KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PARÇALANMIŞ CİSİMLERİN BİRLEŞTİRİLMESİ

DOKTORA TEZ İZLEME RAPORU

Danışman : Prof.Dr. Vasıf NABİYEV
Jüri Üyesi : Prof.Dr. Mustafa ULUTAŞ
Jüri Üyesi : Prof.Dr. Ali GANGAL

Aralık 2019

PARÇALANMIŞ CİSİMLERİN BİRLEŞTİRİLMESİ
TEZ İZLEME RAPORU

Öğr.Gör. Ömer ÇAKIR

Bilgisayar Mühendisliği Bölümü
Karadeniz Teknik Üniversitesi
cakiro@ktu.edu.tr

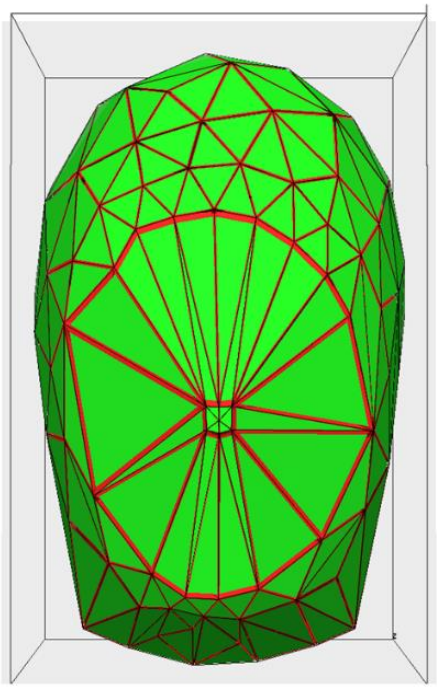
1. GİRİŞ

"Parçalanmış Cisimlerin Birleştirilmesi" konulu doktora teziyle ilgili bu tez izleme raporunda geride kalan 6 ay içinde yapılan çalışmalar şu alt başlıklar halinde sunulacaktır:

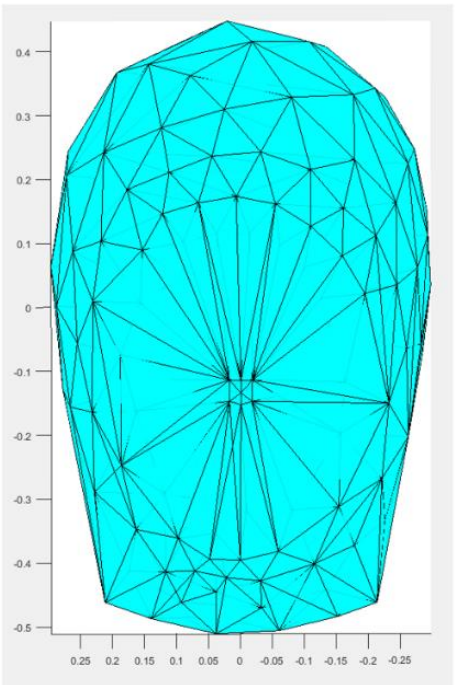
1. Point Cloud formatındaki noktaların üçgenlenmesi (triangulation).
2. Üçgenlenmiş parçaların yüzeylerinin belirlenmesi (segmentation).

2. POINT CLOUD FORMATINDAKİ NOKTALARIN ÜÇGENLENMESİ

Üçgenleme (triangulation) ile ilgili çalışmalardan önceki raporda da bahsedilmişti. MATLAB, TETGEN gibi toollardan yararlanılmaya çalışılmış yalnız iyi sonuçlar elde edilememişti.



```
tetgen -f face.node
```

