GÖRÜNTÜ İŞLEME TEKNİKLERİ KULLANILARAK ÜRETİM BANDI ÜZERİNDE HAREKET EDEN KUTULARIN İZLENMESİ

Hüseyin Nihal KARACA

Yüksek Lisans Tezi

Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı

Aralık – 2004

JÜRİ VE ENSTİTÜ ONAYI

Hüseyin Nihal Karaca'nın "Görüntü İşleme Teknikleri Kullanılarak Üretim Bandı Üzerinde Hareket Eden Kutuların İzlenmesi" başlıklı Bilgisayar Mühendisliği Anabilim Dalındaki, Yüksek Lisans tezi 09.12.2004 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	İmza
Üye (Tez Danışmanı)	: Yard. Doç. Dr. Yusuf Oysal	
Üye	: Yard. Doç. Dr. Cüneyt Akınlar	
Üye	: Yard. Doç. Dr. Ö. Nezih Gerek	

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 12.01.2005 **tarih ve** 3/2 **sayılı kararıyla onaylanmıştır.**

Enstitü Müdürü
Prof. Dr. Altuğ İFTAR
Fen Bilimleri Enstitüsü
Müdürü

ÖZET

Yüksek Lisans Tezi

GÖRÜNTÜ İŞLEME TEKNİKLERİ KULLANILARAK ÜRETİM BANDI ÜZERİNDE HAREKET EDEN KUTULARIN İZLENMESİ

HÜSEYİN NİHAL KARACA

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yard. Doç. Dr. Yusuf OYSAL
2004, 88 Sayfa

Bu tezde, bir üretim bandı üzerinde ilerleyen kutuların izlenmesi için bir görü sistemi tasarlanmıştır. Görü sistemi, üretim bandı üzerine eşit aralıklarla yerleştirilmiş 4 kameradan saniyede 30 gri tonlama görüntüsünü girdi olarak alarak her kutunun üretim bandı üzerindeki konumunu gerçek zamanlı olarak hesaplamaktadır. İzlenen kutulardan elde edilen köşe koordinatları daha sonra bir denetleyiciye gönderilerek kutuların üretim bandı çıkışında tek sırada dizilmeleri sağlanmaktadır. Temel izleme algoritması olarak Lucas Kanade Tomasi(LKT) özellik izleme algoritması kullanılmıştır: Kutuların bir önceki görüntüdeki köşe noktaları LKT algoritmasına verilmiş ve mevcut görüntüdeki yeni köşe noktaları hesaplanmıştır. Lakin, bu yaklaşım ile kutular ancak bir kaç görüntü boyunca izlenebilmiş ve uzun süreli bir başarı elde edilememiştir. Kutuların başarılı olarak izlenebilmeleri için LKT algoritması, bir özellik algılama ve doğru uyarlama algoritması ile desteklenmiştir. Kullanılan algoritmalarla kutular gerçek zamanlı olarak izlenerek kesin başarı sağlanmıştır.

Anahtar Kelimeler: Görüntü işleme, izleme, Lucas Kanade Tomasi algoritması, özellik algılama, doğru uyarlama

ABSTRACT

Master of Science Thesis

TRACKING OF THE PARCELS MOVING ON THE CONVEYER BELT BY USING IMAGE PROCESSING TECHNIQUES.

HÜSEYİN NİHAL KARACA

Anadolu University
Graduate School of Sciences
Computer Engineering Program

Supervisor: Assist. Prof. Dr. Yusuf OYSAL
2004, 88 Pages

In this thesis, the problem of designing a vision system for tracking parcels moving on a conveyor belt is considered. The vision system incorporates 30fps grayscale image input from 4 cameras equally spaced over the conveyor belt, and computes the location of each parcel over the belt in real-time. The corner points of the tracked parcels are then sent to a controller, which arranges the parcels into a single line at the output. Lucas Kanade Tomasi (LKT) feature tracking algorithm is used as the base of our tracking algorithm: Corner points of a parcel from the previous frame are fed into the LKT algorithm to get the new corner coordinates of the parcel in the current frame. Although this approach tracks a parcel for a few frames over the belt, it is not enough for long-term successful tracking of a parcel. To achieve successful parcel tracking, an edge mapping and a feature detection are added as a refinement step following LKT corner tracking. The proposed algorithms are novel and are able to track parcels in real-time.

Keywords: Image processing, tracking, Lucas Kanade Tomasi algorithm, feature detection, edge mapping

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

SİMGELER ve KISALTMALAR DİZİNİ

a	Ortalama ağırlık(önce w idi)
B	Eğriler arasında fark
C	Yerel yapı matrisi
D	Görüntü üzerinde bir piksel
d	Uzaklık veya görüntü hızı
d_x	x ekseninde uzaklık
d_y	y ekseninde uzaklık
f	Frekans
e	Gradyan veya iki görüntü arasındaki fark ile hesaplanan vektör
E	Eğrilti
F	Herhangi bir eğri
g	Gradyan veya optik akış
G	Herhangi bir eğri
h	Yatay eşitsizlik
H	Hessian Matrisi
i_x ve i_y	Pencere komşulukları
I	Görüntü ya da görüntü dizisi
I_x, I_y, I_{xy}	Görüntü yoğunluk fonksiyonu türevleri
J	Görüntü ya da görüntü dizisi
k	Yineleme
K	Katsayı Matrisi
KD	Görüntü üzerinde bir piksel
l	Gaussian filtresi
KFT	Karelenmiş farkların toplamı
L	Norm
LKT	Lucas Kanade Tomasi
M	Görüntü piramit seviyesi
m	Eğim
n	Hata
N	İki görüntü arasındaki fark fonksiyonu

n_x ve n_y	Görüntü genişliği
p	Parametre vektörü
P	İki boyutlu pencere
t	Milisaniye cinsinden zaman
T	Vektör indeksi
T(x)	Şablon görüntüsü
u	Görüntü üzerinde bir nokta
v	Hız
x	Üretim bandının uzunlamasına eksen koordinatı
y	Üretim bandının yanlamasına eksen koordinatı
ε	Eşik değeri
τ	Zaman değişimi
ξ	x ekseninde değişim
η	y ekseninde değişim
λ	Özdeğer
σ_x ve σ_y	Bir koordinatın ondalıklı kısımları
θ	Kutunun x ekseni ile yaptığı açı
α	Kutunun merkezinden geçen ilgili kenar doğrusu paraleli ile x ekseni arasında kalan açı

1. GİRİŞ VE AMAÇ

1.1 Otomasyon

Sanayi devriminin başlangıcının aksine, yakın geçmişimizde üretmek tek başına yeterli olmaktan uzaklaştı. Tüm dünyanın açık bir pazar haline geldiği rekabetçi koşullarda üretimi; hızlı, standart, güvenli, nihayet verimli kılmak bir zorunluluk haline geldi. Endüstride bu zorunluluğun karşılığı şüphesiz ki otomasyondu. Otomasyon, endüstriyel, tarımsal, idari ve bilimsel işlerin yürütülmesinde insan müdahalesinin bir ölçüde veya tamamen ortadan kaldırılması; otomatikleştirme işlemidir. Otomasyon, genel olarak tasarlanan sürecin yeniden elden geçirilmesi ve edinilmiş alışkanlıklarla geleneksel çözümlerin birlikte yardımını gerektirir. Kameralar, algılayıcılar, koşullayıcılar, yükselteçler, düzenleyiciler, bilgi kaynakları ve nihayet bilgisayarlı ekipmanlar veya bilgisayarın bizzat kendisidir.

Otomasyona geçmiş bir tesiste üretim maliyetleri düşer. Kapasite, verimlilik ve üretim güvenliği artar. Üretim denetlenerek hatalı ürün çıkarma ihtimalini en aza indirilir. Sistemlerin akıllı birimler haline dönüştürülmesinin sonucu her aksamın ne zaman ne iş yapacağı programlanmış olacaktır. Sistemlerin elektronik olarak kumanda edilmesi birim zamanda daha fazla üretim yapabilme mümkün olmaktadır. Bu kazanç sadece sistemin daha hızlı çalışmasını temin etmekle değil aynı zamanda arızaların daha hızlı giderilmesi, insana dayalı hataların, dikkatsizliklerin ortadan kalkması gibi aksaklıklarında denetim altına alınmasından kaynaklanmaktadır.

Görüntü işleme son yıllarda otomasyon teknolojisinde önemli bir yer edinmeye başlamıştır. Nesnelere dokunmadan, nesneleri deforme etmeden ölçüm yapabilme, nesneleri tanımlayabilme, ayırt edebilme, izleyebilme mümkün olmaktadır. Bu yöntem sayesinde düşük maliyetli, daha hızlı, daha kaliteli üretim yapmak mümkün hale gelmektedir. Günümüzde kameraların fiyatlarının düşmesi, bilgisayar hızlarının artmasının görüntülerin gerçek zamanda işlenebilmesine olanak sağlaması görüntü işleme teknolojisinin otomasyona entegrasyonunda önemli bir rol oynamaktadır [8].

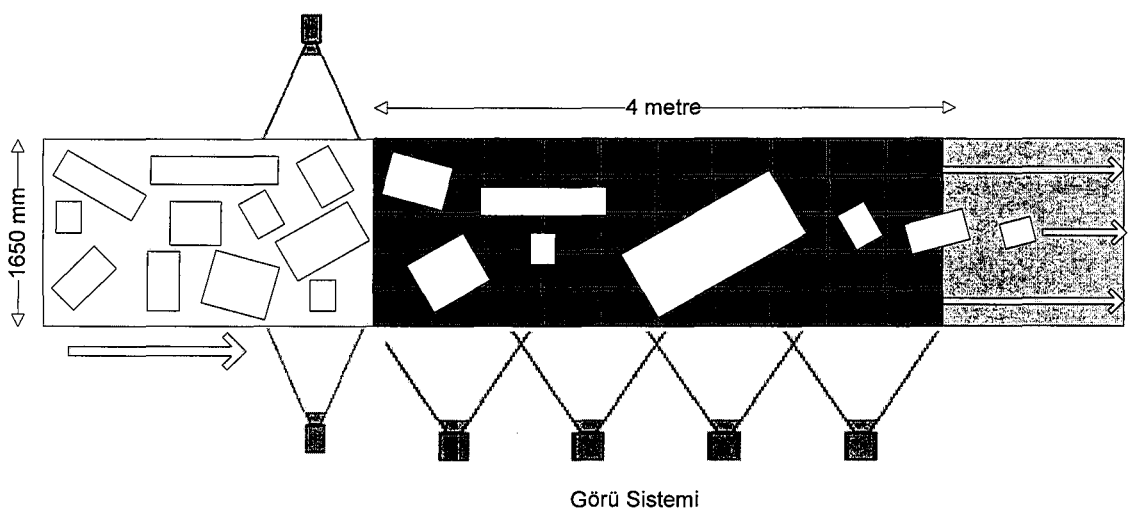
Otomasyon işlemlerine talep arttıkça, hareket ve görü sistemlerinin kullanılmasında da belirgin bir artış olmuştur. Bunun en önemli sebeplerinden

birisi, insan işletmenleri için ayrılan bazı işlerin hareket ve görü sistemleri ile daha hızlı, daha verimli ve daha düşük maliyetlerde yapılabilmesidir. Bir diğer sebep ise, kolay kullanılabilir yazılım araçları ve arabirimler ile bu sistemler uzmanlık istemeksizin daha kolay erişilebilir hale gelmektedir. Bilgisayar ve kamera teknolojisinin ilerlemesi ile eş zamanlı olarak gerçek zamanlı sistemlere olan ihtiyacın artması görü sistemlerinin otomasyonda yerini daha da sağlamlaştırmaktadır.

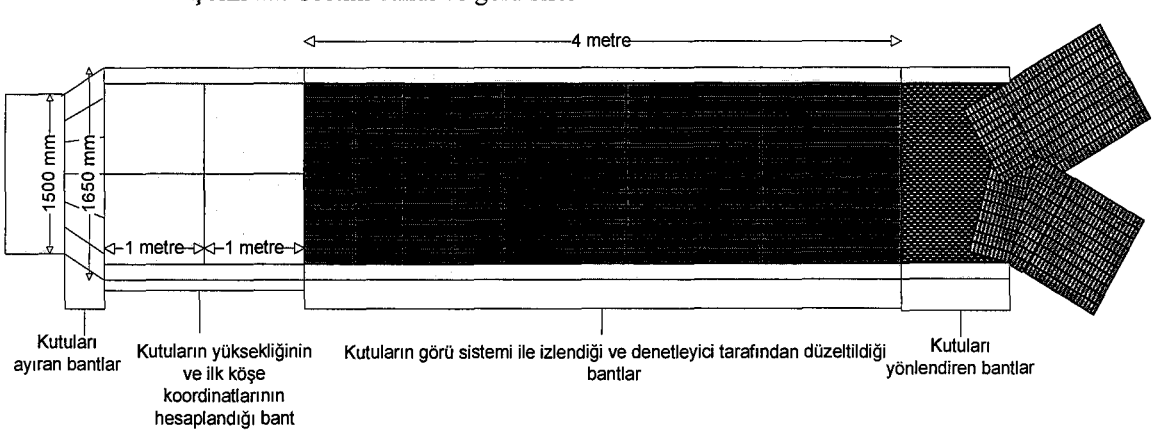
Bu tezde bir otomasyon sürecinde kullanılan bir üretim bandının başlangıç kısmının üzerinde düzensiz olarak yerleşik durumda ilerleyen kutuların, üretim bandının çıkış kısmında tek sırada düzgün olarak çıkmasını sağlayacak bir gerçek zamanlı sistemin görü sistemi kısmı tasarlanmıştır.

1.2 Problem

Üretim bandı Şekil 1.1 ve Şekil 1.2' de görülmektedir.



Şekil 1.1 Üretim bandı ve görü sistemi



Şekil 1.2 Üretim bandı üzerindeki farklı bantlar

Üretim bandı dört parçaya ayrılmıştır. Bunların ilki kutuları birbirinden ayıran banttır. Başlangıç kısmı 1500 mm genişliğinde olan bandın bu kısmının bitiş kısmı 1650 mm'dir. Birbiri ile bitişik birden fazla banttan oluşan bu kısma gelen kutular, küçük bantların üzerinde genişleyen bir alanda ilerledikleri için birbirinden ayrılırlar. Bu sayede kutuların izleme işleminin yapılacağı yere bitişik girmesi engellenmiş olur. Kutuların üretim bandına bitişik olarak girmesi görüntü işleme hesaplamalarında hatalara yol açabilmektedir.

Kutuların birbirinden ayrıldığı bölümden bir sonraki bölüm 2000 mm x 1650 mm alanında birbirine paralel iki kamera ile görüntülenen bölümdür. Bu bölümde kutular ilerlerken yükseklikleri ve ilk köşe koordinatları hesaplanır. Kutular bu bölümde sabit hızla ilerledikleri için bölümün girişinde hesaplanan köşe koordinatlarından yararlanarak kutuların bu bölümden çıkıp izleme yapılan alana girdikleri andaki köşe koordinatları hesaplanır. Bu koordinatlar kutuların görüntü işleme algoritmaları tarafından izlenmesinde ilk değerler olarak kullanılır.

Kutuların yüksekliklerinin ve köşe koordinatlarının hesaplandıkları bölümün hemen sonrasında izleme yapılan bantlar yer almaktadır. İzlemenin yapıldığı kısım, birbirinden bağımsız çalışan küçük bantlardan oluşmaktadır. Bu bantlar bir denetleyiciye bağlıdır. Denetleyici her bir bandın farklı hızlarda dönmesini sağlayabilir. Bantlar farklı hızlarda dönerek kutuların ilerlemesinin yanı sıra dönmelerini de sağlamaktadır. İzlemenin yapıldığı kısımda kutuların x – y eksenlerine paralel konumlara getirilmesi ve tek sırada dizilmeleri amaçlanmaktadır. Bu sebepten, denetleyici bant üzerindeki her kutunun her hangi bir t anında konumunu ve bant üzerinde ki yerleşimini bilmek zorundadır. İzleme işlemi için birbirinden eşit uzaklıkta dört kamera kullanılmıştır. Her bir kameranın görüntüsü kendisinden önce ve sonra gelen kameralar ile kesişmektedir. Bu sayede kutular bir kamera görüntüsünün görüntü alanı dışına çıkarken diğerinin görüntü alanına girerler. Kameraların bu şekilde ayarlanması ile kutuların izlenmesinde kamera geçişlerinde olabilecek olan görüntü kaybı engellenir.

