T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ

MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

YAKIN RESİM FOTOGRAMETRİSİNDE ÖLÇME SİSTEMLERİ İÇİN MATLAB ARAYÜZ ÇALIŞMASI

Avnullah KANÇURA

DANIŞMANI:

Yrd. Doç. Dr. Bahadır ERGÜN

YÜKSEK LİSANS TEZİ

JEODEZİ VE FOTOGRAMETRİ MÜHENDİSLİĞİ

ANABİLİM DALI

GEBZE

2010

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ	YÜKSEK LİSANS JÜRİ ONAY FORMU

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……. tarih ve …… sayılı kararıyla oluşturulan jüri tarafından ……/……/…… tarihinde tez savunma sınavı yapılan …….'ın tez çalışması …….. Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) :

ÜYE :

ÜYE :

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……/……/…… tarih ve ……/…… sayılı kararı.

İMZA/MÜHÜR

ÖZET

TEZİN BAŞLIĞI: YAKIN RESİM FOTOGRAMETRİSİNDE ÖLÇME SİSTEMLERİ İÇİN MATLAB ARAYÜZ ÇALIŞMASI

YAZAR ADI: Avnullah KANÇURA

Günümüzde yakın resim fotogrametrisi çalışmalarında çoklu kamera sistemlerinin kullanımı bize modellenecek cismin en iyi şekilde aktarımı için büyük bir avantaj sağlamaktadır. Bunun için kullanılan profesyonel yazılım donanım sistemlerinin bir benzerinin yapımı bu tezde açıklanmıştır.

Özel olarak bu iş için kullanılan kalibrasyonlu kameralarla (Ueye) oluşturulan bir düzenek ve bunun Matlab tabanlı yazılım vasıtasıyla kullanılması ve model oluşturulması projenin genel bir özetidir. Böylelikle Matlab'ın ActiveX ile erişim, görüntü işleme ve hızlı lineer çözümleme, Opengl desteğiyle grafiksel gücü hem görülüp hem de kullanılmıştır.

Anahtar Kelimeler: Yakın Resim Fotogrametrisi, Kamera Verileri, Matlab Programlama, Matlab Kamera Erişimi, Matlab Opengl

SUMMARY

TITLE: MATLAB GUI DEVELOPMENT FOR NEAR RANGE PHOTOGRAMMETRY MEASUREMENT SYSTEMS

AUTHOR: Avnullah KANÇURA

Today it is a big advantage for us to use multi camera systems in near range photogrammetry studies by enabling us to get much more detailed modelling of the the target object. Development of a similar hardware-software system just like professionals is described in this thesis.

Usage of a system that consists of special calibrated cameras(Ueye) and its usage by a Matblab based application and model derivation are general features of this project. Thats also shows us Matlab's power and usage of: hardware access by ActiveX, image processing and fast lineer solutions and new Opengl graphics practises.

Key Words: Near Range Photogrammetry, Camera datas, Matlab Programming, Matlab Camera Access, Matlab Opengl

TEŞEKKÜRLER

Öncelikle, yüksek lisans boyunca bana sabırla destek olup yön veren ve gerektiğinde benimle laboratuarda geç saatlere kadar çalışan, problem çözme yetisi ve bilimsel araştırma yapma yeteneği kazandıran GYTE Öğretim Üyesi Danışmanım Sayın Yrd. Doç. Dr. Bahadır ERGÜN'e;

Laboratuvarda beraber çalıştığım ve bana destek olan GYTE Araştırma Görevlisi Cumhur ŞAHİN'e;

Tahsilimin bu aşamaya gelmesinden, hayatımın her aşamasında maddi manevi her konuda desteklerini esirgemeyen aileme kucak dolusu teşekkürlerimi sunarım.

İÇİNDEKİLER DİZİNİ

Sayfa

SİMGELER ve KISALTMALAR DİZİNİ

RGB	: Red, Green, Blue (Kırmızı, Yeşil, Mavi)
OpenGL	: Open Graphics Library (Açık Grafik Kütüphanesi)
3D	: 3 Dimensional (ÜçBoyutlu)
GUI	: Graphical User Interface (Görsel Kullanıcı Arayüzü)
x_0,y_0	: Resim koordinat sistemi başlangıcı
c	: Odak uzaklığı
k	: Ölçek faktörü

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

GİRİŞ

Aslı Yunanca olup batı dillerine giren Fotogrametri sözcüğü üç sözcükten oluşur.

Photos(ışık) + Grama(çizim) + Metron(ölçme)

Yani ışık yardımı ile çizerek ölçme anlamına gelmektedir. Bu teknikte ölçülmek istenen nesnenin ve yakın çevresinin ya da arazinin fotoğrafları çekilir. Bunların fotoğraf üzerindeki görüntüleri ölçülerek istenen bilgiler sağlanabilir, ya da özel aletlerde bu görüntüler harita ya da plan ya da model biçimine dönüştürülebilir.

Fotogrametri günümüzde teknolojinin de etkisiyle Computer Vision (Bilgisayar Görmesi) ve Uzaktan Algılama (Remote Sensing) gibi alanlarla yakından ilişkilidir. (Kraus K.,1994). Yani fotogrametrik uygulamalar için geliştirilmiş yazılım-donanım sistemleri mevcuttur. Bu sistemler temelde bir kamera alanı ve bununla bağlantılı yazılım sistemlerinden oluşur. Kullanılacağı alana göre böyle bir sistemin maliyeti on binlerce doları bulabilmektedir. Yani hem kamera sistemleri hem de yazılım sistemleri açısından ciddi maliyet hesapları söz konusu olmaktadır. İşte bu çalışmadaki amacımız böyle bir sistemin en uygun ve ekonomik bir yolla nasıl hazırlanabileceğine dair bir örnek oluşturmaktır.

Bu projede, kamera sistemleri haricindeki donanım ve yazılım tamamen üniversite imkânlarıyla oluşturularak fotogrametrik bir sistem hazırlanmasına çalışılmıştır. Uygulama arayüzünün geliştirileceği platform olarak Matlab seçilmiştir.

1.1 Endüstriyel Yakın Resim Fotogrametrisi Nedir?

Fotogrametri, kullanılan kameranın durumuna, ölçülecek nesnenin yakın ya da uzak oluşuna, edinilecek bilgi türüne, değerlendirme yöntemine ve uygulama alanlarına göre sınıflandırılabilir.

Buna göre:

- Yersel Fotogrametri: Yer üzerinde çekilmiş fotoğraflarla çalışan fotogrametridir. Nesne-kamera uzaklığı en çok 300m olan bir fotogrametri uygulamasıdır
- Hava Fotogrametrisi: Uçaktaki, ya da genel olarak bir hava aracında bulunan bir kamera ile çekilmiş fotoğraflarla çalışan fotogrametri
- Foto-Yorumlama: Fotoğrafik dokuyu inceleyerek nesne ve yakın çevresi hakkında bilgi üreten, arazinin yapısını ve yüzey özelliklerini inceleyen fotogrametri kolu
- Metrik Fotogrametri: Fotoğraflardan konum, yükseklik, uzaklık, alan ve hacim gibi metrik bilgilerin alınmasını, ya da doğrudan doğruya harita çizimini amaçlayan fotogrametri
- Topoğrafik Fotogrametri: Topoğrafik harita üretimi ile ilgili haritacılık fotogrametrisi
- Topoğrafik Olmayan Fotogrametri: Topoğrafik harita yapımı, topoğrafik ölçmeler dışında, başka bir deyişle haritacılık dışında kalan fotogrametri
- Kadastro Fotogrametrisi: Kadastro haritalarının yapımında kullanılan fotogrametri
- Jeodezik Fotogrametri: Jeodezik nokta üretiminde kullanılan fotogrametri
- Mühendislik Fotogrametrisi: Mühendislik projelerinin hazırlanmasında vb. çalışmalarda kullanılan fotogrametri
- Mimarlık Fotogrametrisi: Özellikle tarihsel yapıların belgelenmesinde kullanılan fotogrametri
- Analog Fotogrametri: Değerlendirmelerin, özellikle harita çizimlerinin analog aletlerde yapıldığı fotogrametri

- Analitik Fotogrametri: Çözümlerin matematiksel yöntemlerle yapıldığı, bilgisayar destekli fotogrametri uygulaması
- Sayısal(Dijital) Fotogrametri: Sayısal(Dijital) fotoğraflarla çalışan fotogrametri
- Tek Resim Fotogrametrisi: Tek tek fotoğrafları kullanarak metrik bilgiler üretmeyi amaçlayan, foto-plan, foto-mozaik üreten fotogrametri
- Çift Resim Fotogrametrisi: Ortak alanları olan fotoğraf çiftleri üzerinde ölçüler yaparak bilgi üretmeyi amaçlayan fotogrametri. Stereoskopik(üç boyutlu) görüşte söz konusu olduğundan buna stereo fotogrametri de denir.
- Ortofotografi: Çizgi harita ile aynı geometrik doğruluğa sahip foto haritalarının üretimi ile ilgilenen fotogrametri

şeklinde sınıflandırılabilir.

Yersel fotogrametride, alım uzaklığının (kamera ile ölçülecek nesne arasındaki uzaklık) mikrometre mertebesinden yaklaşık 300m'ye kadar uzandığı kabul edilir.

Bilgisayar teknolojisine bağlı olarak alet üretimindeki gelişmeler, yersel fotogrametrinin özellikle endüstride ve uygulamalı mühendislik gibi alanlardaki uygulama imkânlarını arttırmaktadır. Yersel fotogrametride 25m'ye kadar olan çekim uzaklıklarında uygulanan fotogrametriye "Yakın Resim Fotogrametrisi" adı verilir.

Yakın resim fotogrametrisinin endüstriyel alanlara yönelik kullanılmasına "Endüstriyel Yakın Resim Fotogrametrisi" denir.

1.2 Endüstriyel Fotogrametrik Ölçme Sistemi Nasıl Kurulur?

Bu çalışmanın genel amacı, yüksek teknolojiye ulaşmış bir sanayinin altyapısında bulunması zorunlu olan presizyonlu endüstriyel ölçme sistemlerine bir altlık oluşturmak ve günümüzde bu amaçla gelişmiş endüstrilerde kullanılan fotogrametri yöntemlerini, donanımından yazılımına kadar ülkemizde ve enstitümüzde uygulama olanağı yaratmaktır.

Günümüz sanayisinde en önemli üretim yöntemi seri ve olanakların elverdiğince hatalardan arıtılmış üretimdir. Endüstriyel ölçme şartları göz önüne alındığında Endüstride fotogrametrinin ölçme hızı, veri işleme, değerlendirmesi ve ölçme doğruluğu açısından üretilen malın değerlendirilmesinde en uygun sonucu verdiği ayrıca ölçme işleminin üretim koşullarında yapılabilmesi ile en uygun ölçme yöntemi olduğu görülmüştür. (Fraser, C.S., 1996.)

Dijital Endüstriyel Kontrol Sistemleri ilk yatırım maliyeti açısından çok pahalı olmalarına karşın, uzun süre kullanılabilmeleri, üretimin kontrol ve hızını artırmaları açısından özellikle yüksek maliyetli imalat yapan sanayilerde (uçak motoru ve otomotiv sanayii gibi) tercih edilen sistemlerdir (Ergün B.,2003). Dijital Fotogrametrinin üretim sektöründe geniş bir biçimde kullanılmasının en önemli nedeni, gerçek zaman ve ya gerçek zamana çok yakın (real time, near real time) denilen üç boyutlu konum belirlemedeki özelliğidir. Resim ölçeklerinin diğer yakın mesafe fotogrametrisine oranla daha büyük olması nedeni ile endüstriyel fotogrametride, iki boyutlu resim kordinatlarından üç boyutlu cisim kordinatlarının hesaplanması için aşağıdaki sistem kurulumu düşünülmüştür.

Sistem Kurulumu:

- Kalibrasyon alanı
- Kalibrasyonun matematik modeli
- Hesaplamalar
- Kalibrasyon sonuçlarının sisteme entegresi şeklinde tanımlanabilir.

GYTE de geliştirilen sistem hem donanım hem yazılım esasına dayalı bir uzman sistemdir. Temel amacı, modellemesi güç ve ölçülmesi boyutu itibari ile zor olan cisimlerin istenen presizyonda fotogrametrik olarak ölçülüp modellenebilmesi amacına yöneliktir.

Dolayısı ile bu sistem, sürücü programlarına ek olarak sistem içinde alınan resimlerin yine bölüm bünyesinde hazırlanan Matlab temelli fotogrametrik yazılımda işlenmesinden oluşmaktadır.

Uzman sistem temel olarak iki ana yapıdan oluşmaktadır.

- Donanım
- Yazılım

1.2.1 Donanım

Donanım esas itibari ile iki ana kısımdan oluşmaktadır.

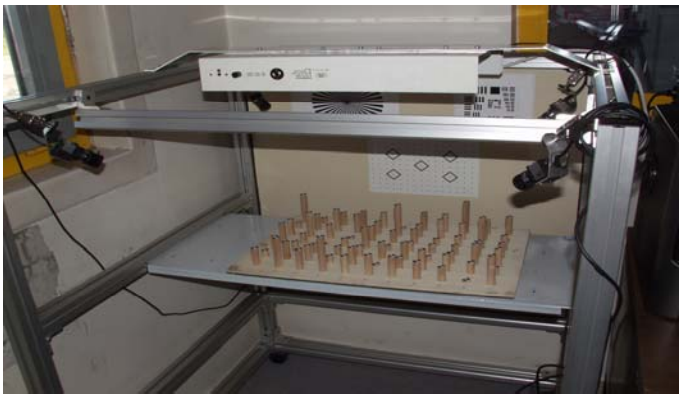
- Yapı Bölümü
- Optik ve Elektronik Bölüm

Sistemin yapısını oluşturan ana çatı Şekil 1.1.'de gösterildiği gibi teknik özellik itibarı ile ısı ve nem değişimlerine dayanıklı ağırlık altında deformasyonu düşük alüminyum çelik profillerden oluşmaktadır. Bu alüminyum çelik alaşımlı yapı için

Bosch firmasının özel endüstriyel ürünleri kullanılmıştır. Sistem çatısını oluşturacak bu yapı için tasarımda öncelikle, kullanılırlık, sağlamlık ve görüntüleme için uygun geometrinin oluşturulabilmesi gibi etkenler öncelikle düşünülmüştür.

Şekil 1.1 Sistemin Yapı Bölümü

Sistemin birinci adımını kalibrasyon çalışması oluşturduğundan orta büyüklükte sabit odak uzaklıklı Pentax CCTV Lens (16 mm) lens kullanılmıştır. Daha sonraki çalışmalarda odak uzaklığı yapılacak çalışmanın şekline göre belirlenerek değiştirilebilir. Kullanılan 16 mm odak uzaklığına sahip lensler endüstriyel ölçmelerde en sık kullanılan ve en optimum görüntüyü veren lensler olmasının yanı sıra sistemin boyutları ve hedefleri içinde ilk planda yeterli olduğu belirlenmiştir. Sistemde yöneltme çalışması için Fotogrametri laboratuvarında 77 noktadan oluşan kalibrasyon alanı tesis edilmiştir. Lensler için kalibrasyon sonuçları 5 mikron (1 piksel) doğrulukla hesaplanmış olup aşağıda verilmiştir.



Şekil 1.2 Sistemdeki Kamera Geometrisi

Kamera Sabiti Lens Sabiti	**c (mm)**	**x_0 (mm)**	**y_0 (mm)**	**K_1**	**K_2**	**P_1**	**P_2**
4002704847 463490	15,897121	1,924324	1,558322	$1{,}702831 \times 10^{-3}$	$-8{,}406524 \times 10^{-5}$	$1{,}325 \times 10^{-3}$	$-6{,}31222 \times 10^{-4}$
4002704846 463459	15,932114	1,892112	1,458439	$1{,}712816 \times 10^{-3}$	$-8{,}406224 \times 10^{-5}$	$2{,}199 \times 10^{-3}$	$-5{,}3632 \times 10^{-4}$
4002704844 460535	15,962356	1,891322	1,481636	$1{,}70198 \times 10^{-3}$	$-6{,}433624 \times 10^{-5}$	$1{,}245 \times 10^{-3}$	$-7{,}221522 \times 10^{-4}$
4002679106 463644	15,886334	1,924322	1,588383	$1{,}702855 \times 10^{-3}$	$-8{,}417302 \times 10^{-5}$	$1{,}224 \times 10^{-3}$	$-6{,}726401 \times 10^{-4}$

Tablo 1.1 Dört Kamera ve Lenslerine Ait İç Yöneltme Parametreleri ve Distorsiyon Değerleri

Sistem için dört adet kamera kullanılmıştır. Uzman sistemin kamera geometrisi Şekil 1.2'de gösterilmektedir. İki boyutlu resimlerden üçüncü boyutun elde edilmesi için farklı açılardan ya da bir baz üzerinden aynı anda resim elde etmek mümkündür. Sistemde görüntü elde etme elemanı olarak kullanılan endüstriyel Alman IDS uEye 2230 kameraları için gerekli teknik bilgi Ek1'de verilmiş olup, IDS uEye 2230 kamera ve Pentax CCTV Lens (16 mm) lens Şekil 1.3'de gösterilmektedir.

Donanımın bilgisayar desteği kameralar için USB bağlantısıdır. Görüntü kartı kullanılmayıp kameralar PC'ye USB bağlantısı ile bağlanmaktadır.

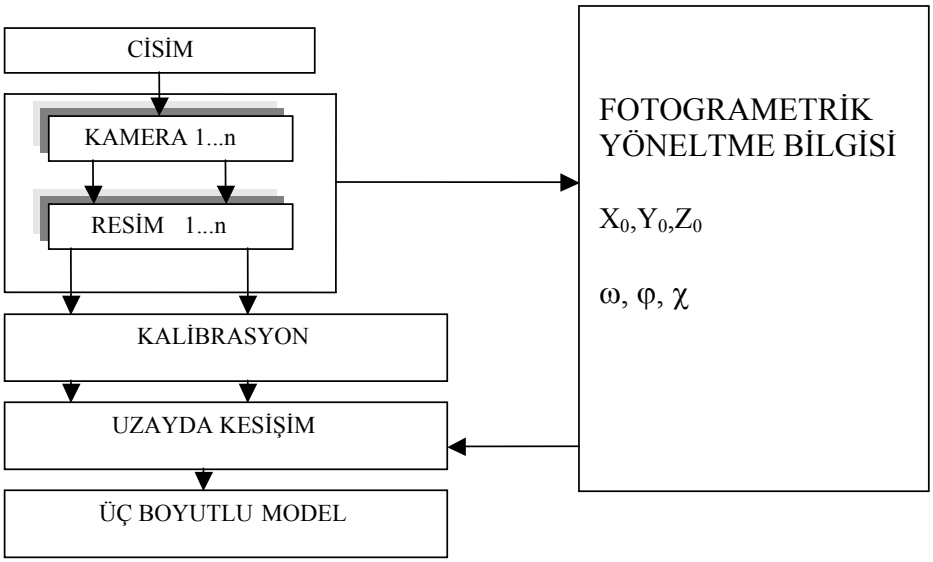
Şekil 1.3 IDS UEYE 2230 Dijital Kamera ve 16 mm. Lensi

IDS uEye 2230 Kameralarının Teknik Özellikleri	
Arayüz	USB 2.0
Sensör Teknolojisi	CCD (Sony)
Model Tanımlama (Renkli)	UI -2230-C
Model Tanımlama (Siyah-Beyaz)	UI -2230-M
Çözünürlük (h x v)	1024 x 768
Çözünürlük Kategorisi/Pixel Sınıfı	XGA
Sensör Boyutu	1/3"
Shutter	Global
Maksimum fps free mod	30
Maksimum fps 1 ms'deki exposure	27
Pixel Büyüklüğü (µm.)	4,65
Resim Boyutları (mm)	4,76 x 3,57
Dönüştürücü Hızı	04:03

Tablo 1.2 IDS uEye 2230 Kameralarının Teknik Özellikleri

1. 2. 2. Kurulan Endüstriyel Fotogrametrik Sistemde İşlem Akışı

Sistem dört kamera ile kurulmuş olup, kamera sayısı arttırılabilir özelliktedir. Şekil 1.4.'te gösterilen endüstriyel fotogrametrik uzman sistem işlem akışında önce kamera kalibrasyonunun yapılması ve numaralandırılmış her bir kameranın iç yöneltme ve distorsiyon değerlerinin önceden belirlenmesini gerektirmektedir. Bu işlem akışına göre dört kamera ölçülen cisim için dört ayrı resim elde etmekte ve bu dört farklı resimden cisim üzerindeki noktalara ait üç konum parametresi (X_0,Y_0,Z_0) ve üç dönüklük parametreleri (ω, φ, χ) uzayda kesişim yöntemi ile hassas olarak belirlenmektedir. Belirlenen nokta koordinatlarına bağlı olarak da cisme ait CAD model CAD ortamında çizilmektedir.



Şekil 1.4 Endüstriyel Fotogrametrik Sistemde İşlem Akışı
(Ergün B., Şahin C.,2009)

Fotogrametrik yöneltme ile kamera yöneltme parametreleri, aynı zamanda tüm sistem parametreleri elde edilir. Bu parametrelerin uygulanması ve görüntülenen obje resimlerindeki ölçümler ile objenin üç boyutlu modellenmesi mümkündür.

1.2.3 uEye Sürücü ve Matlab'la Hazırlanan Fotogrametri Yazılımı

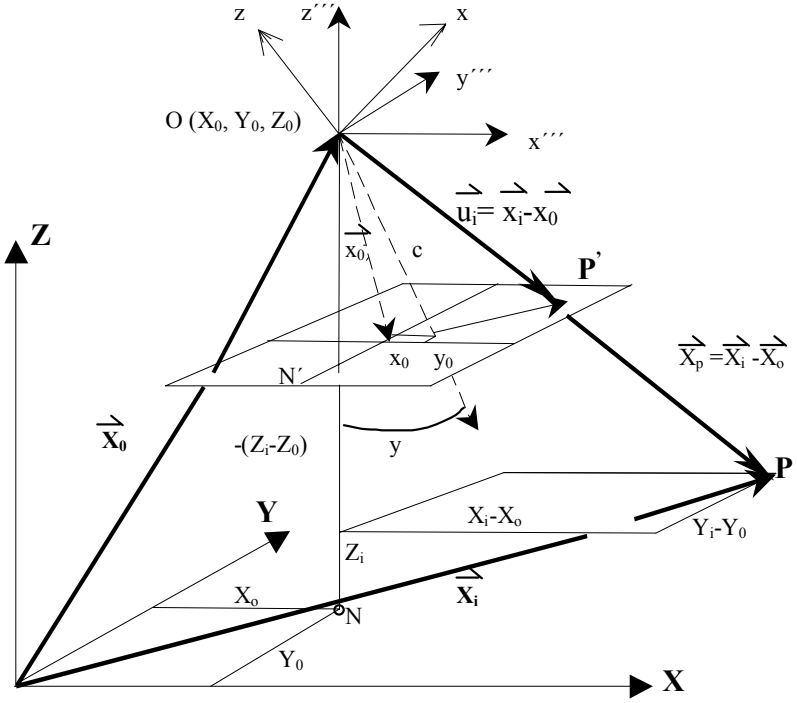
IDS firmasının ürettiği kameralar için görüntü elde etme ve görüntü işleme yazılımı olan IDS uEye Demo yazılımı; elde edilen sürekli video görüntüsünün tamamının ya da istenen bölümlerinin bilgisayara kayıt işleminin çeşitli görüntü dosyası formatlarında yapılabilmesini sağlamaktadır ve temel olarak kamera fonksiyonalitesini gösterir bir yapıdadır.

İlerleyen bölümlerde anlatılacak olan Fotogrametrik yazılımımız ise uEye sürücülerinin ve ActiveX desteğinin kullanımına güzel bir örnek teşkil etmektedir.

