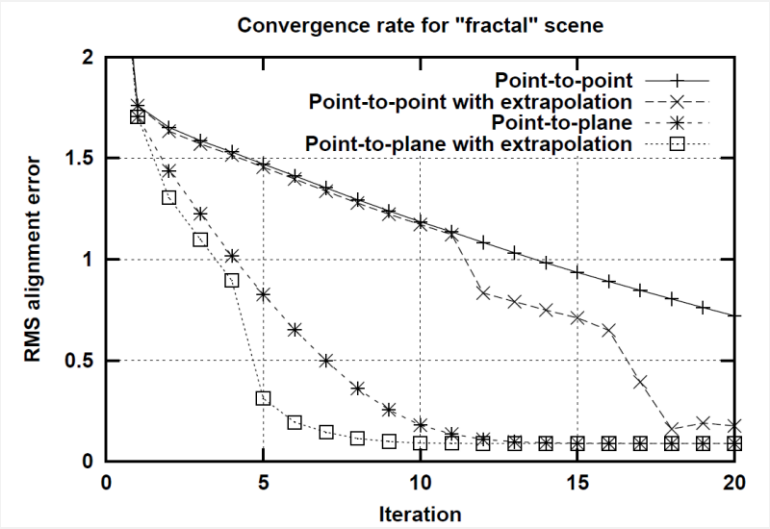
- Küszöbértéknél nagyobb távolság
- Legrosszabb x%
- Szórás alapján
- Környező párok távolságával hasonlítás
- Határon lévő pontok

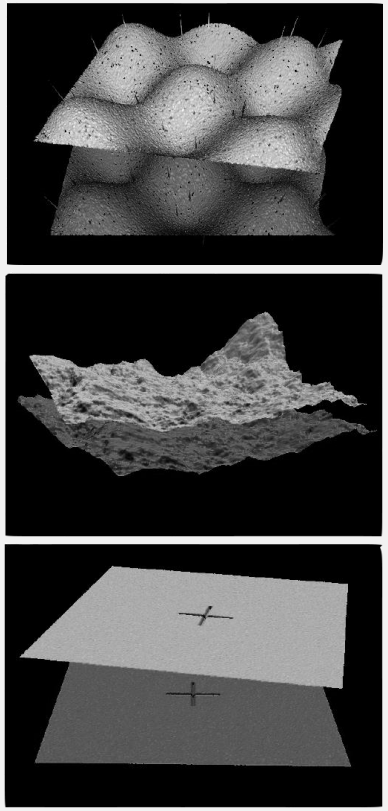




ITERATIVE CLOSEST POINT
Error metric

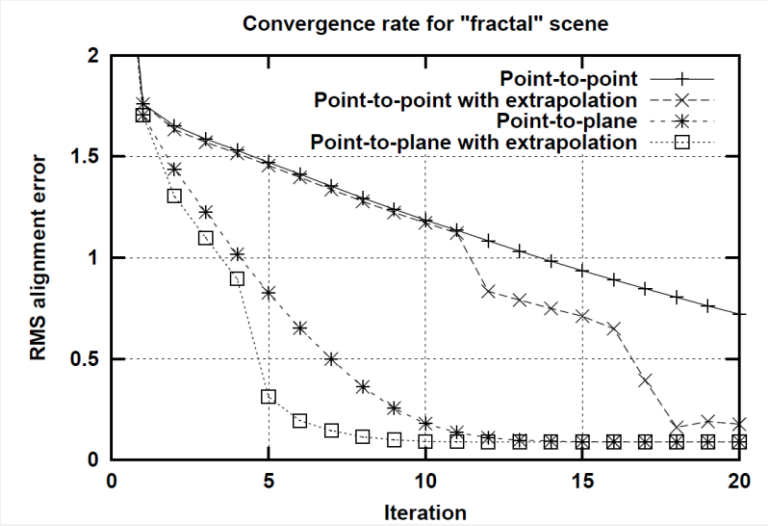
- Négyzetes távolságösszeg (zárt megoldás)
  - SVD, kvaterniók, ortonormált mátrixok
- Fenti kiegészítése színekkel
- Pont-sík távolság