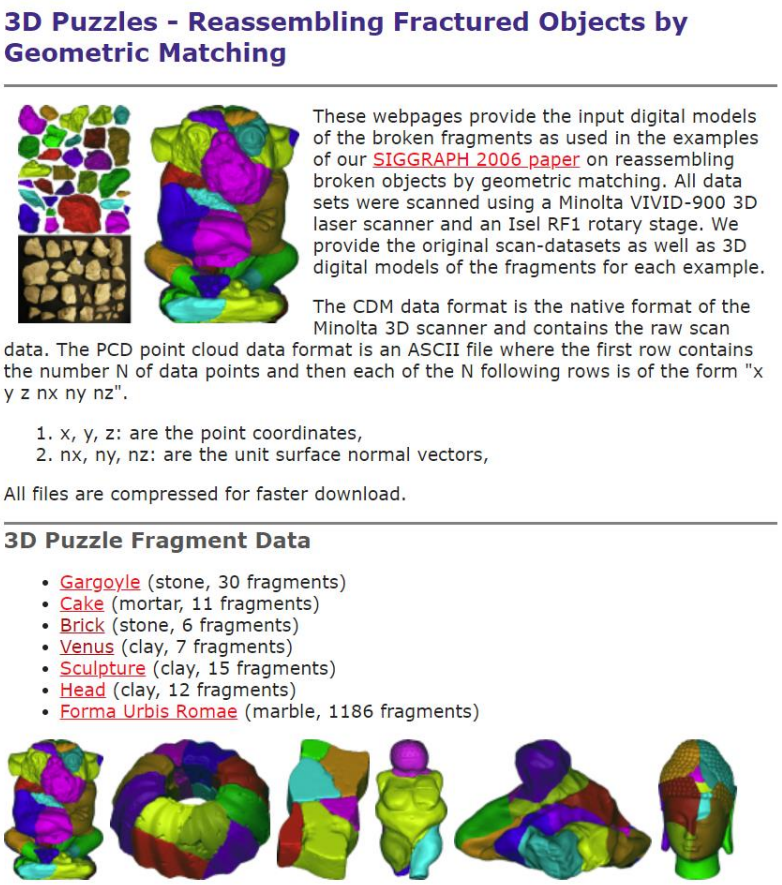



```
dt = delaunayTriangulation(X);
tetramesh(dt, 'FaceColor', 'cyan');
```

Önceki raporda 2 aşamalı kendi üçgenleme algoritmamızdan da bahsetmiştik:

1. aşamada öncelikle birbirine en yakın iki köşe noktası belirlinir ve bunlardan bir kenar üretilir. Sonra bu kenara en yakın köşe noktası ile ilk üçgen üretilir. Sonraki üçgen, ilk üçgenin 3 kenarına en yakın olan köşe noktası ile üretilir. Üçüncü ve sonraki üçgenler mevcut üçgenlerin ortak olmayan (dış) kenarlarına en yakın köşe noktaları ile üretilir. Bütün köşe noktaları için 1. aşama koştuğunda üçgenlenecek 3D modelde boşluklar kalmaktadır. Bu boşlukları da üçgenlemek için algoritmanın 2. aşamasında bu sefer (boşluklardaki) dış kenarlar arasındaki açıya bakılmaktadır. Sırasıyla açısı en düşük olan olan kenarlar üçgenlere çevrilerek boşluklar doldurulmaktadır.

Önceki raporda bahsedilen yukarıdaki algoritmanın en önemli zayıflığı modelin dar girinti/çıkıntılarında göze çarpmaktadır. Örnek kafa modelinin burun kısmındaki problemden önceki tez izleme sunumunda bahsedilmişti. Bu tez izleme raporundaki örnek model gerçek bir cismin laser scanner ile taranmış halidir. Örnek kek modeli geometrie.tuwien.ac.at/ig/3dpuzzles adresinden alınmıştır. Bu adresteki modeller [1] ve [2] yayınlarında kullanılmıştır. Laser scanner ile elde edilen 3D modeller point cloud data (.pcd) şeklinde tutulmaktadır. Yani sadece nokta koordinatları (ve normaller) vardır. Üçgen bilgisi yoktur. O yüzden üçgenleme ile uğraşılmıştır.