Bu tezde çözülmeye çalışılan problem, izleme bandı üzerinde ilerleyen kutuları görüntüleyen dört kameradan saniyede gelen 30 gri tonlama görüntülerini kullanarak bant üzerindeki kutuların köşe koordinatlarını gerçek zamanda hesaplayan ve denetleyiciye gönderen bir görü sistemi yazılımı gerçekleştirmektir.

Denetleyici bu bilgileri esas alarak kutuları altlarındaki bantların hızlarını ayarlayarak düzenleyecek ve tek sıraya girmelerini sağlayacaktır.

Bütün üretim bandı, görü sistemi ile kalibre edilmiştir. Görü sisteminden elde edilen piksel değerleri üretim bandı üzerinde ki koordinatlara milimetre cinsinden çevrilebilir ve aynı şekilde üretim bandı üzerinde ki herhangi bir noktanın koordinatları görü sisteminde piksel değerleri olarak hesaplanabilmektedir.

Üretim bandı üzerinden geçen kutuların minimum ve maksimum en, boy boyutları sırasıyla 150 x 150 mm ve 1500 x 1500 mm' dir. Kutuların yüksekliği ise en fazla 900 mm olabilmektedir. Aynı zamanda saatte en fazla 6000 kutu üretim bandı üzerinden geçebilmektedir.

1.3 Katkılar

Bu tezde temel izleme algoritması olarak Lucas ve Kanade'nin geliştirdiği [1] sonrasında Tomasi'nin ilerlettiği bir algoritma kullanılmıştır [2]. Başta görüntü kalitesinin ve ışıklandırmanın kötü olması algoritmanın tek başına verimli çalışmasını engellemiştir. Aynı zamanda üretim bandının bazı kısımlarının kutularla çok benzer renk yoğunlukları içermesi ciddi problemlere yol açmıştır. Bu problemlerin çözülmesi için farklı görüntü işleme teknikleri kullanılarak izleme algoritması desteklenmiştir.

Kutular üretim bandı üzerinde izleme yapılacak olan bölüme girdikleri anda ilgili görüntüdeki ilk köşe noktaları Lucas Kanade Tomasi(LKT) algoritmasına girdi olarak verilmiştir. İzleme algoritması bu köşe noktalarını bir sonraki görüntüde bulmakta ve bu noktalar bir sonraki görüntü için tekrar girdi olarak kullanılmaktadır. Yukarıda anlatılan sebeplerden izleme algoritması tek başına yeterli seviyede çalışamadığı gözlenmiştir. Algoritmanın bir noktayı diğer görüntüde bulurken yaptığı küçük değerdeki piksel hataları, çıktıların bir sonraki görüntülerde direk girdi olarak kullanılmasından dolayı gittikçe büyümüş ve yaklaşık on görüntü sonra algoritmanın kutuları izlemede başarısız olduğu gözlemlenmiştir.

Bu hataların sonucunda izlenemeyen köşe noktaları sebebiyle kutuların izlenen üst yüzey yapılarının bozulduğu gözlemlenmiştir. Kutular izleme bölümüne ilk girdiklerinde köşe koordinatları bilindiği için kenar uzunlukları

hesaplanabilmektedir. Bir kutunun her hangi iki köşe koordinatları ve kenar uzunlukları bilindiği için bu kutunun üretim bandı üzerindeki üst yüzünü tekrar oluşturmak mümkündür. Böylece kutuların izlenebilen her hangi iki köşesinden ve kenar uzunluklarından faydalanılarak diğer köşelerin koordinatları hesaplanmıştır. Bu algoritma ile kutuların izleme sırasında oluşabilecek hatalardan dolayı üst yüz yapılarının bozulması engellenmiştir ancak izlemede beklenilen sonuçlar elde edilememiştir

İzleme algoritmasının yaptığı bu hataların düzeltilmesi için izlemenin sonucunun alındığı görüntüde Lucas Kanade [1] özellik algılama metodu kullanılmıştır. Bu metot sayesinde izleme sonucunda elde edilen noktalar özellik algılama algoritması tarafından hesaplanan kendilerine en yakın noktalara ötelenmiştir.

Bu algoritmaların sonucunda hala istenilen verimlilikte sonuçlara ulaşılamadığı için bir doğru uyarlama algoritması kullanılmasına karar verilmiştir. Bu algoritmada kutuların kenarlarının çevresi ile olan renk yoğunluk farkı dikkate alınmıştır. İzleme ve özellik algılama metotlarının sonucunda elde edilen köşe noktaları arasında oluşan doğru parçalarının doğruluğu bu algoritma ile kontrol edilmiş ve kenar doğrularının renk yoğunluk farkından faydalanılarak gerektiğinde bu doğru parçaları olmaları gereken yere kaydırılmıştır.

Bütün bu algoritmalar sonucunda başarılı sonuçlara ulaşılmıştır.

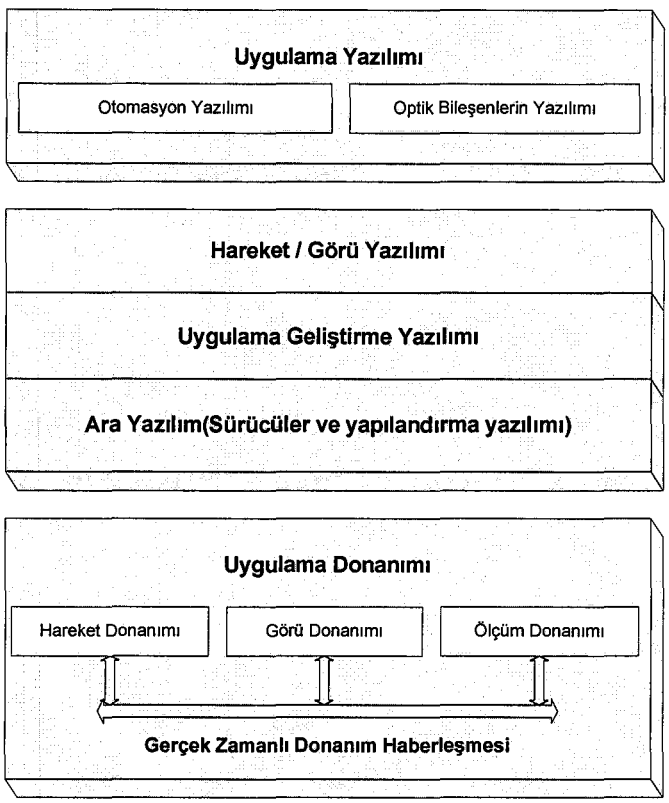
1.4 Tez Organizasyonu

İkinci bölümde, görü sistemleri tasarımı hakkında bilgiler ve bu tezde kullanılan LKT algoriması ile bu algoritmanın alternatifleri açıklanarak neden LKT algoritmasının seçildiğine yer verilmiştir. Üçüncü bölümde ise problem tekrar ele alınmış, kullanılan algoritmalar açıklanmış ve zaman değerleri bildirilmiştir. Dördüncü bölümde ise sonuç yorumları yapılmıştır.

2. ALTYAPI VE İLGİLİ ÇALIŞMALAR

2.1 Görü ve Hareket Sistemleri

Bir hareket ve görü sistemi tasarımı Şekil 2.1'de görüldüğü gibi iki ana bölümde incelenebilir: yazılım ve donanım.



Şekil 2.1 Entegre hareket ve görü sistemi

Donanım kısaca veriyi bilgisayara götürür ya da bilgisayardan alır. Yazılım ise kararlar verir ve donanımdan gelen veriyi yorumlar.

2.1.1 Yazılım

Yazılımın gerçekleştirilmesi, sistemin geliştirilme sürecinde en önemli ve en çok zaman alan kısmıdır. Doğru yazılımın yazılması ya da seçilmesi, sistemin doğrudan başarısını ya da başarısızlığını belirler. Şekil 2.1'de görülen yazılımlardan bir tanesi ara yazılımdır. Ara yazılım, sürücüler ve yapılandırma yazılımlarından oluşur. Sistemin doğru çalıştığını ve programlamaya hazır olduğunu doğrular. Eğer sistemin sürücüleri ve yapılandırma yazılımları birbirlerine benzer ise sistemi çalıştırmak ve programlamak için farklı arabirimlere ihtiyaç duyulmaz.

Uygulama geliştirme yazılımı sistemdeki bir diğer yazılımdır. Bir hareket ve görü sisteminde en ideali yalnız bir yazılım ortamı kullanmaktır. Bu sayede farklı uygulamaların birbirleriyle haberleşme sorunundan kaçınılmış olur. Hareket kontrolü ve görüntü işleme uygulamalarında geleneksel yazılım ortamlarının (Visual Basic, C, C++, .Net, Java vb.) kullanılması yazılımda esnekliği ile hareket ve görü kodlarının yalnız bir ortamda bulunabilme seçeneğini sağlar. Bunların yanı sıra, yazılım geliştirme sürecinde ve hata ayıklamada belirgin zaman kazancı sağlar. Bu tezde yazılım geliştirme ortamı olarak Microsoft Visual C++ 6.0 kullanılmıştır.

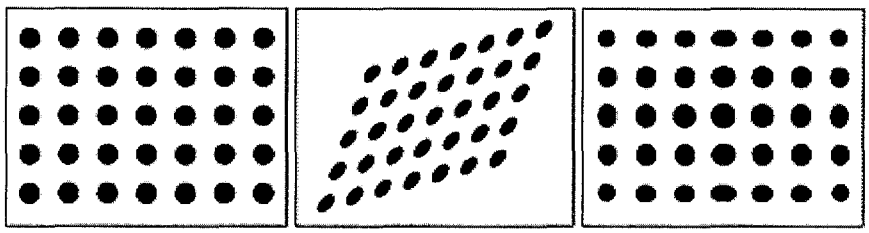
2.1.2 Yazılımın sistemler arası haberleşmesi

Uygun olan uygulama geliştirme yazılımı seçildikten sonra görü sistemindeki yazılımın hareket sistemindeki yazılım ile nasıl haberleşeceği sorunu çözülmelidir. Örneğin bu tezde, görü sistemi için yazılan yazılım, bant üzerindeki kutuların köşe noktalarını denetleyiciye göndermek için hareket sistemi ile hareket sisteminin anlayabileceği bir dilde haberleşmektedir. En basit şekilde görü sisteminde uzaklıklar piksel cinsinden hesaplanır. Hareket sisteminde ise, adımlı motorların adım sayısı, servo motorların kodlayıcı sayısı, üç boyuttaki koordinatlar uzaklık hesabı olarak kullanılabilir. Hareket ve görü sistemlerinin etkin bir şekilde haberleşebilmesi için iki sistem arasında bir kalibrasyon yapılmalıdır. Bu sayede görüntü sisteminin geri döndürdüğü piksel değerleri, adım sayısına, kodlayıcı sayısına ya da üç boyuttaki koordinat sistemine çevrilebilir.

2.1.3 Görü sisteminin kalibrasyonu

Görü ve hareket sistemlerinin kalibrasyonu yapıldıktan sonra kamera izlenen cisimlere göre farklı bir konuma getirilirse kalibrasyon bozulacaktır ve tekrar yapılmalıdır. Kalibrasyonda, izlenecek nesnelere olan uzaklık, kamera açısı ve mercek bozukluğu etkilidir.

Mercek bozukluğuna kameradaki optik hatalar neden olur ve düz doğruların görüntüde eğri olarak görünmesine yol açar. Perspektif bozukluk Şekil 2.2'de görüldüğü gibi, kameranın izlenecek nesneye dik açıyla bakmadığı zaman oluşur. Bu durumda izlenen nesnenin uzakta kalan kısmı görüntüde daha küçük görülür.



Şekil 2.2 Mercek Bozukluğunun gösterimi,

(A) İyi koşullarda ki kalibrasyon şablonu,

(B) Perspektif bozukluk

(C) Mercek bozukluğu

Bu tipteki hataların düzeltilmesi için hem görü hem de hareket sistemi uzamsal olarak kalibre edilmelidir. Bu durum için en çok bilinen ve kullanılan yöntem, bir gerçek dünya birimine kalibrasyondur (milimetre gibi). Bir hassas cetvel ya da iki boyutlu ızgara gibi bilinen bir standart kullanarak görü sisteminin standardı ölçülebilir. Görü yazılımı, kalibrasyon ızgarasında nokta görüntüler kullanarak ve parametre olarak nokta çapı ve uzaklıklarını kullanarak perspektif ve mercek bozukluklarını düzeltir.

2.1.4 Hareket sisteminin kalibrasyonu

Hareket sisteminin kalibrasyonunda da görü sisteminde olduğu gibi bir standarda ihtiyaç duyulur. Doğrusal ya da döner sistemler en çok kullanılan hareket ve görü sistemleridir. Genelde üreticiler, ürün ile birlikte *en küçük hareket miktarı* ismi verilen bir özelliği kalibrasyon için kullanıcıya sunarlar. Bu miktar bir adımdaki hareket miktarını gösterir. Bu miktara sıklıkla *çözünürlük* ismide verilir. Çözünürlüğü, motorun doğruluğu, sistemin mekaniği ve denetleyicinin kapasitesi doğrudan etkiler. Genelde bir görü sistemi, yüksek güçlü mercekler ve ideal ışıklandırma altında 5–10 mikronluk hareketleri gözlemleyebilir. Bir hareket sisteminin görü sistemi ile beraber kalibrasyonu için en basit şekilde aşağıdaki adımlar izlenebilir:

1. Görü ve Hareket Sistemlerinin x, y koordinat eksenlerinin aynı olacak şekilde ayarlanması

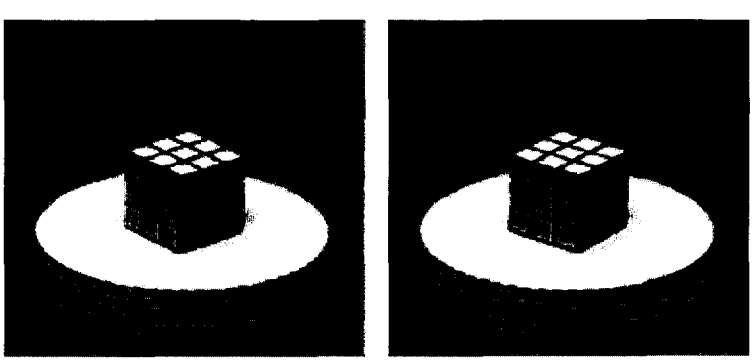
2. Hareket sisteminde, kenar, köşe gibi görü sistemi tarafından kolaylıkla izlenebilecek hareket eden bir özelliğin konumlandırılması
3. Hareket sisteminde belli bir adım/birim sayısınca hareket yapılması
4. Hareketin sonucunda, hareket eden özelliğin tekrar görü sistemi tarafından konumlandırılması
5. Görüntü işlemi yazılımı ile hareket uzaklığın hesaplanması
6. Görü sistemi ile bulunan uzaklık ile hareket sisteminde uygulanan uzaklığın birbirine eşitlenerek bir hareket sabitinin elde edilmesi.

2.2 Optik Akış

Görüntü serilerinin işlenmesinde şüphesiz en temel sorunlardan bir tanesi optik akışın ya da görüntü hızının hesaplanmasıdır. Amaç, üç boyuttaki yüzey noktalarının hızlarının, iki boyuttaki hareket alanı üzerindeki iz düşümlerinin görüntü yoğunluğunun uzamsal örneklerinden faydalanılarak hesaplanmasıdır [13].

