DOKUZ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KAMERALI LAZER TARAMA SİSTEMİ İLE NESNE SINIFLANDIRMASI VE UYGULAMALARI

Ferhat İdgü BALBOZAN

Ekim, 2011
İZMİR

KAMERALI LAZER TARAMA SİSTEMİ İLE NESNE SINIFLANDIRMASI VE UYGULAMALARI

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
Mekatronik Mühendisliği Bölümü, Mekatronik Mühendisliği Anabilim Dalı

Ferhat İdgü BALBOZAN

Ekim, 2011
İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

FERHAT İDGÜ BALBOZAN tarafından **YAR. DOÇ. DR. NALAN ÖZKURT** yönetiminde hazırlanan **"KAMERALI LAZER TARAMA SİSTEMİ İLE NESNE SINIFLANDIRMASI VE UYGULAMALARI"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

.....

Yar. Doç. Dr. Nalan Özkurt

Danışman

.....

Doç. Dr. Zeki KIRAL

Jüri Üyesi

.....

Y. Doç. Dr. M. Makinacı

Jüri Üyesi

.....

Prof. Dr. Mustafa SABUNCU

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Bu çalışmada, beni yaratıcı fikirleriyle destekleyen arkadaşım N. Engin TOKLU'ya, çalışmanın başlamasına bulunduğu katkılardan dolayı Prof. Dr. Erol UYAR'a, hem bilgi hem de manevi yönden bana yardımını hiç esirgemeyen proje danışmanım Yar. Doç. Dr. Nalan ÖZKURT'a ve hayat boyu desteklerini hiç esirgemeyen aileme teşekkürü borç bilirim.

Ferhat İdgü BALBOZAN

KAMERALI LAZER TARAMA SİSTEMİ İLE NESNE SINIFLANDIRMASI VE UYGULAMALARI

ÖZ

Bir nesne grubunda, nesnelerin birbirinden fiziksel olarak farklı olduğu özelliklerini göz önünde bulundurarak yapılan ayrıştırma işlemi nesne sınıflandırması olarak adlandırılır. Sınıflandırma işlemi aynı zamanda, bir nesnenin kendisine ait olabilecek özellikler, paternler ve fiziksel yapısının belirlenen sınıfa aitlik durumunun ve yakınlığının belirlenmesidir.

Nesne tanıma ve sınıflandırma işlemi için mevcut yazılım ortamları, kütüphaneler ve derleyiciler bulunmaktadır. Bununla birlikte mikro işlemcili kameralar gibi donanımsal sistemler de mevcuttur. Fakat bu yazılım araçları lisanslı, kullanılan cihazlar yüksek maliyetlidir. Bu çalışmada tasarlanan kütüphaneler ile nesne tanıma ve sınıflandırma işleminin maliyetinin düşürülmesi hedeflenmektedir. Nesne tanıma ve sınıflandırma sistemi donanım olarak sadece bir çizgi lazer kaynağı ve bir kameraya ihtiyaç duymaktadır. Bu donanımla üç boyutlu nesne taraması gerçekleştirilip yüzey taraması yapılan nesneler yapay sinir ağları aracılığıyla tanınmakta ve sınıflandırmaktadır.

Anahtar Sözcükler: Görüntü İşleme, Kohonen Ağı, Nesne Tanıma, Patern Tanıma, Yapay Sinir Ağları, Zincir Kodu

OBJECT CLASSIFICATION USING LASER SCANNING WITH CAMERA AND ITS APPLICATIONS

ABSTRACT

In a group of object, defining them with their physical differences is called object classification. Object classification also can be defined by object's special features, pattern and structure that the object belongs to a class or how close it is.

There are development environments and software libraries designed for object classification. Devices like micro controller aided cameras are also used for object classification. But these devices and development environments are licensed and their costs are high. In this study purpose is to reduce the cost of object classification by using designed software libraries. These libraries use image processing and artificial neural network methods. Object recognition and classification system only uses an camera and a laser source as hardware. Object classification is done by using this hardware configuration and designed software libraries.

Keywords: Artificial Neural Networks, Chain Code, Image Processing, Kohonen Network, Object Recognition, Pattern Recognition

İÇİNDEKİLER

Sayfa

BÖLÜM BİR
GİRİŞ

1.1 Nesne Tanıma ve Sınıflandırma Hakkında

Tanıma veya sınıflandırma işlemi, herhangi bir nesneyi, görüntüyü ya da paterni bulma, ayırt etme işlemidir. Nesne tanıma, sınıflandırma ve benzetme uygulamaları, sanayide, üretimde ve gıda sektöründe yoğun olarak kullanılmaktadır. Günümüzde üretimde öne çıkan unsurlar zaman ve kalitedir. Üretim alanında ham madde olarak birbiriyle aynı fakat şekil olarak birbirlerinden farklı bir çok malzeme üretilmektedir. Bu malzemelerin üretim aşamasında farklı işlemlerden geçirilip, üretim sonrası aşamalarda birbirinden ayrılması, sınıflandırılması gerekmektedir. Aynı yöntem gıda sektöründe zarar görmemiş defne yapraklarının ayrılması ve bakliyat ürünlerinden hasarsız olanların tespiti gibi uygulamalarda da kullanılmaktadır. Bu işlemi mümkün olduğu kadar hatası az ve kısa zamanda yapmak bu konunun ilerleme noktalarıdır.

Nesne tanıma, biz insanlar için sıradan ve rahatlıkla çözülebilen ama bilgisayar için kompleks bir problemdir. Bu problemin çözümü için birçok yaklaşım geliştirilmiştir. Bu yaklaşımlarda en iyi sonuçlar yapay sinir ağları kullanarak alınmıştır.

Yapılan bir çalışmada üç boyutlu nesneler bir robot kolu yardımıyla algılanmış ve yapay sinir ağları kullanarak tanıma işlemi gerçekleştirilmiştir. Nesne modelleme için 3 parmaklı ve 10 eklemli bir robot eli kullanılmış, parmaklardaki sensörler ile cismin geometrik modeli çıkartılmıştır. Çıkartılan modele ait bilgiler bir vektör haline getirilmiş ve yapay sinir ağları tarafından girdi olarak kullanılmıştır. Yapay sinir ağı modeli olarak 30 nöronlu bir Kohonen Haritası (“Kohonen Map”) kullanılmıştır. Kohonen haritası toplam 10 nesne ile eğitilmiş, her nesne için 20 özellik vektörü kullanılmıştır. Kohonen haritasının tercih edilme nedeni, Kohonen haritalarının tek kademeli, eğitim aşamasının denetimsiz ve kompleks matematik işlemleri gerektirmeyen, hızlı bir yapay sinir ağları yaklaşımı olmasıdır. Çalışmanın sonucuna baktığımızda robot kolunun bir nesneyi ortalama 4 veya 5 kavrayışta, yaklaşık %90 oranında tanıma başarısına ulaştığını görüyoruz (Faldella, Fringuelli, Passeri, Rosi, 1997),

Bir diğer yapılan çalışmada hareketli bir kamera sistemi ile gerçek dünyadaki üç boyutlu nesnelerin modellenmesi yapılmış, daha sonra Kohonen haritalarından türeyen gerektiğinde büyüyen ağ (“GWR, growing-when-required”) yaklaşımıyla nesnelerin tanınması gerçekleştirilmiştir. Gerektiğinde büyüyen ağlar, Kohonen haritalarıyla aynı yapıda olup boyutları dinamik olarak değişebilmektedir. Bu yaklaşımın kullanılmasının nedeni, özellik vektörünün boyutlarının değişken olması olarak açıklanmıştır. Nesnelerin bilgilerinin alınması için nesnenin arka plandan ayrılması gerekmektedir. Fakat gerçek dünya uygulamalarında renk fark kullanarak bu işlem yapılamaz.Dolayısıyla bu çalışmada farklı method kullanılmış, hareketli kamera nesneye odaklanarak hareket ettirilmiş ve “birlikte hareket eden noktalar, aynı düzlemdedir” prensibiyle nesnenin arka plandan ayrımı gerçekleştirilmiştir. Nesne üzerindeki sabit noktalar belirlenmiş ve bu noktalar arasında komşuluk ilişkisi kurulmuştur. Elde edilen bu bilgi ile yapay sinir ağı eğitilir. Çalışmanın test aşamasında 7 farklı nesne kullanılmış, 2 farklı deney yapılmıştır. İlk deneyde nesneler sabit bir açıdan resimleri çekilerek tanıma işlemi gerçekleştirilmiş, ikinci deneyde ise 10'ar derecelik kaymalarla nesneler taranmıştır. İki sonuç karşılaştırldığında başarı oranında %73'ten %90'a varan bir düzelme görülmüştür (Kootstra, Ypma, de Boer, 2007).

1.2 Yapılan Çalışma Hakkında

Nesne sınıflandırmak için piyasada çeşitli çözümler bulunmaktadır. Bu çalışmanın amacı piyasaya düşük maliyetli bir alternatif getirmektir. Çalışmada kayan bantlı bir sistem oluşturulmuştur. Bu sistem üzerinden geçen nesnelerin yüzeyleri taranmış ve alınan bilgiler dahilinde sınıflandırmaları yapılmıştır.

Bu çalışmadaki nesne tanıma ve sınıflandırma sistemi, incelenen nesnenin lazerle taranması, taranan nesnenin sınırlarının çıkarılıp kodlanması ve çıkarılan kodların sınıflandırılması adımlarıyla çalışmaktadır. Bu üç adımın ayrıntılı açıklamalarına ilerideki bölümlerde değinilmektedir.

1.3 Tezin içeriği

Yapılan çalışma temel olarak üç bölümde incelenmektedir. İkinci bölümde nesnelerin sistemin ilk aşaması olan tarama kısmı anlatılmaktadır. Tarama kısmında

cisimler lazerli tarama yöntemiyle taranıp yüzey bilgileri alınmıştır. Yöntem hakkında daha geniş bilgi ikinci bölümde mevcuttur. Sistemin ikinci aşaması görüntü işleme işlemlerinin yapıldığı kısımdır. Bu kısımda taranan cismin yüzey bilgisi yorumlanıp, benzersiz bir bilgi elde edilmesi amaçlanmıştır. Bu aşamada kullanılan yöntemler üçüncü bölümde ayrıntısıyla açıklanmıştır. Sistemin son aşaması nesnelerin sınıflandırılması kısmıdır. Bu kısımda yapay sinir ağlarından faydalanılmaktadır. Son aşama, yapay sinir ağlarının sınıflandırılacak nesnelerden çıkarılan benzersiz bilgilerle eğitilmesini ve bu işlemin ardından taranan nesnenin sınıflandırılmasını kapsar. Eğitim ve sınıflandırma yöntemi hakkında ayrıntılı bilgi dördüncü bölümde sunulmaktadır.

Nesne tanıma ve sınıflandırma sistemini oluşuran kısımlar anlatıldıktan sonra, sistemin nasıl çalıştığı hakkında bilgi verilecektir. Sistemin işleyişi, kod örnekleri ve ekran görüntüleriyle adım adım beşinci bölümde anlatılacaktır. Tezin altıncı bölümü sistem üzerinde yapılan deneylere ayrılmıştır. Bu bölümde test sonuçları tartışılacak ve sistem üzerinde yapılabilinecek gelecek çalışmalar tartışılacaktır. Son bölümde ise çalışmanın sonucu ve getirilerinden bahsedilmiştir.

BÖLÜM İKİ
NESNE TARAMA SİSTEMİ

2.1 Donanım

Nesne tanıma ve sınıflandırma sisteminde tarama işlemi için kullanılan kamera ve çizgi lazer kaynağı yürüyen bir bant üzerine kurulmuştur. Taranacak nesneler yürüyen bantla kaydırlarak, bant üzerine bir köprü aracılığıyla kurulan tarama sistemi taramayı gerçekleştirmektedir. Sistem bir çizgi lazer kaynağı ve farklı işletim sistemlerinde aynı ayarlarda çalışması için evrensel kamera sürücüsüyle uyumlu bir kamera kullanılmıştır.