1. 2. 4. Kullanılan Matematik Model

Matlab'da hazırlanan Fotogrametrik yazılım kullanılarak yapılan kalibrasyon ve yöneltme hesaplarının matematik modeli cisim uzay koordinat sisteminde mevcut noktaların fiziksel oluşum esaslarının matematiksel ifadelerle gösterimidir.

Şekil 1.5'de gösterilen kolinearite koşuluna göre, cisim uzayındaki noktaları gösteren ışınlar resim çekme makinesi perspektif merkezinden geçerek resim düzlemine bir doğru boyunca izdüşürülür. Bu şekilde oluşan görüntüler, yeniden inşaa probleminin çözümünü iki adımda gerçekleştirir.

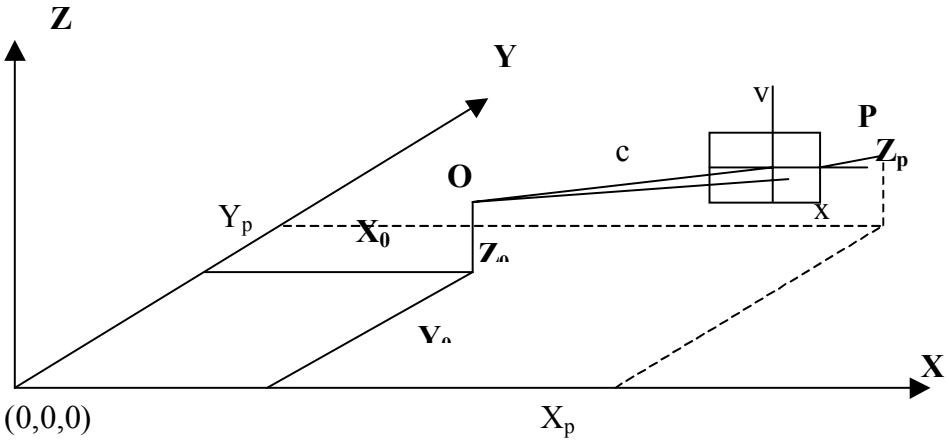


Şekil 1.5 Kolinearite Koşulu

İlk adım, resim çekme makinesi iç yöneltme parametrelerinin (ana nokta uzunluğu ve ana nokta koordinatları) belirlenmesidir.

İkinci adım, dış yöneltme adımıdır. Burada X_0, Y_0, Z_0 koordinatlarına sahip istasyon noktasından ω, φ, κ dönüklük değerleri ile çekilen resimlerin cisim uzay koordinat sistemindeki koordinatlarının hesaplanması amaçlanır.

Cisim uzay koordinat sistemi, noktaları X, Y, Z cisim koordinatlarını, resim koordinat sistemi x, y resim koordinatlarını gösteren sağ el kartezyen koordinat sistemleridir.



Şekil 1.6 Yersel Fotogrametrik Koordinat Sistemi

Resim ve cisim uzay koordinat sitemleri arası açısal ilişkiler, koordinat eksenleri arası doğrultu kosinüslerinin oluşturduğu (3X3) ortogonal dönüşüm matrisi ile belirlenir.

X eksen doğrultusunda dönüklük matrisi:

$$D_\omega = \begin{vmatrix} 1 & 0 & 0 \\ 0 & Cos\omega & Sin\omega \\ 0 & -Sin\omega & Cos\omega \end{vmatrix} \qquad (2.1)$$

Y eksen doğrultusunda dönüklük matrisi:

$$D_\varphi = \begin{vmatrix} Cos\varphi & 1 & -Sin\varphi \\ 0 & 1 & 0 \\ Sin\varphi & 0 & Cos\varphi \end{vmatrix} \qquad (2.2)$$

Z eksen doğrultusunda dönüklük matrisi:

$$D_\kappa = \begin{vmatrix} Cos\kappa & Sin\kappa & 0 \\ -Sin\kappa & Cos\kappa & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (2.3)$$

Toplam dönüklük matrisi:

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \quad (2.4)$$

a_{11}=Cosφ·Cosκ (2.5)

a_{12}=Sinω.Sinφ.Cosκ+Cosω.Sinκ (2.6)

a_{13}= -Cosω.Sinφ.Cosκ+Sinω.Sinκ (2.7)

a_{21}= -Cosφ.Sinκ (2.8)

a_{22}= -Sinω.Sinφ.Sinκ+Cosω.Cosκ (2.9)

a_{23}=Cosω.Sinφ.Sinκ+Sinω.Cosκ (2.10)

a_{31}=Sinφ (2.11)

a_{32}= -Sinω.Cosφ (2.12)

a_{33}=Cosω.Cosφ (2.13)

olur.

Resim koordinat sisteminde p noktasının konumunu belirleyen resim vektörü,

$$p = \begin{vmatrix} x_p - x_0 \\ y_P - y_0 \\ 0 \quad -c \end{vmatrix} \quad (2.14)$$

Burada; x_0, y_0 ana nokta koordinatları, c odak uzaklığıdır.

Resim çekme makinesi sonsuza odaklanmadıkça ana nokta uzaklığı, odak uzaklığa eşit değildir. Bu durumda ana nokta uzunluk değeri, odak uzaklığından Δf kadar sapma değerine sahiptir.

$$c = f + \Delta f \tag{2.15}$$

Cisim uzay koordinat sisteminde P noktasının konum vektörü,

$$P=\begin{vmatrix} X_p - X_0 \\ Y_P - Y_0 \\ Z_P - Z_0 \end{vmatrix} \quad \text{olur.} \tag{2.16}$$

Kolinearite eşitliğine göre, p resim ve P cisim vektörleri arası fiziksel durum,

$$P= k.D.P \tag{2.17}$$

$$\begin{vmatrix} x_p - x_0 \\ y_P - y_0 \\ 0 \quad -c \end{vmatrix} = k.D. \begin{vmatrix} X_p - X_0 \\ Y_P - Y_0 \\ Z_P - Z_0 \end{vmatrix} \tag{2.18}$$

şeklinde ifade edilir. Burada k ölçek faktörü olup,

$$k = \frac{p}{P} \tag{2.19}$$

olarak yazılır.

Resim koordinatlarının bilinen değerler olması durumunda cisim koordinatları,

$$P = \frac{1}{k} D^T p \tag{2.20}$$

$$\begin{vmatrix} X_p - X_0 \\ Y_P - Y_0 \\ Z_P - Z_0 \end{vmatrix} = \begin{vmatrix} 1 \\ D^T \\ k \end{vmatrix} . \begin{vmatrix} x_p - x_0 \\ y_P - y_0 \\ -c \end{vmatrix} \qquad (2.21)$$

k ölçek faktörünün, kolinearite eşitliklerinde her bir ışın için ayrı ayrı belirlenmesi gerekmektedir. Bu durumda oluşan matematiksel ifadelerden **k** ölçek faktörünün etkisi yok edilirse,

$$f_1 = x - x_0 + c.\frac{a_{11}(X - X_0) + a_{12}(Y - Y_0) + a_{13}(Z - Z_0)}{a_{31}(X - X_0) + a_{32}(Y - Y_0) + a_{33}(Z - Z_0)} \qquad (2.22)$$

$$f_2 = y - y_0 + c.\frac{a_{21}(X - X_0) + a_{22}(Y - Y_0) + a_{23}(Z - Z_0)}{a_{31}(X - X_0) + a_{32}(Y - Y_0) + a_{33}(Z - Z_0)} \qquad (2.23)$$

fonksiyonel ifadeleri yazılır.

2.ENDÜSTRİYEL ÖLÇME SİSTEMİ İÇİN YAZILIM GELİŞTİRME

Bir önceki bölümde bahsi geçen fotogrametrik sistemin yazılım kısmında belirtilen tez konusu olarak geliştirilen fotogrametrik yazılım Matlab ortamında geliştirilmiştir. Bundaki temel etkenler Matlab'ın mühendislik uygulamalarındaki yadsınamaz gücü, işlevselliği ve kullanışlılığı ile beraber yine Matlab'ın GUI (Graphical User Interface – Grafik Kullanıcı Arabirimi) geliştirme ortamının projenin hem programlama hem de kullanım aşamalarında görsel ve kolay kullanılabilir olmasıdır. Böylelikle hazırlanan uygulama çoğu bilgisayar kullanıcısının aşina olduğu Windows GUI (görsel) tabanlı bir sistem özelliği taşımaktadır.

GUI nesneleri menüler, araç çubukları, radyo butonlar, liste kutuları veya kaydırıcılar olabilir. Bunların yanında MATLAB GUI ile MATLAB'IN sunduğu hesaplama imkânları kullanılarak da veri alımı ve grafik çizimi gibi pek çok işlem gerçekleştirilebilir.

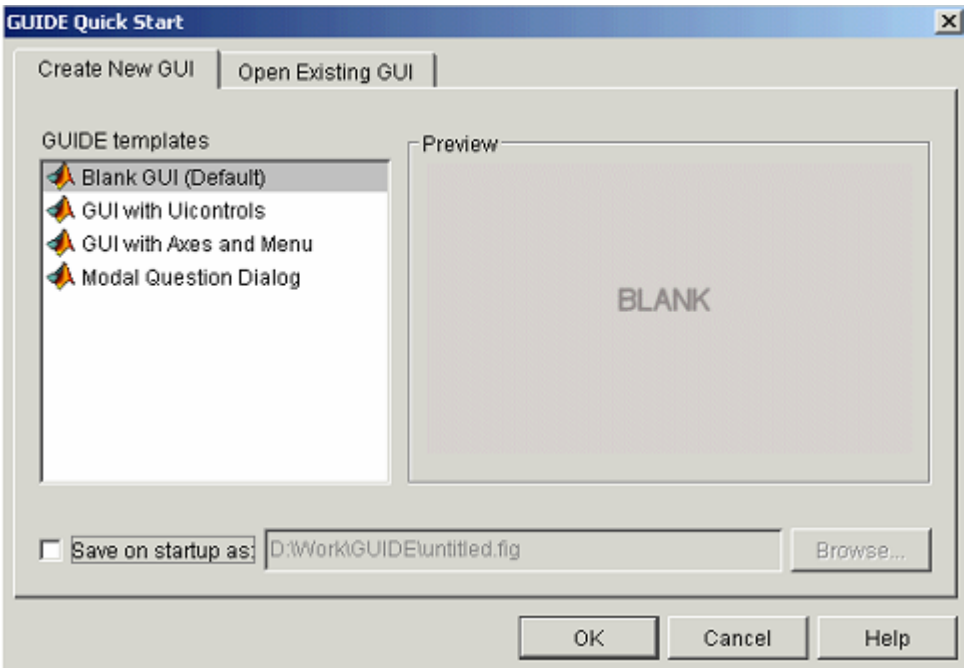
Her bir GUI nesnesi (veya komponent) GUI için tanımlanan programlama dosyasında callback diye adlandırılan ayrı alt rutin programlama parçalarına sahiptir. Bu şekilde her bir nesnede oluşan olaylara (örnek olarak bir buton nesnesinin tıklanması ile click event (tıklama olayı) oluşması gibi) GUI o olaya ait callback rutinlerini icra ettirir. Yani, GUI hem bir arayüz hem de bir program çağrılarını icra ettirme mekanizması olarak çalışır. (Uzunoğlu M.,2003)

Yukarıda bahsedilen programlama olay tabanlı programlama diye adlandırılır. Bu tür programlamada her bir olaya ait alt program parçaları birbirinden bağımsız olarak MATLAB GUI tarafından çalıştırılır.

2.1 MATLAB GUIDE Aracı ile GUI Tasarımı Oluşturma

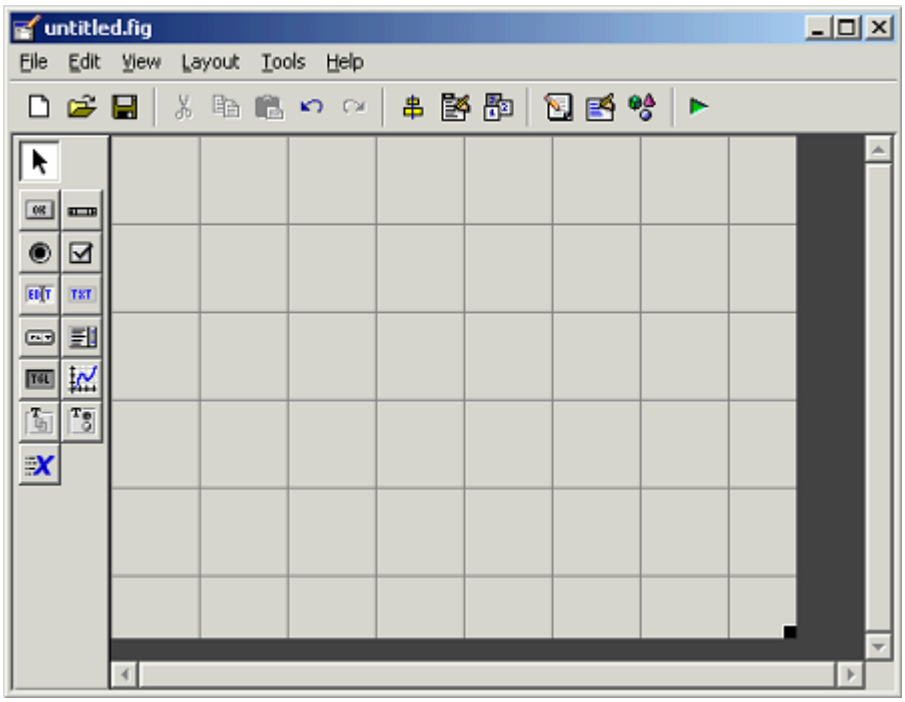
GUIDE Matlab'ın GUI tasarımcılarına sunduğu içerisinde çeşitli araçlar içeren ve kolaylık sağlayan bir grafiksel GUI geliştirme ortamıdır. GUIDE kullanılarak tıkla ve sürükle-bırak tekniği ile GUI arayüzüne nesneler (örneğin butonlar, text kutuları, liste kutuları, grafikler v.s.) kolaylıkla eklenebilir. Ayrıca, eklenen nesnelerin hizalanması, tab sırasının değiştirilmesi, görsel ayarlar üzerinde manipülasyonlar yapılması da bu ortamın tasarımcılara sunduğu imkânlardan bazılarıdır. (Uzunoğlu M.,2003)

Bu aracı çalıştırmak için ya MATLAB komut satırından GUIDE komutu verilir ya da Start düğmesi tıklanarak MATLAB/GUIDE komutu verilir. Bu adımdan sonra Şekil 2.1'deki gibi bir pencere gelir.



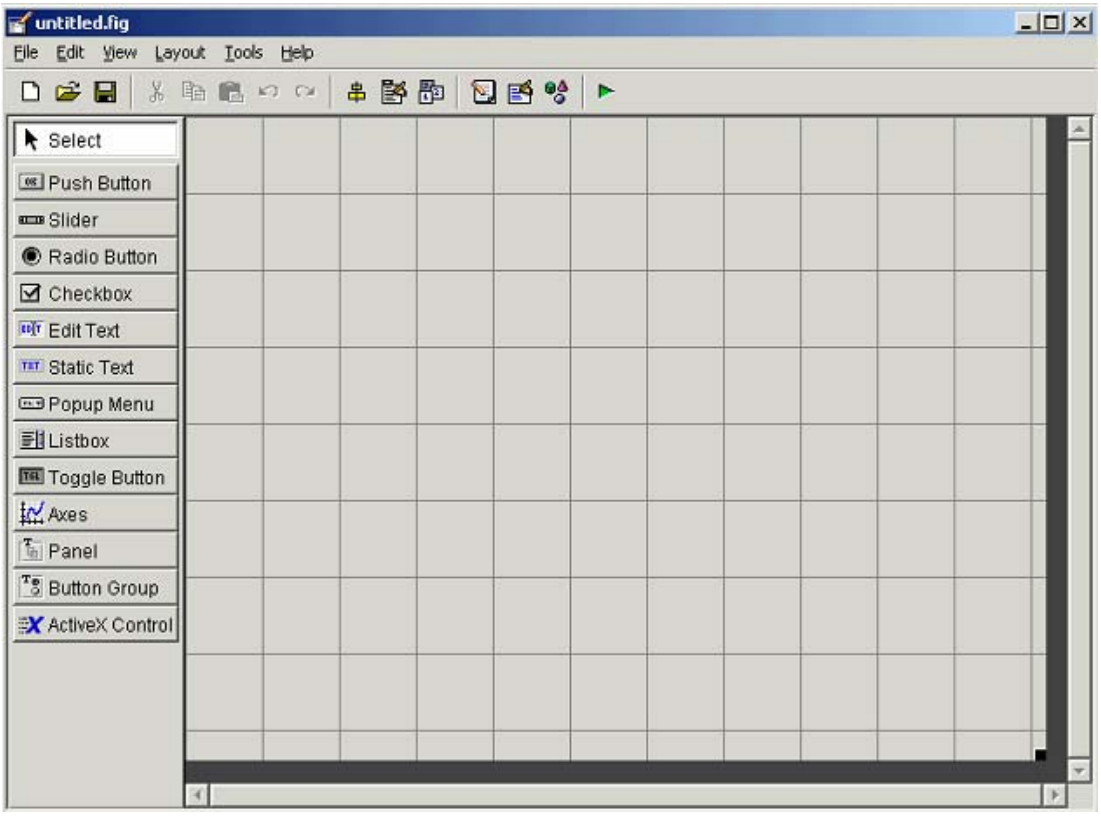
Şekil–2.1 GUIDE Açılış Ekranı

Bu pencereden eğer yeni bir GUI tasarımı yapılacaksa Blank GUI seçeneği seçilir. Şayet önceden yapılmış bir tasarım açılmak isteniyorsa Open Existing GUI sekmesinden sonra istenilen dosya seçilir. Burada yeni bir tasarım oluşturulacağını kabul edildiğinde şöyle hareket edilir. Bundan sonra OK düğmesi tıklanılarak Şekil 2.2'teki GUIDE LAYOUT Editor (GUIDE Görünüm Çalışma Alanı) penceresine ulaşılır.



Şekil–2.2 GUIDE Layout Editor Ekranı

Bu adımdan sonra File/Prefences/Guide yolu kullanılarak gelen pencereden "Show names in component palette" seçeneği tıklanıp OK düğmesine basılır.Bu durumda Şekil 2.3'teki gibi bir pencere gelecektir.



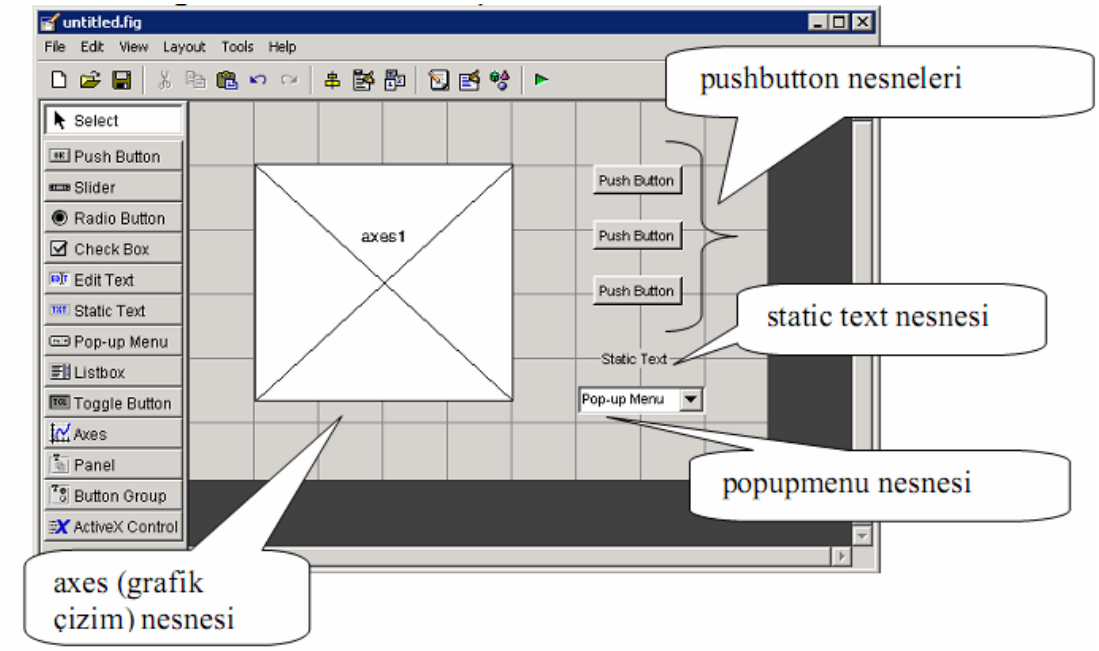
Şekil–2.3 GUIDE Komponent Açıklamaları

2.1.1 Komponentleri Çalışma Alanına Ekleme

Bunun için sol tarafta bulunan nesne butonlarından istenilen nesneye ait buton tıklanır ve daha sonra çalışma alanında uygun görülen bir noktaya tıklandığında o noktayla ilgili nesne eklenmiş olacaktır. İstenirse çalışma alanındaki bir nesne farenin sol tuşu ile tıklanıp bırakılmadan çalışma alanının herhangi bir yerine sürüklenebilir.

Çalışma Alanının Boyutlarını Değiştirmek

Burada da çalışma alanının sağ alt tarafında bulunan siyah karenin üzerine fare işaretçisi getirilir ve fare işaretçisi konum değiştirdiğinde farenin sol tuşu basılı tutularak çalışma alanı istenilen boyutlarda olacak şekilde düzenleme yapılabilir.



Şekil–2.4 GUIDE Genel Kullanıma Yönelik Bir Örnek

Burada GUI arayüzünde

- Bir adet grafik çizim (axes) nesnesi,
- Bir adet popup menü,
- Bir adet popup menü başlığı sunan statik text nesnesi,
- Üç adet buton nesneleri

yer almaktadır.

Nesnelerin özellikleri değiştirilmek istenirse ya ilgili nesne farenin sol tuşu ile çift tıklanır ya da ilgili nesne önce seçilip daha sonra View/Property Inspector komutu ile özellikler penceresi açılır.

2.1.2 GUI Tasarımını Kaydetme ve Çalıştırma

Bundan sonra bitmiş olan bu GUI arayüzü çalıştırarak görmek için öncelikle Tools/Run yolundan Run (Çalıştır) komutu verilir. Daha sonra gelen pencereden çalışmanın Run edilebilmesi için kaydedilmesi gerektiğini bildiren bir pencere çıkar Burada Yes butonuna basılır. Bu adımdan sonra MATLAB GUIDE tasarımın kaydedileceği dosya ismini soran bir pencere getirir. Bu pencereden çalışmaya bir

isim vererek tasarıyı kaydetmiş oluruz. Ardından “Change the MATLAB Directory” gibi bir ekran gelirse burada bu ekran OK tuşuna basılarak kapatılabilir. Bu ekranda kaydedilen dosya MATLAB tanımlı dizinler dışında bir yere kaydedilme söz konusu olduğunda kullanıcı uyarılmaktadır. Sonra da GUI tasarımının çalışması sonucu gözükecek uygulama penceresi gözükecektir.

2.2 Matlab GUI Nesneleri

MATLAB GUIDE aracı kullanarak boş (blank) bir GUI çalışma ekranı açıldığında sol tarafta görülen component panelde pek çok nesnenin kullanılabileceği görülmektedir.

Şimdi bu nesnelerin sırasıyla özellikleri ile ilgili bilgiler verilecek ve nasıl programlanacağı gösterilecektir.

Push Button:

Normal bir buton özelliği taşımaktadır. Bir buton üzerine tıklanması ile yapılacak komutlar bu buton ile ilgili callback’lerin altına yazılır.

Toggle Buton:

Çift durumlu bir buton özelliği taşıyan bu nesne ile iki farklı seçenek içeren durumlarda örneğin bu buton basılı ise bir işlemin, bu buton basılmamış ise başka işlemlerin yapılması gerektiği yerlerde tercih edilen bir nesnedir. Buton grubu nesnesi ile beraber kullanımı tavsiye edilir.