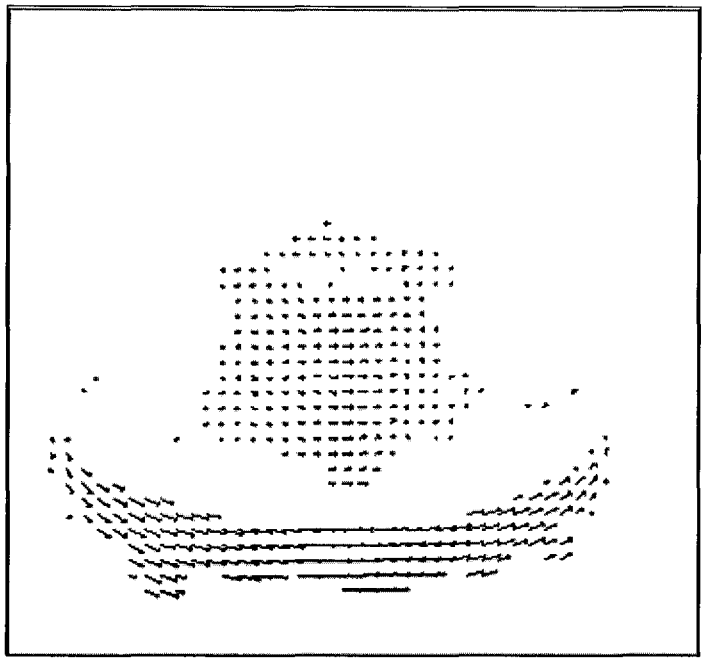
Optik akış, üç boyutlu bir ortamda kameranın ya da kameranın gözlemlediği cismin hareket etmesiyle, görüntü üzerinde oluşan sonuç hareketidir. Başka bir deyişle optik akış, bir görüntü üzerindeki belirgin özelliklerin (köşeler gibi) hareketinin hız ve yönünü belirtir.

Örnek bir optik akış Şekil 2.3'de görüldüğü gibi Rubic küpünün dönmesi ile elde edilmiştir.



Şekil 2.3 Rubic küpünün dönmesi ile elde edilen optik akış [10]

Şekil 2.4'de ise Rubic küpünün dönmesini gösteren iki görüntü karşılaştırılarak akış vektörleri oluşturulmuştur.



Şekil 2.4 Rubic'in küpünün dönmesi ile elde edilen optik akış vektörleri [10]

Optik akış, bilgisayar ile görü alanında birçok alanda kullanılmaktadır. Özellikle, görüntü kesimleme, üç boyutta yeniden oluşturma ve izleme sorunlarında sıklıkla kullanılmaktadır [9]. Genellikle bu konuda çalışan araştırmacılar, çalışmalarını Horn ve Stuck [11] ya da Lucas ve Kanade [1]' nin yaptığı çalışmalar üzerine odaklamışlardır. Bu iki çalışmanın yanı sıra en çok bilinen diğer metotlar Uras [14], Nagel [15], Anandan[16, 17], Singh [18, 19], Heeger [20], Waxman [21] ile Fleet ve Jepson [22, 23] metotlarıdır.

Farklılıklarının yanı sıra bu metotlar kavram olarak üç işlem aşamasında incelenebilir [13]:

1. Alçak geçirme/bant geçirme filtreleri ile süzme ya da yumuşatma yapılarak sinyal yapısının ve sinyal – gürültü oranın hesaplanması
2. Hızın normal bileşenlerinin hesaplanması için zamansal türevler ya da yerel ilgileşim yüzeyleri gibi ölçümlerin hesaplanması
3. Bu hesaplamaların akış alanının yumuşaklığını varsayarak iki boyuttaki akış alanına entegrasyonu

Optik akış teknikleri aşağıda görüldüğü gibi dört ana başlık içerisinde gruplanabilir:

1. Türevsel Teknikler: Horn ve Schunck [11], Lucas ve Kanade [1], Nagel [15], Uras [14]
2. Bölge Tabanlı Eşleşme: Anandan[16, 17], Singh [18, 19]
3. Enerji Tabanlı Metotlar: Heeger [20]
4. Faz Tabanlı Teknikler: Waxman [21], Fleet ve Jepson [22, 23]

2.2.1 Türevsel teknikler

Bir görüntü serisinin optik akışı, her bir görüntüyü kendinden bir sonra gelen görüntü ile ilişkilendiren vektör alanlarıdır. Her vektör alanı, her pikselin görüntüden görüntüye yer değişimini belirtir. Eğer piksellerin yoğunluklarını korudukları varsayılırsa Denklem 2.1'de görülen parlaklık korunum denklemi elde edilir.

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.1}$$

Denklem 2.1'de I görüntü serisi, $[dx, dy]$ ise x, y koordinatlarındaki bir pikselin yer değişim vektörüdür. t ve dt ise görüntü ve görüntünün zamansal yer değişimidir. Parlaklığın korunması ve optik akış ilk olarak Fennema tarafından sunulmuştur [12].

Denkem 2.1'in çözümü için ön görülen en basit çözüm şablon tabanlı arama stratejileridir. Her piksel çevresinde belli boyutlarda bir şablon oluşturularak bu şablona bir sonraki görüntüde en iyi eşleşme aranır. En iyi eşleşme genellikle, ilgileşim, salt farklılık ya da metriklerin toplam karelenmiş farkları ile bulunur. Bu işleme genelde blok eşleşme metodu adı verilir.

Blok eşleşme metotlarının genellikle hesaplama maliyetleri yüksektir ve bu metotlar alt-piksel yer değişimlerini veremezler [38, 39]. Bu yüzden son yirmi yılda geliştirilen metotların çoğu gradyan temelli metotlardır. Gradyan temelli metotlar Denklem 2.1'de görülen denklemin türevsel biçimini Taylor açılımı ile çözer [9]. Yüksek dereceli terimler atıldıktan sonra, bu ifade Denklem 2.2'de görüldüğü gibidir.

$$\frac{\partial I}{\partial x}\mathbf{v}_1 + \frac{\partial I}{\partial y}\mathbf{v}_2 + \frac{\partial I}{\partial t} = 0 \tag{2.2}$$

Denklem 2.2'de görülen denklemde iki bilinmeyen vardır ve bu kötü konumlanmış bir denklemdir. Bir sonuca varabilmek için mutlaka farklı kısıtlar eklenmelidir.

İkinci dereceden türevsel metotlar ise ikinci derece türevleri(I serisinin Hessian'ı) kullanarak iki boyuttaki hızı hesaplar.

$$\begin{bmatrix} I_{xx}(x,t) & I_{yx}(x,t) \\ I_{xy}(x,t) & I_{yy}(x,t) \end{bmatrix} \cdot \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} + \begin{pmatrix} I_{tx}(x,t) \\ I_{ty}(x,t) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{2.3}$$

Denklem 2.2'ye kısıt getirmenin başka bir yolu ise yerel bileşen hızları ile iki boyuttaki uzay ve zaman cinsinden zamanın birleştirilmesidir [18]. Bunu sağlamak için en yaygın kullanılan metot, her komşuluktaki ölçümleri iki boyuttaki hız yerel modeline oturtmaktır. Genellikle karelenmiş hataların minimize edilmesi ile ya da Hough dönüşümü işe gerçekleştirilir [1, 12, 19, 24, 25].

2.2.2 Bölge Tabanlı Eşleşme

Gürültü ve yetersiz sayıda görüntü ve görüntü yakalama işlemlerinde örtüşme gibi sorunlar sebebiyle kusursuz sayısal türev sonuçları elde edilemeyebilir. Bu tarz durumlarda türevsel yaklaşım yerine, bölge tabanlı eşleşme yaklaşımı kullanmak daha uygun olur [17, 26, 27, 28]. Bölge tabanlı eşleşme yaklaşımları **v** hızını **d** = (d_x, d_y) ötelemesi olarak ifade eder ve farklı zamanlarda görüntü bölgelerinde en iyi eşleşmeyi arar. En iyi eşleşmeyi bulmak **d** üzerinden benzerlik ölçütünü maksimize etmek demektir. Başka bir deyişle Denklem 2.4'de görülen karelenmiş farkların toplamında(KFT) olduğu gibi, yer değişim ölçütünün minimize edilmesi demektir [9].

$$KFT_{1,2}(x;d) = \sum_{j=-n}^{n} \sum_{i=-n}^{n} P(i,j) \cdot \left[I_1(x+(i,j)) - I_2(x+\mathbf{d}+(i,j)) \right]^2$$

$$= P(x) * [I_1(x) - I_2(x + \mathbf{d})]^2$$

P, kesikli iki boyutlu pencere fonksiyonudur. $\mathbf{d} = (d_x, d_y)$ ise tam sayı değerleri alır. KFT uzaklık ölçümü ile türevsel teknikler arasında büyük benzerlikler vardır.

2.2.3 Enerji tabanlı metotlar

Enerji tabanlı optik akış teknikleri, hız ayarlı filtrelerin çıktısına bağlıdır [20, 29, 30, 31]. Enerji tabanlı metotlar aynı zamanda hız ayarlı filtrelerin Fourier etki alanında da tasarlandığı için frekans tabanlı metotlar olarakda isimlendirilirler [32, 33].

İki boyuttaki örüntünün Fourier dönüşümü Denklem 2.4'de görüldüğü gibidir.

$$\hat{I}(k, f) = \hat{I}_0(k)\delta(f + v^T k) \tag{2.4}$$

$\hat{I}_0(k)$, $I(x,0)$'ın Fourier dönüşümüdür. $\delta(k)$, Dirac delta fonksiyonu ve f, frekansı göstermektedir [13].

Enerji tabanlı metotlar, ilgileşim tabanlı metotlar ile Lucas ve Kanade'nin gradyan bazlı yaklaşımıyla eşdeğerdir [29, 32, 34].

2.2.4 Faz tabanlı teknikler

Faz tabanlı teknikler, bant geçirme filtrelerinin çıktılarının faz davranışlarının hız olarak tanımlandığı tekniklerdir. Sıfır-geçiş teknikleri aynı zamanda düzey faz-geçiş teknikleri olarak kabul edildikleri için faz tabanlı metotlar sınıfına sokulabilirler [35, 36, 37]. Genelleştirilmiş faz bilgisinin optik akışta kullanılması ilk olarak Fleet ve Jepson tarafından geliştirilmiştir [22, 23].

2.3 Lucas Kanade Tomasi İzleyici Algoritması

2.3.1 Lucas Kanade görüntü tescil algoritması

Bu tezde kullanılan izleme algoritmasında, Lucas ve Kanade'nin 1981 yılında yaptığı bir çalışmadan faydalanılmıştır [1]. Lucas ve Kanade'nin bu yıllarda yaptığı çalışma bilgisayar ile görü alanında en çok kullanılan tekniklerden biri olmuştur. Uygulama alanları genelde optik akış, izleme, katmanlı hareket,

tıbbi görüntü tescili ve yüz tanımadır. Bu algoritmanın üzerine çok sayıda eklentiler yapılmış ve geliştirilmiştir [5]. Bu eklentilerin en önemlisi Tomasi ve Kanade tarafından geliştirilmiştir [2]. Ancak şu anda bu çalışmayı anlatan yayınlanmış tek makale Shi ve Tomasi tarafından yayınlanmıştır [3]. Bu algoritma üzerinde Tomasi, hesaplama simetriğini iki resme dayandıran değişiklikler yapmıştır.

Orijinal Lucas Kanade algoritması, bir $I(\mathrm{x})$ görüntüsünden, $\mathrm{x} = (x, y)^{\mathrm{T}}$ piksel koordinatlarını tutan sütun vektörü olmak üzere bir $T(\mathrm{x})$ şablon görüntüsü oluşturur. Eğer Lucas Kanade algoritması, $t = 1$ zamanından $t = 2$ zamanına optik akışta ya da görüntü izlemede kullanılırsa $T(\mathrm{x})$; 5 x 5, 7 x 7 vs. gibi bir alt-bölge ve $I(\mathrm{x})$ $t = 2$ anındaki görüntü olur.

$\mathrm{E}(\mathrm{x};\ \mathrm{p})$, $\mathrm{p} = (p_1,\ldots p_n)$ parametre vektörleri olacak şekilde parametrize edilmiş eğriltiler olsun. $\mathrm{E}(\mathrm{x};\ \mathrm{p})$ eğriltisi, T şablonunun koordinat çerçevesinden x pikselini alır ve I görüntüsünün koordinat çerçevesindeki alt-piksel konumu ile eşleştirir. Eğer optik akış hesaplanıyorsa Denklem 2.5'deki $\mathbf{e}(\mathrm{x};\ \mathrm{p})$ eğriltisi, $\mathbf{p} = (p_1, p_2)$ vektör parametreleri olmak üzere örnek olarak kullanılabilir [5].

$$\mathbf{e}(\mathrm{x};\ \mathrm{p}) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix} \tag{2.5}$$

Eğer üç boyutta görüntüde daha büyük bir hareket izleniyorsa altı parametre, $\mathbf{p} = (p_1,\ p_2,\ p_3,\ p_4,\ p_5,\ p_6)^{\mathrm{T}}$ olacak şekilde aşağıdaki eğriltiler kullanılabilir [6]:

$$\mathbf{e}(\mathrm{x};\mathrm{p}) = \begin{pmatrix} (1+p_1)\cdot x & + & p_3 \cdot y & + p_5 \\ p_2 \cdot x & + & (1+p_4)\cdot y & + p_6 \end{pmatrix} = \begin{pmatrix} (1+p_1) & p_3 & p_5 \\ p_2 & (1+p_4) & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Lucas Kanade algoritmasında amaç, T şablonu ve I görüntüsü arasındaki karelenmiş toplam hatanın minimize edilmesidir [1].

$$\sum_{x} \left[I(\mathbf{e}(\mathrm{x};\ \mathrm{p})) - T(\mathrm{x}) \right]^2 \tag{2.6}$$

$I(\mathbf{e}(x;\ p))$' nin hesaplanmasında, I görüntüsünün $\mathbf{e}(x;\ p)$ alt-piksel konumlarında interpolasyonuna gerek duyulur. Denklem 2.6' da minimize işlemi p'ye göre yapılır ve toplam $T(x)$, şablon görüntüsü üzerinde bütün x piksellerinin kullanılması ile bulunur. Denklem 2.5'in minimize edilmesi; $\mathbf{e}(x;\ p)$, **p**'ye göre doğrusal olsa bile doğrusal olmayan bir optimizasyon ister. Bunun sebebi ise genellikle $I(x)$'in x'e göre doğrusal olmamasıdır. Sonuçta, $I(x)$ piksel değerleri x piksel koordinatları ile ilişkisizdir. Denklem 2.6'nın optimizasyonu için Lucas Kanade algoritması p'nin mevcut tahmini değerinin bilindiğini varsayar ve yinelemeli olarak Δp parametresinin artışını hesaplar [5].

$$\sum_{x} \left[I(\mathbf{e}(x;\ \mathbf{p} + \Delta\mathbf{p})) - T(x) \right]^2 \tag{2.7}$$

Denklem 2.7, $\Delta\mathbf{p}$'ye göre minimize edilir ve parametreler Denklem 2.8'de görüldüğü gibi güncellenir.