Yürüyen bantın ilerlemesi için bir adet 0-12 V gerilime sahip sincap kafesli asenkron motor kullanılmıştır. Motorun kontrolü için Telemecanique Altivar 71 model motor sürücüsü kullanılmıştır. Sürücüye yerleştirilen bir potansiyometre aracılığıyla bandın hız kontrolü manuel olarak ayarlanabilmektedir. Kullanılan enstrümanlar Şekil 2.1'de gözlemlenebilir.

Şekil 2.1 Yürüyen bant, motor sürücüsü ve hız kontrolünde kullanılan potansiyometre.

2.2 Yazılım

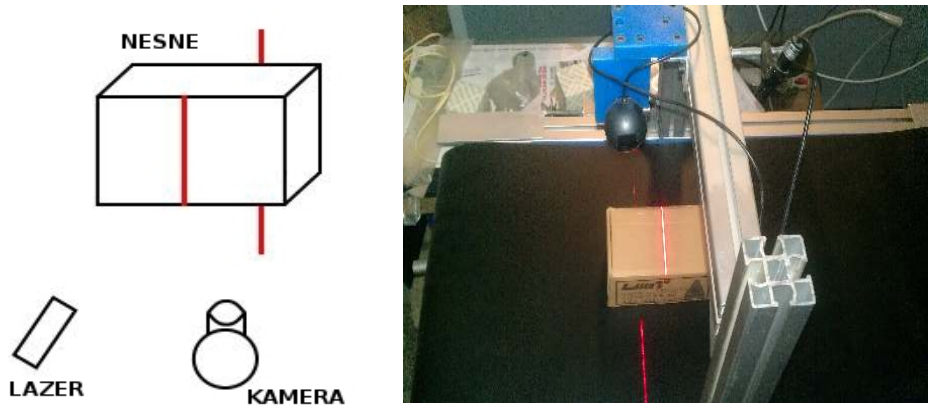
Nesne tanıma ve sınıflandırma sisteminde, lazerli tarama için Yakartarar adlı bir kütüphane kullanılmaktadır. YakarTarar, lazer üçgenlemesi (“Laser Triangulation”)

yöntemini (Lathrop, 1997; Laser Design Inc., 2008) uyarlayan bir C++ kütüphanesidir (Toklu, 2009; Bilgisayarlı Görüntü İşleme ile 3–Boyutlu Nesne Tarama ve Modelleme). Kütüphane tamamen taşınabilirdir ve işletim sisteminden bağımsızdır. YakarTarar kütüphanesini kullanarak tarama yapmak için sadece bir çizgi lazer kaynağı ve kamera yeterlidir. Dairesel tarama ve düzlem üzerinde tarama olmak üzere iki farklı nesne tarama yöntemini kullanır. YakarTarar'ın en büyük avantajlarından biri üç boyutlu tarama işlemlerinin maliyetini aşağıya çekmesidir. Kullandığı donanım minimal olduğu için farklı ortamlarda kısa bir kalibrasyon işlemiyle tarama işlemini gerçekleştirebilmektedir.

2.3 Tarama Sisteminin İşleyişi

Nesne tanıma ve sınıflandırma sisteminde, nesnelerin bilgilerinin bilgisayar ortamına aktarılması lazerli tarama ile yapılmaktadır. Lazer taramasında lazer üçgenlemesi yöntemi kullanılmaktadır. Lazer üçgenlemesi yönteminde, bir çizgi lazer kaynağı tarama yapılacak yüzeye, bir kamera ile farklı açıdan bakacak şekilde yerleştirilir. Çizgi lazer kaynağının, taraması yapılan nesne üzerindeki ve yüzeydeki iz düşümü farkı kamera tarafından algılanır. Yöntem, bu fark ile yükseklik arasında bağıntı kurarak, nesnenin üç boyutlu modelini çıkartır.

Lazer üçgenlemesi yönteminin, nesne tanıma ve sınıflandırma sisteminde yerleşimi aşağıdaki Şekil 2.2'de gösterilmiştir. Görüldüğü gibi çizgi lazer kaynağı dikdörtgen kutuya eğik bakarken, kamera dik bir açıyla yerleştirilmiştir. Böylece sistem, kamera aracılığıyla iz düşümü farkını algılamaktadır.



Şekil 2.2 Lazer üçgenlemesi yönteminde lazer kaynağı, kamera ve nesnenin konumu

Nesne tanıma ve sınıflandırma sisteminde kamera, lazer kaynağı konumları sabit ve tarama yüzeyiyle mesafeleri korunmaktadır. Bu sayede sistem bir kez kalibre edilip, bu kalibrasyon ayarlarıyla taramalar gerçekleştirilmektedir.

BÖLÜM ÜÇ
GÖRÜNTÜ İŞLEME

3.1 Görüntü İşleme Teknikleri

Görüntü işleme ya da diğer adıyla sayısal görüntü işleme, sayısal bir resmin girdi olarak alınıp amaca yönelik sinyal işleme algoritmaları kullanılarak, sonuçta istenilen görüntünün çıktı olarak elde edilmesidir. Görüntü işleme teknikleri, medikal, astronomi, bilgi arama gibi birçok alanda, bilgiyi filtrelemek ve bilgiye daha hızlı ve hatasız ulaşmak için kullanılır.

Görüntü işleme teknikleri kullanarak bir resim üzerinde birçok değişiklik yapmak mümkündür. Bir resmin sıkıştırılması, bozuk bir resmin onarımı, resimdeki gürültülerin kaldırılması, renkli resim işleme, resmin içeriklerinin ayrılması ve resmin tekrar işlenip görüntülenmesi gibi birçok işlem görüntü işleme teknikleri ile yapılmaktadır.

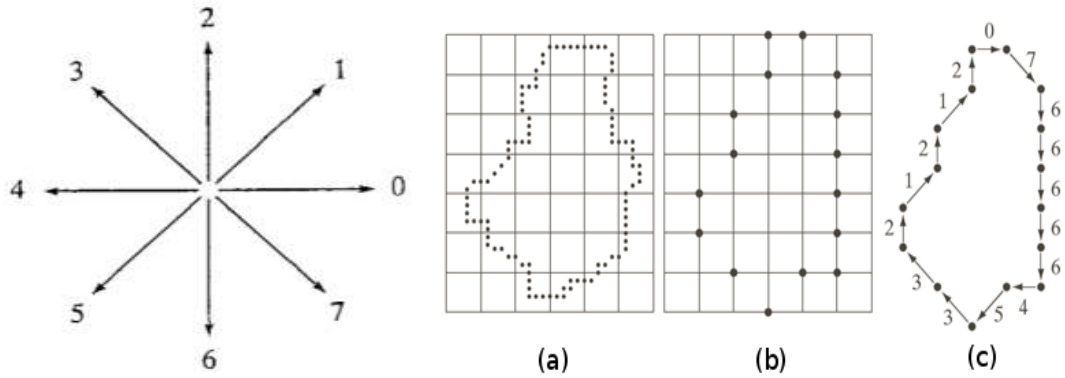
Sayısal görüntü işlemede renkli bir resim ile gri tonlardaki bir resim farklı ifade edilir. Renkli resimde her noktanın kırmızı, yeşil ve mavi (RGB) olmak üzere 3 değeri vardır. Bu değerlerin sayısal değişimlerinden renkler elde edilir. Gri tonlarda ise bu durum tek bir değer üzerinden gösterilir. Gri tonlardaki bir resimden bir bilgi edinilmek istendiğinde eşikleme tekniği kullanılır. Eşikleme tekniği resimde yakalanmak istenen ayrıntıya göre tanımlanan bir değerin altında kalan yerlerin kaybolması, üstünde kalan yerlerin ise belirginleştirilmesidir. Bu işlem aynı zamanda resmi siyah beyaz hale getirir. Bu çalışmada eşikleme değeri nesnenin tespit edilmesi için kullanılmıştır.

Görsel dizayn işlemlerinde görüntü işleme teknikleri kullanan “adobe photoshop” ve “gimp” gibi hazır programlar mevcuttur. Fakat dijital bir görüntüden spesifik bir bilgi elde edilmek isteniyorsa ya da amaca yönelik özel bir yöntem uygulanması gerekiyorsa, “opencv” gibi birçok dilde uygulanabilen görüntü işleme kütüphaneleri kullanılmaktadır. Ayrıca “Matlab” da içinde görüntü işleme araçları barındırmaktadır.

3.2 Zincir Kod Tanımı

Zincir kodu (Chain Code) bir nesnenin bir düzlem üzerindeki rotasyonunu tespit etmek için kullanılan bir görüntü işleme tekniğidir. Temel prensip, bir nesnenin etrafının taranarak sınırlarının belirlenmesidir. Yani zincir kodu resimdeki bir nesnenin sınırlarını belirleyerek, nesnenin kenar bilgisini içerir.

Zincir kodu genellikle iki boyutlu uygulamalarda kullanılır. Bu uygulamalar resimdeki bir şeklin sınırlarının kodlanmasıdır. Bunun için bazı görüntü işleme teknikleri kullanılarak, ilk olarak resimdeki şeklin arka plandan ayırılması gerekir. Şekil arka plandan ayırt edilebilir hale gelince, şeklin sınırında herhangi bir noktadan başlayarak etrafı taranır. Taraması gerçekleştirilen her sınır noktası, bir önceki noktayla olan konum farkına göre kodlanır. Kodlama yapılırken Şekil 3.1'de gösterilen yönler ve o yönlere ait kodlar kullanılmaktadır ve bu kodlamaya göre zincir kodunun nasıl belirlendiği örnek bir şekil üzerinde aşama aşama gösterilmiştir. İşlemin sonunda çıkartılan bu kod kaydedilir ve o şeklin zincir kodunu oluşturur.



Şekil 3.1 Zincir kodunun yönlere göre kodlanması (a) Genel hatlarıyla şekil (b) Şeklin sınırları üzerinde seçilen noktalar (c) Zincir kodunun oluşturulması.

3.3 Nesne Tanıma ve Sınıflandırma Sisteminde Bilginin Elde Edilmesi

Nesne tanıma ve sınıflandırma sisteminde, görüntü işleme ile ilgili işlemlerin tümünü çözmek için NTSShape kütüphanesi tarafımdan dizayn edilmiştir. Tarama sisteminden alınan tarama sonuçlarını işleme alan kütüphanedir. Tarama sonrası alınan şekil değerleri matrise yerleştirildikten sonra işleme alınır. Matrisin hatalardan

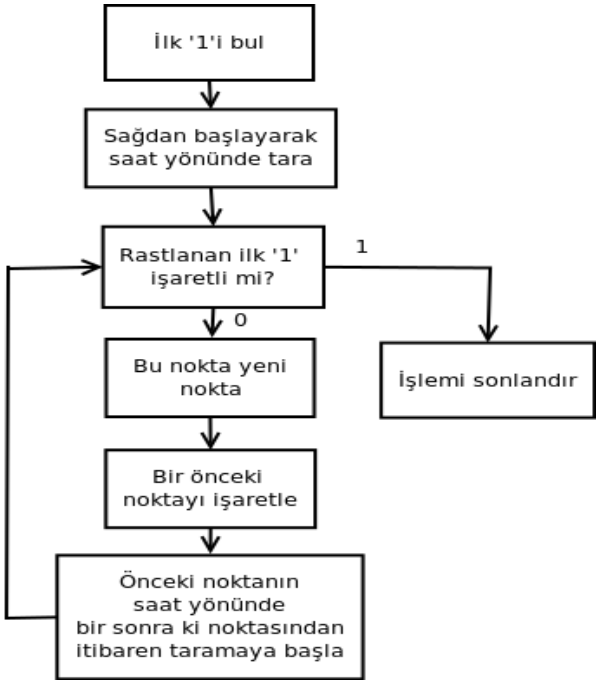
ayıklanıp, filtrelenmesini bu kütüphane çözer. Matrisin içerdiği şekil tesbit edilir, sınırları taranır ve sonuç olarak zincir kodu oluşturulur.

