T.C.
GEDİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAVALİMANLARINDA YOLCU BAGAJ ve KARGOLARININ BOYUT HACİM VE AĞIRLIK ÖLÇÜMLERİNİN RGB-D KAMERA ve YÜK HÜCRESİ (LOADCELL) KULLANILARAK TESPİTİ

TEVFİK AKKUŞ
YÜKSEK LİSANS TEZİ

MEKATRONİK MÜHENDİSLİĞİ

DANIŞMAN
Yrd. Doç. Dr. Savaş DİLİBAL

İSTANBUL - 2015

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün safhalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığı beyan ederim.

Tevfik AKKUŞ

TEŞEKKÜR

Tezimi hazırlarken yol gösteren, ilgilenen ve tez danışmanlığımı yapan Gedik Üniversitesi öğretim üyesi Sayın Yrd. Doç. Dr. Savaş DİLİBAL hocama, projemdeki görüntü işleme yazılımını gerçekleştirmede yardımcım olan Fırat Üniversitesi öğretim üyesi Doç. Dr. Abdülkadir ŞENGÜR hocama, bu tez projesinde kullandığım donanımları temin eden ve bana her türlü konuda destek olan SNS TEKNOLOJİ LTD. ŞTİ. sahiplerinden Faruk TÜTEN Bey'e, Pendik Teknik ve Endüstri Meslek Lisesi öğretmenlerinden Sinan ALEMDAR hocama, hacim ölçümü projesine ve lojistik otomasyonu alanına bizi yönlendiren SİSMAK MAKİNA LTD.ŞTİ. sahibi Murat ALBAYRAK Bey'e, yüksek lisans eğitimi için her türlü desteği veren güzide kurumumuz TÜBİTAK'a, tezimi hazırlarken ihmal ettiğim halde bana anlayış gösteren ve destek veren sevgili eşim Günay AKKUŞ'a, çocuklarım F.Betül ve B.Reyyan'a teşekkürü bir borç bilir, tezimi doğacak olan 3. çocuğuma armağan ederim.

Mayıs, 2015 **Tevfik AKKUŞ**

İÇİNDEKİLER

KISALTMALAR

RGB-D: Red Green Blue-Depth

API: Application Programmer Interface (Uygulama geliştirme arayüzü)

CCD: Charged Coupled Device

CMOS: Complimentary Metal Oxide Semiconductor

VMS: Volume Measurement System

WDM: Water Diplacemet Measurement

STM: Surface Tape Measurement

HSI: Hue Saturation Intensity

IR: Infrared

SL: Structured Light

ToF: Time of Flight

FOV: Field of View

PMD: Photonic Mixer Device

MS KINECT: Microsoft firmasının üretmiş olduğu KINECT markalı kamera.

FPS: Frames Per Second (Saniye başına görüntü karesi sayısı)

NUI: Natural User Interface (Kullanıcı arayüzü)

GUI: Graphic User Interface

SDK: Software Development Kit (Yazılım geliştirme kiti)

ROI: Region of Interest

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

HAVA LİMANLARINDA YOLCU BAGAJ VE KARGOLARININ BOYUT HACİM VE AĞIRLIK ÖLÇÜMLERİNİN RGB-D KAMERA VE YÜK HÜCRESİ (LOADCELL) KULLANILARAK TESPİTİ

Hazırlayan: Tevfik AKKUŞ, Mekatronik Mühendisliği

Danışman: Yrd. Doç. Dr. Savaş DİLİBAL

Son yirmi yıl içerisinde nesnelerin boyut ve hacimlerinin temassız olarak ölçülmesi giderek önem kazanmıştır. RGB-D sensörler, kameralar görüntü işleme yazılımları ve görüntü işleme API' leri sayesinde bu alanda çalışma yapan araştırmacı sayısı artmıştır.

Hava limanlarında hız ve kaliteli hizmet iki önemli parametredir. Hava limanına ilk girişte uçağa biniş öncesinde bagaj ve kargolarının boyut ve hacimlerinin ölçülmesi hali hazırda kullanılan teknikle oldukça fazla zaman almaktadır. Bu durum çoğu zaman gecikmelere veya yolcuların uçuşu kaçırmasına neden olmaktadır.

Bu çalışmada RGB-D kamera kullanılarak yolculara ait bagaj ve kargoların boyut ve hacim ölçümü amaçlanmıştır. Aynı zamanda ağırlıklarının ölçümü yük hücresi vasıtası ile sağlanmıştır. Daha sonra bütün bu ölçülen değerler hazırlanan MATLAB GUI ara yüzünde aynı anda gösterilmiştir.

Yukarıda bahsedilen bu yeni ölçüm tekniği ile hava limanına gelen yolcular bagaj ve kargolarını bir konveyör üzerine koyarak boyut hacim ve ağırlık bilgilerini kolayca görebilmektedir. Bagaj veya kargo üzerine eklenen barkod etiketi sayesinde bütün bu bilgiler, bagaj ve kargoya ait fotoğrafla birlikte veri tabanına kolayca kaydedilmektedir. Bu sayede kargo ve bagajların karışması veya kaybolması durumunda veri tabanına kayıtlı bu bilgiler kullanılabilecektir.

Anahtar Sözcükler: Görüntü işleme, Optik metroloji, RGB-D kamera, 3D ölçüm, MATLAB

Mayıs, 2015 **Tevfik AKKUŞ**

ABSTRACT

THE DETERMINATION OF THE MEASUREMENT OF DIMENSION, VOLUME AND WEIGHT OF PASSENGER LUGGAGES USING RGB-D CAMERA AND LOADCELL AT THE AIRPORTS

Tevfik AKKUŞ, Mechatronics Engineering

Supervisor: Asst. Prof. Dr. Savaş DİLİBAL

Contactless measurement of the size and volume of any object has been becoming significantly important in the last twenty years. The number of researchers who work on this field has been increasing thanks to the RGB-D sensors, cameras, image processing softwares and image processing APIs.

Speed and quality are two important parameters at the airports. During the check-in process, the measurement of size and volume of the luggage and cargo packages takes too much time in the present conventional technique due to the fact that this kind of situation sometimes causes delays or misses of flight for the passengers.

In this study, we aimed at obtaining 3D object image by means of RGB-D camera based image analysis techniques estimating the real size and volume of the object. The measurement of the weight of object was conducted using loadcell sensor. Additionally, the barcode label of the object was read using a barcod reader instrument. Later, all these data was demonstrated in a MATLAB GUI interface simultaneously.

Using above-mentioned new measurement technique, the passengers who started the check-in process will easily read the size, volume and weight of their hand luggage after putting them onto the conveyor. The attached barcode label will also give detailed information about the luggage. These data will be saved in the system database for further cases such as loses, breaks etc.

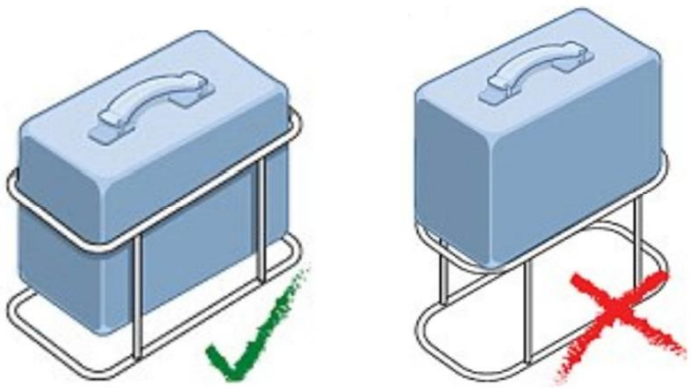
Keywords: Image processing, Optical metrology, RGB-D camera, 3D measurement, MATLAB

May, 2015 **Tevfik AKKUŞ**

1. GİRİŞ ve AMAÇ

Günümüzde, hava limanlarında, yolculara ait el bagajı ve kargoların uçağa kabulü öncesi, boyut, hacim ve ağırlık ölçümleri halen geleneksel metotlarla yapılmaktadır. Yolcu hava limanına geldiğinde, yanında getirdiği el bagajlarını ve diğer kargo paketlerini, görevli personele teslim ederken, personel, kabin içi bagajı olarak kabul edeceği valiz ve paketleri bir mekanik sistemin içine sokmakta, sonrasında ise tartmaktadır. Eğer yolcunun elindeki valiz veya paket bu mekanik sistemin içine sığarsa ve belirlenen limit ağırlığın altındaysa yolcuya kabin içi bagajı olarak verilmekte aksi takdirde uçak altındaki kargo bagajına kabul edilmektedir. Yolcunun yanında getirdiği diğer valiz ve paketler ise sadece kantara koyulup tartılmakta ve bu tartım sonucu çıkan ağırlığa göre yolcudan kargo ücreti talep edilmektedir. Hali hazırda kullanılan bu geleneksel sistem, çalışan personelin zamanını alan, yolcuları ise çok fazla bekleten bir sistemdir.

Şekil 1-1 Yolcu bagajı ve mevcut boyut ölçüm metotları



Şekil 1-2 Yolcu valizinin kabin içine alınmasına kabul / red

1.1. Tez / Proje Çalışmasının Amaç ve Hedefleri

Bu tez projemiz, hava limanlarında, yolculara ait bagajların ve kargoların temassız boyut, hacim ve ağırlıklarının ölçüm işini kapsamaktadır. Bu tez çalışması sonucu ortaya çıkarılan çözüm, hava limanları dışında, lojistik sektöründe ve birçok endüstriyel sektörde doğru ve hızlı olarak boyut, hacim ve ağırlık ölçümünde de kullanılabilir.

1.2. Bu Tez Projesinin Gerçeklemesi ile Sağlanacak Faydalar

Bu tezin gerçekleştirilmesi sonucunda aşağıdaki faydalar sağlanacaktır.

1. Hava limanlarında uçağa kabul öncesi, yolcuların bagaj ve kargolarının ölçümleri hızlı ve doğru bir şekilde yapılarak yolcuların bekleme süresini azaltılacak, uçuşu kaçırmaları engellenecektir.
2. Hava limanında çalışan personelin çalışmasını hızlandırılarak gereksiz vakit kayıplarının önüne geçilerek esas işlerine odaklanmasını sağlamış olunacaktır.
3. Tüm bu ölçümler bagaj ve kargoya ait fotoğraf ve barkod etiketi ile veri tabanına kaydedilecek ve ileride olabilecek kaybolma veya karışma durumlarında kullanılabilecektir.

2. GENEL BİLGİLER

Bu tez projemiz, hava limanlarında yolculara ait bagajların ve kargoların boyut, hacim ve ağırlıklarının temassız ölçüm işini kapsamaktadır. Bu kapsamda,

1. El bagajları ve kargo paketlerinin ebatlarının ölçümü için RGB-D kamera kullanılacaktır.
2. Ağırlık ölçümü için yük hücresi (loadcell) ve ağırlık göstergesi/RS232 çevirici kullanılacaktır.
3. Barkod bilgisini okumak için USB çıkışlı barkod okuyucu kullanılacaktır.
4. Yazılım olarak MATLAB 8.2 R2013b, Image Proccessing Toolbox ve Image Aquation Toolbox kullanılacaktır.

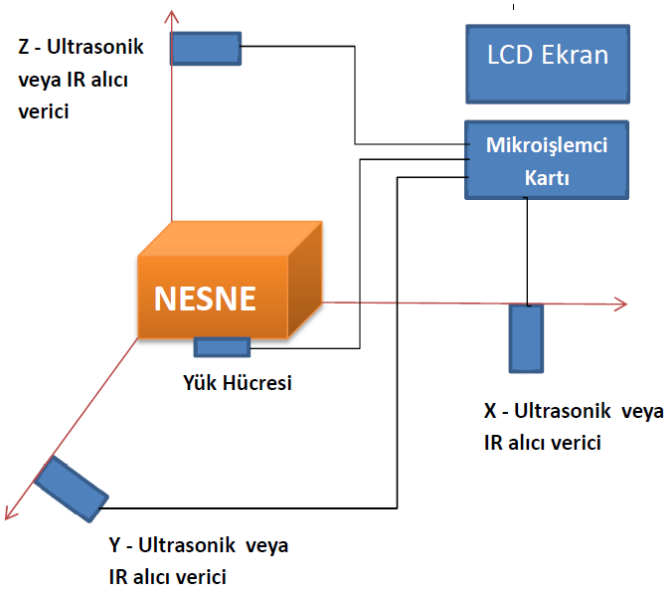
Image Aquation Toolbox yardımıyla kameradan alınan renkli görüntü ve derinlik görüntüsü, Image Proccessing Toolbox komutları ile işlenerek, görüntüler üzerinden en, boy, yükseklik ve hacim bilgileri hesaplanacaktır. Bagaj ve kargoya ait hacimsel ağırlık bilgisi (Hava limanlarında genelde hacim değerinin 5000'e bölünmesi ile elde edilir.) ile yük hücresinden elde edilen ağırlık bilgisi ve kargo paketi üzerindeki barkod bilgisi, hazırlanan MATLAB GUI arayüzünde gösterilecektir.

2.1. Temassız Boyut ve Hacim Ölçüm Teknikleri Hakkında

Standart kameralar x-y düzleminde 2D görüntü alır [43]. Bu çalışmada temassız olarak üç boyutlu ölçüm yapılacağından, bugüne kadar kullanılmış olan bütün 3D ölçüm teknikleri [1] açıklanacaktır.

2.1.1. Ultrasonik veya IR Alıcı-Verici Çiftleri ile Temassız Boyut Ölçümü

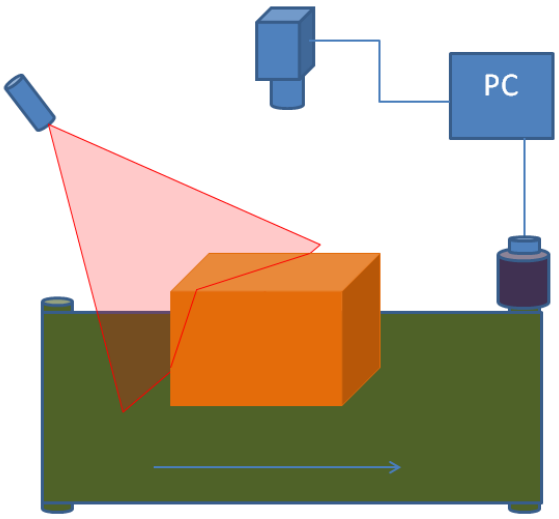
Bu metotta x,y ve z eksenlerine yerleştirilen 3 adet sensör mevcuttur. Sensörler ultrasonik verici alıcı çiftinden veya IR verici alıcı çiftinden oluşur. Sensör verici tarafı nesneye sinyal gönderir. Nesneden yansıyıp alıcı göze dönen sinyaller bir mikroişlemci tarafından değerlendirilir. Eğer ultrasonik alıcı verici sensör kullanılmışsa ultrasonik ses dalgasının nesneye çarpıp dönme süresinden mesafe hesaplanır. Eğer IR alıcı verici sensör kullanılmışsa ışığın nesneye çarpıp dönme süresinden mesafe hesaplanır. Ölçüm düzeneği Şekil 2-1'de gösterilmiştir.



Şekil 2-1 Ultrasonik veya IR sensörlerle 3D ölçümü

2.1.2. Çizgi Lazer Tekniği

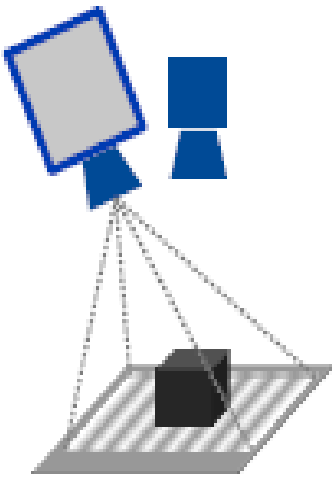
Bu teknikte, genellikle bir çizgi lazer kaynağı ve kalibre edilmiş bir kamera kullanılır. Bazen çizgi lazer ve kamera birlikte tek bir kutu içinde yerleştirilebilir. Sistemin çalışabilmesi için, 3D olarak ölçülmek istenen nesnenin kontrollü bir şekilde hareket etmesi gerekmektedir. Bir enkoder yardımıyla nesnenin ne kadar gittiği hesaplanır. Kamera sürekli fotoğraf çekerek lazer çizgisindeki profil değişikliğinden faydalanarak, görüntü işleme yazılımı vasıtasıyla nesnenin 3D profili çıkarılır.



Şekil 2-2 Çizgi lazer ve kamera ile 3D ölçümü

2.1.3. Desen Lazer Tekniği

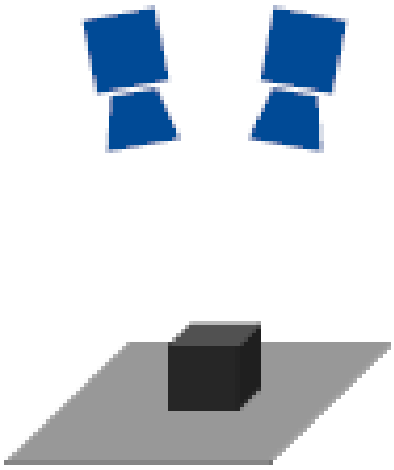
Bu teknikte, algılanacak 3D nesne üzerine bir projektör yardımıyla özel bir desen (genelde çizgiler vb.) düşürülür. Kamera bu çizgileri algılar. Çizgilerdeki değişim, görüntü işleme yazılımı ile işlenerek nesnenin 3D profili çıkarılır. Çizgi lazer tekniğinde nesne hareketli idi. Bu teknikte nesne sabit olup üzerine tek bir lazer çizgi yerine, projektör vasıtasıyla desen düşürülmektedir.



Şekil 2-3 Desen lazer kaynağı ve kamera ile 3D ölçümü

2.1.4. Pasif Stereo Görüntü Tekniği

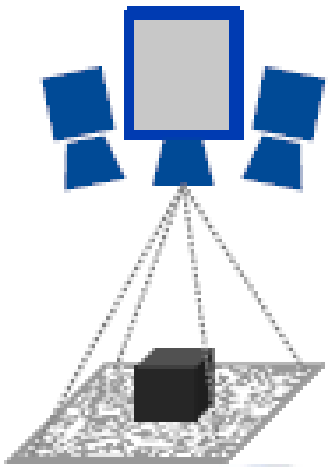
En yaygın olarak kullanılan 3D teknolojisidir. Bu teknikte sadece 2 kamera kullanılır. Kameraların arasındaki mesafe ve merkez doğrultu açıları net olarak bilindiğinden, her bir noktanın, her iki kameradaki izdüşümü, basit geometrik hesaplamalar ile bulunabilmektedir. Her iki kameradan alınan görüntüler işlenerek nesneye ait 3D bilgisi elde edilir. Işık değişimleri ve nesne zemin benzeşmeleri durumunda hatalı ölçüm yapmaya açık bir tekniktir.



Şekil 2-4 Stereo görüntü tekniği ile 3D ölçümü

2.1.5. Aktif Stereo Görüntü Tekniği

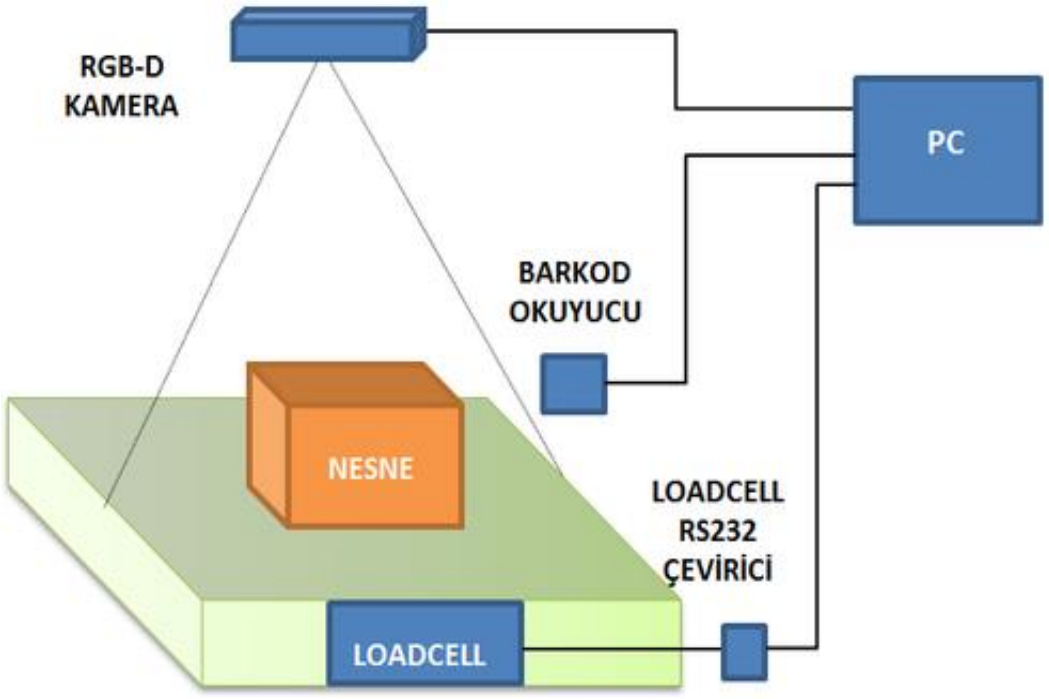
Bu teknikte, pasif stereo görüntü tekniği biraz geliştirilerek araya bir projektör eklenmiştir. Projektör nesne üzerine görüntü işleme yazılımı tarafından bilinen bir desen gönderir. Her iki kamera tarafından alınan görüntüler görüntü işleme yazılımı ile işlenir. Gönderilen desenin cisim üzerindeki değişimlerinden faydalanarak nesneye ait 3D bilgisi çıkarırılır.



Şekil 2-5 Aktif stereo görüntü tekniği ile 3D ölçümü

2.1.6. RGB-D Kamera Tekniği

Bu teknikte tek bir RGB-D kamera 3D ölçümü yapılacak nesnenin üzerine yerleştirilir. Önceki tekniklerdeki gibi ölçüm sırasında nesnenin veya kameranın hareket ettirilmesine gerek olmadığı gibi nesnenin üzerine lazer ışık kaynağı veya projektör yerleştirilerek nesne üzerinde çizgi veya desen oluşturmaya da gerek yoktur. RGB-D kamera, 3D ölçümü yapılacak olan nesnenin üzerine gözümüzle göremediğimiz yapılandırılmış IR ışıkları gönderir. Nesneye çarpıp RGB-D kameraya dönen IR ışıkları RGB-D kamera içerisinde bulunan renkli (RGB) ve derinlik (depth) kameraları tarafından algılanır ve bu kameralar tarafından algılanan görüntülerden nesneye ait en, boy ve yükseklik bilgileri görüntü işleme yazılımları vasıtası ile elde edilir. Diğer ölçme yöntemlerinden üstünlüğü sebebiyle bu tez çalışmasında bu teknik kullanılmıştır. Literatür araştırmasında RGB-D kamera ve yük hücresi birlikte kullanılarak yapılmış bir çalışmaya rastlanmamıştır.



Şekil 2-6 RGB-D kamera tekniği ile 3D ölçümü

Açıklanan tekniklerin avantaj ve dezavantajları Tablo 2-1'de verilmiştir.

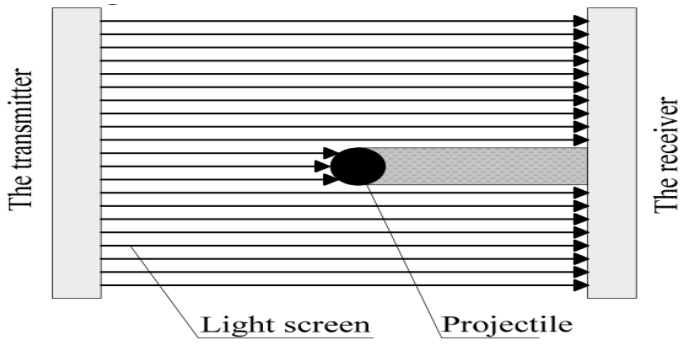
Tablo 2-1 Kamera ile 3D ölçüm metotlarının karşılaştırılması

S.No	3D Ölçüm Teknikleri	Kulllanılan Tekniğin Avantaj ve Dezavantajları
1	Ultrasonik veya IR verici alıcı sensörler tekniği	Ölçümü yapılacak nesnenin köşeye dayandırılması zorunludur. Hareketli cisimlerin ölçümü yapılamamaktadır.
2	Çizgi lazer tekniği	Çizgi lazer kaynağı gereklidir. Tarama için nesne veya lazer kaynağı hareket ettirilmelidir. Hareketsiz nesnelerin ölçümü yapılamamaktadır.
3	Desen lazer tekniği	Kameranın yanında lazer desen üreten projektör kullanılması gereklidir.
4	Pasif stereo görüntü tekniği	İki adet kamera kullanmak gereklidir. İyi bir ölçüm için nesne üzerinde desen olmalıdır.
5	Aktif stereo görüntü tekniği	İki adet kamera ve desen üreten bir adet projektör gereklidir.
6	RGB-D kamera tekniği	Tümleşik yapıdaki bir adet RGB-D kamera ile hem hareketli hem de sabit duran nesnelerin ölçümü hızlı ve doğru bir şekilde yapılabilmektedir.

2.2. Literatür Araştırması

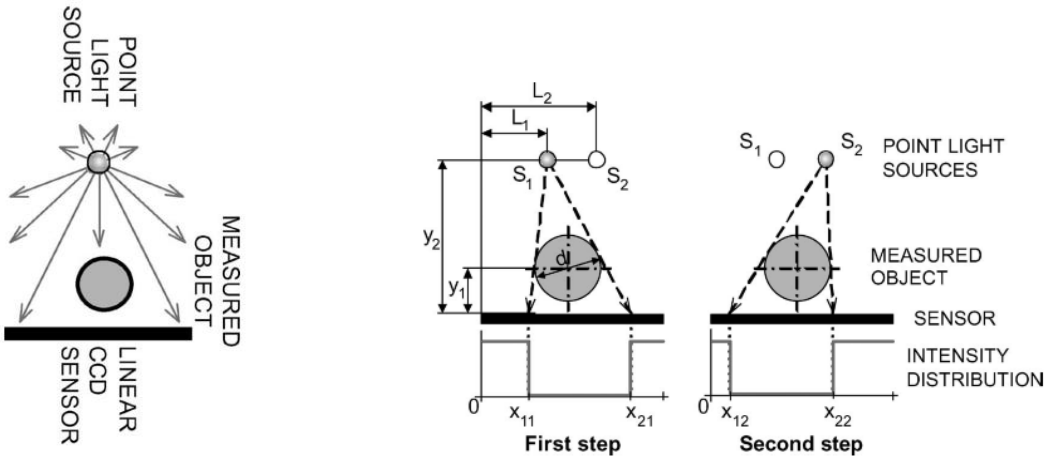
Kamera ve görüntü işleme yöntemleri kullanılarak temassız boyut ve hacim ölçümü ile ilgili bugüne kadar yapılan çalışmalar incelenerek bu bölümde detaylı bir şekilde sunulmuştur.

Qian ve Jinping [39] Şekil 2-7'deki düzenek ile hareketli nesnelerin boyutlarını ölçmeyi başarmışlardır.



Şekil 2-7 VMS düzeneği (Qian ve Jinping, 2011)

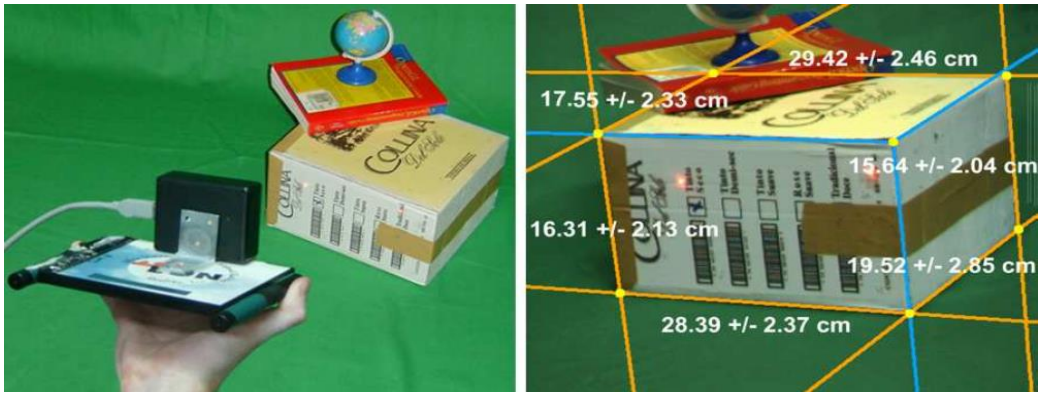
Radil ve ark. "Dimension Measurement of Objects With Circular Cross Section Using Point Light Sources and An Image Sensor Without Lens" adlı çalışmalarında bir ışık kaynağı ile CCD kamera arasındaki nesnenin boyutlarını, lens kullanmadan, yüksek doğrulukta ölçmüşlerdir [2].