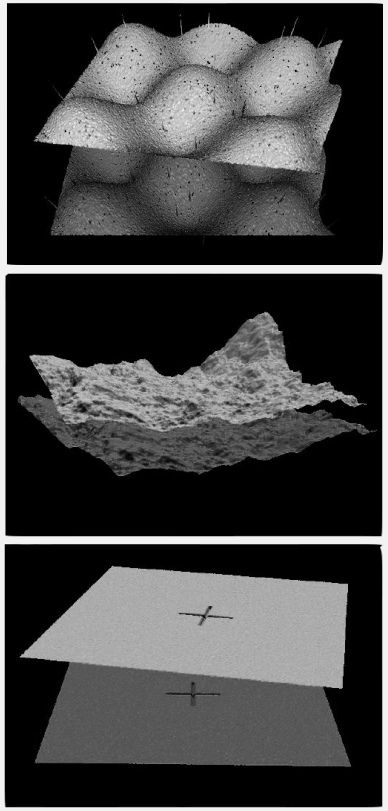




ITERATIVE CLOSEST POINT
Minimization

- Egyszerű iterálás
- Fenti, extrapolációval
- Kezdeti feltétel perturbációja, legjobb kiválasztása
- Véletlenszerű részhalmazok, legjobb kiválasztása
- Sztochasztikus keresés





ITERATIVE CLOSEST POINT
State of the art

- Problémák: zaj, hiányzó adatok, részleges fedés

- The Trimmed Iterative Closest Point Algorithm [Chetverikov 02]
- Affine iterative closest point algorithm for point set registration [Du 10]
- Sparse Iterative Closest Point [Bouaziz 13]
- Probability iterative closest point algorithm for m-D point set registration with noise [Du 15]

PROJEKT
Feladat

- Input: több pontfelhő kül. koordináta rendszerben (azonos méretben)
- Output: egy, az összes adatot összesítő pontfelhő
- ~~Felhasználó segíthet 3-3 összetartozó pont megadásával~~
  - (a használt környezet nem interaktív, csak megjelenít)
- Szeminárium, prototípus implementáció

PROJEKT
Algoritmus

1. Selection: véletlenszerű válogatás
2. Matching: euklideszi távolság (kd-fa)
3. ~~Weighting: konstans~~
4. Rejection:
   - Legrosszabb x%
   - Szórás alapján
5. Error metric, minimization: négyzetes távolságösszeg
   - Kilépési feltétel: átlag távolságnégyzet

PROJEKT
Matek

- Minimalizálandó: $\sum\|\boldsymbol{x}_i - \boldsymbol{M}_r\boldsymbol{p}_i - \boldsymbol{v}\|$ =$\sum\|\boldsymbol{x}_i - \boldsymbol{R}(\boldsymbol{q}_r)\cdot\boldsymbol{p}_i - \boldsymbol{q}_t\|$
- Regisztrációs vektor: $\boldsymbol{q} = [\boldsymbol{q}_r|\boldsymbol{q}_t]$
- Forgatás kvaternió: $\boldsymbol{q}_r$, eltolásvektor: $\boldsymbol{q}_t$
- Optimális forgatás: sajátérték-sajátvektor problémára vezethető vissza
- Optimális eltolás: $\boldsymbol{q}_t = \boldsymbol{\mu}_x - \boldsymbol{R}(\boldsymbol{q}_r)\cdot\boldsymbol{\mu}_p$

PROJEKT
Matek

- Minimalizálandó: $\sum\|\boldsymbol{x}_i - \boldsymbol{M}_r\boldsymbol{p}_i - \boldsymbol{v}\| = \sum\|\boldsymbol{x}_i - \boldsymbol{R}(\boldsymbol{q}_r)\cdot\boldsymbol{p}_i - \boldsymbol{q}_t\|$
- Regisztrációs vektor: $\boldsymbol{q} = [\boldsymbol{q}_r|\boldsymbol{q}_t]$
- Forgatás kvaternió: $\boldsymbol{q}_r$, eltolásvektor: $\boldsymbol{q}_t$
- Optimális forgatás: sajátérték-sajátvektor problémára vezethető vissza
- Optimális eltolás: $\boldsymbol{q}_t = \boldsymbol{\mu}_x - \boldsymbol{R}(\boldsymbol{q}_r)\cdot\boldsymbol{\mu}_p$

opcionális

$$\Sigma = \frac{1}{n}\sum(p_i - \mu_p)(x_i - \mu_x)^T$$

Kovariancia-mátrix

$$Q(\Sigma) = \begin{bmatrix} \mathrm{tr}(\Sigma) & \Delta^T \\ \Delta & \Sigma + \Sigma^T - \mathrm{tr}(\Sigma)\mathbf{I}_3 \end{bmatrix}$$

$Q(\Sigma)q_R = \lambda_1 q_R$ és $\lambda_1 = \max \lambda_i$

$$\Delta = \begin{bmatrix} \Sigma_{23} - \Sigma_{23}^T \\ \Sigma_{23} - \Sigma_{23}^T \\ \Sigma_{23} - \Sigma_{23}^T \end{bmatrix}$$

