DOKUZ EYLÜL ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYARLI GÖRÜNTÜ İŞLEME İLE 3–BOYUTLU NESNE TARAMA VE MODELLEME

Nihat Engin TOKLU

Nisan, 2009
İZMİR

BİLGİSAYARLI GÖRÜNTÜ İŞLEME İLE 3–BOYUTLU NESNE TARAMA VE MODELLEME

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü

Yüksek Lisans Tezi

Mekatronik Mühendisliği Bölümü, Mekatronik Mühendisliği Anabilim Dalı

Nihat Engin TOKLU

Nisan, 2009

İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

NİHAT ENGİN TOKLU tarafından **PROF. DR. EROL UYAR** yönetiminde hazırlanan **"BİLGİSAYARLI GÖRÜNTÜ İŞLEME İLE 3-BOYUTLU NESNE TARAMA VE MODELLEME"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

.....

Prof. Dr. Erol UYAR

Danışman

.....	.....
Yrd. Doç. Dr. Zeki KIRAL	Yrd. Doç. Dr. Metehan MAKİNACI
Jüri Üyesi	Jüri Üyesi

_____

Prof. Dr. Cahit HELVACI

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Çalışmalarım sırasında sundukları faydalı önerileri için arkadaşlarım Onur KESKİN ve Meriç KAYALIK'a, yönlendirmeleri için proje danışmanım Prof. Dr. Erol UYAR'a, her türlü desteği için de aileme teşekkürü bir borç bilirim.

Nihat Engin TOKLU

BİLGİSAYARLI GÖRÜNTÜ İŞLEME İLE 3-BOYUTLU NESNE TARAMA VE MODELLEME

ÖZ

Gerçek dünyaya ait olan bir nesnenin yüzeyinden, üç boyutlu şekil bilgisini elde etmek amacıyla lazerle geçilmesi işlemine lazer taraması adı verilmektedir. Tıp alanında hastaların ölçülerinin alınması, fabrikalarda seri üretim bandındaki nesnelerin yüzeylerindeki hataların bulunması, hareketli robotların çevrelerindeki yüzeyleri algılaması gibi bir çok alanda, lazer taraması yöntemi kullanılmaktadır.

Lazer taraması yöntemiyle farklı endüstrilerin gereksinimleri karşılamak için, farklı yüzey tarayıcı donanım ve yazılım ürünleri bulunmaktadır. Ancak, bu ürünlerden bazıları, kendi belirledikleri amaç doğrultusunda tasarlandıkları için, her türlü yüzey tarama görevini üstlenememektedirler. Örneğin, seri üretim bandındaki ürünlerin şekil hatalarını yakalaması için tasarlanmış bir üç boyutlu lazerli tarayıcı, sadece o iş için tasarlanmış olabilir ve bir hastanede hastaların ölçülerini almak için ya da bir robotun çevresini algılaması için kullanılamayabilir.

Bu tez, lazer üçgenlemesi (“Laser Triangulation”) yöntemiyle yüzey tarama konusunda yapılan bir çalışmayı sunmaktadır. Bu uygulamada, dikey bir lazer ışın demeti ile aydınlatılan yüzeyin görüntüleri, kamera aracılığıyla bilgisayara aktarılarak, tez çalışması sırasında yazılmış olan “YakarTarar” adlı bir yazılım tarafından işlenmektedir. Bu görüntülerin işlenmesi sonucunda, taranan yüzeyin üç boyutlu sayısal modeli bilgisayara kaydedilmektedir.

YakarTarar yazılımının amacı, hem farklı endüstrilerde yüzey tarama gereksinimlerini aynı anda karşılayabilen, hem de bu iş için gereken maliyeti düşük

tutan bir yazılım paketi olmaktır.

Anahtar Sözcükler: Lazer Üçgenlemesi, Üç Boyutlu Modelleme, Yüzey Tarama, Lazer Taraması

3-DIMENSIONAL OBJECT SCANNING AND MODELLING VIA COMPUTER VISION

ABSTRACT

The operation of passing a laser beam on the surface of a real-world object for acquiring its 3D shape data is called laser scanning. Laser scanning has usages like acquiring a patient's 3D body model for a medical operation, finding the shape errors of the objects in a production line, or a robot's discovery of its environment's 3D shape.

To meet the requirements of the different industries, different laser scanner software or hardware products exist. However, some of these products are designed for their special purposes and cannot be used for other purposes. For example, a laser scanner designed to find the shape errors of objects on a production line, might be designed only for that purpose so that it can't be used to acquire the shape information of a patient in a hospital or it can't be used by robot as an environment sensor.

This thesis presents a study on laser scanning using the laser triangulation method. In this application, the images of a surface marked by a vertical laser beam are recorded by a camera and processed by a software called “YakarTarar”, which was developed during the studies. As a result of this processing, the 3D model of the surface is saved in the computer.

The goals of YakarTarar software are to meet different requirements of different industries and to have a minimal base cost.

Keywords: Laser Triangulation, 3D Modelling, Surface Scanning, Laser Scanning

İÇİNDEKİLER

Sayfa

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU..... ii
TEŞEKKÜR iii
ÖZ iv
ABSTRACT vi

BÖLÜM BİR – GİRİŞ 1

1.1 Üç Boyutlu Model Elde Etme Hakkında 1
1.2 YakarTarar Yazılım Projesinin Amacı 1

BÖLÜM İKİ – YAKARTARAR YAZILIM PROJESİNE GENEL BAKIŞ... 3

2.1 YakarTarar Kullanıcı Arayüzü 3
2.2 YakarTarar C++ Kütüphanesi 4
2.3 YakarTarar Python Kütüphanesi 5

BÖLÜM ÜÇ – YAKARTARAR'IN ÇALIŞMA YÖNTEMİ 6

3.1 Yönteme Genel Bakış 6
3.2 YakarTarar'ın Kalibrasyon Yöntemi 8
3.3 YakarTarar'ın Tarama Algoritması 11

BÖLÜM DÖRT – YAKARTARAR'IN UYGULAMALARI 12

4.1 Tarama Sonuçları Hakkında 12

4.2 Sonuçları Başka Yazılımlara Aktarmak …..... 14
4.3 YakarTarar'ın C++ Kütüphanesine Erişmek …..... 17
4.4 YakarTarar'la Tarama Sonuçlarındaki Hataları Düzeltmek …..... 20
4.5 YakarTarar'ı Gerçek Zamanlı Algılayıcı Olarak Kullanmak …..... 25
4.6 YakarTarar Sonuçlarını Gauss Yöntemiyle Bulandırmak …..... 27
4.7 Bir Yüzey Taraması Sonucundaki Ana Nesneyi Kirlerden Ayırmak..…..... 29
4.8 İki Tarama Sonucunu Karşılaştırmak.....…..... 31

BÖLÜM BEŞ – YAKARTARAR'IN İÇ YAPISI …..... 32

5.1 Yapıya Genel Bakış …..... 32
5.2 Kalibrasyon Yapısı …..... 33
5.2.1 X ve Z Vektörleri …..... 33
5.2.2 “satır_başına_düşen_birim” Değişkeni …..... 34
5.3 Tarama Sütunu Yapısı …..... 35
5.4 Tam Yüzey Tarama Yapısı …..... 35
5.4.1 Kalibrasyon Bilgileri …..... 36
5.4.2 İlerletme ve Döndürme Bilgileri …..... 36
5.4.3 Sütun Dizisi …..... 36
5.4.4 “kaçıncı_satırda(y)” İşlevi …..... 37
5.4.5 “kaçıncı_sütunda(x)” İşlevi …..... 37
5.4.6 “xyz_konumunu_bul(j, i)” İşlevi …..... 38
5.4.7 Yüzey Düzeltme İşlevi …..... 38
5.4.8 Modeli Dışa Aktarma İşlevi …..... 39
5.5 Gauss Yöntemiyle Bulandırıcı Modül …..... 40
5.6 Nesne Ayrıştırıcı Modül …..... 41
5.7 Nesne Karşılaştırıcı Modül …..... 42

BÖLÜM ALTI – SONUÇLAR..... 44

KAYNAKÇA..... 45

BÖLÜM BİR
GİRİŞ

1.1 Üç Boyutlu Model Elde Etme Hakkında

Gerçek dünyaya ait olan nesnelerin üç boyutlu modellerinin elde edilmesi, bir çok endüstriyel alanın gereksinimidir (Seitz, Curless, 1999). Bu alanlardan bazıları aşağıda verilmiştir:

- **Seri üretim:** Üretilmiş olan parçalar, üretim bandından geçerken, lazer ile taranarak üç boyutlu şekil bilgileri alınır. Daha sonra üç boyutlu şekil bilgileri incelenerek, şekil bozuklukları, dolayısıyla üretim hataları, varsa tespit edilir (Petrovic, 2007).
- **Sağlık bilimi:** Bir uzvu eksik olan hasta için protez üretilirken, protezin vücuttaki en doğru konumu ve vücuda göre en uygun boyutu bulunmalıdır. Bunun için, vücudun protezle ilgili olan bölümleri lazerle taranır ve üç boyutlu şekil bilgisi elde edilir. Bu şekil bilgisinden yararlanarak, uygun ölçülerde protez üretilir ve hastanın kullanımına sunulur (Coward, Scott, Watson, Richards, 2002).
- **Hareketli cihazlar:** Hareketli cihazlar, çevrelerini lazerle tarayarak hareketlerini engelleyecek yüzeyleri tespit ederler ve ona göre yeni yönlerini belirlerler (Klančnik, Balič, Planinšič, 2007; Roberts, Corke, 2000).

1.2 YakarTarar Yazılım Projesinin Amacı

Üç boyutlu model elde etmenin bu kadar kullanım alanı olması, farklı yüzey tarayıcıların piyasaya sürülmesine sebep olmuştur. Piyasadaki bazı yüzey tarayıcılar ise, bazı sorunları beraberinde getirebilmektedir:

- Bazı üç boyutlu tarayıcılar, belirli bir amaç için tasarlandıklarından dolayı, genel amaçlı kullanılamazlar. Örneğin, bir firma, seri üretimde gerçek zamanlı kalite kontrolü için bir yüzey tarayıcı kullanırken, aynı tarayıcıyı nesne modeli çıkartma amacıyla kullanamayabilir. Bu durumda, nesne modeli çıkartmak için farklı bir yüzey tarayıcı sipariş etmek zorunda kalır.
- Bazı yüzey tarayıcılar, bazı firmalar için tam istenen kaliteyi sağlıyorken, bazı firmalar için de fazla pahalı bir çözüm sunuyor olabilirler. Bu durum, pahalı tarama çözümlerinin yanında, ucuz tarayıcı yazılımlara olan gereksinimi doğurmuştur (Winkelbach, Molkenstruck, Wahl, 2006).

Bu belirtilen sorunları çözmek için, "YakarTarar" adlı bir yazılım geliştirilmiştir. Getirdiği çözümler aşağıdaki gibidir (Uyar, Toklu, 2008):

- YakarTarar, yüzey taramasını, kullanıcının belirlediği herhangi bir kamera aracılığıyla yapabilmektedir. Bu durumda, fiyat-kalite konusundaki ayarlama kullanıcının eline geçmiş olur
- YakarTarar, belli bir alana yönelmek yerine, farklı alanlarda kullanılabilecek genel amaçlı yüzey tarama işlevlerini, bir yazılım kütüphanesinde barındırmaktadır. Bu durumda, daha geniş bir kullanım alanına sahip olur.