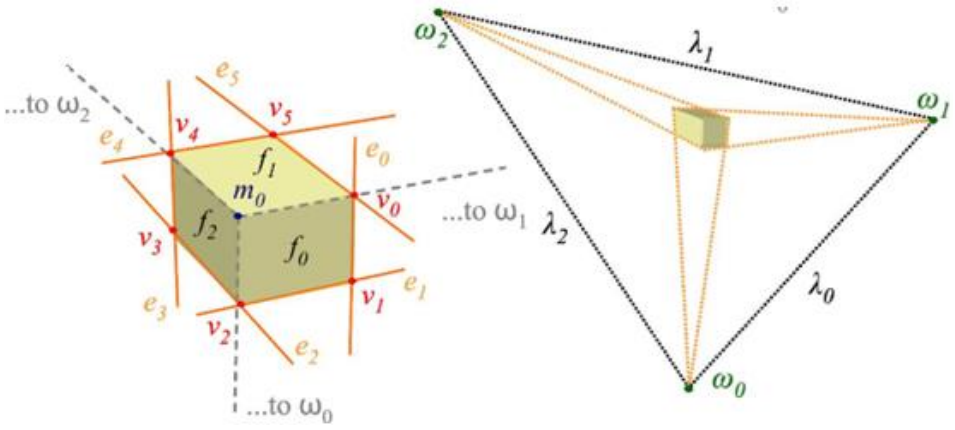
Şekil 2-8 VMS düzeneği (Radil ve ark., 2007)

Jelinek ve Taylor "Reconstruction of Linearly Parameterized Models From Single Images With a Camera of Unknown Focal Length" adlı çalışmalarında dikdörtgen yapılı nesnelerin perspektifinden aldıkları görüntülerden kaybolma noktası (vanishing point) tekniği ile nesnenin 3D ölçülerini hesaplamayı başarmışlardır [3].

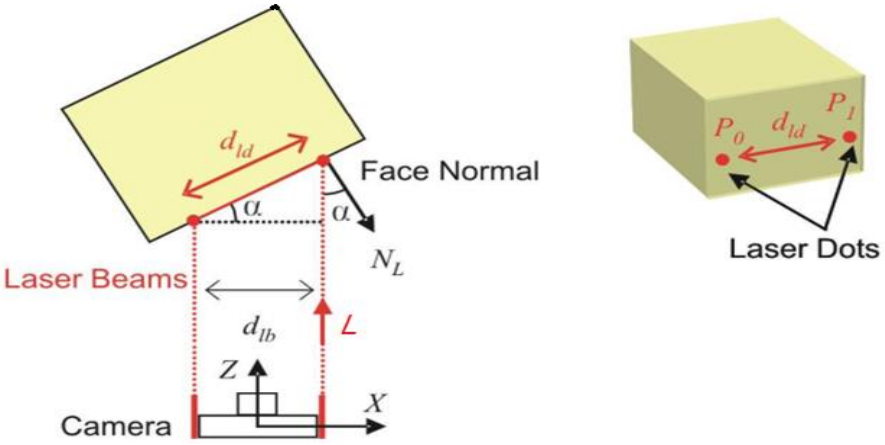
Fernandes ve ark. " Computing Box Dimensions from Single Perspective Images in Real Time" adlı çalışmalarında aralarındaki mesafe sabit olan iki adet nokta lazer kaynağı ve 1 adet kamera ile Şekil 2-9'daki düzeneği kurmuş ve dikdörtgen yapılı nesnelerin perspektifinden aldıkları görüntülerden kaybolma noktası (vanishing point) tekniği ile nesnenin 3D ölçülerini hesaplamayı başarmışlardır [4].



Şekil 2-9 VMS düzeneği (Fernandes ve ark., 2006)

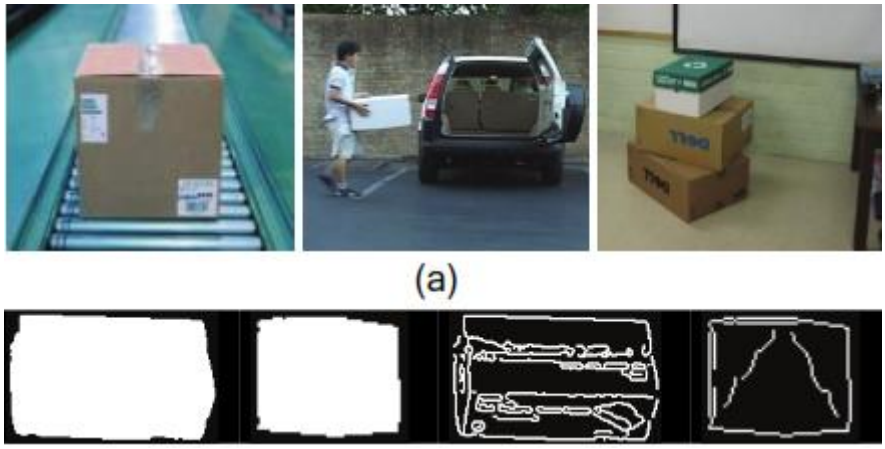


Şekil 2-10 Kaybolma noktaları (Vanishing points) gösterimi



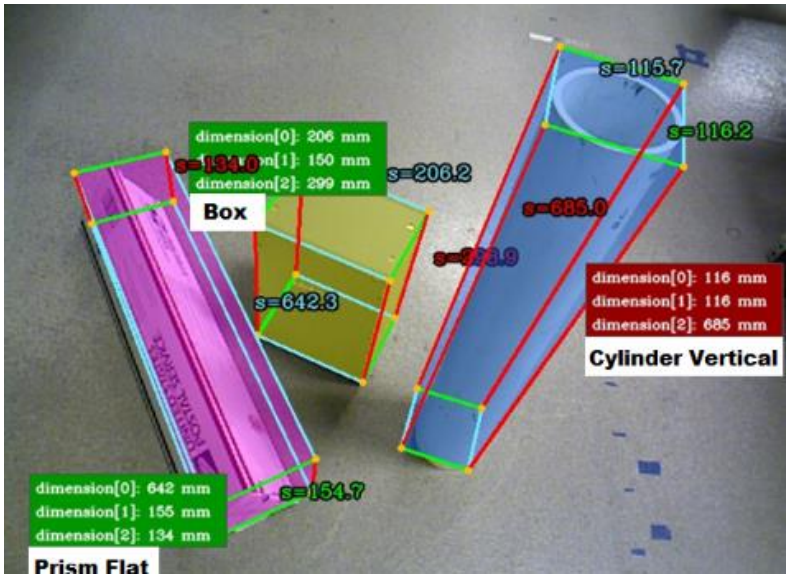
Şekil 2-11 Ölçüm düzeneği üstten görünüşü ve noktaların yerleri

Chen ve Aggarwal "Recognition of Box-like Objects by Fusing Cues of Shape and Edges" adlı çalışmalarında kutu ve kolilerin tanınması ve tasnif edilmesi ile ilgili bir çalışma yapmışlardır [5].



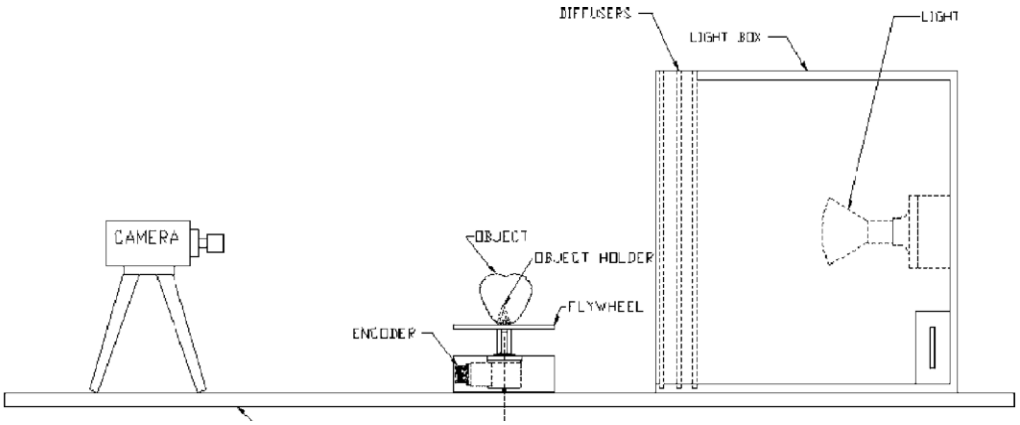
Şekil 2-12 Kutu tanıma gösterimi (Chen ve Aggarwal, 2008)

Lloyd ve McCloskey "Recognition of 3D Package Shapes for Single Camera Metrology" adlı çalışmalarında Microsoft Kinect kamera kullanarak, geometrik şekli düzgün nesnelerin paketlenebilir boyutlarını ölçmeyi başarmışlardır [6].



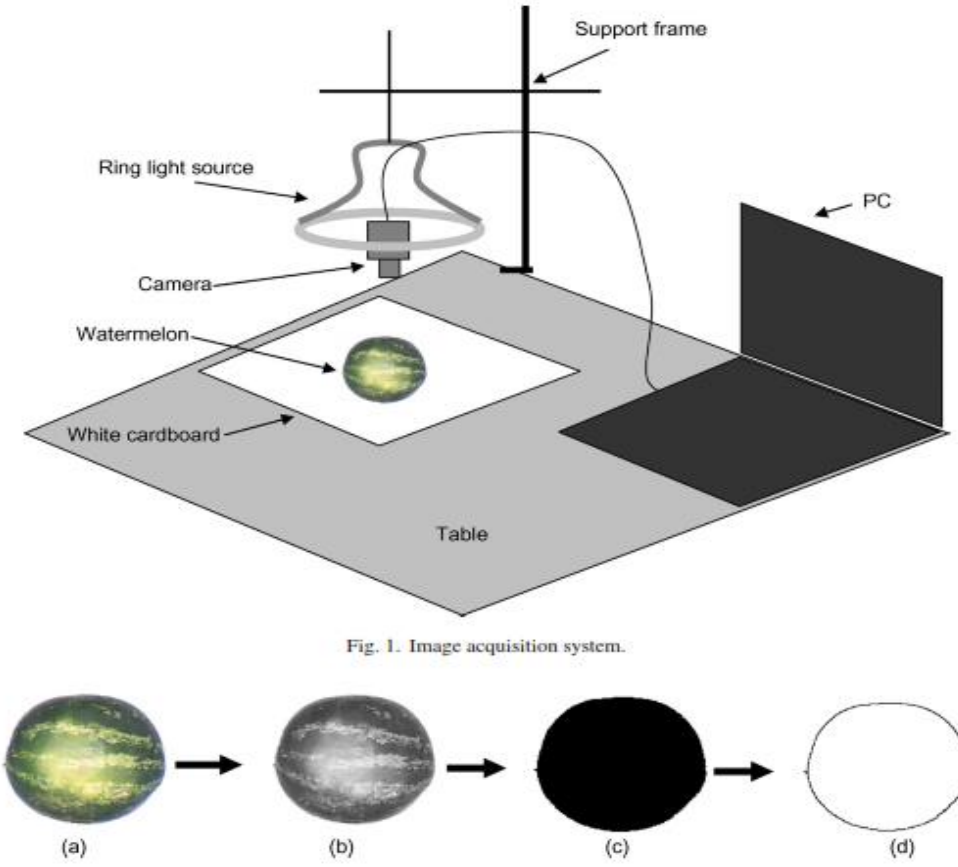
Şekil 2-13 Paketlenebilir 3D ölçüm sonuçları (Lloyd ve McCloskey, 2014)

Lee ve ark. "Area and Volume Measurements of Objects with Irregular Shapes Using Multiple Silhouettes" adlı çalışmalarında bir döner tabla üzerine yerleştirilen meyveyi motorla çevirmiş, o sırada meyvenin bir tarafını ışık kaynağı ile aydınlatılmış tam aksi tarafına ise CCD kamera yerleştirilerek cismin kameraya yansıyan gölgesinden alan ve hacim ölçümü yapmışlardır [7].



Şekil 2-14 VMS düzeneği (Lee ve ark., 2006)

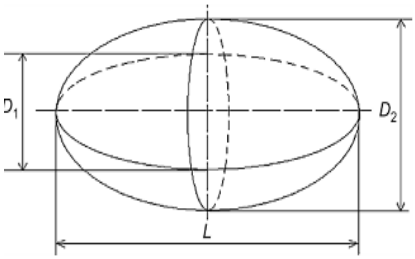
Koç "Determination of watermelon volume using ellipsoid approximation and image processing" adlı çalışmasında Şekil 2-15'teki düzeneği oluşturarak tek bir CMOS kamera altında beyaz zemin üzerine konulan karpuzun yüzey alanı ve hacmini yüksek doğrulukta ölçmeyi başarabilmiştir [8].



Şekil 2-15 VMS düzeneği (Koç, 2007)

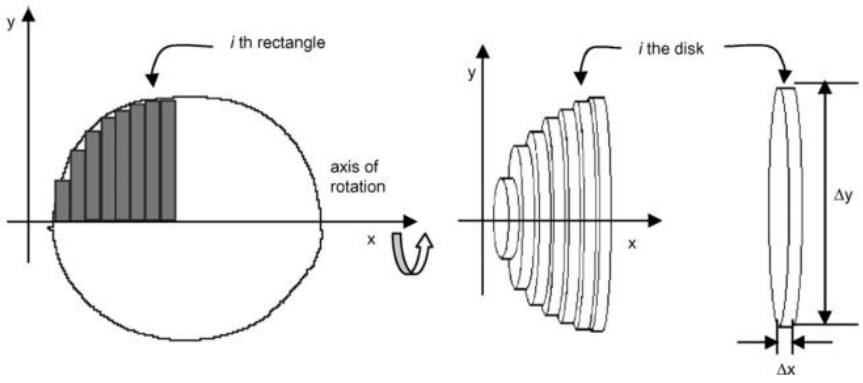
İşlemler aşağıdaki sıraya göre yapılmıştır.

1. Karpuz, su dolu bir kaba batırılmış ve taşan suyun hacmi hesaplanarak gerçek karpuz hacmi bulunmuştur (Arşimet su taşma metodu).
2. Daha sonra karpuz hacmi önce elipsoid hacim hesaplama formülü ile (2-1)'deki gibi hesaplanmıştır.



$$V_{\text{ellipsoid}} = \frac{4}{3}\pi\frac{L}{2}\frac{D_1}{2}\frac{D_2}{2} = \pi\frac{LD_1D_2}{6} \qquad (2\text{-}1)$$

3. CMOS kamera altına, beyaz zemin üzerine konulan karpuzun görüntüsü alınmıştır. Labwiev programı ve görüntü işleme teknikleri kullanılarak karpuzun kenarları belirlenmiş bu belirlenen kenarlardan yola çıkılarak karpuz görüntüsü 1 piksel kalınlığında dilimlere ayrılmıştır.

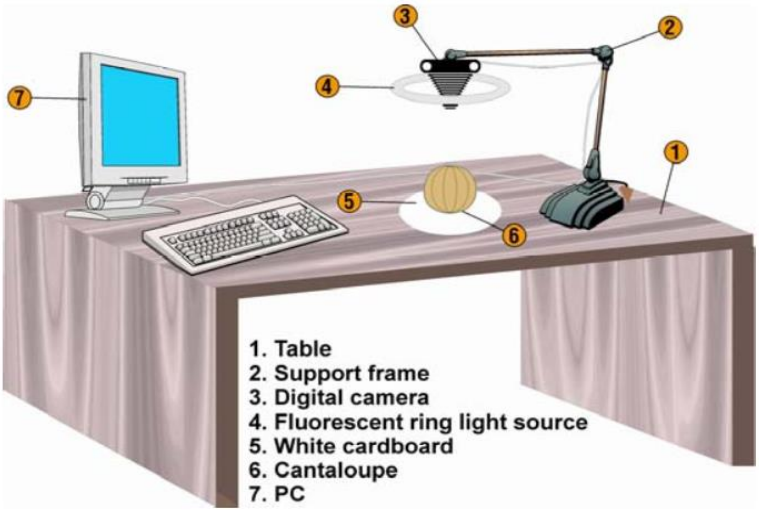


Şekil 2-16 Karpuz hacim hesaplaması için 1 piksellik dilimlere ayrılması

(2-2)'deki formüller ile önce her bir dilimin yüzey alanı $\mathbf{A_i}$ hesaplanmış ve kalınlık Δ_x ile çarpılarak $\mathbf{V_i}$ dilim hacmi bulunmuş ve en sonunda dilim hacimleri toplanarak karpuz hacmi $\mathbf{V_t}$ değeri yüksek doğrulukta hesaplanmıştır.

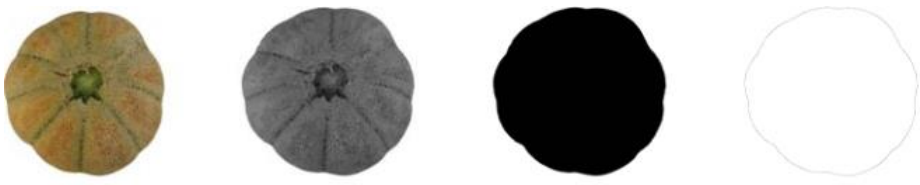
$$A_i = \pi\left(\frac{\Delta y}{2}\right)^2 \quad , \quad V_i = A\Delta x \quad , \quad V_t = \sum_{i=1}^{n} V_i \qquad (2\text{-}2)$$

Rashidi ve Ark. "Cantaloupe Volume Determination through Image Processing" adlı çalışmalarında Şekil 2-17' daki düzeneği kurmuş ve kantalop kavunu hacmini yüksek doğrulukta hesaplamayı başarmışlardır [9].



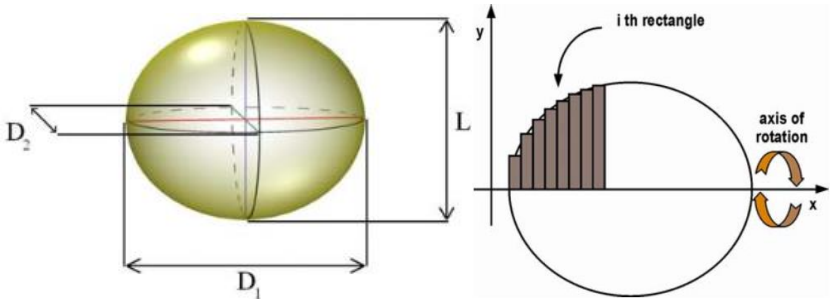
Şekil 2-17 VMS düzeneği (Rashidi ve ark., 2009)

Kavunun tam üzerine yerleştirilen tek kamera ile kavuna üstten bakıldığından önce kavunun mevcut pozisyonunda renkli görüntüsü alınmış sonra kavun kendi etrafında 90^{o} çevrilmiş ve tekrar renkli görüntüsü alınmıştır. Görüntü işleme teknikleri kullanılarak Şekil 2-18' de görüldüğü gibi kavuna ait renkli görüntülerden sırasıyla gri seviyeli görüntü, siuah-beyaz görüntü ve kanar görüntüsü elde edilmiştir. Bu kenar görüntüsülerinden alan hesabı yapılmıştır [9].



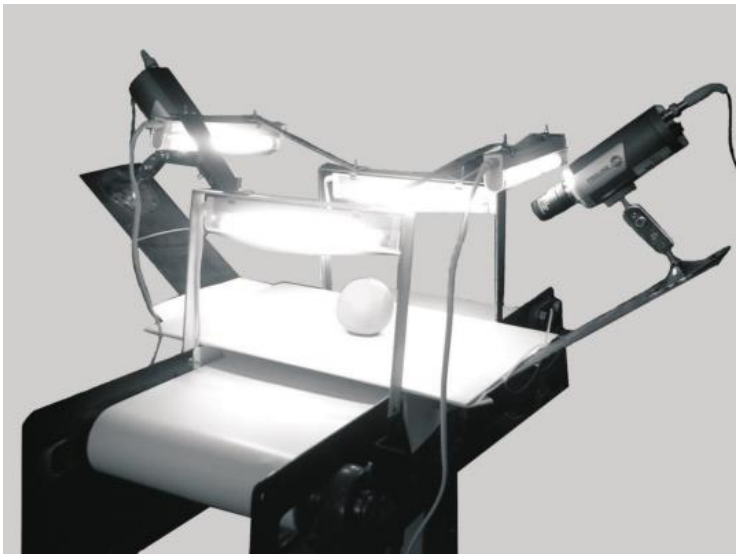
Şekil 2-18 Kantalop kavununa ait işlenmiş görüntüler

Şekil 2-19' da görüldüğü gibi kavun 1 piksellik dilimlere ayrılmış ve çıkan dilim sayısından hacim hesaplanmıştır.



Şekil 2-19 Kavun hacim hesaplaması için 1 piksellik dilimlere ayrılması

Khojastehnazhand ve ark. "Determination of Orange Volume and Surface Area Using Image Processing Technique" adlı çalışmalarında 510x492 piksel çözünürlükte 2 adet CCD sensörlü renkli kamera ve uygun ışıklandırmayı kullanarak, portakalın yüzey alanını ve hacmini yüksek doğrulukta ölçmeyi başarmışlardır [10].



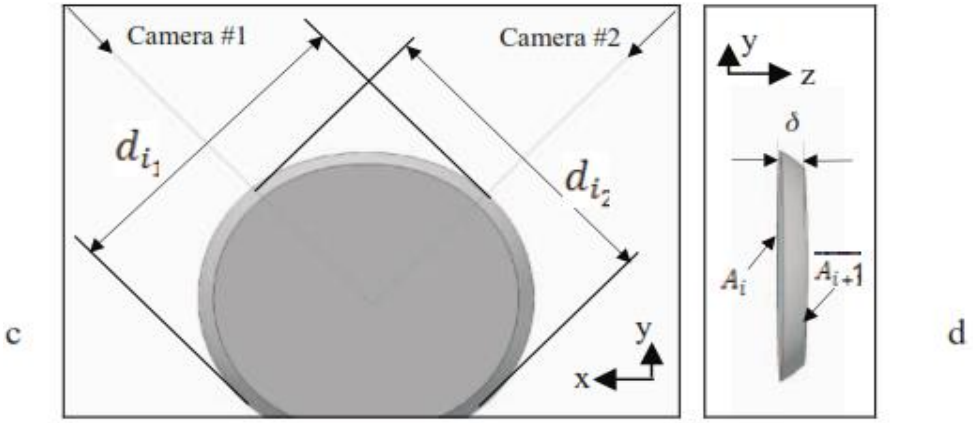
Şekil 2-20 VMS düzeneği (Khojastehnazhand ve ark., 2008)

Şekil 2-20'de görülen ölçüm düzeneğinde beyaz zemine yerleştirilen portakal üzerine biri diğerine 90^{o} açıyla ve portakala 25 cm mesafede bakan 2 adet CCD sensörlü renkli kamera yerleştirilmiştir. Yüksek yoğunlukta florasan lamba ışıkları ile aydınlatılan portakal etrafında gölge oluşumuna izin verilmemiştir. Gölgesiz renkli görüntü CCD sensörlü kameralar tarafından algılanmış ve bilgisayara aktarılmıştır. Visual Basic 6.0 programı ile önce arka plan çıkarma (backround segmentation), sonra görüntüyü gürültü piksellerinden arındırma (image enhancement) işlemleri yapılmıştır.

Arka plan çıkarma işlemi için önce boş beyaz arka olan görüntüsü ve her pikselin renk değeri,

P(x,y) = B 216 + G 28 + R (2-3)

Formülü ile hesaplanarak veri tabanına kaydedilmiştir. Portakal görüntüsü kameralar tarafından algılanarak yukarıdaki P(x,y) formülü ile her pikselin renk değeri hesaplanmış ve arka plan değeri ile karşılaştırılmıştır. Karşılaştırma sonucu fark değerleri bulunmuş ve portakal görüntüsü arka plandan çıkarılmıştır.



c d

Şekil 2-21 Portakalın x, y ve z eksenlerinde dilimlenmiş gösterimi

[8]'de olduğu gibi burada da portakal görüntüsü önce **δ** kalınlıkta dilimlere ayrılmıştır. İki kamera ile çalıştığımızdan kamera1 ve kamera2'den alınan portakal görüntülerinden $\mathbf{d_{i1}}$ ve $\mathbf{d_{i2}}$ çapları elde edilmiştir. Her bir dilimin yüzey alanı A_i (2-4)'deki formüllerle hesaplanmıştır.

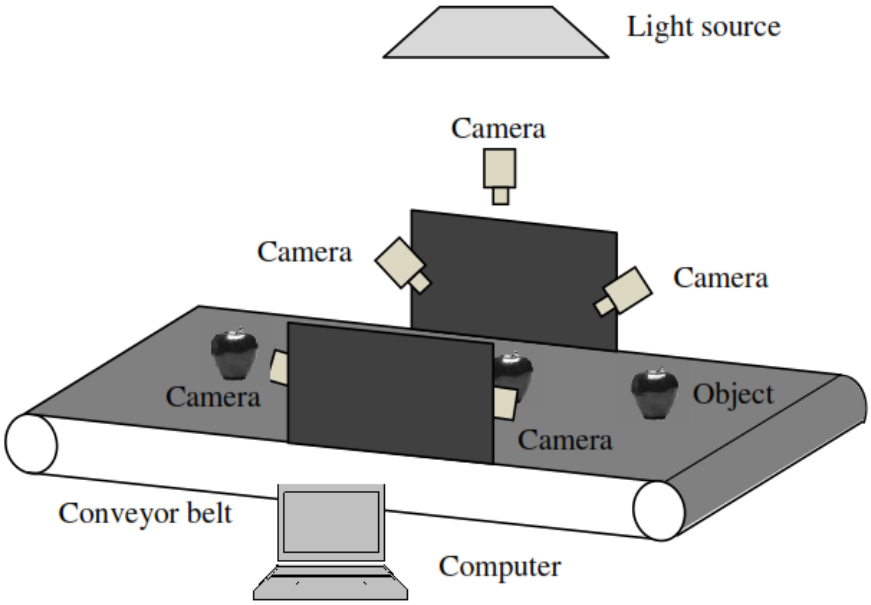
$$A_i = \pi \frac{d_{i1} d_{i2}}{4} \quad , \quad V_i = \frac{A_i + A_{i+1}}{2} \quad , \quad V_{IP} = \sum_{i=1}^{n} V_i \qquad (2\text{-}4)$$

(2-4) formüllerinden görüldüğü gibi, $\mathbf{A_i}$ değeri $\mathbf{di_1}$ ve $\mathbf{di_2}$ yarıçapları kullanılarak hesaplanmıştır. Daha sonra **Ai** değeri **δ** ile çarpılarak her bir dilimin hacim değeri V_i hesaplanır. Tüm piksellerin hacimleri toplanarak toplam portakal hacmi $\mathbf{V_{ıp}}$ hesaplanır. Portakalın toplam yüzey alan $\mathbf{S_{ıp}}$ ise;

$$S_i = \frac{\pi}{4}(d_{i1} + d_{i2} + d_{i1+1} + d_{i2+1}) \sqrt{\delta^2 + \left(\frac{d_{i_1} + d_{i_2}}{4} - \frac{d_{i_1+1} + d_{i_2+1}}{4}\right)^2} \qquad (2\text{-}5)$$

$$S_{IP} = \sum_{i=1}^{n} S_i \qquad \text{(2-6)}$$ formülleri ile hesaplanır.

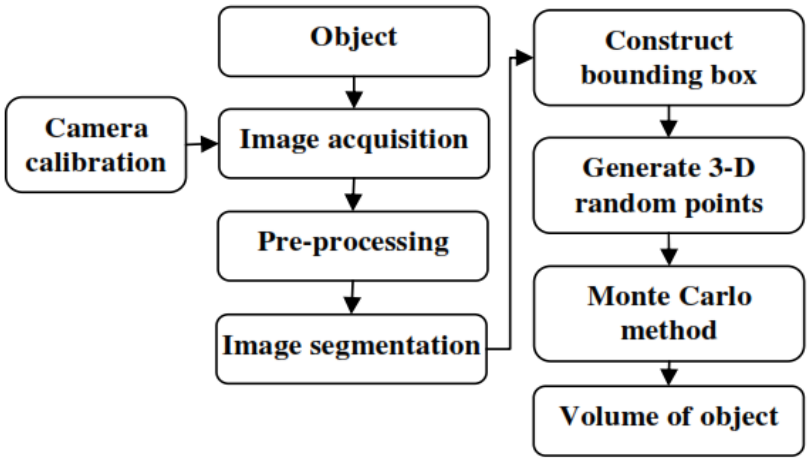
Siswantoro ve ark. "Volume Measurement Algorithm for Food Product with Irregular Shape using Computer Vision Based on Monte Carlo Method" adlı çalışmalarında Şekil 2-22'deki düzeneği oluşturmuşlardır [11].



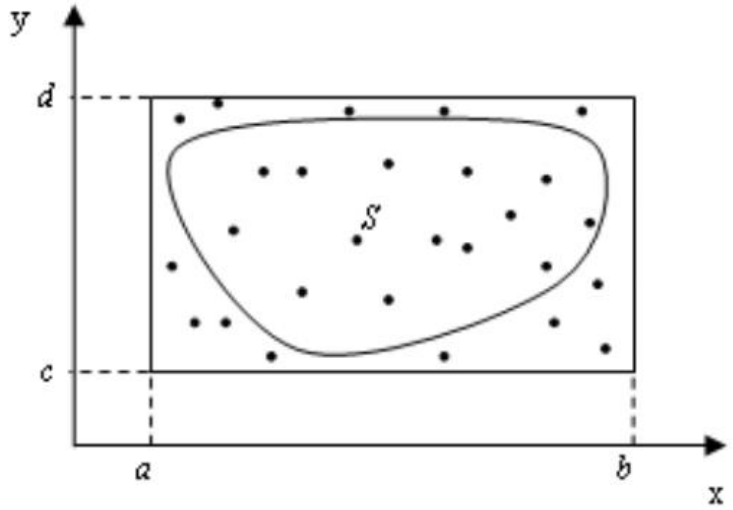
Şekil 2-22 VMS düzeneği (Siswantoro ve ark., 2014)

Ölçüm düzeneğinde beş adet kamera, ışık kaynağı ve siyah renkli taşıyıcı konveyör kullanılmıştır. Bir kamera ölçülecek nesnenin tam üzerine diğer dört kamera nesneyi çevreleyecek şekilde eşit aralıklarla ve eşit açılarla yerleştirilmiştir. Kameralar kalibre edildikten sonra konveyör üzerinde hareket eden nesneler kameraların altından geçerken renkli görüntüleri beş kamera tarafından alınmıştır.