Radio Buton:

Birden fazla seçeneğin olduğu, ancak seçeneklerden sadece herhangi birinin seçilebileceği hallerde bu nesne kullanılır. Buton grupları ile kullanılması genellikle tercih edilir.

Check Box:

Kullanıcıya seçim yapabileceği ve birden fazla şıkkı işaretleyebileceği durumlarda bu nesne kullanılır.

Edit Text:

Bir kullanıcıdan bilgi girişi ya da bir değerin alınması söz konusu olduğunda giriş elemanı olarak sıklıkla kullanılan bir nesnedir.

Static Text:

Kullanıcıya herhangi bir bilgi verme ya da bulunan bir sonuç veya değeri gösterme amacıyla sıklıkla kullanılan bir nesnedir.

Slider:

Kullanıcıdan bir giriş değerini kaydırılmak suretiyle kolaylıkla alınmasına imkân veren bir nesnedir.

List Box:

Kullanıcıya bilgi verme amacıyla kullanılabileceği gibi bir değeri listeden seçmek amacıyla da kullanılan sabit bir liste kutusu niteliğinde kullanılan bir nesnedir.

Pop-Up Menu:

Kullanıcıdan alınmak istenilen bilgileri açılan bir listeden seçme özelliği taşıyan bir nesnedir.

Axes:

Yapılan iş ile ilgili grafik çizimlerinin kullanıcıya gösterilmesini sağlayan bir nesnedir.

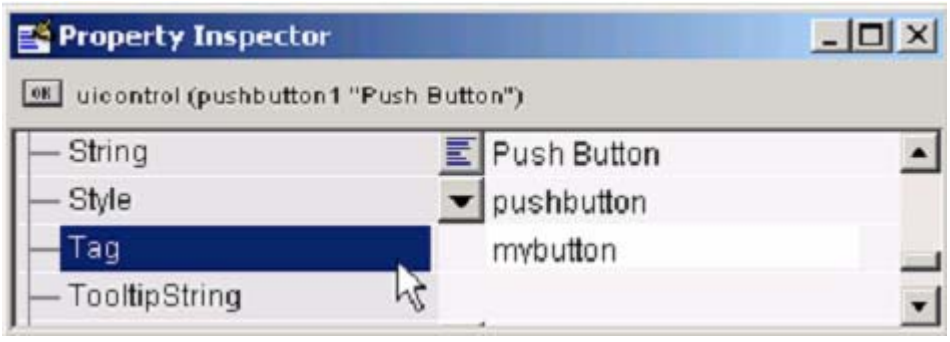
Panel:

GUI yüzeyi nesnelerinin kullanıcıya daha anlamlı ve güzel gözükmesini sağlayan, ayrıca tasarımcıya GUI dizaynında kolaylık sunan bir nense olup, GUI yüzeyi nesnelerinin gruplanması ve bir arada gösterilmesi amacıyla kullanılır.

Button Group:

Radio veya toggle tipteki buton nesnelerinin bir arada kullanılarak kullanıcının birden fazla seçenekten sadece bir tanesini seçmesini sağlamak amacıyla kullanılan bir nesnedir.

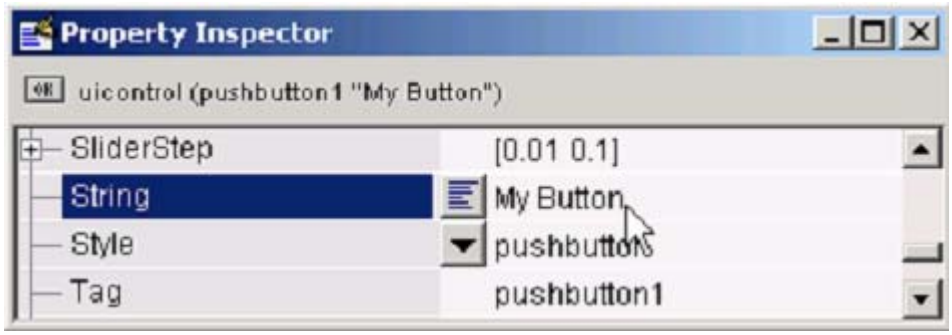
Bir nesneye tanıtıcı ve o nesneye özel bir isim vermek için öncelikle o nesne farenin sol tuşu ile GUI tasarım yüzeyinde seçilir. Daha sonra View menüsünden Property Inspector komutu verilir. Gelen özellikler penceresinden nesnenin Tag özelliğine istenilen bir isim verilebilir. Ayrıca, Property Inspector penceresini açmak için nesne üzerinde farenin sol tuşu ile çift tıklanılabilir. Şekil 2.5'de örnek olarak bir buton nesnesi için bu durum görülmektedir.



Şekil–2.5 GUIDE Tag (Tanıtıcı İsim) Ataması

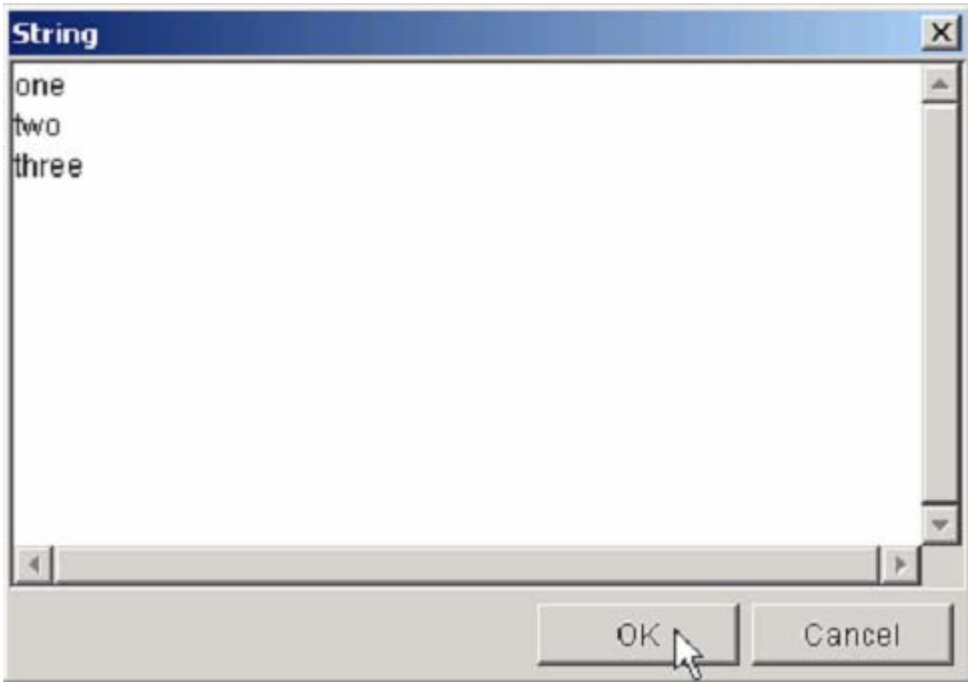
Bazı nesneler özellikleri gereği kullanıcıya bilgi vermek veya bir seçenek sunmak amacıyla string bilgiler içerirler. Bu nesneler Push Button, Toggle Button, Radio Button, Check Box, Text, List Box, Popup Menu, Panel ve Button Group nesneleridir. Bu nesneler de yapıları gereği değişik özellikler içermektedir ve bu özellikleri üzerinden string değerler atamak mümkün olmaktadır. Bu nesnelere yazı bilgilerinin nasıl verildiği aşağıda sırasıyla açıklanmıştır.

Nesnelere yazı eklemek için ilgili nesne seçilerek özellikler penceresinden String özelliklerine istenilen bir metin bilgisi girilebilir. Ayrıca, bu özellik programlama komut satırlarıyla da değiştirilebilir. Örnek olarak Şekil 2.6'da görülen örnekte bir push butonun üzerindeki yazının değiştirilmesi gözükmektedir. Eğer ki bu nesnelere alt alta olacak şekilde birden fazla satır içeren bilgiler girilmek istenirse bir sonraki başlık altında anlatılan teknik kullanılmalıdır.



Şekil–2.6 GUIDE Nesnelere yazı eklemek

Bu nesneler çoklu string bilgiler içeren liste kutusu tarzı yapılardır. Bu nesnelere string bilgiler eklemek istenirse öncelikle nesne seçilir. Ardından String özelliğinin yanında yer alan butona tıklanır. Karşımıza aşağıdaki gibi bir pencere gelecektir. Bu pencereye gerekli bilgiler girildikten sonra OK butonuna basılır. Böylece string verilerin girilmesi işlemi tamamlanmış olur. İstenirse bu özellik programlama yoluyla da değiştirilebilir. Bu yöntem başlıkta belirtilen nesnelere birden fazla bilgi girilmesi istendiğinde bu nesneler için de kullanılabilir.

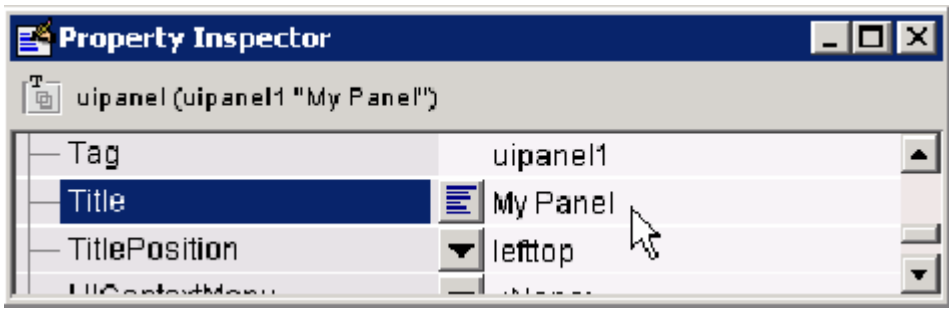


Şekil–2.7 GUIDE Çoklu String İçeren Nesnelere String Ataması(List Box ve Popup Menu)

Şekil 2.7'de görünen bilgiler girildiğinde örnekteki popup menü nesnesinin görüntüsü Şekil 2.8'de görüldüğü gibi olacak ve kenarındaki buton tıklandığında da tüm seçenekler kullanıcıya sunulacaktır.

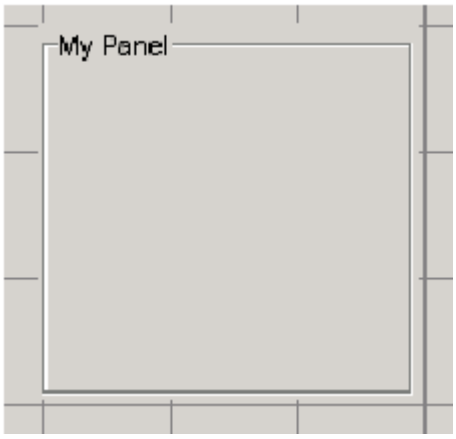
Şekil–2.8 GUIDE Popup Menu

Bu nesneler içinde yer alan pek çok nesne gruplandırma imkânına sahip olup bu nesnelere başlık eklemek için farklı bir özellik kullanılmaktadır. Bu nesnelere başlık ekmek için nesne seçili iken Property Inspector penceresinden Title özelliğine gerekli bilgi girilmelidir. Bu durum Şekil 2.9'da görülebilmektedir.



Şekil–2.9 GUIDE Nesneye (Panel) Başlık Verilmesi

Örnekte bir panel nesnesi kullanılmış olup başlığı My Panel olarak değiştirilmiştir. Bu değişiklik yapıldıktan sonra panelin GUI çalışma alanındaki görüntüsü Şekil 2.10'daki gibi olacaktır.



Şekil–2.10 GUIDE Panel Görünümü

GUI Çalışma Alanında Nesneler ile Çalışma

GUI yüzeyindeki nesnelere tasarım ortamında istenildiği şekilde müdahale edilebilir. GUI çalışma alanındaki nesneler üzerinde kopyalama, silme, taşıma, öne getirme, en arkaya gönderme, hizalama, tab tuşu ile seçim sırasının değiştirilmesi, başka bir noktaya taşınması veya boyutlarının değiştirilmesi, cetvel ve ızgara kullanılarak işlemlerin yapılması, GUI uygulamasında ana menü oluşturmak ya da istenilen bir nesne üzerinde context menü oluşturmak, GUI uygulamasına araç çubuğu eklemek, GUI tasarımında kullanılan nesnelerin görülmesi gibi pek çok işlem için GUIDE tasarımcıya pek çok kolaylık sağlamaktadır.

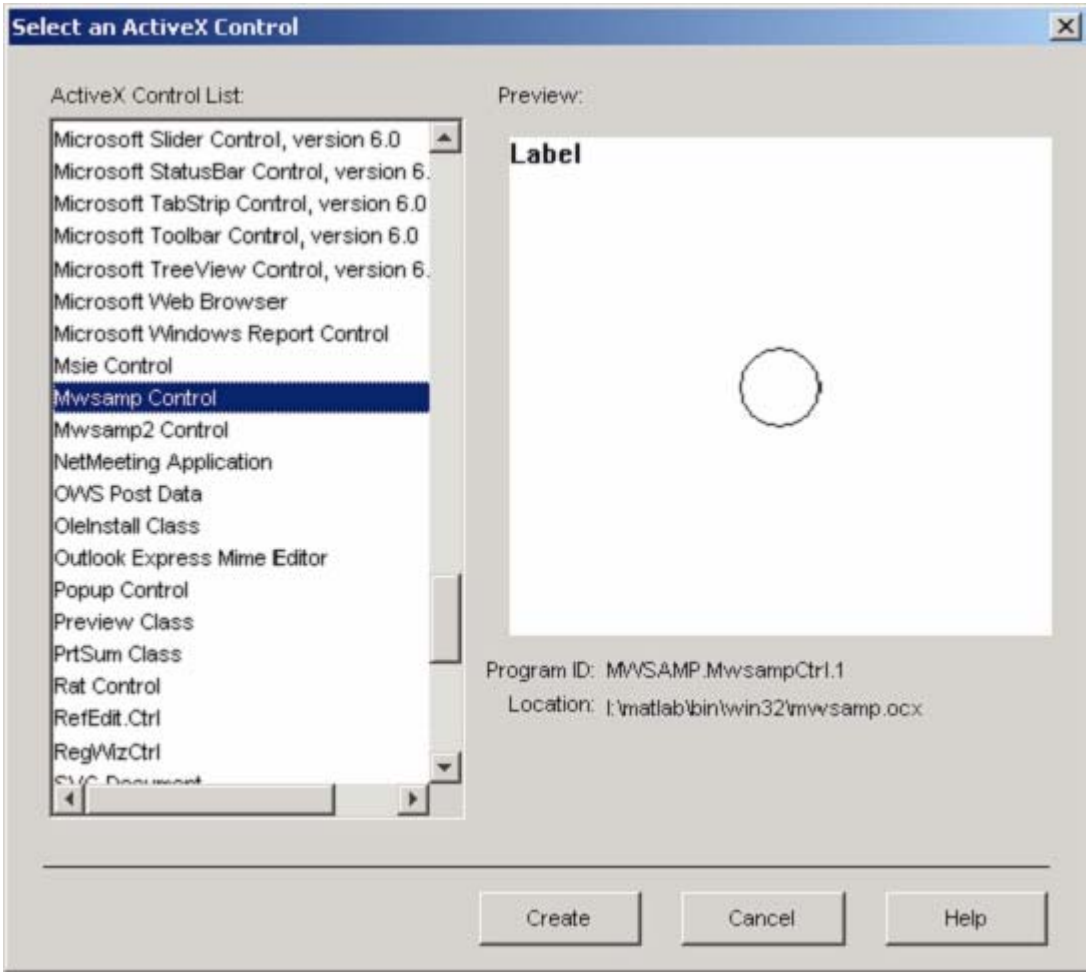
2.3 ActiveX Component-ActiveX Kontrol – OCX:

Matlab Gui tasarımları sadece yukarıda belirtilen nesneler ile sınırlı değildir. Matlab Gui tasarımcının veya programcının ayrıca, ActiveX adı verilen ve değişik alternatifi olan nesneleri kullanmasına da imkân verir. Böylece hem tasarımcı hem tasarlanılacak GUI arayüzünün kullanımı bakımından kullanıcıya esneklik sağlanmış olunur.

Bu özellik sayesinde belirli bir işlem için özel olarak hazırlanmış nesneler Matlab uygulamasına dâhil edilip istenilen şekilde konumlandırılabilir. ActiveX onları oluşturmakta kullanılan programlama dilinden bağımsız olarak belirli görev veya görevleri yapan yeniden kullanılabilir yazılım bileşenlerini tanımlayan bir frameworktur.

Birçok Windows uygulaması ki buna Internet Explorer, Microsoft Office, Microsoft Visual Studio ve Windows Media Player da dâhildir; hem ActiveX kontrollerini kullanır hem de bünyelerindeki fonksiyonları başka uygulamalarda kullanım için ActiveX şeklinde sunarlar.

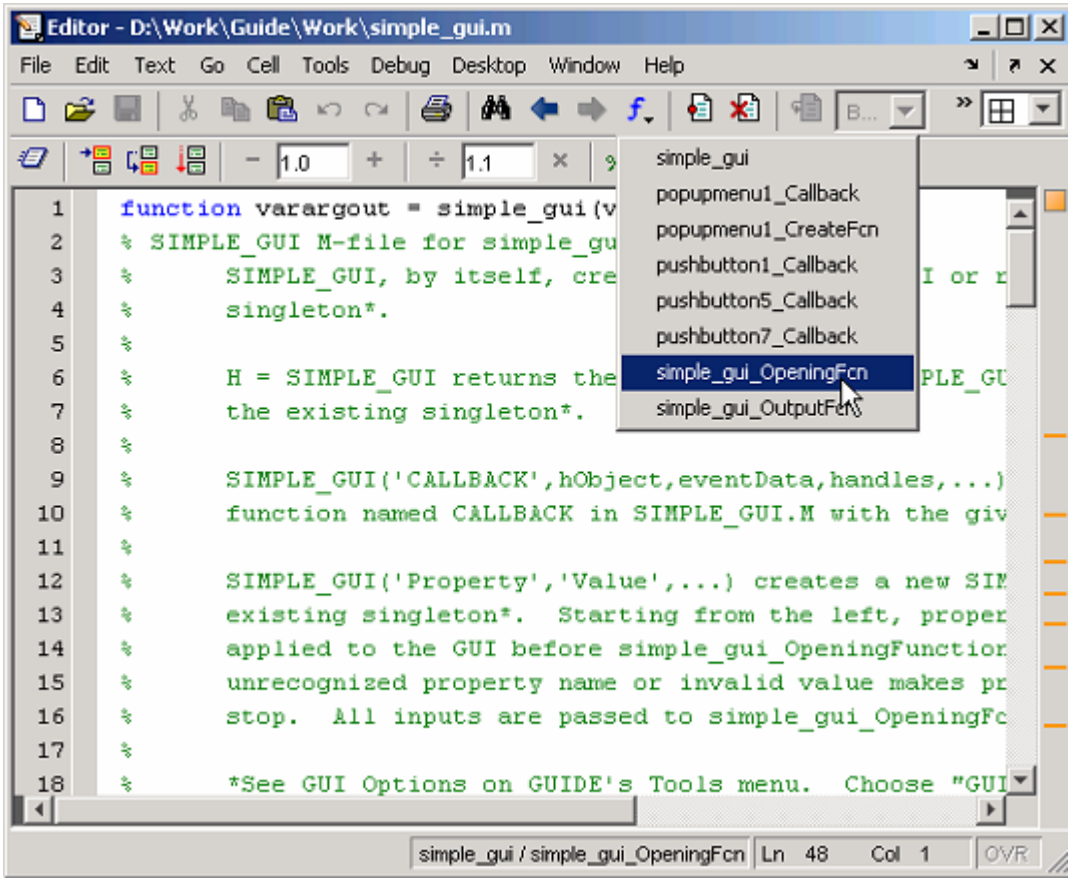
Bir ActiveX nesnesini GUI yüzeyine eklemek için öncelikle component panelden seçilir. Daha sonra GUI alanında yerleştirilmesi düşünülen bir yere farenin sol tuşu ile tıklanır. Bu adımda yeni gibi bir pencere gelecektir. Bu pencerede GUI yüzeyine yerleştirilmek istenen ActiveX componenti sol taraftaki listeden seçilmeli ve ardından Create butonuna basılmalıdır.



Şekil–2.11 GUIDE ActiveX Control Seçimi

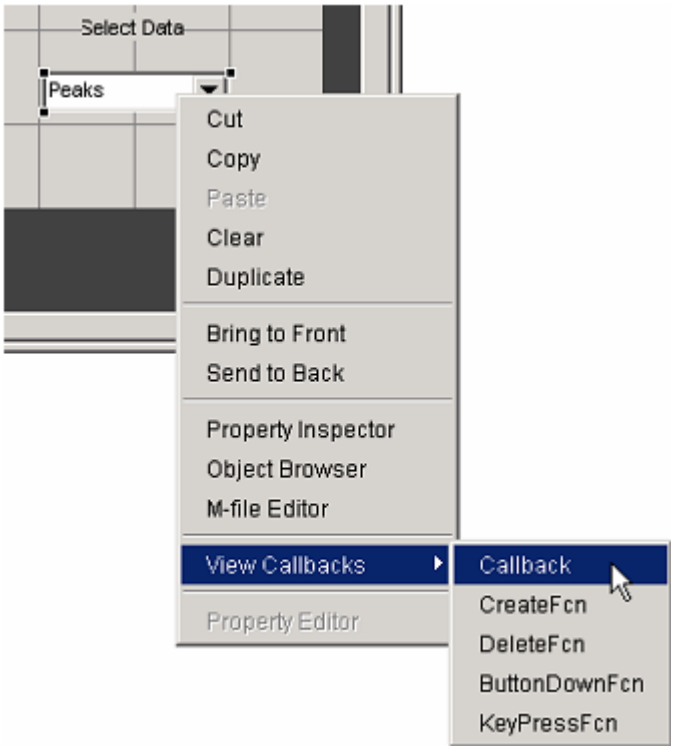
2.4 GUI Arayüzlerine Program Yazma

Bir GUI arayüzünün programlanması demek çalışmanın kaydedildiği isimle aynı zamanla ve aynı adda oluşturulan ".m" uzantılı dosya içerisine kodlama satırlarının eklenmesi demektir. Bu dosyanın içini görebilmek, değişiklik yapabilmek için GUIDE çalışma ekranı penceresinden View/M-File Editor komutu işletilebilir. Ardından karşımıza Şekil 2.12'deki gibi bir pencere gelecektir.



Şekil–2.12 GUIDE Arayüzün Programlanması

Şekil 2.12'deki pencerede hazırlanan GUI tasarımına ait kodlar gözükmektedir. Burada pek çok kodun hazır eklenmiş olduğu görülecektir. Bu kodlar otomatik olarak MATLAB GUIDE tarafından eklenmiştir. Burada ilgili butonlara ve liste kutularına ya da istenilen bir nesneye ait callback isimli alt program parçalarına ilgili kodları yazılır. Bir nesneye ait callback'in bulunduğu satıra gitmek için araç çubuğunda yer alan "f" simgeli butona tıklanır ve açılan listeden ilgili nesneye ait callback'in ismi seçilir. Bu durum yukarıdaki pencerede de görülmektedir. Ayrıca, GUIDE çalışma ekranından da direk istenilen bir callback satırına gidilebilir. Bunun için ilgili nesne üzerinde sağ tıklanır ve açılan pencereden "View Callbacks" menüsünden ilgili callback tıklanır ya da ilgili nesne seçilip View/View Callbacks yolu üzerinden açılan listeden gidilmek istenilen callback tıklanması yeterlidir.



Şekil–2.13 GUIDE Callback erişimi

Callbackler önceki konularda da bahsedildiği üzere oluşan herhangi bir olaya bağlı olarak her nesne için ayrı ve olayın türüne göre çalıştırılan alt program parçalarıdır.