BÖLÜM İKİ
YAKARTARAR YAZILIM PROJESİNE GENEL BAKIŞ

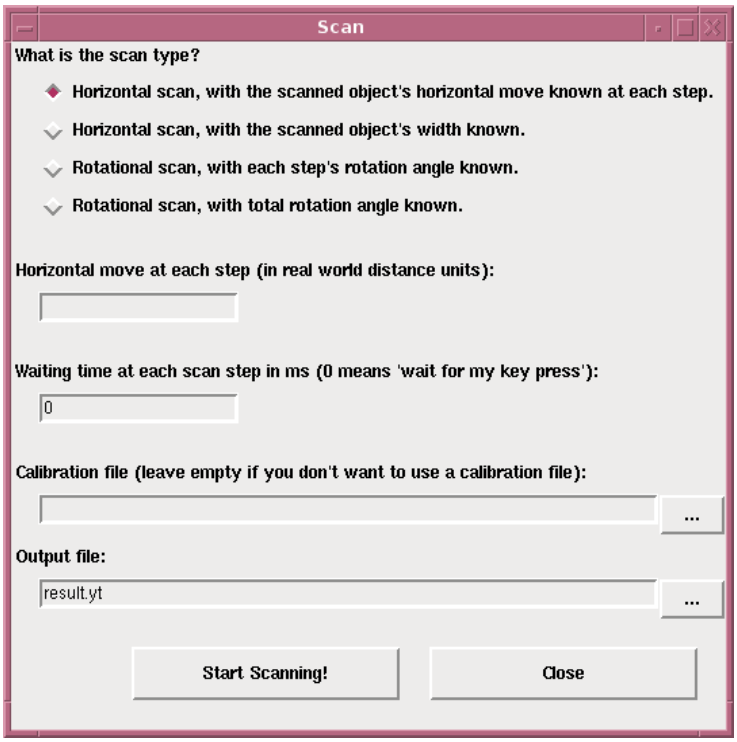
YakarTarar projesi, dikey bir lazer demetiyle aydınlatılan bir yüzeyin görüntülerini kamerayla alarak üç boyutlu şekil bilgisi elde etme işlevlerini barındıran bir yazılım paketidir. Bu paket aşağıdaki bileşenlerden oluşur:

- **YakarTarar Kullanıcı Arayüzü:** YakarTarar, kullanıcıların tarama işlemlerini yapmaları için bir arayüz sunar. Bu arayüz sayesinde kalibrasyon ve tarama gibi temel işlemler yapılabilmektedir.
- **YakarTarar C++ Kütüphanesi:** C++ programlama dilinde hazırlanmış olan bu yazılım kütüphanesi, YakarTarar'ın temelini oluşturmaktadır. Bu kütüphane, YakarTarar'ın tüm işlevlerini barındırmaktadır. Bu kütüphane sayesinde, yüzey tarama işlemleri gerçekleştiren özel amaçlı yazılımlar kolayca hazırlanabilmektedir.
- **YakarTarar Python Kütüphanesi:** YakarTarar'ın tarama sonuçlarının Python programlama dilinde incelenebilmesi amacıyla, YakarTarar Python kütüphanesi de geliştirilmiştir.

YakarTarar yazılım paketi, Windows ve Linux tabanlı işletim sistemlerinde, işletim sistemi tarafından tanınan herhangi bir kamera ile çalışabilmektedir.

2.1 YakarTarar Kullanıcı Arayüzü

YakarTarar kullanıcı arayüzü, YakarTarar'ın işlevlerini kullanıcıya doğrudan sunan bileşendir. Bu arayüzü kullanarak bir kullanıcı kalibrasyon yapabilir, bir nesnenin yüzeyini tarayarak üç boyutlu modelini elde edebilir (bkz. Şekil 2.1) ve sonuçları kaydedebilir.



Şekil 2.1 YakarTarar'ın kullanıcı arayüzünün nesne tarama ekranı

2.2 YakarTarar C++ Kütüphanesi

Bir yüzey taraması işlemi, tarama çeşidine göre farklı düzeneklere gereksinim duymaktadır. Bu düzeneklerden bazıları aşağıda verilmiştir:

- Taranacak nesneyi yürüyen bantta yatay olarak ilerleten bir düzenek
- Taranacak nesneyi olduğu yerde döndüren bir düzenek
- Hareketli bir cihazın (örneğin hareketli robot ya da uzaktan kumandalı araba) önünde takılı olan lazer demetinin yol üstünde bıraktığı izi, yine cihaza monte edilmiş kamera ile okutup kablosuz iletişim aracılığıyla bilgisayara aktaran bir düzenek

Düzeneğe göre ise, bilgisayarın düzenekle olan iletişimi de değişir. Bu iletişim paralel port, seri port, USB, kablosuz iletişim olabilir.

Bu farklı düzeneklerin, farklı şeylere farklı tepkiler veren özel amaçlı yazılımlara gereksinimleri vardır. YakarTarar'ın, bütün işlevlerini bir kütüphane olarak sunması sayesinde, kurulmuş olan düzeneğe uyum sağlayacak bir yazılım, çok kısa sürede oluşturulabilmektedir. YakarTarar C++ kütüphanesini kullanarak oluşturulan yazılımlar, aşağıdaki becerilere sahip olurlar:

- Kalibrasyon yapmak
- Tarama yapmak
- Tarama sonuçlarını kaydetmek
- Tarama sonuçlarını başka yazılımların okuyabileceği dosya türlerine çevirmek:
- Üç boyutlu tasarım programları için VRML (“.wrl”) veya Wavefront (“.obj”) dosya türü
- MATLAB ve Octave için “.m” dosya türü

C++ dilinde, Intel'in OpenCV kütüphanesini kullanarak yazılmış olan bu kütüphane, YakarTarar'ın temelini oluşturmaktadır.

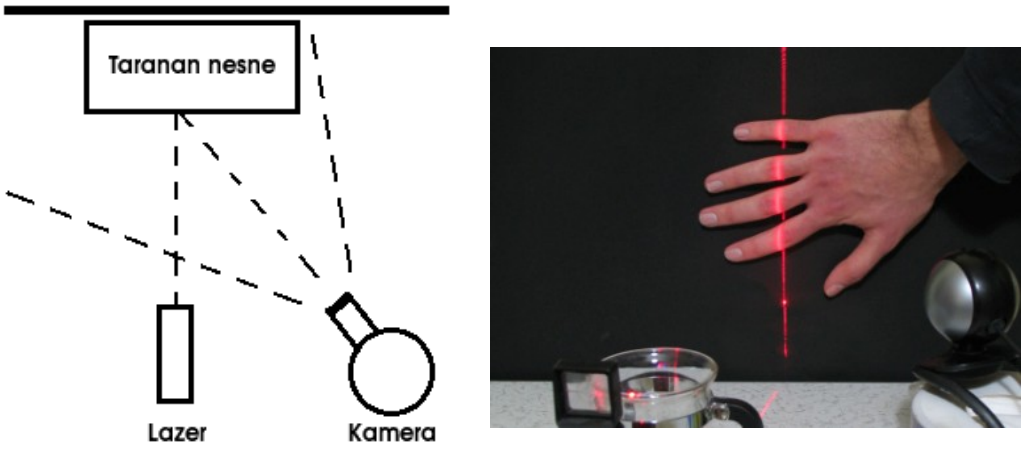
2.3 YakarTarar Python Kütüphanesi

YakarTarar'ın sonuç dosyaları üzerinde, Python programlama dilinde inceleme ve hesap yapmayı mümkün kılmak için, YakarTarar Python kütüphanesi de yazılmıştır. Python programlama dili, yüksek seviyeli bir dil olarak kabul edilmektedir ve programcının üretkenliğini artırmaktadır (Python Software Foundation, 2008). Bu kütüphane, bu üretkenlikten faydalanarak, sonuç dosyaları üzerinde yapılacak hesap algoritmalarının kolayca programa dökülmesini sağlamaktadır.

BÖLÜM ÜÇ
YAKARTARAR'IN ÇALIŞMA YÖNTEMİ

3.1 Yönteme Genel Bakış

YakarTarar, lazer üçgenlemesi (“Laser Triangulation”) adı verilen bir yöntem kullanmaktadır. Bu yönteme göre, lazerin yüzey üstünde oluşturduğu parlak iz, lazer kaynağından farklı bir açıdan bakan bir kamera tarafından algılanır. Algılanan görüntüdeki parlak izin konumunun ve şeklinin incelenmesiyle, yüzeyin lazerle aydınlatılmış kısmının şekil bilgisi elde edilir (Lathrop, 1997; Laser Design Inc., 2008). YakarTarar'ın gereksinim duyduğu yerleşim düzeni, Şekil 3.1'de gösterilmektedir.

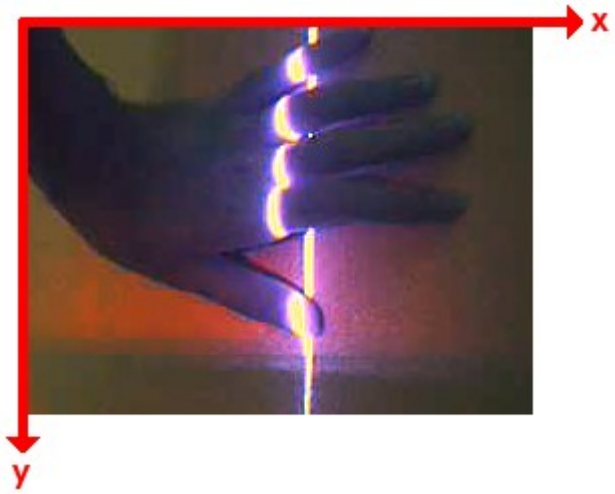


Şekil 3.1 Lazer üçgenlemesi yöntemine göre kamera, lazer kaynağı ve taranan nesnenin yerleşim düzeni

Bu düzene göre, lazer kaynağı, nesne üzerine dikey bir çizgi şeklinde ışın demeti yollar. Bu demet, nesnenin yüzeyinde, yüzeyin şeklini alan parlak bir iz oluşturur (Şekil 3.2). Yüzeyin şekli, kamera tarafından kaydedilen görüntülerdeki bu parlak izin bükülmelerinin incelenmesi aracılığıyla çıkartılır.

Şekil 3.2 Taranmakta olan bir insan elinin yüzeyinde dikey lazer ışın demetinin oluşturduğu iz.

YakarTarar, kamera aracılığıyla aldığı görüntüyü, Şekil 3.3'te görünen iki boyutlu uzayda yeniden oluşturur.



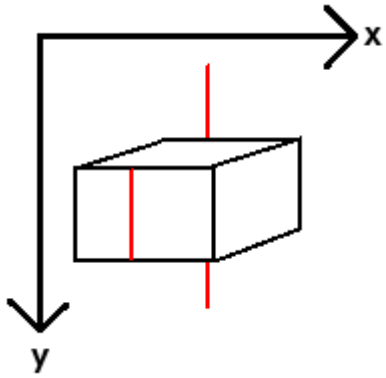
Şekil 3.3 Kamera tarafından alınan görüntünün, YakarTarar tarafından oluşturulan iki boyutlu uzaydaki duruşu. Buradaki x ve y düzlemleri, bilgisayarlardaki piksel yerleşim düzenine göre oluşturulmuştur. Buna göre, x ekseni sağ yönde, y ekseni aşağı yöndedir. (0,0) noktası sol üsttedir.

Şekil 3.2 ve Şekil 3.3'teki parlak iz, üçüncü boyutu belirler. YakarTarar, bu üçüncü boyutu duvardan uzaklık olarak tanımlar ve z olarak adlandırır. Yüzeyin, dikey lazer çizgisiyle aydınlatılan sütunundaki her satırı için bir z değeri ortaya çıkmaktadır. Bir nesnenin üç boyutlu modelinin bütünüyle oluşturulması için, her sütunun aynı yöntemle z değerleri çıkartılır. Bunu sağlamak için, nesne, başından sonuna kadar yatay olarak ilerletilir ya da döndürülür.

3.2 YakarTarar'ın Kalibrasyon Yöntemi

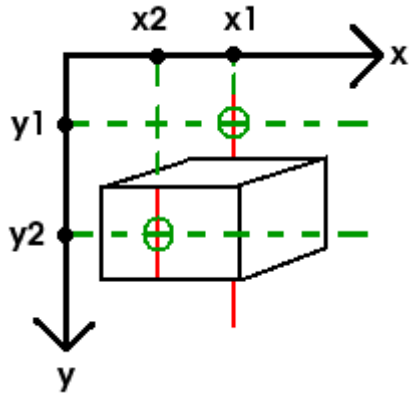
Bir nesneyi taramadan önce, z değerlerinin doğru ölçülebilmesi için kalibrasyon işleminin yapılması gerekmektedir. Kalibrasyon işlemi sayesinde, lazer demetinin oluşturduğu kırmızı iz ile yüzeyin kameraya yakınlığı arasında doğrusal bir ilişki kurulur.

Kalibrasyon için, boyutları bilinen bir cisim, tarama ortamına konur ve lazerin bu nesne üzerinde iz bırakması sağlanır (Şekil 3.4).



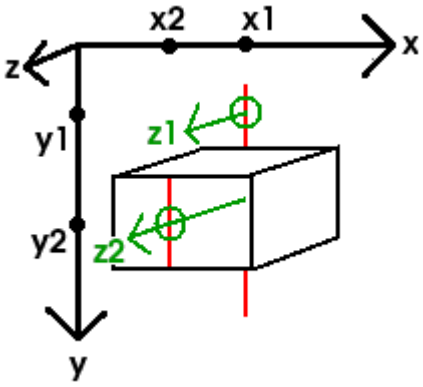
Şekil 3.4 Dikdörtgenler prizması şeklindeki bir kalibrasyon nesnesinin, kamera tarafından algılanmış hali.

Sonra, kullanıcı, alınan görüntü içinde iki ya da daha fazla satır işaretler. YakarTarar, lazerin oluşturduğu kırmızı izin, seçilen satırlarda hangi x konumundan geçtiğini bulur (Şekil 3.5).



Şekil 3.5 Kalibrasyon sırasında, iki satır seçildikten sonraki durum.