Şekil 2-23'teki iş akış şemasından da görülebileceği gibi nesneye ait görüntüler kameralar tarafından alındıktan sonra çeşitli görüntü işleme yöntemleri uygulanmış ve en sonunda Monte Carlo metodu ile nesneye ait hacim hesaplanmıştır.



Şekil 2-23 İş akış şeması



Şekil 2-24 Monte Carlo metodu kullanılarak alan hesabı yaklaşımı

Şekil 2-24'te görüldüğü gibi Monte Carlo alan hesabı yaklaşımında, ölçümü yapılacak S alanının dışına bir dikdörtgensel [a, b, c, d] alanı çizilerek bu alan içerisine rasgele noktalar serpiştirilir. Daha sonra S alanı içerisine düşen noktalar (N adet) sayılır. Yaklaşık alan $[(b - a) \times (d - c)] \times N$ olarak hesaplanır.

Monte Carlo yöntemi düzgün olmayan alanların hesaplamasında etkili bir yöntemdir. Monte Carlo yöntemi yaygın olarak matematikte çok boyutlu integral problemlerinin çözümünde lineer ve lineer olamayan, boundary (sınır, kenar) değeri problemlerinde, istatistiksel fizik ve kimya, nükleer fizik, trafik, desen modellemede ve ekonomide kullanılan bir metottur [12].

Bu yöntemde sırasıyla;

1. Gerçek koordinat sistemine göre kameraların kalibrasyonu yapılmıştır.

2. ölçülecek nesnenin renkli görüntüsü beş ayrı kamera ile alınmıştır.

3. Görüntüdeki gürültü yok edilmiş görüntü kalitesi ve kontrast arttırılmıştır.

4. Görüntü gri seviyeli formata çevrilmiştir.

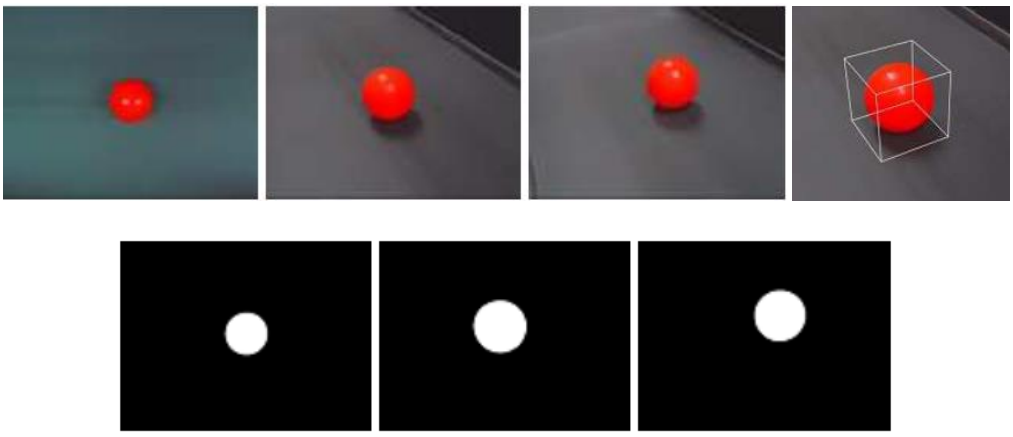
5. Görüntüden arka plan çıkartılmıştır.

6. Monte Carlo çerçeve kutu (bounding box) oluşturulmuştur.

7. Nokta bulutu rastgele çerçeve kutuya serpiştirilmiştir.

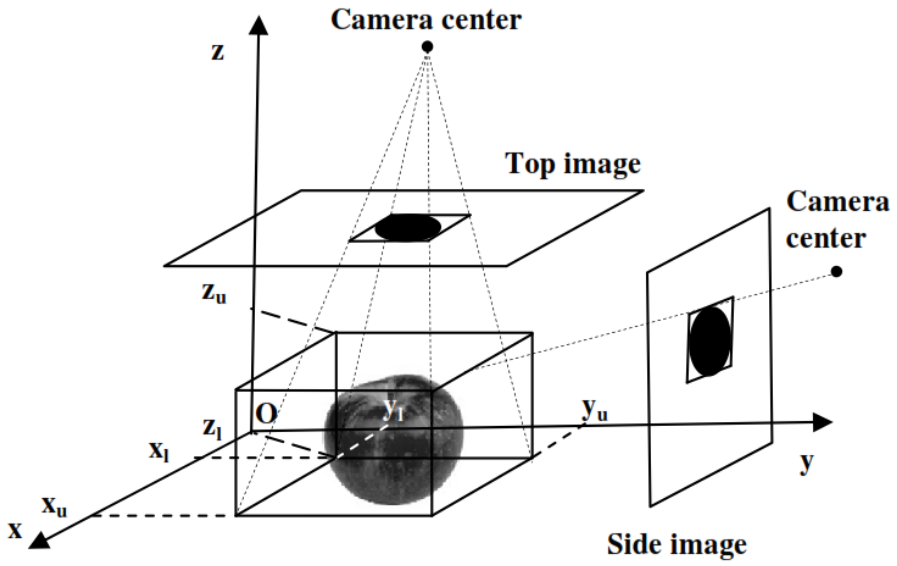
8. Ölçümü yapılan nesne içerisine düşen nokta sayısı $\mathbf{N_o}$, toplam çerçeve kutu içerisindeki nokta sayısı $\mathbf{N}$ olmak üzere;

$$V = (x_u - x_l)(y_u - y_l)(z_u - z_l)\frac{N_0}{N}$$ (2-7) ile hacim hesabı yapılmıştır.



Şekil 2-25 Nesnelerin renkli ve binary görüntüleri

Şekil 2-26'da Monte Carlo çerçeve kutu hacim yapısı 3 boyutlu olarak gösterilmiştir.



Şekil 2-26 Monte Carlo çerçeve kutu hacim yapısının gösterimi

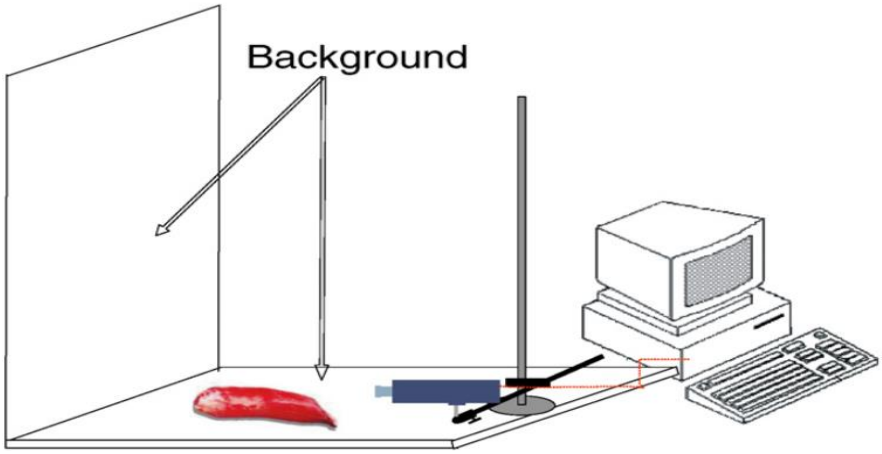
Ayrıca Siswantoro ve ark. daha önce meyve hacmi ölçümü ile ilgili yapılan çalışma sonuçlarını ve doğruluklarını Tablo 2-2'de göstermişlerdir.

Tablo 2-2 Daha önce yapılan ölçümler tablosu

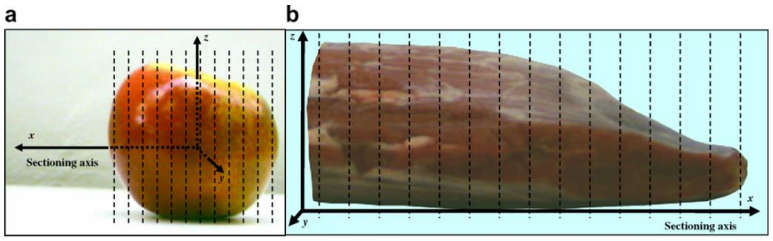
Authors	Methods	Object	Accuracy
Lee et al. [1]	Wire frame model	Apple, cantaloupe, strawberry, tomato	0.025[a] 0.956[b]
Chalidabhongse et al. [20]	Space carving	Mango (SS size)	0.0889[a]
		Mango (S size)	0.0996[a]
		Mango (L size)	0.0963[a]
Goñi, et al. [21]	Lofting	Granny Smith apple	0.992[b]
		Red delicious apple	0.998[b]
		Meat	0.995[b]
Castaneda & Turchiuli [6]	Volume intersection	Agglomerated milk	0.044[a]

Goni ve ark. "Three-dimensional Reconstruction of Irregular Foodstuffs" adlı çalışmalarında, geometrik şekilleri düzgün yapıda olmayan gıda maddelerinin hacim ve yüzey alanı hesaplamalarını yüksek doğrulukta yapmayı başarmışlardır [13].

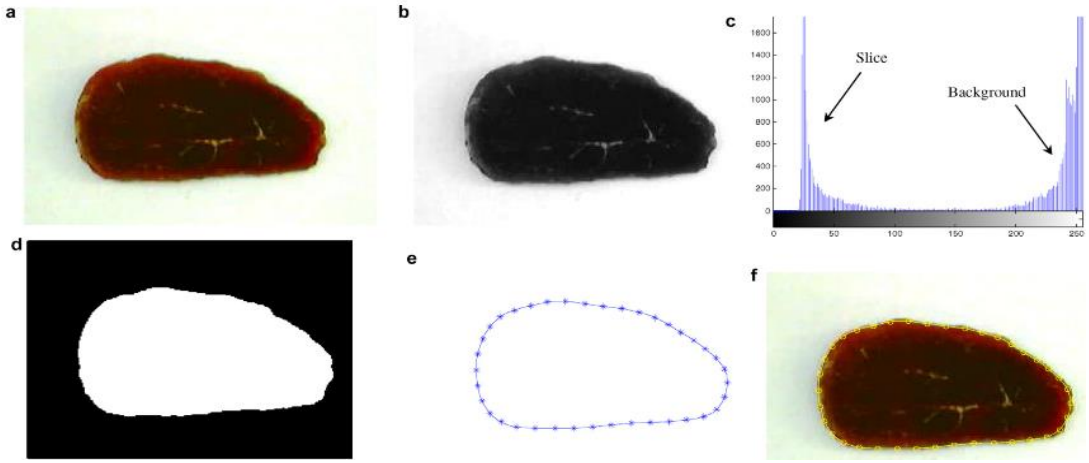
Şekil 2-27'de ölçüm düzeneği görülmektedir.



Şekil 2-27 VMS düzeneği (Goni ve ark., 2007)



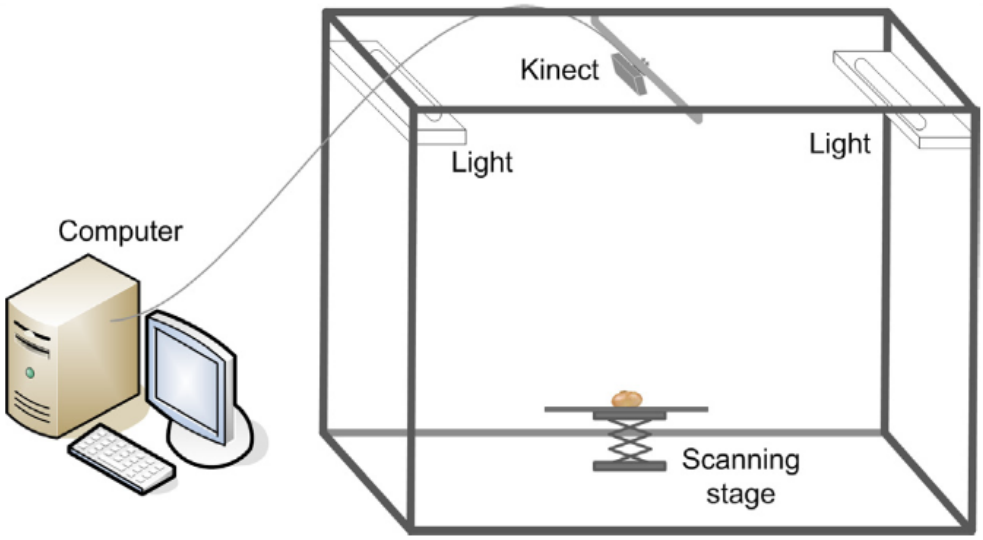
Şekil 2-28 Elma ve et parçasının dikey eksende dilimlenmesi



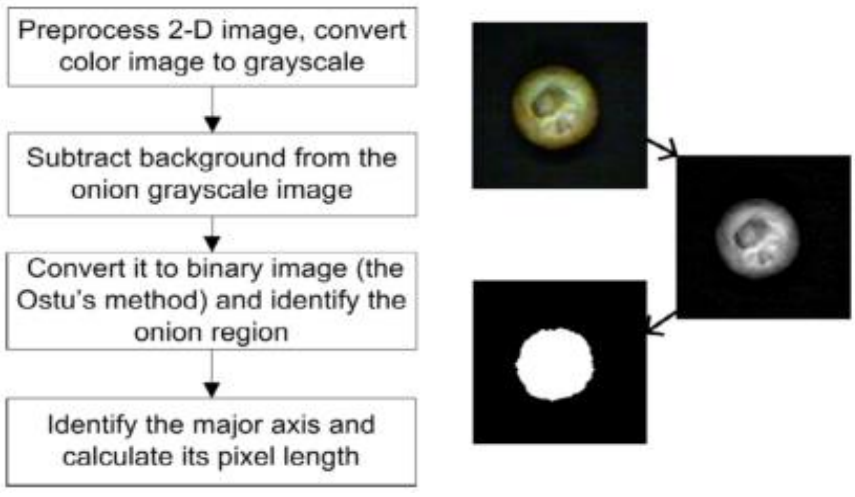
Şekil 2-29 Et parçasına ait işlenmiş görüntüler

Şekil 2-29'da a) Renkli görüntü, b) Gri seviyeli görüntü, c) Gri seviyeli histogram, d) Binary görüntü, e) Binary görüntünün B-Spline kenar noktaları, f) Renkli görüntü ve B-Spline kenar noktalarının birleşimi gösterilmiştir.

Wang ve Li “Size Estimation of Sweet Onions Using Consumer-Grade RGB-Depth Sensor” adlı çalışmalarında soğanın boyutlarını ölçmek için Şekil 2-30’daki ölçüm düzeneğini kurmuşlardır. 1100x600x1100 büyüklüğündeki kabin içerisine iki duvarında florasan lamba aydınlatması ve soğanı tam yukarıdan gören 1 adet MS Kinect RGB-D kamera yerleştirilmiştir. Yazılım olarak MATLAB R2013b ve Image Proccessing Toolbox kullanılmışlardır [14].



Şekil 2-30 VMS düzeneği (Wang ve Li, 2014)



Şekil 2-31 RGB görüntüden maksimum soğan çapı hesabı akış şeması

Şekil 2-31’deki akış şemasına göre önce renkli görüntüden soğanın olduğu yere karşılık gelen 160x160 piksellik alan alınmış ve gri seviyeli görüntüye çevrilmiştir.

Daha sonra arka plan görüntüden ayrıştırılarak sadece soğan görüntüsü elde edilmiştir. Bu soğan görüntüsü Otsu metodu kullanılarak binary formata çevrilmiş ve morfolojik açma ve kapatma işlemleri ile üzerindeki benekler yok edilmiştir. Daha sonra binary görüntüden beyaz bölge pikselleri toplanarak maksimum soğan çapı bulunmuştur.

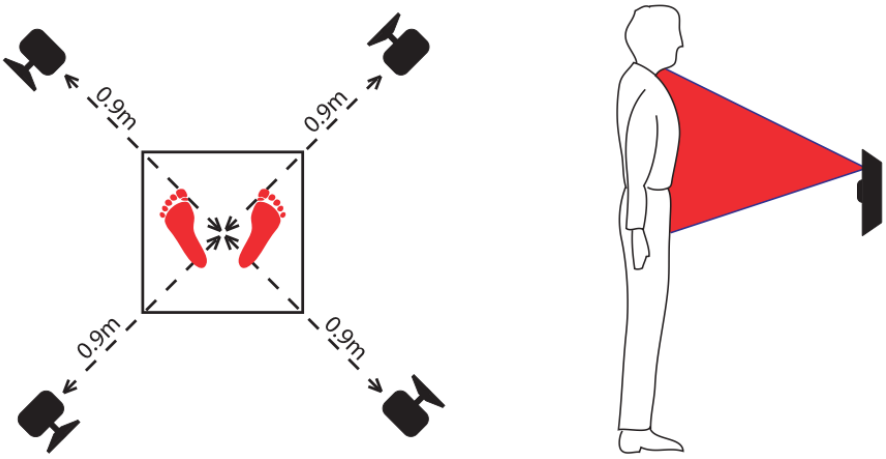
Daha sonra kameradan gelen 11-bit derinlik görüntüsünden soğanın yer aldığı 160x160=25600 piksellik kısmı alınmış ve arka plandan ayrıştırılmıştır. Ortaya çıkan derinlik görüntüsünün piksellerinden soğanın ait ortalama derinlik hesaplanmıştır. Bu hesaplanan derinlik bilgisinden her bir pikselin derinlik bilgisi çıkartılarak her bir nokta için $2^{11=}$ 2048 bitlik I (x,y,z) nokta bulutu elde edilmiştir. En üstteki nokta ile zemin arasındaki farktan soğan yüksekliği hesaplanmıştır.

Carreira ve ark. "Volumetrics - Measuring Free Volumes" adlı çalışmasında Micrososft Kinect kamera kullanarak 480x640= 307200 adet noktadan oluşan nokta buluntundan (depth point cloud) konveyörde ilerleyen kasaların boş olup olmadığının kontrolü ve tır konteyneri içindeki boş kalan hacim ölçümü yapmışlardır [15].

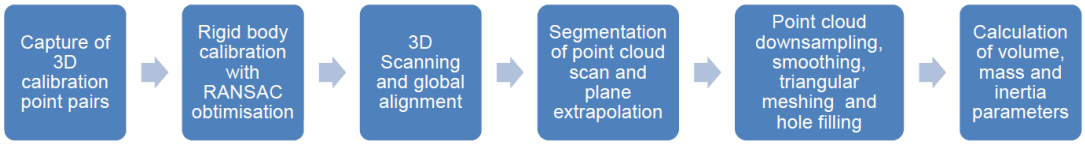
Şekil 2-32 VMS düzeneği **(**Carreira ve ark., 2013)

Clarkson ve ark. İnsan vücut hacmini ölçmek için etrafına 4 adet RGB-D kamera yerleştirip her bir kameranın derinlik nokta bulutundan vücut hacmini yüksek doğrulukta hesaplamışlardır [16].

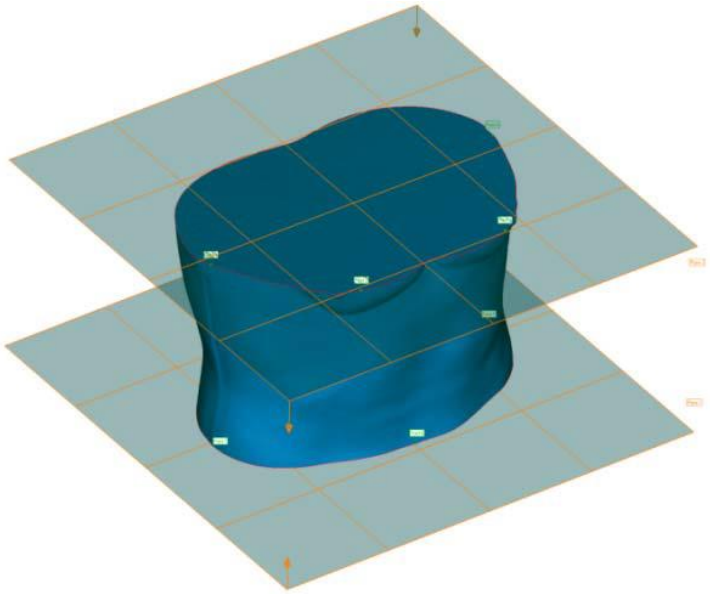
Bu çalışmayla ilgili olarak Şekil 2-33'te ölçüm düzeneği, Şekil 2-34'te iş akış şeması ve Şekil 2-35'te insan vücudunu kesitinin çıkarılması görülmektedir.



Şekil 2-33 VMS düzeneği (Clarkson ve ark., 2014)



Şekil 2-34 İş akış şeması



Şekil 2-35 İnsan vücudu kesitinin çıkarılması

3. YÖNTEM

Bu tez projemiz, hava limanlarında, yolculara ait bagajların ve kargoların boyut, hacim ve ağırlıklarının ölçüm işini kapsamaktadır. Bu kapsamda,

1. El bagajları ve kargo paketlerinin boyutlarının ölçümü için RGB-D kamera kullanılacaktır.
2. Ağırlık ölçümü için yük hücresi (loadcell) ve ağırlık göstergesi/RS232 çevirici kullanılacaktır.
3. Barkod bilgisini okumak için USB çıkışlı el tipi barkod okuyucu kullanılacaktır.
4. Yazılım olarak MATLAB kullanılacaktır.

Yapılacak MATLAB GUI arayüzünde, kameradan alınan renkli görüntü, derinlik görüntüsü, ölçümü yapılan nesneye ait en, boy, yükseklik ve hacim bilgileri, ağırlık bilgisi, hacimsel ağırlık bilgisi ve nesne üzerindeki barkod bilgisi gösterilecektir.

3.1. Dijital Sensörlü Kameralar ve Çalışma Prensipleri Hakkında

Dijital sensörlü kameralar, görüntüleri piksel denilen küçük kareciklere sığdırır. Görüntünün 1 karesindeki piksel sayısı ne kadar büyük olursa fotoğrafımız da o kadar büyük olur. İnsanlar piksel arttıkça görüntü kalitesinin arttığını sanmaktadır. Ancak işin aslı öyle değildir. Piksel sadece fotoğrafın boyutuyla ilgilidir.

Piyasada görüntü almak için yaygın olarak kullanılan iki çeşit kamera teknolojisi vardır. CCD sensörlü kameralar CMOS sensörlü kameralar. CCD ve CMOS sensörler bildiğimiz yarı iletken transistörlü elektronik devreler gibidir sürekli kullanılabilen fotoğraf filmi gibi görüntüleri alırlar. Bu cihazların üzerinde, en az cihazın çözünürlüğü kadar sensör/devre vardır ve bu devreler, o noktaya düşen ışığı piksel cinsinden dijital ortama yansıtırlar. 5MP bir dijital fotoğraf makinesi üzerinde 2560 x 1920 adet, yani yaklaşık 5 milyon adet mini sensör bulunur. Şekil 3-1'de CCD ve CMOS sensörler gösterilmiştir.

Şekil 3-1 CCD ve CMOS sensörler

3.2. CCD Sensörlü Kameralar

Dijital görüntü almak için üretilen ilk kameralarda, görüntüleri analog ışık sinyallerinden dijital piksellere çevirmek için CCD (Charged Coupling Devices) kullanılmaktaydı. CCD sensörler mükemmel görüntü sağlayan yüksek kaliteli sensörlerdir. CCD sensörler ışığa karşı daha hassastırlar, bunun sonucunda loş ortamlarda bile kaliteli görüntü alabilme yetenekleri vardır. Aldıkları görüntüde parlaklık ve netlik olarak daha kalitelidir. Aynı model CCD sensörler arasında görüntü farklılıkları en alt seviyededir. CCD sensörler arka planda daha düşük gürültü üretirler. CCD sensörlerde her piksele ait akım genellikle tek bir çıkış noktasından aktarılır, voltaja dönüşür, depolanır ve analog sinyal olarak sensörden dışarı verilir. Böylelikle piksellerin tümü ışığı yakalamak için kullanılır. Bunun sonucunda da görüntünün tek tipliliği oluşmuş olur. Bu durum görüntünün kalitesini etkileyen en önemli faktörlerden birisidir.

3.3. CMOS Sensörlü Kameralar

CMOS (Complimentary Metal Oxide Semiconductor) sensörler, üzerine düşen fotonları algılamak ve elektriksel sinyale dönüştürerek iletmek için her pikselde bir çift metal oksit transistör (MOS) kullanmaktadır. Her piksel birbirinden bağımsız olarak işlem gördüğü için bu durum esneklik sağlamaktadır. CMOS sensörün üretim teknikleri mikroçiplerin üretim teknikleri ile aynıdır. CMOS sensörleri üretmek daha kolay olduğu için CMOS sensörler CCD sensörlerinden daha ucuzdur. CMOS sensörler dijital kameraların fiyatlarının düşmesine sebep olmuştur.

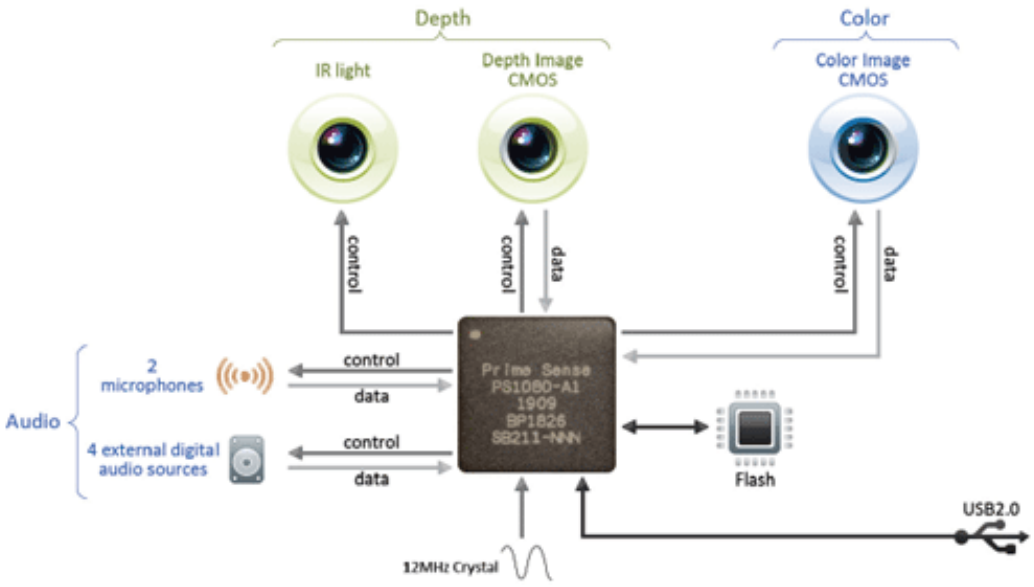
3.4. CCD ve CMOS Sensörlü Kameralar Arasındaki Farklar

En büyük fark CCD sensörlerin düşük gürültülü yüksek kaliteli görüntüler oluşturmasıdır. CMOS sensörler gürültülü olmaya daha meyillidir. CMOS sensörleri düzgün bir ışıklandırmada düşük parazitli görüntü oluşturmak için daha fazla ışığa ihtiyaç duymaktadır. Bu, CMOS sensörlerin CCD'den tamamen düşük kaliteli olduğu anlamına gelmez. CCD sensörler dijital kameralarda daha uzun süredir piyasada bulunmaktadır ve bu yüzden teknolojisi daha çok geliştirilmiştir. CMOS sensörlerde yavaş yavaş bu teknolojiyi yakalamaktadır ve sonuçta çözünürlük ve kalite açısından CCD ile eşdeğer olacaktır. CMOS sensörleri üretmek daha kolay olduğu için CMOS sensörleri CCD sensörlerinden daha ucuzdur. CMOS sensörleri güç tüketimi bakımından CCD sensörlerinden daha üstündür. Daha uzun batarya ömrü CMOS kamera ile daha fazla resim çekmeniz anlamına gelmektedir [17].

CCD sensörler çalışmaları esnasında daha fazla enerji harcarlar buda aynı çekim sayısında daha fazla enerji kaynağı tüketimi demektir. CMOS sensörlere göre üretimleri daha pahalıdır. Ebatlarının büyüklüğü ve fazla enerji kullanmaları nedeniyle cep telefonu gibi kısıtlı enerji kaynağı olan ürünlerde kullanılamazlar. Ekrana çok parlak bir ışık geldiğinde direkt gün ışığı veya doğrudan aydınlatma durumlarında sensör altında ve üstünde şeritler oluşturabilir. Bu durum "lekeli çiçek açması" diye tanımlanır [18].

3.5. RGB-D Kameralar ve Çalışma Prensibi

Şekil 3-2' de görüldüğü gibi RGB-D kameralar içerisinde cisimden gelen renkli (RGB) görüntüyü algılamak için renkli görüntü sensörü (color image sensor) ve derinlik görüntüsünü algılayabilmek için algılanacak cisme infrared dalga boyunda yapılandırılmış infrared ışık gönderen bir IR ışık kaynağı ve cisimden yansıyan IR ışıkları algılayacak bir derinlik sensörü (depth sensor) mevcuttur.



Şekil 3-2 RGB-D kamera iç yapısı

CMOS yapıdaki RGB (color image) sensör üzerinde, her piksel de, kırmızı renk için, yeşil renk için ve mavi renk için olmak üzere 3 adet CMOS sensör bulunur. Cisimden yansıyan ışıktan oluşan renk fotonlarının renklere göre dalga boyları farklı olduğundan bu sensörlerin her biri kendisine ait rengi algılayarak 8-bit dijital bilgiye dönüştürür. Yani renkli görüntüde her piksel, algıladığı rengi 8+8+8=24 bitlik bir bilgi olarak dışarıya verir. 640x480 matristen oluşan bir CMOS RGB kamera, 307200 adet piksele sahiptir ve anlık olarak 307200x24 bit bilgi üretir.