Bir m file dosya yapısı gereği bir GUI uygulaması tasarımında da aynı dosya ismini taşıyan fonksiyon ismi ile başlayan bir m function yapısına sahiptir. Ancak, giriş ve çıkış varargout parametrelerinin dinamik olmasına dikkat edilmelidir. Yani, "varargin" deyimi ile giriş parametreleri hücre dizisi formatında birden fazla olabilir. Aynı, şekilde GUI uygulamasının kapatılacağı zaman aynı fonksiyon mantığı ile "varargout" ile dışarıya aktarılacak parametreler bu değişkene aktarılabilir.

Dışarıdan fonksiyon içerisine gönderilen giriş parametreleri ile ilgili bilinmesi gereken iki değişken vardır. Bunlar nargin ve varargin değişkenleridir.

- nargin değişkeni: Fonksiyona (ya da GUI uygulamasına) dışarıdan gönderilen toplam parametre sayısını tutar. “Number of Function Arguments In” söz öbeğinin kısaltılmış halidir. Böylelikle fonksiyon kendisine gönderilen parametre sayısına göre farklı işlemler yapabilir. (İnan A.,2007)

- varargin değişkeni: Fonksiyona gönderilen parametrelerin alınmasını sağlar. Hücre dizisi yapısında olduğundan parametrelerin alınması için dizi indislerinin “{“ ve ”}” işaretleri arasında yazılması gerekir. Hücre yapısının tercih edilmesinin nedeni birçok türde verinin (integer, float, string) gruplanarak gönderilebilmesidir.

Fonksiyonunun içinden dışarıya GUI uygulaması sonlandırılırken gönderilen çıkış parametreleri ile ilgili bilinmesi gereken iki değişkeni vardır. Bunlar nargout ve varargout değişkenleridir.

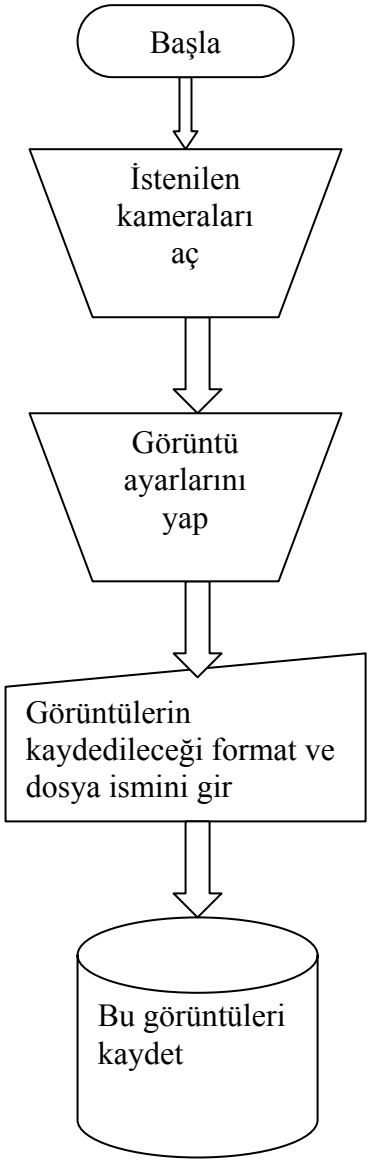
- nargout değişkeni: Fonksiyondan (ya da GUI uygulamasından) dışarı gönderilen toplam parametre sayısını tutar. “Number of Function Arguments Out” söz öbeğinin kısaltılmış halidir. Böylelikle fonksiyon kendisinden gönderilen parametre sayısına göre farklı işlemler yapabilir. (İnan A.,2007)

- varargout değişkeni: Fonksiyondan dışarıya parametrelerin gönderilmesini sağlar. Hücre dizisi yapısında olduğundan parametrelerin bu değişkene atanması sırasında dizi indislerinin “{“ ve ”}” işaretleri arasında yazılması gerekir.

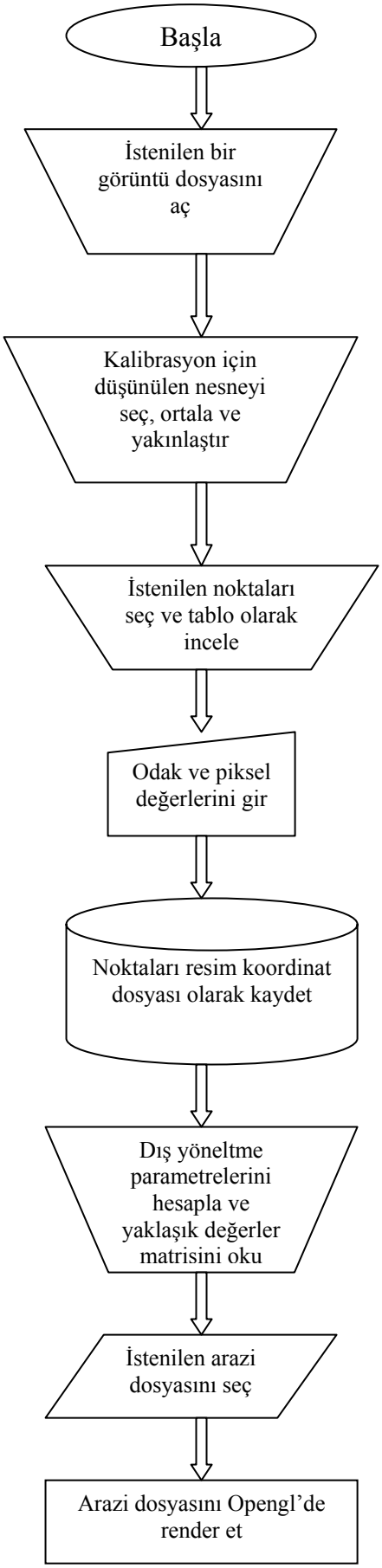
Bu dört komut kullanılarak bir fonksiyonun ne kadar sayıda girdi kabul edeceğini, ne kadar sayıda çıktı üreteceğini ve bunları hangi değişkenlerde saklayacağını, istenilenden az girdiyle çağrıldığında ne yapması gerektiğini belirleyerek fonksiyon gayet esnek ve hatasız çalışacak bir hale getirilebilir.

3. FOTOGRAMETRİK SİSTEM YAZILIM UYGULAMASI

Bu yazılım uygulaması iki temel arayüz üzerine inşa edilmiştir. Bunun sebebi resim alımı ile resim işlenmesinin iki farklı durum olmasıdır. Resim alımında çoklu kameralar kullanıldığı için amaç dört kameradan aynı anda istenilen bölgeye ait resimlerin elde edilmesidir. Fotogrametrik bölümde ise alınan resimler üzerinde tek tek fotogrametrik ölçme ve yöneltme işlemleri yapılmaktadır. Fotogrametrik sistem yazılımı çalışma şeması aşağıda verilmektedir.



Şekil–3.1 Fotogrametrik Sistem birinci temel arayüz çalışma şeması

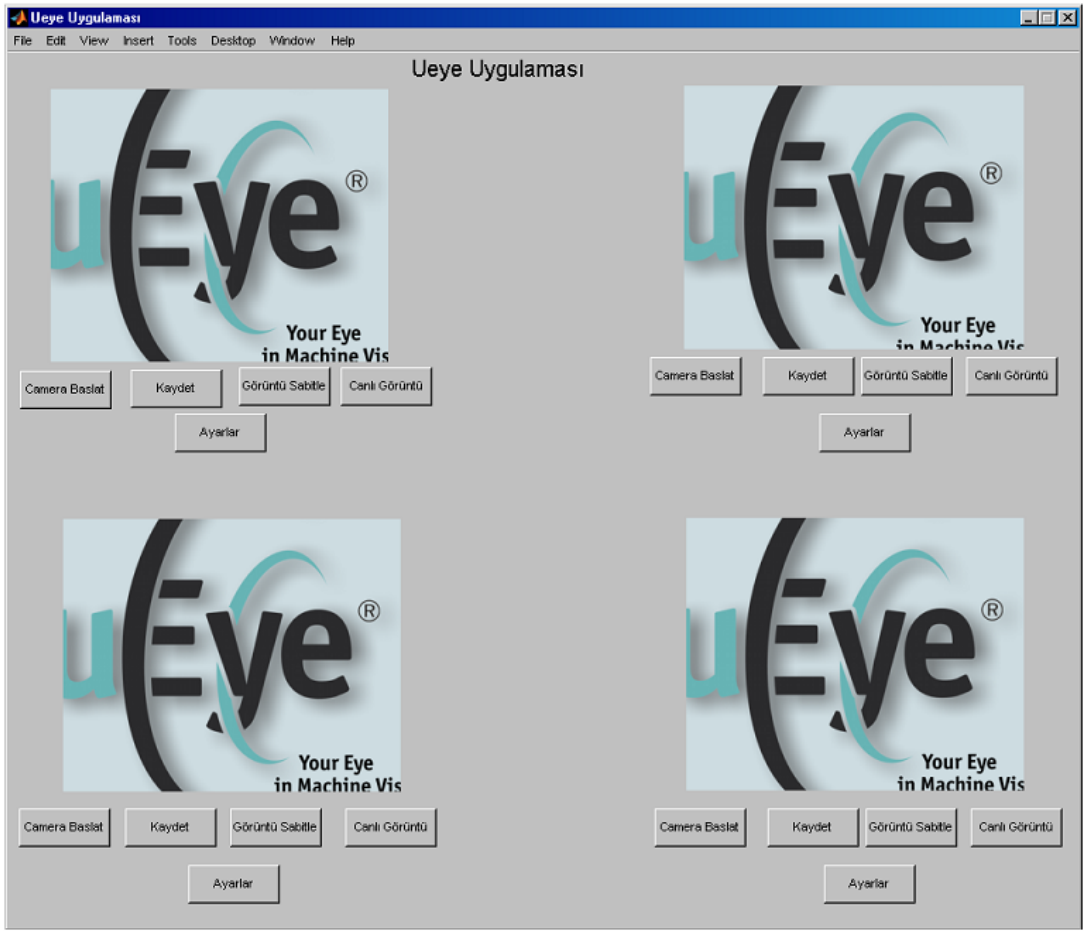


Şekil–3.2 Fotogrametrik Sistem ikinci temel arayüz çalışma şeması

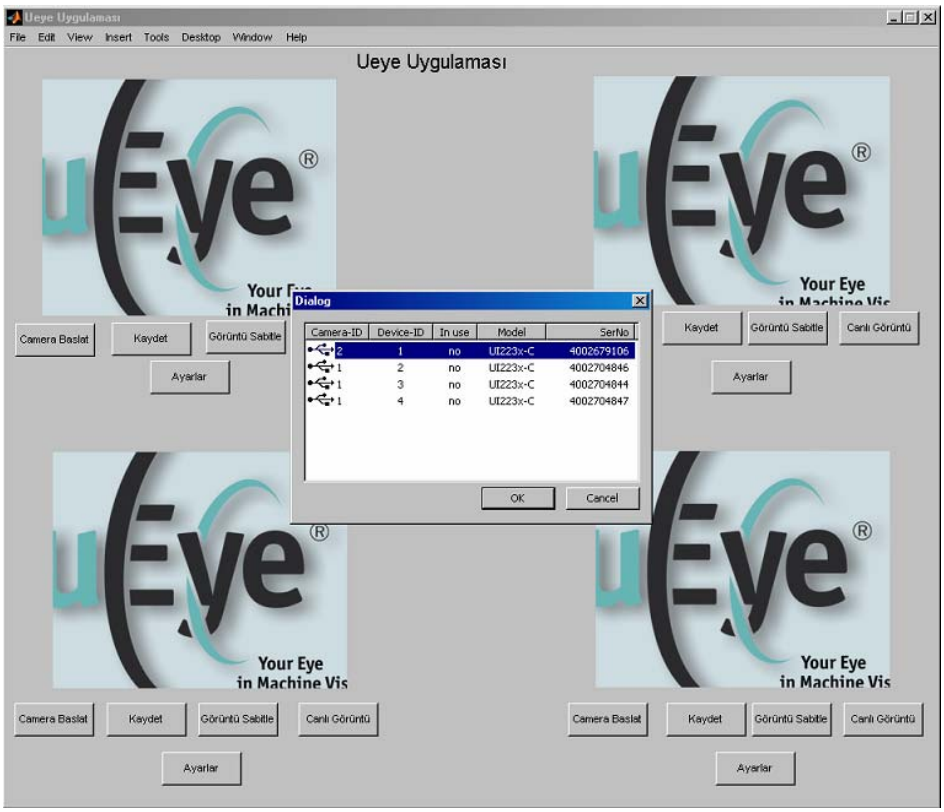
3.1 Çoklu Kamera Arayüzü

Bu çalışmada öncelikle yüzeyin resminin kameralarla elde edildiği kısımda kameralar tarafından desteklenen Ueye ActiveX kullanılarak özel kalibrasyonlu kameralardan görüntü alımı, ayarları ve kaydı sağlanmıştır.

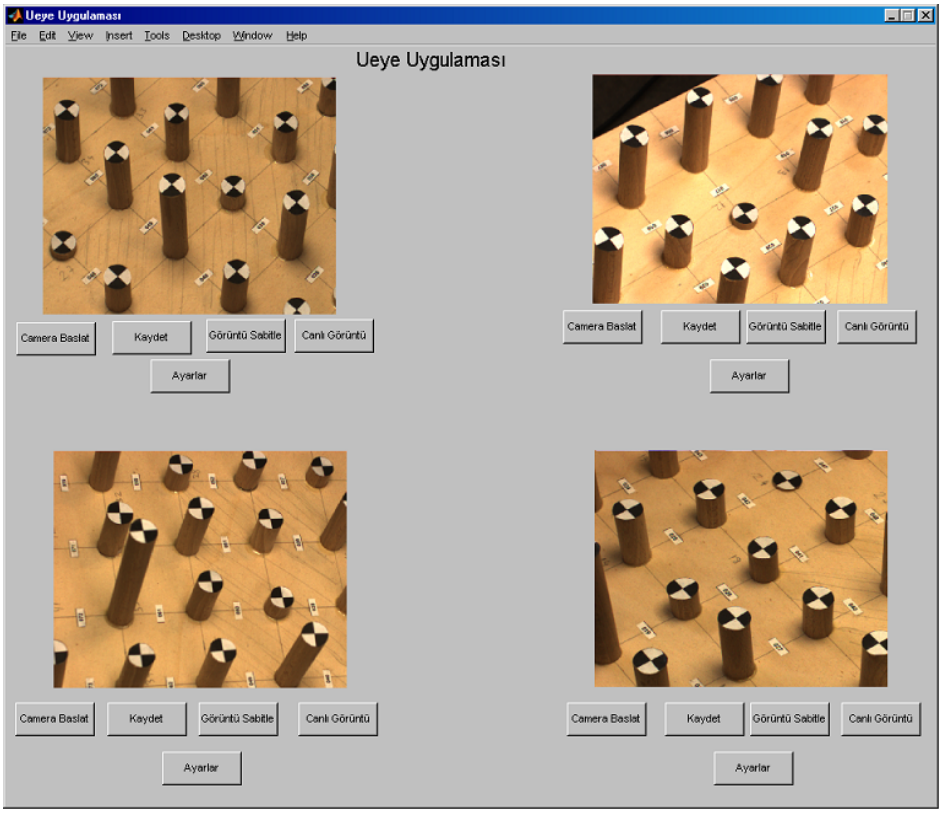
Aşağıdaki resimde çoklu kamera kullanımı için hazırlanan arayüz görülmektedir. Kameranın odak, lens, ışık, rgb veya parlaklık gibi birçok detayına "Ayarlar" butonundan erişilebilmektedir. O anki görüntü istenirse sabit veya canlı olarak işlenilmek üzere dosyaya kaydedilebilmektedir. Bu kayıt formatları resim standartları olan "JPEG, JPG, TIFF, BMP" olabilmektedir. Tabi canlı görüntüyü sabitleyerek kaydetmek olası görsel hatalara karşı bir avantaj sağlayabilmektedir.



Şekil–3.3 Çoklu kamera arayüzü



Şekil–3.4 Çoklu kamera arayüzünde istenilen bölgede istenilen kameranın açılması

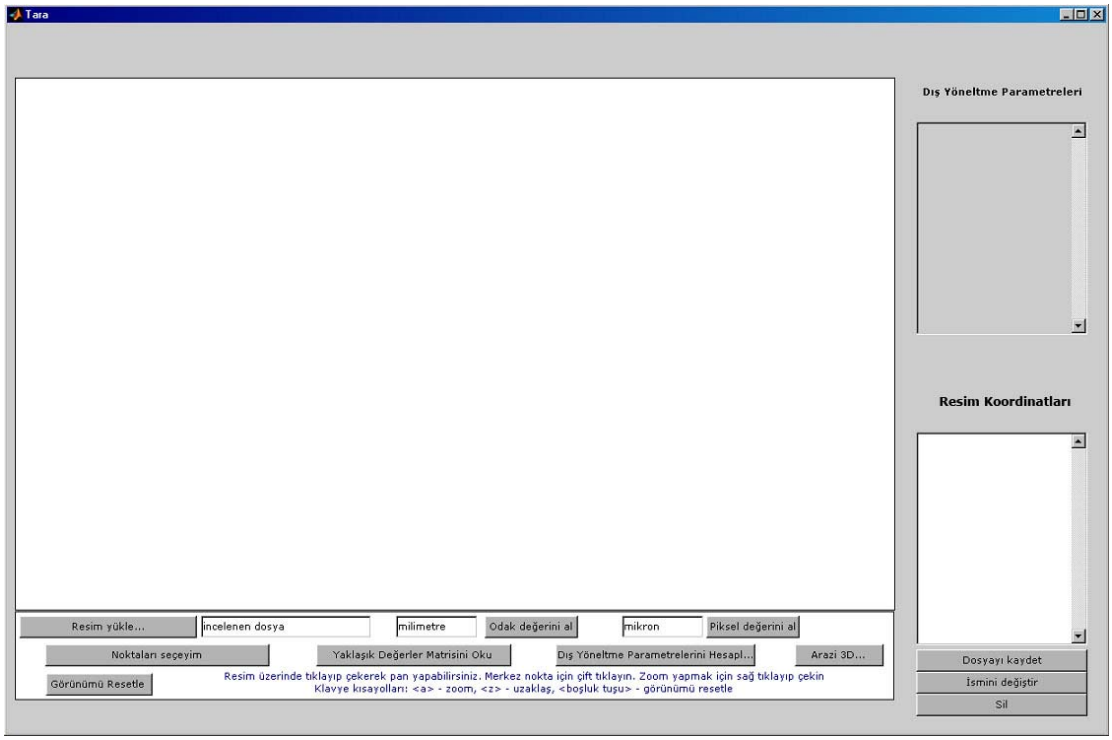


Şekil–3.5 Çoklu kamera arayüzünde 4 kameranın aktif halde çalışması

3.2 Fotogrametrik Arayüz Uygulaması

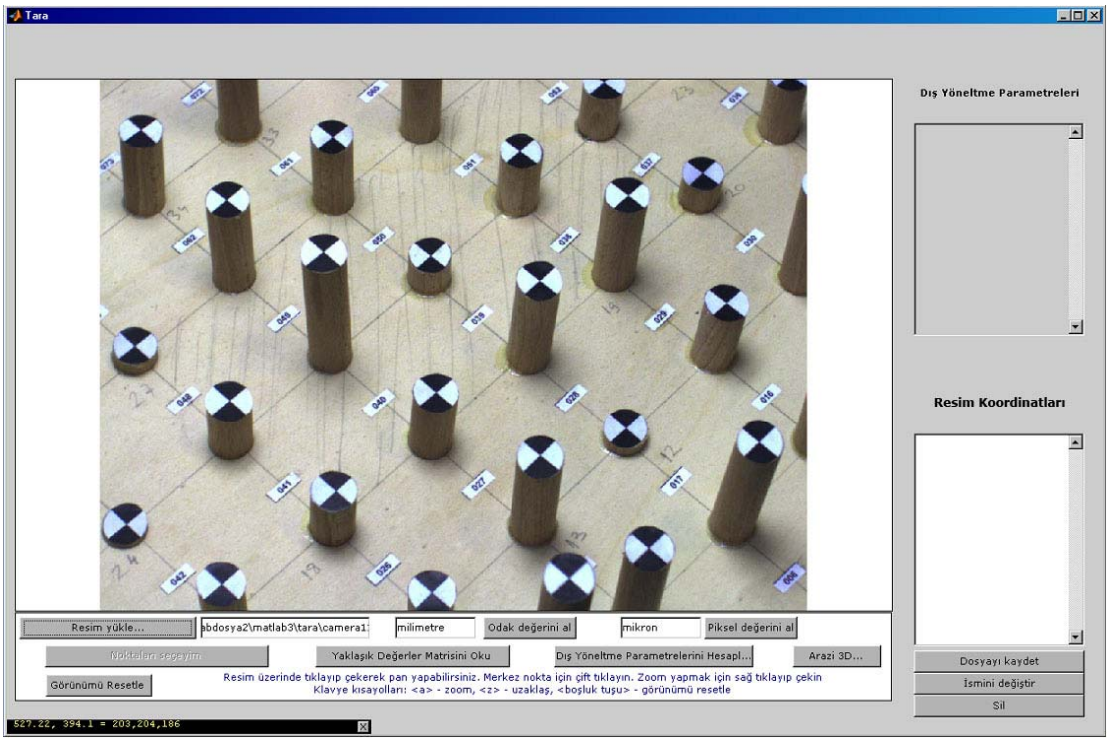
Bir sonraki arayüz ise kalibrasyon noktalarının belirlenerek “x,y” resim koordinat değerlerinin ölçülmesi ve aynı noktalara ait “X,Y,Z” cisim koordinat değerlerinin yardımıyla her bir resme ait dış yöneltme parametrelerinin elde edilmesi çalışmasıdır.

Bu işlemlerin yapılacağı arayüz aşağıda gösterilmiştir.



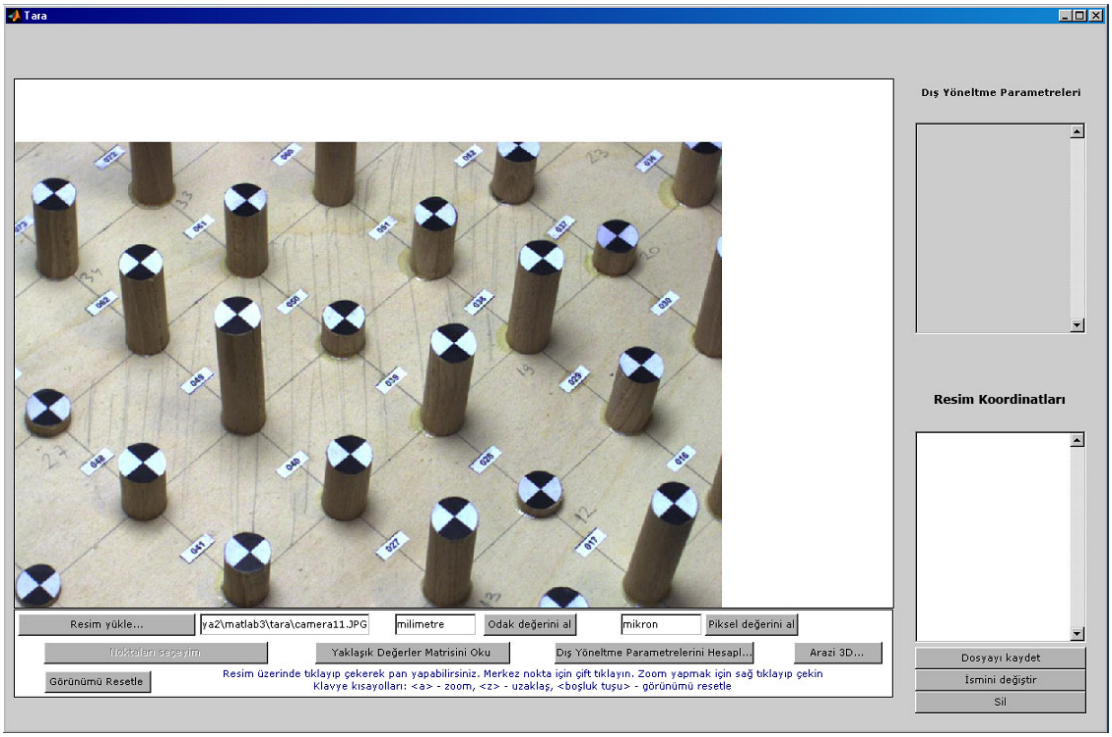
Şekil–3.6 Kalibrasyon ve Fotogrametrik Uygulama Arayüzü

Daha sonra öncelikle üzerinde işlem yapılmak istenen resim açılmıştır. Bu jpg ve benzeri standart dosya tiplerinde olabilir. Bu resim zaten daha önceki arayüzden elde edilen resimler arasından seçilebilir. Bu uygulamada kullanılan resimler 1024x768 çözünürlüğünde ve 32 bit JPG formatındadır.