Nesne tanıma ve sınıflandırma sisteminde nesnelerin sınıflandırılması için her nesneye ait, benzersiz bir bilginin elde edilmesi gerekmektedir. Bu nedenle cisimlerin fiziksel bilgileri bilgisayar ortamına aktarıldıktan sonra bir dizi işlemden geçirilmektedir. Bu işlemler aslında cisimden zincir kodunun elde edilmesi için hazırlık aşamasıdır.

Sistemde nesnenin zincir kodunun çıkarılması için ilk olarak, alınan üç boyutlu tarama sonucunun iki boyuta indirgenmesidir. Bu işlem NTSShape kütüphanesi içindeki bir foksiyo aracılığıyla yapılmaktadır. Fonksiyon parametre olarak aldığı bir eşik değeriyle tarama sonucunu '0' ve '1' lerden oluşan bir matris haline getirir.

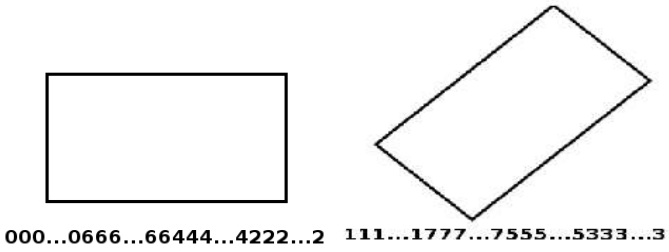
Yapılan ikinci hazırlık aşaması, sonuç olarak elde edilen iki boyutlu matrisin çözünürlüğünün düşürülmesidir. Bu işlem zincir kod çıkarma aşamasının hızlı olması ve sonuçta elde edilecek zincir kodun boyutunun düşük olması için gereklidir. Çözünürlük düşürülmesi işlemi 1/3 oranında yapılmıştır. Bu oranla tanıma işlemi için yeterli ayrıntı elde tutulmuş ve zincir kod boyutu önemli ölçüte azaltılmıştır.

Sonuç matrisini iki boyutlu hale getirip, çözünürlüğü düşürüldükten sonra nesnenin zincir kodunun çıkartılması işlemi başlatılmaktadır. Bu işlem NTSShape kütüphanesindeki bir fonksiyonla yapılmıştır. İki boyutlu matris sol üstten başlayarak taranır ve nesneye ulaşıldığında nesnenin etrafını dolaşarak taramaya devam eder. Nesnenin çevresi tamamen dolaşıldığında zincir kodu çıkrtma işlemi tamamlanmış olur. Aşağıdaki Şekil 3.2'de zincir kod çıkartma işleminin diyagramı görülmektedir.



Şekil 3.2 Zincir kod çıkartılması işleminin diyagramı.

Zinci kodu işlemi sayesinde, nesnenin çevre uzunluğuna bağlı olarak uzunluğu değişen, nesneye ait benzersiz bir vektör elde edilmiş olunur. Bu vektörün her elemanı 'mod 7' altında '1' ile toplanır. Böylelikle 45 derecelik rotasyonda cismin çevre bilgisine ulaşılır. Bu işlem 7 kez yapıldığında nesnenin 45 derecelik farklarla 8 farklı yöndeki rotasyon bilgisine sahip olunur. Bu bilgiler kaydedilir. Sınıflandırma esnasında, nesne yürüyen bant üzerinde rotasyona uğrasa bile tanıma işlemi gerçekleştirilir. Aşağıdaki Şekil 3.3'te dörtgen bir nesneyi ve onun 45 derece rotasyona uğramış halinin zincir kodlarıyla birlikte karşılaştırılması gözlemlenmektedir.



Şekil 3.3 Zincir kodlarıyla dörtgen nesne ve 45 derecelik rotasyonu.

BÖLÜM DÖRT
NESNE SINIFLANDIRMA SİSTEMİ

4.1 Yapay Sinir Ağları

Sistemde sınıflandırma yapay sinir ağları ile yapılmıştır. Yapay sinir ağları, model olarak insan sinir sistemini örnek alan bir programlama yaklaşımıdır. Geleneksel programlama metodlarında beklenen girdilere göre sonuçlar alınır. Bir programın kendisini değişen koşullara adapte edebilmesi mümkün değildir. Dolayısıyla Beklenenin dışında verilen bir girdi, programın doğru sonuç vermemesine neden olur. İşte yapay sinir ağları bu soruna çözüm sağlamak için doğmuştur. Yapay sinir ağları metodları kullanılarak yazılmış bir program öğrenme kabiliyetine sahiptir. İlk başta, eğitim aşamasından geçen program çalışma aşamasında gelen bilgileri eğitildiği bilgi kümesine göre yorumlayabilmektedir. Girdilerin tahmin edilemediği yüz tanıma, parmak izi ayırma, yazı karakteri tanımlama, sınıflandırma gibi birçok kompleks sorun yapay sinir ağları kullanarak çözümlenebilir.

Yapay sinir ağları üzerine araştırmaların başlaması 1940'lı yıllara kadar uzanır. 1943 yılında Warren McCulloch tarafından yayınlanan bir makalede Turing Makinası (“Turing Machine”) uygulaması, yaratılan bir grup yapay nöron üzerinde çalıştırılmaya çalışılmıştır. Bu tarihlerdeki olanakların kısıtlı olması, bu alanın gelişiminin ancak 1990'lardan sonra hızlanmasına neden olmuştur (ODTÜ Bilgisayar Topluluğu Elektornik Dergi, Şubat 2008).

Yapay sinir ağları yapı olarak biyolojik sinir ağlarını simüle ederler. Nöronların dizilimindeki gibi birbirine bağlı değişkenler sisteminden oluşurlar. Genel prensip sürekli aktive olan nöronun (değişken) ağırlığının daha fazla arttırılarak o problemin çözümünde daha etkin bir yol olduğunun belirlenmesidir. Yapay sinir ağları konusunda farklı yaklaşımlar mevcuttur. Eğitim aşaması bütün yapay sinir ağları için en önemli kısımdır. Eğitim şekillerine göre denetimli (supervised), denetimsiz (unsupervised) ve takviyeli (reinforced) olarak incelenebilirler. Denetimli eğitim yapısına sahip olan yapay sinir ağları eğitim süresince bir çözüm fonksiyonu üretir. Eğitimin devamında alınan her sonuçta kendi ürettiği cevapla karşılaştırmalı bir şekilde ağırlıklarını düzenler. Üretilen çıktı, gerçek sonuca en yakın olduğu anda eğitim tamamlanmış olur. Denetimsiz eğitimde ise herhangi bir karşılaştırma yoktur.

Yapay sinir ağına ilk girdi verilir ve sorunun yapısına göre sistem kendisini eğitir. Takviyeli eğitim yapı olarak denetimli öğrenmeye benzer. Ağ deneme yanılma yöntemiyle kendini eğitir. Denetimli eğitimden farklı olan ise girdi ve çıktının belirsiz olmasıdır. Eğitim aşamasında üretilen her cevabın gerçek çözümle yakınlığı önemli değildir. Geri yayılma (“Back propagation”), Adaline gibi yapay sinir ağları denetimli eğitim yapısına sahip ağlara örnek verilirken, Kohenen ağları ve adaptif rezonans teori (“Adaptive Resonans Theory”) gibi ağlar da denetimsiz eğitim yapısına sahip ağlara örnektir.

4.2 Kendini Organize Eden Haritalar

Nesne tanıma ve sınıflandirma sisteminde, nesnelerin tanınma aşaması için yapay sinir ağları kullanılmaktadır. Yapay sinir ağları, insanlardaki sinir ağı yapısını örnek alarak oluşturulmuş bir yaklaşımdır. Yapay sinir ağlarında bir çok farklı yaklaşım mevcuttur. Bu çalışmada kullanılan yapı Kendini Organize eden Harita ("Self Organizing Map") adı altındaki bir yapay sinir ağları yaklaşımıdır.

Kendini organize eden haritalar (KOH) Teuvo Kohonen tarafından bulunmuştur, bu nedenle aynı zamanda “Kohonen Haritaları” olarak da adlandırılmaktadır. Kendini organize eden haritalar denetimsiz öğrenme (“Unsupervised training”) özelliğine sahip ve sınıflandırma problemlerinde tercih edilen bir yapay sinir ağları yöntemidir.

Her yapay sinir ağı yöntemi gibi kendine organize eden haritaları da işleyiş bakımından eğitim aşaması ve çalışma aşaması olarak iki bölümde inceleyebiliriz. Eğitim aşaması, haritanın bilgi bakımından beslenilip, kendini eğitip hazırlamasından oluşmaktadır. Çalışma aşamasında ise sunulan bilginin nereye ait olduğunu saptamaktadır.

Kendini organize eden haritaları yapı bakımından açıklayacak olursak bir matrise benzetebiliriz. Bu matristeki her bir nokta aslında bir nörondur. Bu nöronlar bizim sınıflandırmak istediğimiz bilgileri içermektedir. Bu bilgiler harita yaratılırken rastgele olarak dağıtılmıştır. Eğitim aşamasında bu bilgileri mümkün olduğunca gerçek hallerine yakınlaştırmaya çalışılır. Tanımlama aşamasında ise bu şekillerin ait olduğu sınıfı haritadaki bilgilerle karşılaştırarak belirler.

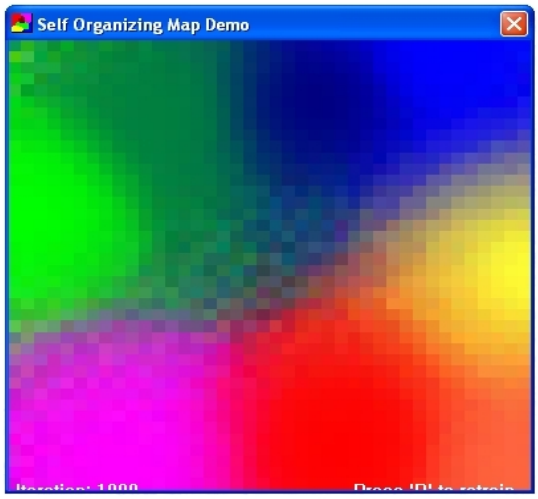
Kendini organize eden haritaların çalışma şeklini açıklamak için bir renk ayırma örneği kullanalım. Bilgisayar ortamında renkler kırmızı, yeşil ve mavi (RGB) olarak üç boyutlu bir veri kümesi olarak tutulmaktadır. Bu da haritamızın her bir nöronunun üç tane değeri olacağı anlamına gelir. Haritamız ilk yaratıldığı anda her bir nöron da rastgele olarak değer alır. Bu ilk aşamada haritamıza baktığımızda Şekil 4.1'te görebileceğimiz gibi rengarenk bir matris olarak gözükür.

Şekil 4.1 Kendini organize eden haritanın ilk aşaması, her bir piksel bir nöronu temsil etmekte ve renk değerleri rastgele olarak dağılmış durumda.

Eğitim aşamasında haritamıza sınıflandırılacak değerleri belirtmek gerekmektedir. Bu değerlere göre harita kendi içinde tarama yaparak, hangi nöronun seçilmiş değere en yakın olduğunu bulmaktadır. Mesela eğitim için kullandığımız renklerden biri kırmızı ise, değeri [255,0,0] gibi bir vektör olmalıdır. Bu vektör, kırmızı değerin maksimum, yeşil ve mavinin ise minumum olduğu anlamına gelir. Kırmızıya en yakın nokta ya da nöron bulunduktan sonra, bu nokta kırmızı için en uygun nokta ("Best matching unit") olarak adlandırılır. Sınıflandırılmak istenen her renk için bu işlem tekrarlanır. Sonuç olarak sınıfları oluşturan her olgu için haritamızda en uygun noktalar bulunmuş olur. Bu noktalar o sınıfların merkez noktalarını oluşturur.

Eğitim aşamasının bir diğer kısmı ise bulunan en uygun noktaların daha da belirginleştirilmesidir. Belirginleştirme işlemi en uygun noktanın ait olduğu sınıfın bilgisine daha çok benzetilmesidir. Fakat bu benzetme işlemi sadece merkez noktada

değil çevre nöronlarda da uygulanır. Çevre nöronlara uygulanan benzetme işleminin etkisi merkezden uzaklaştıkça azaltılmaktadır. Bu işlem haritadaki bütün noktalara defalarca uygulanır. Aşağıdaki Şekil 4.2'te 1000 iterasyon sonunda Şekil 1.3'teki haritamızın geldiği hali gözlemleyebiliriz.