3D Puzzles - Reassembling Fractured Objects by Geometric Matching

These webpages provide the input digital models of the broken fragments as used in the examples of our SIGGRAPH 2006 paper on reassembling broken objects by geometric matching. All data sets were scanned using a Minolta VIVID-900 3D laser scanner and an Isel RF1 rotary stage. We provide the original scan-datasets as well as 3D digital models of the fragments for each example.

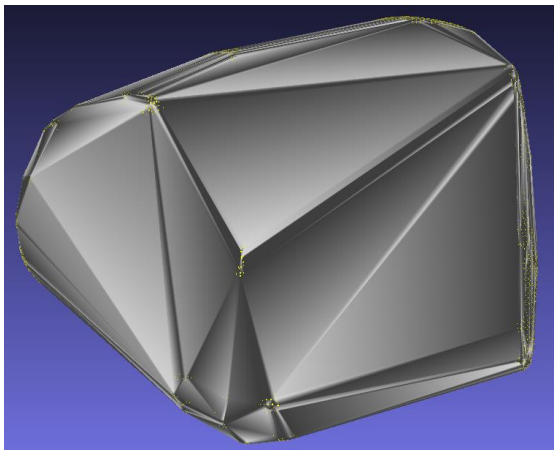
The CDM data format is the native format of the Minolta 3D scanner and contains the raw scan data. The PCD point cloud data format is an ASCII file where the first row contains the number N of data points and then each of the N following rows is of the form "x y z nx ny nz".

1. x, y, z: are the point coordinates,
2. nx, ny, nz: are the unit surface normal vectors,

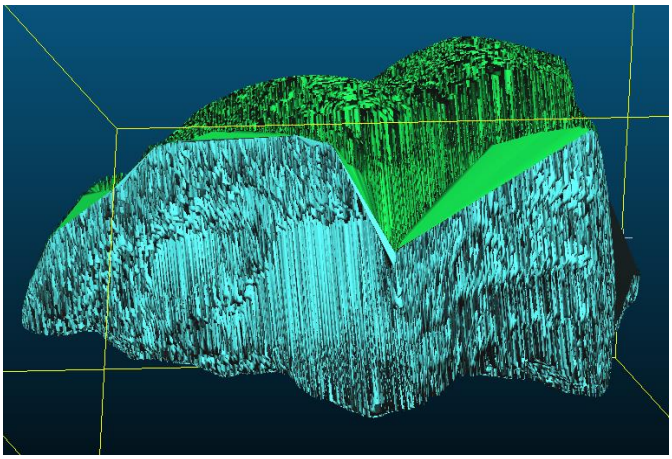
All files are compressed for faster download.

3D Puzzle Fragment Data

- Gargoyle (stone, 30 fragments)
- Cake (mortar, 11 fragments)
- Brick (stone, 6 fragments)
- Venus (clay, 7 fragments)
- Sculpture (clay, 15 fragments)
- Head (clay, 12 fragments)
- Forma Urbis Romae (marble, 1186 fragments)