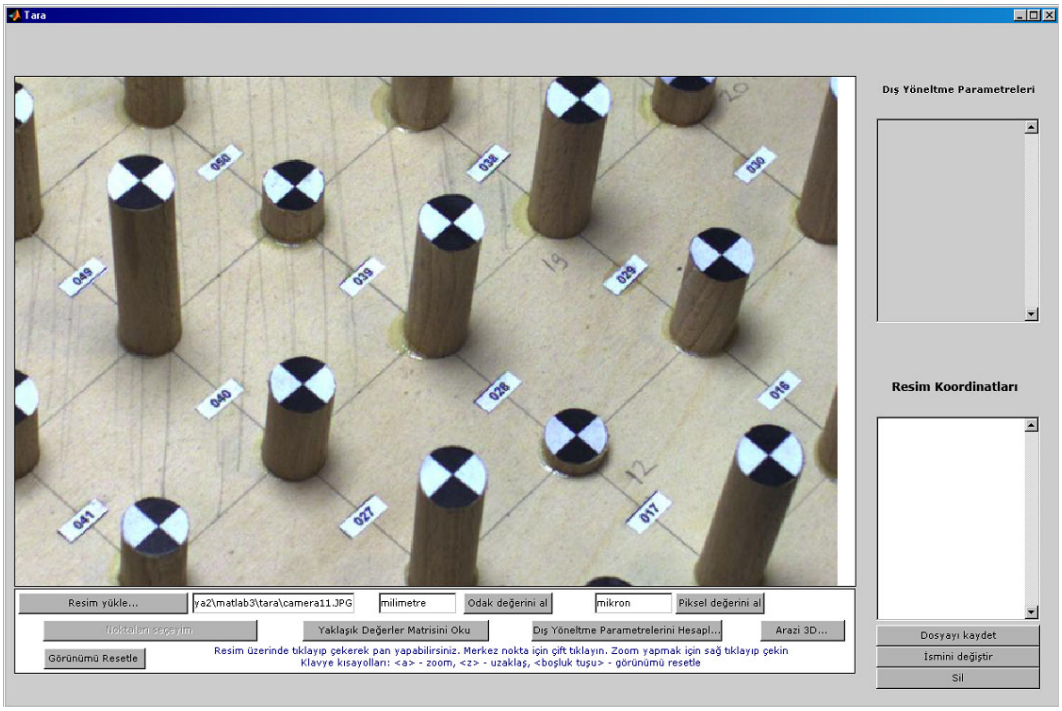
Şekil–3.7 Kalibrasyon ve Fot. Uyg. Arayüzünde bir resim dosyasının açılmış hali

Burada hassas nokta seçimi için resim üzerinde pan (yani kayma) hareketi yapılmıştır.



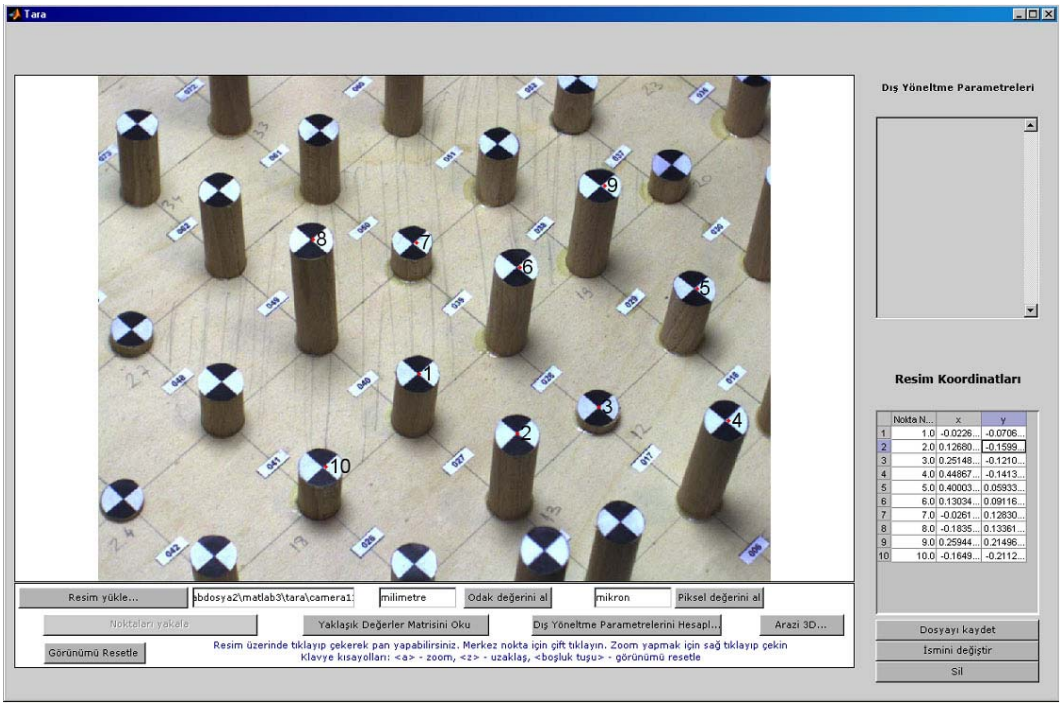
Şekil–3.8 Kalibrasyon ve Fot. Uyg. Arayüzünde resim üzerinde istenilen bölgeye pan (kaydırma) yapılması

Yine hassas nokta seçimi için resim üzerinde 28. noktaya zoom yapılmıştır.



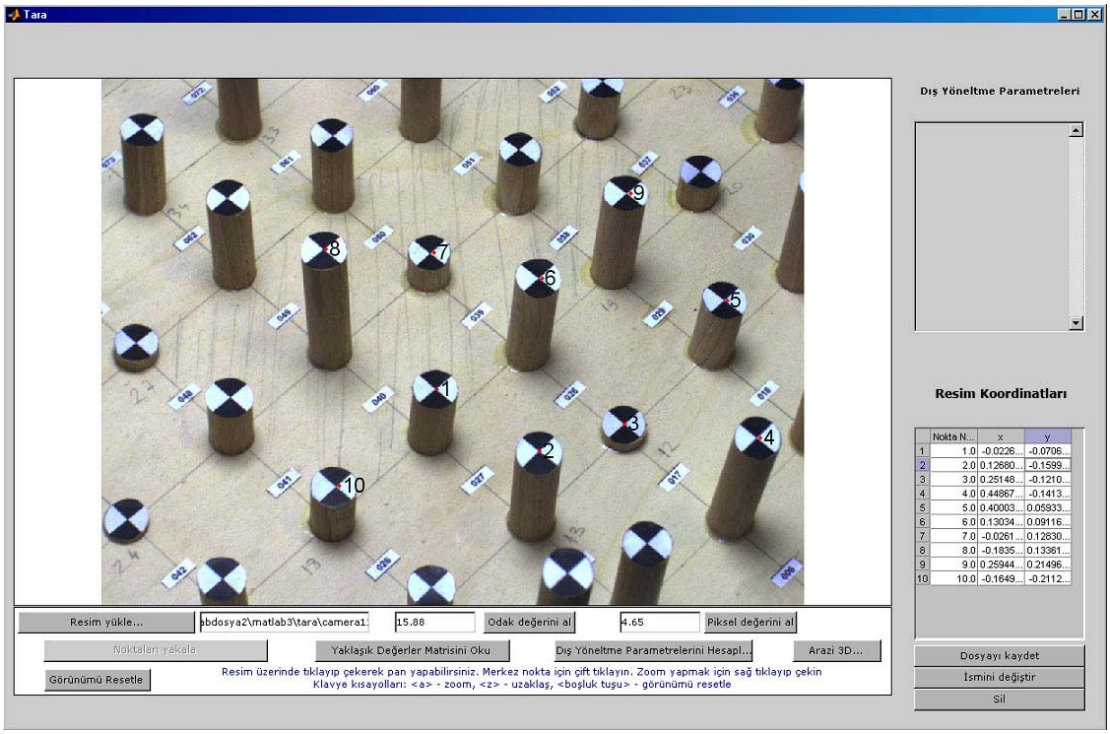
Şekil–3.9 Kalibrasyon ve Fot. Uyg. Arayüzünde resim üzerinde istenilen bölgeye zoom yapılması

Resimde seçilen kırmızı noktalar yan tarafta x ve y değerleriyle beraber bir matris haline getirilmiştir.

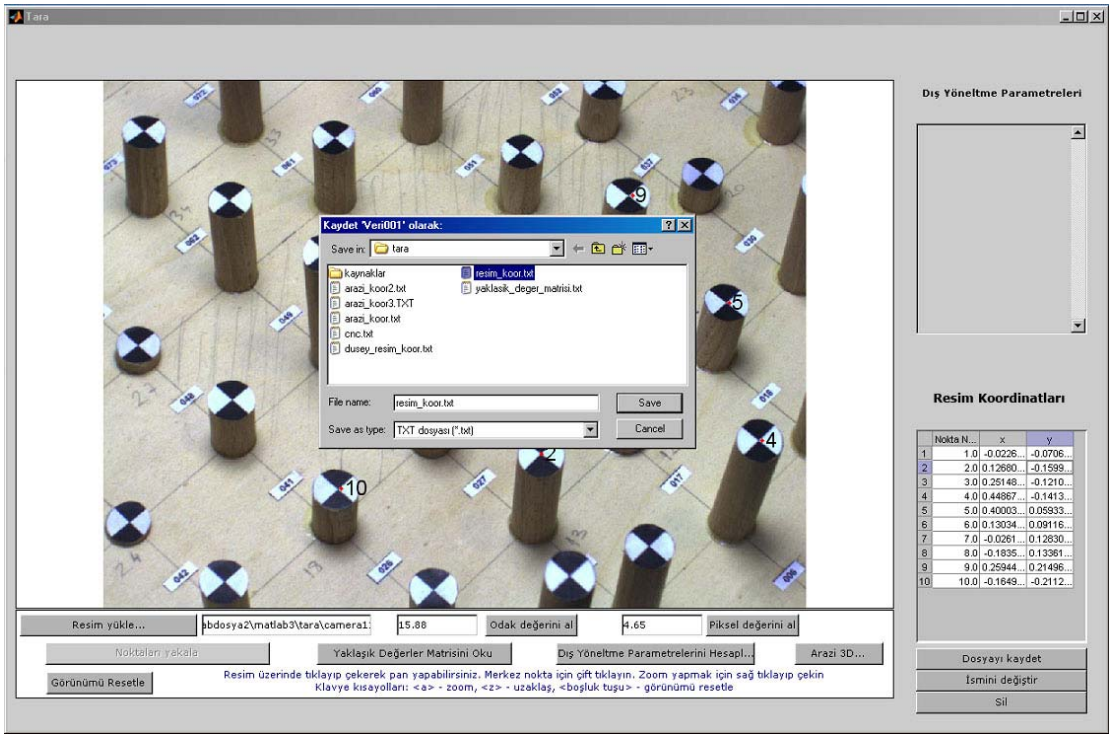


Şekil–3.10 Kalibrasyon ve Fot. Uyg. Arayüzünde seçilen noktalarının tablo halinde incelenmesi

Kameranın iç yöneltme parametrelerini belirleyen özelliklerinden olan odak değeri milimetre ve piksel değeri mikron cinsinden girilmiştir.



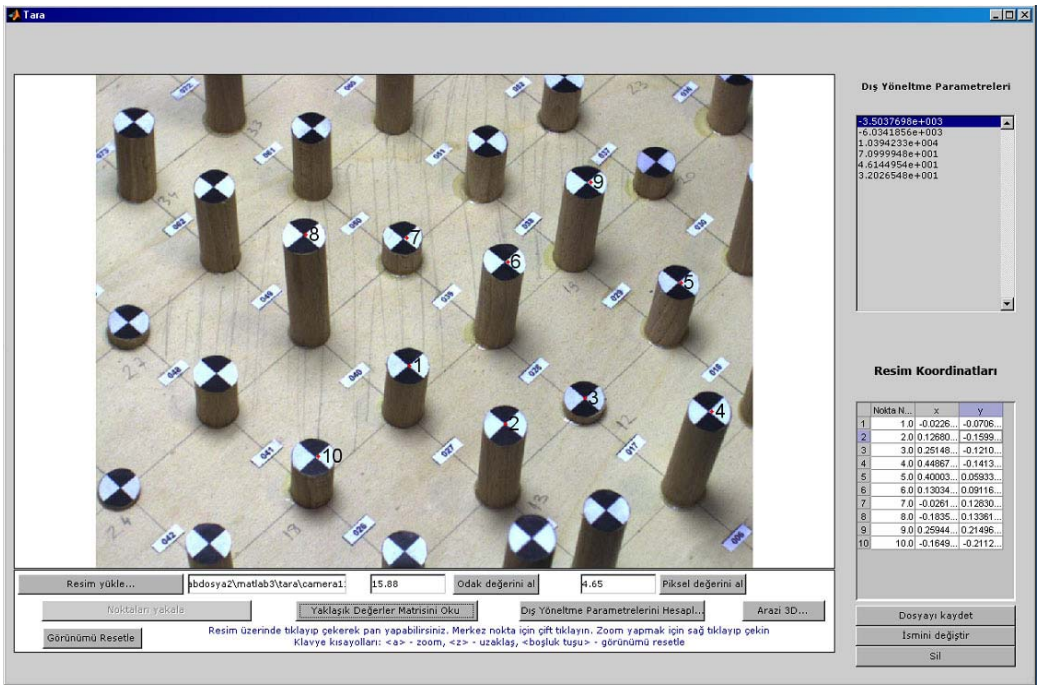
Şekil–3.11. Kalibrasyon ve Fot. Uyg. Arayüzünde odak ve piksel değerlerinin girilmesi



Şekil–3.12 Kalibrasyon ve Fot. Uyg. Arayüzünde tablodaki değerlerin kaydedilmesi

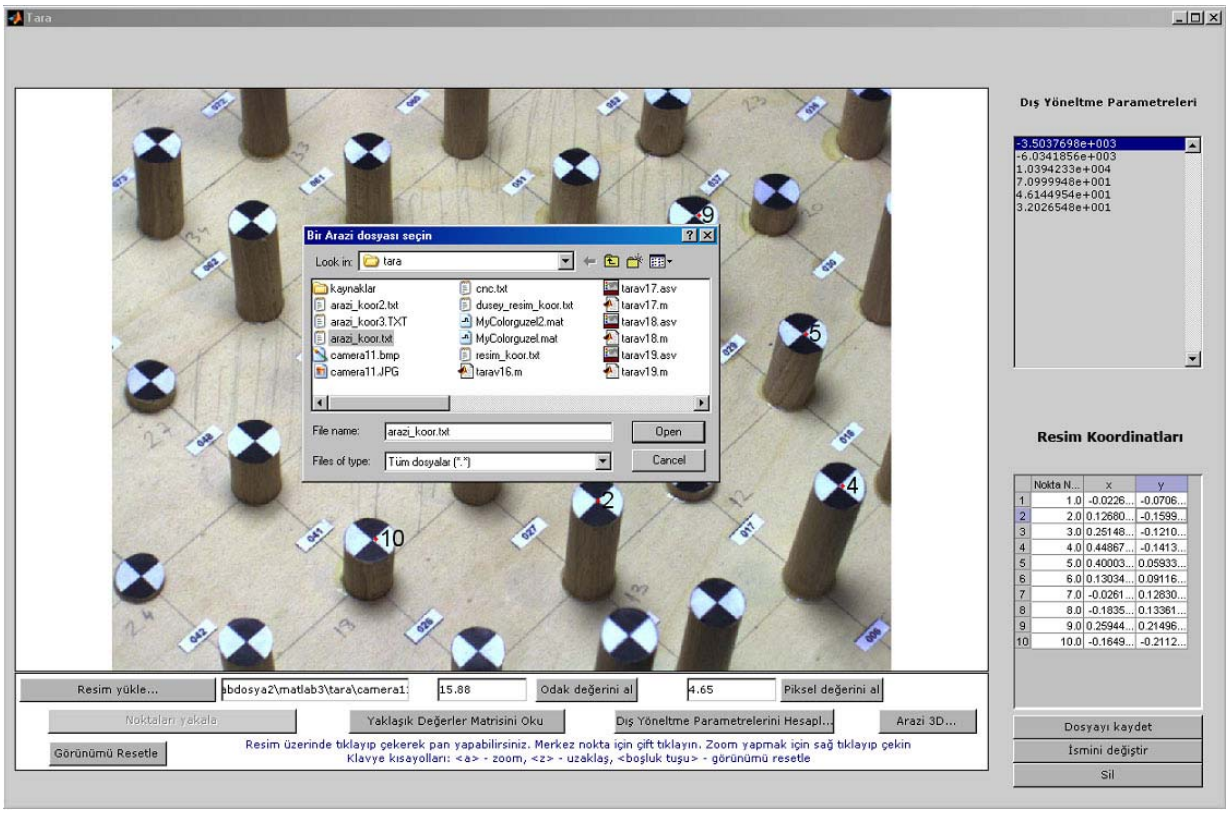
Önce “dış yöneltme parametrelerini hesapla” fonksiyonu, daha sonra da “yaklaşık değerler matrisini oku” fonksiyonu çalıştırılmıştır.

“Dış Yöneltme Parametrelerini Hesapla” fonksiyonu çalıştırıldığında “Space Resection” fonksiyonu çağrılmıştır. Böylece önceki adımda kaydedilen “x,y” değerlerini içeren nokta verileri alınarak dış yöneltme parametreleri dediğimiz X_0, Y_0, Z_0 ve “ω, φ, κ (omega, phi, kappa)” değerleri hesaplanmıştır. Bu değerler yine sırasıyla “Dış Yöneltme Parametreleri” olarak sol üst köşedeki listbox’da listelenmiştir.



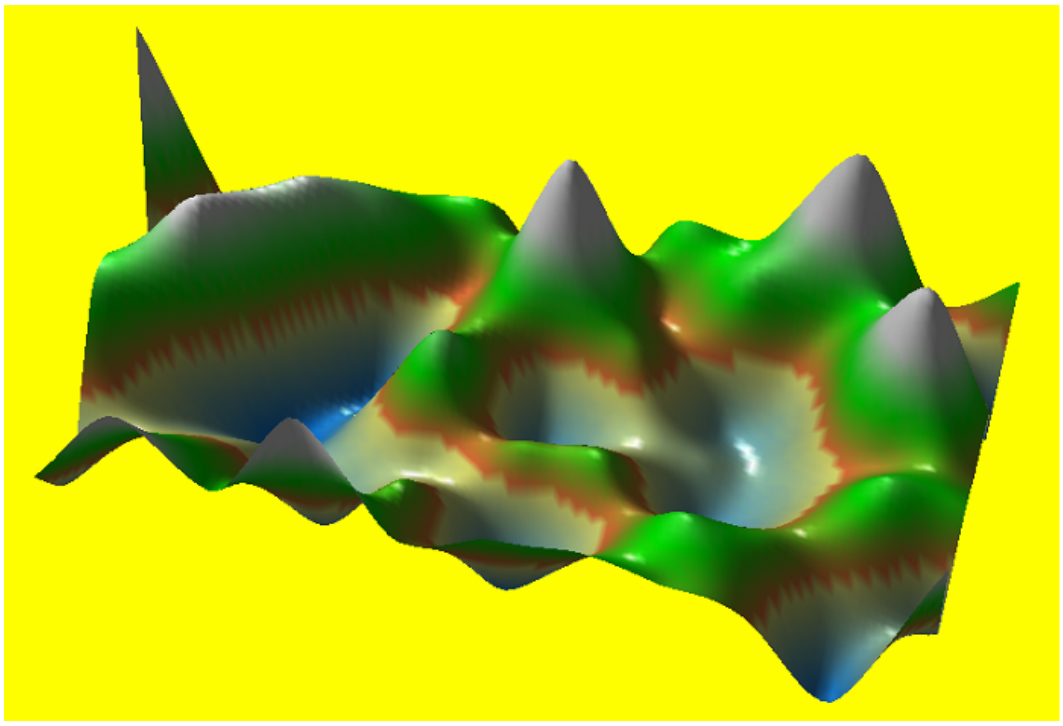
Şekil–3.13 Kalibrasyon ve Fot. Uyg. Arayüzünde dış yöneltme parametrelerinin hesaplanıp gösterilmesi

Kullanıcı bu şekilde değerlendirmek istediği resimleri işleyerek “x,y,z” değerlerini içeren üç boyutlu arazi yüzeyini elde ettikten sonra OpenGl ortamında render (üç boyutlu gösterim) edilmek üzere istenilen arazi yüzeyini içeren dosya seçilmiştir.



Şekil–3.14 Kalibrasyon ve Fot. Uyg. Arayüzünde OpenGl'de gösterilecek arazi yüzeyinin istenilen dosyadan alınması

Geliştirilen arayüzde yukarıda anlatılan fotogrametrik hesaplama adımlarından sonra ölçülen noktalara ait tüm cisim koordinatları kullanılarak OpenGl ortamında üç boyutlu arazi modelini gösteren bir arayüz geliştirilmiştir. OpenGl donanım hızlandırmalı olarak yani bilgisayar sistemi üzerindeki Graphics Card (Grafik Kartı) kullanılarak istenilen üç boyutlu veri kümesinin gerçek zamanlı (o an işlenerek) olarak ışıklandırma ile beraber gösterimini sağlayan bir grafik arabirimidir. Böylelikle OpenGl kullanılarak bu tarz bir projede elde edilebilecek yüksek yoğunluklu bir üç boyutlu verinin rahatça görselleştirilmesi mümkün olmaktadır.



Şekil–3.15 Kalibrasyon ve Fot. Uyg. Arayüzünde seçilen yüzeyin OpenGl'de üç boyutlu olarak gösterilmesi

4. SONUÇ ve ÖNERİLER

Bu tez çalışmasıyla Matlab ortamında hem vidyo kameralardan veri aktarımı hem de üç boyutlu olarak fotogrametrik dış yöneltme çalışmasını yapabilen bir arayüz geliştirilmiştir. Geliştirilen arayüzde bunlara ilave olarak üç boyutlu cisim koordinatlarının oluşturduğu sayısal yüzey modelini gösteren bir Opengl arayüzü de çalışmaya ilave edilmiştir.

Geliştirilen bu Matlab arayüz modülünün avantajları:

- Matlab ortamında oluşturulduğundan çok rahat incelenip geliştirilebilir. Programlama projelerinde kaynak kod C ve C++ gibi dillerde yazılır. Bu diller yüksek seviye diller ailesinden olmasına rağmen hazırlanan bir projenin modüler yani fonksiyonlara ayrılmış bir yapıya sahip olması ve her fonksiyonun dokumente edilmesi projenin geliştirilebilir olması açısında önem arz etmektedir. Geliştirilebilir projenin iki önemli özelliğinden birincisi modüler yapıda olmasıdır. Mesela iki sayıyı toplayan bir kod yazıldıysa ve bu kod proje içerisinde birçok defa kullanılacaksa bu kod yapısını fonksiyon haline getirmemiz gereklidir. Çünkü birçok satırdan oluşan bir yapının anlaşılması güçleşir. Fonksiyon şeklinde yazıldığında ise işlem açıkça anlaşılır ismiyle çağrıldığından hem kod karmaşası giderilir hem kod anlaşılır hale gelir. Projenin dokumente edilmesi ise her fonksiyonun işlevinin başlıklar halinde özetlenmesi ve fonksiyonlar içerisinde anlaşılır değişken isimlendirmesi ile beraber yapılan işlemlerin açıklama satırlarıyla izah edilmesidir. Bu da projenin asıl yazarından sonra geliştirmeye açık olmasını sağlayan diğer önemli faktördür. İşte Matlab bu saydığımız hususlarda özellikle matematiksel işlemler ve GUI noktasında bize sunduğu fonksiyonlarla hem bizi ek programlama maliyetinden kurtarır hem de kodun modüler halde anlaşılabilir ve geliştirilebilir olmasını sağlar.