CMOS yapıdaki derinlik görüntü sensöründe, her pikselde bir adet CMOS sensör çifti bulunmakta ve bu sensör IR ışık kaynağından cisme gönderilip cisimden geri yansıyan IR ışıkları algılar ve bu algılama sonucu her piksel 16-bit gri seviye bilgi üretir. Bu bilgi aslında ışığın uçuş süresinden elde edilen bir bilgi olduğundan her piksel, kendisine görüntü üzerinde karşılık gelen derinlik bilgisini mm cinsinden vermiş olacaktır. IR ışık kaynağı ve "Depth Image Sensor" çiftinden oluşan bu kamera çeşidi genel olarak ToF (Time of Flight) kameralar olarak isimlendirilmektedir.

3 boyutlu uygulamalar için üretilen ToF kameralar cisme tam üst noktadan bakıp derinlik bilgisi aldıklarından genelde gerçek zamanlı 3D kameralar olarak kullanılırlar. 640x480 piksel çözünürlükte 30 fps ToF kameralar endüstriyel ölçümlerde, tüketici elektroniğinde ve el, kol, yüz hareketini algılamada sıkça

kullanılmaya başlanmıştır. Aşağıda çeşitli firmalar tarafından üretilen ve piyasada kullanılan RGB-D kameralar gösterilmiştir.

Şekil 3-3 Kinect for Windows V1 kamera ve Kinect for Windows V2 kamera

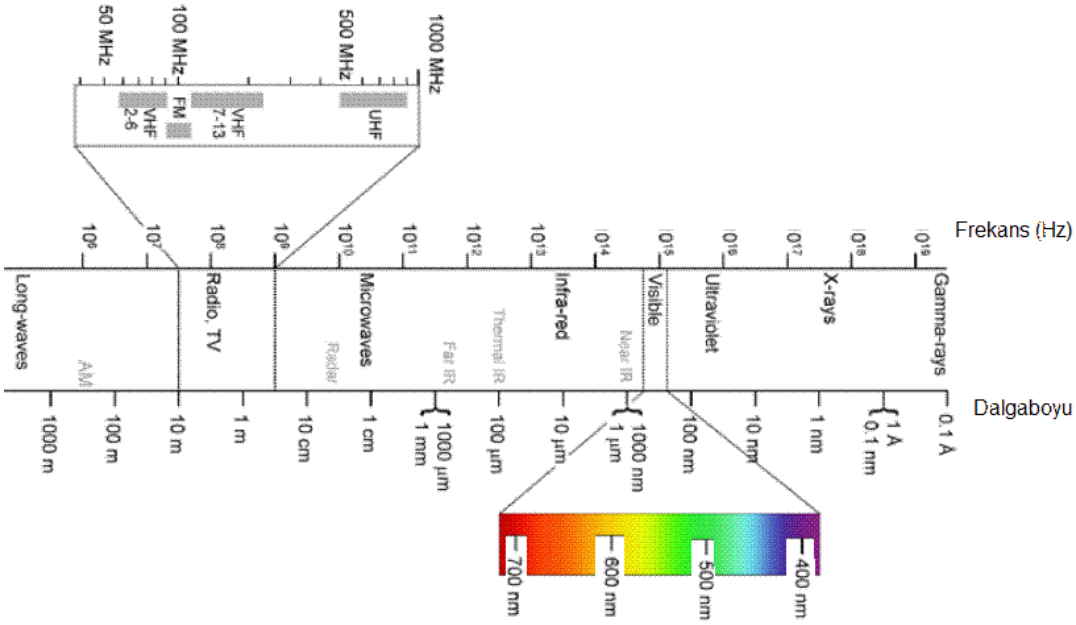
Şekil 3-4 Asus Xtion Pro Live kamera ve Primesense Carmine kamera

Şekil 3-5 Creative Senze3D kamera ve Softkinetic DS325 kamera

4. RESİM FORMATLARI VE DÖNÜŞÜM YÖNTEMLERİ

4.1. Renkli Görüntü (RGB Images) Renk Formatı

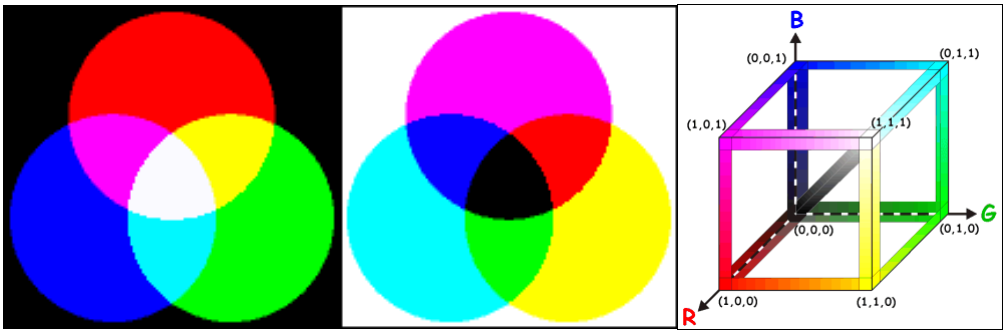
Işıktaki ana renkler kırmızı, yeşil ve mavidir. Bu üç ana rengin dalga boyu dağılımı Şekil 4-1'de gösterilmiştir. Cisimlerden yansıyıp gözümüze giren 700 nm dalga boyundaki ışık kırmızı, 500 nm dalga boyundaki ışık yeşil, 400 nm dalga boyundaki ışık mavi ve tonları olarak algılanır.



Şekil 4-1 Elektromanyetik yayılım ve renklerin dalga boyları

Literatürde bu renkler ana renkler (additive colors) olarak geçer. Ana renkler RGB (R=kırmızı, G=yeşil, B=blue) kısaltması olarak da karşımıza çıkar. Ana renkler siyah zemin üzerine kırmızı, yeşil ve mavinin yerleştirilmesi sonucu başlar. Şekil 4-2'de görüldüğü gibi üç ana rengin karışımı sonucu beyaz renk ortaya çıkar.

Şekil 4-2'de RGB renk uzayı ve bu renk uzayında kırmızı, yeşil ve mavi ana renkler ve bu ana renklerin birbiri ile karışımı sonucu oluşabilecek diğer ara renkler gösterilmiştir.



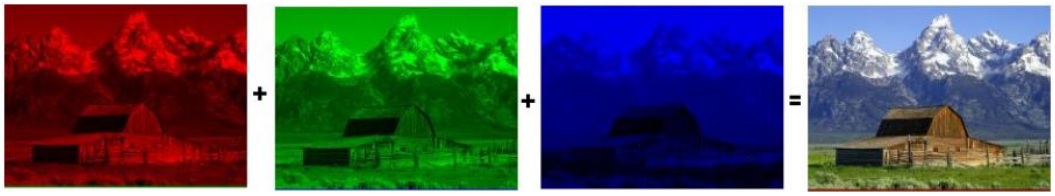
Şekil 4-2 Ana ve ara renkler ve RGB renk uzayı

Alt renkler (subtractive colors) ana renklerin ikişer ikişer karışımı sonucu elde edilen renklerdir. Bu alt renkler camgöbeği (cyan), magenta ve sarıdır. Yukarıdaki şekilde ana ve alt renkler gösterilmiştir. Alt renkler beyaz zemin üzerine cyan, magneta ve sarının yerleştirilmesi ile oluşturulur. Bu üç alt rengin karışımı sonucu siyah renk ortaya çıkar. Alt renk modeli genelde CMYK (**C**yan, **M**agneta, **Y**ellow, blac**K**) model olarak adlandırılır ve genelde beyaz zemin üzerinde oluşturulduğundan baskı, boya ve matbaacılık sektöründe bu model kullanılır.

Şekil 4-2'deki renk küpünde görüldüğü gibi (0,0,0) noktasında hiçbir renk yoktur ve siyahtır. (1,1,1) noktasında tüm renkler en yüksek parlaklık (intensity) değerinde bulunduğundan beyaz renk ortaya çıkmıştır. Beyaz ve siyah nokta arasındaki çizgi gri çizgidir ve eşit miktardaki kırmızı, mavi ve yeşil rengin birleşmesi sonucu gri renk ortaya çıkmıştır.

RGB görüntünün her pikseli, kırmızı renk ve tonları için 8-bit, yeşil renk ve tonları için 8-bit, mavi renk ve tonları için 8-bit olmak üzere toplam 24-bitlik bilgi içerir. Buna birde 8-bit parlaklık (intensity) değeri eklendiğinde 1 pikselin rengi 32-bit veri ile ifade edilebilir.

Aslında RGB görüntü her piksel için, her biri kendi renk aralığında ifade edilen kırmızı, yeşil ve mavi renk tonlarından oluşan 3 görüntünün sırası ile üst üste konulması ile elde edilir. Şekil 4-3'te sırasıyla görüldüğü gibi en üste kırmızı rengin tonlarından oluşmuş 8-bitlik görüntü, ortaya yeşil rengin tonlarından oluşmuş 8-bitlik görüntü, en alta ise mavi tonlarından oluşmuş 8-bitlik görüntü konulduğunda renkli (RGB) görüntü ortaya çıkar.



Şekil 4-3 RGB resimlerin üç ana renkten oluşumu

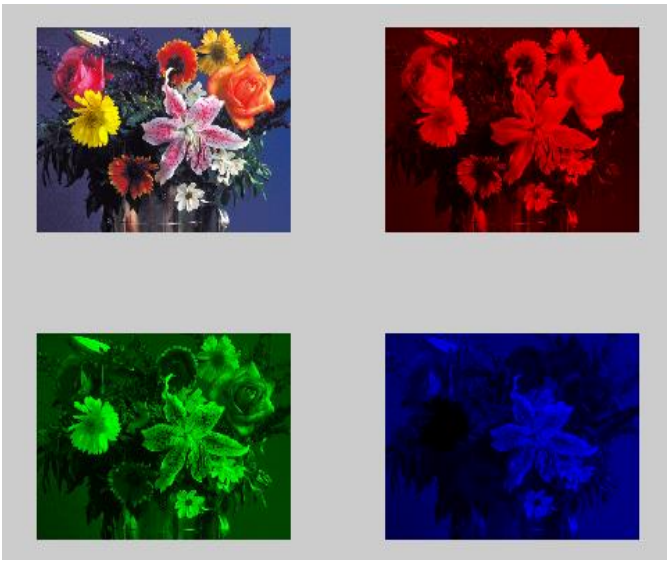
RGB görüntüde parlaklık değeri her bir renk pikseli için 0-255 arasında ifade edilebilir. Her üç ana renk için parlaklık değeri [ab]x[cd] matris görüntüsü için,

***f*: [*a*,*b*] x [*c*,*d*]** ,

I= *f*(*x*, *y*) ise **(*x*, *y*)** pozisyonundaki **parlaklık** değeri,

$$\boldsymbol{f}(x,y)= \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$ [19] şeklinde ifade edilir.

Bir RGB görüntüdeki herhangi bir f(x,y) pikseli için Irgb(x,y)= [0,0,0] - [255,255,255] aralığında değer alabilir. Aşağıda MATLAB'de RGB görüntünün bileşenlerine ayrılması gösterilmiştir.



Şekil 4-4 MATLAB'de RGB görüntünün bileşenlerine ayrılması

```
a=imread('flowers.tif');
subplot(2,2,1);
imshow(a);
R=a; G=a; B=a;
R(:,:,2:3)=0;
subplot(2,2,2);
imshow(R);
G(:,:,1)=0;
G(:,:,3)=0;
subplot(2,2,3);
imshow(G);
B(:,:,1)=0;
B(:,:,2)=0;
subplot(2,2,4);
imshow(B);
```

Aşağıda, MATLAB'de RGB görüntünün boyutlarının küçültülmesi, gri seviyeli resme ve sonrasında siyah-beyaz (binary) resme dönüştürülmesi gösterilmiştir.

Şekil 4-5 MATLAB'de RGB görüntünün gri ve ikilik formata çevrilmesi

```
A=imread('flowers.tif');
subplot(2,2,1);
imshow(a);
subplot(2,2,2);
b=imresize(a,[256,256]);
imshow(b);
subplot(2,2,3);
c=rgb2gray(b);
imshow(c);
subplot(2,2,4);
d=im2bw(c);
imshow(d);
```

4.2. HSV Görüntü (Hue, Saturation, Value) Renk Formatı

HSV (Hue, Saturation, Value) veya HSB (Hue, Saturation, Brightness) renk uzayı, renkleri sırasıyla renk özü, doygunluk ve parlaklık olarak [42] tanımlar. Renk özü, rengin baskın dalga boyunu belirler. Doygunluk, rengin "canlılığını" belirler. Yüksek doygunluk canlı renklere neden olurken, düşük doygunluk rengin gri tonlarına yaklaşmasına neden olur. Parlaklık ise rengin aydınlığını yani içindeki beyaz oranını belirler. MATLAB'de RGB renk formatından HSV renk formatına dönüşüm "***rgb2hsv***" komutu ile yapılmaktadır. HSV renk formatından RGB renk formatına dönüşüm "***hsv2rgb***" komutu ile yapılmaktadır.

4.3. YCbCr Görüntü (YUV) Renk Formatı

YCbCr renk formatı, dünya çapında sayısal video standardı oluşturma çabaları sırasında ortaya çıkmıştır. Bu standart PAL, SECAM, NTSC kompozit renkli video standartlarında kullanılır. YCbCr formatında, Y luminance (parlaklık) değerini, Cb ve Cr ile ise chrominance (renk) değerlerini gösteren değerlerdir. (4-1), (4-2) ve (4-3) numaralı denklemlerde RGB renk formatının YCbCr (YUV) renk formatına nasıl dönüştürüldüğü görülmektedir.

$$Y=0.299*R + 0.587*G + 0.114*B \quad (4\text{-}1)$$

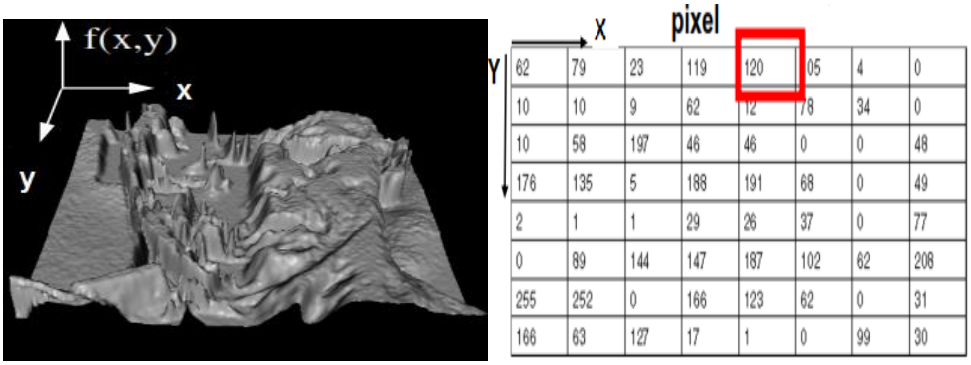
$$Cb=U=0.436*(B-Y) / (1 - 0.114) \quad (4\text{-}2)$$

$$Cr= V= 0.615*(R-Y) / (1 - 0.299) \quad (4\text{-}3)$$

Y parlaklık değeri, 8-bit olarak 16-235aralığında tanımlanmaktadır. Cb ve Cr ise de 16-240 arasında tanımlanmaktadır. MATLAB'de RGB renk formatından YCbCr renk formatına dönüşüm "***rgb2ycbcr***" komutu ile yapılmaktadır. YCbCr renk formatından RGB renk formatına dönüşüm "***ycbcr2rgb***" komutu ile yapılmaktadır.

4.4. Gri Seviyeli Görüntü (GreyScale Image) Formatı

8-bitlik gri seviyeli bir görüntünün her pikseli $\mathbf{2^{8=}}$ 0-255 arasında, 16-bitlik gri seviyeli bir görüntünün her pikseli $\mathbf{2^{16}}$=65535 arası değer alabilir.



Şekil 4-6 8-bitlik gri seviyeli görüntü ve 0-255 aralığında gösterimi

PAL, NTSC, SECAM kompozit renkli video standartlarına göre RGB formatlı resimler gri seviyeli görüntülere (4-4) numaralı formülle çevrilir. Y, gri seviyeli görüntünün her hangi bir pikselinin gri seviye değeri olmak üzere,

$$Y=(0.299xR) + (0.587xG)+ (0.114xB) \quad (4\text{-}4)$$ olarak hesaplanır.

(4-4) numaralı dönüşüm formülünde kırmızı(R), yeşil(G) ve mavi(B) renklerinin griye dönüşüm katsayıları 0.333 olmalıydı ama farklı olarak alınmıştır. Bunun sebebi insan gözündeki koni (cone) algılayıcılarının kırmızı, yeşil ve mavi renklere duyarlılıklarının farklı olmasından kaynaklanmaktadır. İnsan beyni gözden gelen bu üç ana rengin birleşiminden cisimlerin renklerini oluşturmaktadır.

Örnek verecek olursak, f(x,y) = [100,0,150] olan RGB resimdeki f(x,y) pikselin gri dönüşüm karşılığı,

Y = (0.299*100)+(0.587*0)+(0.114*150) = 47 olur.

Tablo 4-1'de bazı RGB renklerin gri seviyeli renklere dönüşümü gösterilmiştir.

Tablo 4-1 Bazı RGB renklerin gri seviyeli renklere dönüşümü

Kırmızı (255,0,0)		Gri Karşılığı 76	
Yeşil (0,255,0)		Gri Karşılığı 150	
Mavi (0,0,255)		Gri Karşılığı 29	
Camgöbeği (0,255,255)		Gri Karşılığı 179	
Magenta (255,0,255)		Gri Karşılığı 105	
Sarı (255,255,0)		Gri Karşılığı 226	
K.rengi (158,85,54)		Gri Karşılığı 103	
Zeytin (155,160,52)		Gri Karşılığı 146	
Mor (100,0,150)		Gri Karşılığı 47	

Gri Seviyeli Görüntüden RGB Görüntüye Dönüşüm:

Birçok rengin gri seviyede karşılığı aynı olduğundan gri seviyeden RGB görüntüye dönüşüm tam olarak yapılamayabilir. Ama neticede gri seviyeli bir görüntüden RGB görüntüye dönüşüm mümkündür. Örneğin (100, 100, 100) gri seviyeli bir görüntü pikselinin RGB karşılığı Tablo 4-2'deki renklerden herhangi birisi olabilir.

Tablo 4-2 Gri seviyeden RGB' ye dönüşüm

	Gri Değer		RGB Değer	Dönüşüm Formülü
	100		0,170,0	Irgb(x,y) = [0.299(0)+0.587(170)+0.114(0)] =100
	100		230,53,0	Irgb(x,y) = [0.299(250)+0.587(53)+0.114(0)]=100
	100		80,83,240	Irgb(x,y) = [0.299(80)+0.587(83)+0.114(240)]=100
	100		230,14,200	Irgb(x,y) = [0.299(230)+0.587(14)+0.114(200)]=100

Bir gri seviyeli görüntüdeki her hangi bir f(x,y) pikseli için Igrey(x,y)=[0…255] aralığında değer alabilir [19] [20].

Aşağıda, RGB görüntünün gri seviyeli görüntüye dönüşümü 2 farklı yöntemle MATLAB'de yapılmıştır.

Klasik Yöntem:

```
i=imread('resim1.jpg');
[row col byt]=size(i);
R=i(:,:,1);          % kırmızı renkler
G=i(:,:,2);          % yeşil renkler
B=i(:,:,3);          % mavi renkler
R=double(R)
G=double(G)
B=double(B)
for x=1:1:row
   for y=1:1:col
exactgray(x,y)=0.3.*R(x,y)+0.59.*G(x,y)+0.11.*B(x,y);
```

```
    end
end
    figure; imshow(uint8(i))
     figure; imshow(uint8(exactgray))
```

MATLAB Yöntemi:

```
rgb2gray(i);   % MATLAB Image Processing Toolbox fonksiyonudur.
figure; imshow(i);
```

MATLAB yönteminde görüldüğü gibi tek bir komut satırı ile RGB görüntü gri seviyeli formata dönüştürülmüştür.

4.5. Siyah-Beyaz Görüntü (Monochrome, Binary Image) Formatı

Görüntüdeki her piksel ikilik 0 veya 1 değerinden birisini alır [21]. Görüntü piksellerindeki 0 siyah rengi, 1 ise beyaz rengi temsil eder.

d, 0-255 arası bizim belirlediğimiz bir eşik değer olmak üzere gri formatlı görüntüler binary formatlı görüntülere şu formül ile çevrilir.

Ibin(pixel) = 1, eğer Igrey(pixel) ≥ d

Ibin(pixel) = 0, eğer Igrey(pixel) < d

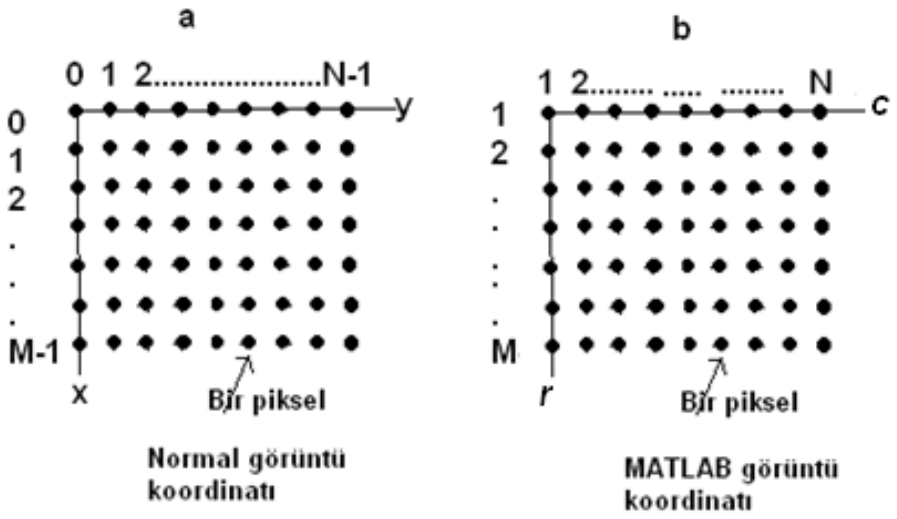
Aşağıda RGB görüntüyü binary görüntüye çeviren MATLAB kodları bulunmaktadır.

```
rgb=imread('resim1.jpg');   % image1.jpg görüntüsünü rgb değişkenine ata.
d=0.55;   % eşik değerini d değişkenine yaz.
bw=im2bw(rgb,d);   % görüntüyü eşik değerine göre binary görüntüye çevir.
imshow(bw);     % binary görüntüyü ekranda göster.
```

5. MATLAB İLE GÖRÜNTÜ İŞLEME

Bir sayısal görüntü, analog görüntünün örneklenmesi ve kuantalanması sonucunda elemanları reel sayılardan oluşan bir matris formunda ifade edilir. Yani f(x,y) şeklindeki bir sayısal görüntü, M satır N sütundan oluşmuş MxN elemanlı bir matristir.

Şekil 5-1 ve Şekil 5-2'de gösterildiği gibi [41] resmin x-y düzlemindeki matris ifadesi [M N] 'in ilk piksel numarası [0 0]'dan başlarken MATLAB'de [1 1]'den başlar. MATLAB'da çalışırken bu durum göze alınmalıdır.



Şekil 5-1 Görüntü piksellerinin a) Matris gösterimi b) MATLAB'de gösterimi

a

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \cdots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

b

$$f(x,y) = \begin{bmatrix} f(1,1) & f(1,2) \ldots \ldots f(1,N) \\ f(2,1) & f(2,2) \ldots \ldots f(2,N) \\ \vdots & \vdots & \vdots & \vdots \\ f(M,1) & f(M,2) & \ldots \ldots & f(M,N) \end{bmatrix}$$

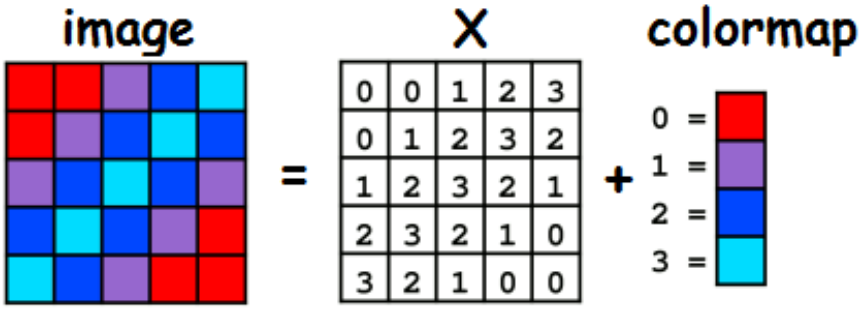
Şekil 5-2 a) Görüntüyü ifade eden matris formu b) MATLAB formu

MATLAB programının desteklediği görüntü formatları; indekslenmiş görüntü (indexed image), gri seviyeli görüntü (grayscale image), renkli görüntü (RGB image) ve siyah-beyaz görüntüdür (binary image).

5.1. Indekslenmiş Görüntü (Indexed Images)

İndekslenmiş görüntü, görüntüleri daha az hafıza alanında saklamak için geliştirilmiş bir formattır.

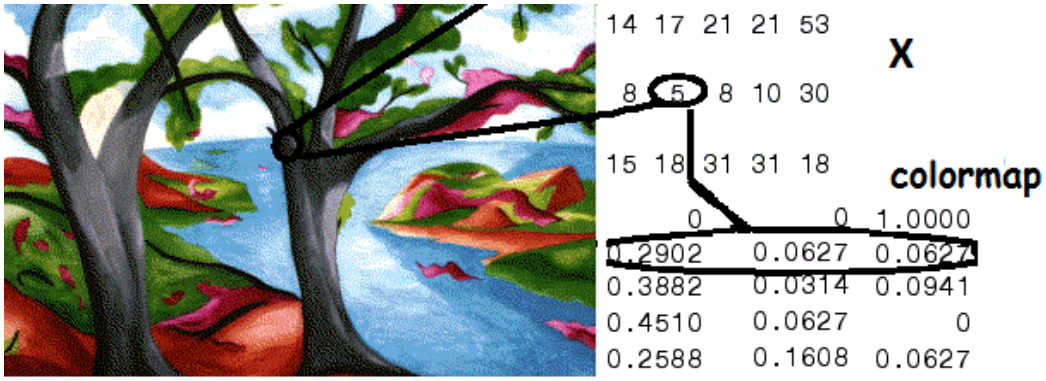
Örneğin RGB görüntüdeki herhangi bir piksel için ona ait üç değerden oluşan 24-bitlik veriyi değilde o renge ait sadece 8-bitlik bir değer ifade etmek ve bir alıcıya göndermek istiyoruz. Alıcının gelen renkleri çözümleyebilmesi için bir eşleştirme anahtarına (colormap) sahip olması gerekir ki örneğin 3 yolladığınız da turuncu demek istediğinizi bilebilsin. Yani indekslenmiş resimler, görüntü matrisi [X] ve 3 sütundan oluşan colormap matrisi [map]'dan oluşmaktadır.



Şekil 5-3 2-bit indeklenmiş görüntü

Şekil 5-3'te RGB görüntüden indekslenmiş görüntünün nasıl oluşturulduğu gösterilmiştir. 4 değişik renkten oluşan yukarıdaki RGB görüntüde kırmızı =0, mor= 1, mavi=2, turkuaz=3 ile ifade edilebilir. 4 çeşit renkten oluşan 5x5 matrisli RGB görüntümüzün hafızada kapladığı alan 5x5x(8bit+8bit+8bit)=6350 bit olup, RGB görüntüyü indekslediğimizde X görüntü matrisi için 25x2bit = 50 bit ve colormap matrisi için 4x8bit= 32 bit, toplamda 82 bitlik bir hafıza yeterli olmaktadır.

Şekil 5-4'te indekslenmiş görüntünün MATLAB'de X ve map matrisleri şeklinde nasıl ifade edildiği gösterilmiştir [22].



Şekil 5-4 MATLAB'de indekslenmiş görüntü örneği

Üstteki şekilde görüldüğü üzere **5** ile ifade edilen renk pikseli için **R=5*0.2902, G= 5*0.6027 ve B= 5* 0.0627** olmaktadır.

Daha önceden indekslenmemiş olan görüntüler,

[X, map] = imread('trees.tif'); komutu ile **X** ve **map** matrislerine atanır.

İndekslenmiş görüntüler tek bir değişkene atanabilir.

S=load('trees.tif');

Bu durumda S değişkeni, X ve Map matrislerin her ikisini de kapsamaktadır.

S=

X: [200x320 double]

Map:[81x3 double]

Caption :[2x1 char]

Gri seviyeli görüntülerin indekslenmiş görüntüye çevrilmesi:

```
I = imread('cameraman.tif');
[X, map] = gray2ind(I, 16);
imshow(X, map);
```

İndekslenmiş görüntülerin boyutlarının %50 küçültülmesi aşağıdaki gibidir.

```
X, map] = imread('trees.tif');
[Y, newmap] = imresize(X, map, 0.5);
imshow(Y, newmap)
```

5.2. Gri Seviyeli Görüntü (Grayscale Images)

Gri seviye görüntü matrisi elemanları ***uint8, uint16, int16, single*** veya ***double*** tipinde olabilirler.