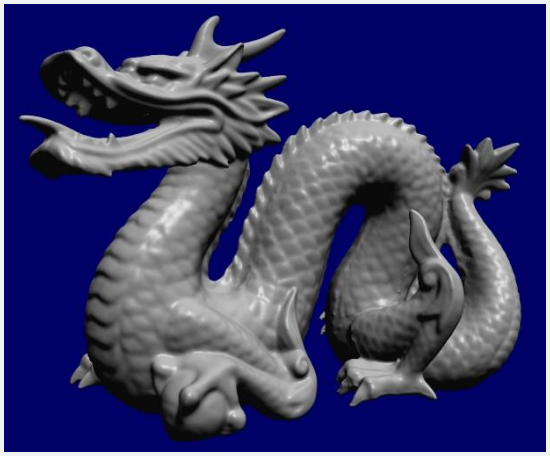
PROJEKT
Julia

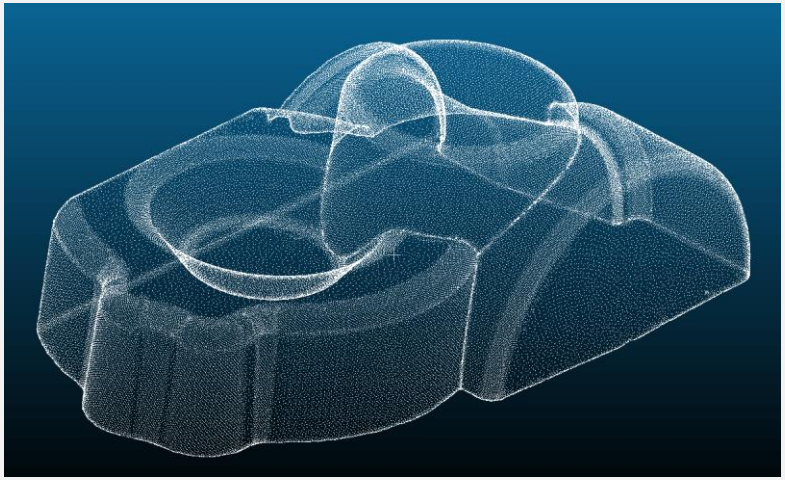
- Julia
  - Ijulia: Jupyter notebookok
  - MeshCat: megjenelítő
  - NearestNeighbors: kd-fa
  - Colors: színszótár
  - StatsBase: véletlenszerű kiválasztás
  - ~~CuArrays: GPU~~
  - https://github.com/cserteGT3/PlayGround/tree/master/SzimGeo-ICP
- CloudCompare: preprocess

PROJEKT
Kód

```
function iterateSigmaRej(aRef, aRed, kdRef, itmax, samp, errmax)
    homTrDict = Dict{Int,Tuple}()
    # Calc the 0-th element of the dict for benchmark
    maxsamp = min(size(aRef,2),size(aRed,2))
    ri = randomSampleIndexes(samp,maxsamp)
    indM, trM = createKnnPairArray(aRed, ri, kdRef)
    refV = @view aRef[1:3,indM[:,1]]
    redV = @view aRed[1:3,indM[:,2]]
    errD = sum(colwise(SqEuclidean(),refV,redV))
    homTrDict[1] = (time_ns(),Matrix{eltype(aRef)}(I,4,4),errD)
    for i in 1:itmax
        ri = randomSampleIndexes(samp,maxsamp)
        indM, trM = createKnnPairArray(aRed, ri, kdRef)
        #reject the worst
        refI, redI = reject25Sigma(indM, trM)
        refV = @view aRef[1:3,refI]
        redV = @view aRed[1:3,redI]
        mu_ref = CoM(refV, 1)
        mu_red = CoM(redV, 1)
        Σ = crosscovMat(redV, refV, mu_red, mu_ref)
        rM = bestRotMat(Σ)
        bTr = mu_ref - rM*mu_red
        hM  = [rM bTr ; 0 0 0 1]
        aRed = hM*aRed
        redV = @view aRed[1:3,redI]
        errD = sum(colwise(SqEuclidean(),refV,redV))/size(refV,2)
        homTrDict[i+1] = (time_ns(),hM,errD)
        if errD < errmax
            break
        end
    end
    @info "Finished with error: $(homTrDict[length(homTrDict)][3]), at threshold: $errmax, at the $(length(homTrDict)-1)th
    return homTrDict
end
```