- Görsel ve anlaşılır bir arayüze sahiptir. Arayüzdeki nesneler ve düğme gruplara sezgisel seviyededir. Yani ne yapılması gerektiğini arayüzü inceleyen biri hızlıca kavrayabilir. Kullanıcının yapabileceği muhtemel hatalar da gerek arayüz davranışlarıyla gerekse de uyarı mesajlarıyla kullanıcıya bildirilir.Yapılan

işlemlerin sonuçları anında görsel olarak kullanıcıya yansıtılır. Böylece kullanıcı yaptıklarının doğruluğunu test edebilir.

- OpenGl desteğini kullanarak grafik hızlandırıcısının nimetlerinden istifade ederek üç boyutlu görselleştirmeyi sağlar. Günümüzde 3B bilgisayar grafiklerindeki iki önemli standarttan birisi DirectX, diğeri de OpenGl'dir. OpenGl özellikle cross-platform (platform-bağımsız; yani Microsoft Windows, Linux gibi birçok işletim sisteminde çalışabilme) ve yaygın kolay kullanım özelliğinden dolayı tercih edilen bir standarttır. Matlab da grafik noktasında OpenGl desteği vermekle bize ciddi bir avantaj sağlamaktadır. Çünkü OpenGl gerektiğinde donanımsal hızlandırmayı sağlamaktadır. Yani elimizdeki bilgisayar mimarisinde bulunan üç boyutlu hızlandırmayı sağlayan grafik kartının gücünü Matlab'da hazırladığımız projede kullanabilmekteyiz. Böylece özellikle yüksek grafik gücü isteyen bu tarz uygulamalarda donanımsal hızlandırmayı kolaylıkla kullanabilmekte ve gerçek zamanlı görsellik elde edebilmekteyiz.

Dezavantajları ise:

- Matlab ortamının yapısından dolayı, C ve C++ gibi hız konusunda etkin bir platformda yazılmış profesyonel yazılımlara göre performans farkı görülebilir. Çünkü bu C ve C++ dilleri daha önce de belirttiğimiz gibi anlaşılma, kodlanma ve geliştirilebilir olma noktasında gerçekten Matlab'dan çok daha fazla emek ister ama aynı zamanda bu dezavantajlarıyla ters orantılı olarak hız olarak çok etkin yazılımlar üretirler. Çünkü yapılan her işlem ve yazılan her kod hız noktasında optimize edilmiştir ve optimize kod yazmak mümkündür. Bu da gayet tabi olarak hız olarak etkin yazılım olarak karşımıza çıkar.

- Büyük veri kümelerinde hız noktasında performansı incelenmemiştir. Yapılan projeler özellikle dışarıdan veri alımını ve işlenmesini sağlıyorsa işlenen veri miktarının projenin performansını etkileyebileceği noktası gözden kaçırılmamalıdır. Dolayısıyla belirli bir miktara kadar hızlı çalışan proje belirli bir noktadan sonra performans kayıpları yaşayabilir. Bu tarz durumlarda yazılımım optimum veri yoğunluğunu tespit etmek veya verileri gruplandırmak veyahut da projeyi performans noktasında güncellemek yapılması akılcı olan çözümlerdir.

Endüstriyel yakın resim ölçme sistemleriyle fotogrametri sıklıkla kullanılan bir yöntemdir. Bu çalışmayla Matlab ortamında geliştirilen arayüzler yardımıyla endüstriyel fotogrametride kullanılan ölçme sistemleri için donanımsal ve yazılımsal bir entegrasyon çalışması yapılmıştır. Geliştirilen Matlab arayüzleri yardımıyla endüstriyel vidyo kamera görüntüleri kendi orijinal arayüzleri olmaksızın Matlab ortamında görüntü alınması ve alınan bu görüntülerin fotogrametrik olarak yöneltilmesi sağlanmıştır. Böylece donanımsal bir entegrasyon Matlab kaynak kodlarıyla sağlanmıştır. Elde edilen bu görüntülerin fotogrametrik yöneltilmesinin uzaysal geriden kestirme (fotogrametri bağıntıları) Matlab arayüzünde bağımsız olarak yapılabilmesi ve üç boyutlu cisim koordinatlarına ait sayısal yüzey modelinin bilgisayar ortamında görselleştirilmesi çalışması yapılarak yazılımsal entegrasyonu sağlanmaktadır.

KAYNAKLAR

Kraus K., Photogrammetry, Dümmler Bonn 1994.

Uzunoğlu M., Her Yönü İle Matlab, Türkmen Kitabevi, 2003.

Ergün B., Yakın Resim Fotogrametrisinde Mikrofotogrametrik Uzman Sistem Tasarımı, Tesisi ve Kalibrasyonu, İTÜ Fen Bilimleri Enstitüsü Doktora Tezi, İstanbul, 2003

İnan A., Matlab Kılavuzu, Papatya Yayıncılık, 2007

Fraser, C.S, 1996 Industrial Measurement Applications, Close Range Photogrammetry And Machine Vision, Scotland: Whittles Publishing

Atkinson, K.B., 1996. Close Randge Photogrammetry And Machine Vision, Whittles Publishing, Scotland.

Fritch, D., Ebner.H, 1991. Digital Photogrammetric Systems, Wichmann, Karlsruhe

Mason, S.O., 1994. Expert System-Based Design of Photogrammetric Networks. Institute of Geodesy and Photogrammetry

Ergün B., Şahin C., Mikrofotogrametrik Endüstriyel Uzman Sistem Tesisi ve Kalibrasyonu, Gebze Yüksek Teknoloji Enstitüsü Bilimsel Araştırma Projesi Bitirme Raporu, Gebze, 2009

ÖZGEÇMİŞ

Kişisel Bilgiler:

Doğum Tarihi: 29.03.1981

Doğum Yeri: Havsa/EDİRNE

Öğrenim Durumu:

2007 - Y. Lisans: GYTE Mühendislik ve Fen Bilimleri Enstitüsü
Jeodezi ve Fotogrametri Müh.

1999 – 2004 Lisans: Fatih Üniversitesi Müh. Fak.
Bilgisayar Müh.