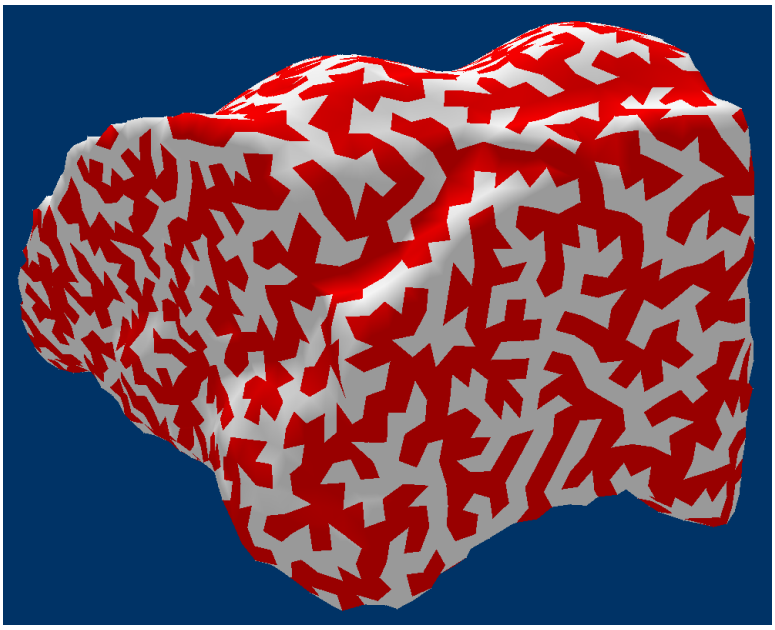


MeshLab

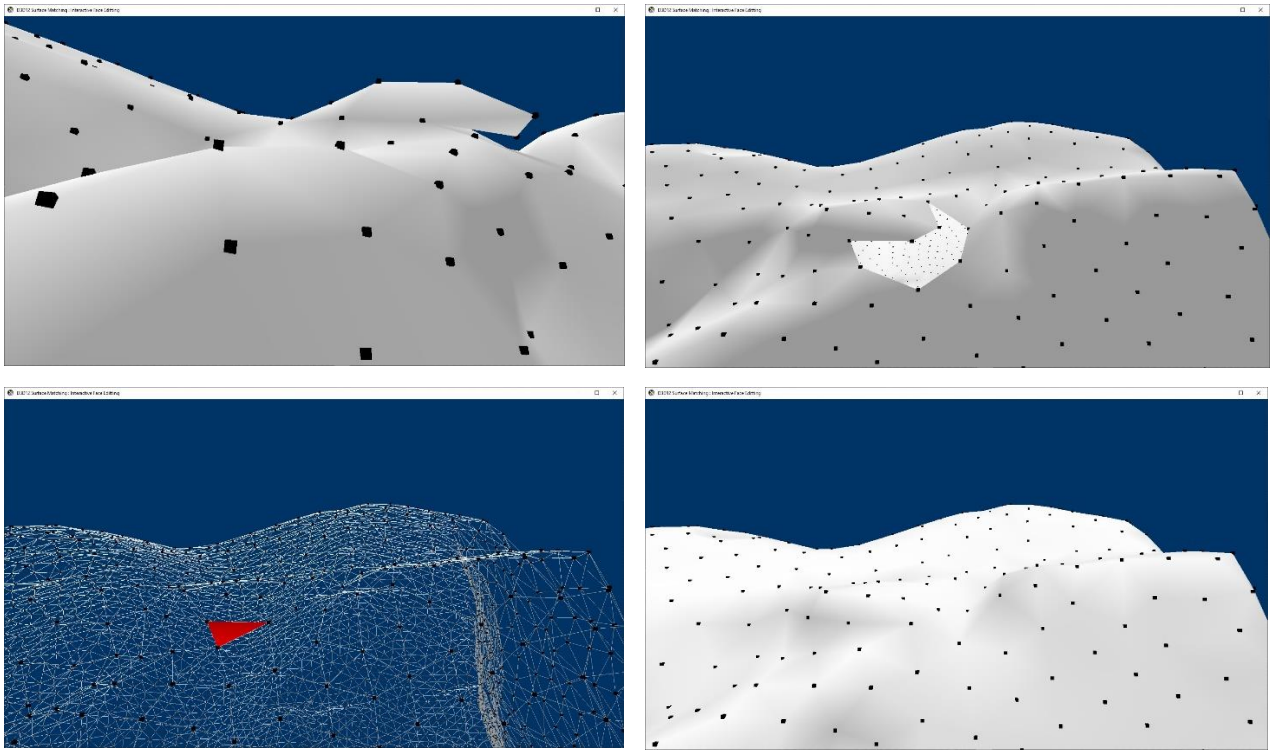


CloudCompare

Kendi üçgenleme algoritmamızın kafa modeli üzerinde istediğimiz başarıyı elde edememesi üzerine MATLAB, TETGEN dışında başka hazır toollar da araştırdık. Bunlardan MeshLab ve CloudCompare programları da istediğimiz sonuçları vermedi. Bizim algoritmamız bu 4 programdan çok daha başarılı sonuçlar üretiyordu. O yüzden algoritmamızı kek modeli üzerinde denedik ve ilginç bir şekilde kafa modelinden çok daha başarılı sonuçlar elde ettik. Kek modelinin laser scanner ile taranması algoritmanın iyi sonuç vermesinde etkili oldu. Çünkü noktaların aralarındaki uzaklık birbirine çok yakındı. Bu da hatalı üçgenlemeyi en aza indirdi. Hatalar yalnızca modelin kenarları ile sınırlı kaldı. Aşağıdaki şekilde kekin 11 parçasından birinin üçgenlenmiş hali verilmiştir. Üçgenlemenin nasıl yapıldığına dair demo videoyu buradan izleyebilirsiniz.

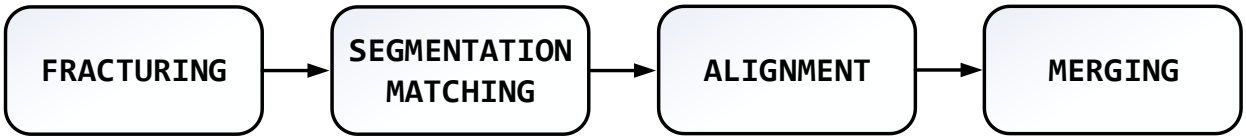


Geliştirdiğimiz üçgenleme algoritmasının modelin kenarlarında hatalar yapabildiğinden (yukarıda) bahsettik. Bu hataları ortadan kaldırmak üzere etkileşimli poligon edit etme toolu yazdık. Bu tool problemli üçgenleri düzeltmek üzere temelde üçgen silme ve ekleme işlemleri yapmaktadır. Hatalı üçgenler silinmekte yerine doğru üçgenler eklenmektedir. Yeni üçgenler üçgenleme algoritmasının 2. aşamasına benzer bir şekilde eklenmektedir. Silinen üçgenlerden oluşan boşlukta aralarındaki açının en düşük olduğu komşu kenarlara sahip üçgenlere tıklanmakta