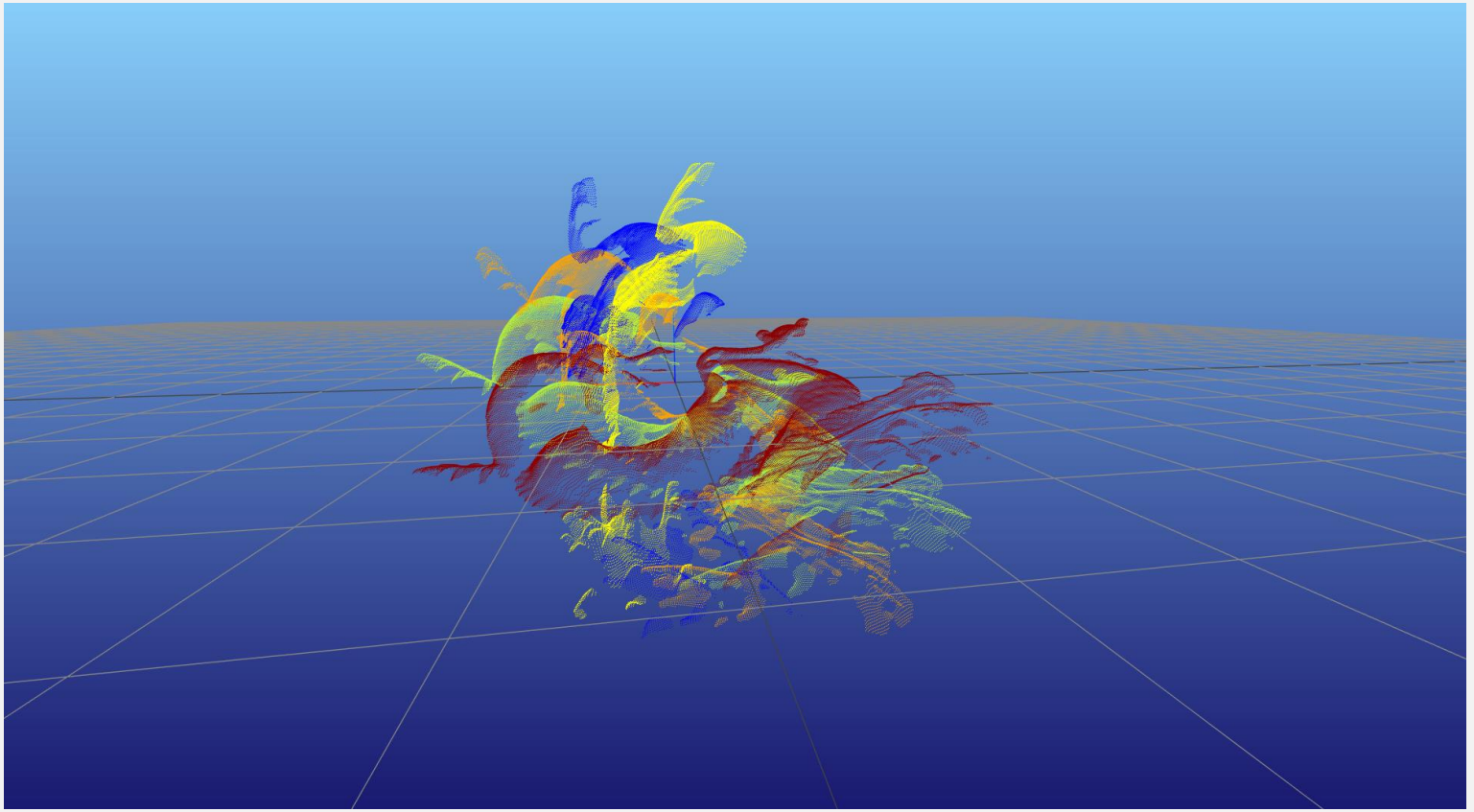
PROJEKT
Modellek



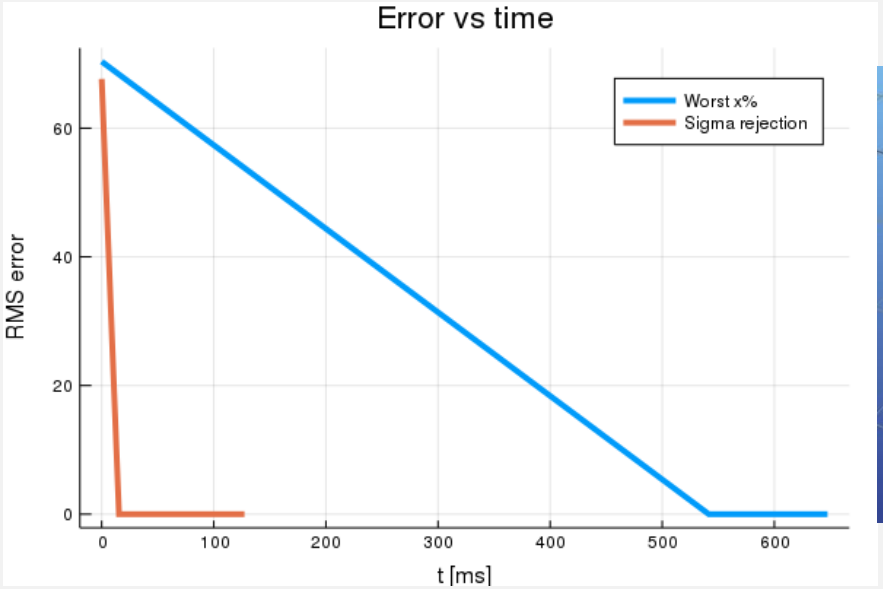