Daha sonra, kullanıcı, işaretli noktaların duvardan ne kadar uzakta olduğunu, yani z değerlerini girer (Şekil 3.6).



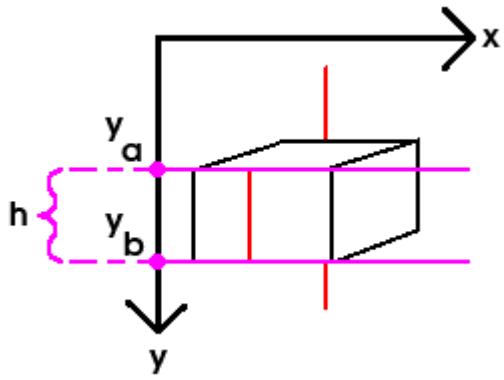
Şekil 3.6 Kalibrasyon sırasında, seçilen noktaların z (duvardan uzaklık) değerleri

Parlak izin, bir satırdaki yatay konumu x, o noktanın duvardan uzaklığı z olduğuna göre, x ve z arasında doğrusal bir ilişki kurulabilir. YakarTarar, bu ilişkiyi şu şekilde kurar:

```
z_eğimi = (z2 – z1) / (x2 – x1)
z_değeri(x) = (z_eğimi * (x – x1)) + z1
```

Eğer kalibrasyon için ikiden fazla nokta seçilmişse, x değerleri sıralı bir biçimde dizilir, her x arası farklı bir bölge kabul edilir ve her bölge için ayrı eğim hesaplanır.

Kalibrasyonun son adımında, kullanıcı resim içinde iki satır daha belirler. Daha sonra bu iki satırın gerçek dünyadaki dikey uzaklığını girer (Şekil 3.7).



Şekil 3.7 Kalibrasyon sırasında, YakarTarar'ın yükseklik kavramını elde etmesi için, iki satır seçilir (y_a ve y_b) ve iki satırın gerçek dünya birimiyle birbirinden uzaklığı belirtilir.

Şekil 3.7'deki bu iki satır sayesinde, bir pikselin kaç gerçek dünya uzaklık birimine denk geldiği şu şekilde bulunur:

$$h = \text{dikey\_piksel\_uzaklığı} = (y_a - y_b)$$

$$\text{piksel\_başına\_dikey\_uzaklık} = \text{gerçek\_dikey\_uzaklık} / h$$

3.3 YakarTarar'ın Tarama Algoritması

YakarTarar, tarama işlemi için bu algoritmayı izlemektedir:

```
döngü (x = 0'dan nesne genişliğine kadar):
   resim = kameradan_yakala()
   döngü (y = 0'dan nesne yüksekliğine kadar):
      kırmızı_x = satırdaki_kırmızılık(resim, y)
      z = z_değeri(kırmızı_x)
      z_değerleri(x,y) = z
   döngü sonu
   nesneyi_ilerlet()
döngü sonu
```

Bu algoritmanın çalışması sonucunda, taranan yüzeyin z değerleri, z_değerleri adlı iki boyutlu matrise yazılmış olur.

BÖLÜM DÖRT
YAKARTARAR'IN UYGULAMALAR

4.1 Sonuç Dosyaları Hakkında

YakarTarar ile yapılan çalışmalar, okunaklı metin dosyaları olarak kaydedilmektedir. Bu çalışmalar şunlardır:

- **Kalibrasyon:** YakarTarar Kalibrasyon dosyası (.ytcal) olarak kaydedilir.
- **Tarama:** YakarTarar Tarama dosyası (.yt) olarak kaydedilir.

Bir yüzey taraması yapmadan önce, kalibrasyon işlemini tamamlamak gereklidir. Kalibrasyon işlemi tamamlandığı zaman, YakarTarar, Tablo 4.1'deki gibi bir dosya oluşturur.

Tablo 4.1 Bir kalibrasyon dosyası örneği

Dosya İçeriği	Açıklaması
`yakartarar (` `  calibration (`	Kalibrasyon dosyasının başlangıcı

`    X (`
`      131`
`      163`
`    )`
	Kalibrasyonda kullanılan X-Z tablosunun X değerleri

`    Z (`
`      5.000000`
`      0.000000`
`    )`
	Kalibrasyonda kullanılan X-Z tablosunun Z değerleri

`    units_per_row (`
`      0.096154`
`    )`
	Satırların birer piksel ile ayrıldığı varsayıldığında, iki satırın, birbirinden kaç gerçek dünya birimi uzak olduğunu gösteren değer

`  )` `)`	Kalibrasyon dosyasının sonu

Kalibrasyon işlemi bittikten sonra, kalibrasyon verileri kullanılabilir ve yüzey taraması yapılabilir. Yüzey taraması işlemi sonucunda, Tablo 4.2'deki gibi bir dosya oluşturulur.

Tablo 4.2 Bir tarama dosyası örneği

Dosya İçeriği	**Açıklaması**
`yakartarar (` `  scan (`	YakarTarar tarama dosyasının başlangıcı

`stepwidth (`
`  2.000000`
`)`
	Eğer nesne:
	• Yatay olarak ilerletilerek taranıyorsa, Her sütunun birbirinden kaç gerçek dünya birimi uzak olduğunu belirten değerdir.
	• Olduğu yerde döndürülerek taranıyorsa, bu değer, 0 olur.

`rotation (`
`  0.000000`
`)`
	Eğer nesne:
	• Olduğu yerde döndürülerek taranıyorsa, bu değer, nesnenin her tarama adımında kaç derece döndüğünü belirten değerdir.
	• Yatay olarak ilerletilerek taranıyorsa, bu değer 0 olur.

`units_per_row (`
`  0.096154`
`)`
	Kalibrasyon işlemi sonucunda elde edilen “units_per_row” değeri, burada da tutulur.

Tablo 4.2 Bir tarama dosyası örneği (Devamı)

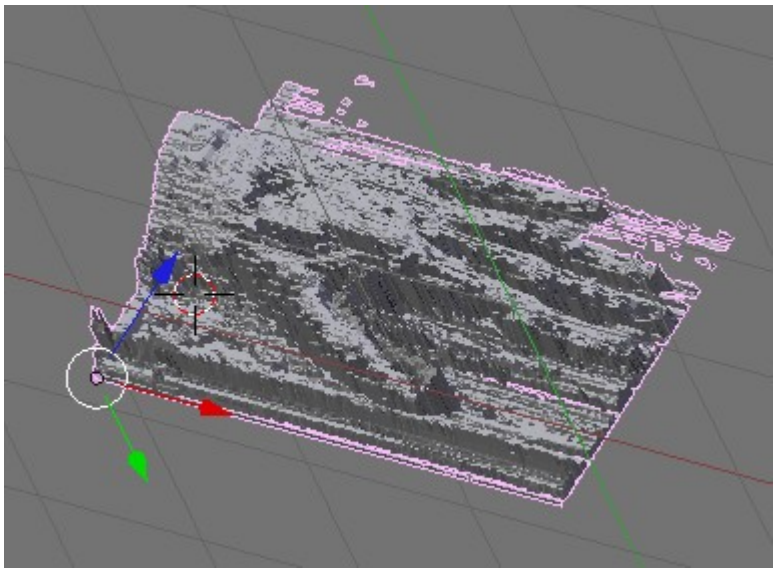
Dosya İçeriği	Açıklaması
column (… *(1. sütunun z değerleri)* …) column (… *(2. sütunun z değerleri)* …) … …	Tarama kaç sütundan oluşuyorsa, o kadar sayıda "column" bölümü açılır. Bu "column" bölümleri, sütunların yukarıdan aşağıya olan duvardan uzaklıklarını ("z"), gerçek dünya biriminde sayısal olarak tutar.
))	YakarTarar tarama dosyasının sonu

4.2 Sonuçları Başka Yazılımlara Aktarmak

YakarTarar'da yapılan taramalar aşağıdaki yazılımlara aktarılabilir:

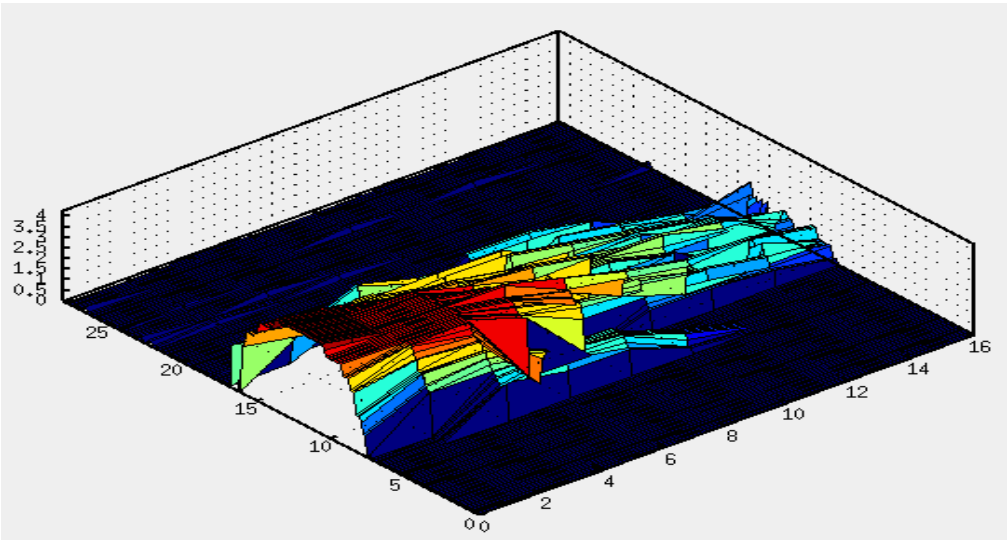
- Blender (ücretsiz açık kaynaklı üç boyutlu tasarım programı)
- MATLAB / Octave
- SolidWorks
- AutoCAD

YakarTarar, tarama sonuçlarını basit olarak çizebilmektedir. Ancak, sonuçları daha ayrıntılı inceleyebilmek için, bir üç boyutlu tasarım yazılımı kullanmak gerekebilir. Bunu sağlamak için, YakarTarar, tarama sonuçlarını Wavefront nesne dosyası (".obj") ve Virtual Reality Modeling Language (VRML) 1.0 dosyası (".wrl") olarak kaydedebilmektedir. Bu dosya türleri, Blender gibi üç boyutlu tasarım programlarınca açılabilmektedir (Şekil 4.1).



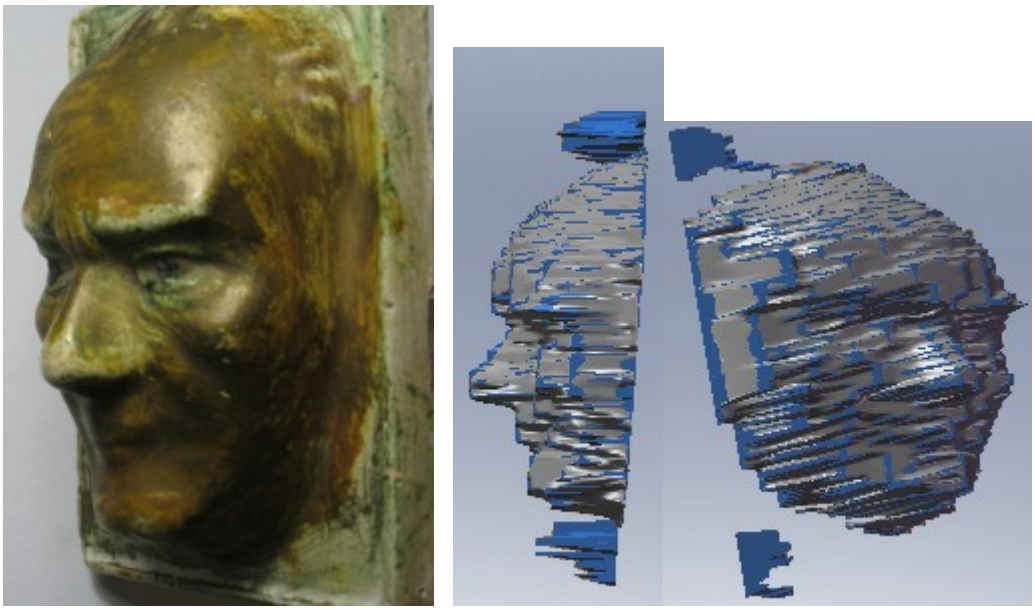
Şekil 4.1 Bir tarama sonucunun Blender yazılımındaki görüntüsü

Tarama sonuçlarının MATLAB'da, ya da MATLAB ile uyumlu olan Octave'da incelenebilmesi için, YakarTarar, tarama dosyalarını ".m" dosyası olarak kaydedebilmektedir. Bunu yapmak için, YakarTarar dosyasındaki veriler, kartezyen uzayına uygun iki boyutlu bir z matrisi olarak kaydedilmektedir. Bu matris, MATLAB veya Octave tarafından okunabilmekte ve çizilebilmektedir (Şekil 4.2).