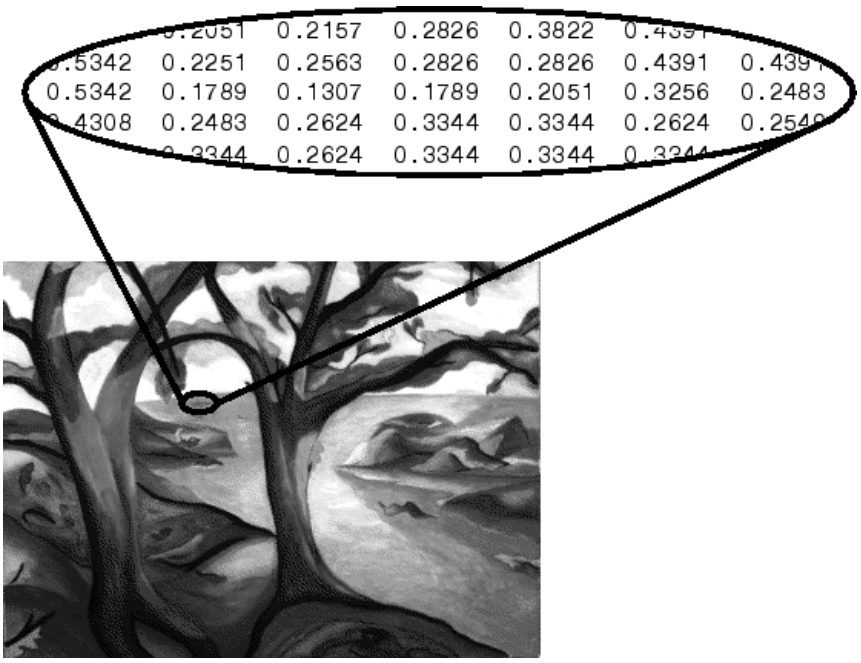
Gri seviyeli görüntü ***single*** veya ***double*** tipinde ise 0 değeri siyahı, 1 değeri beyazı belirtmek üzere görüntü matrisindeki tüm pikseller [0.0…1.0] aralığında sayılardır.

Eğer gri seviyeli görüntü matrisindeki piksel değerleri [0.0…1.0] aralığında yani ***single*** veya ***double*** ise [0…255] aralığında ***unit8*** (8 bitlik gri formatlı görüntü) tipine dönüştürmek için ***im = uint8(im * 256)*** komutu kullanılır.

MATLAB gri seviye görüntüleri ***uint8*** tipinde yüklemektedir. ***Unit8*** tipindeki gri seviyeli görüntüyü ***double*** tipinde gri seviyeli bir görüntüye çevirmek için ***im2double*** komutu kullanılmaktadır.

Diğer görüntü tiplerini ***unit8*** tipine dönüştürmek için ***im2uint8*** komutu kullanılmaktadır. Tüm görüntü dönüşüm komutları ve fonksiyonları için MATLAB programında yardım kısmına bakabilirsiniz.

Şekil 5-5'te ***double*** tipinde gri seviyeli görüntü örneği verilmiştir [22].

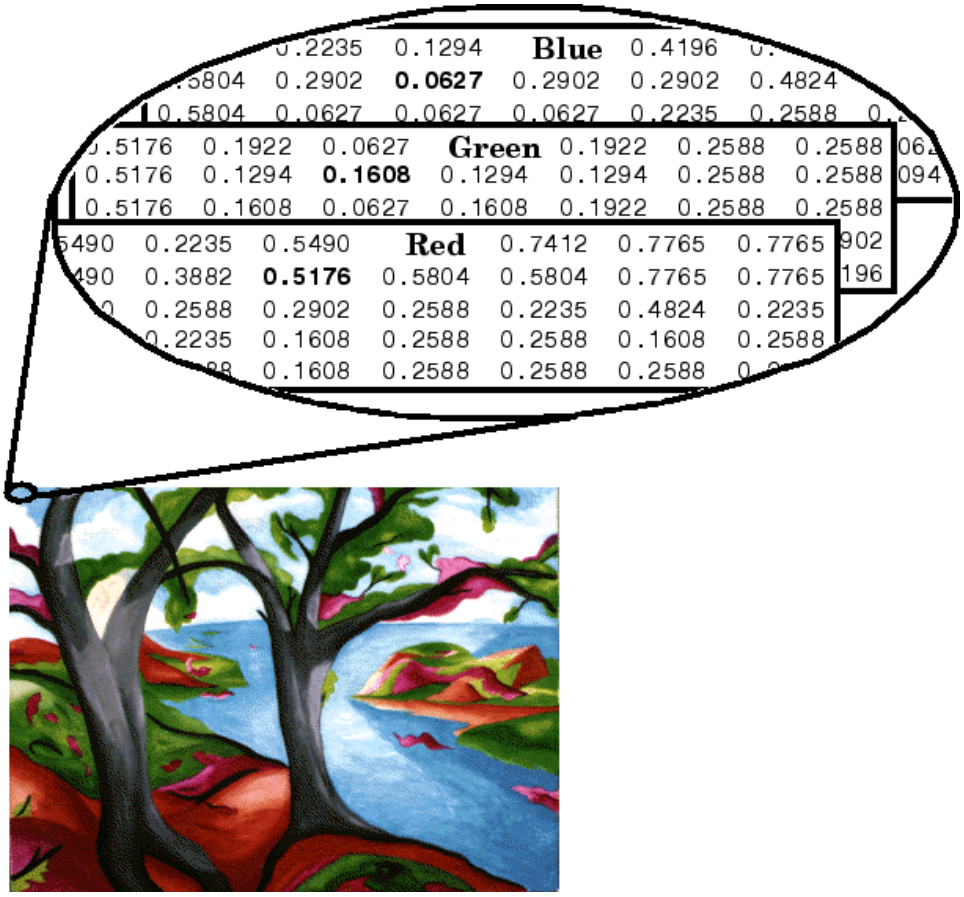


Şekil 5-5 MATLAB'de double tipinde gri seviyeli görüntü örneği

5.3. Renkli Görüntü (RGB Images)

MATLAB'de renkli görüntüler " RGB images" olarak ta adlandırılmaktadır ve görüntü boyutu MxN ise her pikselin değerini tutmak için MxNx3 oyutunda matris kullanılır. Örneğin (10,5) numaralı pikselin renk değerleri (10,5,1), (10,5,2) ve (10,5,3) numaralı matris hücrelerinde saklanır. MATLAB'de RGB görüntüler ***double, unit8*** veya ***unit16*** tipinde olabilir. ***Double*** tipi [0…1] aralığında olduğundan [0,0,0] siyah rengi [1,1,1] beyaz rengi göstermektedir.

Şekil 5-6' daki RGB görüntü ***double*** tipinde olup (2,3) noktasındaki pikselin R, G ve B değerleri için 3 ayrı matris tutulduğu görülecektir. Kırmızı renk değeri için 1. matris (2,3,1) ***0.5176*** değerini, yeşil renk değeri için 2.matris (2,3,2) ***0.1608*** değerini ve mavi renk değeri için 3. matris (2,3,3) ***0.0627*** değerini göstermektedir. Bu durumda (2,3) noktasındaki pikselin değeri ***(0.5176 0.1608 0.0627***) olmaktadır [22] .



Şekil 5-6 MATLAB'de RGB görüntü örneği

5.4. Siyah-Beyaz Görüntü (Binary Images)

Siyah-beyaz (binary) görüntü matrisinin tüm pikselleri 0 veya 1 rakamlarından oluşmaktadır. Bu yüzden bazen ikilik görüntü (logical images) olarakta adlandırılmaktadır.

Herhangi bir türdeki görüntüyü siyah beyaz görüntüye dönüştürmek için ***İm2bw(f,T)*** komutu kullanılır. Burada ***f*** dönüştürülecek gri seviye görüntü, T eşik değeri olup [0 1] aralığındadır.

```
f=imread('resim1.tif');
h= im2bw (f, 0.5)
imtool(h);
```

Herhangi bir türdeki resmi [0 1] aralığında double türünde resme dönüştürmek için ***g=mat2gray(A,[Amin Amax])*** komutu kullanılır.

```
f=imread('resim1.tif');
h=mat2gray(f,[150 155])
```

impixel Komutu;

Bu komut görüntüdeki herhangi bir piksel ile ilgili bilgi almamız için kullanılmaktadır.

Aşağıdaki örnekte ***c*** ve ***r*** ile belirlenen bölgedeki piksellerin değerler okunur**.**

```
RGB = imread('resim1.png');
c = [12 146 410];
r = [104 156 129];
pixels = impixel(RGB,c,r)
pixels =
    62    34    63
   166    54    60
    59    28    47
```

5.5. MATLAB'de Görüntüler Arası Format Dönüşümü

Tablo 5-1'de görüntüler arası format dönüşüm tablosu ve örnek MATLAB kodları [41] verilmiştir.

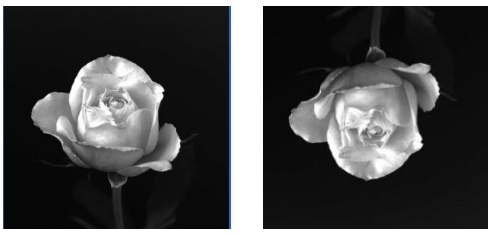
Tablo 5-1 Resim formatları arası dönüşüm MATLAB kodları

Görüntü Formatı	**Dönüştürülen Format**	**MATLAB Komutu**
RGB, Gri seviyeli, İndekslenmiş	Siyah-beyaz	im2bw(f,T)
RGB	Gri seviyeli	rgb2gray()
RGB	İndekslenmiş	rgb2ind()
İndekslenmiş	Gri seviyeli	ind2gray()
Gri seviyeli	İndekslenmiş	gray2ind()
İndekslenmiş	RGB	ind2rgb()
Düzgün Matris	Gri Seviyeli	mat2gray()

Aşağıda MATLAB'de görüntü işleme komutları ile bazı resimler üzerinde bazı işlemlerin nasıl yapıldığına ait örnekler [41] gösterilmiştir.

Bir görüntüyü yatay eksen etrafında ters çevirme:

```
f=imread('rose_512.tif');
imshow(f);
fp=f(end : -1:1, :);
imshow(fp);
```

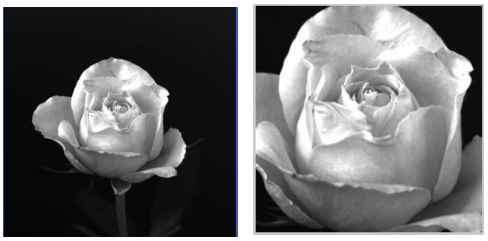


Bir görüntüyü dikey eksen etrafında ters çevirme:

```
f=imread('rose_512.tif');
imshow(f);
fp=f(:, end : -1:1);
imshow(fp);
```

Bir görüntüde odaklanmak:

```
f=imread('rose_512.tif');
imshow(f);
k = f(125:375, 125:375);
imshow(k);
```



Bir görüntünün boyutlarını küçültmek:

```
f=imread('rose_512.tif');
imshow(f);
z = f(1:4:end,1:4:end);
imshow(z);
```

5.6. MATLAB'de Veri Sınıfları (Data Classes) ve Dönüşümleri:

Tablo 5-2 MATLAB'de veri sınıfları [41]

Veri Tipi	***Açıklama***	***Alacağı Değer Aralığı***
logical	1-bit, binary	0 veya 1
unit8	8-bit, 0…255 işaretsiz tam sayılar	[0…255]
unit16	16-bit işaretsiz tam sayılar	[0…65536]
unit32	32-bit işaretsiz tam sayılar	[0…4294967295]
int8	8-bit işaretli tam sayılar	[-128…127]
int16	16-bit işaretli tam sayılar	[-32768…32767]
int32	32-bit işaretli tam sayılar	[-2147483648…2147483647]
single	32-bit floating point sayılar	[0.0…1.0]
double	64-bit floating point sayılar	[0.0…1.0]
char	16-bit karakter	[0…65536]

Tablo 5-3'te veri sınıfları arası dönüşüm yapan MATLAB komutları ve örnek MATLAB kodları ve açıklamaları [41] verilmiştir.

Tablo 5-3 MATLAB'de veri sınıfları arası dönüşüm tablosu

MATLAB Komutu	***Giriş Formatı***	***Çıkış Formatı***	***Örnek***
im2unit8	**binary, unit8, unit16, double**	**unit8**	**f=[-0.5 0.5; 0.75 1.5] ;** **g = im2uint8(f);** **g = [0 128: 191 255]** sonucu çıkar. Giriş görüntü matrisindeki 0'dan küçük değerler 0 olarak atanır. 1'den büyük değerler 255 olarak atanır. Diğer giriş değerleri ise 255 ile çarpılarak dönüşüm sağlanır.
im2unit16	**binary, unit8, unit16, double**	**unit16**	
mat2gray	**binary, unit8, unit16, double**	**[0,1] aralığında double**	**f=('resim1.tif');** **g=mat2gray(f[150 250]);** **mat2gray(A,[Amin Amax])** fonksiyonu, görüntüleri [0 1] aralığında **double** türünde matrise dönüştürür.
im2double	**binary, unit8, unit16, double**	**double**	**h=unit8([25 50;128 200]);** **g=im2double(h);** **g=[0.0980 0.1961 ; 0.4706 0.7843]** sonucu çıkar. Görüldüğü gibi **im2double** komutu 8-bitlik görüntünün piksel değerlerini 255'e bölerek **double** veri türüne dönüştürmektedir.
im2bw	**unit8, unit16, double**	**binary**	**f=('resim1.tif');** **g=im2bw(f,0.5);** **g=im2bw(f,T)** fonksiyonu T eşik değerinde siyah beyaz (binary) görüntüler oluşturur.

5.7. MATLAB'in Desteklediği Önemli Görüntü Formatları

Tablo 5-4'te MATLAB programının desteklediği resim formatları [41] mevcuttur.

Tablo 5-4 MATLAB'in desteklediği resim formatları

Format	**Açıklama**	**Uzantısı**
TIFF	Tagged Image File Format	**.tif, .tiff**
JPEG	Joint Photographic Expert Group	**.jpg, .jpeg**
GIF	Graphics Interchange Format	**.gif**
BMP	Windows Bitmap	**.bmp**
PNG	Portable Network Graphics	**.png**
XWD	X Window Dump	**.xwd**

6. NESNE TABANLI GÖRÜNTÜ İŞLEME YÖNTEMLERİ

6.1. Piksel Bazlı Görüntü İşleme (Pixel-based segmentation)

Gri seviye değerlerindeki benzerliklere göre görüntü işleme yöntemi "piksel based segmentation" olarak adlandırılır ve eşikleme (thresholding), büyütme (growing), ve bölme - birleştirme (split and merge) işlemlerine dayalı olarak gerçekleştirilir.

Görüntüden arka plan çıkarma işlemi bu metotla en iyi sonucu verir. Çünkü arka plan rengini bildiğimizden, eşik değerini doğru olarak seçtiğimizde kolayca arka plan ile nesneleri ayırt edebiliriz [23].

6.2. Bölge Bazlı Görüntü İşleme (Region-based segmentation)

Bölge bazlı görüntü işleme (Region-based segmentation) gri seviyeli görüntülerde komşuluk ilişkilerine göre çalışmaktadır.

Bölge bazlı görüntü işlemede her bölgenin piksellerinin diğerlerinden farklı olması onların köşe kenar tespiti için bize yardımcı olmaktadır. Algoritmaları karışık değildir. Kenar bulmanın zor olduğu görüntülerde bu yöntem uygulanabilir. Çünkü nesneler görüntüde bu yöntemle ayırt edilebilir. Bu yöntemde görüntü üzerindeki işlenecek bölge (ROI) seçilip sadece o bölge işlem yapmak [38] kolaydır. ROI için görüntüye binary maske uygulanır. Maskeleme sonucu 1 çıkan bölgeler ROI bölgesi olarak kabul edilip 0 çıkan bölgeler ise işlem yapılmayacak bölgeler olarak kabul edilir. Bu yöntemde aşağıdaki işlemler en çok kullanılanlardır.

-Bölge genişletme (region growing),

-Bölme ve birleştirme,

-watershed segmentation,

-Region of Interest (ROI)

-Filter a region (roifilt2)

-Fill in a region (roifill)

6.3. Kenar Bazlı Görüntü İşleme (Edge-based segmentation)

Bir gri seviyeli görüntüde gri değerlerdeki süreksizliklere ve ani değişiklere dayalı olarak görüntüdeki kenar ve ayrıntıların belirlenmesine ***kenar bazlı görüntü işleme*** (edge segmentation) denir.

Kenar belirleme, görüntü işlemede temel öneme sahip konulardan birisidir. Bir görüntüdeki kenar, aydınlatma veya yüzey yansımaları gibi bir görüntünün fiziksel görünüşünde oluşan önemli bir değişime karşı düşer ki bu değişim kendisini parlaklık, renk ve doku olarak gösterir. Bu anlamda, bir görüntünün gri seviyelerinde ani değişikliklerin olduğu bölgelere ***kenar*** adı verilecektir.

İyi bir kenar belirleme metodu;

- Kenarları iyi bir biçimde sezebilmelidir,
- Kenarları doğru yerlerinde belirleyebilmelidir,
- Bir kenar için sadece bir kenar çizgisi oluşturabilmeli yani yapay kenarlar üretmemelidir.

6.4. Türev Almaya Dayalı Kenar Belirleme Yöntemleri

Bir görüntü içerisindeki kenarları belirlemek için uygulanabilecek en verimli yöntemlerden birisi, ani gri seviye değişimlerini tespit etmektir. Bu amaç için birçok kenar belirleme yönteminin kullandığı temel yaklaşım, bölgesel türev hesabına dayanır. Bölgesel olarak, görüntünün 1. türevi kenar bölgelerinde en büyük değer olur (local maximum) ve görüntünün 2. türevi ise kenar bölgelerinde sıfır değerini üretir. Bölgesel olarak görüntüye ilişkin 1. ve 2. türevleri hesaplayarak elde edilen lokal maksimum ve sıfır geçiş noktaları ile ilgili görüntü bölgesi için kenarlar belirlenmiş olur.

1. türevi kullanan yönteme ***gradient yöntemi*** ve 2. türevi kullanan yönteme ***laplacian yöntemi*** denilmektedir. Bu yöntemleri kullanan en yaygın kenar belirleme algoritmaları: Roberts, Prewitt, Sobel, Canny kenar belirleme filtreleridir. [24]

6.4.1. Roberts Kenar Filtresi:

Türev almaya dayalı bir filtredir. Dört element kullanılır.

İki köşegen yönünde görüntüye uygulanır.

i, j	i+1, j
İ, j+1	i+1, j+1

$$BV_{i,j} = ((BV_{i,j} - BV_{i+1,J+1})^2 + (BV_{i+1,j} - BV_{i,\,j+1}))^{1/2}$$

X

0	0	0
0	1	0
0	0	-1

y

0	0	0
0	0	1
0	-1	0

6.4.2. Sobel Kenar Filtresi:

Türev almaya dayalı bir kenar bulma filtresidir. 3x3'lük pencere alanına uygulanır.

Eksenler üzerindeki piksellere daha çok ağırlık verir.

A	B	C
D	E	F
G	H	I

$$S = (X^2 + Y^2)^{1/2}$$
$$X = (C + 2F + I) - (A + 2D + G)$$
$$Y = (A + 2B + C) - (G + 2H + I)$$

X

-1	0	1
-2	0	2
-1	0	1

Y

1	2	1
0	0	0
-1	-2	-1

6.4.3. Prewitt Kenar Filtresi:

Türev almaya dayalı bir kenar bulma filtresidir. 3x3'lük pencere alanına uygulanır.

Dikey ve yatay yönlerde ayrı eğimleri hesaplar.

X

-1	0	1
-1	0	1
-1	0	1

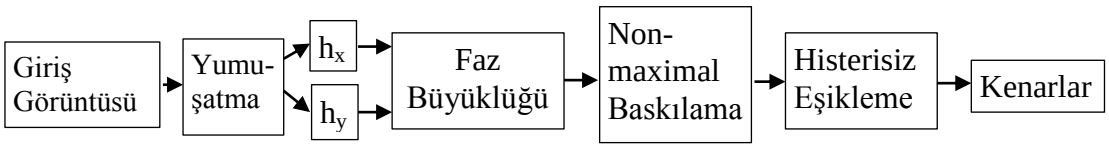
Y

-1	-1	-1
0	0	0
1	1	1

6.4.4. Canny Kenar Filtresi:

Kenar bulmada son derece etkin bir algoritmadır. Önce görüntüdeki gürültü bir σ değerine göre üretilen Gaussian çekirdekle konvolusyonu alınarak azaltılır. Daha sonra, gradyent operatörü uygulanarak, kenar gradyent büyüklüğü ve yönü hesaplanır. Kenarlar, non maxima baskılama uygulanarak inceltilir. Son olarak görüntü, ikili eşikleme uygulanarak istenmeyen ayrıntılardan arındırılır.

Şekil 6-1' de Canny kenar filtresi iş akış algoritması görülmektedir.



Şekil 6-1 Canny filtresi iş akış algoritması [25]

Canny kenar filtresinde görüntüye sırası ile aşağıdaki işlemler uygulanarak kenarların tespiti yapılır.

1. Görüntüdeki gürültü bir σ değerine göre üretilen Gaussian çekirdekle konvolusyonu alınarak azaltılırak yumuşatılır.

$$G(x,y) = \frac{1}{2\pi\sigma^2}\mathbf{e}^{-\frac{x^2+y^2}{2\sigma^2}} \quad (6\text{-}1)$$

2. Gradyent operatörü uygulanarak, kenar gradyent büyüklüğü G_n hesaplanır.

$$G_{\mathbf{n}} = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n}\cdot\nabla G \quad (6\text{-}2)$$

3. Gradyentin yönü n hesaplanır.

$$\mathbf{n} = \frac{\nabla(G*g)}{|\nabla(G*g)|} \quad (6\text{-}3)$$

4. Kenarlar, non-maxima baskılama uygulanarak inceltilir.

$$\frac{\partial^2}{\partial \mathbf{n}^2} G*g = 0 \quad (6\text{-}4)$$

5. Görüntüye histrisiz eşikleme uygulanarak istenmeyen kenarlar temizlenir. Tek piksel kalınlığında kenarlar üretilir ve kırık çizgileri birleştirir

Canny filtresinin diğer filtrelere göre üstünlükleri;

– Gürültüye karşı düşük duyarlılık,

– İyi kenar belirleme,

– Tek kenardaki birden çok karşılığı elemek,

Mohamed ve ark. "Edges detection in depth images for a gesture recognition application using a kinect WSN" adlı çalışmalarında Kinect for Windows kamerasından alınan derinlik görüntülerinde kenar belirleme için Robert, Sobel ve Canny filtrelerini uygulamış ve karanlık ortamlarda Sobel filtresinin, aydınlık ortamlarda ise Canny filtresinin daha iyi sonuçlar verdiğini görmüşlerdir. Bu nedenle tezde aydınlık ortamda çalışacağımızdan MATLAB'de yapacağımız görüntü işleme yazılımında kenar belirleme sırasında Canny filtresini uygulamaya karar verdik [26].

6.5. MATLAB'de Kenar Belirleme Komutları

Aşağıda MATLAB ortamında kullanılan kenar belirleme komutları gösterilmiştir.

```
BW = edge(I)
gpuarrayBW = edge(gpuarrayI)
BW = edge(I,'sobel')
BW = edge(I,'sobel',thresh)
BW = edge(I,'sobel',thresh,direction)
BW = edge(I,'sobel',...,options)
[BW,thresh] = edge(I,'sobel',...)
BW = edge(I,'prewitt')
BW = edge(I,'prewitt',thresh)
BW = edge(I,'prewitt',thresh,direction)
[BW,thresh] = edge(I,'prewitt',...)
BW = edge(I,'roberts')
BW = edge(I,'roberts',thresh)
BW = edge(I,'roberts',...,options)
[BW,thresh] = edge(I,'roberts',...)
BW = edge(I,'log')
```

BW = edge(I,'log',thresh)

BW = edge(I,'log',thresh,sigma)

[BW,thresh] = edge(I,'log',...)

BW = edge(I,'zerocross',thresh,h)

[BW,thresh] = edge(I,'zerocross',...)

BW = edge(I,'canny')

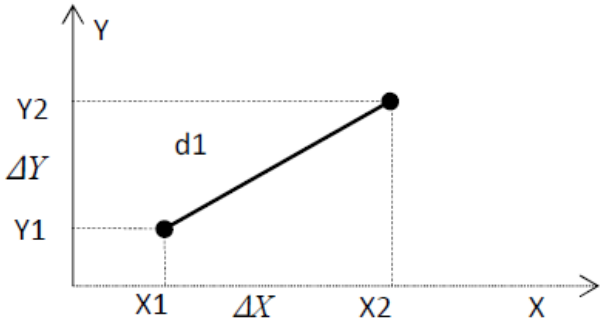
BW = edge(I,'canny',thresh)

BW = edge(I,'canny',thresh,sigma)

[BW,thresh] = edge(I,'canny',...)

6.6. Noktalar Arası Mesafe ve Açı Hesaplaması

X, Y uzaysal düzleminde iki nokta arasındaki uzaklığın nasıl hesaplanacağı aşağıda gösterilmiştir. Bu hesap bize köşe noktaları bilinen bir nesnenin kenar uzunluğunu hesaplamamızda yardımcı olacaktır.



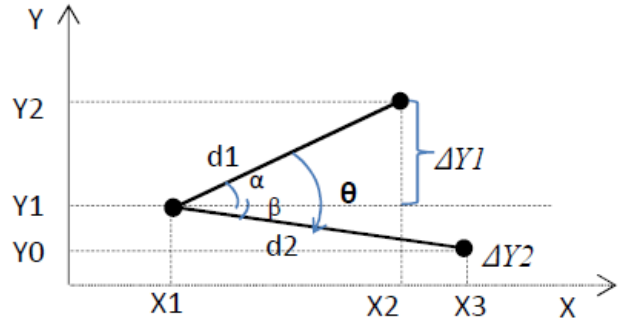
$$\Delta X = X2 - X1$$

$$\Delta Y = Y2 - Y1$$

$$d1 = \sqrt{\Delta X^2 + \Delta Y^2}$$

Şekil 6-2 İki nokta arasındaki mesafenin hesaplanması

X,Y koordinat düzleminde bilinen 3 noktanın oluşturduğu açı aşağıdaki şekilde hesaplanır.



$$\alpha = \sin^{-1}\left(\Delta Y1 / d1\right)$$

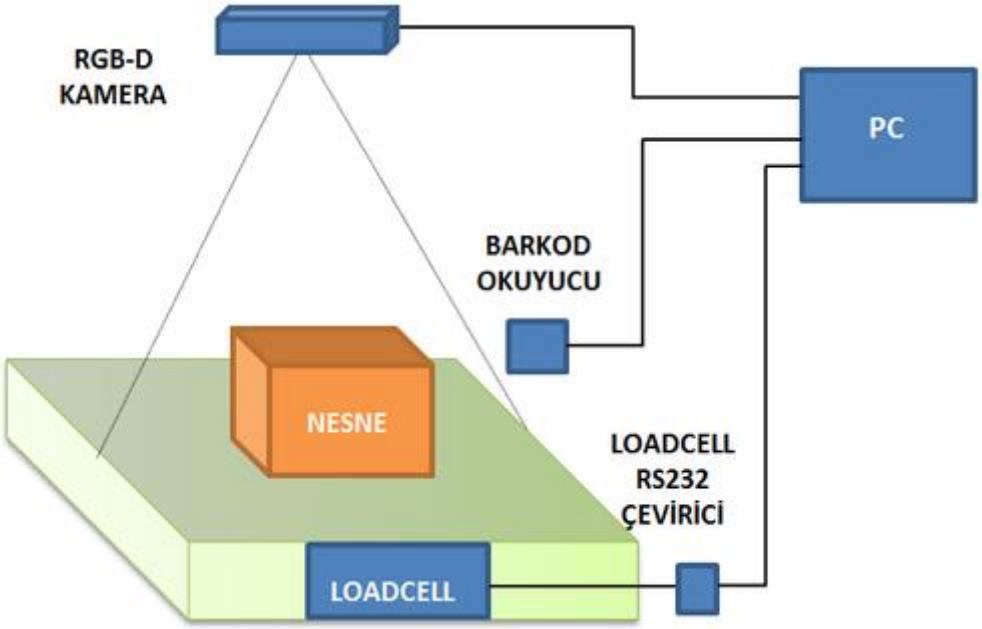
$$\beta = \sin^{-1}\left(\Delta Y2 / d2\right)$$

$$\theta = \alpha + \beta$$

Şekil 6-3 Üç nokta arasındaki açının hesaplanması

7. PROJEDE KULLANILAN DONANIMLAR ve ÖZELLİKLERİ

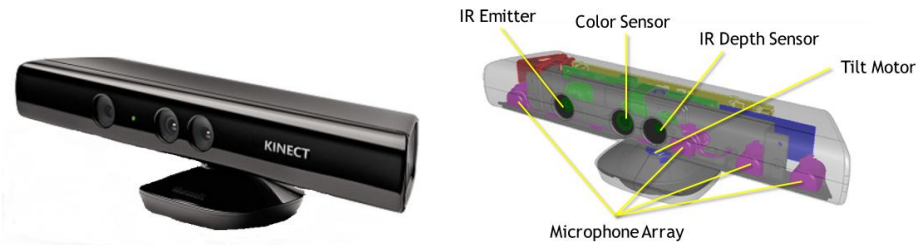
Projemizde donanım olarak 1 adet RGB-D kamera, 1 adet yük hücresi (loadcell), 1 adet ağırlık göstergesi+RS232 çevirici, 1 adet barkod okuyucu ve 1 adet PC kullanılacaktır. Kullanacağımız donanımlar Şekil 7-1'de gösterilmiştir.