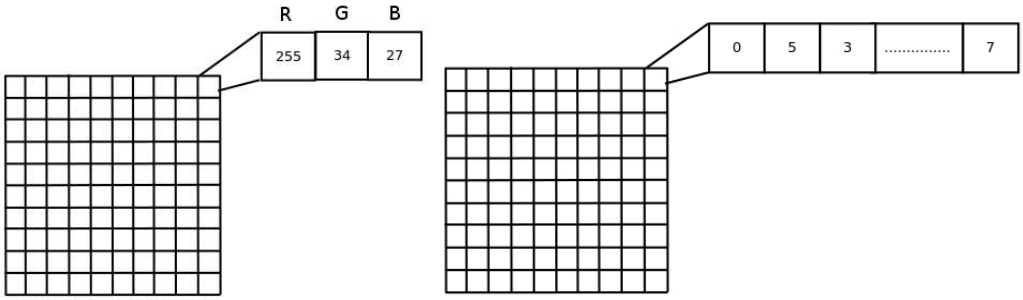
Şekil 4.2 Kendinizi organize eden haritanın tanımlanan renklere göre bin iterasyon sonucu oluşan durum, renklerin birbirlerinde ayrışması.

Sonuç itibariyle oluşan haritada sınıflandırılmak istenen bilgilere ait ayrı alanlar oluşmaktadır. Dolayısıyla sınıfını bilmediğiz bir bilgiye haritadaki en yakın noktayı bularak, o bilginin ait olduğu sınıfı ve o sınıfa ne oranda ait olduğunu bulabilmekteyiz. Bu da kendini organize eden haritaların temel çalışma prensibini oluşturmaktadır.

4.3 Sınıflandırma

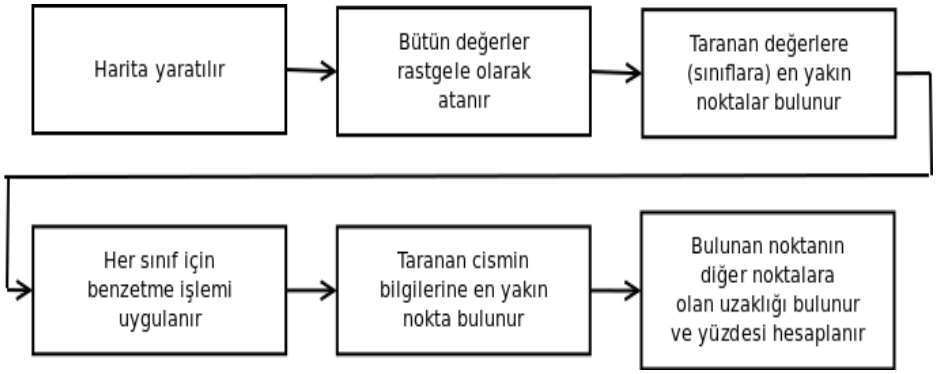
Nesne tanıma ve sınıflandırma sisteminin son aşaması olan yapay sinir ağlarıyla sınıflandırma kısmı için tarafımdan NTSSOM adlı bir C++ kütüphanesi geliştirilmiştir. Sistemin yapay sinir ağı metodları ve bütün eğitim ve sınıflandırma işlemleri bu kütüphanenin içindedir. Sınıflandırmada kullanılan kendini organize eden harita bu kütüphane ile yaratılır. KOH'a sınıflandırılacak bilgiler verilir ve bu kütüphanedeki metodlarla eğitim işlemi sağlanır. Eğitim sonrasında, sınıflandırılacak nesne taranır ve zincir kodu alınır. Nesnenin sınıflandırılması için zincir kodunun benzerinin haritada taranması yine bu kütüphanede gerçekleştirilir.

Sistemde sınıflandırma işlemi için KOH yapısı biraz değiştirilerek kullanılmıştır. Renk sınıflandırma örneğinden farklı olarak haritadaki her nöron 3 değil uzunluğu belirsiz bir vektör barındırmaktadır. Resim 4.3'te iki harite arasındaki fark görülmektedir. Dolayısıyla haritanın hafızada yaratılışının dinamik olarak yapılması gerekmektedir. Bu nedenle zincir kodunun yani özellik vektörünün bulunmasından sonra, nesnelerin vektörlerinden boyutu en büyük olan seçilir. Harita bu vektörün uzunluğuna sahip nöronlarla yaratılır.



Şekil 4.3 Renk sınıflandırma ve nesne sınıflandırma haritalarıdaki nöronlar arasındaki fark.

Haritanın yaratılma aşamasının hemen ardından haritanın bütün nöronlarındaki vektörlerin her değerine '0' ile '7' arasında rastgele sayılar atanır. Bu aşamadan sonra harita eğitim işlemine hazırlanmış olur. Sınıflandırma sisteminin yapısı aşama aşama Şekil 4.4'te görülmektedir.



Şekil 4.4 Sınıflandırma sisteminin adım adım aşamaları.

Eğitim aşamasında her nesne sınıfına ait vektörlere en çok benzeyen noktalar bulunur. Bu noktaların her biri "En Uygun Nokta" olarak adlandırılırlar. Her nesne sınıfı için bulunan en uygun noktalar kaydedilir. Bu kısımdan sonra haritanın kendisini eğitme aşaması gerçekleşir. Bu aşamada haritadaki en uygun noktalar göz önüne alınarak, haritanın tamamı özellik vektörlerine benzetilir. Geleneksel KOH yapısıyla sistemdeki farklar burada ortaya çıkmaktadır. KOH eğitim algoritmasında benzetme katsayısı sabit, en uygun noktaya olan uzaklık ve iterasyon sayısı ise benzetmeye ters orantılıdır. Yani başka bir değişle uzaklık ve iterasyon sayısı arttıkça nöronların özellik vektörüne benzetilmesi azalmaktadır. Nesne tanıma ve sınıflandırma sisteminde ise benzetme katsayısı değişken olarak kullanılmıştır. Her iterasyonda o andaki mevcut noktanın vektör elemanı ile benzetilmek istenen özellik vektörünün elemanı arasındaki fark ikiye bölünerek benzetme katsayısı olarak kullanılmıştır. Bunun amacı daha az iterasyonda daha yakın bir benzetme oranı yakalamaktır. Kullanıcının belirlediği iterasyon sayısı kadar benzetme işlemi tekrarlanmaktadır. Benzetme işleminin bitirilmesiyle haritanın eğitimi tamamlanmış olur.

Sistemin son aşaması sınıflandırma aşamasıdır. Sınıflandırılmak istenen nesne taranır ve zincir kodu çıkartılır. Çıkartılan zincir koduna en yakın noktanın koordinatları haritada tespit edilir. Bulunan noktayla en uygun noktalar arasındaki uzaklıklar karşılarştırılarak taranan nesnenin ait olduğu sınıf tespit edilir.

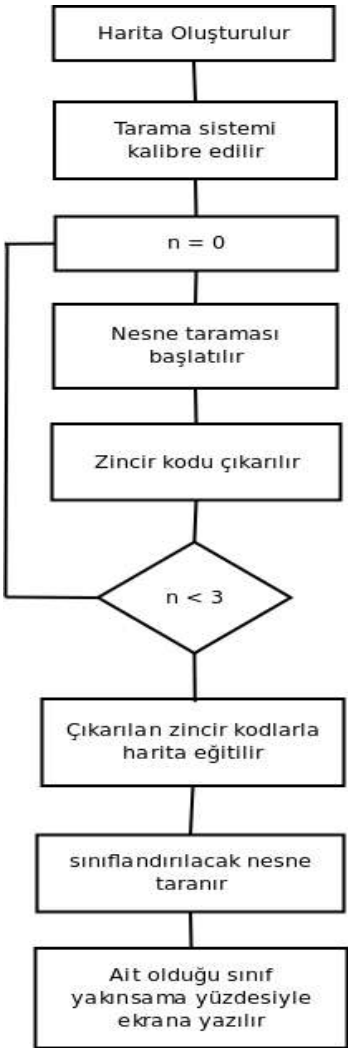
BÖLÜM BEŞ
NESNE TANIMA VE SINIFLANDIRMA SİSTEMİNİN İŞLEYİŞİ

Yapılan bu çalışmada, nesneleri taramak için gerekli donanım edinilmiş ve bu sistemin çalışması için iki C++ kütüphanesi tasarlanmıştır. Son olarak bu sistemin test edilmesi ve sonuç alınması için bir test programı tarafımdan yazılmıştır.

Bu bölümde test programının işleyişi ve sistemin nasıl çalıştığı adım adım anlatılacaktır. Tasarlanan iki kütüphanenin, sistemin işleyişinde kritik olan kısımları kod örnekleriyle birlikte açıklanacaktır.

5.1 Test Programının Genel İşleyişi

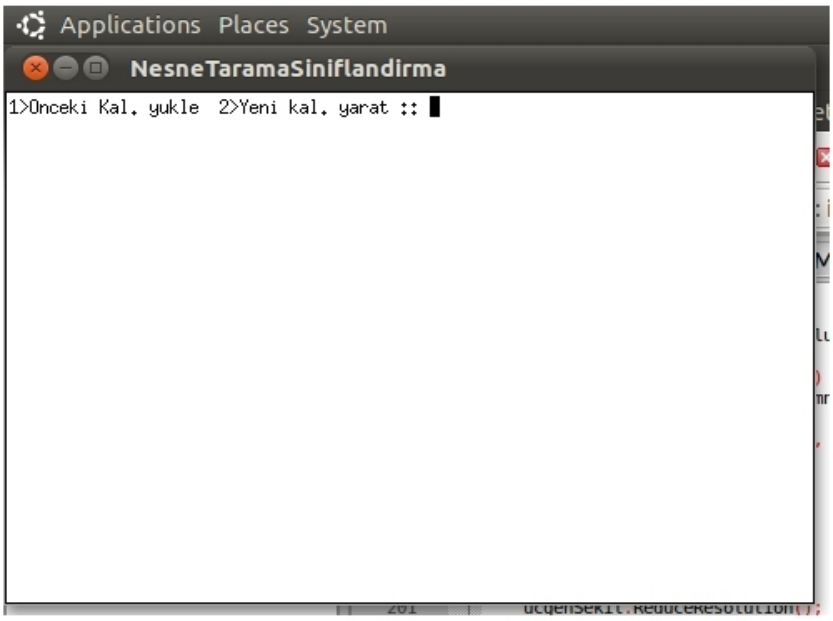
Test programının ilk aşaması tarama sisteminin kalibrasyonudur. Programda tek bir kalibrasyonu saklayan bir yapı mevcuttur. Böylelikle kamera ve çizgi lazer kaynağının konumları değiştirilmedikçe aynı kalibrasyon yüklenerek kullanılabilir. İlk olarak programda bir KOH yaratıp değerleri rastgele atanır. Daha sonra kullanıcıdan üç farklı nesnenin taranması istenir. Bu nesnelerin zincir kodları çıkartılıp haritadaki en uygun noktalar bulunur. Bu aşamadan sonra eğitim aşamasına geçilir. Eğitim aşamasında en uygun noktalar daha yoğun olmak üzere haritadaki noktalar üç nesnenin zincir kodları olan vektörlere yakınlaştırılır. Sistemin eğitiminin bittiği ekranda çıktı olarak belirtilir. Son olarak program kullanıcıdan sınıflandırılacak nesnenin taranmasını ister. Son nesne tarandıktan sonra, nesnenin zincir kodu çıkartılır ve haritada en yakın olduğu nokta bulunur. Bu noktanın eğitim aşamasında kullanılan nesnelerin orta noktalarına uzaklığı yüzde olarak saptanır ve karşılaştırılır. Nesnenin yakınsama yüzdesiyle birlikte benzediği nesne belirlenmiş, böylece ait olduğu sınıf tespit edilmiş olur. Sistemin genel işleyişinin diyagramı Şekil 5.1'de gösterilmiştir.