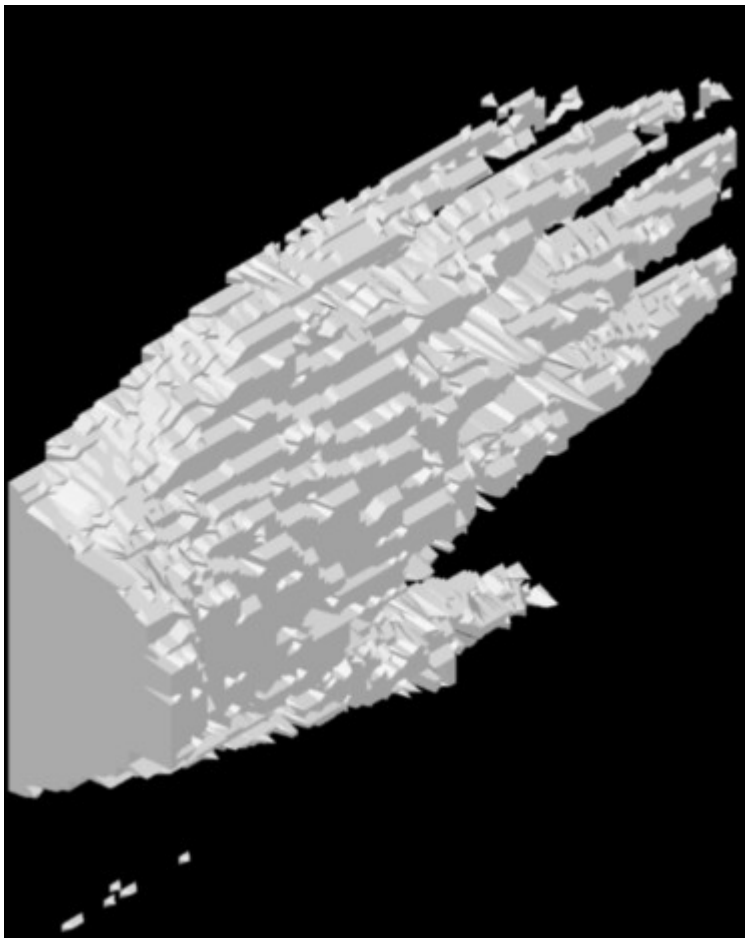
Şekil 4.2 Bir el taramasının Octave'daki görüntüsü

YakarTarar'ın sonuç dosyalarını “.obj” ve “.wrl” olarak kaydetmesi sayesinde, tarama sonuçları SolidWorks'te Mesh dosyası olarak açılabilmektedir (Şekil 4.3). Bu Mesh dosyaları, SolidWorks 2007'nin ScanTo3D eklentisi sayesinde katı parçaya çevrilebilmektedir.



Şekil 4.3 Bir Atatürk heykeli (solda) ve tarama sonucunun SolidWorks'teki görünüşleri (sağda)

YakarTarar'da yapılan çalışmalar AutoCAD'de de açılabilmektedir. Bunun için Blender kullanılmalıdır. Blender, YakarTarar'ın yarattığı “.obj” veya “.wrl” dosyalarını, AutoCAD'ın okuyabileceği “Drawing Exchange Format” (“.dxf”) dosyasına çevirebilmektedir (Şekil 4.4).



Şekil 4.4 DXF dosya türüne çevrilmiş bir el taraması sonucunun üç boyutlu görüntüsü

4.3 YakarTarar'ın C++ Kütüphanesine Erişmek

YakarTarar'ın temel işlevlerinden faydalanmak için kütüphaneye doğrudan erişim yapmaya gerek yoktur. YakarTarar, yüzey tarayıcı bir kullanıcı arayüzü bulundurmaktadır. Ancak, YakarTarar'ın bütün işlevlerine erişmek ve YakarTarar'ı geliştirilmekte olan başka bir yazılımın içinden kullanabilmek için, YakarTarar'ın C++ dilinde geliştirilmiş olan yazılım kütüphanesi kullanılmalıdır.

YakarTarar'ın C++ kütüphanesi, kendi içinde Intel'in OpenCV (Open Computer Vision Library) adlı kütüphanesini kullanmaktadır. C++ derleyicisinin OpenCV kütüphanesine erişmesi için gerekli ayarlar yapıldıktan sonra, YakarTarar'ın C++ başlık dosyalarının içerilmesi yetmektedir. Tablo 4.3'te, kütüphaneyi kullanarak yazılan bir C++ programının kaynak kodu verilmiştir.

Tablo 4.3 YakarTarar'ın C++ kütüphanesini kullanarak yüzey taraması yapan program

C++ Kodu	**Açıklaması**
`#include <cv.h>` `#include <highgui.h>`	OpenCV tanımlarının içerilmesi
`#include "yakartarar.hpp"`	YakarTarar tanımlarının içerilmesi
`using namespace std;`	C++ standart işlevlerine erişim
`using namespace yakartarar;`	YakarTarar işlevlerine erişim
`int main()` `{`	Programın başlangıcı
`CvCapture *kamera;` `IplImage *goruntu;`	Kamera ve görüntü değişkenlerinin tanımları
`kamera = cvCreateCameraCapture(0);`	Kameraya erişim

`ytCalibration ayarlar;`
 	
`ytCalibrationWindow ayar_menusu (kamera, "Kalibrasyon Menusu");`
 	
`ayarlar = ayar_menusu.calibrate();`
	• "ayarlar" adlı kalibrasyon ve "ayar_menusu" adlı kalibrasyon menüsü değişkenlerinin tanımları
	• "ayar_menusu.calibrate()" komutu, kalibrasyon için gerekli soruları kullanıcıya sorar

Tablo 4.3 YakarTarar'ın C++ kütüphanesini kullanarak yüzey taraması yapan program (Devamı)

C++ Kodu	Açıklaması

`double aralik = yt_input_double( "Yüzey her adımda kaç birim ilerliyor?");`
 	
`ytFullScan yuzey(`
`ayarlar,`
`ytFullScan::param_stepwidth,`
`aralik);`
	• Kullanıcıya, nesnenin her adımda kaç birim ilerletildiği soruluyor ve cevap "aralik" değişkenine aktarılıyor
	• Kalibrasyon ayarlarına ve nesnenin ilerleyişine göre yeni bir yüzey değişkeni tanımlanıyor

`int basilan_tus;`
 	
`while(true)`
`{`
`goruntu = yt_capture(kamera, basilan_tus);`
`if(basilan_tus == 27) break;`
`ytColumnScan sutun(ayarlar, goruntu);`
`yuzey.add(sutun);`
`cvReleaseImage(&goruntu);`
`}`
	Burada bir döngü bulunmaktadır. Bu döngünün içinde ise şu işlemler yapılmaktadır:
	• Kameradan görüntü al (Bir tuşa basılıncaya kadar bekler) ve "goruntu" adlı değişkene aktar
	• Basılan tuş ESC ise döngüden çık
	• Kalibrasyon ayarlarına göre, görüntüden z değerlerini bul ve "sutun" adlı değişkene aktar
	• Sütunu yüzeye ekle
	• "goruntu" değişkenini temizle

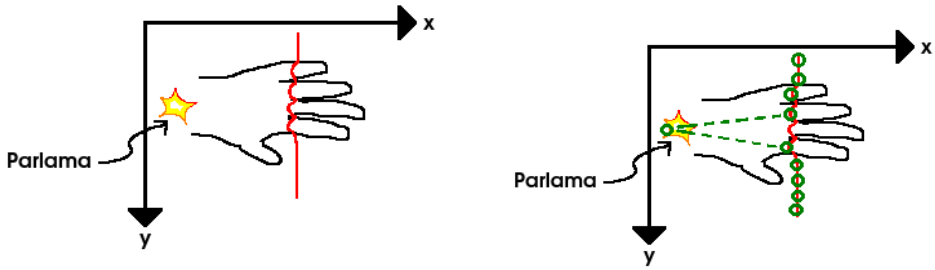
`yuzey.save("sonuclar.yt");`	Sonuçların "sonuclar.yt" adlı dosyaya kaydedilmesi

`cvReleaseImage(&goruntu);`
`cvReleaseCapture(&kamera);`
`return 0;`
`}`
	"goruntu" ve "kamera" adlı değişkenlerin temizlenmesi ve programın sonlandırılması

4.4 YakarTarar'la Tarama Sonuçlarındaki Hataları Düzeltmek

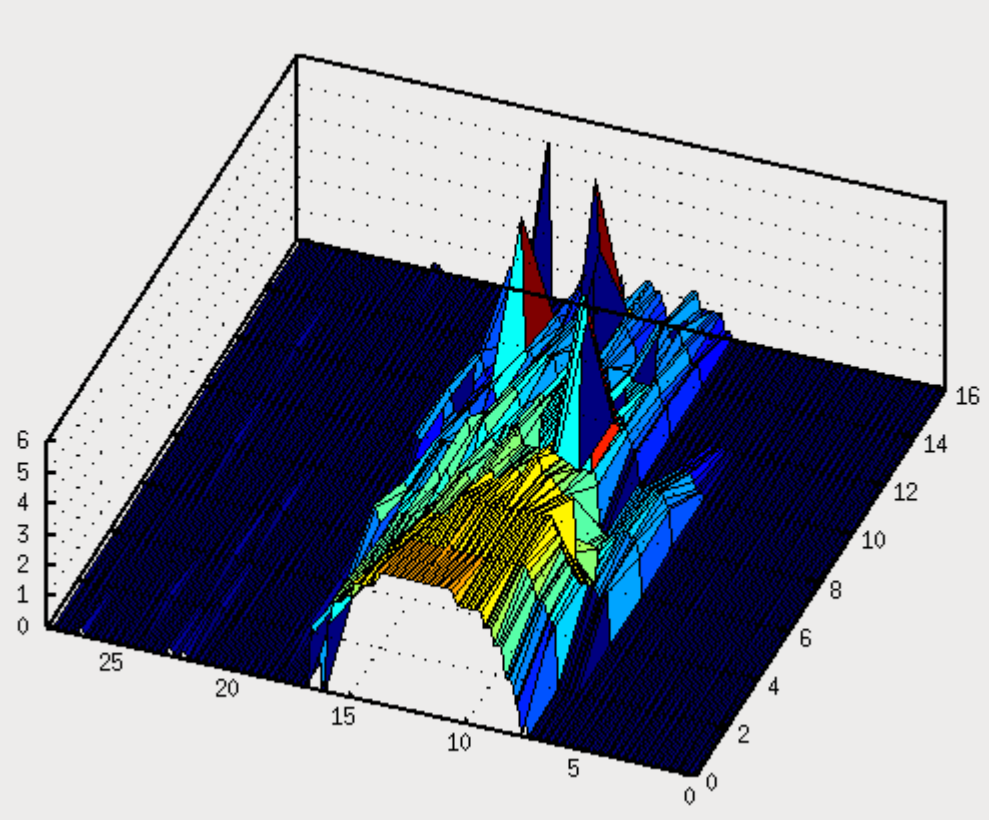
Bir nesnenin modelini çıkarırken, yüzey taraması sırasında oluşan bazı hatalar, yüzeyin sayısal ortamlarda ciddi boyutlarda farklılaşmasına neden olabilir. Bunun için, YakarTarar, hatalı yüzeyleri düzeltme işlevi barındırmaktadır.

Bir yüzey taraması yapılırken, dışarıdan gelen istenmeyen parlamalar, YakarTarar'ın görüntü işleme modülünün lazerin konumunu yanlış saptamasına sebep olabilir.



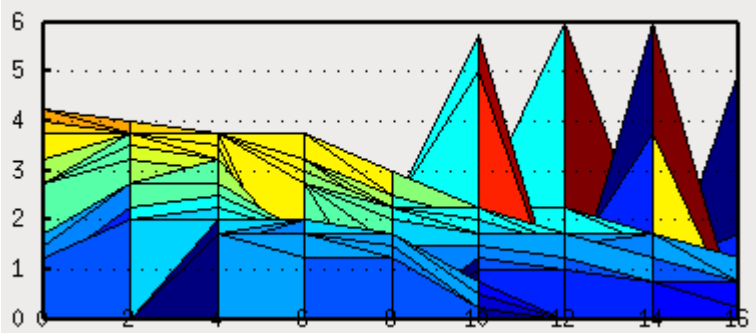
Şekil 4.5 İstenmeyen bir parlama, lazerin konumu yerine parlamanın konumunun hesaba katılmasına neden olur.

Şekil 4.5'teki örnekte, kameranın, lazer kaynağının sağında olduğu düşünülmüştür. Buna göre, aydınlık iz sola doğru gittikçe, yüzeyin duvardan uzaklığı (z değeri) artmaktadır. Hataya sebep olan parlamanın, lazer ışınının oluşturduğu izin solunda olması nedeniyle, tarama sonucunda çok yüksek bir tepe oluşacaktır. Şekil 4.6'da, hatalı parlamalar sonucu oluşan tepeler gösterilmektedir.