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p} \tag{2.8}$$

Bu iki adım p değerleri yakınsayana kadar devam eder. Yakınsama sınaması Δp vektör normunun belli bir eşik değerinin altına düşmesine kadar devam eder.

$$\|\Delta\mathbf{p}\| \leq \varepsilon$$

Lucas Kanade algoritmasında görüntü tescilinde uzamsal gradyan bilgisi kullanılmıştır. Bu şekilde, arama işlemi en iyi eşleşmeyi sağlayacak konuma yönlendirilmiştir. Lucas ve Kanade, tek boyut ele alındığında $F(x)$ ve $G(x)$ eğrileri arasındaki h yatay eşitsizliğini hesaplamak istemiştir [1].

$$G(x) = F(x+h)$$

Lucas ve Kanade bu sorunun çözümünü $F(x)$'in x komşuluğunda doğrusal yakınlaştırılmasında aramıştır. Özellikle küçük h değerleri için

$$\frac{F(x+h) - F(x)}{h} = \frac{G(x) - F(x)}{h} \tag{2.9}$$

böylece

$$h \approx \frac{G(x) - F(x)}{F'(x)} \tag{2.10}$$

olarak hesaplanabilir [1].

Algoritmanın başarısı bu yaklaştırmanın en uygun şekilde olabilmesi için h'ın yeterince küçük olmasına bağlıdır. Denklem 2.10'da görüldüğü gibi h'a yaklaşma x'e bağlıdır. Farklı h hesaplamalarının bir araya getirilmesi için en doğal yöntem farklı x değerlerindeki h değerlerinin ortalamasını almaktır [1].

$$h \approx \sum_x \frac{G(x) - F(x)}{F'(x)} \Big/ \sum_x 1 \tag{2.11}$$

Bu ortalama, Denklem 2.9'daki yaklaştırmanın $F(x)$ doğrusal olduğu zaman daha iyi olması ve $|F''(x)|$'in büyüdükçe kötü olması durumlarının ele alınması ile iyileştirilebilir. Denklem 2.11'de ki her terimin ortalamaya katkısı $|F''(x)|$ ile ters orantılı olarak hesap edilebilir. Bu durumda ortalama ağırlık, Denklem 2.12'de görüldüğü gibi olur [1].

$$a(x) = \frac{1}{|G'(x) - F'(x)|}$$

$$h = \sum_x \frac{a(x) \cdot [G(x) - F(x)]}{F'(x)} \Big/ \sum_x a(x) \tag{2.12}$$

Bu hesaplamalar sonucunda $F(x)$, Newton – Raphson yinelemesi kullanılarak h kadar taşınabilir [2].

$$h_0 = 0$$

$$h_{k+1} = h_k + \sum_x \frac{a(x) \cdot [G(x) - F(x + h_k)]}{F'(x + h_k)} \Big/ \sum_x a(x)$$

Yukarıda anlatılan türetim iki boyut için iyi bir fikir vermez çünkü iki boyuttaki doğrusal yaklaştırma farklıdır. Ayrıca, Denklem 2.10, $F'(x)$'in 0

olduğu durumlarda tanımsızdır. $F(x + h)$ ve $G(x)$ arasındaki fark ölçümleri Denklem 2.13 – 2.14'de gösterilmiştir.

$$L_1\ norm = \sum_{x \in R} |F(x+h) - G(x)| \tag{2.13}$$

$$L_2\ norm = \left(\sum_{x \in R} [F(x+h) - G(x)]^2 \right)^{1/2} \tag{2.14}$$

$$\text{Normalize edilmiş ilgileşim} = \frac{-\sum_{x \in R} F(x+h) - G(x)}{\left(\sum_{x \in R} F(x+h)^2 \right)^{1/2} \cdot \left(\sum_{x \in R} G(x)^2 \right)^{1/2}} \tag{2.15}$$

Yukarıda bahsedilen her iki sorununda çözülmesi, Denklem 2.9'un Denklem 2.16'da görüldüğü gibi L_2 normunu minimize edecek bir h bulunması için yapılacak doğrusal yaklaştırılma ile çözülebilir.

$$F(x+h) \approx F(x) + h \cdot F'(x) \tag{2.16}$$

$$B = \sum_{x \in R} [F(x+h) - G(x)]^2 \tag{2.17}$$

h'a göre hatayı minimize etmek için aşağıdaki denklem kullanılır.

$$h = \sum_x \frac{a(x) \cdot F'(x) \cdot [G(x) - F(x)]}{\sum_x a(x) \cdot F'(x)}$$

Elde edilen sonuç Denklem 2.12 ile aynıdır ancak ağırlık fonksiyonu $a(x) = F'(x)^2$ olacaktır ve bu sonuç iki ya da daha fazla boyut için kullanılabilir [1]. Bu ifadenin yinelemeli olarak yazılımı Denklem 2.18'de görülmektedir.

$$h_0 = 0$$

$$h_{k+1} = h_k + \sum_x \frac{a(x) \cdot F'(x + h_k) \cdot [G(x) - F(x + h_k)]}{\sum_x a(x) \cdot F'(x + h_k)^2} \tag{2.18}$$

Tek boyuttaki algoritmanın çoklu boyutlar için genelleştirilmesi için ise Denklem 2.17'de görülen hatanın, x ve h n-boyutlu satır vektörleri olmak üzere doğrusal yaklaştırılması gerekmektedir.

$$F(x+h) \approx F(x) + h \cdot \frac{\partial}{\partial x} F'(x) \tag{2.19}$$

Denklem 2.19'da görülen $\partial/\partial x$ sütun vektörü şeklinde, x'e göre gradyan operatörüdür.

$$\frac{\partial}{\partial x} = \left[\frac{\partial}{\partial x_1} \quad \frac{\partial}{\partial x_2} \cdots \frac{\partial}{\partial x_n} \right]^T$$

Bu yaklaştırma kullanılarak B minimize edildiğinde aşağıda görülen sonuçlar elde edilir.

$$\begin{aligned} 0 &= \frac{\partial}{\partial h} B \\ &\approx \frac{\partial}{\partial h} \sum_x \left[F(x) + h \frac{\partial F}{\partial x} - G(x) \right]^2 \\ &= \sum_x \frac{\partial F}{\partial x} \left[F(x) + h \frac{\partial F}{\partial x} - G(x) \right] \end{aligned}$$

$$h = \left[\sum_x \left(\frac{\partial F}{\partial x} \right)^T \cdot [G(x) - F(x)] \right] \cdot \left[\sum_x \left(\frac{\partial F}{\partial x} \right)^T \cdot \left(\frac{\partial F}{\partial x} \right)^T \right]^{-1}$$

Sonuç olarak tek boyutta uygulanan yineleme, ağırlık, yumuşatma ve kaba-ince tekniği başarılı bir şekilde n-boyuttada uygulanabilmektedir. Uygulanan metot ile, geçmiş ve şu andaki görüntülerinin sabit boyutlu özellik pencereleri üzerindeki eşleşme ölçütü, pencereler üzerindeki karelenmiş yoğunluk farkları toplamı cinsinden ifade edilir. Yer değiştirme, bu toplamı minimize eden değer olarak tanımlanır. Küçük hareketler için görüntü yoğunluğunun doğrusallaştırılması Newton-Raphson tipinde bir minimizasyon işlemine tekabül eder [2].

2.3.2 Lucas Kanade Algoritmasının Türetimi

Denklem 2.7'de belirtilen doğrusal olmayan ifade $I(\mathbf{e}(\mathrm{x};\mathbf{p}+\Delta\mathbf{p}))$ üzerinde birinci derecen Taylor açılımı yapılarak doğrusallaştırılabilir.

$$\sum_{x}\left[I(\mathbf{e}(\mathrm{x};\ \mathrm{p}))+\nabla I\frac{\partial \mathbf{e}}{\partial \mathbf{p}}\Delta\mathbf{p}-T(\mathrm{x})\right]^2 \qquad (2.20)$$

Bu ifade de $\nabla I=\left(\frac{\partial I}{\partial x},\frac{\partial I}{\partial y}\right)$ $\mathbf{e}(\mathrm{x};\ \mathbf{p})$'de hesaplanan görüntünün gradyanıdır. ∇I, I'nın koordinat çerçevesinde hesaplandıktan sonra, mevcut E(x; p) eğriltisi hesabından yararlanılarak T'nin koordinat çerçevesinde kullanılır. $\frac{\partial \mathbf{e}}{\partial \mathrm{p}}$, eğriltinin Jacobian'ıdır. Eğer $\mathbf{e}(\mathrm{x};\ \mathbf{p}) = (\mathbf{e}_x(\mathrm{x};\ \mathbf{p}),\ \mathbf{e}_y(\mathrm{x};\ \mathbf{p}))^{\mathrm{T}}$ ise Denklem 2.21'de görülen sonuca ulaşılır.

$$\frac{\partial \mathrm{E}}{\partial \mathrm{p}}=\begin{pmatrix}\frac{\partial \mathbf{e}_x}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{e}_x}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{e}_x}{\partial \mathbf{p}_n}\\ \frac{\partial \mathbf{e}_y}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{e}_y}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{e}_y}{\partial \mathbf{p}_n}\end{pmatrix} \qquad (2.21)$$

Denklem 2.20' nin minimize edilmesi bir küçük kareler problemidir. Bu denklemin $\Delta\mathbf{p}$'ye göre kısmi türevi alındığında, $\nabla I\frac{\partial \mathbf{e}}{\partial \mathbf{p}}$ en hızlı iniş görüntüsü olur;

$$2\sum_{x}\left[\nabla I\frac{\partial \mathbf{e}}{\partial \mathbf{p}}\right]^T\left[I(\mathbf{e}(\mathrm{x};\mathbf{p}))+\nabla I\frac{\partial \mathbf{e}}{\partial \mathbf{p}}\Delta\mathbf{p}-T(\mathrm{x})\right] \qquad (2.22)$$

Denklem 2.22'de görülen ifade 0'a eşitlenip çözüldüğünde Denklem 2.23'de görülen ifade elde edilir.

$$\Delta\mathrm{p}=H^{-1}\sum_{x}\left[\nabla I\frac{\partial \mathbf{e}}{\partial \mathbf{p}}\right]^T\left[T(\mathrm{x})-I(\mathbf{e}(\mathrm{x};\mathbf{p}))\right] \qquad (2.23)$$

Denklem 2.23'de görülen H matrisi, n x n Hessian matrisidir [5].

$$H = \sum_{x} \left[\nabla I \frac{\partial \mathbf{e}}{\partial \mathbf{p}} \right]^{T} \left[\nabla I \frac{\partial \mathbf{e}}{\partial \mathbf{p}} \right] \tag{2.24}$$

Lucas Kanade algoritması [1], yinelemeli olarak Denklem 2.24 ve Denklem 2.8'in uygulanmasıdır. Algoritma aşağıda görüldüğü gibidir;

Yinele:

(1) I ile $\mathbf{e}(x; \mathbf{p})$'den $I(\mathbf{e}(x; \mathbf{p}))$'yi hesapla

(2) $T(x)$ - $I(\mathbf{e}(x; \mathbf{p}))$ hata görüntüsünü hesapla

(3) ∇I ile $\mathbf{e}(x; \mathbf{p})$'yi hesapla

(4) $(x; \mathbf{p})$' de $\frac{\partial \mathbf{e}}{\partial \mathbf{p}}$ Jacobian'ı uygula

(5) $\nabla I \frac{\partial \mathbf{e}}{\partial \mathbf{p}}$, en hızlı iniş görüntülerini hesapla

(6) Denklem 2.24'ü kullanarak Hessian matrisini hesapla

(7) $\sum_{x} \left[\nabla I \frac{\partial \mathbf{e}}{\partial \mathbf{p}} \right]^{T} \left[T(x) - I(\mathbf{e}(x; \mathbf{p})) \right]$ Denklemini hesapla

(8) Denklem 2.23'ü kullanarak $\Delta \mathbf{p}$ hesapla

(9) Parametreleri güncelle; $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$

$\| \Delta \mathbf{p} \| \leq \varepsilon$ Olana kadar [5].

2.3.3 Kanade Tomasi algoritması

Lucas ve Kanade'nin çalışmasında, özellik penceresinin nasıl seçileceğine ve bu pencerelerin bir görüntüden diğerine nasıl izleneceğine değinilmemiştir. Kamera veya izlenecek cisim hareket ettikçe buna bağlı olarakta görüntü yoğunluğu değişecektir. Görüntü serisini ifade edebilmek için üç değişkenli bir fonksiyon yeterli olacaktır. $I(x, y, t)$ fonksiyonunda, x ve y uzay değişkenleri, t ise zamandır ve her üç değişkende kesikli zamanda ifade edilebilen, sınırları olan değişkenlerdir. Bununla beraber, kısa zaman aralıklarıyla elde edilen görüntüler

birbiriyle güçlü bir şekilde ilişkilidir. $I(x, y, t)$ fonksiyonu gelişi güzel değildir ve Denklem 2.25'de görüldüğü gibi ifade edilebilir [2].

$$I(x, y, t+\tau) = I(x-\xi, y-\eta, t) \tag{2.25}$$

$(t+\tau)$ anındaki bir görüntü, t anındaki görüntünün piksellerinin $d = (\xi, \eta)$ kadar yer değiştirmesi ile hesaplanabilir. Ancak, statik bir çevrede ve ışıklandırmada dahi bu ifade bozulabilir. Örneğin görüntünün sınır bölgelerinde noktalar kaybolabilir. Diğer bir sorun ise iki görüntü arasında yalnız bir pikselin çok ayırt edici bir parlaklığa sahip olmaması durumunda izlenememesidir. Pikselin parlaklık değeri diğer görüntüde gürültüden dolayı bozulabilir ya da komşu piksellerle karıştırılabilir. Bu yüzden yalnız bir piksel değil, pikseller penceresi izlenmelidir [2].

Piksel pencerelerinin izlenmesinde iki sorunla karşılaşılabilir. İlki, eğer bu pencereler birinci görüntüden ikinci görüntüye zamanla değişiyorsa aynı pencere ikinci görüntüde nasıl bulunacağıdır. İkincisi ise, pencerenin yer değişimi hesaplanırken farklı hızlar nasıl bir araya getirilerek yalnız bir sonuç vektörü elde edilebileceğidir. İlk sorunun çözümü pencerenin zamanla çok fazla değişmediğini varsaymaktır. Eğer değiştiyse o pencere ile işlem yapılmaz. İkinci sorunun çözümü ise Lucas ve Kanade'nin çalışmasında yatar[1]. Pencere değişimleri basit çevirimler yerine ilgin haritası gibi daha karışık dönüşüm işlemleri olarak düşünülerek farklı hızlar pencerenin farklı noktaları ile ilişkilendirilebilir.

$J(\mathrm{x}) = I(x, y, t+\tau)$ ve $I(\mathrm{x}-\mathrm{d}) = I(x-\tau, y-\eta, t)$ olarak yeniden tanımlanır ve zaman değişkeni atılırsa, yerel görüntü modeli Denklem 2.26'da görüldüğü gibi olur.

$$J(\mathrm{x}) = I(\mathrm{x}-\mathrm{d}) + n(\mathrm{x}) \tag{2.26}$$

Denklem 2.26'da n, hatayı belirtmektedir. Yer değiştirme vektörü d, Denklem 2.27'de görülen ifadeyi p penceresi üzerinde hatayı minimize edecek şekilde seçilmelidir [2].

$$\varepsilon = \int_P [I(\mathrm{x} - \mathrm{d}) - J(\mathrm{x})]^2 a dx \qquad (2.27)$$

Denklem 2.27'de görülen a ağırlık fonksiyonu, Lucas ve Kanade'nin yaptığı çalışmaya uygun olacak şekilde seçilmelidir [1].