1996–1999 Lise: Özel Fatih Fen Lisesi

Ek-1: Fotogrametrik Arayüz Uygulaması Kodları

```
function tara(~)
basla

    function basla

% workspace guzelce bir temizleyelim
evalin('base', sprintf('clear'));
clc;

% o anki pencereleri kapat
im  =  findobj('type',  'figure',  'tag',  'TaraGUI');
if ishandle(im)
  close(im);
end

% arka plan renkleri
arkaplan1 = [.8, .8, .8];
arkaplan2 = [ 1,  1,  1];
arkaplan3 = [.7, .7, .7];
arkaplan4 = [ 1, .5, .5];

% piksel cinsinden ekran olculerini al
% o anki ölçü cinsini al ve un içinde sakla
un          = get(0, 'units');
% ölçü cinsini piksel yap çünkü pixel cinsinden ekran ölçülerini
alcaz
set(0, 'units', 'pixels');
screenSize  = get(0, 'ScreenSize');
sW          = screenSize(3); % screenSize(3) screen witdth veriyor
sH          = screenSize(4); % bu da screen height
%daha önceki ölçü birimini yeniden ayarla
set(0, 'units', un);

% figure genislik ve yuksekligi , piksel cinsinden
% fw=figure width
% fh=figure height
fW          = sW-200;
fH          = sH-100;

% çözünürlüğe göre konumlama yapmak için
%burası experimental
konumlama=sH/2+sH/4.45;


%-------------------------------------------------------------------
-------
% Ozel kursor pointerları
%-------------------------------------------------------------------
-------

% zoom yaparken görülen pointer
% buradaki değerler 3 rengi temsil ediyor
zoomPointer = [
   NaN   NaN   NaN     2     2   NaN   NaN   NaN   NaN   NaN   NaN
1     1     1   NaN   NaN
```

```
   NaN   NaN     2     1     1     2   NaN   NaN   NaN   NaN     1
2     2     2     1   NaN
   NaN     2     1     1     1     1     2   NaN   NaN     1     2
2     1     2     2     1
     2     1     1     1     1     1     1     2   NaN     1     2
1     1     1     2     1
     2     1     2     1     1     2     1     2   NaN     1     2
2     1     2     2     1
   NaN     2     2     1     1     2     2   NaN   NaN     1     1
2     2     2     1   NaN
   NaN   NaN     2     1     1     2   NaN   NaN     1     1     1
1     1     1   NaN   NaN
   NaN   NaN   NaN     2     2   NaN   NaN   NaN     1     1   NaN
NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN     2     2   NaN   NaN   NaN   NaN   NaN   NaN
1     1     1   NaN   NaN
   NaN   NaN     2     1     1     2   NaN   NaN   NaN   NaN     1
2     2     2     1   NaN
   NaN     2     2     1     1     2     2   NaN   NaN     1     2
2     2     2     2     1
     2     1     2     1     1     2     1     2   NaN     1     2
1     1     1     2     1
     2     1     1     1     1     1     1     2   NaN     1     2
2     2     2     2     1
   NaN     2     1     1     1     1     2   NaN   NaN     1     1
2     2     2     1   NaN
   NaN   NaN     2     1     1     2   NaN   NaN     1     1     1
1     1     1   NaN   NaN
   NaN   NaN   NaN     2     2   NaN   NaN   NaN     1     1   NaN
NaN   NaN   NaN   NaN   NaN
 ];

% pan yaparken görülen el pointer
HandPointer = [
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN     2     2   NaN     2     2   NaN     2
2   NaN   NaN   NaN   NaN
   NaN   NaN   NaN     2     1     1     2     1     1     2     1
1     2     2   NaN   NaN
   NaN   NaN     2     1     2     2     1     2     2     1     2
2     1     1     2   NaN
   NaN   NaN     2     1     2     2     2     2     2     2     2
2     1     2     1     2
   NaN   NaN   NaN     2     1     2     2     2     2     2     2
2     2     2     1     2
   NaN   NaN     2     1     1     2     2     2     2     2     2
2     2     2     1     2
   NaN     2     1     2     2     2     2     2     2     2     2
2     2     2     1     2
   NaN     2     1     2     2     2     2     2     2     2     2
2     2     2     1     2
   NaN     2     1     2     2     2     2     2     2     2     2
2     2     1     2   NaN
   NaN   NaN     2     1     2     2     2     2     2     2     2
2     2     1     2   NaN
```

```
   NaN   NaN   NaN     2     1     2     2     2     2     2     2
2     1     2   NaN   NaN
   NaN   NaN   NaN   NaN     2     1     2     2     2     2     2
2     1     2   NaN   NaN
   NaN   NaN   NaN   NaN     2     1     2     2     2     2     2
2     1     2   NaN   NaN
 ];


%figur yani temel arayüzün özellikleri
im = figure(...
  'units'                           , 'pixels', ...
  'position'                        , [100, 50, fW, fH], ...
  'backingstore'                    , 'off', ...
  'doublebuffer'                    , 'on', ...
  'name'                            , 'Fotogrametrik Uygulama', ...
  'numbertitle'                     , 'off', ...
  'menubar'                         , 'none', ...
  'color'                           , arkaplan1, ...
  'pointer'                         , 'arrow', ...
  'visible'                         , 'on', ...
  'interruptible'                   , 'off', ...
  'busyaction'                      , 'cancel', ...
  'resizefcn'                       , @figResizeFcn, ...
  'windowbuttonupfcn'               , @winBtnUpFcn, ...
  'keypressfcn'                     , @keyPressFcn, ...
  'deletefcn'                       , 'delete(timerfind(''name'',
''BtnUpTimer''));', ...
  'tag'                             , 'TaraGUI', ...
  'defaultUicontrolUnits'           , 'pixels', ...
  'defaultUicontrolBackgroundColor' , arkaplan3, ...
  'defaultUicontrolFontname'        , 'Verdana', ...
  'defaultUicontrolFontUnits'       , 'pixels', ...
  'defaultUicontrolFontsize'        , 10, ...
  'defaultUicontrolInterruptible'   , 'off', ...
  'defaultUicontrolBusyAction'      , 'cancel', ...
  'defaultAxesFontName'             , 'Verdana', ...
  'defaultAxesUnits'                , 'pixels', ...
  'defaultAxesFontSize'             , 8);


%arayüzdeki frame,text,button,table,listbox,editbox bileşenleri
burada
%oluşturuluyor

%panel yapısı nesneleri daha doğru düzgün konumlandırmamızı saglar
panelW = 0.8*fW;
uicontrol(...
  'style'                           , 'frame', ...
  'position'                        , [10, fH-110-konumlama, panelW,
100], ...
  'backgroundcolor'                 , arkaplan2);
uicontrol(...
  'style'                           , 'pushbutton', ...
  'string'                          , 'Resim yükle...', ...
  'callback'                        , {@loadImageFcn, []}, ...
  'position'                        , [15, fH-40-konumlama, 200,
25], ...
  'tag'                             , 'LoadImageBtn');
uicontrol(...
```

```
  'style'                         , 'edit', ...
  'backgroundcolor'               , arkaplan2, ...
  'string'                        , 'incelenen dosya', ...
  'position'                      , [220, fH-40-konumlama, panelW-
800, 25], ...
  'horizontalalignment'           , 'left', ...
  'enable'                        , 'on', ...
  'tag'                           , 'ImageFileLoc');

uicontrol(...
  'style'                         , 'edit', ...
  'backgroundcolor'               , arkaplan2, ...
  'string'                        , 'mikron', ...
  'position'                      , [695, fH-40-konumlama, panelW-
900, 25], ...
  'horizontalalignment'           , 'left', ...
  'enable'                        , 'on', ...
  'tag'                           , 'Piksel');

uicontrol(...
  'style'                         , 'edit', ...
  'backgroundcolor'               , arkaplan2, ...
  'string'                        , 'milimetre', ...
  'position'                      , [440, fH-40-konumlama, panelW-
900, 25], ...
  'horizontalalignment'           , 'left', ...
  'enable'                        , 'on', ...
  'tag'                           , 'Odak');

uicontrol(...
  'style'                         , 'pushbutton', ...
  'string'                        , 'Odak değerini al', ...
  'callback'                      , @OdakImageFcn, ...
  'position'                      , [890-350, fH-40-konumlama,
105, 25], ...
  'tag'                           , 'OdakBtn');


uicontrol(...
  'style'                         , 'pushbutton', ...
  'string'                        , 'Piksel değerini al', ...
  'callback'                      , @PikselImageFcn, ...
  'position'                      , [890-100, fH-40-konumlama,
105, 25], ...
  'tag'                           , 'PikselBtn');

uicontrol(...
  'style'                         , 'togglebutton', ...
  'string'                        , 'Noktaları seçeyim', ...
  'buttondownfcn'                 , @grabPointsFcn, ...
  'position'                      , [3*(panelW)/4-700, fH-72-
konumlama, panelW/4+5, 25], ...
  'enable'                        , 'on', ...
  'tag'                           , 'GrabPointsBtn');

uicontrol(...
  'style'                         , 'pushbutton', ...
  'string'                        , 'Yaklaşık Değerler Matrisini
Oku ', ...
  'callback'                      , @YaklasikDegerleriOku, ...
```

```
  'position'                          , [350, fH-72-konumlama, 220,
25], ...
  'tag'                               , 'YaklasikDegerleriOkuBtn');

uicontrol(...
  'style'                             , 'pushbutton', ...
  'string'                            , 'Dış Yöneltme Parametrelerini
Hesapla... ', ...
  'callback'                          , @Spaceresection, ...
  'position'                          , [620, fH-72-konumlama, 225,
25], ...
  'tag'                               , 'SpaceRecessionBtn');

uicontrol(...
  'style'                             , 'pushbutton', ...
  'string'                            , 'Arazi 3D... ', ...
  'callback'                          , @Arazi, ...
  'position'                          , [890, fH-72-konumlama, 100,
25], ...
  'tag'                               , 'AraziBtn');


uicontrol(...
  'style'                             , 'pushbutton', ...
  'string'                            , 'Resmi Ekrana Ortala', ...
  'callback'                          , @resetViewFcn, ...
  'position'                          , [45, fH-105-konumlama, 120,
25], ...
  'tag'                               , 'ResetViewBtn');

uicontrol(...
  'style'                             , 'text', ...
  'position'                          , [170, fH-108-konumlama,
panelW-165, 31], ...
  'horizontalalignment'               , 'center', ...
  'foregroundcolor'                   , [0 0 .5], ...
  'backgroundcolor'                   , arkaplan2, ...
  'fontsize'                          , 10, ...
  'string'                            , {'Resim üzerinde tıklayıp
çekerek pan yapabilirsiniz. Merkez nokta için çift tıklayın. Zoom
yapmak için sağ tıklayıp çekin', ...
    'Klavye kısayolları: <a> - zoom, <z> - uzaklaş, <boşluk tuşu> -
görünümü resetle'});

%rPanel yani right panel sağdaki pencereleri içerir
%üstte bir listbox var , burada dış yöneltme parametreleri
listelenicek
%rPanelX panelin X koordinatının nerden başlıycağını belirler
%rPanelW ise genişlik değeridir
rPanelX = 0.82 * fW;
rPanelW = fW - rPanelX - 10;

uicontrol(...
  'style'                             , 'listbox', ...
  'position'                          , [rPanelX+10, 450, rPanelW-20,
0.3*fH], ...
  'backgroundcolor'                   , arkaplan1, ...
  'tooltipstring'                     , sprintf('x0 y0 z0 omega fi
kappa'), ...
  'tag'                               , 'DisYoneltmeList');
```

```
uicontrol(...
  'style'                          , 'text', ...
  'string'                         , 'Dış Yöneltme Parametreleri',
...
  'position'                       , [rPanelX+10, 0.6*fH+230,
rPanelW-20, 20], ...
  'backgroundcolor'                , arkaplan1, ...
  'fontweight'                     , 'bold');


uicontrol(...
  'style'                          , 'text', ...
  'string'                         , 'Resim Koordinatları', ...
  'position'                       , [rPanelX+10, 0.6*fH-125,
rPanelW-20, 20], ...
  'backgroundcolor'                , arkaplan1, ...
  'fontweight'                     , 'bold');

uicontrol(...
  'style'                          , 'pushbutton', ...
  'string'                         , 'Dosyayı kaydet', ...
  'position'                       , [rPanelX+10, 70, rPanelW-20,
25], ...
  'callback'                       , @variableManipulationFcn, ...
  'tooltipstring'                  , sprintf('Değişkeni  MAT
dosyası veya\nçift duyarlıklı, sekme ayrımlı TXT dosyası olarak
kaydet'), ...
  'tag'                            , 'SaveAs');

uicontrol(...
  'style'                          , 'pushbutton', ...
  'string'                         , 'Sil', ...
  'position'                       , [rPanelX+10, 20+25, rPanelW-
20, 25], ...
  'callback'                       , @variableManipulationFcn, ...
  'tooltipstring'                  , 'Değişkeni çalışma alanından
siler', ...
  'tag'                            , 'Delete');

%resmin görüldüğü ve üzerinde nokta işlemlerinin yapıldığı eksen
axes(...
  'position'                       , [10, 140, 0.8*fW, fH-200], ...
  'visible'                        , 'on', ...
  'handlevisibility'               , 'on', ...
  'box'                            , 'on', ...
  'drawmode'                       , 'fast', ...
  'xtick'                          , [], ...
  'ytick'                          , [], ...
  'interruptible'                  , 'off', ...
  'buttondownfcn'                  , @winBtnDownFcn, ...
  'tag'                            , 'ImageAxis');


imAxRatio = (fH - 200) / (0.8 * fW);

%Resim Koordinatları tablosuna bağlayacağımız sağ tıklama menüsü
%Bu menü ile istediğimiz noktayı silebiliriz
```

```
%Bunun için bir normal inceleme modu, bir de silme modu oluşturduk
%default değer inceleme modudur
hcmenu = uicontextmenu;
uimenu(hcmenu, 'Label', 'Noktayı Silme modunu aktifleştir',
'Callback', @Tablodannoktasil);
uimenu(hcmenu, 'Label', 'Noktayı Silmeyi modunu kapat', 'Callback',
@Tablodannoktasilkapat);

% normalize birim sistemini kullan
% böylece figure resize edildiğinde figure elemanları uygun şekilde
% pozisyonlanabilsin
set(findobj(im, 'type', 'uicontrol'), 'units', 'normalized');
set(findobj(im, 'type', 'axes'), 'units', 'normalized');
set(findobj(im, 'type', 'uitable'), 'units', 'normalized');

% handles yapısı
% bu yapı fonksiyon ve gui elemanları arasında değerlerin
paylaşımını sağlar
handles                     = guihandles(im);

% Resim Koordinatları tablosu
% Her nokta için x ve y olarak iki sütundan oluşur
handles.xytable=
uitable('Parent',im,'ColumnName',{'x','y'},'Position',[rPanelX+10,
100, rPanelW-20, 0.3*fH],'ColumnWidth',{53},...
'Units','normalized');
set(handles.xytable,'CellSelectionCallback',@TabloDegerlendir)
set(handles.xytable,'uicontextmenu',hcmenu);

%el ve zoom pointerlarının ayarları
handles.zoomPointer         = zoomPointer;
handles.zoomPointerHotSpot  = [9 9];
handles.HandPointer         = HandPointer;
handles.HandPointerHotSpot  = [9 9];

% rPanelX,rPanelW ve fH değerlerini handles yapısına kaydediyoruz
handles.rPanelX=rPanelX;
handles.rPanelW=rPanelW;
handles.fH=fH;

handles.curPointer                = 'arrow';
handles.curPointerData.CData      = zoomPointer;
handles.curPointerData.HotSpot    = [1 1];
handles.state                     = 'normal';
handles.arkaplan1                  = arkaplan1;
handles.arkaplan2                  = arkaplan2;
handles.arkaplan3                  = arkaplan3;
handles.arkaplan4                  = arkaplan4;
handles.imAxRatio                 = imAxRatio;
handles.I                         = []; %resim datasını içerecek
handles.map                       = []; %resimle alakalı
handles.isPanning                 = false;
handles.curTitle                  = '';
handles.CurrentPointAxes          = [];
handles.CurrentPointFig           = [];
handles.PikselDeger=1;
handles.OdakDeger=1;
handles.hcmenu=hcmenu;
% mouse ile sol tıklandığında tıklamadan sonra mouse bırakılmadan
hareket
```

```
% ettirilirse pan yapılacak , bırakılırsa ve nokta seçimi modundaysa
% istenilen nokta seçilmiş olacak.
% Bununla alakalı zamanlama işlemleri için kullanılan nesne
handles.timer                       = timer(...
                                            'Name', 'BtnUpTimer', ...
                                           'StartDelay', 0.2, ...
                                            'TimerFcn',
{@btnUpTimerFcn, im});
handles.Noktasil=0;
% piksel koordinat sisteminde seçtiğimiz noktaların koordinatları
handles.ImDat    = [];
% resim koordinat sisteminde seçtiğimiz noktaların koordinatları
handles.TrueDat = [];
% guidata sıklıkla kullanılan bir komuttur.
% handles yapısını istenilen guiye entegre eder
guidata(im, handles);

% figure oluştururkem im şeklinde bir isim verdik
% "figure"ü görünür yap
set(im, 'handlevisibility', 'on');
set(im, 'visible', 'on');


% yaklaşık değerleri okuma fonksiyonu
%-------------------------------------------------------------------------
-------
function YaklasikDegerleriOku(varargin)
% Bu fonksiyon space resection ile elde edilen
% yaklaşık değerleri okur ve görüntüler.

obj = varargin{1};
handles = guidata(obj);
% dosyayı aç
fid = fopen('yaklasik_deger_matrisi.txt');
% textscan ile %s yani string şeklinde değerleri okuyoruz
handles.YaklasikDegerler = textscan(fid, '%s');
deger=handles.YaklasikDegerler;
% cellarray şeklinde okuduğundan 1 indeksindeki değeri alıyoruz
degerstr=deger{1};
%listbox değerini güncelliyoruz
set(handles.DisYoneltmeList,'string',degerstr)

fclose(fid);
guidata(obj, handles);



% degisken manipulasyon fonksiyonu
function variableManipulationFcn(varargin)
% Bu fonksiyon degisken listbox altındaki 2 butondan
% biri basıldıgında calısır.

obj = varargin{1};

handles = guidata(obj);

%hangi butona basıldığına bakalım
switch get(obj, 'Tag')
```

```
  case 'SaveAs' % degiskeni bir .mat dosyası olarak kaydet
    [fname, pname] = uiputfile(...
      {...'*.mat', 'MAT dosyası (*.mat)'; ...
      '*.txt', 'TXT dosyası (*.txt)'}, ...
      sprintf('Farklı kaydet:'), ...
      'resim_koor');
    if ~isequal(fname, 0) && ~isequal(pname, 0)
      %değişkenin değerini TrueDat yani resim koordinatlarını tutan
      %matrisden alalım.
      TrueDatWithIndex(size(handles.TrueDat),3)=0;
      for i=1:size(handles.TrueDat)
      TrueDatWithIndex(i,1)=i;
      TrueDatWithIndex(i,2)=handles.TrueDat(i,1);
      TrueDatWithIndex(i,3)=handles.TrueDat(i,2);
      end
      saveDatFcn(pname, fname, TrueDatWithIndex)
    end


  case 'Delete' % değişkeni workspaceden siler
    btn = questdlg(sprintf('Bu veri  çalışma alanından kaldırılsın
mı?'), ...
      'Sil', 'Evet', 'Hayır', 'Hayır');
    switch btn
      case 'Evet'
        set(handles.xytable,'Data','');
        handles.TrueDat=[];
        handles.ImDat=[];

    % Bir Callback tarafından çağrılmadığında resim göstermek için :
    set(0, 'ShowHiddenHandles', 'on');

    %image eksenini temizle
    cla(handles.ImageAxis);

    %resmi yeniden aç
    iH = image(handles.I);

    % resim ve resim ekseniyle alakalı ayarlar
    set(iH, 'HitTest', 'off', 'EraseMode', 'normal');
    set(handles.ImageAxis, 'xtick', [], 'ytick', []);
    axis equal;

    set(handles.TaraGUI, 'nextplot', 'add');
    handles.ImageLine = line(NaN, NaN, ...
  'color'           , 'red', ...
  'linestyle'       , 'none', ...
  'marker'          , '.', ...
  'tag'             , 'ImageLine', ...
  'hittest'         , 'off');


    set(handles.ImageAxis, ...
      'drawmode'        , 'fast', ...
      'tag'             , 'ImageAxis', ...
      'handlevisibility', 'on', ...
      'NextPlot'        , 'add' , ...
      'buttondownfcn'   , @winBtnDownFcn);
```

```
     handles.ImLimits    = [get(handles.ImageAxis, 'xlim'); ...
                            get(handles.ImageAxis, 'ylim')];

        set(handles.GrabPointsBtn, ...
          'enable', 'off', ...
          'value' , 0);

        zoom off;

        guidata(obj, handles);

       case 'Hayır'
    end

end

guidata(obj, handles);

% saveDatFcn
%-------------------------------------------------------------------------
-------
function saveDatFcn(pname, fname, var) %#ok<INUSD>
% bu fonksiyon değişkeni dosyaya kaydeder

[p, fname, ext] = fileparts(fname); %#ok<ASGLU>
switch lower(ext)
  case '.mat'
    eval(sprintf('%s = var;', fname));
    save(fullfile(pname, [fname, ext]), fname, '-v6');
  case '.txt'
    eval(sprintf('%s = var;', fname));
    save(fullfile(pname, [fname, ext]), fname, '-ascii', '-double',
'-tabs');
  otherwise
    errordlg('bilinmeyen uzantı.');
end

%-------------------------------------------------------------------------
-------
% Tablodan nokta sil modunu aç
%-------------------------------------------------------------------------
-------
        function Tablodannoktasil(varargin)

obj = varargin{1};
handles = guidata(obj);
handles.Noktasil=1;
guidata(handles.TaraGUI, handles);


% Tablodan nokta sil modunu kapat
%-------------------------------------------------------------------------
-------
        function Tablodannoktasilkapat(varargin)


obj = varargin{1};
handles = guidata(obj);
handles.Noktasil=0;
```

```
guidata(handles.TaraGUI, handles);


% TabloDegerlendir
%-------------------------------------------------------------------
-------
function TabloDegerlendir(hObject, eventdata, varargin)
% Bu fonksiyonun sebebi hikmeti uitable de yapılan değişikliklere
göre x y
% matrisini değiştirmektir.
% Noktasil=1 ise siler değilse devam eder.

guiobj=hObject;
guihandles=guidata(guiobj);

%Evvelen hazır bulunan Noktasil değişkenin değerine binaen nokta
tablosunda
%tıklanan noktanın silinip silinmeyeceği kontrol edilir
if not(guihandles.Noktasil) % 1 değilse yani 0 ise return yap
    return
else

  obj = eventdata.Indices;
  handles = obj(:,1);
  handles=unique(handles);
  delete(findobj(gca, 'Tag', 'Text'));
  guihandles.TrueDat(handles,:)=[];
  guihandles.ImDat(handles,:)=[];

  set(guihandles.ImageLine,...
  'color'           , 'red', ...
  'linestyle'       , 'none', ...
  'marker'          , '.', ...
  'tag'             , 'ImageLine', ...
  'hittest'         , 'off');

for i=1:size(guihandles.ImDat)

text(guihandles.ImDat(i,1)+5,guihandles.ImDat(i,2),num2str(i),'FontS
ize',18,'Tag','Text');
end

set(guihandles.ImageLine, ...
            'xdata', guihandles.ImDat(:, 1), ...
            'ydata', guihandles.ImDat(:, 2));

c=num2cell(guihandles.TrueDat);
set(guihandles.xytable,'Data',c);

guidata(guiobj, guihandles);
end


% loadImageFcn
%-------------------------------------------------------------------
-------
function loadImageFcn(varargin)
% bu fonksiyon resim dosyasını açar
```

```
[obj, filename] = splitvar(varargin([1, 3]));

handles = guidata(obj);

%resim dosyasının açılması için bir dialog box açıyoruz
%istediğimiz türde resim dosyalarını burada belirtebiliriz
if isempty(filename)
  [fname, pathname] = uigetfile(...
    {'*.bmp;*.jpg;*.jpeg;*.tif;*.tiff;*.gif;*.png', ...
    'Resim dosyaları (*.bmp,  *.jpg,  *.jpeg,  *.tif,  *.tiff,
*.gif,  *.png)';
    '*.bpm', 'Bit eşlem dosyaları (*.bpm)';
    '*.jpg;*jpeg', 'JPG dosyaları (*.jpg,  *.jpeg)';
    '*.tif;*tiff', 'TIFF dosyaları (*.tif,  *.tiff)';
    '*.gif', 'GIF dosyaları (*.gif)';
    '*.png', 'PNG dosyaları (*.png)';
    '*.*', 'Tüm dosyalar (*.*)'}, 'Bir resim dosyası seçin');
  if ischar(fname)
    filename = fullfile(pathname, fname);
  else
    return;
  end
end

set(handles.ImageFileLoc, 'string', filename);

try

  [A, map] = imread(filename);

catch
  errordlg(lasterr);
  return;
end

if ndims(A) == 3   %Bazı TIFF dosyaları yanlış boyutta matrislidir.
  if size(A, 3)>3
    A = A(:, :, 1:3);
  elseif size(A, 3)<3
    errordlg('Bu acaip bir resim dosyası.muhtemelen kötü TIFF
dosyası...');
    return;
  end
end

% Bir Callback tarafından çağrılmadığında resim göstermek için :
set(0, 'ShowHiddenHandles', 'on');

cla(handles.ImageAxis);

if isempty(map)
  imageInfo = imfinfo(filename);
  if strcmpi(imageInfo.ColorType, 'grayscale')
    colormap(gray(2^imageInfo.BitDepth));
  end
else
  colormap(map);
end
iH = image(A);
```

```
set(iH, 'HitTest', 'off', 'EraseMode', 'normal');
set(handles.ImageAxis, 'xtick', [], 'ytick', []);
axis equal;

set(handles.TaraGUI, 'nextplot', 'add');
set(handles.ImageAxis, ...
  'drawmode'        , 'fast', ...
  'tag'             , 'ImageAxis', ...
  'handlevisibility', 'on', ... // callback
  'NextPlot'        , 'add' , ...
  'buttondownfcn'   , @winBtnDownFcn);
set(get(handles.ImageAxis, 'title'), ...
  'string'          , '', ...
  'fontunits'       , 'pixels', ...
  'fontsize'        , 24, ...
  'color'           , 'red');
handles.ImageLine = line(NaN, NaN, ...
  'color'           , 'red', ...
  'linestyle'       , 'none', ...
  'marker'          , '.', ...
  'tag'             , 'ImageLine', ...
  'hittest'         , 'off');


% resim datasının başlatılması
handles.I           = A;
handles.map         = map;

handles.ImLimits    = [get(handles.ImageAxis, 'xlim'); ...
                       get(handles.ImageAxis, 'ylim')];


set(handles.GrabPointsBtn, ...
  'enable', 'off', ...
  'value' , 0);

zoom off;

handles.ResimYatay=imageInfo.Width;
handles.ResimDikey=imageInfo.Height;

guidata(obj, handles);

% pointerFcn
%-------------------------------------------------------------------
-------
function pointerFcn(varargin)
% Kursorun resim ekseni içinde olup olmamasına gore pointer
degistir.

[handles, ptr] = splitvar(varargin(3:4));

pt = get(handles.ImageAxis, 'CurrentPoint');
xl = get(handles.ImageAxis, 'xlim');
yl = get(handles.ImageAxis, 'ylim');
if pt(1,1) > xl(1) && pt(1,1) < xl(2) && pt(1,2) > yl(1) && pt(1,2)
< yl(2)
  set(handles.TaraGUI, 'pointer', ptr);
else
```

```
  set(handles.TaraGUI, 'pointer', 'arrow');
end




% figResize
%--------------------------------------------------------------------------
-------
%--------------------------------------------------------------------------
-------
function figResizeFcn(varargin)
% Bu fonksiyon eksenin eksen boyutunu doldurdugundan emin olmamızı
saglar

obj = varargin{1};

handles = guidata(obj);

axis(handles.ImageAxis, 'equal');
handles.imAxRatio = diff(get(handles.ImageAxis, 'ylim')) / ...
  diff(get(handles.ImageAxis, 'xlim'));
handles.ImLimits = [get(handles.ImageAxis, 'xlim'); ...
    get(handles.ImageAxis, 'ylim')];

guidata(obj, handles);

% keyPressFcn
%--------------------------------------------------------------------------
-------
%--------------------------------------------------------------------------
-------
function keyPressFcn(varargin)
% Klavye kısayolları. 'yakala' modunda, <backspace>
% son tıklanan noktayı siler ve <enter>  'yakala' modunu kapatır.

obj = varargin{1};

handles = guidata(obj);

if ~isempty(handles.I)
  k = lower(get(obj, 'CurrentKey'));

  switch k
    case 'a'          % zoom in
      xl = get(handles.ImageAxis, 'xlim'); xrng = diff(xl);
      yl = get(handles.ImageAxis, 'ylim'); yrng = diff(yl);

      % cok fazla zoom yapılmasın.
      % 64x zoom.
      if xrng >= size(handles.I, 2)/64*2
        % zoomu farkedilir sekilde goster
        for id = 0:0.2:1
          set(handles.ImageAxis, ...
            'xlim', xl + id * xrng / 4 * [1, -1], ...
            'ylim', yl + id * yrng / 4 * [1, -1]);
          drawnow;
        end
      end
```

```
    case 'z'        % uzaklas
      xl = get(handles.ImageAxis, 'xlim'); xrng = diff(xl);
      yl = get(handles.ImageAxis, 'ylim'); yrng = diff(yl);
      % uzaklasmayı farkedilir sekilde goster
      for id = 0:0.2:1
        set(handles.ImageAxis, ...