Şekil 4.6 Bir el taraması sonucu. Bu tarama sırasında oluşan parlamalar, lazerin konumunun yanlış saptanmasına neden olmuştur ve sayısal modele yüksek tepeler olarak yansımıştır.

Şekil 4.5 ve Şekil 4.6'da gösterilen örneğin kalibrasyonu cm cinsinden yapılmıştır. Buna göre, sonuç çizimine yandan bakıldığında görülecektir ki, en yüksek olması gereken kısım olan bilek kısmı 4.5 cm'in altındayken, daha alçak olması gereken parmak kısımlarında, 6 cm'e kadar yükselen tepeler bulunmaktadır (Şekil 4.7).

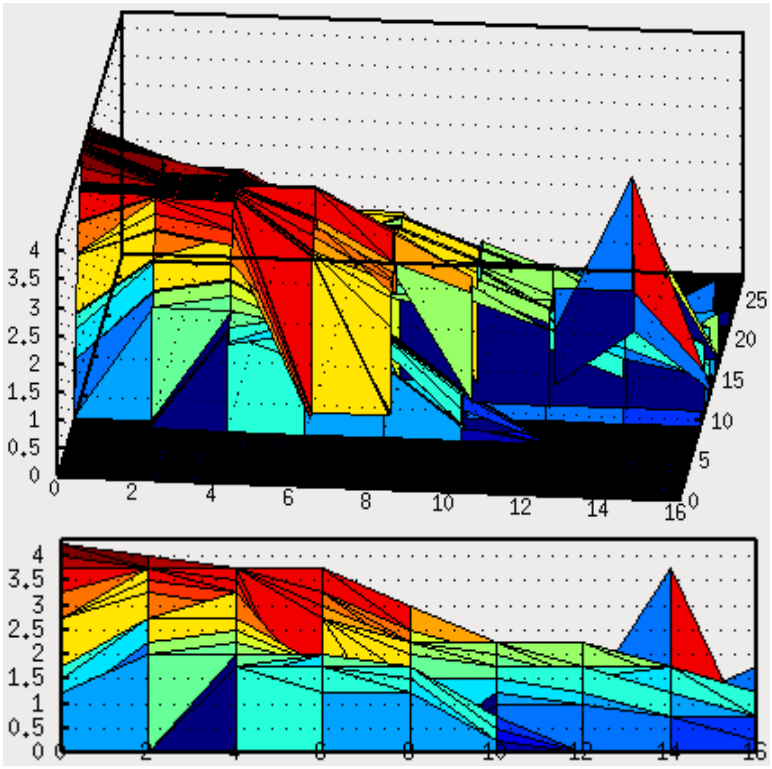


Şekil 4.7 Hatalı el taramasının yandan görünüşü

Bu tepeleri azaltmak için, bir budama işlemi yapmak gerekmektedir. YakarTarar'a, duvardan uzaklığı (Şekil 4.6 ve Şekil 4.7'deki yükseklik boyutu, z değeri) belli bir değeri geçen tepeleri silmesi komutu verilebilir. YakarTarar, bu komutu alınca, yüzeydeki hatalı tepeleri bulur ve o tepelerin yerine, etraflarındaki z değerlerinin ortalaması yüksekliğindeki yüzeylerle dolgu yapar. Bu işlem, aşağıdaki komutla yapılabilmektedir:

```
yakartarar chop el_taramasi.yt chopped1.yt 4.5
```

Bu komut sonucunda, “el_taramasi.yt” adlı tarama sonucu dosyasındaki 4.5 birimden (buradaki örneğe göre, cm) yüksek olan tepeler budanır ve sonuç “chopped1.yt” adıyla kaydedilir. Bu oluşan dosya, Octave'da incelendiğinde, Şekil 4.8'deki görüntü elde edilir.

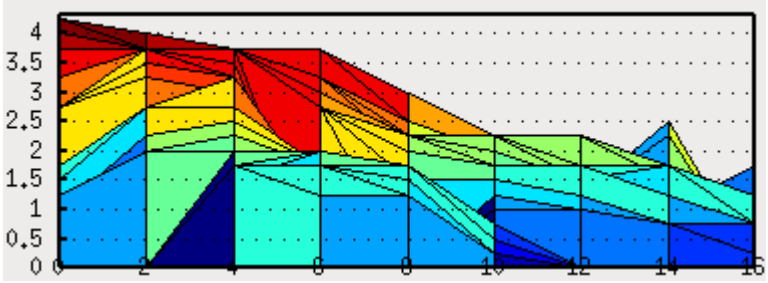


Şekil 4.8 Önceki şekilde (Şekil 4.7) bulunan örneğin, yüzey genelinde budanmış hali

Şekil 4.8'de görüldüğü gibi, yüzey genelindeki budama işlemi başarılı geçmiş ve yüksekliği 4.5 cm üstündeki bütün tepeler silinmiştir. Ancak, parmak uçları kısmında, yüksekliği bilek kısmından daha az olduğu için budama işleminden kurtulmuş olan hatalı bir tepe hala bulunmaktadır. Bu durumu düzeltmek için, x=12 ve x=16 arasında yerel budama işlemi yapmak gerekmektedir. Bunun için aşağıdaki komut çalıştırılabilir:

```
yakartarar chop chopped1.yt chopped2.yt 3.5 cartesian 10 0 16 30
```

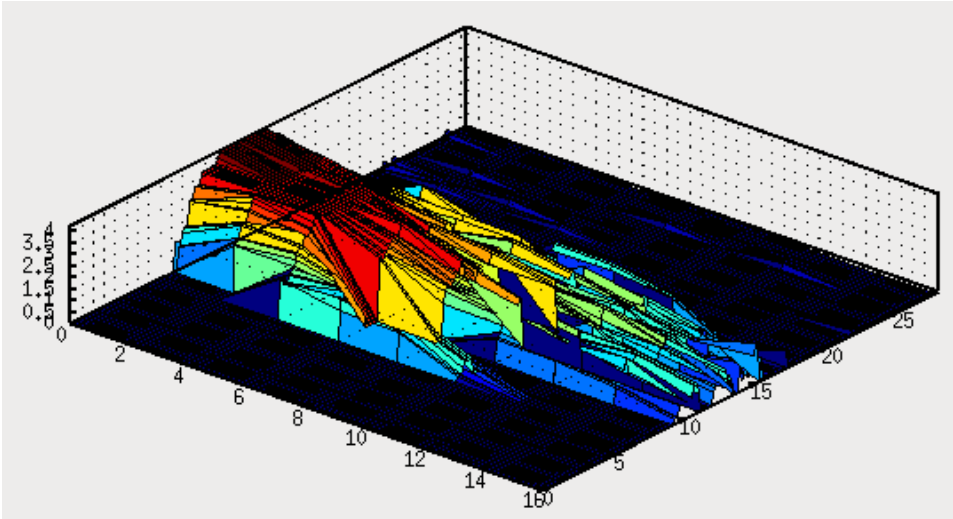
Bu komuta göre, “chopped1.yt” dosyası açılacak, z değeri 3.5'ten büyük olan tepeler, (x=10, y=0) noktasından (x=16, y=30) noktasına kadar olan dikdörtgensel alanda bulunup budanacaktır. Bu komutun çalışması sonucu oluşturulan “chopped2.yt” dosyası, Şekil 4.9'daki gibidir.



Şekil 4.9 İki budama işleminden sonra, hataları azaltılmış olan bir el modelinin yandan görünüşü

Şekil 4.9'da, x=14'te bulunan küçük bir hata tepesi kalmıştır. Köşeleri (2, 14) ve (16, 30) konumlarında olan dikdörtgensel alanda, z değeri 2'den yüksek olan tepeleri silmeye yarayan aşağıdaki komut satırı sayesinde, o küçük hata da yok edilecek ve sonuç Şekil 4.10'daki gibi olacaktır.

```
yakartarar chop chopped2.yt chopped3.yt 2 cartesian 14 0 16 30
```



Şekil 4.10 Hatalı tepeleri yok edilmiş olan el taraması

4.5 YakarTarar'ı Gerçek Zamanlı Algılayıcı Olarak Kullanmak

Gerçek zamanlı üç boyutlu algı, aşağıdaki durumlarda kullanılmaktadır:

- **Seri üretim:** Üretim bandından geçen nesneler, gerçek zamanlı olarak lazerle taranır (Petrovic, 2007).
- **Otonom cihazlar:** Otonom cihazlar, çevrelerini lazerle tarayarak engelleri tanır (Kanade, Amidi, Ke, 2004).

Buna benzer sistemler tasarlanırken, kontrol birimi olan bilgisayardaki yazılımın üç boyutlu algı modülü, YakarTarar'ın işlevleri kullanılarak kısa zamanda hazırlanabilir. Tablo 4.4'teki kod örneği, gerçek zamanlı tarama yapmaktadır ve belirtilen yüksekliğin üstünde bir cisim fark ettiği zaman uyarı vermektedir (Şekil 4.11).

Tablo 4.4 YakarTarar yardımıyla gerçek zamanlı yükseklik denetimi yapan C++ programı

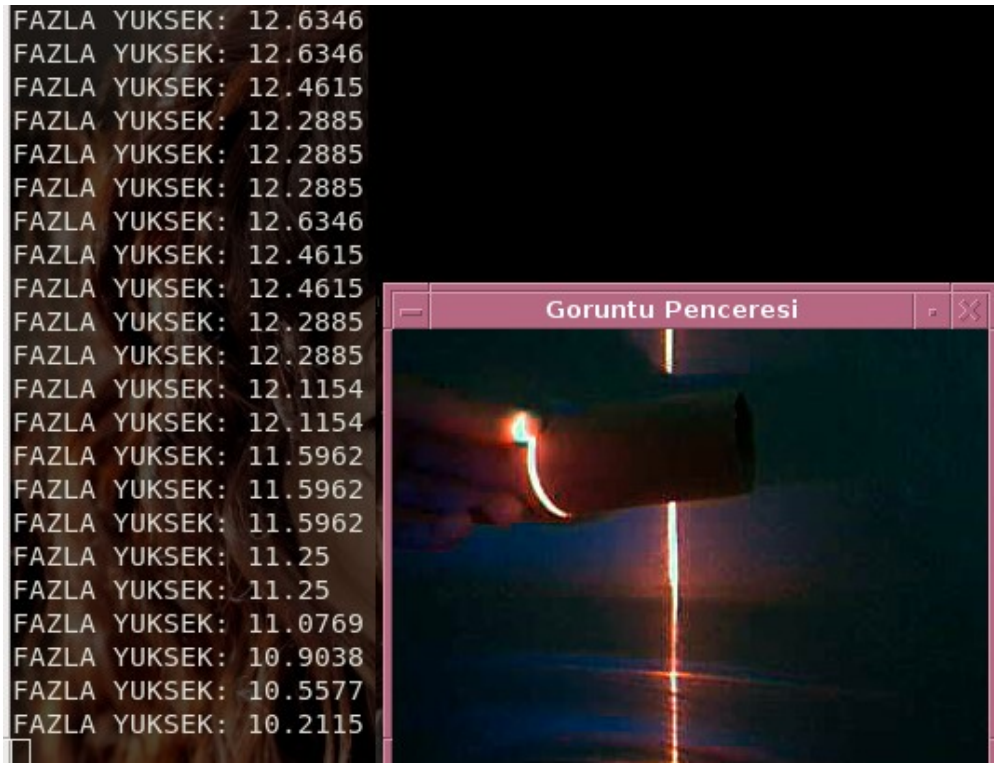
C++ Kodu	Kodun açıklaması

`#include <iostream>`
 	
`#include <cv.h>`
`#include <highgui.h>`
 	
`#include "yakartarar.hpp"`
 	
`using namespace std;`
`using namespace yakartarar;`
	Gerekli olan C++ kütüphanelerinin yüklenmesi:
	• C++'ın standart kütüphaneleri
	• Intel'in OpenCV kütüphaneleri
	• YakarTarar kütüphanesi

Tablo 4.4 YakarTarar yardımıyla gerçek zamanlı yükseklik denetimi yapan C++ programı (Devamı)

`int main()`
`{`
`  ytCalibration kalibrasyon;`
`  CvCapture *kamera;`
`  IplImage *goruntu;`
 	
`  kamera = cvCreateCameraCapture(0);`
 	
`  ytCalibrationWindow kalibrasyon_menusu(kamera, "Kalibrasyon Menusu");`
`  kalibrasyon = kalibrasyon_menusu.calibrate();`
 	
`  cvNamedWindow("Goruntu Penceresi");`
 	
`  int basilan_tus = 0;`
 	
`  const double uygun_yukseklik = 10.0;`
	Programın ana bölümünün başlangıcı.
	