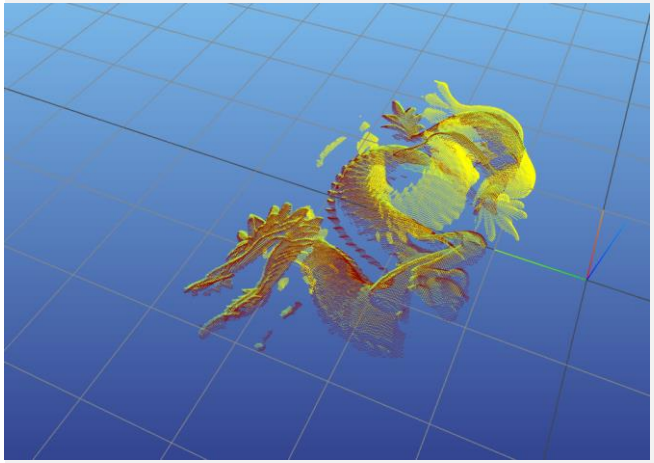
DEMÓ

DEMÓ
Stanford sárkány



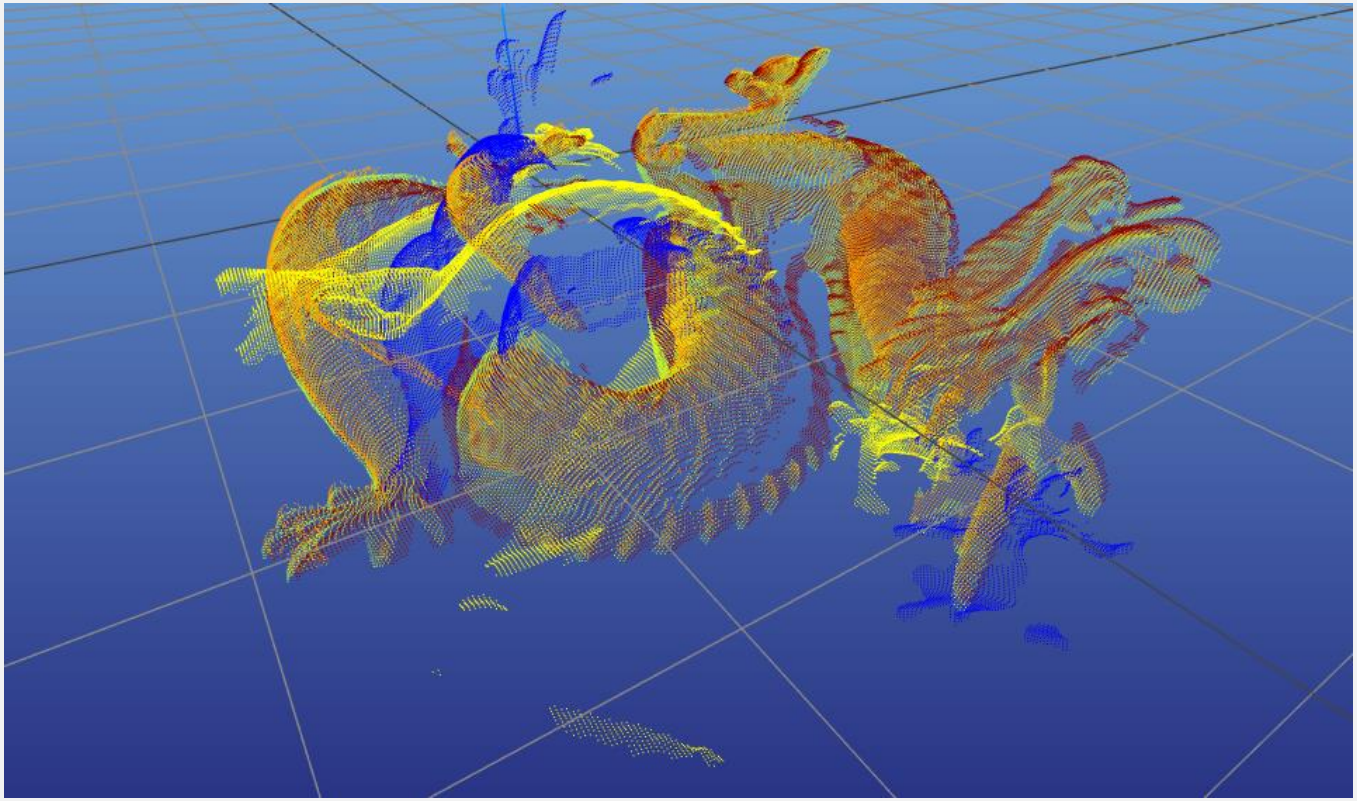
DEMÓ
Stanford sárkány