Yer değişim vektörü yeterince küçükse, yoğunluk fonksiyonu Taylor fonksiyonu kullanılarak Denklem 2.28'de görüldüğü gibi yaklaştırılabilir.

$$I(\mathrm{x}-\mathrm{d}) = I(\mathrm{x}) - \mathrm{g}\cdot\mathrm{d} \qquad (2.28)$$

Denklem 2.27 tekrar düzenlendiğinde Denklem 2.29 elde edilir.

$$h = I(x)-J(x)$$

$$\varepsilon = \int_P [I(\mathrm{x}) - \mathrm{g}\cdot\mathrm{d} - J(\mathrm{x})]^2 a dx = \int_P (h - \mathrm{g}\cdot\mathrm{d})^2 a dx \qquad (2.29)$$

Denklem 2.29, d yer değiştirmesinin ikinci dereceden bir fonksiyonudur. Sonuç olarak minimizasyon kapalı bir biçimde yapılabilir. Denklem 2.29' da görülen ifade d'ye göre türevlenip sonuç sıfıra eşitlendiğinde Denklem 2.30'daki vektör denklemi bulunur.

$$0 = \int_P (h - \mathrm{g}\cdot\mathrm{d})\mathrm{g} a dA \qquad (2.30)$$

$(\mathrm{g}\cdot\mathrm{d})\mathrm{g} = (\mathrm{g}\mathrm{g}^{\mathrm{T}})\mathrm{d}$ olduğu için ve d'nin P içerisinde bir sabit olduğu kabul edildiğinden Denklem 2.31'e erişilebilir.

$$\left(\int_P \mathrm{g}\mathrm{g}^{\mathrm{T}} a dA \right) d = \int_P h\mathrm{g} a dA \qquad (2.31)$$

Denklem 2.31 iki bilinmeyenli bir denklem sistemidir ve Denklem 2.32'de görüldüğü gibi yazılabilir.

$$K\mathrm{d} = \mathrm{e} \qquad (2.32)$$

Bu denklemdeki katsayı matrisi bir simetrik 2 x 2 matrisidir:

$$K = \int_P \mathrm{g}\mathrm{g}^{\mathrm{T}} pdA$$

Denklem 2.29'da eşitliğin sağ tarafındaki iki boyutlu vektör Denklem 2.33'de görüldüğü gibidir.

$$e = \int_P (I\text{-}j)\mathrm{g}pdA \tag{2.33}$$

Denklem 2.33'de e açıkça iki görüntü arasındaki fark olarak yazılmıştır. Denklem 2.32 izleme için en temel adımdır. Her komşu çerçeve için bir çerçevede gradyanların ve ikinci derece momentlerinin hesaplanması ile H matrisi hesaplanabilir. e vektörü ise yukarıda hesaplanan gradyana göre iki çerçevenin farkı olarak hesaplanabilir. Sonuçta sistemin çözümü d yer değiştirmesidir [2]. Bu durumda Denklem 2.34, $d(\mathrm{k})$ k'ncı yinelemedeki yer değişim olarak yazılabilir [7].

$$d(\mathrm{k}+1) = d(\mathrm{k}) + \mathrm{K}^{-1}e(\mathrm{k}) \text{ ve } d(0) = 0 \tag{2.34}$$

2.3.4 Özellik Algılama Algoritması

İzleme için kullanılacak algoritma ne olursa olsun, bir görüntünün bütün bölgelerinin hareket bilgisini taşımaz. Örneğin düz bir kenar boyunca sadece bu kenara dik olan hareket bileşenleri hesaplanabilir. Bu sebeple izlemede zengin doku içeren bölgeler kullanılmalıdır. Bu sebeple köşeler ya da yüksek uzamsal frekanslı içeriği olan pencereler ya da ikinci dereceden türevleri yüksek olan bölgeler izleme için uygun görülür [4].

Bütün bu tanımlar, izlenebilecek olan özelliklerin takip algoritmasından bağımsız bir şekilde tanımlandığını var saymaktadır. Sonuçta elde edilen özellikler sezgisel olabilir ama izleme algoritmasının bu özelliklerin izlenmesinde ne derece iyi sonuçlar vereceği belli değildir. Bu sebeple Tomasi ve Kanade çalışmalarında, özelliklerin seçimini izleme algoritmasına dayandırırlar. Bu durumda iyi bir pencere, verimli izlenebilecek bir penceredir.

Denklem 2.32'de görülen sistem eğer iyi ve güvenilir ölçümler sunarsa bir pencere görüntüden görüntüye izlenebilir. Sonuç olarak sistemin 2 x 2 katsayı

matrisi olan K'nın görüntü gürültü düzeyinden daha yukarıda olması ve izleme için uygun olması anlamına gelir. Gürültü gereksinimi, K'nın bütün özdeğerlerinin yeterince büyük olmasını tanımlarken iyileştirme gereksinimi bu özdeğerlerin bir kaç derece büyüklük farkı gösteremeyeceğini tanımlar. İki küçük özdeğer, pencere içerisinde benzer yoğunluk olduğu gösterir. Bir büyük ve bir küçük özdeğer, tek yönlü örüntüyü gösterir. İki büyük özdeğer ise köşe ya da tuz-biber dokuları gibi iyi derecede izlenebilecek örüntüleri gösterir.

Pratikte, küçük olan özdeğer, gürültü ölçütünü karşılayabilecek kadar büyükse, K matrisi uygun duruma getirilmiş demektir. Bunun sebebi pencere içerisindeki yoğunluk değişimlerinin kabul edilebilir maksimum piksel değeri ile sınırlanmasıdır. Böylece büyük olan özdeğer, K matrisinin uygunluğunu bozacak seviyenin üzerinde olamaz [2].

Sonuç olarak, K'nın iki özdeğeri λ_1 ve λ_2 ve λ önceden belirlenmiş eşik değeri ise aşağıdaki pencere kabul edilebilir:

$$\min(\lambda_1, \lambda_2) > \lambda$$

Eşik değerinin seçiminde görüntü üzerinde öncelikle fazla yoğunluk değişimi olmayan bir pencere seçilir ve alt sınır olan özdeğer bulunur. Daha sonra köşeler yada yoğun dokulu bölgelerde pencereler seçilerek üst sınır özdeğer hesaplanır. Eşik değeri hesaplanması bu aşamadan sonra çok kritik değildir çünkü her iki değer arasında büyük fark elde edilecektir [3].

Bu çalışmada Kanade Lucas Tomasi algoritması kullanılırken, yerel yapı matrisinden yararlanılmıştır.

$$C = l.\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Öncelikle görüntü yumuşatılmış, sonrasında yoğunluk fonksiyonun türevleri ve l Gaussian filtresi hesaplanmıştır.

Yerel yapı matrisi, simetriktir ve koordinat ekseninde döndürülerek λ_1, λ_2 özdeğerleri cinsinden yazılabilir.

$$C = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Bu durumda; bir köşeyi, küçük olan λ_2'nin yeterince büyük olduğu konumlar gösterir. P pencere boyutu, λ eşik değeri, u görüntü üzerinde bir nokta ve L bir noktalar dizisi olmak üzere, u'nun P x P komşuluğunda C matrisi oluşturulur. Küçük olan λ_2 özdeğeri hesaplanır. Eğer bu değer λ eşik değerinden büyükse ilgili u noktası L listesine eklenir. Aynı komşulukta eklenen noktalar silindikten sonra ilgili dizide o görüntünün özellik noktaları saklanmış olur.

2.3.5 İzlemede Piramit Görüntü Yapısının Gerçekleştirilmesi

Görüntülerin piramit yapısında gerçekleştirilmesi ile çoklu çözünürlük izleme yapılabilir. Bu sayede, görüntüler arasında yoğunluğu fazla olan hareketler izlenebilir.

$I(\mathrm{x}) = (x, y)$ ve $J(\mathrm{x}) = (x, y)$ ardışık iki görüntüyü, $\mathrm{x} = [x, y]^T$, x ve y koordinatlar olmak üzere x pikselin konumunu, n_x ve n_y görüntünün genişliğini ve yüksekliğini belirtecek olursa, sol üst köşe piksel koordinat vektörü $[0\ 0]^T$, sağ alt köşe piksel koordinat vektörü ise $[n_x - 1, n_y - 1]^T$ olur.

u, I görüntüsü üzerinde herhangi bir nokta olarak seçilirse, Lucas Kanade [1] izleme algoritmasının amacı, J görüntüsü üzerinde $\mathrm{v} = \mathrm{u} + \mathrm{d} = [u_x - d_x, u_y - d_y]^T$ koordinatını bulmaktır. Açıklık problemini engellemek için tek bir piksel yerine komşuluk ele alınmalıdır. i_x ve i_y iki tamsayı değer olmak üzere, N fonksiyonunu minimize eden d görüntü hızı Denklem 2.35'de görüldüğü gibidir.

$$N(d) = N(d_x, d_y) = \sum_{x=u_x-i_x}^{u_x+i_x} \sum_{y=u_y-i_y}^{u_y-i_y} (I(x,y) - J(x+d_x, y+d_y))^2 \qquad (2.35)$$

Denklem 2.35'de görüldüğü gibi fonksiyon, görüntü üzerinde $(2i_x + 1)$ x $(2i_y + 1)$ bir komşuluk içerisinde hesaplanmaktadır.

n_x x n_y boyutlarında bir I görüntüsünün piramit yapısı oluşturulacak olursa $I^0 = I$, sıfırıncı seviye görüntü olarak adlandırılır. Bu görüntü en yüksek çözünürlükteki görüntüdür ve ham görüntü olarakda isimlendirilir. Bu seviyedeki görüntünün boyutları $n_x^0 = n_x$ ve $n_y^0 = n_y$ olarak tanımlanır. Görüntünün piramit yapısının gerçekleştirilmesi özyinelemeli bir biçimde tanımlanır. Örneğin I^0'dan I^1, I^1'den de I^2 hesaplanır. M piramit seviyesi; I^{M-1}, M-1 seviyesindeki görüntü;

n_x^{M-1}, n_y^{M-1} M seviyesindeki görüntünün boyutları olarak tanımlanırsa I^{M-1} Denklem 2.36'da görüldüğü gibi tanımlanır [42].

$$I^M(x,y) = \frac{1}{4} I^{M-1}(2x, 2y) +$$

$$\frac{1}{8}\left(I^{M-1}(2x-1, 2y) + I^{M-1}(2x+1, 2y) + I^{M-1}(2x, 2y-1) + I^{M-1}(2x, 2y+1)\right) +$$

$$\frac{1}{16}\left(I^{M-1}(2x-1, 2y-1) + I^{M-1}(2x+1, 2y+1) + I^{M-1}(2x-1, 2y-1) + I^{M-1}(2x+1, 2y+1)\right) \quad (2.36)$$

Denklem 2.36 sadece $0 \le 2x \le n_x^{M-1} - 1$ ve $0 \le 2y \le n_y^{M-1} - 1$ için tanımlıdır. Bu durumda n_x ve n_y, Denklem 2.37, 2.38' de görülen denklemleri sağlayan en büyük tamsayılardır.

$$n_x^M \le \frac{n_x^{M-1} + 1}{2} \quad (2.37)$$

$$n_y^M \le \frac{n_y^{M-1} + 1}{2} \quad (2.38)$$

Denklem 2..36-2.38 özyinelemeli olarak kullanılarak görüntülerin piramit yapıları gerçekleştirilmiştir. Görüntülerin piramit yapıları kullanıldığında Denklem 2.35 yeniden düzenlenmelidir. Görüntülerin piramit yapılarının kullanıldığı izleme algoritmasında öncelikle en derin piramit seviyesinde(M_m) optik akış hesaplanır. Daha sonra bu hesaplamanın sonucu bir üstteki seviyeye (M_m - 1) dağıtılır. Bir üst seviyeye gönderilen bu sonuç aslında (M_m - 1) seviyesinde tahmin edilen piksel yer değişimidir. Bu bilgilerle (M_m - 1) seviyesindeki optik akış daha kusursuz olarak hesaplanır ve bir üstteki seviyeye gönderilir(M_m - 2). Bu özyineleme sıfırıncı seviyedeki görüntüye kadar devam eder. Denklem 2.35'in yeniden düzenlenmesiyle elde edilen Denklem 2.39'da görülen yeni ifade de $g^M = [g_x^M \; g_y^M]^T$, M seviyesinde elde edilen tahmini optik akıştır.

$$\varepsilon^M(d^M)=\varepsilon^M(d_x^M,d_y^M)=\sum_{x=u_x^M-i_x}^{u_x^M+i_x}\sum_{y=u_y^M-i_y}^{u_y^M-i_y}(I^M(x,y)-J^M(x+g_x^M+d_x^M,y+g_y^M+d_y^M))^2 \qquad (2.39)$$

Standart Lucas Kanade [1] yinelemeli algoritması bu ifadeye uygulandığında her M seviyesinde aynı işlemler yapılacağı için M üst simgesi düşürülebilir;

$$\frac{\partial\varepsilon(d)}{\partial d}=\sum_{x=u_x-i_x}^{u_x+i_x}\sum_{y=u_y-i_y}^{u_y-i_y}(I(x,y)-J(x+d_x,y+d_y))\cdot\begin{bmatrix}\frac{\partial J}{\partial x} & \frac{\partial J}{\partial y}\end{bmatrix}$$

Piramit yapıdan dolayı $J(x+d_x,y+d_y)$ yerine kendisinin birinci dereceden Taylor açılımı yazılabilir.

$$\frac{\partial\varepsilon(d)}{\partial d}\approx -2\sum_{x=u_x-i_x}^{u_x+i_x}\sum_{y=u_y-i_y}^{u_y-i_y}\left(I(x,y)-J(x,y)-\begin{bmatrix}\frac{\partial J}{\partial x} & \frac{\partial J}{\partial y}\end{bmatrix}d\right)\cdot\begin{bmatrix}\frac{\partial J}{\partial x} & \frac{\partial J}{\partial y}\end{bmatrix}$$

$I(x, y) - J(x, y)$ görüntünün $[x\ y]^T$ noktasında türevi demektir. Bu bilgiden yaralanarak gradyan vektörü yazılır.

$$\nabla I=\begin{bmatrix}I_x\\I_y\end{bmatrix}=\begin{bmatrix}\frac{\partial J}{\partial x} & \frac{\partial J}{\partial y}\end{bmatrix}$$

I_x ve I_y ikinci görüntüden bağımsız olarak hesaplabilir.

$$I_x(x,y)=\frac{I(x+1,y)-I(x-1,y)}{2}$$

$$I_y(x,y)=\frac{I(x,y+1)-I(x,y-1)}{2}$$

Görüntünün türev denklemleri yazıldığında aşağıdaki ifadeye ulaşılır.

$$\frac{1}{2}\frac{\partial\varepsilon(d)}{\partial d}\approx -2\sum_{x=u_x-i_x}^{u_x+i_x}\sum_{y=u_y-i_y}^{u_y-i_y}\left(\begin{bmatrix}I_x^{\,2} & I_xI_y\\I_xI_y & I_y^{\,2}\end{bmatrix}d-\begin{bmatrix}\delta I I_x\\\delta I I_y\end{bmatrix}\right)$$

Bu ifadeden yaralanılarak Denklem 2.32'deki standart Lucas Kanade [1] optik akış denklemine ulaşılabilir. Böylece, görüntülerin piramit yapıları gerçekleştirilerek Lucas ve Kanade [1]'nin optik akış denklemine ulaşılmıştır.

$$\mathrm{d} = K^{-1}\mathrm{e}$$

k yineleme indeksi olmak üzere yinelemeli Lucas Kanade [1] algoritması

$$\mathrm{d}^{-\mathrm{k}} = K^{-1}\mathrm{e}_{\mathrm{k}}$$

şeklinde ifade edilir. Görüntülerde piramit yapısının gerçeklenmesi ile k kez yinelemenin izleme için yeterli olduğu kabul edilirse bulunan optik akış Denklem 2.39'da görüldüğü gibidir.

$$\mathbf{d}^{L} = \sum_{k=1}^{K} \mathbf{d}^{-k}$$

2.3.6 Alt-piksel hesaplamaları

Bu tezde yapılan çalışmada, birçok hesaplamanın alt-piksel seviyesinde yapılması gerektiği açıktır. Alt-piksel seviyesinde hesap yapmak ondalıklı piksel değerleri için görüntü parlaklığına ulaşabilmek demektir. Alt-piksel seviyesinde hesaplamalar yapabilmek için bu tezde çift doğrusal aradeğerleme tekniği kullanılmıştır. Her hangi bir M piramit seviyesindeki $I^M(x, y)$ görüntü değerini gösteren x ve y koordinatlarının tam sayı olmadıkları varsayılırsa, x ve y, x_0 ve y_0 tam sayı kısımları, σ_x ve σ_y ondalıklı kısımları(0 ile 1 arasında) ile ifade edilir:

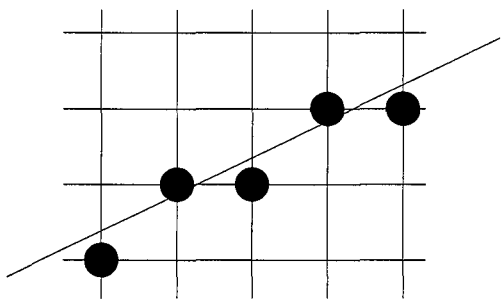
$$x = x_0 + \sigma_x$$
$$y = y_0 + \sigma_y$$

Bu durumda $I^M(x, y)$'nin hesaplanması için çift doğrusal aradeğerleme tekniği orijinal görüntü parlaklık değerlerinde Denklem 2.39'da görüldüğü gibi kullanılır.

$$I^M(x,y) = (1-\sigma_x)(1-\sigma_y)I^M(x_0,y_0) + \sigma_x(1-\sigma_y)I^M(x_0+1,y_0) +$$
$$(1-\sigma_x)\sigma_y I^M(x_0,y_0+1) + \sigma_x\sigma_y I^M(x_0+1,y_0+1)$$

2.4 Bresenham Doğru Uyarlama Algoritması

Doğru uyarlama algoritmaları, ideale yakın ya da ideal doğruların en yakınında uzanan piksellerin 2 boyutta hesaplanabilmesi için geliştirilen algoritmalardır. Bir piksel kalınlığında bir yaklaştırma ele alındığında, eğimin -1 ile 1 arası olması durumunda her sütunda bir piksel kullanılmalıdır. Bu aralığın dışındaki eğim değerlerinde ise her satırda bir piksel kullanılmalıdır.