bu kenarlardan yeni üçgen üretilmektedir. Burada dikkat edilmesi gereken önemli nokta : yeni üçgende hangi kenarlar olacaksa onlara yakın bir noktaya tıklanmalıdır. Hatalı üçgenler silinirken bazen *sahipsiz* köşe noktaları ortaya çıkabilmektedir. Sahipsiz köşe noktası ile herhangi bir üçgene ait olmayan boşlukta duran nokta kastedilmektedir. Bu noktalar onlara en yakın kenarlar ile üçgenlenmektedir. Aşağıdaki şekilde, yazdığımız toolun çıktılarından örnekler verilmiştir. Toolun nasıl kullanıldığına dair demo videoyu buradan izleyebilirsiniz.



Üçgenleme algoritmamızın ilk versiyonu kafa modeli için makul bir hızda çalışıyordu çünkü nokta sayısı nispeten azdı. Kek modelindeki parçalarda nokta sayısı çok daha fazla olunca gözle görülür bir yavaşlama oldu. Veri yapısında değişiklikler yaparak algoritma hızlandırıldı. Önceden kenarları tutan dinamik dizide her bir kenar için dış mı içi mi olduğuna dair boolean bir değişken

vardı ve dış olanlar için o anki aktif (herhangi bir üçgene dahil olmamış) köşe noktalarından en yakın olan belirleniyordu. Üretilecek herbir üçgen için sürekli dış kenarlara en yakın köşe noktası araştırılıyordu. Hızlandırılmış versiyonda kenarları tutan dinamik dizide yalnızca dış kenarlar tutuldu. Dolayısıyla boolean değişkene gerek kalmadı. Ayrıca dizide kenarlar köşe noktalarına uzaklıklarına göre sıralı tutuldu. Dolayısıyla üretilecek üçgene ait kenar hep ilk indiste tutuldu. Üçgenler üretildikçe ilk indisler silinerek dizinin gerek yere büyümesi engellendi. Bu da hızlanmayı getirdi.