	Bu bölümde gerekli tanımlamalar yapılmaktadır:
	• YakarTarar kalibrasyon yapısı
	• OpenCV kamera ve resim yapıları
	• YakarTarar kalibrasyon menüsü
	• OpenCV görüntü penceresi
	• Kabul edilebilir yükseklik sınırı

`  while(true) {`
`  goruntu = yt_capture( kamera, basilan_tus, 20);`
 	
`  cvShowImage( "Goruntu Penceresi", goruntu);`
 	
`  if(basilan_tus == 27) break;`
 	
`  ytColumnScan tarama_sutunu(kalibrasyon, goruntu);`
 	
`  for(int satir = 0; satir < tarama_sutunu.Z.size(); satir++)`
`  {`
`  double z = tarama_sutunu.Z[satir];`
`  if(z > uygun_yukseklik)`
`  {`
`  cout << "FAZLA YUKSEK: " << z << endl;`
`  }`
`  }`
 	
`  cvReleaseImage(&goruntu);`
`  }`
`  cvReleaseCapture(&kamera);`
 	
`  return 0;`
`}`
	ESC tuşuna basılana kadar işlem yapan bir döngü.
	
	Her adımda:
	• Kameradan görüntü alınır
	• Görüntü ekrana yansıtılır
	• Görüntüden bir YakarTarar tarama sütunu yapısı oluşturulur
	• Sütun içindeki her satırın z değerine bakılır, kabul edilemeyecek kadar yüksek olan bir z değeri bulunduğunda uyarı yapılır



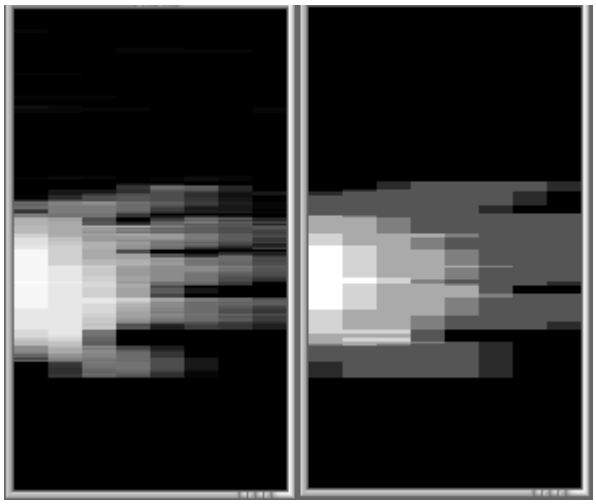
Şekil 4.11 YakarTarar'ın işlevlerinin gerçek zamanlı yükseklik denetimi için kullanımı

4.6 YakarTarar Sonuçlarını Gauss Yöntemiyle Bulandırmak

Gauss bulandırması (“Gaussian Blurring” ya da “Gaussian Smoothing”), aslında iki boyutlu resimler üzerinde uygulanan bir yöntemdir. Bu yönteme göre, bir resmin içindeki bütün noktalar, etraflarındaki noktalardan etkilenerek renk değişimine uğrar. Bir nokta, işlem gören noktaya ne kadar yakınsa, işlem gören noktayı o kadar çok etkiler. Bu etkinin katsayısı ise, noktanın uzaklığının Gauss çanı (“Gaussian Curve”) fonksiyonuna verilmesi sonucunda elde edilen sayıdır (Forsyth, Ponce, 2002, s. 185).

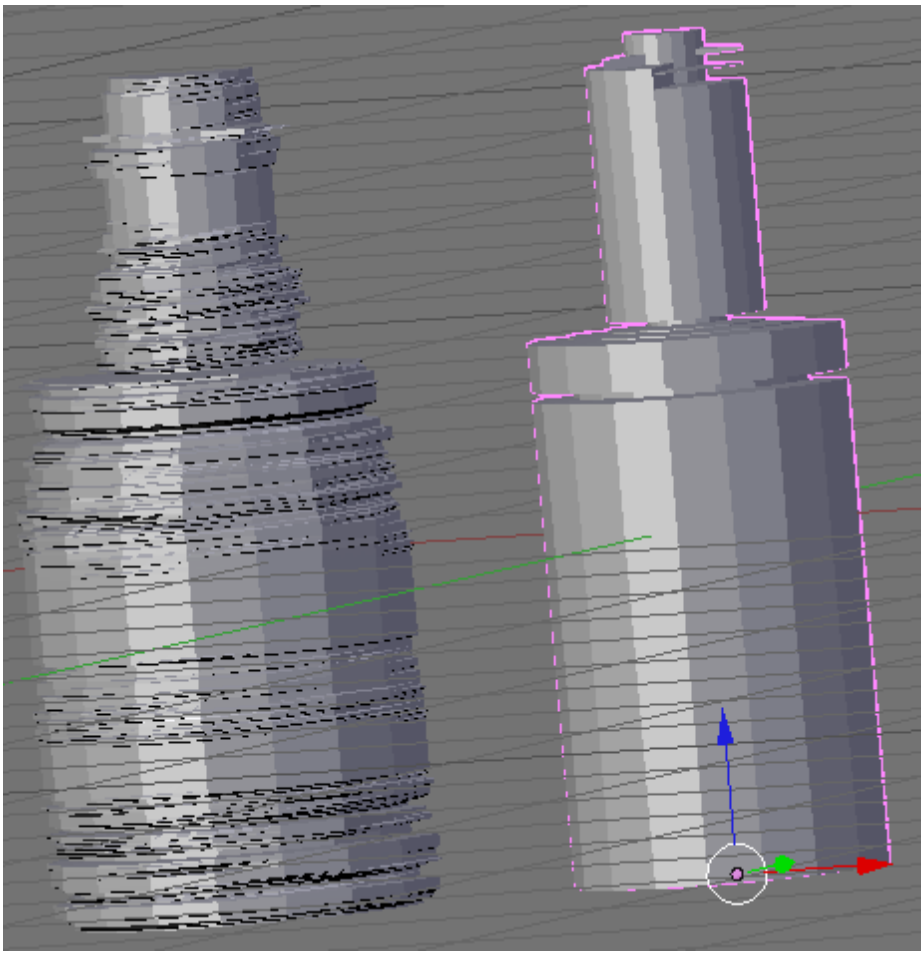
YakarTarar'ın Gauss yöntemiyle bulandırma modülü de, üç boyutlu yüzeyin iki boyutlu bir resim olduğunu, üçüncü boyutun ise renk olduğunu varsayarak hareket eder. Buna göre, noktaların konumları x ve y, renkleri ise z olur. Yani, birbirine yakın olan noktalar birbirlerinin z değerlerini etkiler.

YakarTarar sonuçları Gauss yöntemiyle bulandırıldığı zaman, yüzeyin ayrıntıları kaybolur. Bu, tarama sonucunun daha da bozulmasına neden olabilir (Şekil 4.12).



Şekil 4.12 Bir tarama sonucu (solda) ve Gauss yöntemiyle bulandırıldıktan sonraki ayrıntılarını kaybetmiş hali (sağda).

Ancak, yüzeyi pürüzsüz olması gerekirken istem dışı parlamalar sonucu pürüzlü olarak modeli oluşturulan bir yüzey, bulandırma yöntemiyle pürüzlerinden kurtulabilir (Şekil 4.13).



Şekil 4.13 Silindirik bir yüzeyin pürüzlü hali (solda) ve Gauss yöntemiyle bulandırıldıktan sonraki pürüzsüz hali (sağda)

4.7 Bir Yüzey Taraması Sonucundaki Ana Nesneyi Kirlerden Ayırmak

YakarTarar ile yüzey taraması yapıldığı zaman, sadece odaklanılmış olan nesnenin yüzeyi değil, kameranın görebildiği bütün yüzeyler tarama sonucuna dahil olur. Bu durum, elde edilen sonuçların şişmesine neden olmaktadır. Buna ek olarak, eğer arkaplanda çıkıntılar varsa, odaklanılan nesnenin yüzeyinin dışında, istenmeyen üç boyutlu yüzeyler de sonuca eklenir. Bu istenmeyen yüzeyleri "kir" olarak adlandırabiliriz (Şekil 4.14).

Şekil 4.14 Bir el taraması sonucu. Bu sonuçta, arkaplandaki bir çıkıntı da, istenmeyen bir şekilde sonuca yansımıştır.

YakarTarar, odaklanılan yüzeyi, kendisiyle bağlantısı olmayan kirlerden ve gereksiz boşluklardan arındırma seçeneği sunar. Bu işlemde, kullanıcı, ana yüzeyin bir noktasının koordinatlarını girer. YakarTarar, o koordinattan sağa, sola, yukarı ve aşağı yönlerde ilerleyerek o yüzeyin sınırlarını belirler. Bu sınırların dışında kalan herşey silinir. Bu işlemin sonucu Şekil 4.15'teki gibidir.

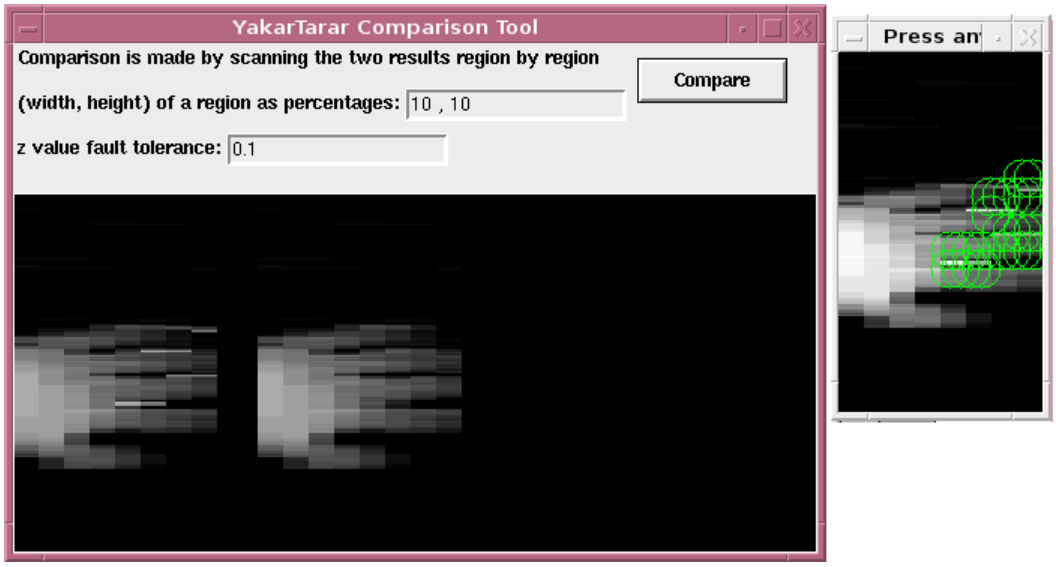
Şekil 4.15 Önceki şekilde (Şekil 4.14) gösterilen el taramasının, gereksiz kirlerden ve boşluklardan arınmış hali

Boşlukların ve kirlerin silinmesi, yüzey karşılaştırma ya da sınıflandırma gibi işlemler yapan yüzey inceleme algoritmalarının şaşırıp hata yapma olasılığını azaltabilir.

4.8 İki Tarama Sonucunu Karşılaştırmak

YakarTarar, iki adet yüzey taramasını karşılaştırmada kullanılabilecek işlevlere sahiptir. Bu işlevleri kullanan basit bir karşılaştırıcı yazılmış ve kullanıcı arayüzüne eklenmiştir.

Karşılaştırıcı, tarama sonuçlarını bölgelere ayırır. Bölgelerin boyutları, tarama sonuçlarının genişlik ve yüksekliğine bağlantılı olarak yüzde ile belirtilir. Her bölgenin duvardan uzaklık değerlerinin ortalamaları (ortalama z değerleri) hesaplanır. Belirtilen sınır değerinin üzerinde olan farklılıklar işaretlenerek kullanıcıya gösterilir. Örnek bir karşılaştırma sonucu, Şekil 4.16'da gösterilmiştir.