Şekil 5.1 Sistemin çalışma diyagramı.

5.1.1 Kalibrasyon

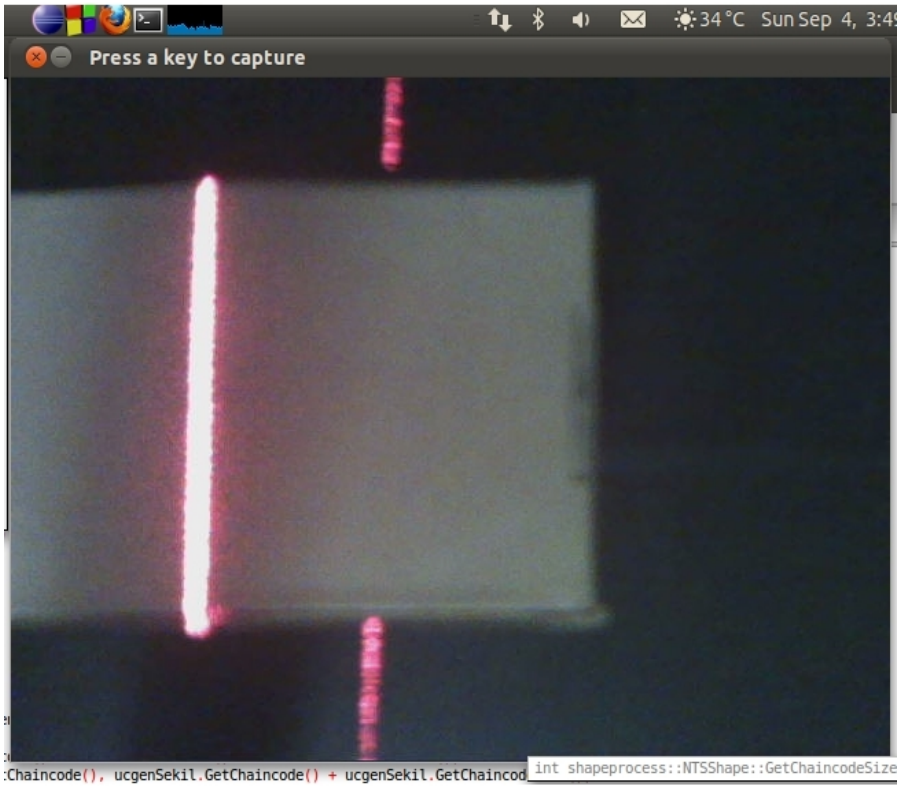
Kalibrasyon işlemi tarama sisteminin doğru çalışması için önemlidir. Tarama sisteminin pozisyonu tarama işlemine göre değişebileceği için sistem çalıştırıldığında ilk yapılan işlem kalibrasyondur. İlk gelen kalibrasyon ekranı Şekil 5.2'deki gibi bir menüdür. Menüde sistem kullanıcıya eski kalibrasyonu yükleme ve yeni kalibrasyon yapma seçeneklerini sunar.



Şekil 5.2 Kalibrasyon menüsündeki seçeneklerin ekran görüntüsü.

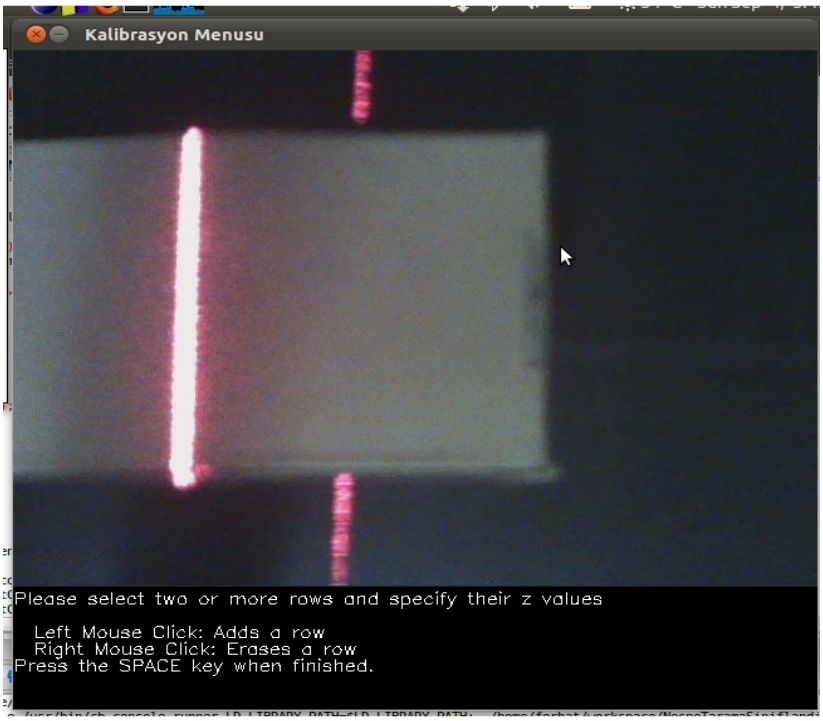
Kalibrasyon menüsündeki ilk seçenek, eğer tarama sisteminin konumu değiştirilmemiş ise, bir önceki ayarları yükler. Tarama sisteminin konumu farklı ise ikinci seçenek seçilip, sistem tekrar kalibre edilmelidir.

Kalibrasyon menüsünden, yeni kalibrasyon yaratmayı seçtiğimizde ekrana kamera görüntüsü gelir. Kameranın önüne, çizgi lazerin izdüşümüne denk gelecek şekilde, boyutları bilinen bir nesne yerleştirilir. Bu örnekte Şekil 5.3'te görüldüğü üzere, boyutları bilinen bir karton kutu yerleştirilmiştir. Kalibrasyon nesnesi yerleştirildikten sonra, herhangi bir tuşa basarak görüntünün fotoğrafı çekilir.



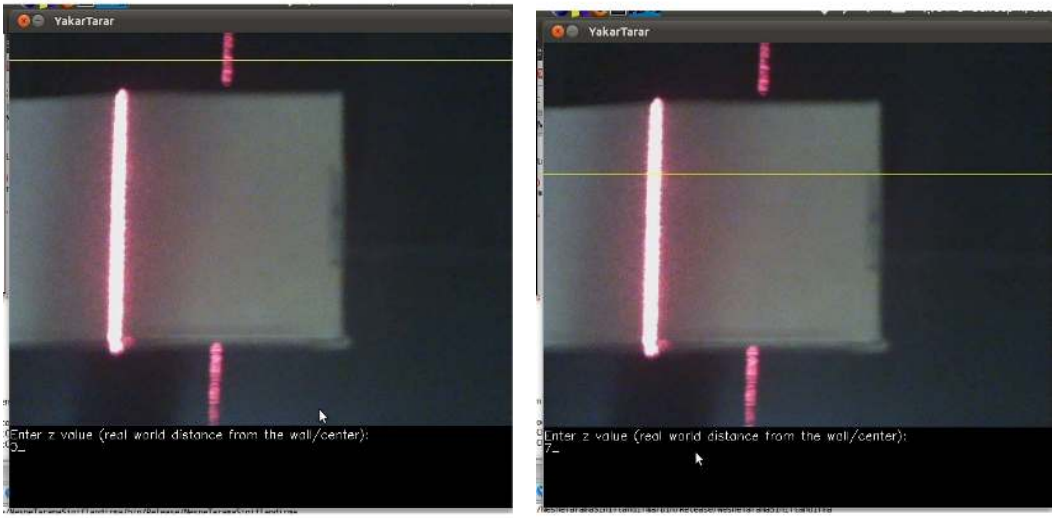
Şekil 5.3 Kalibrasyon nesnesi kameranın önüne yerleştirilir ve fotoğrafı çekilir.

Kalibrasyon nesnesinin görüntüsü alındıktan sonra, yüksekliklerin belirlenmesi gerekmektedir. Yükseklikleri belirlemek için lazer izdüşümünün farklı olduğu iki satırı seçmek yeterlidir. Satırları seçmek için, herhangi bir satırda sol fare tuşunu basmak yeterlidir. Sağ fare tuşuyla yanlış yapılmış bir seçim iptal edilebilinir. Kalibrasyon menüsünün bu kısmını Şekil 5.4'te görebiliriz.



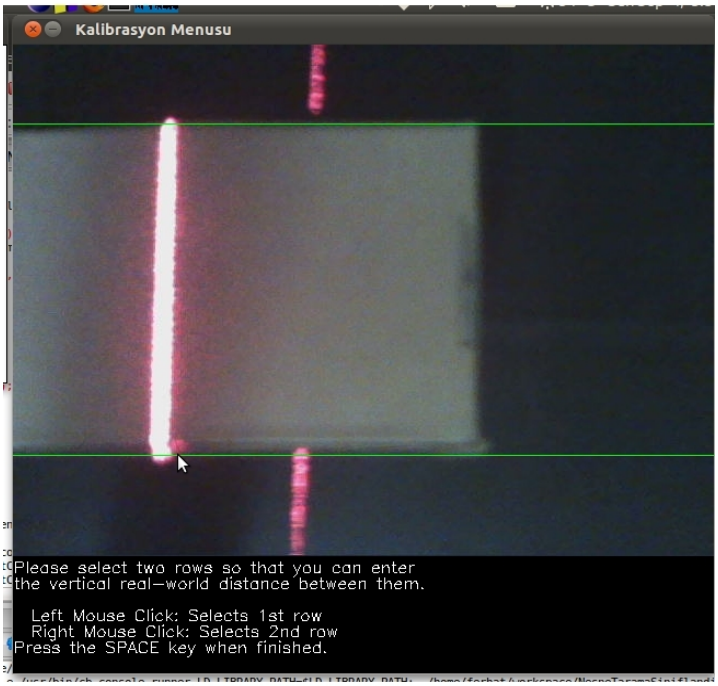
Şekil 5.4 Yükseklik kalibrasyon aşaması.

Yüksekliklerin belirlenmesinde yakartarar istenildiği kadar satır seçme konusunda kullanıcıyı özgür bırakmıştır. Bu örnekte iki satır seçmek yeterlidir. Siyah bantın üzerine düşen lazer izdüşümü '0' noktasındadır. Yani kayan bantın yüzeyi zemin olarak alınmaktadır. Karton kutunun üzerine düşen herhangi bir satır ise ikinci belirleme noktasıdır. Bu iki satırın seçimi Şekil 5.5'te daha net anlaşılmaktadır. Bu iki satır arasındaki piksel sayısı farkı ile girilen yükseklikler arasında bağlantı kurulur. Bu bağlantı kurulduktan sonra tarama sistemi artık çizgi lazer izdüşümünün konumundan yüksekliği algılayabilmektedir.



Şekil 5.5 Yüksekliğin farklı olduğu yerlerde değerlerin girilmesi.