3. PARÇALARDAKİ YÜZEYLERİN BELİRLENMESİ (SEGMENTATION)

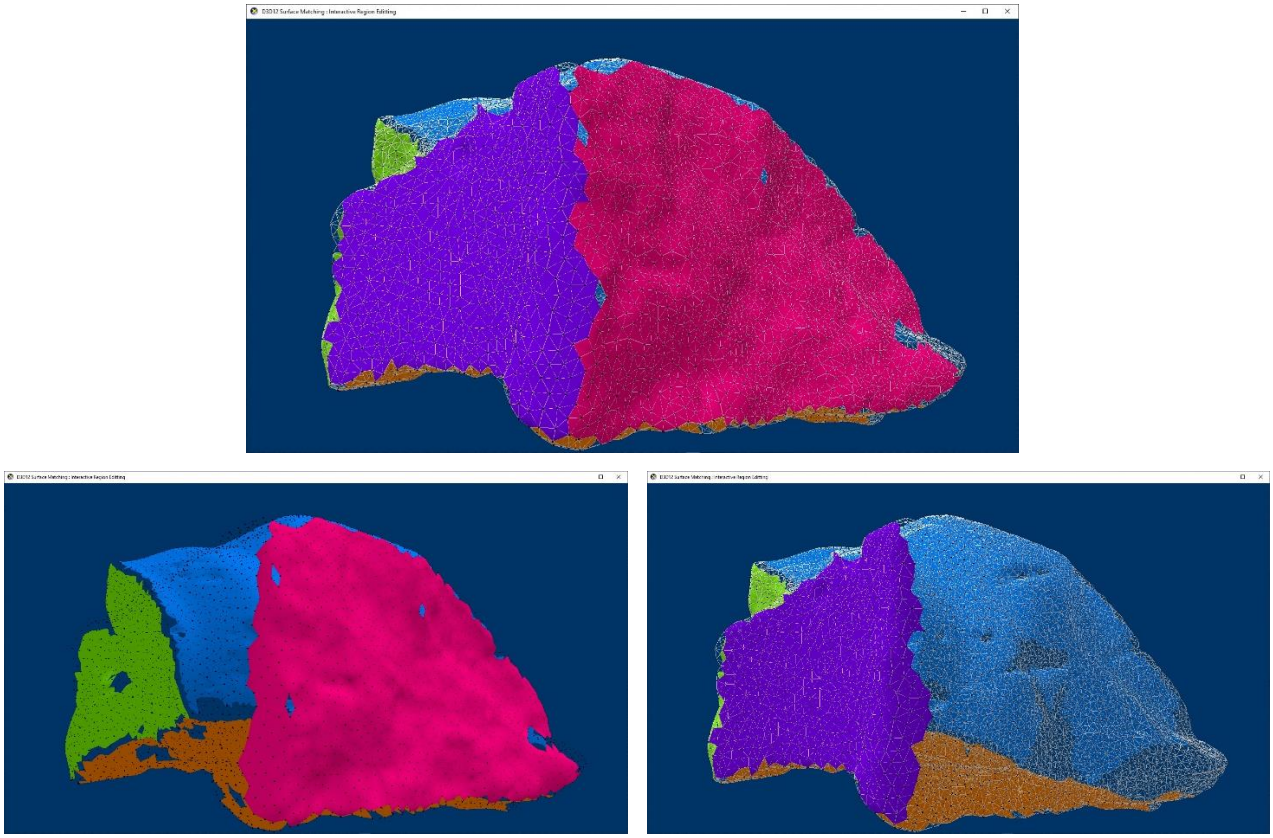
Önceki tez izleme sunumunda örnek vazo modelinin *sentetik* olarak parçalanıp birleştirilme aşamaları yukarıdaki şekildeki gibi verilmişti. Şu an parçalanmış gerçek cisimlerin laser scanner ile taranmış halleri ile uğraştığımızdan fracturing aşamasının doğal yollarla gerçekleştirildiğini varsayıp bu bölümde ikinci aşamanın ilk kısmı olan *segmentation* ile ilgili çalışmalarımızdan bahsedeceğiz.