Şekil 4.16 Hataları olan bir el taraması ile hataları giderilmiş olanının karşılaştırılması (solda) ve karşılaştırma işleminin sonucu (sağda)

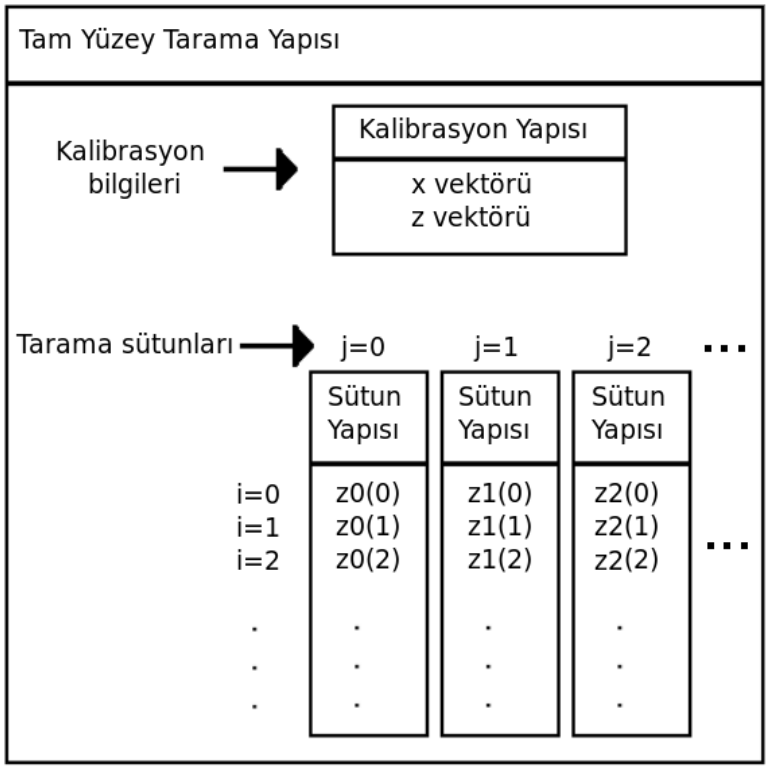
BÖLÜM BEŞ
YAKARTARAR YAZILIMININ İÇ YAPISI

5.1 Yapıya Genel Bakış

YakarTarar yazılımının iç yapısında, üç ana öğe bulunmaktadır:

- **Kalibrasyon Yapısı (ytCalibration):** Bir kalibrasyon işleminin sonucunda, x ve z arasındaki bağıntının hesaplanması için, x-z tablosunu tutan yapı.
- **Sütun Yapısı (ytColumnScan):** Dikey lazer ışınının bıraktığı iz üzerindeki z değerlerini tutan yapı.
- **Tam Yüzey Tarama Yapısı (ytFullScan):** Bir kalibrasyon nesnesi yardımıyla kalibrasyon bilgilerini tutan, taranan yüzeyin her sütunu için birer sütun nesnesi içeren ve bir yüzeyin bütün sayısal modelini ifade eden yapı.

Bu öğelerden kalibrasyon ve sütun nesneleri, taranan yüzeylerin sayısal modellerinin ifadesinde kullanılmak üzere, tam yüzey tarama nesneleri tarafından içerilmektedir. Bu üç öğenin birbiri ile olan ilişkisi, Şekil 5.1'deki gibi özetlenebilir.



Şekil 5.1 YakarTarar'ın iç yapısı. Bu yapı, kendi içinde, bir yüzey tarama yapısı, bir adet kalibrasyon yapısı, sütun sayısı kadar da sütun yapısı barındırmaktadır.

5.2 Kalibrasyon Yapısı

Bir kalibrasyon yapısı, görüntülerdeki z değerlerinin doğru çıkartılması için yapılan ayarları barındıran nesnedir.

5.2.1 X ve Z Vektörleri

Bir kalibrasyon yapısı, kendi içinde X ve Z adlı vektörler barındırır. Bu vektörler, beraber x-z tablosunu oluştururlar.

Örneğin, vektörler X=<100,80> ve Z=<0,10> ise;

- Görüntü içinde x=20 konumunda lazerin izi belirlendiyse, o noktanın duvardan uzaklığı z=0 olur.
- Görüntü içinde x=80 konumunda lazerin izi belirlendiyse, o noktanın duvardan uzaklığı z=10 olur.
- Görüntü içinde x=90 konumunda lazerin izi belirlendiyse, o noktanın duvardan uzaklığı, eğim denkleminden bulunur:

```
z_eğimi = (z2 – z1) / (x2 – x1) = (10 – 0) / (80 – 100) = -0.5

          z_değeri(x) = (z_eğimi * (x – x1)) + z1
=> z_değeri(90) = (-0.5 * (90 - 100)) + 0 = 5
```

Eğer X ve Z adlı vektörlerin öğe sayısı ikiden fazla ise, z değeri bulunmak istenen noktanın x değerinin, vektör içinde hangi x değerlerinin arasında olduğu bulunur. Bu aralığın vektör içinde a ve b sayılı öğeler olduğunu varsayarsak, x1 = X(a), x2 = X(b), z1 = Z(a), z2 = Z(b) olur. x1, x2, z1, z2 değerleri bulunduktan sonra, z_değeri(x) fonksiyonu kullanarak z değeri bulunur.

5.2.2 "satır_başına_düşen_birim" Değişkeni

Bu değişken, kalibrasyon yapısı içinde tutulan sayısal bir değerdir. Kalibrasyon yapısı, X ve Z adlı vektörler sayesinde, duvardan uzaklık boyutunu gerçek dünya birimleriyle bulabilmektedir. Yüzeyin yatay ölçüleri ise, tarayıcı modülü tarafından sütun sayısına göre bulunacaktır. Ancak, taranan yüzeyin dikey ölçüleri bu bilgiler içinde yer almamaktadır. Dikey ölçülerin gerçek dünya birimleriyle bilinmemesi ise, taranan yüzeyin sayısal modelinin eksik çıkartılması anlamına gelmektedir. Bunun için, kalibrasyon yapısı, kendi içinde *satır_başına_düşen_birim* (yazılım içindeki adıyla

“units_per_row”) adlı bir değişken tutar. Bu değişken, kamerayla alınan görüntü içinde bir piksel uzaklıkla ayrılmış olan iki satırın, gerçek dünyada kaç birim uzaklığa denk geldiğini tutar.

5.3 Sütun Yapısı

Bu nesne, tarama sırasında elde edilmiş bir sütunun z değerlerini tutar. Çalışmak için, parametre olarak kalibrasyon yapısı ve bir kamera görüntüsü aldıktan sonra, kalibrasyon bilgilerine göre, görüntüdeki her satırın z değerlerini hesaplayıp Z adlı bir vektör yaratır. Bu vektörün uzunluğu, kameradan alınan görüntünün yükseklik ölçüsü kadardır. Örneğin, 640x480 boyutlarında alınan bir görüntüden elde edilen bir sütunun Z vektörünün boyutu 480'dir.

5.4 Tam Yüzey Tarama Yapısı

Bu yapı, gerçek dünyada bulunan bir yüzeyin sayısal modelinin tamamını barındırır. Yapı, kendi içinde şunları barındırır:

- Bir adet Kalibrasyon Yapısı nesnesi
- Sütun sayısı kadar Sütun Yapısı nesnesi

Bu yapının önemli öğeleri arasında kalibrasyon bilgileri, ilerletme veya döndürme bilgileri ve nesne üzerinde oynamalar yapılmasını sağlayan işlevler bulunmaktadır.

5.4.1 Kalibrasyon Bilgileri

Bir Tam Yüzey Tarama yapısı, bu bilgileri, kendi içinde bir kalibrasyon yapısında barındırmaktadır. Burada kalibrasyon yapısının en önemli barındırılma amacı, *satır_başına_düşen_birim* (yazılım içindeki adıyla "units_per_row") adlı değişkendir. Bu değişken, sayısal modelin dikey ölçülerinin gerçek dünya birimine göre belirlenmesini sağlar.

5.4.2 İlerletme ve Döndürme Bilgileri

Bir Tam Yüzey Tarama yapısı, bu bilgileri, sırasıyla *yüzeyin_ilerleyişi* (yazılım içindeki adıyla "stepwidth") ve *yüzeyin_döndürülme_açısı* (yazılım içindeki adıyla "rotation") adlı değişkenlerde tutmaktadır. *yüzeyin_ilerleyişi* değişkeni, her sütunun arasındaki uzaklığı gerçek dünya birimlerinde tutmaktadır. Eğer nesne ilerletilerek değil de, döndürülerek tarandıysa, *yüzeyin_döndürülme_açısı* adlı değişken, her sütunun kaç derece çevirme sonucunda elde edildiğini tutar. Bu değişkenler sayesinde, taranan yüzeyin yatay ölçüleri belirlenmektedir.

5.4.3 Sütun Dizisi

Tarama sırasında elde edilen her sütun, bir Tam Yüzey Tarama yapısı içindeki sütunlar dizisinde tutulmaktadır. Bu dizinin her öğesi, bir sütunun z değerlerini tutmakla yükümlüdür. Bu sayede, taranan yüzeyin üçüncü boyut (duvardan uzaklık) ölçüleri de belirlenmiş olur.

5.4.4 "kaçıncı_satırda(y)" İşlevi

Tam Yüzey Tarama yapısının "kaçıncı_satırda(y)" adlı işlevi, gerçek dünya ölçülerinde bir yükseklik değeri olan y'nin, yüzeyin sayısal modelinde kaçıncı satıra denk geldiğini bulur. Bu işlev, şu şekilde tanımlanabilir:

```
kaçıncı_satırda(y):
   n = tamsayı(y / satır_başına_düşen_birim)

   eğer n < 0:
      n = 0
   eğer n >= satır_sayısı ise:
      n = satır_sayısı - 1

   sonuç = n
```

5.4.5 "kaçıncı_sütunda(x)" İşlevi

Tam Yüzey Tarama yapısının "kaçıncı_sütunda(x)" adlı işlevi, parametre olarak x değişkenini alır. Bu değişken, eğer nesne ilerletilerek tarandıysa, yatay konumun gerçek dünya ölçülerine göre belirtilmiş halinin sayısal değerini tutar. Eğer nesne döndürülerek tarandıysa, x, açı derecesi olarak verilmelidir. İşlevin sonucu, belirtilen konumun sayısal model içinde kaçıncı sütunda olduğudur. Bu işlev, aşağıdaki gibi tanımlanabilir:

```
kaçıncı_sütunda(x):
   eğer yüzey_döndürülerek_tarandı ise:
      n = tamsayı(x / yüzeyin_döndürülme_açısı)
   eğer yüzey_ilerletilerek_tarandı ise:
      n = tamsayı(x / yüzeyin_ilerletilme_çokluğu)

   eğer n < 0:
      n = 0
   eğer n >= sütun_sayısı ise:
      n = sütun_sayısı - 1

   sonuç = n
```

5.4.6 "xyz_konumunu_bul(j, i)" İşlevi

Tam Yüzey Tarama yapısının "xyz_konumunu_bul(j, i)" adlı işlevi, sayısal model içindeki, kaçıncı satırda(i) ve kaçıncı sütunda(j) bulunduğu bilinen bir noktanın, gerçek dünya ölçüleriyle(x, y, z) hangi konumda olduğunu belirtir. Burada yüzeyin en solu x=0, yüzeyin en üstü de y=0 konumlarında kabul edilmektedir. Bu işlevin algoritmik ifadesi şu şekildedir:

```
xyz_konumunu_bul(j, i):
   son_satır = satır_sayısı – 1

   eğer yüzey_ilerletilerek_tarandı ise:
      x = j * yüzeyin_ilerleyişi
      y = (son_satır – i) * satır_başına_düşen_birim

      ilgili_sütun = sütunlar_dizisi(j)
      o_sütundaki_dikey_z_vektörü = ilgili_sütun.Z
      z = o_sütundaki_dikey_z_vektörü(i)
   eğer yüzey_döndürülerek_tarandı ise:
      açı = (j * yüzeyin_döndürülme_açısı) + 180

      ilgili_sütun = sütunlar_dizisi(j)
      o_sütundaki_dikey_z_vektörü = ilgili_sütun.Z
      r = o_sütundaki_dikey_z_vektörü(i)

      x = r * cos(açı)
      y = (son_satır – i) * satır_başına_düşen_birim
      z = r * sin(açı)

   sonuç = <x, y, z>
```

5.4.7 Yüzey Düzeltme İşlevi

Tam Yüzey Tarama yapısının "düzelt(düzeltme_türü, yükseklik_sınırı, alan)" olarak tanımlanmış olan yüzey düzeltme işlevi, yüzeyin belirtilen alanında, hatalı tepeleri budama ve hatalı çukurları doldurma olmak üzere iki çeşit düzeltme yapar. Hatalı tepeleri budama görevi verildiğinde, z değeri yükseklik sınırından fazla olan bölgeler, hatalı olarak kabul edilir. Hatalı çukurları doldurma görevinde ise, z değeri yükseklik sınırından az olan bölgeler hatalı olarak kabul edilir. Hatalı bölgelerin z değerleri,