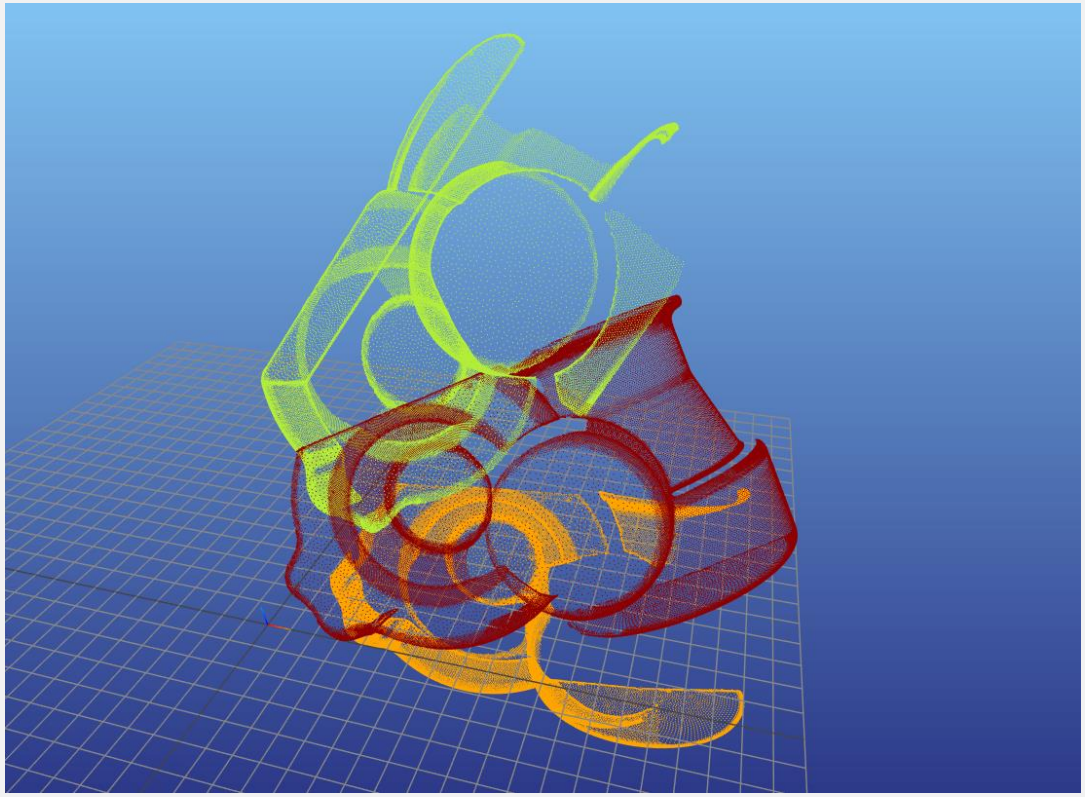
- Az ábra csalóka lehet, mert a futásidő nem determinisztikus a random sample miatt