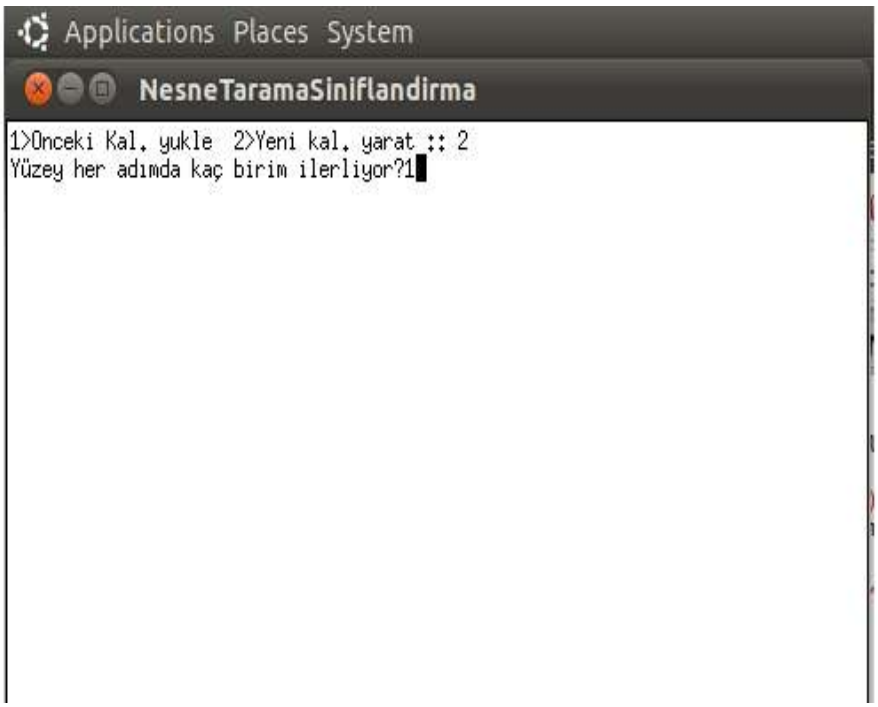
Yatay düzlem ile yükseklik arasındaki bağlantıyı kurduktan sonra, dikey düzlem ile gerçek dünya ölçüleri arasında da bağlantı kurmak gerekmektedir. Bunun için kalibrasyonda bir sonraki aşamaya geçilir. Şekil 5.6'da da görüldüğü gibi sistem kullanıcıdan farklı iki satırın seçilmesini istemektedir.



Şekil 5.6 Dikey düzlem ile gerçek dünya birimlerinin bağlantısının kurulması.

Dikey düzlemin ayarlanması yapmak için sol fare tuşuyla ilk satır, sağ fare tuşuyla ise ikinci satır seçilir. Seçim bittikten sonra iki yeşil çizgi arasında kalan gerçek mesafe girilir. Böylece sistem dikey düzlemde de pikseller ve uzunluk arasındaki bağlantıyı kurmuş olur.

Kalibrasyonun geriye kalan son aşaması, tarama hızının ayarlanmasıdır. Eğer bu aşama olmasaydı sistem tarama sonucunda, taranan nesnenin yatay düzlemdeki gerçek uzunluğunu bilemezdi. En son kalibrasyon ayarı olarak kullanıcının karşısına Şekil 5.7'deki ekran görüntüsü çıkmaktadır. Burada sistem kullanıcıya tarama esnasında alınan her görüntü için nesnenin kaç birim ilerlediğini sormaktadır. Girilen değer santimetre cinsindendir. Nesne tanıma ve sınıflandırma sisteminde bu ayar bantın hızına göre manuel olarak ayarlanmaktadır.

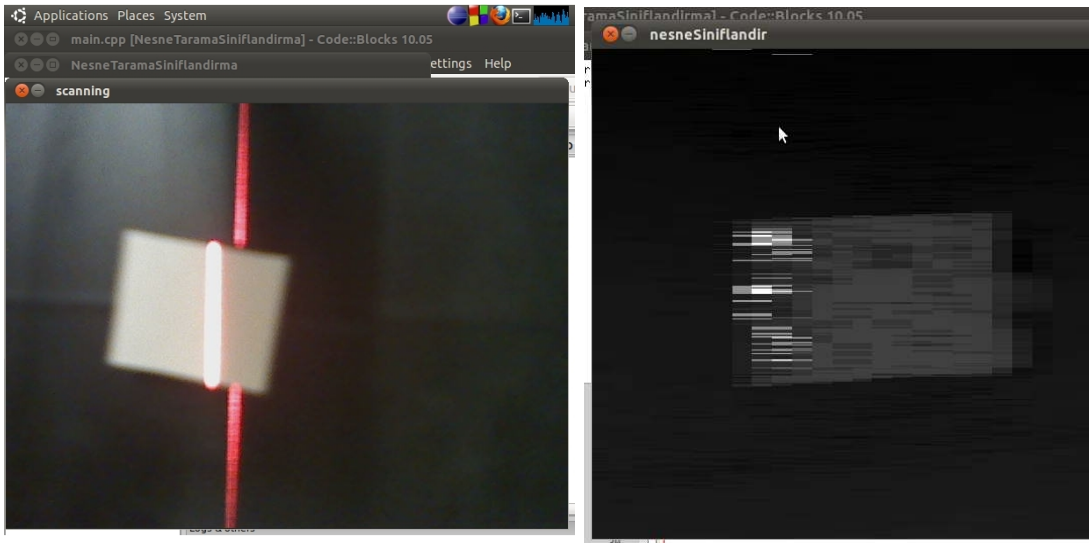


Şekil 5.7 Görüntü başına ilerleme mesafesinin girilmesi.

Bütün bu ayarlar yapıldıktan sonra kalibrasyon aşaması tamamlanmıştır ve kaydedilmiştir. Tarama sistemindeki kamera veya lazer kaynağının konumu değiştirilmediği sürece aynı ayarlar tekrar yüklenip kullanılabilinir.

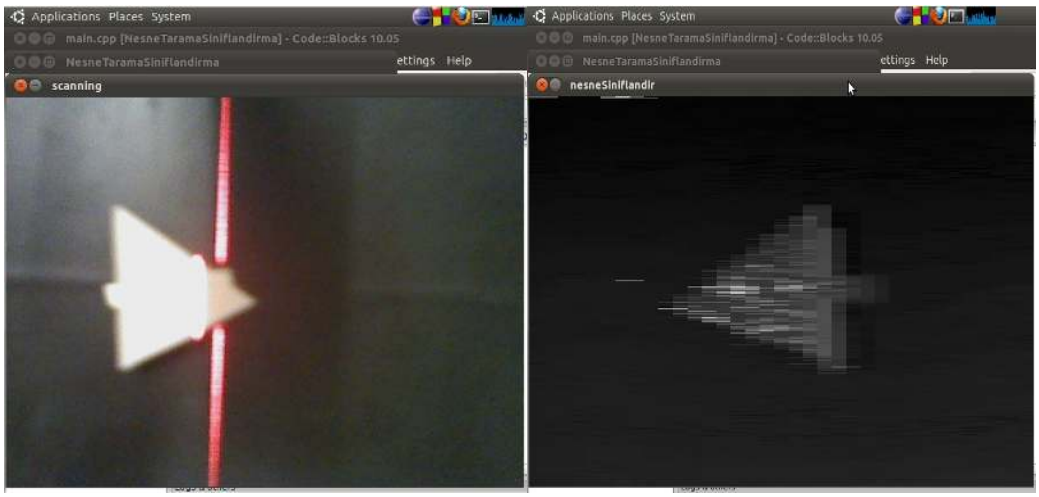
5.1.2 Nesnelerin Taranması ve Sınıflandırılması

Kalibrasyon aşamasından hemen sonra sistem ilk taramayı yapacak hale gelmektedir. Test programının üç farklı nesneyi KOH'un eğitimi için kullandığını söylemiştik. Dolayısıyla, ilk üç tarama bu üç eğitim nesnesine aittir. Yapılan testlerde kullanılan ilk nesne kareydi. Şekil 5.8'de kare şeklindeki nesneyi taranırken durumu ve tarandıktan sonra çıkan şekil görülmektedir.



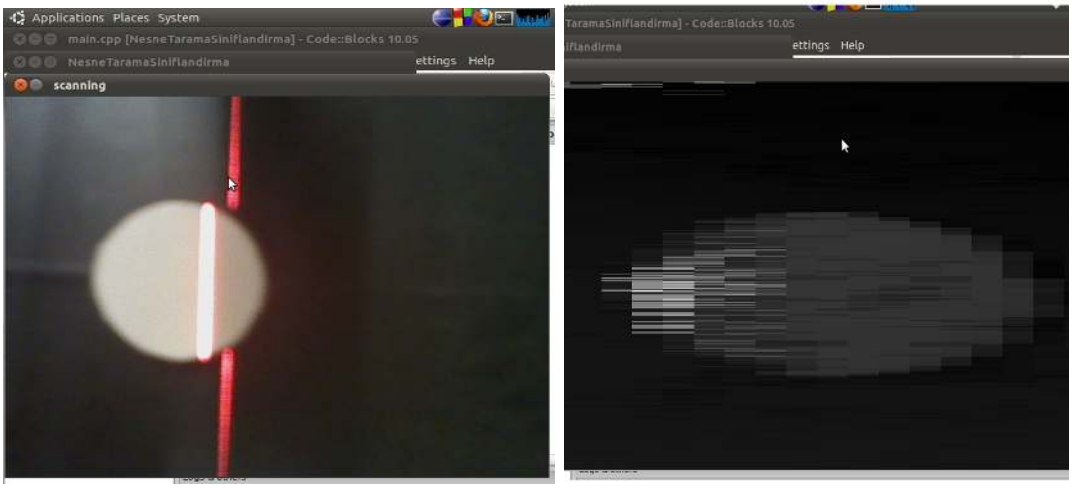
Şekil 5.8 Kare nesnenin taranması ve elde edilen sonucun resmi.

Tarama sonrası ekrana gelen, nesnenin görüntüsü, çalışmanın daha rahat anlaşılması ve tarama sonucunun gözlemlenebilmesi için programa konmuştur. İlk nesne tarandıktan sonra yükseklik bilgileri bir matriste tutulur. Bu matristeki bilgiler iki boyuta dönüştürülür. Böylelikle sistemin kaba şekli elde edilmiş olunur. İz düşümü şekli oluşan nesnenin zincir kodu çıkarılmaya hazırdır. Fakat program büyük veri kalıplarıyla uğraşmamak ve hız kazanmak için iki boyutlu resime bir işlem daha uygulamaktadır. Bu işlem de çözünürlüğün üç kat küçültülmesidir. Çözünürlüğü küçültülen resimden zincir kod çıkarılır. Çıkarılan zincir kodu haritadaki bilgilerle karşılaştırılır ve en yakın olan nokta o şeklin en uygun noktasını oluşturur. Şekil 5.9'da ikinci tarama olarak üçgen bir nesne, Şekil 5.10'da ise şekli daire olan üçüncü nesne görülmektedir.



Şekil 5.9 Üçgen nesnenin tarama anı ve tarama sonrası görüntüsü.

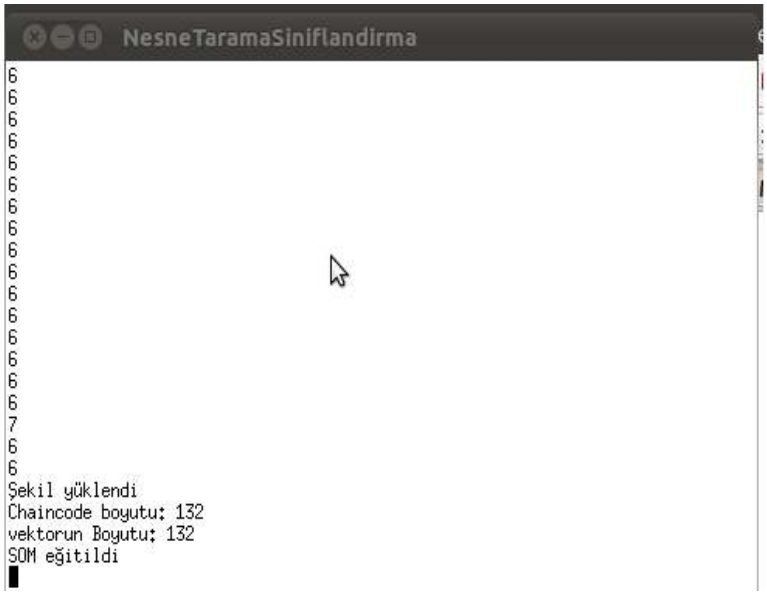
Zincir kodun çıkarılması ve haritadaki en uygun noktanın bulunması diğer iki şekil içinde gerçekleştirilir. Artık haritamızda üç farklı şeklin orta noktaları belirlenmiştir.



Şekil 5.10 Daire nesnenin tarama anı ve tarama sonrası görüntüsü.