Şekil 2.5 İdeal bir doğrunun uyarlanması

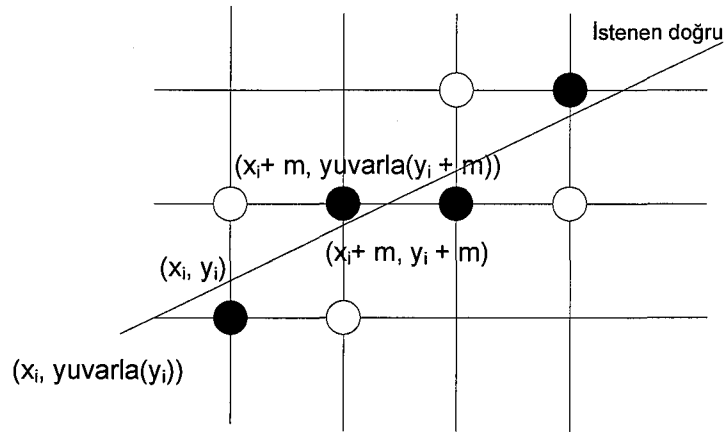
Bir doğrunun uyarlanmasında kullanılabilecek olan en basit algoritma, eğimin $\Delta y/\Delta x$ olarak hesaplanması ve en soldaki noktadan itibaren x'in birer birer artırılmasıdır. Bu sayede her x_i için yi = $mx_i + B$ hesaplanır. y_i ise 0.5 ile toplanarak yuvarlanır. Bu işlemler ondalıklı sayılarla ile çarpma toplama ve yuvarlama işlemleri yapıldığı için verimli değildir. Bu durumda çarpma işlemi aşağıdaki gibi elenebilir.

$$y_{i+1} = mx_{i+1} + B = m(x_i + \Delta x) + B = y_i + m\Delta x \qquad (2.40)$$

Denklem 2.40'da eğer $\Delta x = 1$ ise $y_{i+1} = y_i + m$ olur. Bu durumda x'deki bir birim değişiklik y'de m kadar değişikliğe yol açar. Eğer $x_{i+1} = x_i + 1$ ise $y_{i+1} = y_i + m$ olacaktır. Bu ifade ile x_i ile y_i değerleri bir önceki değerleri cinsiden ifade edilmiştir. Artımlı tanımlanan bu algoritma ile hesaplamalar bir önceki hesaplamalar temel alınarak yapılır.

Şekil 2.6' da görüldüğü gibi doğrunun sol uç noktası başlangıç noktası olarak seçilir ve artımsal algoritmaya ilk değer olarak (x_0,y_0) değeri verilir.

Eğer $|m| > 1$ ise x'deki 1 birimlik değişim y'de 1'den büyük bir değişime yol açar. Bu durumda x ve y'nin üstlendikleri roller değişerek y'ye birim adım uygulanır. x ise $\Delta x = \Delta y / m = 1 / m$ kadar artırılır.

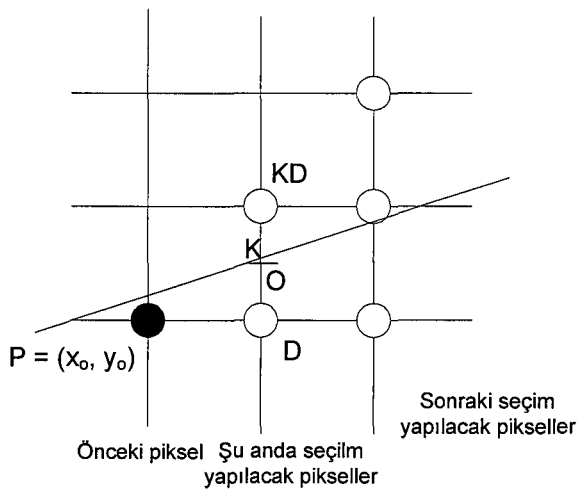


Şekil 2.6 Artımlı tanımlanan algoritma

Bu algoritmada, yuvarlama işlemleri hesaplama zamanında fazla yer tutmaktadır. y ve m değişkenleri ise eğim ondalıklı olacağı için ondalıklı sonuçlar vermektedir. Bu problemin çözümü için Bresenham [43] bir algoritma geliştirmiştir. Bu algoritma ile (x_{i+1}, y_{i+1}) değerleri bir önceki (x_i, y_i) değerlerinden sadece tam sayı aritmetiği kullanılarak hesaplanabilir. Bu tezde Bresenham [43] ve orta nokta tekniği adı verilen, ilk olarak Pitteway [44] tarafından yayınlanan daha sonra Van Aken [45] tarafından geliştirilen bir teknik kullanılmıştır. Bresenham çalışmasında tam sayı aritmetiği kullanılarak yapılan doğru uyarlama algoritmasının gerçek doğrulara en iyi uyan yaklaştırmaların hatanın uzaklık cinsinden minimize edilmesiyle bulunduğunu ortaya koymuştur [46].

Şekil 2.7'de bir önce seçilen piksel siyah daire olarak ve içlerinden birisinin seçileceği iki piksel ise beyaz daire olarak gösterilmiştir. (x_o, y_o) 'de ki P pikselinin henüz seçildiği varsayılırsa sonraki seçim bir birim sağdaki piksel(D) ya da bir birim sağdaki pikselin üstündeki (KD) pikseller arasından yapılacaktır. K noktası ideal doğrunun koordinat sistemi ile kesişimi olarak kabul edilirse $x = x_o + 1$ olur. Bresnham'ın formülasyonuna göre D ve KD piksellerinin K noktasına olan uzaklığı hesaplanır ve farkların işareti hangi pikselin seçileceğinde kullanılır. Bu şekilde en küçük fark hesaplanarak en iyi yaklaşım yapılmış olur. Eğer K orta

noktası ideal doğrunun üzerinde ise D pikseli, altında ise KD pikselinin seçileceği açıkça gözlemlenmektedir. İdeal doğru, D ve KD piksellerinin arasından geçebileceği gibi iki noktayıda altında ya da üstünde bırakacak şekilde de geçebilir. Her iki durumda da algoritma orta noktaya en yakın olan pikseli bulmaktadır. Bu algoritma ile hata ise her zaman 0.5 değerinden küçük olmaktadır.



Şekil 2.7 Orta nokta algoritması

Algoritma Şekil 2.7'de görülen KD pikselini bir sonraki piksel olarak seçer. Bu işlemden sonra orta noktanın doğrunun hangi tarafında kaldığı hesaplanır. Doğru, a, b ve c sabitleri cinsinden $F(x, y) = ax + by + c = 0$ olarak ifade edilir. $dy = y_1 - y_0$ ve $dx = x_1 - x_0$ olarak ifade edilirse eğim aşağıdaki gibi olur.

$$y = \frac{dy}{dx}x + B$$

Bu durumda $F(x, y) = dy.x - dx.y + B.dx = 0$ olarak yazılır. Aynı zamanda $a = dy$, $b = -dx$ ve $c = B$'dir.

Doğrunun üzerindeki noktalar için $F(x, y)$, 0'a eşittir. Doğrunun altındaki noktalar için $F(x, y)$ fonksiyonu negatif, üzerindeki noktalar için ise pozitif olur. Orta nokta algoritmasını uygulamak için

$$F(O) = F\left(x_p + 1, y_p + \frac{1}{2}\right)$$

hesaplanır ve sonucun işareti dikkate alınır. Piksel seçiminde verilecek olan karar, fonksiyonun $F\left(x_p + 1, y_p + \frac{1}{2}\right)$ değerine bağlı olacağı için d = $F\left(x_p + 1, y_p + \frac{1}{2}\right)$ şeklinde bir karar değişkeni atanır.

Tanım olarak, $d = a(x_p + 1) + b(y_p + ½) + c$ 'dir. Eğer d değeri 0'dan küçük ise KD pikseli, d değeri 0'dan büyük ise D pikseli ve eğer d değeri 0'a eşitse herhangi birisi seçilebilir. Şekil 2.13'deki örnekte D pikseli seçilir.

Bu noktadan sonra, O noktasının ve d değişkeninin bir sonraki seçim için alacağı değerler hesaplanmalıdır. Bu hesaplar bir önceki yapılan D ve KD piksellerinin seçiminde bağlı olarak değişir. Eğer D pikseli seçildi ise O noktası x yönünde bir artırılır. Böylece:

$$d_{yeni} = F\left(x_p + 2, y_p + \frac{1}{2}\right) = a(x_p + 2) + b\left(y_p + \frac{1}{2}\right) + c$$

$$d_{eski} = a(x_p + 1) + b\left(y_p + \frac{1}{2}\right) + c$$

olacaktır.

Artış farkının hesaplanması için d_{yeni}'den d_{eski} çıkarılır.

$$d_{\text{yeni}} - d_{\text{eski}} = a$$

Bu ifadeden yararlanarak bir sonraki karar değişkenin değeri ΔD'nin eklenmesiyle $F(O)$'nun hesaplanmasına gerek kalmadan hesaplanabilir.

Eğer KD seçildiyse; O, x ve y yönlerinde bir artırılır. Bu durumda

$$d_{yeni} = F\left(x_p + 2, y_p + \frac{3}{2}\right) = a(x_p + 2) + b\left(y_p + \frac{3}{2}\right) + c$$

d_{yeni}'den d_{eski} çıkarıldığında

$$d_{yeni} - d_{eski} = a + b$$

olarak hesaplanır. KD seçildikten sonra d'ye ΔKD eklenir.

Sonuç olarak, algoritma her adımda bir önceki iterasyonda hesaplanan karar değişkenine göre iki pikselden birini seçmektedir. Daha sonra karar değişkenini yapılan seçime göre ΔD ya da ΔKD ekleyerek günceller.

Başlangıç noktası (x_0,y_0) olduğuna göre d'nin ilk değeri D ve KD arasında bir seçim yapılarak doğrudan hesaplanabilir. İlk orta nokta $\left(x_0+1, y_0+\frac{1}{2}\right)$'dir. Bu durumda

$$F\left(x_0+1, y_0+\frac{1}{2}\right) = a(x_0+2) + b\left(y_0+\frac{1}{2}\right) + c$$

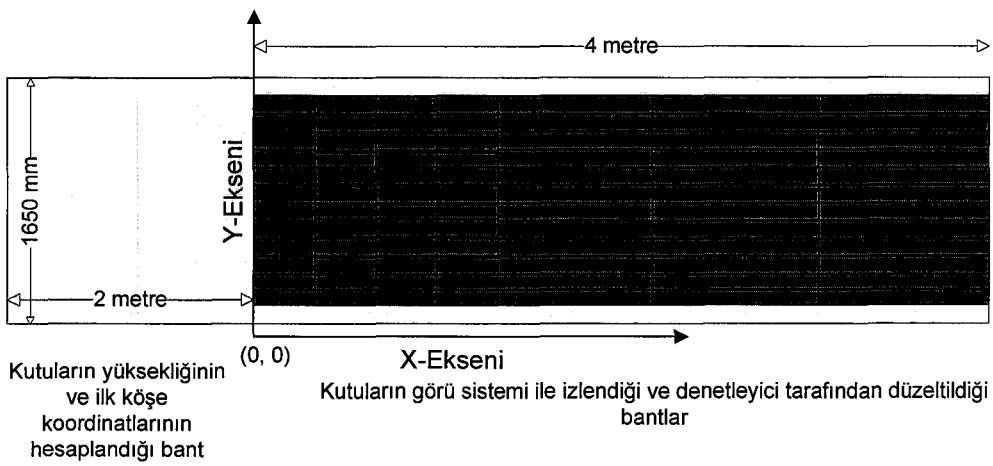
$$= ax_o + by_0 + c + \frac{b}{2}$$

$$= F(x_o, y_0) + a + \frac{b}{2}$$

Lakin, (x_0,y_0) doğru üzerinde bir nokta olduğu için $F(x_0,y_0)$, 0'a eşit olur. Bu durumda başlangıç d değeri $a + b/2 = dy - dx/2$ olur. Başlangıç d değerini kullanarak ikinci piksel seçilir ve algoritma bu şekilde devam eder. Bu şekilde ideal bir doğruyu belirten pikseller hesaplanmış olur.

3. ÇÖZÜMLER ve UYGULAMALAR

Bu tezde bir otomasyon sürecinde kullanılan bir üretim bandının başlangıç kısmının üzerinde düzensiz olarak yerleşik durumda ilerleyen kutuların üretim bandının çıkış kısmında tek sırada düzgün olarak çıkmasını sağlayacak bir gerçek zamanlı sistemin, görü sistemi kısmı tasarlanmıştır.



Şekil 3.1 Üretim bandı ve koordinat ekseni

İzleme bandı üzerinde ilerleyen kutuları görüntüleyen dört kameradan saniyede gelen 30 gri tonlama görüntülerini kullanarak bant üzerindeki kutuların köşe koordinatlarını gerçek zamanda hesaplayan ve denetleyiciye gönderen bir izleme yazılımı gerçekleştirilmiştir. Denetleyici bu bilgileri esas alarak kutuları altlarındaki bantların hızlarını ayarlayarak düzenleyecek ve tek sıraya girmelerini sağlayacaktır.

Bütün üretim bandı, görü sistemi ile kalibre edilmiştir. Görü sisteminden elde edilen piksel değerleri üretim bandı üzerinde ki koordinatlara mm cinsinden çevrilebilir ve aynı şekilde üretim bandı üzerinde ki herhangi bir noktanın koordinatları görü sisteminde piksel değerleri olarak hesaplanabilmektedir.

Bu tezde; Şekil 3.1'de görüldüğü üzere kutular iki metrelik banda geldiklerinde en, boy ve yüksekliklerinin hesaplandığı kabul edilmiştir. Bir kutu için izleme işlemi kutunun Şekil 3.1'de görülen koordinat sisteminde $x = 0$ noktasına gelmesi ile başlar. Bu noktadan önce, kutuların boyutlarının hesaplandığı bantta kutular sabit hızla ilerledikleri için bu bant izleme algoritmasına dahil edilmemiştir.