Şekil 7-1 Ölçüm düzeneğimizin prensip şeması

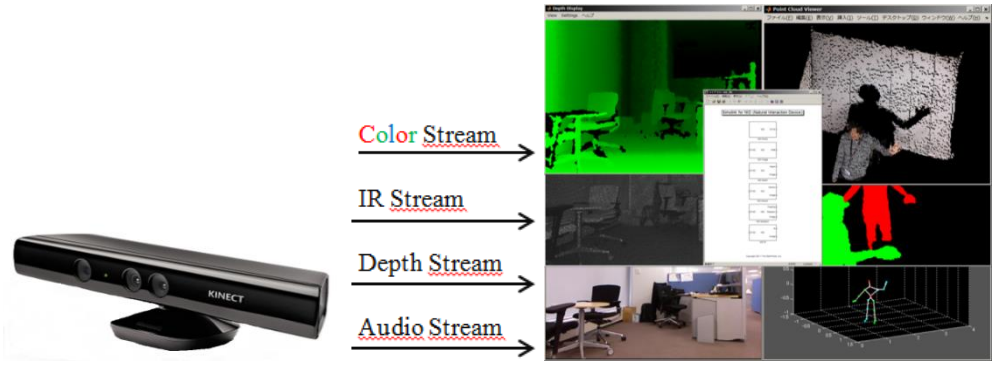
7.1. Kinect for Windows Kamera ve Veri Akışı (Data Streams)

Projemizde kullandığımız RGB-D kamera tümleşik yapıda olup aslında bir adet RGB kamera ve 1 adet derinlik (depth) kamerası ve 1 adet IR ışık kaynağının birleşmesinden oluşmuştur. RGB-D kamera olarak Microsoft firmasının piyasaya sunduğu Kinect for Windows V1 kamerası kullanılacaktır. Şekil 7-2'de bu kamera ve içyapısı gösterilmiştir.



Şekil 7-2 Microsoft Kinect for Windows V1 kamera ve içyapısı [27]

Bu kameraya ait tüm veri akışı (Data streams) Şekil 7-3'te gösterilmiştir.



Şekil 7-3 Kinect for Windows veri akışı (Data streams) [28]

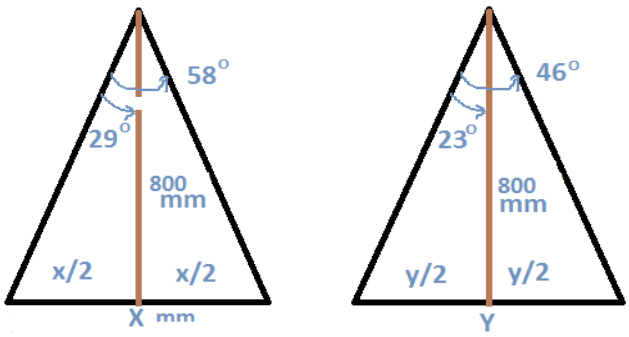
7.2. Renkli Görüntü Veri Akışı (Color Stream)

Kinect for Windows kamerasının içerisindeki RGB kamera tarafından alınan görüntü formatları aşağıda listelenmiştir.

- RGB_1280x960x12fps
- RGB_640x480x30fps
- RawYUV_640x480x15fps
- YUV_640x480x15fps
- Infrared_640x480x30 fps (16-bit/pixel (LSB 6-bit =0)) [29]

RGB Kamera En-Boy Çözünürlüğü:

Kinect for Windows V1 kamerasının yatay eksende görüş açısı 58^o, dikey eksende görüş açısı 46^o olduğundan ve ölçüm düzeneğimizde kameramızı zeminden 800 mm yükseklikte yerleştirdiğimizden renkli görüntü çözünürlüğünü aşağıdaki gibi hesaplayabiliriz.



Şekil 7-4 Kinect for Windows V1 RGB kamera görüş açıları

X=En = [800mm x tan 29]x2 = 886 mm ,

Y=Boy = [800 mm x tan 23]x2 = 679 mm,

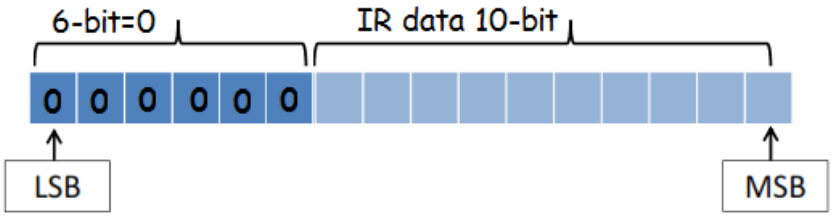
(886 x 679)/(1280 x 960) = 601594/1228800 = 0,49 mm/piksel.

Bu durumda 1280x960 piksel çözünürlükte RGB görüntünün 80 cm mesafeden x,y çözünürlüğü = 0,49 mm/pikseldir.

7.3. IR Görüntü Veri Akışı (IR Stream)

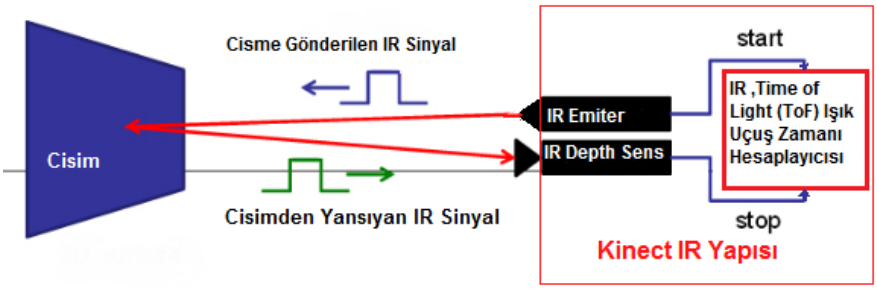
Kamera içerisinde bulunan IR ışık kaynağı, belli bir frekansta yapılandırılmış (Structured lights) infrared ışıkları yayar. Bu IR ışıklar cisimlere çarpıp geri yansır. Cisimlerden yansıyan bu IR ışıkların oluşturduğu IR görüntü RGB kameradan IR görüntü [40] olarak okunabilir. Kameramız dama tahtası şeklindeki bir test şekline tutulursa, RGB görüntü ve IR görüntülerin görüntü uzay haritalarının birbirine göre kalibrasyonları yapılabilir. IR görüntülerin ikinci kullanım alanı ise karanlık cisimlerin IR görüntüleri alınabilir.

IR data akış data formatı 640x480x30 fps, 16 bit olup en düşük değerli (LSB) 6-biti 0 dır. Diğer 10-bit ise IR verisidir. [30]



Şekil 7-5 IR veri yapısı

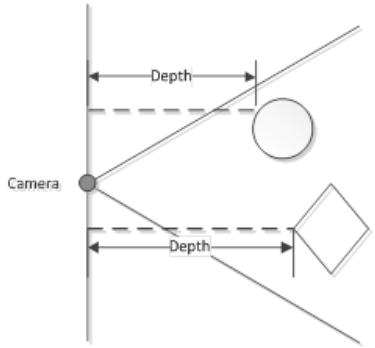
Şekil 7-6'da MS Kinect V1 kameraya ait IR görüntü prensibi gösterilmiştir.



Şekil 7-6 Kinect IR görüntü prensip şeması

7.4. Derinlik Görüntüsü Veri Akışı (Depth Stream)

Kinect for Windows 640x480x30fps, 320x240x30fps, 80x60x30fps gri formatta görüntü alma seçeneklerine sahiptir. Şekil 7-7'de görüldüğü gibi nesnenin kameraya olan uzaklığı izdüşüm olarak gönderilmektedir.

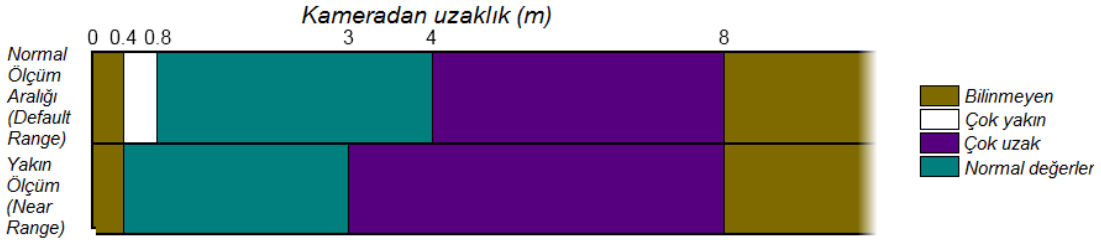


Şekil 7-7 Kinect V1 görüş alanındaki derinlik mesafesi

Kinect kamerada derinlik verisi aşağıdaki iki formattan birisi ile alınır.

1. Paketlenmiş Derinlik Verisi (Packed Depth Information):

Paketlenmiş derinlik verisi (Packed Depth Information) için normal ölçüm aralığı (default range) 80 cm-400 cm arası, yakın ölçüm aralığı (near range) 40 cm - 300 cm arasıdır.

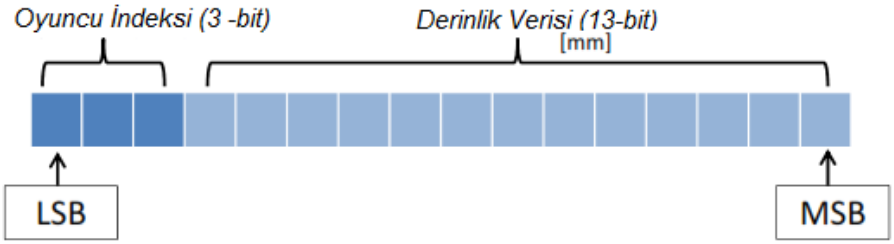


Şekil 7-8 Kinect V1 ölçüm aralıkları [31]

Bu güvenilir ölçme mesafesine ve FOV a giren nesneler için 16-bitlik derinlik verisi Şekil 7-9'da gösterilmiştir

İlk 3-bit (LSB) player (kişi) indeks bilgisidir. MS Kinect V1 kamera aynı anda 6 kişinin iskelet tanıma (skelation) takibini yapabilir ve bu 6 kişinin sadece kameraya en yakın olan 2'si üzerinde aktif iskelet tanıma işlemleri gerçekleştirebilmektedir. Diğer 4 kişinin iskelet tanınması pasif modda izlemede kalır. Bu özellik daha çok Kinect Xbox oyun cihazlarında 2 kişilik oyunların tek bir Kinect kamera ile oynanmasına olanak vermektedir.

Kamera görüş alanı (FOV) içerisinde kalan kameraya en yakın olan nesnenin her pikseli için (x,y) kordinatlarındaki uzaklığının (z) mm cinsinden derinlik verisini (Depth value), 13-bit grayscale veri formatında kullanıcıya iletir bu aralığın altında kalan ve kameraya çok yakın (too near) nesneler için 0x0000 verisi, bu aralığın üstünde kalan kameraya çok uzak , (too far) cisimler için 0x0FFF verisi, bu aralığın içerisinde tanınmayan (unkown) nesneler için 0x1FFF verisi kullanıcıya iletilir [32].



Şekil 7-9 Kinect V1 derinlik verisi [31]

2. Tam Derinlik Verisi (Full Depth Information):

Tam derinlik veri modunda çalışırken derinlik görüntüsü her piksel için 16-bit derinlik verisi ve 16-bit oyuncu indeksi (player index) olmak üzere 32-bit olarak gönderilir. Kameranın görüş alanı içerisine giren tüm cisimler uzaklık kısıtlaması olmaksızın 16-bit içerisinde kullanıcıya iletilir. Bilinmeyen veya algılanamayan pikseller 0 olarak kullanıcıya iletilir. Tam derinlik modunda veri, her pikselin zemin düzlemine göre kameraya olan uzaklığı (ground plane position) zaman etiketi (time stamp) ile beraber gönderilir.

Derinlik (Depth Stream) Çözünürlüğü: Biz tam derinlik veri modunda çalışacağımızdan derinlik verisini 13-bit olarak alacağız. Aşağıda 16-bit, 13-bit, 12-bit binary data örnekleri gösterilmiştir.

[0,1,1,0,0,0,1,0,0,0,1,1,1,0,0,0] - 16 bits number

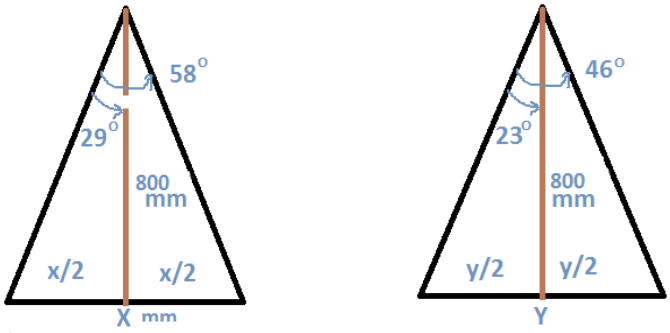
[0,1,1,0,0,0,1,0,0,0,1,1,1] - 13 bits number

[1,1,0,0,0,1,0,0,0,1,1,1] - 12 bits number

Bilinmeyen (unkown) 0x1FFF verisi hariç derinlik bilgisi 12-bit formatınadır. 12-bit ise onluk sistemde en fazla $2^{11} = 2048$ değerini alabilir.

Derinlik Görüntüsünde En-Boy Çözünürlüğü:

Derinlik görüntüsü için görüş alanı yatayda 58°, dikeyde 46° olduğuna göre, kameramız zeminden 800 mm yukarıya yerleştirildiğinde;



Şekil 7-10 Kamera X ve Y yönünde görüş alanı (FOV) açıları

X=En = [800mm x tan 29] x 2 = 886 mm,

Y=Boy = [800 mm x tan 23] x 2 = 679 mm,

(886 x 679) / (640 x 480) = 601594 / 307200 = 1,96 mm / piksel olur.

Bu durumda 640x480 çözünürlükte derinlik görüntüsünün 80 cm mesafeden x,y çözünürlüğü = 1,96 mm / pikseldir.

7.5. Kinect V1 Ses Verisi (Audio Data Stream)

MS Kinect V1 icerisinde 4 adet mikrofon vardır. Bu 4 mikrofon ile 24-bit formatta 1 ila 3 metre arasındaki sesleri sesin geldiği yer ve ses dalgasının yonunun net bir şekilde algılanabilmektedir [33].

7.6. Kinect for Windows V2 Kamera Özellikleri

Kinect V2 Aralık 2014 te Amerika Birleşik Devletleri' nde satışa sunulmuş olup henüz Türkiye'de satışı başlamamıştır.

7.7. Kinect for Windows V1 ve V2 Karşılaştırması

Tablo 7-1'de MS Kinect V1 kamera ile Kinect V2 kameranın karşılaştırılması gösterilmiştir.

Tablo 7-1 Kinect V1, Kinect V2 karşılaştırma tablosu

Özellik	**Kinect V1**	**Kinect V2**
RGB Color , fps	1280x960, 12 fps 640x480, 30 fps	1920x1080, 30 fps
YUV,fps Raw YUV , fps RawBayer, fps RawBayer, fps	640x480, 15 fps 640x480, 15 fps 1280x960, 12 fps, Mono8 640x480, 30 fps, Mono8	
Depth, fps	640x480, 30 fps 320x240, 30 fps 80x60, 30 fps	512x424, 30 fps
IR teknolojisi	Structured Light	ToF (Time of Flight)
IR ölçüm uzaklığı	500 – 4000 mm (Default) 400 – 3000 mm (Near Mode)	500 – 4500 mm (Default)
Görüş Açısı	46° dikey 58° yatay	60° dikey 89° yatay
Min. Depth res.	0.8 mm	0.5 mm
Max. Depth res.	4.0 mm	4.5 mm
Skelation Joints	20	25
Audio	4 speakers, 16 kHz, 24 bit Pulse Code Modulation (PCM)	4 speakers, 16 kHz, 32 bit IEEE float
USB	2.0	3.0
İşletim sistemi	Win7	Win8, Win8.1
InfraRed (Spreads Dots)	Var	Yok
Tilt Motor	Var	Yok
Vertical tilt range	± 27 derece	Yok

8. AĞIRLIK ÖLÇÜM SİTEMİ

8.1. Yük Hücresi (Loadcell)

Projemizde KELI firmasına ait 100 kg a kadar hassas ölçüm yapabilen UDB C3 model yük hücresi (loadcell) kullanılmıştır.

Şekil 8-1 Yük hücresi (Loadcell)

Aşağıda bu yük hücresine ait teknik özellikleri görülmektedir.

Tablo 8-1 Yük hücresi (loadcell) özellikleri

TEKNİK ÖZELLİKLER TECHNICAL PARAMETERS		**EXC + (Kırmızı,** Red) **EXC − (Siyah,** Black)	**SIG + (Yeşil,** Green) **SIG − (Beyaz,** White)
Kapasite Rated capacities (Emax)	**15, 20, 30, 35, 50, 60, 100, 150, 200, 300, 500 KG**	**Sıcaklık (Kompanse)** Temperature range, compensated	**-10°C ~ +40°C**
Hassasiyet Sensitivity	**2.0 ± 0.2 mV/V**	**Çalışma Sıcaklığı** Temperature range, operating	**-20°C ~ +50°C**
Creep Error (30 min)	**± 166 ppm**	**Maksimum Güvenli Yükleme** Maximum safe overload	**150 %F.S.**
Sıfır Dengesi Zero Balance	**± 3 %F.S.**	**Doğru Tartım Limiti** Ultimate overload	**300 %F.S.**
TCO (Temp. Effect on Min. Dead Load Output)	**± 0.03 %F.S./10°C**	**Tavsiye Edilen Akım** Excitation, recommend	**10 VDC**
TC SPAN (Temperature Effect on Sensitivity)	**+20°C~+40°C, ±17.5 ppm°C -10°C~+20°C, ±11.67 ppm°C**	**Maksimum Akım** Excitation, maximum	**15 VDC**
Giriş Direnci Input Impedance	**404 ± 15 Ω**	**Koruma Sınıfı** Protection Class	**IP65**
Çıkış Direnci Output Impedance	**350 ± 3 Ω**	**Gövde Malzemesi** Construction	Aluminium (Alüminyum)
İzolasyon Direnci Insulation Impedance	**≥ 2000 MΩ (50 VDC)**	**Kablo Çapı** Cable Diameter: **5 mm**	**Kablo Uzunluğu** Cable Length: **1.8 m**
Doğruluk Sınıfı Accuracy Class	**OIML C3**	**Maksimum Platform Ebatları** Max. Platform Size: **400 x 400 mm**	**Tavsiye Edilen Tork** Installation torque, recommended: **10 Nm**

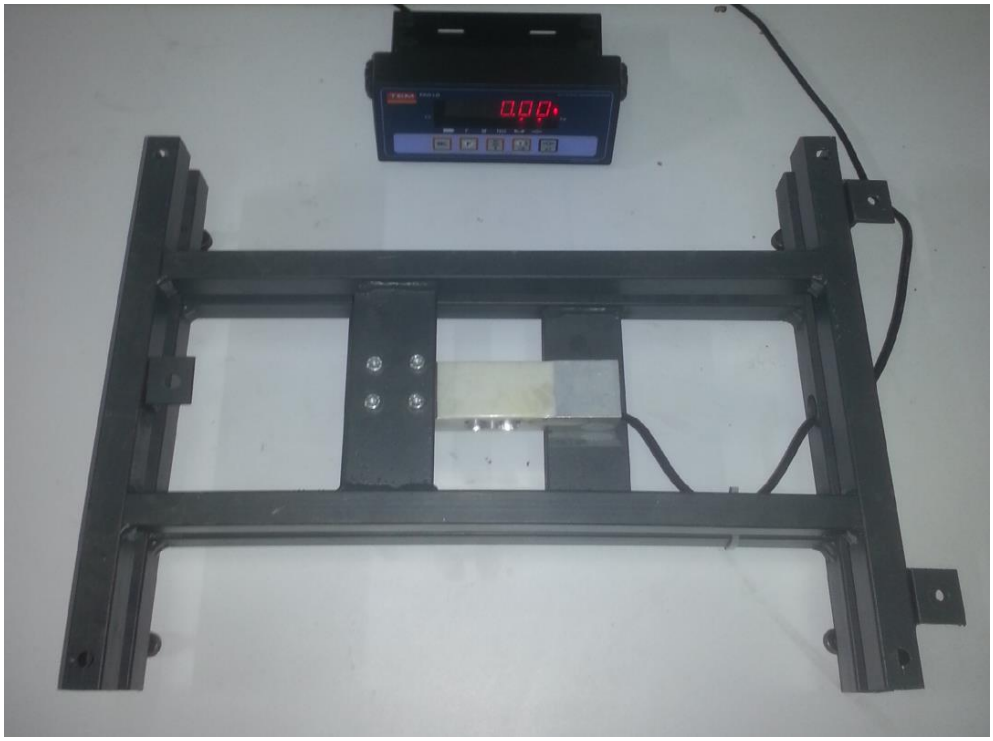
8.2. Ağırlık Göstergesi/RS232 Çevirici

Yine yük hücresinden gelen ağırlık verisini gösteren ve RS232 formatında PC'ye gönderen cihaz olarak TEM ELEKTRONIK firmasının EKO-LD model cihazı kullanılmıştır.

Şekil 8-2 Ağırlık göstergesinin önden görünüşü

Şekil 8-3 Ağırlık göstergesinin arkadan görünüşü

Aşağıda ise bir şasede toplanmış halde tartım sistemi görülmektedir.



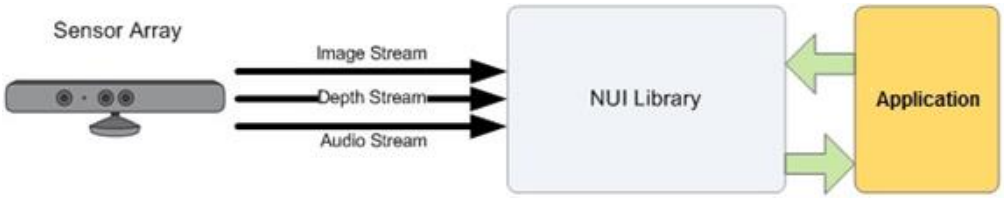
Şekil 8-4 Tartım sistemi

9. PROJEDE KULLANILAN YAZILIMLAR ve ÖZELLİKLERİ

9.1. Kinect SDK 1.7 - 64 bit

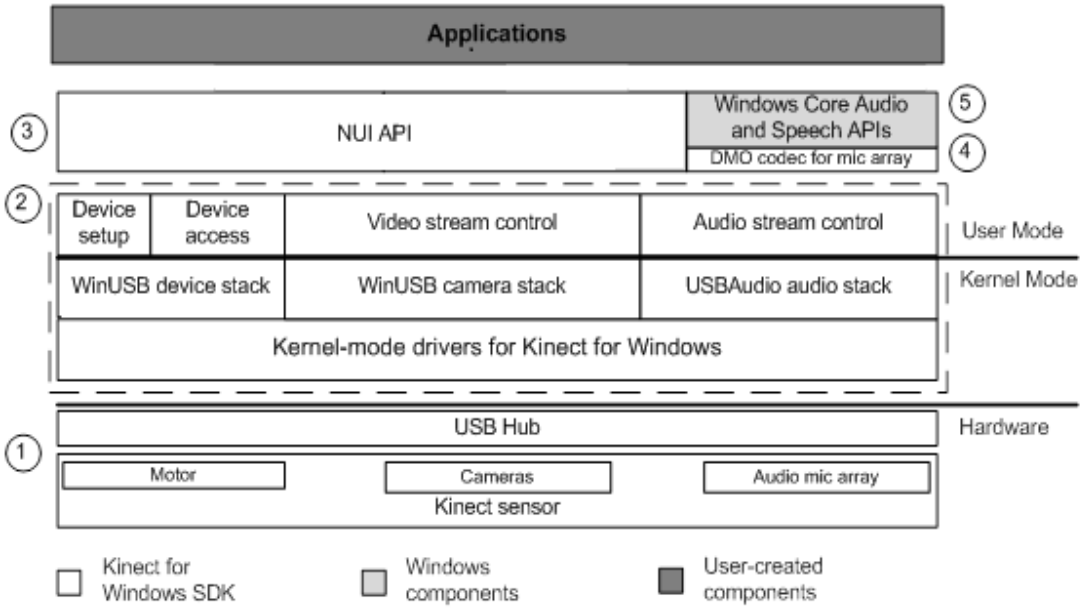
Kinect for Windows SDK 1.7, yazılımcılar ve geliştiriciler için Kinect tabanlı cihazlardan gelen gerçek dünya görüntülerinin bilgisayar ortamında işlenebilmesi için zengin ve kullanışlı kütüphaneler ve NUI'ler sunar. Kinect for Windows SDK V1.7, Windows7 uyumludur bu yüzden bilgisayarımızın işletim sistemi Windows7 olmalıdır.

Aşağıdaki şekilde Kinect for Windows V1 NUI Yapısı görülmektedir.



Şekil 9-1 Kinect for Windows V1 NUI arayüzü

Kinect for Windows SDK 1.7 yapısı aşağıdaki şekilde görülmektedir.



Şekil 9-2 Kinect V1 SDK yapısı [34]

9.2. Microsoft SDK 7.1 - 64bit

2009 yılında Microsoft firması tarafından Windows7 işletim sistemi için piyasaya sürülen Windows SDK for Windows7, uygulama geliştirmek isteyen

yazılımcılara yardımcı olması için .NET Framework 3.5SP1, C++ compiler ve kütüphanelerini (libraries), .h dosyalarını (header files), uygulama örneklerini (samples), dökümanlarını ve diğer bazı gerekli olan yazılım yardımcı araçlarını barındırmaktadır [35].

9.3. OpenNI 2.2 - 64 bit

OpenNI 2.2 PrimeSense ve Kinect uyumlu RGB-D kameralardan gelen derinlik görüntüsü, renkli görüntü ve IR video görüntülerini başlatmak, bilgisayara aktarmak ve işlemek için kütüphane dosyaları içerir [36].

9.4. MATLAB 8.2 - R2013b

MATLAB, temel olarak nümerik hesaplama, grafiksel veri gösterimi ve programlamayı içeren teknik ve bilimsel hesaplamalar için yazılmış yüksek performansa sahip bir yazılımdır. MATLAB programının tipik kullanım alanları; matematiksel hesaplama işlemleri, algoritma geliştirme, modelleme, simülasyon, veri analizi ve görsel efektlerle destekli gösterim, bilimsel ve mühendislik grafikleri gibi uygulama geliştirme alanları olarak özetlenebilir.

MATLAB adı, MATrix LABoratory kelimelerinden gelir. MATLAB ilk olarak Fortran Linpack ve Eispack projeleriyle geliştirilen ve bu programlara daha etkin ve kolay erişim sağlamak amacıyla 1970'lerin sonlarında yazılmıştır. İlk başlarda bilim adamlarına problemlerin çözümüne matris temelli teknikleri kullanarak yardımcı olmaktaydı. Bugün ise geliştirilen yerleşik kütüphaneleri ve uygulama ve programlama özellikleri ile gerek üniversite ortamlarında (başta matematik ve mühendislik olmak üzere tüm bilim dallarında) gerekse sanayi çevresinde yüksek verimli araştırma, geliştirme ve analiz aracı olarak yaygın bir kullanım alanı bulmuştur. Ayrıca işaret işleme, kontrol, fuzzy, sinir ağları, wavelet analiz gibi bir çok alanda ortaya koyduğu Toolbox adı verilen yardımcı alt programlarla da özelleştirilmiş ve kolaylaştırılmış imkanlar sağlamış ve sağlamaya da devam etmektedir.

9.5. MATLAB Image Processing Toolbox

Image Processing Toolbox, görüntü işlemeyle ilgili tüm fonksiyon ve komutları içinde barındıran MATLAB araç kutusudur. Görüntü işleme üzerinde çalışma

yapanlar için uzun kodlar sınıflandırılmış ve düzenlemiş, bu konuda çalışanlara çok büyük kolaylık sağlanmıştır. Bu araçKutusuna MATLAB Help, Product Help, Image Processing Toolbox yolu takip edilerek veya internet üzerinden www.mathworks.com adresinden ulaşılabilir.