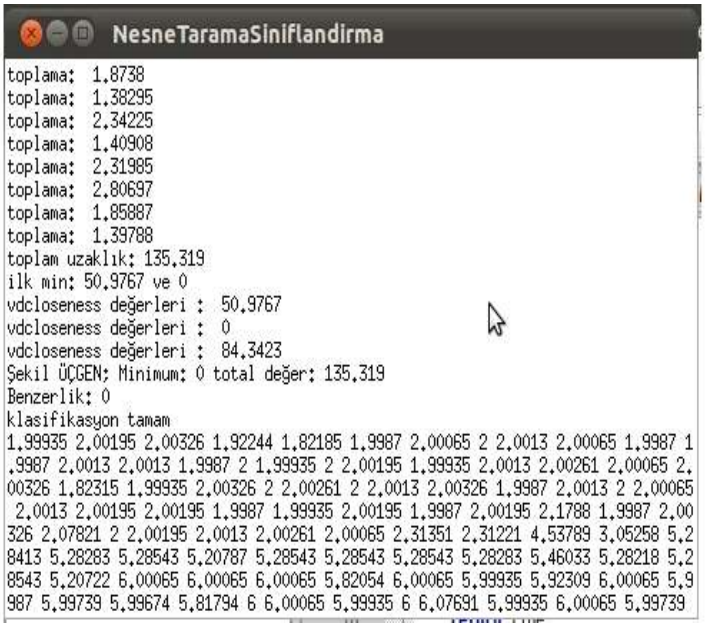
Haritada bütün uygun noktalar belirlendikten sonra artık KOH'un kendini eğitmesi aşamasına gelinmiştir. Bu aşamada sistem en uygun noktalardan (nesnelerin merkez noktalarından) başlayarak, haritadaki bütün noktaları, nesnelerin zincir kodlarından oluşan özellik vektörüne benzetmeye başlar. Bu benzetme işlemi orta noktada en yoğunken, uzaklaştıkça katsayıları azalmaktadır. Bu işlem defalarca tekrarlandığında, haritada her nesne kendi alanını belirlemiş olur. Bu bölge belirleme işleminin sonucunda eğitim aşaması tamamlanmıştır. Program eğitim aşamasının

tamamlandığını Şekil 5.11'de gösterilen ekran çıktısıyla kullanıcıya bildirir.



Şekil 5.11 KOH'un eğitiminin tamamlandığını gösteren ekran çıktısı.

Eğitim tamamlandıktan sonra sistem bu kez sınıflandırma işlemi için bir nesnenin taranmasını beklemektedir. Sınıflandırılacak nesne de tarandıktan sonra aynı işlemler bu nesne için de uygulanır. Sonuç olarak çıkarılan zincir kodu ile haritadaki noktalar karşılaştırılır. Haritada bulunan, sınıflandırılacak nesnenin zincir koduna en yakın nokta saptanır. Bulunan nokta hangi nesnenin orta noktasına daha yakınsa, sınıflandırılacak nesne o sınıfa aittir. Program sınıflandırılacak nesnenin haritadaki yeri ile diğer nesnelerin orta noktalarının uzaklıkları toplamından, nesnenin ait olduğu sınıfa ne kadar yakın olduğunu yüzde cinsinden hesaplar. Sonuç Şekil 5.12'deki ekran çıktısında görülmektedir. Benzerlik olarak geçen değer, nesnenin o sınıfa ne kadar yakın olduğudur. Değer ne kadar küçükse yakınlık o kadar fazladır.



Şekil 5.12 Sınıflandırmanın sonucu ve benzerlik değerinin ekran görüntüsü.

5.2 Sistemin İç Yapısı

Bu bölümde, çalışmada tasarlanmış olan iki kütüphanenin yaptığı işlemler, kodlarıyla birlikte anlatılacaktır.

5.2.1 NTSShape Kütüphanesi

NTSShape kütüphanesi nesne tanıma ve sınıflandırma sisteminde görüntü işleme metodlarını içerir. NTSShape kütüphanesi bir sınıftan oluşmaktadır. Bu sınıf aynı zamanda tarama sonucunu içinde barındıran değişkeni de üyesi olarak içerir. Sınıfın içerdiği değişkenlerin listesini Tablo 5.1'de görebilirsiniz.

Tablo 5.1 NTSShape objesinin içerdiği üyeler

int m_nColumns;	Nesne matrisinin kolon sayısı
int m_nRows;	Nesne matrisinin satır sayısı
double** m_adScannedShape;	Taranan Nesnenin yükseklik matrisi
int* m_anChainCode;	Zincir kodunu tutan dizi
int m_nChainCodeMemberCount;	Zincir kodu dizisinin eleman sayısı

Kütüphanede yapılan ilk işlem 3-boyutlu nesne bilgisinin 2-boyuta indirilmesidir. Bunun için 'ReduceToBinary' adlı metod kullanılmıştır. Metod nesnenin yüksekliğine

göre belirlenen bir eşik değerini parametre olarak alır. Tarama sonucunda parametrenin altında kalan bütün değerler '0', üstünde kalan değerler de '1' değerini alır. Böylece tarama sonucu matrisi 2-boyuta indirgenmiş olur. Bu yapılan işlemin kodları ve açıklamaları **T**ablo 5.2'de görülmektedir.

Tablo 5.2 Tarama sonucunu 3-boyuttan 2-boyuta indiren metod

```
void ReduceToBinary(double dTreshold)
{
    for (int nRowCount = 0; nRowCount < m_nRows; nRowCount++)
       for (int nColumnCount = 0; nColumnCount < m_nColumns; nColumnCount++)
          if (m_adScannedShape[nRowCount][nColumnCount] < dTreshold)
          {
             m_adScannedShape[nRowCount][nColumnCount] = 0;
          }
          else
          {
             m_adScannedShape[nRowCount][nColumnCount] = 1;
          }
}
```

Tarama matrisindeki her nokta taranır. Nokta eşik değerinin altındaysa, o nokta sıfıra eşitlenir Değilse '1' değerini alır.

Kütüphanede yapılan bir diğer işlem ise çözünürlüğü düşürmektir. Çözünürlüğü düşürmenin nedeni daha düşük bir veri kümesiyle işlemleri yapmaktır. Böylece sistem hız da kazanmış olur. Metod bütün tarama matrisini 3x3 matrisler halinde tarar. O bölgedeki '1'lerin sayısı dört ve üzerindeyse, yeni pikselin değeri '1' olur. Metodun kodları ve açıklamaları tablo 5.3'te gözlemlenebilir.

Tablo 5.3 Tarama sonucunun çözünürlüğünü düşüren kod parçası

```
for (int nRowCount = 0, nRowScan = 0; nRowScan < nReducedRow;
     nRowCount += 3, nRowScan++)
  for (int nColumnCount = 0, nColumnScan = 0;
       nColumnScan < nReducedColumn; nColumnCount += 3,
     nColumnScan++)
  {
     nCountOfOnes = 0;
     for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
        {
           if (m_adScannedShape[nRowCount + i][nColumnCount + j] == 1)
              nCountOfOnes++;
        }
        if (nCountOfOnes > 3)
          m_adScannedShape[nRowScan][nColumnScan] = 1;
        else
          m_adScannedShape[nRowScan][nColumnScan] = 0;
```

Tarama matrisi 3x3 matrisler halinde taranır.

Haritadaki '1'lerin sayısını tutan değer sıfırlanır.

3x3 matriste bulunan '1'ler sayılır.

Birlerin sayısı üç'ten fazla ise nokta '1' olur, değilse 0' olur.

NTSShape kütüphanesinin esas görevi, zincir kodunun çıkarılmasıdır. Zincir kodunu çıkarma işleminde iki metod önemli rol oynamaktadır. Zincir kodunu çıkarmanın ilk aşamasında 'AddToChainCode' metodu kullanılır. Bu metod noktanın yönünü alarak numerik değerleri atamada kullanılır. Metodun ayrıntılı açıklaması Tablo 5.4'te incelenebilir

Tablo 5.4 Zincir kodunu diziye ekleyen metod

void AddToChaincode(Direction eNextChain)
{
switch (eNextChain)
{
case DIRECTION_RIGHT:
m_anChainCode[m_nChainCodeMemberCount] = 0;
break;
case DIRECTION_DOWNRIGHT:
m_anChainCode[m_nChainCodeMemberCount] = 1;
break;
case DIRECTION_DOWN:
m_anChainCode[m_nChainCodeMemberCount] = 2;
break;
case DIRECTION_DOWNLEFT:
m_anChainCode[m_nChainCodeMemberCount] = 3;
break;
case DIRECTION_LEFT:
m_anChainCode[m_nChainCodeMemberCount] = 4;
break;
case DIRECTION_UPLEFT:
m_anChainCode[m_nChainCodeMemberCount] = 5;
break;
case DIRECTION_UP:
m_anChainCode[m_nChainCodeMemberCount] = 6;
break;
case DIRECTION_UPRIGHT:
m_anChainCode[m_nChainCodeMemberCount] = 7;
break;
}
}
	Zincir kodunu çıkaran metod.
	eNextChain yönleri içeren bir veri tipidir.
	
	Metoda gelen yön bilgisine göre zincir kodu diziye eklenir.

Tablo 5.5 Şekil bilgisinden zincir kodunu çıkaran metod

void BuildChaincode()
{
int nRowOfFirstSpot, nColumnOfFirstSpot, nRepeatFlag;
Direction ePreviousSpot;
m_anChainCode = new int [300];
m_nChainCodeMemberCount = 0;
 	
for (int nRowCount = 0; nRowCount < m_nRows; nRowCount++)
for (int nColumnCount = 0; nColumnCount < m_nColumns; nColumnCount++)
	
	Tarama sırasında ilk bulunan noktanın koordinatları
	ve bir önceki noktanın yön bilgisini tutan değişkenler.
	Objenin zincirkod bilgisini tutan dizi yaratılır ve
	eleman sayısı '0'a eşitlenir.
	
	Döngü bütün haritadaki taramayı başlatır.

Tablo 5.6 Zincir kod çıkarılmasında şekil kenarının bulunmasını gösteren kod parçası

if(m_adScannedShape[nRowCount][nColumnCount] == 1)
{
nRowOfFirstSpot = nRowCount;
nColumnOfFirstSpot = nColumnCount;
ePreviousSpot = DIRECTION_LEFT;
	Nokta '1' ise devam eder
	
	Bulunan noktanın satır
	ve kolon sayısını tutar.
	Önceki noktanın yönüdür.

Tablo 5.7 Şeklin sınırlarının taranıp, zincir kodunu çıkaran kod parçası

do
{
nRepeatFlag = 1;
 	
switch (ePreviousSpot + 1)
{
 	
case DIRECTION_DEFAULT:
nRepeatFlag = 0;
case DIRECTION_LEFT:
if (m_adScannedShape[nRowCount][nColumnCount - 1] == 1)
{
AddToChaincode(DIRECTION_LEFT);
m_nChainCodeMemberCount++;
nColumnCount--;
ePreviousSpot = DIRECTION_RIGHT;
nRepeatFlag = 0;
break;
}
case DIRECTION_UPLEFT:
if (m_adScannedShape[nRowCount - 1][nColumnCount - 1] == 1)
{
AddToChaincode(DIRECTION_UPLEFT);
m_nChainCodeMemberCount++;
nRowCount--;
nColumnCount--;
ePreviousSpot = DIRECTION_DOWNRIGHT;
nRepeatFlag = 0;
break;
}
.....
}
if (nRepeatFlag != 0)
ePreviousSpot = DIRECTION_DEFAULT;
} while (!(nRowOfFirstSpot == nRowCount && nColumnOfFirstSpot == nColumnCount));
	Zincir kodunu belirleyen döngü.
	
	Önceki noktanın bir sonrasından taramaya başlanır.
	Yönün 'DEFAULT' olması ilk tarama anlamına gelir.
	
	Sol yönden başlanır ve Nokta '1' ise devam edilir. Bu metod tablo 5.4'te açıklanmıştır.
	Zincir kodu eklenir, eleman sayısı artırılır, bir önceki yön işaretlenir ve tekrar etme değişkeni sıfırlanır.
	
	Yukarıdaki tarama işlemi 1' değerini bulana kadar her yön için yapılır.
	
	NRepeatFlag '0' değilse işlemde problem vardır.