Bu tezde tasarlanan izleme algoritması, bantta izlemenin yapıldığı dört metrelik kısımda her kutunun dört köşe noktasının koordinatlarını hesaplayıp denetleyiciye gönderecektir.

3.1 Temel İzleme Algoritması

Kutular, üretim bandında izleme yapılacak olan alana gelmeden önce boyutları ve dolayısıyla izleme bandına girdiklerindeki andaki köşe koordinatları ve yüksekliği bilinmektedir. Sonuç olarak herhangi bir kutunun t anındaki köşe noktalarının koordinatları bilinmektedir. Saniyede 30 görüntü geldiği için kullanılan izleme algoritması ile t anında köşe noktalarının koordinatları bilinen kutunun t + 33ms'de, yani yeni görüntüdeki, köşe noktalarının koordinatları hesaplanmaya çalışılmıştır.

Bu tezde kullanılan temel izleme algoritmasında aşağıdaki adımlar uygulanmıştır:

1. Kutunun köşelerinin üç boyuttaki koordinatlarının kalibrasyondan yararlanılarak iki boyuttaki görüntü piksel değerlerine çevrilmesi

2. Bu köşe noktalarının LKT algoritmasına bir sonraki görüntüde izlenmek üzere girdi olarak kullanılması

3. LKT algoritmasından döndürülen çıktıların yani yeni köşe piksel koordinatlarının kalibrasyondan faydalanılarak üç boyuttaki koordinatlara çevrilmesi ve LKT algoritmasının izleyemediği köşe koordinatlarının, izleyebildiklerinden faydalanılarak tekrar hesaplanması ile kutu üst yüzey yapısının korunması ve LKT yaptığı piksel hatalarının kutu üst yüzey yapısının (en, boy) daha önce hesaplanan bilgilerinden faydalanılarak düzeltilmesi.

LKT algoritmasının bu çalışmada kullanılmasının sebebi, Bölüm 2.3.7 deneyler bölümünde görüldüğü gibi köşelerin bulunmasında oldukça başarılı olmasında ve Bölüm 2.2'de yapılan karşılaştırmalardaki başarısında yatar. Tanım itibariyle köşeler, x ve y yönlerindeki kısmi türevlerin yüksek olduğu özellik noktalarıdır. Bu algoritma günümüzde en sık kullanılan özellik bulma algoritmalarından birisidir çünkü göreceli olarak basit, verimli ve güvenilirdir.

3.1.1 LKT İzleme Algoritması

Bu tezde köşe noktalarını izlemek için kullanılan LKT algoritması, Newton Raphson yöntemi ile 2 x 2 gradyan matrisindeki minimum özdeğerleri konumlandırır ve iki pencere arasındaki farkı minimize eder. İki görüntü arasındaki büyük değişimleri(kutuların dönmesi gibi) izleyebilmek için çoklu çözünürlük izleme algoritması kullanılmıştır. Görüntüler algoritmaya girdi olarak ardışık bir biçimde verildiği için ardışık olmayan görüntülerde izleme olanağı sağlayan affine hesaplamalar kullanılmamıştır.

Kutuların yüksekliklerinin ve ilk köşe noktalarının bandın izleme yapılacak olan alandan bir önceki alanda hesaplandığı varsayılmıştır ve bu değerler bu tezde kullanılan LKT algoritmasının ilk değerleri olarak kullanılırlar. Kutular üretim bandında izleme yapılacak olan kısma geldiklerinde ilk olarak izlenecek olan köşelerin üç boyuttaki koordinatları kalibrasyondan yararlanılarak iki boyuttaki görüntü piksel değerlerine çevrilir. LKT algoritmasına mevcut olan görüntü, bir önceki görüntü ve bir önceki görüntüdeki kutuların köşe noktaları verilir. Algoritma mevcut olan görüntüde bu köşeleri izlediği takdirde yeni köşe koordinatlarını geri döndürür.

İlk olarak LKT algoritmasına görüntüdeki takip edilecek köşe noktaları verilerek ilk değerlendirme yapılır. Öncelikle bir izleme içeriği oluşturulur. Bu içerikte algoritmada kullanılacak olan parametreler ve sabitler tanımlanır. Bu parametreler:

1. Seçilen özellikler arasındaki minimum uzaklık.
2. Pencere boyutları.
3. Yumuşatma yapılıp yapılmayacağı ve yapılacaksa sigma değerleri.
4. Görüntünün x ve y gradyanlarının hesaplanmasında kullanılacak olan gradyan sigma değeri.
5. Belli bir özdeğerin üzerindeki özelliklerin seçilmesi ya da izlenmesi isteniyorsa bu özdeğer değeri.
6. Bir köşe noktasının takip edilemediğinin geri döndürüleceği durumda kabul edilebilen minimum belirteç değeri.

7. En küçük yer değişim değeri; yinelemeli olarak çalışan izleyicinin bir noktayı izlediğinde başarılı bir sonuç geri döndürerek durması için gereklidir.
8. En büyük yineleme değeri; bu değer aşıldığında ilgili noktanın izlenemediğine karar verilir.
9. İki pencere arasındaki ortalama piksel yoğunluk değerlerinin farkı; Bu değer aşıldığında ilgili noktanın izlenemediğine karar verilir.
10. Görüntünün sınır koordinatları
11. Piramit düzey sayısı. Çoklu çözünürlükte izleme için görüntülerin piramit gösterimlerinin hesaplanmasında kullanılır.
12. Komşu piramit düzeyleri arasındaki alt-örnekleme miktarı
13. Görüntülerin piramit biçimleri

İzleme içeriği başarılı bir şekilde oluşturulduktan sonra bir sonraki görüntüde izlenecek olan köşe noktalarını içeren bir özellik listesi oluşturur. Bu özellik listesi köşe noktalarının görüntü üzerindeki x ve y koordinatlarını ile özdeğerlerini içerir.

Özellik listesi, bu listenin oluşturulduğu görüntü ve izleme içeriği kullanılarak algoritma, özellik listesini izlemeye hazır hale getirilir. Bu aşamada özellik listesindeki noktaların özdeğerleri hesaplanarak özellik listesine yazılır. Köşe noktalarının özdeğerlerinin hesaplanması için öncelikle görüntü bir gaussian filtresi ile yumuşatılır. Kullanılan filtredeki sigma değeri izleme içeriğinde belirtilen sigma değerinin pencere boyutu ile çarpılması ile hesaplanır.

Yumuşatma işleminden sonra görüntünün x ve y eksenlerinde gradyanları ayrı ayrı hesaplanır. Gradyanların hesaplanması için görüntü izleme içeriğinde belirtilen gradyan sigmasının türevi ile evrişim yapılır. Hesaplanan gradyanlardan yararlanılarak izlenecek olan her bir noktayı çevreleyen pencere içerisindeki x, y ve x - y gradyan toplamları bulunur. Bu toplamlar ile aşağıdaki 2 x 2 gradyan matrisi oluşturulur.

$$G = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

Bu matrisin en küçük özdeğeri;

$$\min \lambda = \frac{I_{xx} + I_{yy} - \sqrt{(I_{xx} - I_{yy})^2 + 4 \cdot I_{xy}^{\ 2}}}{2}$$

şeklinde hesaplanır. İzlenecek olan her köşe noktası için bu hesaplamalar yapılarak minimum özdeğerleri bulunur. Bu özdeğerler ile özellik listesi güncellenir.

İzleme içeriği ve özellik listesinin oluşturulması ile ilk değerlendirme safhası biter. Bu safhadan sonra, LKT algoritmasına özellik listesinin oluşturulduğu görüntü, bu listedeki özelliklerin izleneceği görüntü, özellik listesi ve izleme içeriği gönderilir.

Her iki görüntüde bir gaussian filtresi ile evrişim yapılarak yumuşatılır. Kullanılan filtredeki sigma değeri izleme içeriğinde belirtilen sigma değerinin pencere boyutu ile çarpılması ile hesaplanır. Yumuşatma işlemlerinden sonra görüntünün çoklu çözünürlük piramidi hesaplanır. Piramit seviye sayısı ve sigma değeri, izleme içeriğinde belirtildiği gibidir. Her seviye için, izleme içeriğinde belirtildiği miktarda pikselin alt-örneklemesi yapılır. Piramitlerin her seviyesinde gradyanlar, görüntü izleme içeriğinde belirtilen gradyan sigmasının türevi ile evrişim yapılarak hesaplanır.

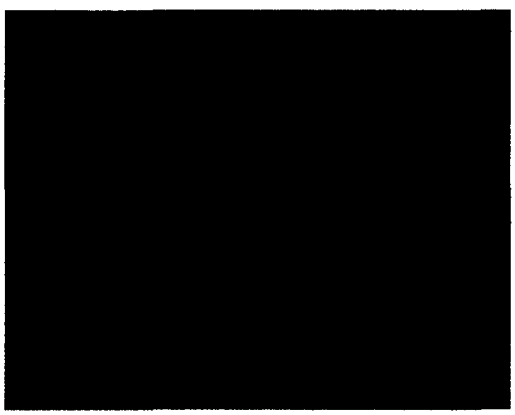
Özellik listesindeki izlenebilen her özellik, en kötü çözünürlükten en iyi çözünürlüğe kadar izlenir. Her çözünürlük, bir sonraki çözünürlük için başlangıç değeridir. Her çözünürlükte izleme, iki görüntüden alınan pencereler arasındaki yoğunluğun Newton-Raphson yinelemesi ile minimize edilmesi yoluyla hesaplanır.

Yinelemenin durması için beş farklı koşul vardır. Bunlardan sadece birincisi, özelliğin izlenebildiği durumda gerçekleşir. Bu koşullar:

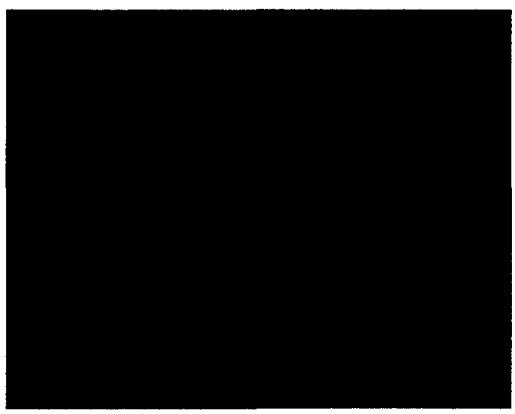
1. Özelliğin, izleme içeriğinde belirtilen minimum yer değiştirme miktarından daha az yer değiştirmiş olması
2. Gradyan matrisinin belirteç değerinin izleme içeriğinde belirtilen minimum belirteç değerinden daha küçük olması

3. Yineleme sayısının, izleme içeriğinde belirtilen yineleme sayısını aşmış olması
4. Özelliğin görüntü sınırlarında bulunması
5. İki pencere arasındaki ortalama piksel yoğunluk değerlerinin farkının izleme içeriğinde belirtilen değerden fazla olması.

Bir özellik eğer birden fazla kamerada birden görüntüleniyorsa, görüntülendiği bütün kameralarda izleme işlemleri yapılır ve her bir kameradan gelen sonuç ayrı olarak değerlendirilir. Birden fazla kamerada görüntülenen özellik için en iyi özdeğer hangi kamerada bulunduysa o kameradan alınan yeni piksel değerleri özelliğin izlendikten sonraki koordinatları olarak kullanılır. Başarılı olarak elde edilen iki boyuttaki koordinatlar kalibrasyon sayesinde tekrar üç boyuttaki koordinatlarına çevrilirler. Daha sonra mevcut görüntüdeki izlenen köşe noktaları algoritmaya tekrar ilk değerlendiri olarak verilir ve bu değerler bir sonraki görüntüde izlenir. Şekil 3.2 ve 3.3' de LKT algoritması ile üç kutunun köşelerinin başarılı bir şekilde bir sonraki görüntüde izlenmesi gösterilmiştir.



Şekil 3.2 LKT algoritmasına köşelerin izlenecek özellikler olarak verilmesi

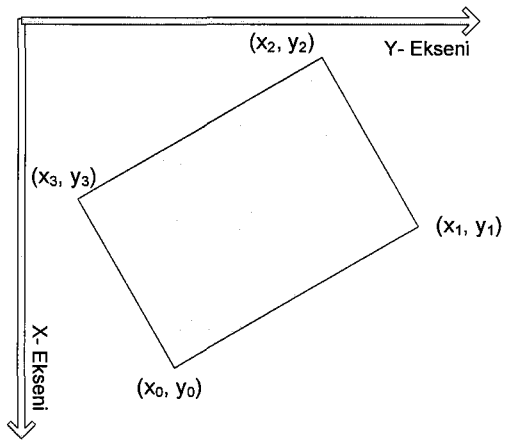


Şekil 3.3 Özellik noktalarının bir görüntü sonra LKT algoritması ile izlenmesi

3.1.2 Kutuların Yeni Üç Boyuttaki Koordinatlarının Hesaplanması

Kutuların gerek izlenmesi, gerek bilinen köşe koordinatları ve kenar uzunluklarından faydalanılarak kutuların üst yüzey yapılarının tekrar oluşturulabilmesi için köşelerin belirli bir düzende tutulması gerekmektedir. Bu durumda Şekil 3.4'de görüldüğü gibi kutuların x ekseninde en büyük değere sahip

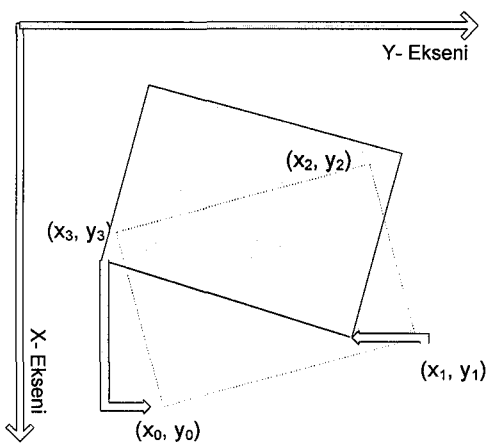
olan köşesinin indeksine 0 atanması ve diğerlerine saat yönü tersinde 1, 2 ve 3 olarak atanması uygun görülmüştür.



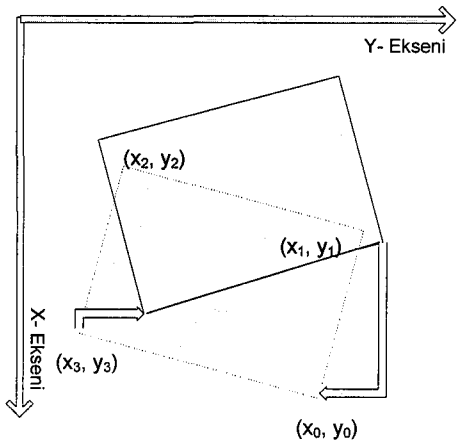
Şekil 3.4 Koordinat ekseninde bir kutunun konumu

Kutular hareket halinde iken x eksenindeki en büyük değere sahip olan köşeleri değişebilir. Bir kutunun dönmesi, indeksi 3 olan köşenin ya da 1 olan köşenin mevcut indeksi 0 olan köşenin x ekseni göz önüne alınarak önüne geçmesi demektir. Bu durumda bütün köşelerin indeksleri tekrar düzenlenmelidir.

Kutular Şekil 3.5 veya Şekil 3.6'da görüldüğü gibi hareket ettiğinde indeksi 3 olan ya da 1 olan köşe x ekseninde en büyük değere sahip olacağı için indeksi 0 olarak değiştirilir ve buna bağlı olarak diğer köşelerde saatin tersi yönünde olmak üzere yeni indekslerini alırlar.