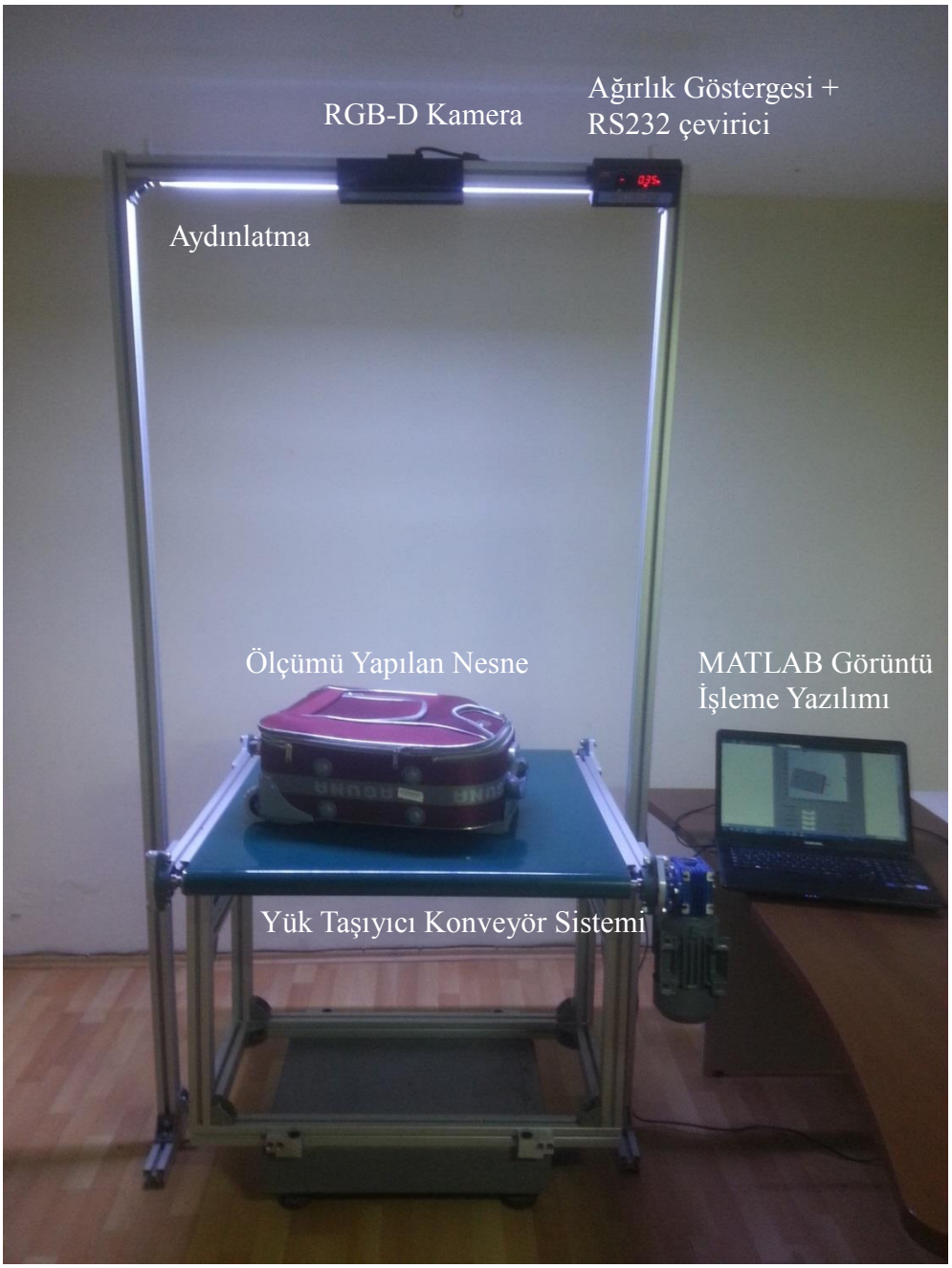
Tüm MATLAB Image Processing Toolbox komutları ve açıklamaları Ek-2 de verilmiştir.

9.6. MATLAB Image Aquation Toolbox

Image Acquisition Toolbox, içerdiği adaptörler ve yazılımlar sayesinde görüntü işleme alanında en çok kullanılan endüstriyel ve bilimsel kameralar, RGB-D kameralar gibi görüntü toplama cihazları ile haberleşme amacı ile kullanılır. Bu toolbox'ı kurmazsak harici görüntü alma cihazlarından görüntü alıp [44] Image Proccessing Toolbox'ta işleyemeyiz.

10.DONANIMLARIN BİR ARAYA GETİRİLMESİ

Projemizde kullandığımız donanımların bir araya getirlimiş hali aşağıda görülmektedir.



Şekil 10-1 Proje donanımlarının bir araya getirilmiş hali

11.MATLAB PROGRAMI ALGORİTMASI

BAŞLA

RGB-D kamerayı başlat

```
SAMPLE_XML_PATH='SamplesConfig.xml';
KinectHandles=mxNiCreateContext(SAMPLE_XML_PATH);
```

Barkod okuyucuyudan oku, GUI'de göster

```
uicontrol(handles.edit1_Barkod);
```

RGB görüntüyü oku, GUI'de göster

```
I=mxNiPhoto(KinectHandles);
I=permute(I,[3 2 1]);
      axes(handles.axes2);

imshow(I);
axis off;
```

Kutu şeklini GUI'de göster

```
axes(handles.axes3);
R=imread('kutu12.png');
imshow(R);
axis off;
```

Derinlik görüntüsünü oku

```
D=mxNiDepth(KinectHandles);
```

Derinlik görüntüsünde en ortadaki pikselin yerini bul

```
D=permute(D,[2 1]);
[m,n]=size(D);
mesafe = double(D(m/2,n/2));
```

Kamera ile nesnenin arasındaki mesafeyi bul

```
mesafe = double(D(m/2,n/2));
```

Görüntüde ilgilendiğmiz alanı (ROI) yi belirle

```
D=D(:,35:end-35)
```

Derinlik görüntüsünde en yüksek noktayı bul

```
D=max(max(D))-D;
 [m,n]=find(D==max(max(D)));
```

Görüntüdeki [0,0] başlangıcını MATLAB için [1,1] den başlat

```
ko=length(m);
for i1=1:ko  D(m(i1),n(i1))=0;
end
```

Derinlik görüntüsünü netleştir, Median filitre uygula

```
D= imadjust(D); % intensity değerlerini keskinleştir.
D=medfilt2(D,[3 3]);  %  [3 3] median filtre uygula
D=medfilt2(D,[5 5]);  % [5 5] median filtre uygula parazitleri gider
```

Derinlik görüntüsünü GUI'de göster

```
axes(handles.axes1);
imshow(D);
axis off;
```

Derinlik görüntüsünde köşeleri tespit et

```
h_zemin=990;
[cout,mimage,] = corner1(D,[],[],[],0.2);  % Corner1 fonksiyonu
```

Derinlik görüntüsünde köşeleri ve kenarları tespit et

```
boyut=[dist(cout(1,:),cout(2,:)') dist(cout(2,:),cout(3,:)')
dist(cout(3,:),cout(4,:)') dist(cout(4,:),cout(1,:)')];
boyut1=(boyut(1)+boyut(3))/2;
boyut2=(boyut(2)+boyut(4))/2;
```

```
Kenar uzunluklarını hesap et 1
if boyut1<boyut2
gercek_boyut1=(boyut1*mesafe)/2787;
gercek_boyut2=(boyut2*mesafe)/2837;
end
```

```
Kenar uzunluklarını hesap et 2
if boyut2<boyut1
gercek_boyut2=(boyut1*mesafe)/2837;
gercek_boyut1=(boyut2*mesafe)/2787;
 end
```

```
Nesne yüksekliğini hesapla
z=(h_zemin-mesafe)/10
```

```
Nesne hacmini hesapla
Hacim= gercek_boyut2*gercek_boyut1*z/100;
```

```
GUI'de yazdırılacak verilerin formatlarını ayarla
x2= sprintf('%3.2f',gercek_boyut2);
y2= sprintf('%3.2f',gercek_boyut1);
z2= sprintf('%3.2f',z);
Hacim2=sprintf('%3.2f',Hacim);
```

```
En, boy, yükseklik ve hacim bilgilerini GUI'de göster

set(handles.text1_x, 'String', x2);
set(handles.text1_y, 'String', y2);
set(handles.text1_z, 'String', z2);
set(handles.text1_Hacim, 'String',Hacim2);
```

```
Kamera h_zemin değerinden yükseksekteyse ikaz sesi ver
if mesafe<h_zemin
load train.mat;
 sound(y,3*Fs)
 end
```

Seri portu loadcell çeviriciyiden ağırlık bilgisini oku

```
s=serial('com4');
fopen(s);
gr = fscanf(s);
```

Ağırlık bilgisini GUI'de göster portu kapat

```
set(handles.text1_Agirlik, 'String',gr);
pause(2);
fclose(s);
```

Hacimsel ağırlığı hesapla GUI'de göster

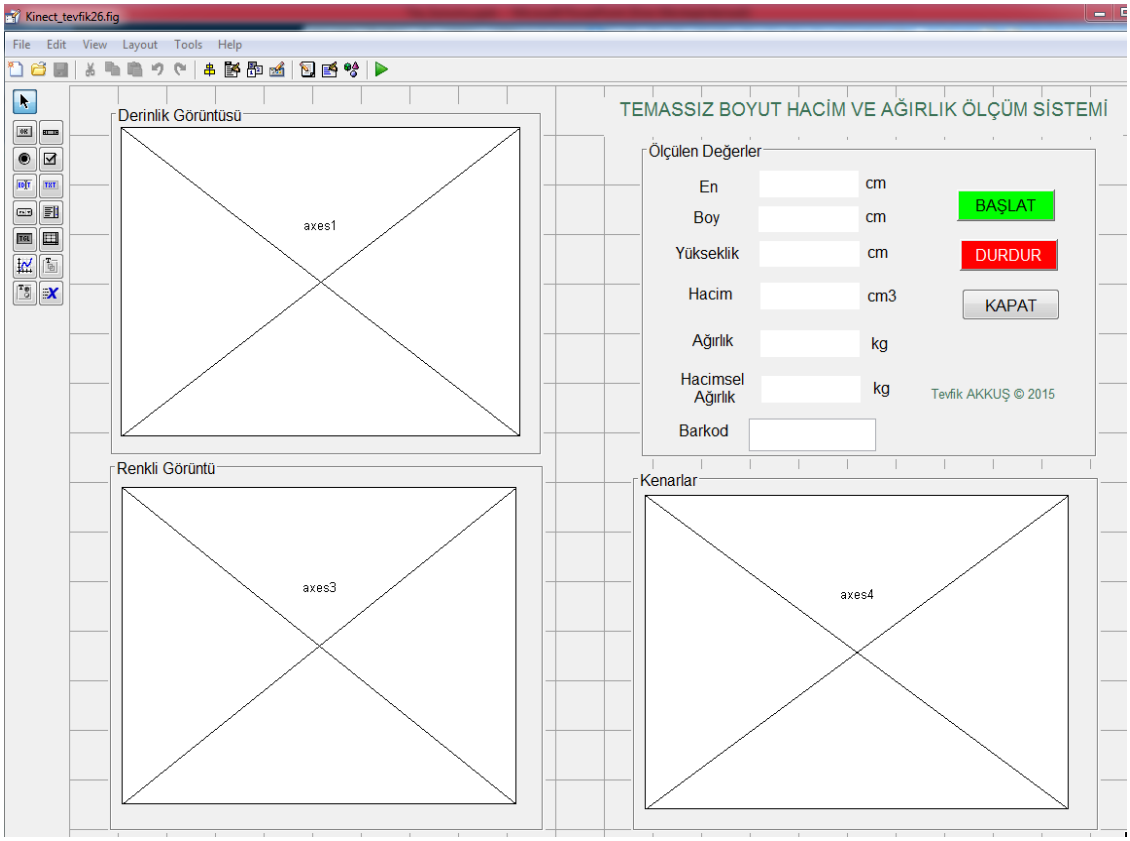
```
Ha=Hacim/5000 ;

set(handles.text6_ha, 'String', Ha);
```

BİTİR

12.MATLAB'de GUI ARA YÜZÜ OLUŞTURMA

MATLAB'de GUI arayüzümüz aşağıdaki gibi oluşturulmuştur. Sol taraftaki ilk pencerede RGB-D kameradan gelen derinlik görüntüsü, hemen altındaki pencerede renkli görüntü, onun yanındaki pencerede görüntüye ait kenarlar ve üst sağ kısımda yer alan pencerede ise tüm ölçülen ve hesaplanan değerler gösterilmiştir. Nesneye ait en, boy, yükseklik, hacim, ağırlık, hacimsel ağırlık ve barkod bilgisi bu alanda kullanıcıya eş zamanlı olarak gösterilmiştir.



Şekil 12-1 MATLAB'de hazırlanan GUI arayüz

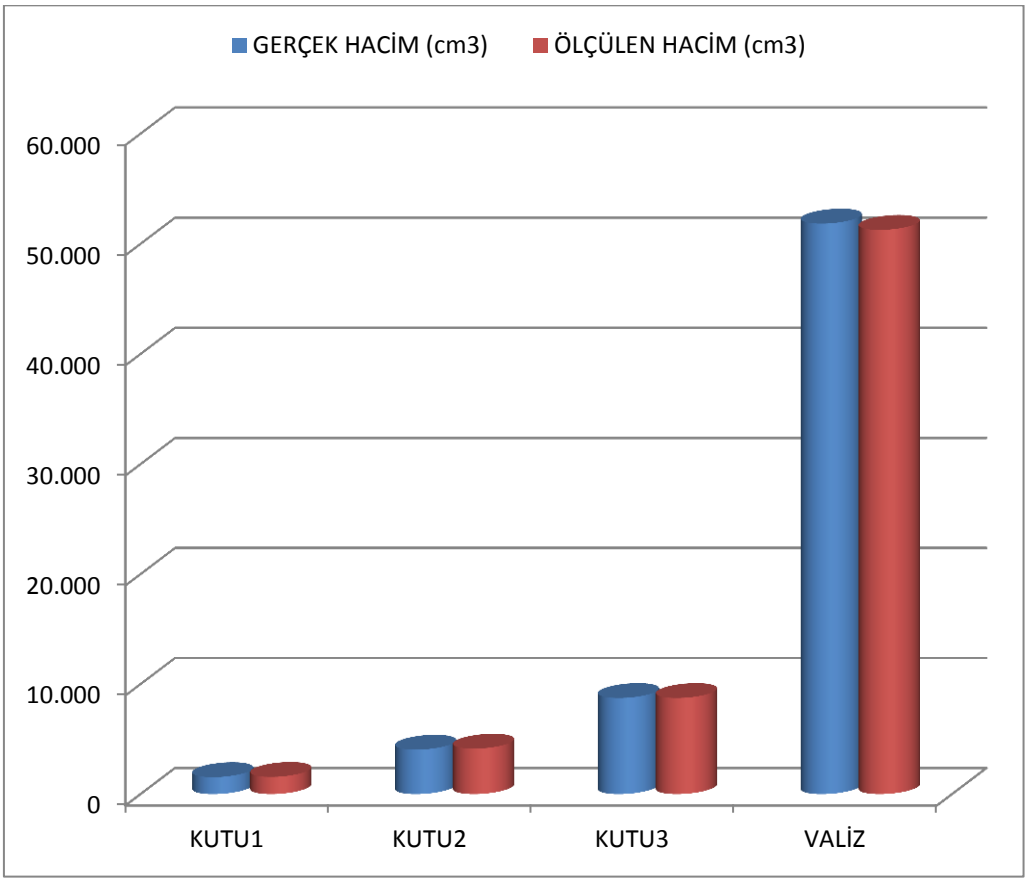
13.BULGULAR

"Temassız Hacim ve Ağırlık Ölçme Sistemimiz" ile 3 değişik boyuta ve ağırlığa sahip kutunun ve 1 adet valizin ölçümünü yaptık. Elde ettiğimiz sonuçlar Tablo 13-1'de gösterilmiştir.

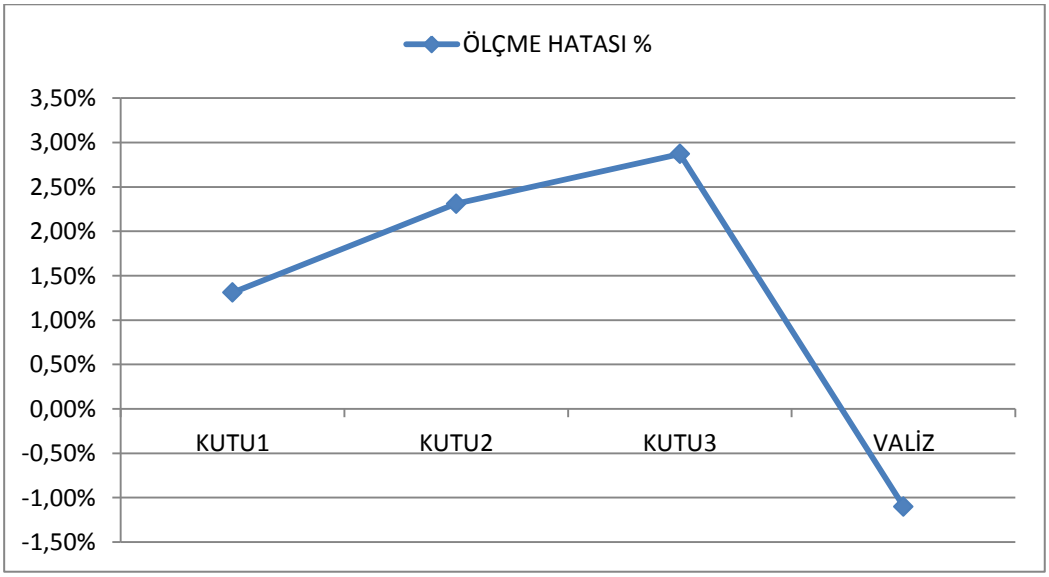
Tablo 13-1 Ölçülen nesneler ve sonuçlar

S.No	Kutular ve Gerçek Ölçüleri	Kutu Fotoğrafı	Ölçülen Hacim ve Ağırlık	Hata Oranları
1	24,5 x 10,5 x 6,0 cm 1.543 cm^3 209 gr		1.563 cm^3 210 gr	% 1,31 % 0,31
2	24,5x16,5x10,0 cm 4.042 cm^3 1.102 gr		4.135 cm^3 1.106 gr	% 2.31 % 0,32
3	27,5x22,0x14,0 cm 8.740 cm^3 3.002 gr		8.993 cm^3 3.013 gr	%2.87 % 0,35
4	54,0x40,0x24,0 cm 51.840 cm^3 4.400 gr		51.270 cm^3 4.413 gr	%-1,10 %0,30

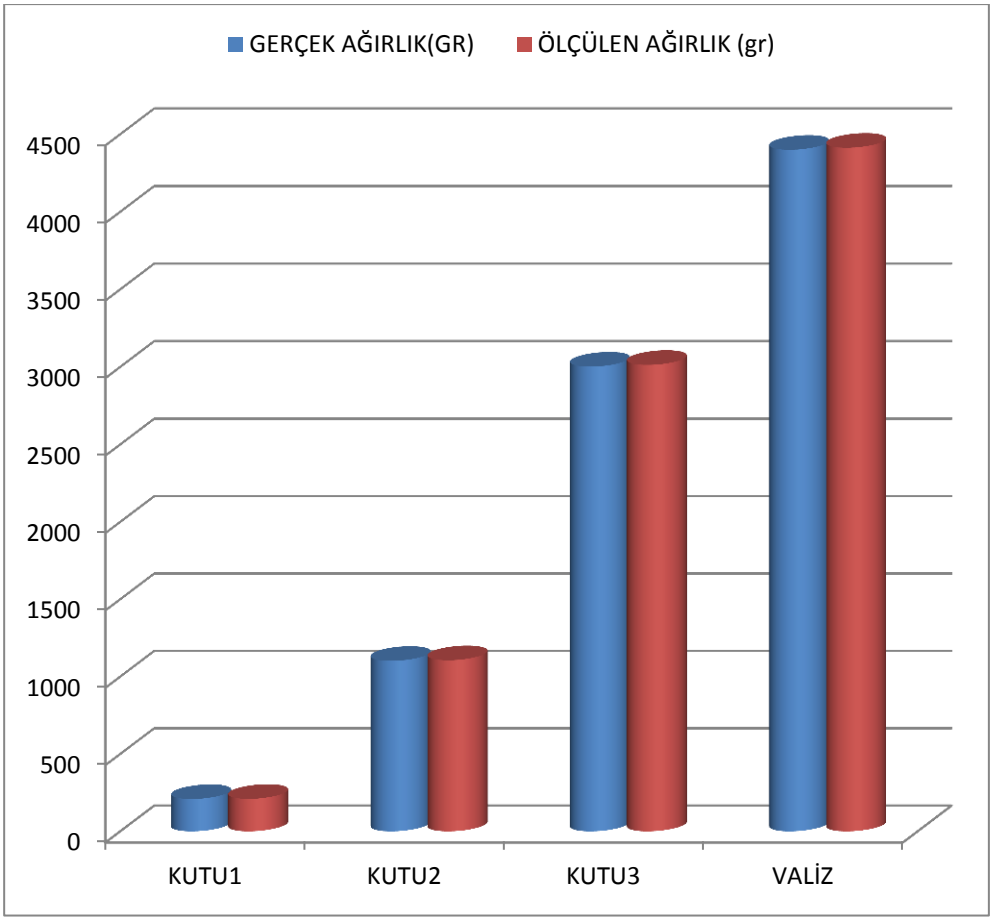
Bu ölçüm sonuçlarına göre haim, ağırlık ve ölçüm doğruluğunu gösterir grafikler aşağıdadır.



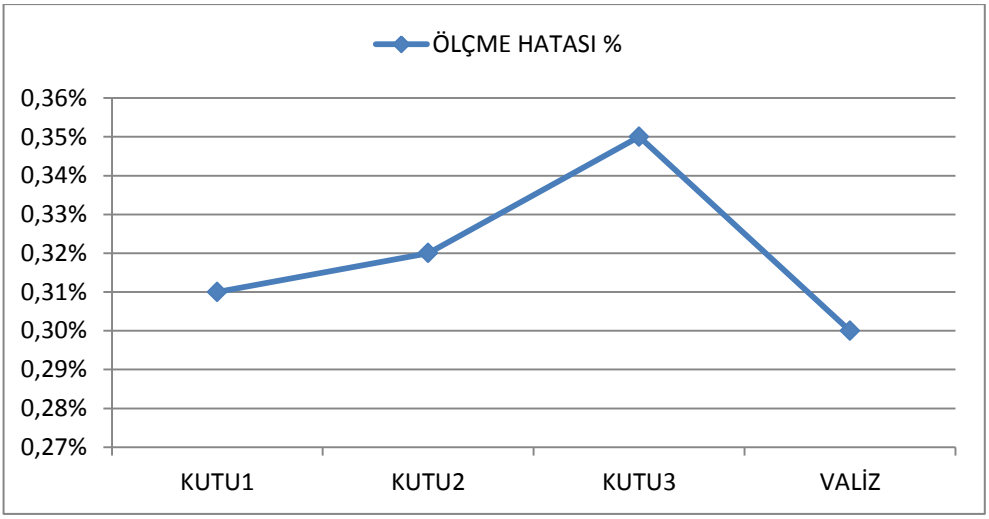
Şekil 13-1 Hacim grafiği



Şekil 13-2 Hacim ölçümü hata grafiği



Şekil 13-3 Ağırlık grafiği



Şekil 13-4 Ağırlık ölçümü hata grafiği

14.TARTIŞMA ve SONUÇ

Bu proje ile hava limanlarında yolcuların el bagajları ve kargo paketlerinin boyut, hacim ve ağırlıklarının RGB-D kamera kullanılarak temassız olarak ölçümü hedeflenmiş ve proje bu konuda derinleştirilmiştir. RGB-D kameraların çalışma prensipleri cisimden aldığı derinlik ve renkli görüntü yapıları bir deneysel ölçüm düzeneği oluşturularak incelenmiş, derinlik ve renkli görüntüler MATLAB ortamında yazılan kod vasıtasıyla işlenmiştir. Görüntü işleme yöntemleriyle nesnelerin köşe noktaları ve kenarları tespit edilmiş ve bu bilgiler kullanılarak kutuların yüzey alanları hesaplanmıştır. Derinlik görüntüsünden ise nesnenin yükseklik bilgisi yine MATLAB ortamında hesaplanmıştır. Yüzey alanı bilgisi ve yükseklik bilgisinden hacim bilgisine ulaşılmıştır. Nesnenin ağırlık bilgisinin tespiti için yük hücresinden (loadcell) gelen sinyal ağırlık göstergesine girilmiş ve ağırlık bilgisi dijital ekranda görülmüştür. Ağırlık göstergesi arkasında bulunan seri haberleşme portundan bu ağırlık bilgisi RS232 formatında PC'ye aktarılmıştır. MATLAB ortamında bu veri alınarak işlenmiş ve nesnenin gerçek ağırlık bilgisi MATLAB GUI ekranında gösterilmiştir. Kullanıcı dostu olarak tasarlanan MATLAB GUI ekranında kameradan alınan derinlik görüntüsü ve renkli görüntü gösterilmiş aynı zamanda nesnenin kenar görüntüleri derinlik görüntüsü üzerinde çizdirilmiş ve her bir kenar çizgisi üzerinde o kenara ait uzunluk değeri yazdırılmıştır. Nesneye ait boyutların çarpımından hacim bilgisi hesaplanmış ve hacim değerinin 5000 değerine bölünmesi sonucu elde edilen değer GUI' deki hacimsel ağırlık bölümünde gösterilmiştir. Aynı zamanda bu bilgilere ilave olarak eğer nesne üzerinde barkod varsa bu barkodu okuyabilmek için bilgisayara bir USB barkod okuyucu bağlanmış ve barkod okuyucudan okunan barkod bilgisi MATLAB GUI ekranında gösterilmiştir. Böylece nesneye ait tüm bilgiler bu barkod numarası ile ilişkilendirilmiştir.

Problemin tespiti ve yaptığımız bu çalışmayı bundan önce yapılan diğer çalışmalardan üstün kılan yönler aşağıda sıralanmıştır.

14.1. Problemin Tespiti

1. Havalimanlarında yolcu bagaj ve kargolarının boyut, hacim ve ağırlık ölçümlerinin halen geleneksel metotlarla (Şerit metre ve kantar kullanılarak) yapılması.
2. Bazen hatalı ölçümlerin yapılması, hatalı ücretlendirme ve kargaşa.
3. Yolcuların ölçümler sırasında çok fazla beklemeleri ve bazen uçuşu kaçırma riskinin oluşması.
4. Personelin ölçüm sırasında zaman kaybetmesi ve yoğun iş temposu.
5. Ölçümlerin veri tabanına girilmemesi.

14.2. RGB-D Kamera ve Yük Hücresi Kullanılarak Elde Edilen Çözüm ve Üstünlükleri

1. RGB-D kamera kullanılarak temassız boyut ve hacim ölçümü ie ağırlık ölçümünü birlikte gerçekleştiren ilk çalışmadır.
2. Ölçümü yapılacak nesnenin veya kameranın hareket ettirilmesi gerekmez.
3. Lazer ışık kaynağı veya projektör gerektirmez.
4. Hem duran hem de konveyör üzerinde hareket eden nesnelerin boyut, hacim ve ağırlık ölçümünü yapabilir.
5. Ölçümü yapılacak nesnenin kamera altındaki duruş pozisyonu önemli değildir. Kamera görüş alanına giren nesneler rahatlıkla ölçülebilmektedir.
6. Çok hızlı ve doğru ölçümler yapmaktadır.
7. Ölçümü yapılan nesnenin üzerinde barkod etiketi varsa, sistemimizde bulunan el tipi barkod okuyucu cihaz ile bu etiket okunabilmektedir.
8. Ölçümü yapılan nesnenin fotoğrafı çekilerek ölçüm ve barkod bilgisi ile birlikte veri tabanına kaydedilebilmektedir.
9. Yolcu bagajının kaybolması, karışması veya hasar görmesi durumunda veri tabanından bu bilgiler çağrılarak yolcu bagajı ile ilgili yukarıda bahsedilen sorunlar daha kolay çözüme kavuşturulabilecektir.

KAYNAKÇA

[1] M. Sarı, "Maviş Kameralı Kontrol Projeleri," 14 July 2014. [Online]. Available: http://www.mavis.com.tr/blog/?p=2201. [Accessed 13 Jan. 2015].

[2] T. Radil, J. Fischer and J. Kuera , "Dimension Measurement of Objects With CircularCross Section Using Point Light Sources and an Image Sensor Without Lens," IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, vol. 56, no. 4, p. 1403, Aug. 2007.

[3] D. Jelinek and C. Taylor, "Reconstruction of linearly parameterized models from single images with a camera of unknown focal length," Pattern Analysis and Machine Intelligence, IEEE Transactions, vol. 23, no. 7, pp. 767-773, Jul. 2001.

[4] L. A. F. Fernandes, M. M. Oliveira, R. da Silva and G. J. Crespo, "A Fast and Accurate Approach for Computing the Dimensions of Boxes from Single Perspective Images*," Journal of the Brezilian Computer Society, 2006.

[5] C. C. Chen and J. K. Aggarwal, "Recognition of Box-like Objects by Fusing Cues of Shape and Edges," in Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, Tempa,FL, 8-11 Dec. 2008.

[6] R. Lloyd and S. McCloskey, "Recognition of 3D package shapes for single camera metrology," in Applications of Computer Vision (WACV), 2014 IEEE Winter Conference, 24-26 Mar. 2014.

[7] D. J. Lee, X. Xu, J. Eifert and P. Zhan, "Area and volume measurements of objects with irregular shapes using multiple silhouettes," Optical Engineering, vol. 45, no. 2, p. 45, 03 Feb. 2006.

[8] A. B. Koç, "Determination of watermelon volume using ellipsoid pproximation and image processing," vol. 45, pp. 366-371, 11 Mar. 2007.

[9] M. Rashid, M. Gholam and S. Abbasi, "Cantaloupe Volume Determination through Image Processing," J. Agr. Sci. Tech., vol. 11, pp. 623-631, 2009.

[10] M. Khojastehnazhand, M. Omid and A. Tabatabaeef, "Determination of orange volume and surface area using image processing technique," vol. 23, pp. 237-

242, 2009.

[11] J. Siswantoro, A. S. Prabuwono and A. Abdullah, "Volume Measurement Algorithm for Food Product with Irregular Shape using Computer Vision Based on Momte carlo Method," Journal of ICT Research and Applications, vol. 8, no. 2337-5787, pp. 1-17, 28 Feb. 2014.

[12] P. A. W. Malvin H. Kalos, Monte Carlo Methods second,Revised and Enlarged Edition, VILEY-VCH Verlag GmbH, 2008.

[13] S. M. Goni, E. Purlis and V. O. Salvador, "Three-dimensional reconstruction of irregular foodstuffs," Journal of Food Engineering, vol. 82, pp. 536-547, 21 Mar. 2007.

[14] W. Wang and C. Li, "Size estimation of sweet onions using consumer-grade RGB-depth sensor," Journal of Food Engineering, vol. 142, pp. 153-162, Jun. 2014.

[15] A. Carreira, R. Ventura and J. Gaspar, "Volumetrics - Measuring free volumes," Instituto Superior Technico, Lisbon Portugal, 2013.