DEMÓ
Stanford sárkány



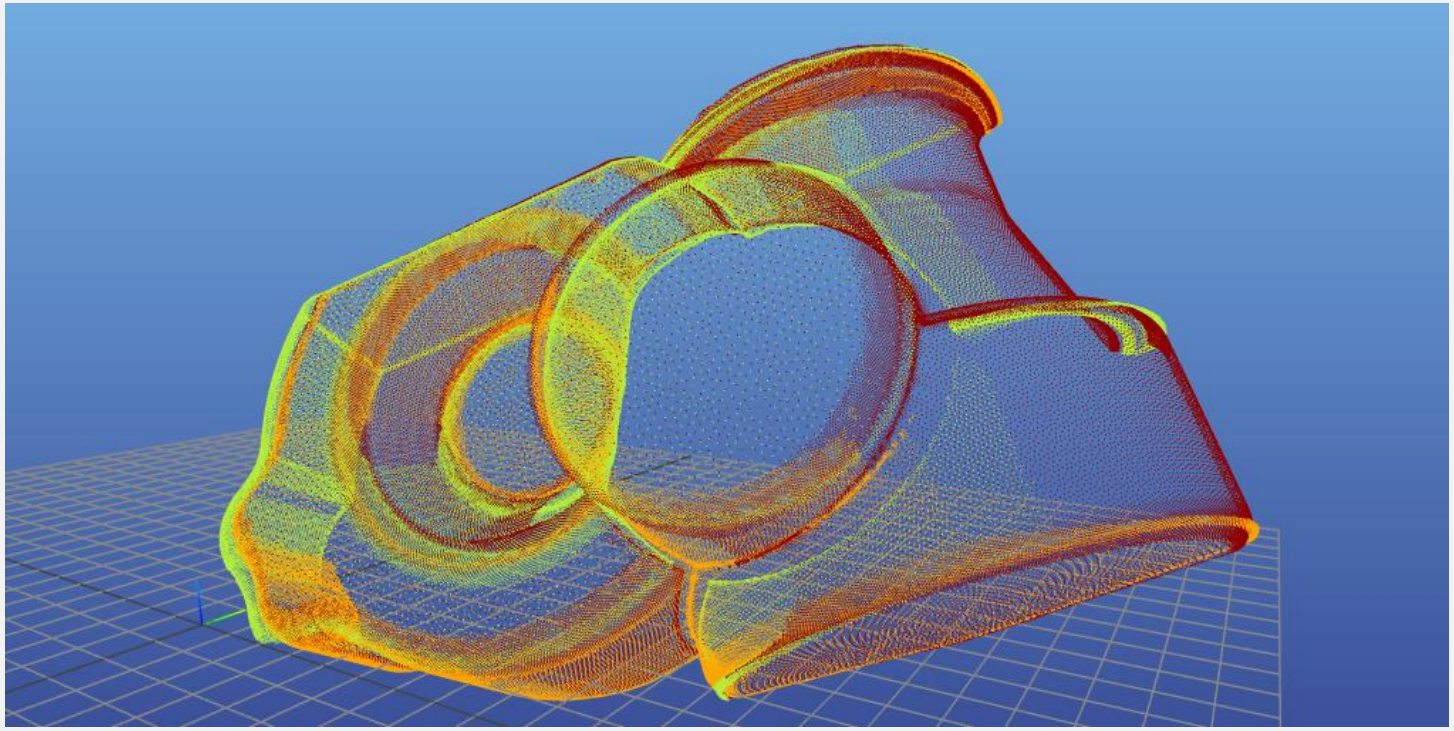
- 24, 48 fok: szépen illeszti
- 72, 96 fok: nem birkózik vele