5.2.2 NTSSOM Kütüphanesi

NTSSOM kütüphanesi yapay sinir ağları işlemlerinin yapıldığı kütüphanedir. Kütüphane üye değerler olarak haritayı ve boyutlarını içerir. Bunların yanısıra özellik vektörü bilgisini ve en uygun nokta koordinatlarıda üye değişkenler arasındadır.

Değişkenlerin listesi ve açıklamaları Tablo 5.8'de görülmektedir.

Tablo 5.8 NTSSOM sınıfına üye değişkenler ve açıklamaları

int m_nSOMColumns;	Haritanın kolon ve
int m_nSOMRows;	satır sayıları.
NTSNode** m_acSOM;	Haritanın başlangı adresi.
vector< vector<double> > m_vdFeatureVectors;	Özellik vektörleri.
vector<int> m_vnBMURowPoint;	En uygun noktaların
vector<int> m_vnBMUColumnPoint;	koordinatlları.
int m_nFeatureVectorNumber;	Özellik vektörü asayısı.

Kütüphanemizin ilk yaptığı işlem haritayı yaratıp rastgele değerler atamaktır. NTSSOM sınıfının oluşturucu metodunda eğer haritanın boyutları belirtilmemişse otomatik olarak '10 x 10' boyutlarında yaratılır. Harita yaratıldıktan sonra değerler rastgele atanır. Kütüphanenin oluşturucu metodu Tablo 5.9'de açıklamalarıyla birlikte verilmiştir.

Tablo 5.9 Oluşturucu metod

NTSSOM()	
{	
m_nSOMRows = 10;	Satır sayısı ve
m_nSOMColumns = 10;	kolon sayısı belirlenir.
m_acSOM = NULL;	
m_nFeatureVectorNumber = 0;	Özellik vektörü sayısı sıfırlanır.
CreateMap(m_nSOMRows, m_nSOMColumns);	Harita yaratılıp rastgele değerler
}	atanır.

Harita yaratıldıktan sonra, tarama işlemleri yapılır ve her tarama işleminden çıkarılan zincir kodu, yani özellik vektörleri, haritadaki nöronlarla karşılaştırılır. Özellik vektörlerine en çok benzeyen noktalar, en uygun noktalar olarak belirlenir ve koordinatları kaydedilir. Bütün en uygun noktalar belirlendikten sonra eğitim aşamasına geçilir.

Eğitim aşamasında en uygun noktalar birebir örnek aldıkları özellik vektörüne benzetilirler. Diğer KOH uygulamalarından farklı olarak bir sınıf için bütün haritadaki nöronlar eğitilir. Nöronlar eğitilirken yakınsama en uygun noktadan uzaklaştıkça ve iterasyon sayısı arttıkça ters orantılı olarak azalır. Bu formül genel prensip itibariyle diğer uygulamalardan esinlenerek ama yapı olarak farklı bir şekilde

tasarlanmıştır. Bu orantı şu şekilde kurulur:

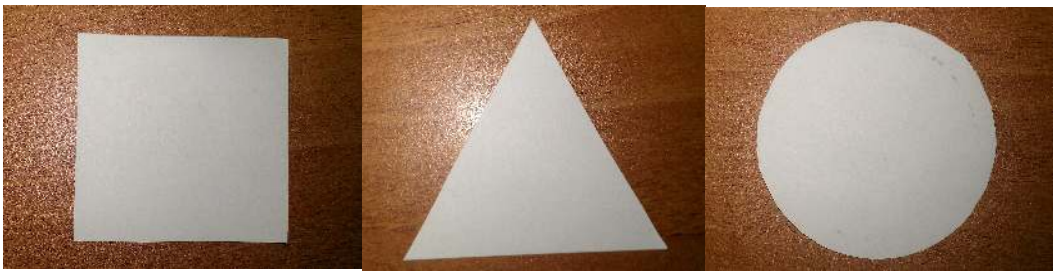
fark = özellik vektörünün elemanı – noktanın vektörünün elemanı
vektör elemanının yeni değeri = vektör elemanının eski değeri +
(fark / 1.5) x (1 / ((en uygun noktaya olan uzaklık + 1) x iterasyon sayısı))

Bu formül her noktanın her elemanı için uygulanarak eğitim aşaması tamamlanmış olur.

Sınıflandırma aşamasında eğitim aşamasındaki metodlar kullanılır. Sınıflandırılmak istenen nesne taranıp özellik vektörü bulunduktan sonra en uygun nokta haritada aranır. Bulunan nokta ile diğer en uygun noktalar arasındaki uzaklık hesaplanıp toplamlarının yüzdesi alınır. Böylece sınıflandırılacak nesnenin ait olduğu sınıf aitlik yüzdesiyle belirlenmiş olur.

BÖLÜM ALTI
YAPILAN DENEYLER

Yapılan çalışmanın sonucunu görmek için bir test programı tasarlanmış ve iki deney yapılmıştır. Programın yapısında ilk olarak tarama sisteminin kalibrasyonu vardır. Kalibrasyonun ardından yaratılan Kohonen haritasının eğitim aşaması gelir. Eğitim için 3 farklı nesne kullanılmıştır. Şekil 6.1'de de görüldüğü gibi eğitim nesneleri kare, üçgen ve dairedir. Nesneler kartondan kesilip, yine karton bir parça aracılığıyla yükseltilmiş ve tarama sistemi için 3 boyutlu bir nesne görüntüsü sağlanmıştır.



Şekil 6.1 Eğitimde kullanılan nesneler.

Tasarlanan deneme programında harita kare, üçgen ve daire olmak üzere üç farklı nesne sınıfıyla eğitilmiştir. Yapılan ilk deneyde her nesne sınıfına ait nesneler 10 kez taranmıştır. Deneyin istatistiksel sonuçlarına bakıldığında, her bir nesnenin tanınmasının ortalama başarı oranlarının %67,5 olduğunu görülmüştür. Genel izlenim olarak kare ve daire nesneler yaklaşık %70 başarı sağlarken, üçgen nesnelerde bu başarı %60 civarındadır. Bu denemelerin yaklaşık %30 kadar kısmı tamamen başarısız olarak sonuçlanmıştır.

Yapılan ikinci deneyde her sınıfa ait nesnelerin boyutlarında değişiklikler yapılmıştır. Her nesne sınıfından küçük, normal ve büyük boy nesneler beşer kez taranmış, son olarak o nesne sınıfına yakın fakat ait olmayan bir nesne 5 kez taranmıştır. Böylelikle her nesne sınıfı için toplan 20 deneme yapılmış, çıkarılan sonuçlarla bir kararlılık matrisi oluşturulmuştur. Ortaya çıkan kararlılık matrisini Tablo 6.1'de gözlemleyebiliriz.

Tablo 6.1 Kare, üçgen ve daire nesneler için oluşturulan kararlılık matrisi

Kararlılık Matrisi	Kare	Üçgen	Daire
Kare	70	10	20
Üçgen	30	60	10
Daire	5	10	85

Kararlılık matrisinde görüldüğü üzere benzerlerini tanıma konusunda sistemin en başarılı olduğu nesne dairedir. Bunun en büyük nedeni dairenin yatay düzlemde raotasyona uğramasının zincir kodunu değiştirmemesidir.

BÖLÜM YEDİ
SONUÇLAR VE GELECEK ÇALIŞMALAR

Yapılan çalışmada görüntü işleme ve yapay sinir ağları için iki kütüphane tasarlanmış ve bir deneme programı yazılmıştır. Sistemin denenmesi için lazer destekli 3-boyutlu nesne modelleme yöntemi kullanılmıştır. Tasarlanılan nesne tanıma ve sınıflandırma sistemi, tarama yöntemi bakımından bağımsız olup, Linux ve Windows tabanlı işletim sistemlerinde çalışan, nesne tanıma sistemlerine maliyeti düşük bir alternatif teşkil etmektedir.

Çalışmayı tarama, görüntü işleme ve sınıflandırma olarak üç bölüme ayıracak olursak, tarama kısmında kullanılan donanımın kalitesi, sonucu doğrudan etkilemektedir. Kullanılan kameranın lens kalitesi ve çözünürlüğü ne kadar iyi olursa tarama sonuçları o kadar hatasız olmakta, bu da sistemin hassasiyetini artırmaktadır.

Deneylerin sonucunda sistemin daha etkin çalışması için tasarında değişikliklere gidilmek gerektiği sonucuna varılmıştır. Yapılması gereken değişikliklere getirilen bir öneri, zincir kodu sisteminin değiştirilip geliştirilmesidir. Tezin önceki kısımlarında anlatıldığı üzere, zincir kod nesnenin sınırları taranarak çıkarılmış bir sayı dizisidir. Bu sayı dizisi parçalara bölünerek, elde edilen kısımların eğimlerinin hesaplanması planlanmıştır. Böylelikle sonuçların veriminin artacağı düşünülmektedir.

Bu çalışmanın amacı, endüstri alanında kullanılan örneklerine maliyeti mümkün olduğunca düşüren ve çok daha esnek bir alternatif sunmak, gelecek yapılacak akademik çalışmalar için bir taban oluşturmaktır.

KAYNAKÇA

Aksoy, S., Boyaci, H., ve Gokcay, D. (2008). The importance of context and semantic descriptions in object recognition: Studies in computer vision and human vision . (Kasım 21 2010), http://ieeexplore.ieee.org.

Asentics Vision Technology, (b.t.). (Ağustos 12 2011), http://www.asentics.de/.

Ayhan, E., Karslı, F., ve Tunç, E. (b.t.). Uzaktan Algılanmış Görüntülerde Sınıflandırma ve Analiz. (Kasım 21 2010), http://ocw.metu.edu.tr/mod/resource.

Camea Katalog, (b.t.). (Ağustos 12 2011), www.camea.cz.

Faaborg, A. J. (2002). Using Neural Networks to Create an Adaptive Character Recognition System . (Kasım 21 2010), http://citeseerx.ist.psu.edu.

Fardella, E., Fringuelli, B., Passeri, D., ve Rosi, L. (1997). A neural appraoach to robotic haptic recognition of 3-D objects based on a Kohonen self-organizing feature map. IEEE Transactions, 44(2), 267 - 269. 10 Kasım 2011, IEEE xplore database.

Frauel, Y., Javidi, B. (2001). Neural network for three-dimensional object recognition based on digital holography. Optics Letters, Vol. 26, Issue 19. (Kasım 21 2010), http://www.opticsinfobase.org .

Gonzalez, R. C., Woods, R. E., ve Eddins, S. L. (Ed.). (2008). Digital Image Processing An Algorithm, Prentice Hall.

Huang, T. S., Kohonen, T., ve Schroeder, M. R. (Ed.). (2001). Self-Organizing Maps (3rd ed). Springer.

Kohonen's Self Organizing Feature Maps (b.t.). (Mart 17 2010), http://www.ai-junkie.com/ann/som/som1.html.

Kootstra, G., Ypma, J., ve de Boer, B. (2007). Exploring objects for recognition in the real world. IEEE International Conference, 429 - 434. 10 Kasım 2011, IEEE xplore database.

Lathrop, R. R. (1997). Solder Paste Print Qualification Using Laser Triangulation. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, 20* (3), 174 – 182.

Morse, B. (2000). Tresholding. (Aralık 11 2010), http://morse.cs.byu.edu/~morse.

Nair, V., Hinton, G. E. (2008). 3D Object Recognition with Deep Belief Nets . (Aralık 11 2010), http://citeseerx.istipsu.edu.

Özkurt, N., Balbozan, F. İ. (2011). Lazerli Nesne Tanıma ve Sınıflandırma Sistemi. TOK 2011 Otomatik Kontrol Ulusal Toplantısı.

Pekel, I. Ö. (Şubat 2008). Yapay Sinir Ağları. ODTÜ Bilgisayar Topluluğu Elektronik Dergisi, (Kasım 10 2011), http://e-bergi.com/2008/Subat/Yapay-Sinir-Aglari.

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.

Yapay Sinir Ağları, (b.t.). (Kasım 4 2011), http://www.yapay-zeka.org.