[16] S. Clarkson, J. Wheat and S. Choppin, "Assessing the Suitability of the Microsoft Kinect for Calculating Person Specific Parameters," in 4th IEEE Workshop on Consumer Depth Cameras for Computer Vision, Zurich, 2014.

[17] CCD_CMOS_Farklari, "http://www.neutron.com.tr," 05 Jun. 2014. [Online]. Available: http://www.neutron.com.tr/haber/ccd-ve-cmos-sensor-arasindaki-farklar-nedir/210. [Accessed 04 Jan. 2015].

[18] M. Özel, "http://www.canonturk.com," Haziran 2010. [Online]. Available: http://www.canonturk.com/profesyonel-seri/3826-cmos-sensor-mu-ccd-sensor-mu.html. [Accessed 05 Jul. 2015].

[19] S. Savarese, "Electrical and Computer Engineering Division/ University of Michigan," 2009. [Online]. Available: http://web.eecs.umich.edu/~silvio/teaching/EECS556_2009/lectures/lecture1.pdf. [Accessed 28 Dec. 2014].

[20] "http://www.had2know.com/technology/hsi-rgb-color-converter-equations.html," [Online]. [Accessed 1 Dec. 2014].

[21] K. Grauman, "http://www.cs.utexas.edu/~grauman/courses/378/slides/lecture3_full.pdf," [Online]. [Accessed 01 Dec. 2014].

[22] Mathworks.com, "http://www.mathworks.com/help/matlab/creating_plots/image-types.html?refresh=true#f2-155," [Online]. [Accessed 12 Dec. 2014].

[23] A. Şengür , İ. Türkoğlu and M. C. İnce, "A Comparative Study On Entropic Thresholding Methods," Journal of Electrical & Electronics Engineering, vol. 6, no. JOURNAL OF ELECTRICAL & ELECTRONICS ENGINEERING, pp. 183-188, 2006.

[24] A. Kızılkaya, "http://akizilkaya.pamukkale.edu.tr/B%C3%B6l%C3%BCm4_goruntu_isleme.pdf," [Online]. [Accessed 01 Dec. 2014].

[25] J. Canny, "A Computational Approach of Edge Dedection," IEEE Transactions on Pattern Analysis and Machine Intelegience, Vols. PAMI-8, no. 6, Nov. 1986.

[26] A. B. H. Mohamed, A. Baghdadi, T. Val, L. Andrieux and A. Kachour, "Edges detection in depth images for a gesture recognition application using a Kinect WSN," in Conference on web and Information Technologies, Hammaet, Tunusia, 09-12 May 2013.

[27] Kinect_Yapısı, "http://msdn.microsoft.com/en-us/library/jj131033.aspx," [Online]. [Accessed 27 Dec. 2014].

[28] Data_Streams, "http://msdn.microsoft.com/en-us/library/hh973075.aspx," Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/hh973075.aspx. [Accessed 27 Dec. 2014].

[29] Color_Stream, "http://msdn.microsoft.com/en-us/library/jj131027.aspx," [Online]. [Accessed 27 Dec. 2014].

[30] IR_Stream, "http://msdn.microsoft.com/en-us/library/jj663793.aspx," Microsoft. [Online]. [Accessed 27 Dec. 2014].

[31] Depth_Ranges, "http://msdn.microsoft.com/en-

us/library/hh973078.aspx#Depth_Ranges," Microsoft. [Online]. [Accessed 27 Dec. 2014].

[32] Depth_Stream, "http://msdn.microsoft.com/en-us/library/jj131028.aspx," Microsoft. [Online]. [Accessed 27 Dec. 2014].

[33] Audio_Stream, "http://msdn.microsoft.com/en-us/library/jj131026.aspx," Microsoft. [Online]. [Accessed 27 Dec. 2014].

[34] Kinect_SDK_Yapısı, "http://msdn.microsoft.com/en-us/library/jj131023.aspx," [Online]. [Accessed 27 Dec. 2014].

[35] SDK7, "http://www.microsoft.com/en-us/download/details.aspx?id=3138," [Online]. [Accessed 27 Nov. 2014].

[36] OpenNI2, "OpenNI_Programmers_Guide," 2015. [Online]. Available: http://com.occipital.openni.s3.amazonaws.com/OpenNI_Programmers_Guide.pdf. [Accessed 05 Jan. 2015].

[37] S. Ya-Lin and B. Chen-Xi, "Research and Analysis of Image Processing Technologies Based on DotNet Framework," in 2012 International Conference on Solid State Devices and Materials Science, Macao, 2012.

[38] P. . I. Santosa, T. B. Adji and R. Y. Tara, "Hand Segmentation from Depth Image using Anthropometric Approach in Natural Interface Development," vol. 3, 2012.

[39] L. Qian and N. Jinping, "One-dimension light screen scanning method of measuring the volume of the flying objects" in Electronic Measurement & Instruments (ICEMI), 2011 10th International Conference, 16-19 Aug.2011.

[40] Y. J. Xu, C. Chen, S. J. Huang and Z. H. Zhang, "Simultaneously measuring 3D shape and colour texture of moving bjects using IR and colour fringe projection techniques," Optics and Lasers in Engineering, vol. 61, pp. 1-7, 10 Jan. 2014.

[41] İ. Aydın, "web.firat.edu.tr," 10 Dec. 2014. [Online]. Available: http://web.firat.edu.tr/iaydin/bmu357/Bolum_1.pdf.

[42] HSI, "http://en.wikipedia.org/wiki/HSL_and_HSV," [Online]. [Accessed 11 Dec. 2014].

[43] D. G. Lowe, "Thee-Dimensional Object Recognition from Single Two-Dimensional Images," Artificial Intelligence, vol. 3, pp. 355-395, Mar. 1987.

[44] mathworks, "http://www.mathworks.com/help/imaq/acquiring-from-the-color-and-depth-devices-simultaneously.html," [Online]. [Accessed 02 Dec. 2014].

EKLER

EK-1 .NET IMAGE PROCCESSING

1. Digital Image Processing Technologies Based on .Net

There are five methods about image processing based on .Net [37].

(1) GetPixel/SetPixel method;

(2) Memory array method;

(3) Unsafe pointer method;

(4) Windows Presentation Foundation WPF method;

(5) Direct 3D method.

Moreover, the Graphics class of GDI+ library may be used to operate graphic. The above five methods of the image processing technologies of the advantages and disadvantages will be compared in this paper using image inverse transformation which is one of the most common algorithms and the transformation formula is:

R2 = 255 - R1

G2 = 255 - G1

B2 = 255 - B1 (1)

Among them, R2, G2, B2 were the processed red color value, the processed blue color value and the processed green value respectively. R1G1B1 were color values in Picture source.

1.1 GetPixel/SetPixel Method

The GetPixel/SetPixel method is the simple and direct in image processing using .net. The Bitmap class in .Net Framework provided these functions to implement and the class can load many Picture format such as BMP, JPG, Gif, Tiff, Png etc.

The main steps of processing:

1) Load image to save to the Bitmap object or get the image object of pictureBox, which can convert to the Bitmap object;

2) traverse all pixels of image and get the color value of each pixel using Getpixel(x,y) of Bitmap. As color is a structure in .Net framework, the red value, the blue value and the green value can get easily through calling the variant of the structure;

3) calculate and get the color value of the current point (x,y) using formula (1); set the new color value of the point(x,y) through SetPixel.

The main source codes:

```
Bitmap bmpIn = new Bitmap("E:\\test.bmp"); //Load picture
for (int i = 0; i < bmpIn.Width; i++)
for (int j = 0; j < bmpIn.Height; j++){
color cl = bmpIn.GetPixel(i, j); //get the color value of the current pixel point
int r =255 - cl.R; //get Red value
int g =255 - cl.G; // get Green value
int b =255 - cl.B; // get Blue value
bmpIn.SetPixel(i, j,Color.FromArgb(r,g,b)); //set the color of the specified pixel
}
```

The method can be implemented easily and has the satisfied result and cannot reach the real-time demand because of its poor efficiency.

1.2 Memory Array Method

The array method has the different processing strategy from the above, and improves executing efficiency greatly. The main idea of memory array is that load pixels of image to memory, which will be save as array, then do some operation of processing, and take these processed data into the final image files. The BitmaData class need be used here.

The main steps are:

1) Load image to save to the Bitmap object or get the image object of PictureBox, which can convert to the Bitmap object;

2) Instance BitmapData class, and lock Bitmap object to system memory through LockBits() of Bitmap object;

3) Copy the data of image to array using Marshal.Copy() in BitmapData object;

4) Process the data in array;

5) copy the processed data in array to new image;

6) Unlock the Bitmap object using UnlockBit().

The main source codes:

```
Bitmap bmp = new Bitmap("E:\\test.bmp");
Rectangle rect = new Rectangle( 0 ,0 , bmp.Width , bmp.Height);
BitmapData bmpData = bmp.LockBits(rect, ImageLockMode.ReadWrite ,
bmp.PixelFormat);
IntPtr ptr = bmpData.Scan0; //gets the address of the first pixel data in the bitmp
int bytes = bmp.Width * bmp.Height * 3;
byte[] rgbValues = new byte[bytes]; //declare an array of byte
System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues,0,bytes); //copy
data
for(int count=0;count<rgbValues.Length; count ++)
rgbValues[count] = (byte)(255 - rgbValues[count]); //inversing operation
System.Runtime.InteropServices.Marshal.Copy
(rgbValues,0,ptr,bytes);
bmp.UnlockBits(bmpData);
```

GetPixel/SetPixel method was not used because of the low efficiency, then the method improved the processing speed greatly.

1.3 Unsafe pointer Method

The operation speed of array method above had been improved greatly, but the efficiency of processing is not satisfied when raising complexity of operating algorithm or using bigger size picture. We need a high-performance program of image processing as well as C++ program, which processed picture using memory pointer that can operate data of memory directly so the performance efficiency is improved remarkablely. C# language of .Net the memory pointer was not used generally, but it will be used in some specific conditions. In order to correctly use this method the current project should support unsafe codes and use unsafeto mark the source code with pointer. To get the address of the first pixel data the pointer method need use the property Scan 0 of BitmapData class. The main steps of the memory pointer method is:

1) Load image to save to the Bitmap object or get the image object of PictureBox, which can convert to the Bitmap object;

2) Instance BitmapData class, and lock Bitmap object to system memory through LockBits() of Bitmap object;

3) Get the address of the first pixel data by using Scan0;

4) Traverse data of image and directly adjust these datum;

5) Unlock the Bitmap object using UnlockBit().

The store format should be considered as operating data of memory with pointer method.

1) The order of three-component of each pixel is Blue then Green then Red in system memory;

2) Byte alignment problems that a blank area need be jump after traversing pixel data of one row to avoid the result of twist deformation of processed image.

The main source codes:

```
Bitmap bmp = new Bitmap("E:\\test.bmp");
BitmapData bmpdata = bmp.LockBits(rect, ImageLockMode.ReadWrite,
```

```
bmp.PixelFormat);
Unsafe{
byte * ptr = (byte*)( bmpdata.Scan0);
//gets the address of the first pixel data in the bitmp
for (int i = 0; i < bmpdata.Height; i++)
for (int j = 0; j < bmpdata.Width; j++){
ptr[0] = (byte)(255 - ptr[0]);
ptr[1] = (byte)(255 - ptr[1]);
ptr[2] = (byte)(255 - ptr[2]);
ptr += 3;
}
ptr += bmpdata.Stride - bmpdata.Width * 3; // stride the blank area
}
bmp.UnlockBits(bmpdata);
```

As using pointer to operate data of memory in this method, the processing speed is very high .

1.4 WPF Method

Windows presentation foundation (WPF) is a new generation graphic system of

Microsoft, which need be run at .Net Framework 3.0. It provided the unified description and operation method to deal with user interface, 2D/3D graphic, document and multimedia, etc. WPF brings 2D/3D interface of unprecedented based on the technology of DirectX 9/10 and can make the image processing function great improvement of Microsoft Windows. WPF provides new serials API functions to display and edit image compared with winform programs' GDI+ API.

The main processing steps with WPF are:

1) Instance BitmapSource object and load picture;

2) Copy the data of image to array with CopyPixels function;

3) Process the data in byte array;

4) Create a new Image and get the final result.

The main source codes:

```
BitmapSource bmpsrc = new BitmapImage(new Uri("E:\\test.bmp")); //load picture

int stride = bmpsrc.PixelWidth* bmpsrc.Format.BitsPerPixel/8;

int offset = stride % 3;

int index = 0;

bmpsrc.CopyPixels(ImagePixelData, stride, 0); // Copy the data of image to array

for (int i = 0; i < h; i++)

for (int j = 0; j < w; j++)

{

ImagePixelData[index] = (byte)(255 - (int)ImagePixelData[index]);

ImagePixelData[index + 1] = (byte)(255-(int)ImagePixelData[index + 1]);

ImagePixelData[index + 2] = (byte)(255-(int)ImagePixelData[index+2]);

index += 3;

}

index += offset; // stride the blank area

BitmapSource bmpnew = BitmapSource.Create(w, h, bmpsrc.DpiX, bmpsrc.DpiX ,

PixelFormats.Bgr32, null, ImagePixelData, stride);
```

The performance efficiency of WPF is similar to memory array method, but it has the problem of byte alignment.

1.5 Direct 3D method

Direct 3D is a kind of 3D graphic API functions based on common object mode (COM) of Microsoft. For graphic display interface (GDI) was bypassed, it can directly operate bottom hardware supported the API to improve the speed of processing. As suited to multimedia, entertainment, real-time 3D animation and so on, Direct 3D had played an important role in some application filed of computer simulation. However, Direct 3D need to use texture to process picture.

To make the code easy managed Direct 3D pattern was used and the main steps are:

1) Create the display module of Direct 3D and load picture with texture mouthed;

2) Lock texture and get image data;

3)Traverse data of image and directly adjust these datum;

4) Unlock texture and get the final result.

The main source codes:

```
Texture texture = TextureLoader.FromFile(device, "E:\\test.bmp" ); //loads picture as texture

SurfaceDescription s = texture.GetLevelDescription(0);

uint* pData = (uint*) texture.LockRectangle(0,

LockFlags.None).InternalData.ToPointer();

//gets the pointer of image data

for (int i = 0; i < s.Width; i++)

for (int j = 0; j < s.Height; j++){

uint c1 = *pData;

byte[] ff = BitConverter.GetBytes(*pData);

ff[0] = (byte)(255 - ff[0]);

ff[1] = (byte)(255 - ff[1]);
```

```
ff[2] = (byte)(255 - ff[2]);
ff[3] = (byte)(255 - ff[3]);
*pData = (uint)BitConverter.ToInt32(ff,0);
pData++;
}
texture.UnlockRectangle(0);
```

The performance efficiency of Direct 3D is similar to memory pointer method.

2.Performance Analysis

The Methods above are implemented by programming in Microsoft Visual Studio 2008 and the programming language is C#. In order to accurately test the efficiency of all methods, a high precision time counter, which is QueryPerformanceCounter, was used in test program and the Timer component was not used for its low precision. As compiled source codes, code optimization and release mode were adopted to make all methods of image processing achieve the optimal effect. The program runs on the computer, whose processor is core 2 duo 1.8G and Memory is 2G and operating system is windows Xp sp3.

The image inverse transformation algorithm is used and arbitrarily chooses two pictures. The size of one is 1024×768 pixels, another is 2560×1920 pixels, and the color depth is 24 bit and image format is bitmap (BMP).

The conversion effect of five processing methods was very satisfied, but their performance efficiency is different greatly. Different methods had different efficiency, see Table Ek-1. The different features were shown Table Ek-2

Table Ek-1 Performance comparison

Methods	Performance time（ms）	
	1024*768 pixels 24bit	2560*1920 pixels 24bit
GetPixel /SetPixel	2366	15068
Memory array	7	50
Unsafe pointer	2	13
WPF	9	53
Direct3D	2	17

Table Ek-2 Parameters comparison

	GetPixel/ SetPixel	Memory array	Unsafe pointer	WPF	Direct3D
Easily use	easy-used	easy	not easy	easier	not easy
Efficiency	very low	higher	very high	higher	very high
Code complexity	low	higher	high	lower	very high
Byte alignment	no	no	yes	yes	no

According to the experimental results, we find out the advantages and disadvantages of the five methods above. The method of GetPixel/SetPixel is easy but its poor efficiency can not be neglected. It is only applied these image processing projects that do not require high efficiency; the memory array method improves executing efficiency greatly relative to GetPixel/SetPixel method. However, the efficiency of processing is not satisfied when raising complexity of operating algorithm or using bigger size picture;

WPF processing method is worth to be popularized and applied for its self-advantages while its performance efficiency is similar to memory array. Undoubtedly, the unsafe pointer method and Direct 3D are very high efficiency, and processing image will be completed transiently and the change of Picture size is few influence. The unsafe pointer method was commonly used to some projects of image processing and 3D simulation development while the Direct 3D operation are tedious. But Direct 3D was generally applied to these development of three-dimensional simulation and 3D graph presentation, etc [37].

Ek-2: MATLAB IMAGE PROCCESSING TOOLBOX

image	Display image.
imagesc	Scale data and display as image.
imageview	Show an image preview in a figure window
ind2rgb	Convert indexed image to RGB image.
rgb2ind	Convert RGB image to indexed image.
print	Print figure or model. Save to disk as image or M-file.
imread	Read image from graphics file.
imwrite	Write image to graphics file.
imagedemo	Images and Matrices
imageext	Examples of images with a variety of colormaps
cfrimage	Image
cmunique	Eliminate unneeded colors in colormap of indexed image.
imapprox	Approximate indexed image by one with fewer colors.
contrast	Gray scale color map to enhance image contrast.
dither	Convert image using dithering.
bwarea	Area of objects in binary image.
bwareaopen	Morphologically open binary image (remove small objects).
bwboundaries	Trace region boundaries in binary image.
bwconncomp	Find connected components in binary image.
bwdist	Distance transform of binary image.
bwdist_old	Distance transform of binary image.
bweuler	Euler number of binary image.
bwfill	Fill background regions in binary image.
bwlabel	Label connected components in 2-D binary image.

bwlabeln	Label connected components in binary image.
bwmorph	Morphological operations on binary image.
bwpack	Pack binary image.
bwperim	Find perimeter of objects in binary image.
bwselect	Select objects in binary image.
bwtraceboundary	Trace object in binary image.
bwunpack	Unpack binary image.
checkerboard	Create checkerboard image.
cornermetric	Create corner metric matrix from image.
deconvblind	Deblur image using blind deconvolution.
deconvlucy	Deblur image using Lucy-Richardson method.
deconvreg	Deblur image using regularized filter.
deconvwnr	Deblur image using Wiener filter.
decorrstretch	Apply decorrelation stretch to multichannel image.
demosaic	Convert Bayer pattern encoded image to a truecolor image.
dilate	Perform dilation on binary image.
edge	Find edges in intensity image.
entropy	Entropy of intensity image.
entropyfilt	Local entropy of intensity image.
erode	Perform erosion on binary image.
gray2ind	Convert intensity image to indexed image.
grayslice	Create indexed image from intensity image by thresholding.
graythresh	Global image threshold using Otsu's method.
im2bw	Convert image to binary image by thresholding.
im2col	Rearrange image blocks into columns.

im2double	Convert image to double precision.
im2int16	Convert image to 16-bit signed integers.
im2java2d	Convert image to Java BufferedImage.
im2single	Convert image to single precision.
im2uint16	Convert image to 16-bit unsigned integers.
im2uint8	Convert image to 8-bit unsigned integers.
imabsdiff	Absolute difference of two images.
imadd	Add two images or add constant to image.
imadjust	Adjust image intensity values or colormap.
imclearborder	Suppress light structures connected to image border.
imclose	Morphologically close image.
imcomplement	Complement image.
imcontour	Create contour plot of image data.
imcrop	Crop image.
imdilate	Dilate image.
imdivide	Divide two images or divide image by constant.
imerode	Erode image.
imfeature	Compute feature measurements for image regions.
imfill	Fill image regions and holes.
imfilter	N-D filtering of multidimensional images.
imhist	Display histogram of image data.
imlincomb	Linear combination of images.
immultiply	Multiply two images or multiply image by constant.
imnoise	Add noise to image.
imopen	Morphologically open image.

impyramid	Image pyramid reduction and expansion
imresize	Resize image.
imresize_old	Resize image (old version).
imrotate	Rotate image.
imsubtract	Subtract two images or subtract constant from image.
imtransform	Apply 2-D spatial transformation to image.
ind2gray	Convert indexed image to intensity image.
isbw	Return true for binary image.
isgray	Return true for intensity image.
isind	Return true for indexed image.
isrgb	Return true for RGB image.
label2rgb	Convert label matrix to RGB image.
mat2gray	Convert matrix to intensity image.
phantom	Create head phantom image.
rangefilt	Local range of image.
regionprops	Measure properties of image regions.
rgb2gray	Convert RGB image or colormap to grayscale.
roifill	Fill in specified polygon in grayscale image.
stdfilt	Local standard deviation of image.
stretchlim	Find limits to contrast stretch an image.
batchDetectCells	Algorithm to detect cells in image.
batchProcessFiles	Process image files.
ipexbatch	Batch Processing Image Files in Parallel
ipexblind	Deblurring Images Using the Blind Deconvolution
ipexblockprocedge	Block Processing Large Images

ipexblockprocstats	Computing Statistics for Large Images
ipexcell	Detecting a Cell Using Image Segmentation
ipexcheckerboard	Creating a Gallery of Transformed Images
ipexlanstretch	Enhancing Multispectral Color Composite Images
ipexlucy	Deblurring Images Using the Lucy-Richardson Alg.
ipexndvi	Finding Vegetation in a Multispectral Image
ipexnormxcorr2	Registering an Image Using Normalized Cross-Correlation
ipexprops	Measuring Regions in Grayscale Images
ipexreconstruct	Reconstructing an Image from Projection Data
ipexregularized	Deblurring Images Using a Regularized Filter
ipexrotate	Finding the Rotation and Scale of a Distorted Image
ipexshear	Padding and Shearing an Image Simultaneously
ipexwiener	Deblurring Images Using a Wiener Filter
iptdemos	Index of Image Processing Toolbox demos.
LanAdapter	Example ImageAdapter for Erdas LAN images.
propsSynthesizeImage	propsSyntheticImage create image for ipexprops demo
getimage	Get image data from axes.
getimagemodel	Get image model object from image object.
imageinfo	Image Information tool.
imattributes	Information about image attributes.
imgca	Get handle to current axes containing image.
imgcf	Get handle to current figure containing image.
imgetfile	Open Image dialog box.
imhandles	Get all image handles.
immovie	Make movie from multiframe image.

imoverview	Overview tool for image displayed in scroll panel.
imoverviewpanel	Overview tool panel for image displayed in scroll panel.
implay	Play movies, videos, or image sequences.
imputfile	Save Image dialog box.
imsave	Save Image tool.
imscrollpanel	Scroll panel for interactive image navigation.
imshow	Display image in Handle Graphics figure.
imtool	Display image in the Image Tool.
imview	Display image in the image viewer.
montage	Display multiple image frames as rectangular montage.
subimage	Display multiple images in single figure.
truesize	Adjust display size of image.
warp	Display image as texture-mapped surface.
analyze75read	Read image file of Mayo Analyze 7.5 data set.
dicomread	Read DICOM image.
dicomwrite	Write images as DICOM files.
hdrread	Read Radiance HDR image.
ImageAdapter	Interface for image format I/O.
interfileread	Read images in Interfile 3.3 format.
makehdr	Create high dynamic range image.
nitfread	Read NITF image
rsetwrite	Create reduced-resolution dataset from image file.
tonemap	Render high dynamic range image for viewing.
getrangefromclass	Get dynamic range of image based on its class.
iptgetpref	Get value of Image Processing Toolbox preference.

iptprefs	Display Image Processing Toolbox preferences dialog.
iptsetpref	Set value of Image Processing Toolbox preference.
imaqfind	Find image acquisition objects.
imaqgate	Gateway routine to call Image Acq Toolbox private func.
imaqhelp	Return image acquisition object function and property help.
imaqmem	Limit or display memory in use by the Image Acq Toolbox.
imaqregister	Register third party Image Acquisition Toolbox adaptors.
imaqreset	Disconnect and delete all image acquisition objects.
imaqsupport	Image Acquisition Toolbox troubleshooting utility.
imaqtool	Launch the Image Acquisition Tool
imaqmontage	Display a sequence of image frames as a montage.
imaqinitlib	Initializes the Image Acquisition blockset library.
imaqparsehwinfo	Parse the Image Acquisition Toolbox hardware information.
imaqsfcncreate	Creates a Image Acquisition Toolbox object for the S-Func.
imaqslgate	Gateway routine to call Image Acquisition Toolbox SL private functions.
ind2rgb8	Convert indexed image to uint8 RGB image
mapbbox	Compute bounding box of georeferenced image or data grid
mapoutline	Compute outline of georeferenced image or data grid
pixcenters	Compute pixel centers for georeferenced image or data grid
mapexrefmat	Creating a Half-Resolution Georeferenced Image
mapexreg	Georeferencing an Image to an Orthotile Base Layer
grid2image	Display regular data grid as image
imagem	Display a regular matrix map as an image.
geotiffread	Read georeferenced image from GeoTIFF file

getworldfilename Derive worldfile name from image file name

replacecolor Replace a color in truecolor image with another

xregimage Create a modified image object

rgb2rgb565 Convert 8-bit RGB image to RGB565 representation.

bmp2rgb565 Convert bitmap image to RGB565 representation.

aboutvipblks Displays version number of the Video and Image

blackimage Return black image of desired size and data type.

checker_board RGB test image using a checker-board pattern.

sl_customization Customization file for Video and Image Proc. Blockset.

vipbhelp Video & Image Processing Blockset on-line help function.

viplib Open Video and Image Processing Blockset library.

vipliblist Return list of Video and Image Processing block libraries.

vipblkimsrc - Mask callback function for Image From Workspace block

dw2dimgs Discrete wavelet 2-D image selection.

dw2drwcd Discrete wavelet 2-D read-write Cdata for image.

getimgfiletype Getimage file types.

imgxtool Image extension tool.

wfusimg Fusion of two images.

wfustool Discrete wavelet 2D tool for image fusion.

wimgcode Image coding mode.

wpropimg Give image proportions.

denoisingsignalsdemo De-Noising Signals and Images

dguiiext Demonstrates Image extension GUI tools in the Wavelet Toolbox.

dguiwfus Demonstrates Image Fusion tool in the Wavelet Toolbox.

imagefusiondemo	Image Fusion
wcompress	True compression of images using wavelets.
wconvimg	Image transform for images truecolor to grayscale
frame2im	Return image data associated with movie frame.
im2frame	Convert indexed image into movie format.
im2java	Convert image to Java image.
rtw_c	Creates the makefile used to build the RTW C code
hdfdfr8	MATLAB gateway to HDF 8-bit raster image interface.
aeroimage	function for the icon images of Aerospace Blockset.
imageneread	reads ImaGene Results Format files.
maimage	displays a spatial image of microarray data.
msheatmap	creates a heat map image of a set of spectra
scaleimagefigure	resizes the figure window to fit the image size.
parsebinary	Write binary object to disk and display if image.
myind2rgb	IND2RGB Convert indexed image to RGB image.
vipaviread	Read AVI file. Used by Video & Image Proc. Blockset's
HeatMap	A false color 2D image of the data values in a matrix.
imagemodel	Access to properties of an image relevant to its display.
blkproc	Distinct block processing for image.
blockproc	Distinct block processing for image.

ÖZGEÇMİŞ

Adı	**Tevfik**	**Soyadı**	**AKKUŞ**
D. Yeri	Sinop / Boyabat	D. Tarihi	1974
Uyruğu	Türkiye Cumhuriyeti	Tel	+90 534 251 85 51
E-mail	tevfik.akkus@gmail.com		

Eğitim Düzeyi

	Mezun Olduğu Kurumun Adı	Mezuniyet Yılı
Doktora/Uzmanlık	-	-
Yüksek Lisans	-	-
Lisans	Sakarya Üniversitesi Mühendislik Fakültesi Elektrik ve Elektronik Mühendisliği Bölümü	1997
Lise	Tuzla Teknik Lisesi Elektrik Bölümü	1992

İş Deneyimi

Görevi	**Kurum**	**Süre (Yıl)**
Başuzman Araştırmacı	TÜBİTAK	2
Firma Sahibi	TERA ELEKTRONİK ve MÜHENDİSLİK HİZMETLERİ	9
Servo ve Robotik Sistemler Sorumlusu	SEW-EURODRIVE HAREKET SİS. LTD. ŞTİ.	1
End. Otomasyon Proje Müh.	ÜLKER GIDA A.Ş.	2
Elektrik Elektronik Bakım Müh.	EMİN OPTİK CAM SAN. LTD.ŞTİ.	2
Elektrik Elektronik Bakım Şefi	TEKEL SAMSUN BALLICA SİGARA FABRİKASI	1

Yabancı Dilleri	Okuduğunu Anlama	Konuşma	Yazma
İngilizce	Çok iyi	Çok İyi	Çok iyi
Almanca	Orta	Zayıf	Orta

Yabancı Dil Sınav Notu								
YDS	ÜDS	IELTS	TOEFL IBT	TOEFL PBT	TOEFL CBT	FCE	CAE	CPE
-	-	-	-	-	-	-	-	-

	Sayısal	Eşit Ağırlık	Sözel
ALES Puanı	63,49	62,22	56,70
(Diğer) Puanı	-	-	-

Bilgisayar Bilgisi

Program	Kullanma becerisi
MATLAB, SIMULINK	İyi
MS Windows, Office	İyi