DEMÓ
Darmstadt modell

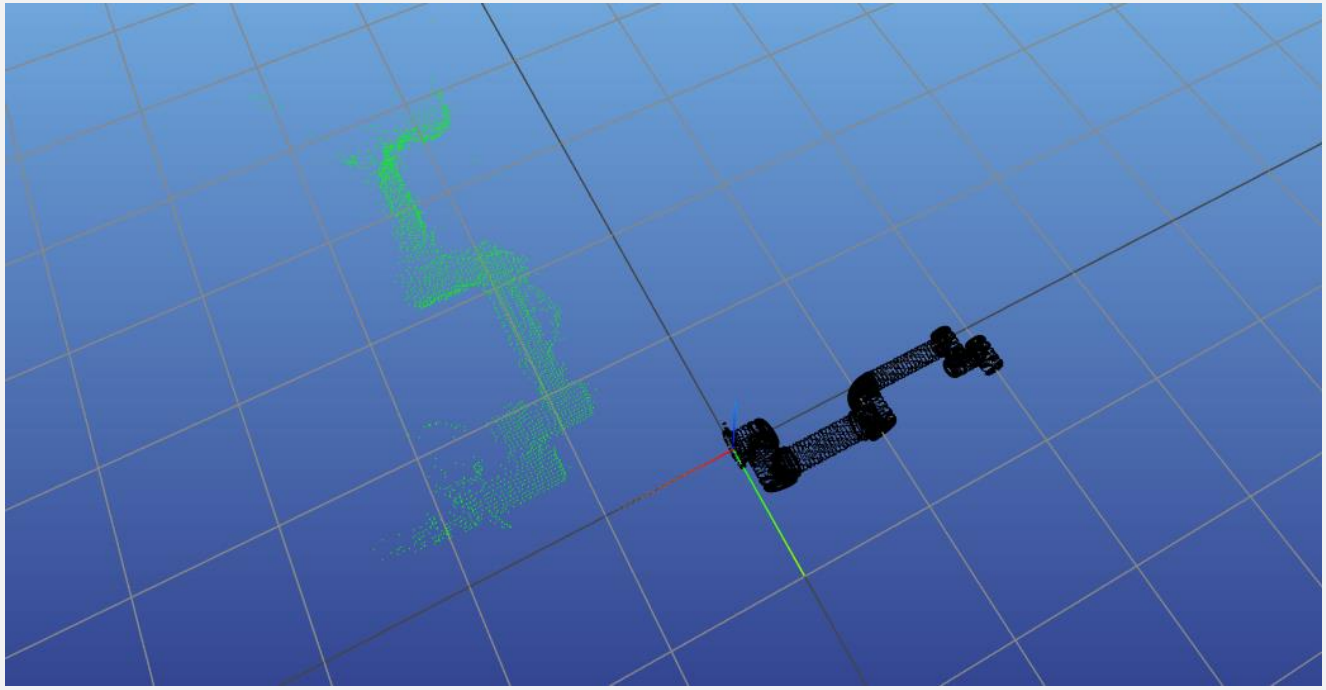


- Részleges fedés, a középsőhöz illesztünk

DEMÓ
Darmstadt modell

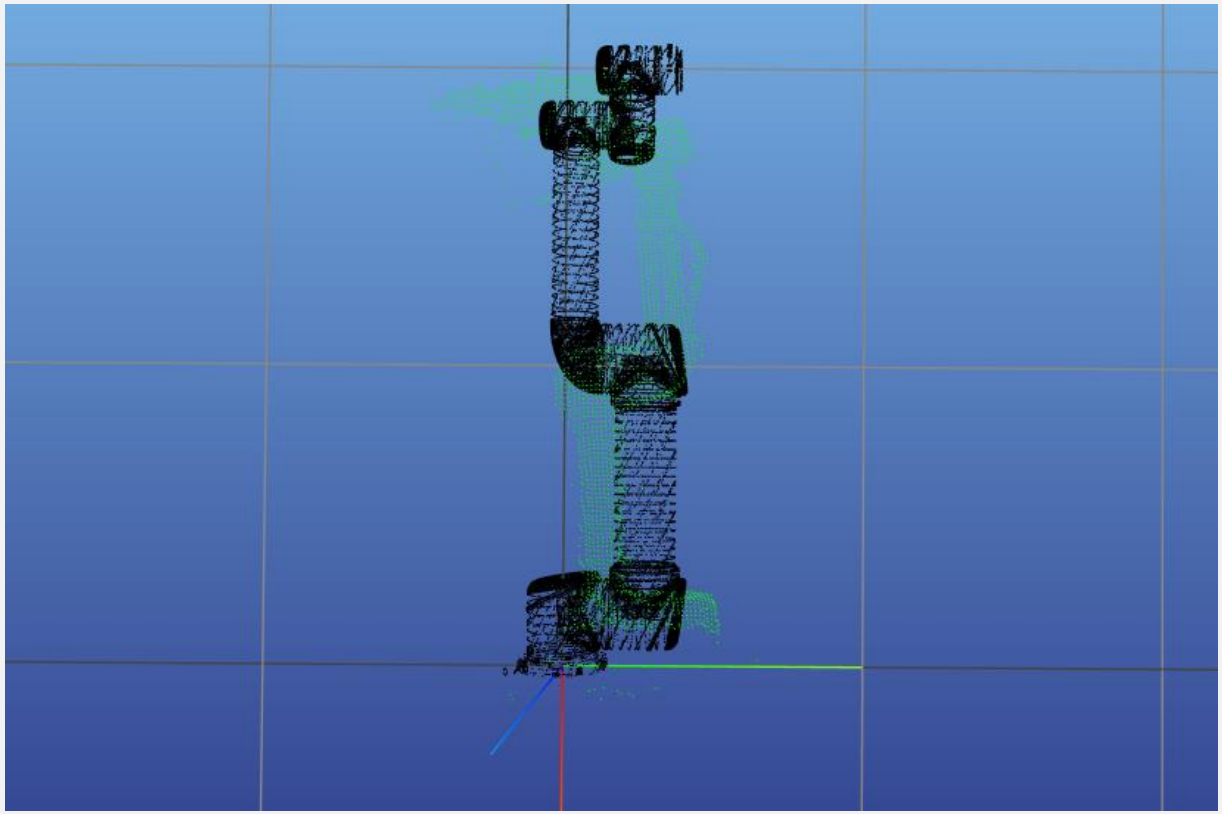


DEMÓ
Universal Robot – UR5



- Kalibráció Kinect pontfelhő segítségével

DEMÓ
Universal Robot – UR5



- Nem áll a kezdeti feltétel: „kellően közeli pontfelhők”

KÖSZÖNÖM A FIGYELMET!

Cserteg Tamás

https://github.com/cserteGT3/PlayGround/tree/master/SzimGeo-ICP