          'xlim', xl + id * xrng / 2 * [-1, 1], ...
          'ylim', yl + id * yrng / 2 * [-1, 1]);
        drawnow;
      end

    case 'space'    % gorunumu resetle
      resetViewFcn(handles.ResetViewBtn);

    case {'backspace', 'delete'}
      %son eklenen noktanın silinmesi

      %eğer state yani durum değişkeni Grab ise yani nokta yakalama
modu
      %ise ve pan yapılmıyorsa
      switch handles.state
        case 'grab'
          if ~handles.isPanning

            if isempty(handles.ImDat)
              return;
            else
                % resim ve piksel koordinatlarını tutan dizilerin
son
                % elemanlarını sil
              handles.ImDat(end, :)    = [];
              handles.TrueDat(end, :) = [];
            end

            delete(findobj(gca, 'Tag', 'Text'));

            for i=1:size(handles.ImDat)

text(handles.ImDat(i,1)+5,handles.ImDat(i,2),num2str(i),'FontSize',1
8,'Tag','Text');
            end

            %xytable tablosunu güncelle
            c=num2cell(handles.TrueDat);
            set(handles.xytable,'Data',c);

            set(handles.ImageLine, ...
              'xdata', handles.ImDat(:, 1), ...
              'ydata', handles.ImDat(:, 2));


            set(handles.GrabPointsBtn, 'string', sprintf('Noktalar
alınıyor (%d)', size(handles.ImDat, 1)));

            guidata(obj, handles);
          end

      end
```

```
    case {'return', 'enter'}

      switch handles.state
        case 'grab'
        grabPointsFcn(handles.GrabPointsBtn);

      end

  end
end




% resetViewFcn
%-------------------------------------------------------------------------
-------
function resetViewFcn(varargin)
% bu fonksiyon gorunumu resetler

obj = varargin{1};

handles = guidata(obj);

if ~isempty(handles.I)
  xl = get(handles.ImageAxis, 'xlim');
  yl = get(handles.ImageAxis, 'ylim');
  xd = (handles.ImLimits(1, :) - xl) / 10;
  yd = (handles.ImLimits(2, :) - yl) / 10;
  % zoomu farkedilir sekilde goster
  for id = 0:10
    set(handles.ImageAxis, ...
      'xlim', xl + id * xd, ...
      'ylim', yl + id * yd);
    drawnow;
  end
end

% odaklanmayı değiştir
loseFocusFcn(handles)




% loseFocusFcn
%-------------------------------------------------------------------------
-------
function loseFocusFcn(handles)
%  ENABLE ozelligini once kapatarak sonra orjinaline dondurerek
"focus
%  away" yap.

settings = get([handles.LoadImageBtn, ...
    handles.ResetViewBtn, ...
    handles.SaveAs, ...
    handles.Delete], ...
  'enable');
```

```
set([handles.LoadImageBtn, ...
    handles.ResetViewBtn, ...
    handles.SaveAs, ...
    handles.Delete], ...
  'enable', 'off');
drawnow;
set([handles.LoadImageBtn, ...
    handles.ResetViewBtn, ...
    handles.SaveAs, ...
    handles.Delete], ...
 {'enable'}, settings);


% winBtnDownFcn
%-----------------------------------------------------------------------
-------
%-----------------------------------------------------------------------
-------
function winBtnDownFcn(varargin)
% mouse tıklama baslayınca bu fonksiyon calısır

obj = varargin{1};

handles = guidata(obj);

if strcmpi(get(handles.timer, 'Running'), 'on') ||
isempty(handles.I)
  return;
end

handles.CurrentPointAxes = get(handles.ImageAxis, 'CurrentPoint');
handles.CurrentPointFig  = get(handles.TaraGUI, 'CurrentPoint');

switch get(handles.TaraGUI, 'SelectionType')
  case 'normal'

    % hemen tıklayıp cekmeyi engellemek
    % icin winBtnMotionPauseFcn fonksiyonunu once cagır
    set(handles.TaraGUI, ...
      'WindowButtonMotionFcn', ...
      {@winBtnMotionPauseFcn, handles,
handles.CurrentPointAxes(1,1:2), clock});

  case 'alt'

    xl = get(handles.ImageAxis, 'XLim');midX = (xl(1)+xl(2))/2;
    yl = get(handles.ImageAxis, 'YLim');midY = (yl(1)+yl(2))/2;
    figPos = get(handles.TaraGUI, 'Position');
    handles.curTitle = get(get(handles.ImageAxis, 'title'),
'string');
    set(handles.TaraGUI, ...
      'Pointer', 'custom', ...
      'PointerShapeCData'     , handles.zoomPointer, ...
      'PointerShapeHotSpot'   , handles.zoomPointerHotSpot, ...
      'WindowButtonMotionFcn' , {@zoomMotionFcn, handles, ...
                                 get(handles.TaraGUI,
'CurrentPoint'), ...
                                 figPos(4), ...
                                 size(handles.I, 2), ...
```

```
                                    midX, ...
                                    midY, ...
                                    diff(xl)/2, ...
                                    diff(yl)/2});

    set(get(handles.ImageAxis, 'title'), 'string', 'Zoom
yapılıyor...');

end

guidata(obj, handles);


% zoomMotionFcn
%-----------------------------------------------------------------------
-------
%-----------------------------------------------------------------------
-------
function zoomMotionFcn(varargin)
% Bu fonksiyon cek bırak zoom fonksiyonunu saglar. Baslangıca goreki
% pointer pozisyonu zoomun miktarını belirler.

[obj, handles, initPt, figHt, horizPx, midX, ...
  midY, rngXhalf, rngYhalf] = splitvar(varargin([1, 3:end]));

pt = get(obj, 'CurrentPoint');

% pointer lokasyonunu al goreceli olarak (y-koordinatı).
%   C^(x) * C^(-x) = 1
% uygun zoom faktorunu belirlemek icin C yi secelim
r = 30 ^ ((initPt(2) - pt(2)) / figHt);

% cok fazla zoom yapılmadını emin olalım.
% orjinal resmin boyutuna gore zoom yapalım.
% 64x zoom
if r < horizPx/64/rngXhalf/2  % zoom durdur
  set(get(handles.ImageAxis, 'title'), 'string', 'Maximum Zoom');
  set(obj, ...
    'Pointer'               , 'arrow', ...
    'WindowButtonMotionFcn' , '');
else
  set(handles.ImageAxis, ...
    'XLim', [midX - r * rngXhalf, midX + r * rngXhalf], ...
    'YLim', [midY - r * rngYhalf, midY + r * rngYhalf]);
end



% winBtnMotionPauseFcn
%-----------------------------------------------------------------------
-------
%-----------------------------------------------------------------------
-------
function winBtnMotionPauseFcn(varargin)
% X saniye surukle bırak fonksiyonunu engeller.Bu kullanıslıdır
% cunku kullanıcı tıklarken fareyi hareket ettirebilir.

[obj, handles, xy, c] = splitvar(varargin([1, 3:end]));
```

```
if etime(clock, c) > .15  % cekme olmadan once .15 saniye bekle
  set(obj, ...
    'Pointer'             , 'custom', ...
    'PointerShapeCData'   , handles.HandPointer, ...
    'PointerShapeHotSpot' , handles.HandPointerHotSpot);
  set(obj, 'WindowButtonMotionFcn', {@winBtnMotionFcn,
handles.ImageAxis, xy});
  handles.curTitle = get(get(handles.ImageAxis, 'title'), 'string');
  set(get(handles.ImageAxis, 'title'), 'string', 'Kaydırma
işlemi...');
  handles.isPanning = true;
  guidata(obj, handles);
end


%-------------------------------------------------------------------
-------
% winBtnMotionFcn
%-------------------------------------------------------------------
-------
%-------------------------------------------------------------------
-------
function winBtnMotionFcn(varargin)
% bu fonksiyon pan işlemi sırasında cagrılır

[axH, xy] = splitvar(varargin(3:4));

pt = get(axH, 'CurrentPoint');
set(axH, ...
  'xlim', get(axH, 'xlim')+(xy(1)-pt(1,1)), ...
  'ylim', get(axH, 'ylim')+(xy(2)-pt(1,2)));


% winBtnUpFcn
%-------------------------------------------------------------------
-------
%-------------------------------------------------------------------
-------
function winBtnUpFcn(varargin)
%  Mouse butonu bırakıldıgında bu calısır. Bu çift tıklamayı kontrol
etmek için yapılır.
%  Eger çift tıklama yapıldıysa, tek tıklama olayı çalıştırılmaz.

obj = varargin{1};

handles = guidata(obj);
if ~isempty(handles.I)  % gosterilen bir resim var
  switch get(obj, 'SelectionType')
    case 'normal'
      if strcmpi(get(handles.timer, 'Running'), 'off')

        %Çift tıklamanın olup olmadıgına bakmak için zamanlamayı
baslat.
        start(handles.timer);

        set(obj, ...
          'Pointer'               , handles.curPointer, ...
          'PointerShapeCData'     , handles.curPointerData.CData,
...
```

```
          'PointerShapeHotSpot'    , handles.curPointerData.HotSpot,
...
          'WindowButtonMotionFcn' , {@pointerFcn, handles,
handles.curPointer});

        set(get(handles.ImageAxis, 'title'), 'string',
handles.curTitle);
      end

    case 'alt'
      set(obj, ...
        'Pointer'               , handles.curPointer, ...
        'PointerShapeCData'     , handles.curPointerData.CData, ...
        'PointerShapeHotSpot'   , handles.curPointerData.HotSpot,
...
        'WindowButtonMotionFcn' , {@pointerFcn, handles,
handles.curPointer});

      set(get(handles.ImageAxis, 'title'), 'string',
handles.curTitle);
  end
end


% btnUpTimerFcn
%-------------------------------------------------------------------------
-------
%-------------------------------------------------------------------------
-------
function btnUpTimerFcn(varargin)

figH = varargin{3};

handles = guidata(figH);

handles.TrueDat;


switch get(handles.TaraGUI, 'SelectionType')
  case 'open' % çift tıklama
    % bu gorunumu merkezileştirecek

    % su anki birimleri al
    un = get([0, handles.TaraGUI, handles.ImageAxis], 'units');

    set([0, handles.TaraGUI, handles.ImageAxis], 'units', 'pixels');
    pt     = get(0, 'PointerLocation');
    figPos = get(handles.TaraGUI, 'Position');
    pt2    = pt - figPos(1:2);
    axPos  = get(handles.ImageAxis, 'position');

    % pointerin resim eksenleri içinde olup olmadığını kontrol et
    if pt2(1) > axPos(1) && pt2(1) < axPos(1)+axPos(3) && ...
        pt2(2) > axPos(2) && pt2(2) < axPos(2)+axPos(4)
      xl = get(handles.ImageAxis, 'xlim'); xrng = diff(xl);
      yl = get(handles.ImageAxis, 'ylim'); yrng = diff(yl);
      x  = (pt2(1)-axPos(1))/axPos(3)*xrng+xl(1);
      y  = (axPos(2)+axPos(4)-pt2(2))/axPos(4)*yrng+yl(1);
```

```
      % Yeni pozisyona kayma hareketi yap.bu gorunum degismesini
daha iyi
      % algılamamızı saglar

      % pointerin yerine gore ne kadar hızlı bir geçis yapacagımızı
hesapla
      interval = ceil(norm((figPos(1:2)+axPos(1:2)+axPos(3:4)/2) -
pt)/30);

      if interval
        ld = (([x, y] - [xrng, yrng]/2) - [xl(1), yl(1)])/interval;
        pd = ((figPos(1:2)+axPos(1:2)+axPos(3:4)/2) - pt)/interval;
      else
        % interval == 0, ayni nokta anlamına geliyor
        ld = [0, 0];
        pd = [0, 0];
      end

      for id = 0:interval
        set(handles.ImageAxis, ...
          'xlim', xl + id * ld(1), ...
          'ylim', yl + id * ld(2));

        % pointerin yerini merkez yap
        set(0, ...
          'PointerLocation', pt + id * pd);
        drawnow;
      end
    end

    % UNITS resetle
    set([0, handles.TaraGUI, handles.ImageAxis], {'units'}, un);

  case 'normal'  % tek tıklama

    switch handles.state
      case 'grab'
        if ~handles.isPanning && ~isempty(handles.CurrentPointAxes)

          % ekranda tıklanan noktanın X ve Y koordinatlarını
alıyoruz
          X = handles.CurrentPointAxes(1, 1);
          Y = handles.CurrentPointAxes(1, 2);
          % Piksel koordinat sisteminden resim koordinat sistemi
          % dönüşüm yapıyoruz ve newX , newY olarak iki ayrı
değişkene
          % gereken atama işlemini yapıyoruz.
          newX=(X-handles.ResimYatay/2)*handles.PikselDeger/1000;
          newY=(-Y+handles.ResimDikey/2)*handles.PikselDeger/1000;
          xycolor=impixel(handles.I,X,Y); %renk datası RGB
formatında
          % seçilen nokta kaçıncı seçilen ise ekrana yazdırıyoruz

text(X+5,Y,num2str(size(handles.TrueDat,1)+1),'FontSize',18,'Tag','T
ext');



          % resmin üzerindeki nokta
          % ImDat piksel koordinatlarını saklayacak
```

```
            handles.ImDat(end + 1, :) = [X, Y];
            % TrueDat ise resim koordinatlarını saklayacak
            handles.TrueDat(end + 1, :) = [newX,newY];
            c=num2cell(handles.TrueDat);
            set(handles.xytable,'Data',c)

              % seçilen noktanın koordinatına işaret olarak renkli bir
daire
              % çizdiriyoruz.
              set(handles.ImageLine, ...
              'xdata', handles.ImDat(:, 1), ...
              'ydata', handles.ImDat(:, 2));

            set(handles.GrabPointsBtn, 'string', sprintf('Yakalanan
nokta (%d)', size(handles.ImDat, 1)));

          end



      end
end

handles.CurrentPointAxes = [];
handles.CurrentPointFig  = [];
handles.isPanning        = false;
guidata(handles.TaraGUI, handles);


% grabPointsFcn
%-----------------------------------------------------------------------
-------
%-----------------------------------------------------------------------
-------
function grabPointsFcn(varargin)
% Bu fonksiyon kullanıcının resmin üzerinden
% seçerek data noktalarının elde edilmesini sağlar.

obj = varargin{1};

handles = guidata(obj);
if (handles.PikselDeger==1)
   warndlg('Piksel ve Odak değerlerini bizahmet giriniz.','!! Uyarı
!!')
   return
end
switch get(obj, 'value')
  case 0  % nokta yakalamayı başlatalım

      set(handles.ImageAxis, ...
      'xlim', handles.ImLimits(1,:), ...
      'ylim', handles.ImLimits(2,:));
    set(handles.ImageLine, 'xdata', NaN, 'ydata', NaN);
    handles.curTitle = {'Gereken noktları tıklayarak seçiniz.', ...
    '<BACKSPACE> veya <DEL> önceki noktayı siler. <ENTER>
tamamlar.'};
    set(get(handles.ImageAxis, 'title'), ...
      'string', handles.curTitle);
```

```
    handles.curPointer          = 'crosshair';
    set(handles.TaraGUI, ...
      'WindowButtonMotionFcn', {@pointerFcn, handles,
handles.curPointer});

    set(obj, ...
      'value'           , 1, ...
      'string'          , 'Yakalanan nokta (0)', ...
      'backgroundcolor' , handles.arkaplan4);

    set([handles.LoadImageBtn, ...
        handles.SaveAs, ...
        handles.Delete], ...
      'enable', 'off');

     handles.state = 'grab';

  case 1  % nokta yakalamayı bitir
    set(obj, ...
      'value'           , 0, ...
      'string'          , 'Noktaları yakala', ...
      'backgroundcolor' , handles.arkaplan1);

    handles.curTitle = '';
    set(get(handles.ImageAxis, 'title'), 'string', '');
    handles.curPointer            = 'arrow';
    set(handles.TaraGUI, 'WindowButtonMotionFcn', {@pointerFcn,
handles, handles.curPointer});

    set([handles.LoadImageBtn, ...
        handles.SaveAs, ...
        handles.Delete], ...
      'enable', 'on');

   %TruDat boş değilse
    if ~isempty(handles.TrueDat) % bazı noktalar yakalandı demektir
        % xytable cell türünde değişken kabul ediyor
        % dolayısıyla TrueDat değişkenini cell türüne cevirip 'c'
değişkenine
        % atıyoruz ve xytablo'nun Data property değerini 'c'ye
eşitliyoruz.
        c=num2cell(handles.TrueDat);
          set(handles.xytable,'Data',c);
     end

    handles.state = 'normal';
end

guidata(obj, handles);


%-------------------------------------------------------------------
-------
% PikselImageFcn
    function PikselImageFcn(varargin)
% bu fonksiyon piksel değerini al butonuna tıklandıgında calısır.

obj = varargin{1};
handles = guidata(obj) ;
```

```
handles.PikselDeger=str2double(get(handles.Piksel, 'string'));
guidata(handles.TaraGUI, handles);

%--------------------------------------------------------------------
-------
% OdakImageFcn
%--------------------------------------------------------------------
-------
        function OdakImageFcn(varargin)
% bu fonksiyon odak değerini al butonuna tıklandıgında calısır.

obj = varargin{1};
handles = guidata(obj);
handles.OdakDeger=str2double(get(handles.Odak, 'string'));
guidata(handles.TaraGUI, handles);

%--------------------------------------------------------------------
-------
% AraziFcn
%--------------------------------------------------------------------
-------
  function Arazi(varargin)

  % dosya browse et arazi için
[fname, pathname] = uigetfile(...
    {
    '*.*', 'Tüm dosyalar (*.*)'}, 'Bir Arazi dosyası seçin');
  if ischar(fname)
    filename = fullfile(pathname, fname);
  else
    return;
  end

filename=load(filename);
x=filename(:,2);
y=filename(:,3);
z=filename(:,4);
h=figure;

% cubuklar
set(h,'Renderer','OpenGL');
stem3(x,y+2,z+50,'fill' ,'MarkerSize',2 ,'LineWidth' ,8 );
xlabel('X');
ylabel('Y');
zlabel('Z');
close ; % su anki cubuklu gorunumu kapat ve 3 boyutlu yüzeye devam
et

... 3 boyutlu yüzey
screen_size = get(0, 'ScreenSize');
 figure('Position', [0 0 screen_size(3)+10 screen_size(4)+1]
...'Units','normalized','Position',[0 0 1 1],
  ,'color','yellow','name','Opengl Render','NumberTitle', 'off');
set(gcf,'Renderer','OpenGL');

gx=0:5:500;
gy=0:5:300;

[XI, YI] = meshgrid(gx,gy);
g=griddata(x,y,z,XI,YI,'v4');
```

```
load('MyColorguzel2','mycmap')
colormap(mycmap);

s=surf(gx,gy,g...'FaceColor','texturemap'
);

set(gca,'Color',[1 1 0],...
       'GridLineStyle','none'...
       ,'FontName','times',...
       'FontAngle','italic',...
       'FontSize',14,...
       'XColor',[1 1 0],...
       'YColor',[1 1 0],...
       'ZColor',[1 1 0]);
grid off;

camproj('perspective')

lighting phong;
shading interp
%light
camlight left
%box off;
xlabel('X');
ylabel('Y');
zlabel('Z');

rotate3d

for i=1:100
rotate(s,[0 0 1],-2);
pause(0.1);
drawnow;
end
```

```
function Spaceresection(varargin)
% Bu fonksiyon space resection programını calıstırır.

obj = varargin{1};
handles = guidata(obj);


%'dusey_resim_koor.txt' MATRİSİ HESAPLANIR.-------------------------
-------

f=handles.OdakDeger;

resim=load('resim_koor.txt');
arazi=load('arazi_koor.txt');


[mm,nn]=size(resim);
```

```
X1=arazi(1,2);
Y1=arazi(1,3);
Z1=arazi(1,4);

X2=arazi(2,2);
Y2=arazi(2,3);
Z2=arazi(2,4);

xa=resim(1,2);
ya=resim(1,3);
xb=resim(2,2);
yb=resim(2,3);

%İLK İKİ NOKTA SEÇİLEREK ARALARINDAKİ YATAY UZAKLIK BULUNUR.--------
-------

AB=sqrt((X2-X1).^2+(Y2-Y1).^2);

A=(1/f^2)*(xb^2-2*xb*xa+xa^2+yb^2-2*yb*ya+ya^2);
B=(2/f^2)*(-xb^2*Z2+xb*xa*Z1+xb*xa*Z2-xa^2*Z1-
yb^2*Z2+yb*ya*Z1+yb*ya*Z2-ya^2*Z1);
C=[(1/f^2)*(xb^2*Z2^2-2*xb*xa*Z2*Z1+xa^2*Z1^2+yb^2*Z2^2-
2*yb*ya*Z2*Z1+ya^2*Z1^2)]-(AB^2);

%H DEĞERİNİ HESAPLIYORUZ.

delta=sqrt((B^2)-(4*A*C));

H=(-B+delta)/(2*A);

Matris_C(mm,2)=0;

    for i=1:mm;
        Matris_C(i,1)=resim(i,2)*[(H-arazi(i,4))/f];
        Matris_C(i,2)=resim(i,3)*[(H-arazi(i,4))/f];
    end

save 'dusey_resim_koor.txt' 'Matris_C' -ascii

%BENZERLİK DÖNÜŞÜMÜ BİLİNMEYENLERİNİN HESABI----------------------
-------

olcu=load('dusey_resim_koor.txt');

x=olcu(:,1);
y=olcu(:,2);
AX=arazi(:,2);
AY=arazi(:,3);
[m,n]=size(resim);

Matris_A(2*m,4)=0;

 for i=0:m-1;
    Matris_A(2*i+1,1)=x(i+1,1);
    Matris_A(2*i+1,2)=-y(i+1,1);
    Matris_A(2*i+1,3)=1;
    Matris_A(2*i+1,4)=0;
```

```
    Matris_A(2*i+2,1)=y(i+1,1);
    Matris_A(2*i+2,2)=x(i+1,1);
    Matris_A(2*i+2,3)=0;
    Matris_A(2*i+2,4)=1;
 end;

 Matris_L(2*m,1)=0;

 for i=0:m-1;
      Matris_L(2*i+1,1)=AX(i+1,1);
      Matris_L(2*i+2,1)=AY(i+1,1);
 end;

 Matris_N = transpose(Matris_A)*(Matris_A);
 Matris_n = transpose(Matris_A)*(Matris_L);
 Matris_X= inv(Matris_N)*Matris_n;

 a=Matris_X(1,1);
 b=Matris_X(2,1);
 Tx=Matris_X(3,1);
 Ty=Matris_X(4,1);

 ome=0;
 phi=0;
 kappa=atan(b/a);

  %KAPPA AÇISI DÜZELTİLİYOR.----------------------------------------
-------

        dkappa=kappa*180/pi;

       if (dkappa <= -180)
            kat=dkappa/-360;
            kat=ceil(kat);
            dkappa = dkappa+((kat-1)*360);
       end;
       if ((-180 < dkappa) & (dkappa < 0))
           dkappa = dkappa+180;
       end;
       if ((0 <= dkappa) & (dkappa <= 180))
           dkappa = dkappa;
       end;
       if (dkappa > 360)
            kat=dkappa/360;
            kat=ceil(kat);
            dkappa = dkappa-((kat-1)*360);
       end;

        kappa = dkappa*pi/180;

%DÖNÜKLÜK MATRİSİ TANIMLANIYOR.-----------------------------------
-------

Matris_M(3,3)=0;

%-----------------------------------------------------------------
-------
%İTERASYON BAŞLIYOR.
```

```
for k=1:100;

 m11 = cos(phi)*cos(kappa);
 m12 = sin(ome)*sin(phi)*cos(kappa)+cos(ome)*sin(kappa);
 m13 = -cos(ome)*sin(phi)*cos(kappa)+sin(ome)*sin(kappa);
 m21 = -cos(phi)*sin(kappa);
 m22 = -sin(ome)*sin(phi)*sin(kappa)+cos(ome)*cos(kappa);
 m23 = cos(ome)*sin(phi)*sin(kappa)+sin(ome)*cos(kappa);
 m31 = sin(phi);
 m32 = -sin(ome)* cos(phi);
 m33 = cos(ome)*cos(phi);

 Matris_M(1,1)=m11;
 Matris_M(1,2)=m12;
 Matris_M(1,3)=m13;
 Matris_M(2,1)=m21;
 Matris_M(2,2)=m22;
 Matris_M(2,3)=m23;
 Matris_M(3,1)=m31;
 Matris_M(3,2)=m32;
 Matris_M(3,3)=m33;

 %KATSAYILAR (r,s ve q) MATRİSİ OLUŞTURULUR.-----------------------
-------

 Matris_K(3*m,1)=0;

 for i=0:m-1;

 r(3*i+1,1)=m11*(arazi(i+1,2)-Tx)+m12*(arazi(i+1,3)-
Ty)+m13*(arazi(i+1,4)-H);
 s(3*i+2,1)=m21*(arazi(i+1,2)-Tx)+m22*(arazi(i+1,3)-
Ty)+m23*(arazi(i+1,4)-H);
 q(3*i+3,1)=m31*(arazi(i+1,2)-Tx)+m32*(arazi(i+1,3)-
Ty)+m33*(arazi(i+1,4)-H);

 end

 for i=0:m-1;

 Matris_K(3*i+1,1)=r(3*i+1,1);
 Matris_K(3*i+2,1)=s(3*i+2,1);
 Matris_K(3*i+3,1)=q(3*i+3,1);

 end

 %MATRİS_B KATSAYILAR MATRİSİ OLUŞTURULUR.-------------------------
-------

 Matris_B(2*m,6)=0;

 for i=0:m-1;
     DX(i+1)=arazi(i+1,2)-Tx;
     DY(i+1)=arazi(i+1,3)-Ty;
     DZ(i+1)=arazi(i+1,4)-H;
 end

 DX=transpose(DX);
 DY=transpose(DY);
```

```
 DZ=transpose(DZ);

 for i=0:m-1;
     Matris_B(2*i+1,1)=
(f/Matris_K(3*i+3,1).^2)*[Matris_K(3*i+1,1)*(-
m33*DY(i+1)+m32*DZ(i+1))-Matris_K(3*i+3,1)*(-
m13*DY(i+1)+m12*DZ(i+1))];
     Matris_B(2*i+2,1)=
(f/Matris_K(3*i+3,1).^2)*[Matris_K(3*i+2,1)*(-
m33*DY(i+1)+m32*DZ(i+1))-Matris_K(3*i+3,1)*(-
m23*DY(i+1)+m22*DZ(i+1))];
     Matris_B(2*i+1,2)=
(f/Matris_K(3*i+3,1).^2)*[Matris_K(3*i+1,1)*(cos(phi)*DX(i+1)+sin(om
e)*sin(phi)*DY(i+1)-cos(ome)*sin(phi)*DZ(i+1))-Matris_K(3*i+3,1)*(-
sin(phi)*cos(kappa)*DX(i+1)+sin(ome)*cos(phi)*cos(kappa)*DY(i+1)-
cos(ome)*cos(phi)*cos(kappa)*DZ(i+1))];
     Matris_B(2*i+2,2)=
(f/Matris_K(3*i+3,1).^2)*[Matris_K(3*i+2,1)*(cos(phi)*DX(i+1)+sin(om
e)*sin(phi)*DY(i+1)-cos(ome)*sin(phi)*DZ(i+1))-
Matris_K(3*i+3,1)*(sin(phi)*sin(kappa)*DX(i+1)-
sin(ome)*cos(phi)*sin(kappa)*DY(i+1)+cos(ome)*cos(phi)*sin(kappa)*DZ
(i+1))];
     Matris_B(2*i+1,3)= (-
f/Matris_K(3*i+3,1))*(m21*DX(i+1)+m22*DY(i+1)+m23*DZ(i+1));
     Matris_B(2*i+2,3)=
(f/Matris_K(3*i+3,1))*(m11*DX(i+1)+m12*DY(i+1)+m13*DZ(i+1));
     Matris_B(2*i+1,4)= -
1*(f/Matris_K(3*i+3,1).^2)*(Matris_K(3*i+1,1)*m31-q(3*i+3,1)*m11);
     Matris_B(2*i+2,4)= -
1*(f/Matris_K(3*i+3,1).^2)*(Matris_K(3*i+2,1)*m31-q(3*i+3,1)*m21);
     Matris_B(2*i+1,5)= -
1*(f/Matris_K(3*i+3,1).^2)*(Matris_K(3*i+1,1)*m32-q(3*i+3,1)*m12);
     Matris_B(2*i+2,5)= -
1*(f/Matris_K(3*i+3,1).^2)*(Matris_K(3*i+2,1)*m32-q(3*i+3,1)*m22);
     Matris_B(2*i+1,6)= -
1*(f/Matris_K(3*i+3,1).^2)*(Matris_K(3*i+1,1)*m33-q(3*i+3,1)*m13);
     Matris_B(2*i+2,6)= -
1*(f/Matris_K(3*i+3,1).^2)*(Matris_K(3*i+2,1)*m33-q(3*i+3,1)*m23);
 end

 %MATRİS_E OLUŞTURULUR.-----------------------------------------------
-------

 Matris_E(2*m,1)=0;

 xo=0;
 yo=0;

 for i=0:m-1;
     J(2*i+1,1)=resim(i+1,2)-xo+f*(r(3*i+1,1)/q(3*i+3,1));
     K(2*i+2,1)=resim(i+1,3)-yo+f*(s(3*i+2,1)/q(3*i+3,1));
 end

 for i=0:m-1;

 Matris_E(2*i+1,1)=J(2*i+1,1);
 Matris_E(2*i+2,1)=K(2*i+2,1);

 end
```

```
 %Matris_D BİLİNMEYENLER MATRİSİ OLUŞTURULUR.----------------------
-------

 Matris_D(2*m,1)=0;


Matris_D=inv(transpose(Matris_B)*Matris_B)*(transpose(Matris_B)*Matr
is_E);
 Matris_D;
 Matris_V=(Matris_B*Matris_D)-Matris_E;

 %DÖNÜKLÜK AÇILARI TAZELENİR----------------------------------------
-------

 dome=ome+Matris_D(1,1);
 dphi=phi+Matris_D(2,1);
 dkappa=kappa+Matris_D(3,1);

 %RÇM DEĞERLERİ TAZELENİR-------------------------------------------
-------

 XL=Tx+Matris_D(4,1);
 YL=Ty+Matris_D(5,1);
 ZL=H+Matris_D(6,1);

 %RESİM KOORDİNATLARI TAZELENİR-------------------------------------
-------

for i=0:m-1;
     resim(i+1,2)=resim(i+1,2)+Matris_V(2*i+1,1);
     resim(i+1,3)=resim(i+1,3)+Matris_V(2*i+2,1);
end

 %DÖNÜKLÜK AÇILARI KONTROL EDİLİR-----------------------------------
-------

 %OMEGA AÇISI DÜZELTİLİYOR.-----------------------------------------
-------

 dome=dome*180/pi;

        if (dome<=-180)
           kat=dome/-360;
           kat=ceil(kat);
           dome = dome+((kat+1)*360);
        end;
        if ((-180<dome) & (dome<0))
           dome = dome;
        end;
        if ((0<=dome) & (dome<=180))
           dome = dome;
        end;
        if (dome>=180)
            dome=dome-180;
        end;
        if (dome>360)
           kat=dome/360;
           kat=ceil(kat);
           dome = dome-((kat-1)*360);
```

```
            if (dome>=180)
                dome=dome-180;
            end;
        end;

        ome=dome*pi/180;

 %PHİ AÇISI DÜZELTİLİYOR.-----------------------------------------
-------

 dphi = dphi*180/pi;

        if (dphi<=-90)
            kat=dphi/-180;
            kat=ceil(kat);
            dphi = dphi+((kat+1)*180);
            if (dphi>=180)
               dphi=dphi-90;
            end;
         end;
        if ((-90<dphi) & (dphi<0))
            dphi = dphi+90;
         end;
        if ((0<=dphi) & (dphi<=90))
            dphi = dphi;
        end;
        if (dphi>90)
            kat=dphi/180;
            kat=ceil(kat);
            dphi = dphi-((kat-1)*180);
            if (dphi>=90)
               dphi=dphi-90;
            end;
        end;

        phi=dphi*pi/180;

 %KAPPA AÇISI DÜZELTİLİYOR.----------------------------------------
-------

 dkappa = dkappa*180/pi;

       if (dkappa <= -180)
            kat=dkappa/-360;
            kat=ceil(kat);
            dkappa = dkappa+((kat-1)*360);
       end;
       if ((-180 < dkappa) & (dkappa < 0))
           dkappa = dkappa;
       end;
       if ((0 <= dkappa) & (dkappa <= 180))
           dkappa = dkappa;
       end;
       if (dkappa > 360)
            kat=dkappa/360;
            kat=ceil(kat);
            dkappa = dkappa-((kat-1)*360);
       end;

        kappa = dkappa*pi/180;
```

```
%DENGELEME KONTROLLERİ-----------------------------------------------
-------

Kont=sum(Matris_V);
th=abs(Kont);

%iterasyonun kesilmesi için şart Kont değeridir.

if  (th <= 0.00001)
     break
end;
end;

%İTERASYON SONU...---------------------------------------------------
-------
%-------------------------------------------------------------------
-------

%DÖNME MATRİSİ TEKRAR DOLDURULUR.-----------------------------------
-------

 m11 = cos(phi)*cos(kappa);
 m12 = sin(ome)*sin(phi)*cos(kappa)+cos(ome)*sin(kappa);
 m13 = -cos(ome)*sin(phi)*cos(kappa)+sin(ome)*sin(kappa);
 m21 = -cos(phi)*sin(kappa);
 m22 = -sin(ome)*sin(phi)*sin(kappa)+cos(ome)*cos(kappa);
 m23 = cos(ome)*sin(phi)*sin(kappa)+sin(ome)*cos(kappa);
 m31 = sin(phi);
 m32 = -sin(ome)* cos(phi);
 m33 = cos(ome)*cos(phi);




%YAKLAŞIK DEĞERLER MATRİSİ YAZDIRILIR.------------------------------
------

 Matris_Yaklasik(6,1)=0;
 Matris_Yaklasik(1,1)=XL;
 Matris_Yaklasik(2,1)=YL;
 Matris_Yaklasik(3,1)=ZL;
 Matris_Yaklasik(4,1)=dome;
 Matris_Yaklasik(5,1)=dphi;
 Matris_Yaklasik(6,1)=dkappa;

 save 'yaklasik_deger_matrisi.txt' 'Matris_Yaklasik' -ascii

guidata(obj, handles);

 function varargout = splitvar(varargout)
% Bu fonksiyon giriş argumanlarını ayrı ayrı degiskenlere atar.
```

Not: 3d render işlemi esnasında

```
load('MyColorguzel2','mycmap')
```

kod satırı "Mycolorguzel2.mat" isminde bir colormap dosyasını açmaktadır. Dolayısıyla böyle bir dosyanın oluşturulup aynı klasörde bulunması gereklidir. Colormap oluşturulması hakkında bilgi için Matlab'da "Help" kısmına bakılabilir.