çevrelerindeki hatasız olan noktaların z değerlerinin ortalamaları ile değiştirilir. Bu işlev, şu şekilde tanımlanabilir:

```
düzelt(düzeltme_türü, yükseklik_sınırı, alan):
   alan içindeki her konum için:
      yatay_konum = konum.j
      dikey_konum = konum.i

      z = yüzeyin_z_değerleri(yatay_konum, dikey_konum)

      eğer düzeltme_türü = tepeleri_buda ise:

         eğer z > yükseklik_sınırı ise:

            yeni_z = çevresinin_ortalaması(
               düzeltme_türü, yükseklik_sınırı,
               yatay_konum, dikey_konum)

            yüzeyin_z_değerleri(yatay_konum, dikey_konum) = yeni_z

      eğer düzeltme_türü = çukurları_doldur ise:

         eğer z < yükseklik_sınırı ise:

            yeni_z = çevresinin_ortalaması(
               düzeltme_türü, yükseklik_sınırı,
               yatay_konum, dikey_konum)

            yüzeyin_z_değerleri(yatay_konum, dikey_konum) = yeni_z
```

5.4.8 Modeli Dışa Aktarma İşlevi

Tam Yüzey Tarama yapısının “modeli_aktar(dosya_adı)” olarak tanımlanmış olan işlevi, hafızadaki yüzeyi Wavefront Obj dosyası (“.obj”) ya da VRML 1.0 (“.wrl”) dosyası olarak kaydetmeye yarar. Bunun için, yüzeyi dikdörtgen şekli oluşturan küçük parçalara ayırır. Daha sonra bu dikdörtgenlerin, gerçek dünyadaki konumlarını bulur ve bu konumlardan da ikişer üçgen yaratır. Elde edilen bütün üçgenleri birbirine bağlayarak bir üçgenler ağı elde eder. Obj veya VRML türü dosyalar bu ağlardan oluşturulur. Bu sayede Blender ve SolidWorks'ün açabileceği bir dosya oluşturulmuş olur. Bu işlev, şu şekilde tanımlanabilir:

```
modeli_aktar(dosya_adı):

   üçgenler_kümesi = {}
   yüzey içindeki her dikdörtgen için:
      sütun1, satır1 = dikdörtgen.nokta1
      x1, y1, z1 = xyz_konumunu_bul(sütun1, satır1)

      sütun2, satır2 = dikdörtgen.nokta2
      x2, y2, z2 = xyz_konumunu_bul(sütun2, satır2)

      sütun3, satır3 = dikdörtgen.nokta3
      x3, y3, z3 = xyz_konumunu_bul(sütun3, satır3)

      sütun4, satır4 = dikdörtgen.nokta4
      x4, y4, z4 = xyz_konumunu_bul(sütun4, satır4)

      üçgen1 = { <x1, y1, z1>,
                 <x2, y2, z2>,
                 <x3, y3, z3> }
      üçgen2 = { <x2, y2, z2>,
                 <x3, y3, z3>,
                 <x4, y4, z4> }

      üçgenler_kümesi.ekle(üçgen1)
      üçgenler_kümesi.ekle(üçgen2)

   dosyaya_yaz(dosya_adı, üçgenler_kümesi)
```

5.5 Gauss Yöntemiyle Bulandırıcı Modül

Bu modül, YakarTarar sonuçları üzerinde Gauss bulandırma işlemi gerçekleştiren modüldür. Yüzey üzerinde dikdörtgen şeklinde tarama yapar ve z değerlerinin çevrelerinden etkilenerek bulanmalarını sağlar. Algoritması şu şekildedir:

```
modül bulandırıcı:
   bulandır(yüzey):
      yüzey içindeki her bölge için:
         merkez = orta_nokta(bölge)

         toplam_z = 0
         toplam_ağırlık = 0
         bölge içindeki her nokta için:
            toplam_z = toplam_z + z_değeri(nokta)
            çan_değeri = gauss_çanı(xy_uzaklığı(nokta, merkez))
            toplam_ağırlık = toplam_ağırlık + çan_değeri

         yeni_z = toplam_z / toplam_ağırlık
         z_değerini_yenile(yüzey, merkez, yeni_z)
```

5.6 Nesne Ayrıştırıcı Modül

Bu modül, YakarTarar sonuçlarındaki ana nesneyi diğer boşluklardan ve kir yüzeylerinden ayırıp temizleme işlemini gerçekleştiren modüldür. Bu modülün içerdiği iki önemli işlev yürüme ve ayrıştırma işlevleridir. Yürüme işlevi, kendi kendini çağırarak, başlangıç noktasından itibaren dört yöne yürür ve z değeri 0 olan noktalara çarpana kadar karşılaştığı bütün noktaları bir kümeye ekler. Bu sayede, ana nesnenin bütün noktalarının koordinatları belirlenmiş olur. Ayrıştırma işlevi ise, bu kümenin içindeki noktalar hariç bütün noktaların z değerlerini sıfır yapar ve bu sayede boşalan bütün kenar satırlarını ve sütunlarını tarama sonucundan çıkarır.

Bu modül, aşağıdaki gibi algoritmik olarak ifade edilebilir:

```
modül nesne_ayrıştırıcı:
   noktalar_kümesi = {}

   yürü(yüzey, sütun, satır):
      eğer ( (z_değeri(yüzey, nokta) > 0)
           ve (noktalar_kümesi içinde nokta yok) ) ise:
         noktalar_kümesi.ekle(nokta(sütun, satır))
         yürü(yüzey, sütun + 1, satır)
         yürü(yüzey, sütun – 1, satır)
         yürü(yüzey, sütun, satır – 1)
         yürü(yüzey, sütun, satır + 1)

   ayrıştır(yüzey, sütun, satır):
      yürü(yüzey, sütun, satır)
      yüzey içindeki her nokta için:
         eğer noktalar_kümesi içinde nokta yok ise:
            z_değerini_yenile(yüzey, nokta, 0)

      üstten_boş_satırları_sil(yüzey)
      alttan_boş_satırları_sil(yüzey)
      soldan_boş_sütunları_sil(yüzey)
      sağdan_boş_sütunları_sil(yüzey)
```

5.7 Nesne Karşılaştırıcı Modül

Bu modül, iki YakarTarar tarama sonucunun karşılaştırılmasını sağlar. Karşılaştırma işlemini yapmak için, iki adet yüzey taramasını belirtilen boyutlarda bölgelere ayırır ve bu bölgelerin ortalama z değerlerindeki farkları ayrı ayrı hesaplar. Modül, parametre olarak şu verileri almaktadır:

- Karşılaştırmak için iki adet Tam Yüzey Tarama Yapısı nesnesi
- Her bir tarama bölgesinin yüksekliği ve genişliği. Yükseklik ve genişlik bilgileri yüzde olarak belirtilir. Örneğin, (bölge_genişliği, bölge_yüksekliği) değerleri (10, 20) ise, her bir bölgenin genişliği, yüzeyin genişliğinin yüzde 10'u ve her bir bölgenin yüksekliği, yüzeyin yüksekliğinin yüzde 20'sidir.

Bu modülün algoritmik ifadesi şu şekildedir:

```
modül yüzey_karşılaştırıcı:

   değişken:
      yüzey1, yüzey2
      bölge_genişliği, bölge_yüksekliği, fark_sınır_değeri
      farklı_bölgeler = {}

   bölgelere_ayır():
      bölgeler1 = {}
      bölgeler2 = {}

      yüzde_x = 0
      yüzde_y = 0

      (yüzde_x + bölge_genişliği) <= 100 oldukça:

         (yüzde_y + bölge_yüksekliği) <= 100 oldukça:

            bölge1 = bölgesini_al(
               yüzey1, yüzde_olarak,
               yüzde_x, yüzde_y,
               yüzde_x + bölge_genişliği, yüzde_y + bölge_yüksekliği)

            bölge2 = bölgesini_al(
               yüzey2, yüzde_olarak,
               yüzde_x, yüzde_y,
               yüzde_x + bölge_genişliği, yüzde_y + bölge_yüksekliği)

            kümeye_ekle(bölgeler1, bölge1)
            kümeye_ekle(bölgeler2, bölge2)

            yüzde_y = yüzde_y + bölge_yüksekliği
            yüzde_x = yüzde_x + bölge_genişliği

      sonuç = bölgeler1, bölgeler2


   karşılaştır():

      bölgeler1, bölgeler2 = bölgelere_ayır()
      bölge_sayısı = uzunluk(bölgeler1)

      n = 0

      n < bölge_sayısı oldukça:
         fark = ortalama_z(bölgeler1(n)) – ortalama_z(bölgeler1(n))

         mutlak(fark) > fark_sınır_değeri ise:
            kümeye_ekle(farklı_bölgeler, bölgeler1(n))
```

BÖLÜM ALTI
SONUÇLAR

Yüzey tarama konusunda yapılan çalışmalar sırasında geliştirilmiş olan YakarTarar adlı yazılım projesi, hem çok işlevli, hem de maliyeti düşük bir yüzey tarama çözümü getirmektedir.

YakarTarar yazılımında, belli bir amaca yönelmek yerine, farklı amaçlar doğrultusunda kullanılabilecek genel yüzey tarama işlevleri tanımlanmıştır. Bu işlevlerin doğrudan çalıştırılması için bir kullanıcı arayüzünün yanında, C++ programlama dilinde hazırlanmış olan bir yazılım kütüphanesi de sunulmuştur. Bu sayede, yüzey taraması işlemi gerçekleştirecek farklı yazılımlar, bu işlevleri kullanarak kolayca geliştirilebilir. Sunulan işlevler arasında, yatay olarak ilerletilen ya da döndürülen nesneleri taramanın yanında, yüzeyler üzerinde düzeltme ve temizleme yapmak da bulunmaktadır. Bütün bunlar, YakarTarar'ın çok amaçlı ve çok işlevli olma hedefine yaklaşmasını sağlamaktadır.

YakarTarar'ın sunduğu çözümün düşük maliyetli olması da, herhangi bir tarama donanımından bağımsız olması sayesinde gerçekleşmektedir. Bir tarama, dikey bir ışın demeti yollayan lazer kaynağı ve herhangi bir kamera kullanılarak yapılabilir. Kalite, kameranın çözünürlüğüyle yükselir. Bu sayede, maliyet ve kalite arasındaki dengeyi kullanıcı belirler. YakarTarar, Linux tabanlı ücretsiz ve açık kaynaklı işletim sistemlerinde de çalışabildiği için, taban maliyetinden işletim sistemi maliyetini de çıkarmış olur.

YakarTarar projesi, sunduğu bu özelliklerle, kişisel araştırmaları, endüstriyi ve akademiyi de içeren geniş bir kullanım alanındaki projelere katkıda bulunmayı hedeflemektedir.

KAYNAKÇA

Coward, T. J., Scott, B. J. J., Watson, R. M. ve Richards, R. (2002). Identifying the position of an ear from a laser scan: The significance for planning rehabilitation. *International Journal of Oral and Maxillofacial Surgery*. *2002* (31), 244 – 251.

Forsyth, D. ve Ponce, J. (2002). *Computer Vision: A Modern Approach* (4. Taslak). 6 Ocak 2009. http://www.cis.udel.edu/~cer/arv/readings/forsyth-book-4.pdf .

Kanade, T., Amidi, O., Ke, Q. (2004). Real-Time and 3D Vision for Autonomous Small and Micro Air Vehicles. *43rd IEEE Conference on Decision and Control*, *2*, 1655 – 1662.

Klančnik, S., Balič, J. ve Planinšič, P. (2007). Obstacle Detection With Active Laser Triangulation. *Advances in Production Engineering & Management, 2* (2), 79 – 90.

Laser Design Inc. (2008). *FAQs on Laser Scanning: Laser Design | GKS*. 6 Ocak 2009. http://www.laserdesign.com/resources_and_downloads/faq .

Lathrop, R. R. (1997). Solder Paste Print Qualification Using Laser Triangulation. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, 20* (3), C174 – C182.

Petrovic, P. B. (2007). Rubberized Cord Thickness Measurement Based on Laser Triangulation - Part I: Technology. *FME Transactions*, *35* (2), 23 – 30.

Python Software Foundation. (2008). *About Python*. 30 Temmuz 2008. www.python.org/about .

Roberts, J. ve Corke, P. (2000). Obstacle Detection for a Mining Vehicle using a 2D Laser. *Australasian Conference on Robotics and Automation (ACRA) Konferansı, 2000.* 10 Ekim 2008. http://www.araa.asn.au/acra/acra2000/papers/paper32.pdf .

Seitz, S. ve Curless, B. (1999). *3D Photography Course Notes*, 28 Temmuz 2008, http://www.cs.cmu.edu/~seitz/course/SIGG99 .

Uyar, E., Toklu, N. E. (2008). "YakarTarar" Yazılımı ile Lazer Işınlı ve Kameralı Yüzey Tarama Uygulaması. *TOK'08 Otomatik Kontrol Ulusal Toplantısı*, *2* (2008), 704 – 709.

Winkelbach, S., Molkenstruck, S. ve Wahl, F. M. (2006). Low-Cost Laser Range Scanner and Fast Surface Registration Approach. *German Association for Pattern Recognition (DAGM) Sempozyumu, 2006.* 28 Temmuz 2008. http://www.cs.tubs.de/rob/literatur/download/swi_2006_09_konferenz_dagm.pdf .