[1]'deki veritabanını kullandığımızdan yukarıda bahsetmiştik. [1]'de "Data Segmentation" başlıklı 3. bölümde segmentasyonun nasıl yapıldığından bahsedilmiştir. Herbir üçgen için yüzey pürüzlülük (surface roughness) katsayısı tutulmaktadır. Herhangi bir üçgenin o anki segmentasyon bölgesine dahil edilip edilmeyeceğine karar verilirken o üçgenin pürüzlülük katsayısı ve ortalama pürüzlülük katsayısı üzerinden hesaplanan varyans değerinin 0.3'ün altında olup olmadığına bakılmaktadır.

Bizim yarı otomatik segmentasyon algoritmamız da benzer bir yol izlemektedir : Yüzey üzerindeki herhangi bir üçgen seçildikten sonra o üçgene komşu üçgenlerle ortak olmayan köşe noktalarının normallerinin skaler çarpımları hesaplanmaktadır. Skaler çarpımı en büyük olan üçgen ilk üçgene eklenerek yüzey bölgesi büyütülmeye (region growing) başlanır. Sonraki adımlarda da üçgenlemeye benzer şekilde mevcut yüzey bölgesinin kenarlarındaki üçgenler için yine onlara komşu olan üçgenlerde ortak olmayan köşe noktaları ile arlarındaki skaler çarpımlardan max. değere sahip olan yüzey bölgesine eklenerek yüzey genişletme işlemine devam edilir. Yüzey genişletmeyi sonlandırma kriteri olarak skaler çarpımlardan dönen max değerin açısal olarak 45 derecenin üstüne çıkması olarak belirledik. Yani yüzeyin dış kenarlarındaki tüm üçgenler için onlara komşu olan

üçgenlerin tamamı ile aralarındaki skaler çarpımlarda 45 derecenin üstüne çıkılıyorsa yüzey genişletme sonlandırılır.

Segmentation algoritmamızın koştuğu demoya ait videoyu buradan izleyebilirsiniz. Videoda 02:48 'den itibaren tuşlarla ön yüzey yok edilip tekrar render edilmektedir. Bundan amaç ön yüzey ile sağ yüzeyin bir miktar üst üste bindiğini göstermektedir. Algoritmamız çok başarılı sonuçlar elde etmekle birlikte az da olsa bazı yüzeyler arasında çakışmalar olabilmektedir. Çakışan üçgenleri yüzeylerden sadece birine dahil etmek için de yine etkileşimli bir tool yazılmıştır.



Çalışmamızın bundan sonraki aşamasında yüzeylerden elde edeceğimiz ağırlıklandırılmış minimum cost spanning tree'ler üzerinde koşacak bir graph matching algoritması geliştireceğiz. Bu algoritma ile yüzeyler arasındaki ortak bölgeleri (MATCHING) belirlemeye çalışacağız.

4. KAYNAKLAR

[1] Q. X. Huang, S. Flry, N. Gelfand, M. Hofer, H. Pottmann, "Reassembling fractured objects by geometric matching". ACM Trans. Graph. (TOG) 25(3), 569–578, 2006.

[2] T. Son, J. Lee, J. Lim, K. Lee, "Reassembly of fractured objects using surface signature", Visual Computer 34:1371–1381, 2018.