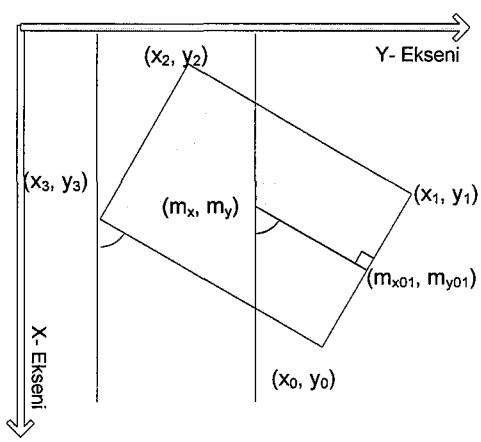
Şekil 3.5 Bir kutunun indeksi 3 olan köşesinin x ekseninde önce geçmesi



Şekil 3.6 Bir kutunun indeksi 1 olan köşesinin x ekseninde önce geçmesi

Kutuların köşeleri uygun bir şekilde sıraya konulduktan sonra izlenilen köşelere göre kutuların üst yüzey yapısı tekrar oluşturulur. Bu sayede izlenebilen köşelerden yararlanılarak izlenemeyen köşeler tekrar hesaplanır. Kutuların bant üzerinde izleme yapılan bölüme geldikleri andaki köşe koordinatları bilindiği için kutuların üst yüzeylerinin kenar uzunlukları hesaplanabilir. Kutuların kenar uzunlukları bilindiği için, herhangi iki köşenin izlenebilmesi sonucunda diğer köşelerin koordinatları hesaplanabilir. Sırasıyla 0-1, 0-2, 0-3, 1-2, 1-3, 2-3 indeksli köşe çiftlerinin izlenme durumuna göre altı şekilde bir kutunun üst yüzey yapısı tekrar oluşturulabilir. Bu çiftlere göre kutunun ağırlık merkezi koordinatları ve x ekseni ile yaptığı açı bulunur. Her çift için bulunan ağırlık merkezi koordinatlarının ve açıların ortalaması alınarak farklı bir algoritma ile kutuların üst yüzey yapısı tekrar oluşturulur.

Herhangi bir kutunun 0 ve 1 indeksli köşelerinin izlendiği durumda, öncelikle kutunun x ekseni ile yaptığı açı hesaplanır.



Şekil 3.7 Bir kutunun 0 ve 1 indeksli köşelerinin izlenmesi durumu

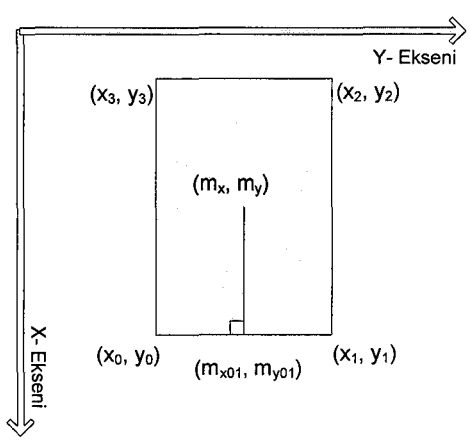
Şekil 3.7'de görülen (m_{x01}, m_{y01}) koordinatları, (x_0, y_0) ve (x_1, y_1) noktalarından geçen doğrunun orta noktasıdır. (m_x, m_y) koordinatları kutunun ağırlık merkezi koordinatlarıdır. Şekil 3.7'de görüldüğü gibi, kutunun x ekseni ile yaptığı açı (m_{x01}, m_{y01}) ve (m_x, m_y) noktalarından geçen doğrunun x ekseni ile yaptığı açıya eşittir. (x_0, y_0) ve (x_1, y_1) noktaları bilindiğine göre bu doğrunun eğimi

$$m_1 = (y_0 - y_1) / (x_1, y_1)$$

şeklinde bulunur. Bu doğru, (m_{x01}, m_{y01}) ve (m_x, m_y) noktalarından geçen doğru ile dik olacağı için, (m_{x01}, m_{y01}) ve (m_x, m_y) doğrusunun eğimi;

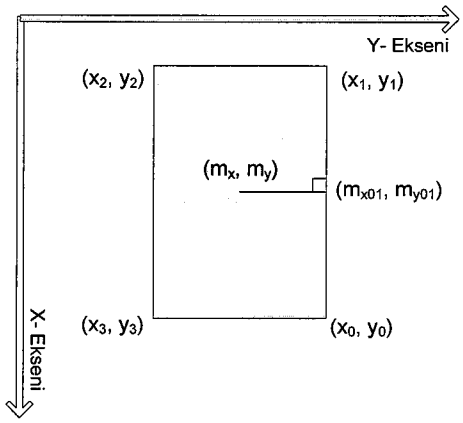
$$m_2 = -1 / m_1$$

şeklinde bulunur ve bu eğiminin ters tanjantı alındığında x ekseni ile yapılan açı hesaplanır. Eğimler hesaplanırken 0'a bölünme hatası ile karşılaşmamak için kutunun x eksenine paralelliği ayrıca ele alınmıştır. Kutunun 0 ve 1 indeksli köşelerinin x koordinatları arasında ve y koordinatları arasında 10 mm'lik bir fark eşik değeri olarak başarılı sonuçlar vermiştir. Şekil 3.8'de görüldüğü gibi $|x_0 - x_1|$ değerinin 10 mm. eşik değerinden küçük olması durumunda kutunun x ekseni ile yaptığı açı 0 radyana eşitlenir.



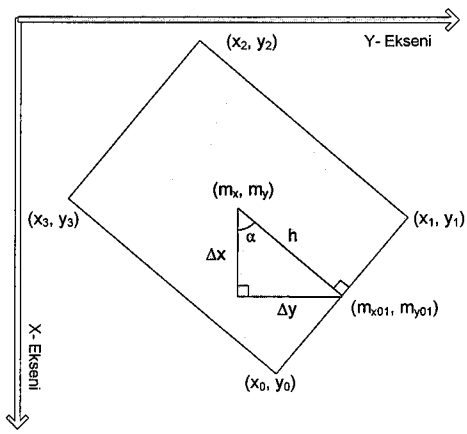
Şekil 3.8 $|x_0-x_1|$ icin 10 mm eşik değeri

Şekil 3.9'da görüldüğü gibi $|y_0 - y_1|$ değerlerinin 10 mm eşik değerinden küçük olması durumlarında ise kutunun x ekseni ile yapılan açı π / 2 radyana eşitlenir.



Şekil 3.9 $|y_0 - y_1|$ için 10 mm eşik değeri

x ekseni ile yapılan açı hesaplandıktan sonra kutunun ağırlık merkezi koordinatları hesaplanır.



Şekil 3.10 0 ve 1 indeksli köşelere göre ağırlık merkezi hesaplanması

Kutuların kenarları uzunlukları bilindiğine göre Şekil 3.10'da görülen, h uzaklığı hesaplanabilir:

$$h = D_x / 2$$

Kutunun Şekil 3.10'da görülen konumu için α değeri kutunun x ekseni ile yaptığı açıya(θ) eşittir. h ve θ' dan faydalanılarak Δx ve Δy değerleri hesaplanır.

$$\alpha = \theta$$

$$\Delta x = h \,.\, \cos(\alpha)$$

$$\Delta y = h \,.\, \sin(\alpha)$$

m_{x01}, m_{y01} koordinatlarından x ekseninde $-\Delta x$, y ekseninde $-\Delta y$ kadar ilerlendiğinde kutunun ağırlık merkezi koordinatlarına ulaşılır.

$$m_x = m_{x01} - \Delta x$$

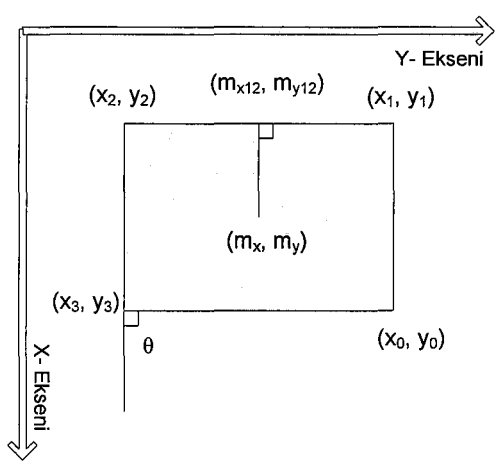
$$m_y = m_{y01} - \Delta y$$

Bir kutunun, 1 ve 2 indeksli köşelerin takip edildiği durumlarda öncelikle, $|x_1 - x_2|$ ve $|y_1 - y_2|$ değerlerinin 10 mm. eşik değerinden küçük olup olmadığına bakılır.

Eğer Şekil 3.11'de görüldüğü gibi $|x_1 - x_2|$ değeri 10 mm. eşik değerinden küçük ise, kutunun x ekseni ile yaptığı açı

$$\theta = \pi / 2 \text{ radyan}$$

olarak eşitlenir. Bu durumda α açısının değeri ise 0 radyandır.

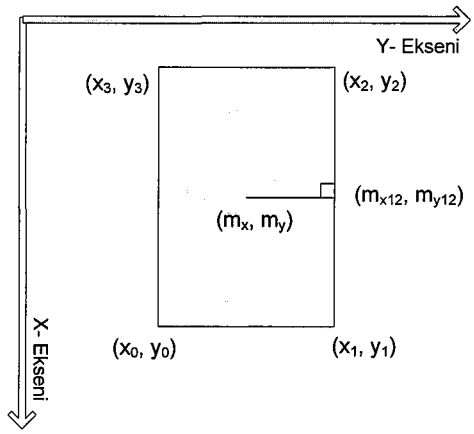


Şekil 3.11 $|x_1-x_2|$ icin 10 mm eşik değeri

Eğer Şekil 3.12'de görüldüğü gibi $|y_1 - y_2|$ değeri 10 mm eşik değerinden küçük ise, kutunun x ekseni ile yaptığı açı

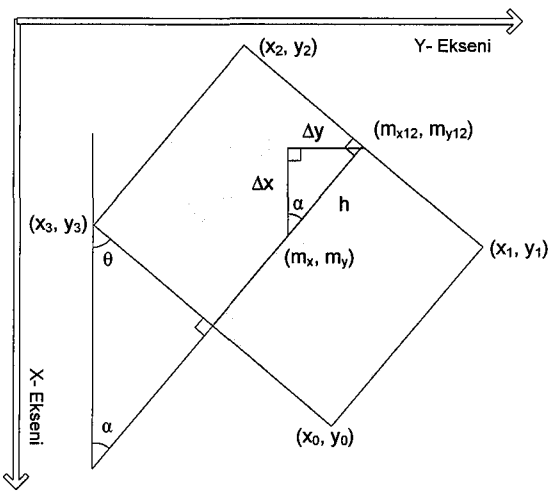
$\theta = 0$ radyan.

olarak eşitlenir. Bu durumda α açısının değeri ise $\pi / 2$ radyandır.



Şekil 3.12 $|y_1-y_2|$ icin 10 mm eşik değeri

Eğer $|x_1 - x_2|$ ve $|y_1 - y_2|$ değerlerinin 10 mm eşik değerinden küçük değilse, kutu Şekil 3.13' de görüldüğü gibi bir konumdadır. Bu durumda (x_1, y_1) ve (x_2, y_2) noktalarından geçen doğrunun eğimi hesaplanır.



Şekil 3.13 Bir kutunu 1 ve 2 indeksli köşelerinin izlenmesi durumu

$(m_x, m_y) - (m_{x12}, m_{y12})$ noktalarından geçen doğrunun eğimi m_2, $(x_1, y_1) - (x_2, y_2)$ noktalarından geçen doğrunun eğimi m_1 olsun.

$$m_1 = (y_1 - y_2) / (x_1 - x_2)$$

Bu iki doğru bir birine dik olduğu için :

$$m_2 = -1 / m_1$$

olarak hesaplanır. m_2 eğiminin ters tanjantı alındığında α açısı negatif yönlü olarak bulunur. Şekil 3.13' de görüldüğü gibi kutunun x ekseni ile yaptığı θ açısı

$$\theta = ((\pi / 2) - \alpha) \text{ radyan}$$

olarak hesaplanır. Kutunun kenar uzunlukları bilindiğinden h değeri:

$$h = D_y / 2$$

olarak hesaplanır. α açısı ve h'dan faydalanılarak Δx ve Δy değerleri hesaplanır.

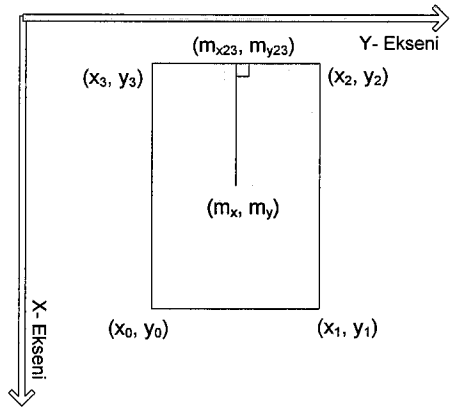
$$\Delta x = h \, . \cos(\alpha)$$
$$\Delta y = h \, . \sin(\alpha)$$

m_{x01}, m_{y01} koordinatlarından x ekseninde $+\Delta x$, y ekseninde $-\Delta y$ kadar ilerlendiğinde kutunun ağırlık merkezi koordinatlarına ulaşılır.

$$m_x = m_{x12} + \Delta x$$
$$m_y = m_{y12} - \Delta y$$

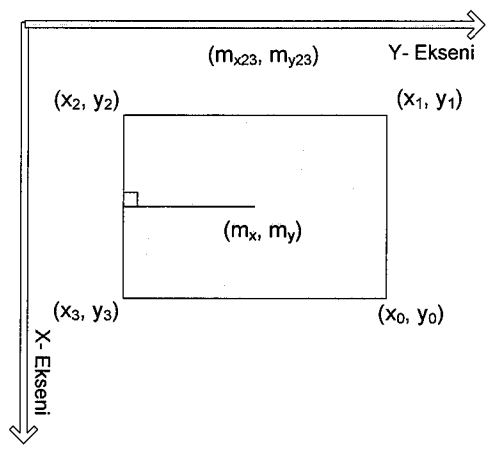
Bir kutunun 2 ve 3 indeksli köşelerin takip edildiği durumlarda, öncelikle, $|x_2 - x_3|$ ve $|y_2 - y_3|$ değerlerinin 10 mm. eşik değerinden küçük olup olunmadığına bakılır.

Eğer Şekil 3.14' de görüldüğü gibi $|x_2 - x_3|$ değeri 10 mm. eşik değerinden küçük ise, kutunun x ekseni ile yaptığı açı θ ve α açıları 0 radyana eşittir.



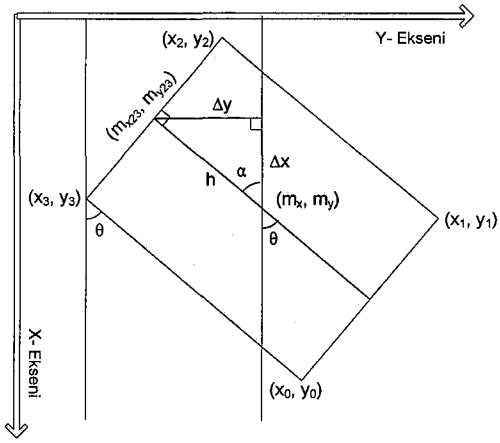
Şekil 3.14 $|x_2-x_3|$ icin 10 mm eşik değeri

Eğer Şekil 3.15'de görüldüğü gibi $|y_2 - y_3|$ değeri 10 mm eşik değerinden küçük ise, kutunun x ekseni ile yaptığı açı θ ve α açıları $\pi / 2$ radyana eşittir.



Şekil 3.15 $|y_2-y_3|$ icin 10 mm eşik değeri

Eğer $|x_2 - x_3|$ ve $|y_2 - y_3|$ değerleri 10 mm eşik değerinden küçük değilse, kutu Şekil 3.16'da görüldüğü gibi bir konumdadır. Bu durumda $(x_2, y_2) - (x_3, y_3)$ noktalarından geçen doğrunun eğimi hesaplanır.



Şekil 3.16 Bir kutunu 2 ve 3 indeksli köşelerinin izlenmesi durumu

$(m_x, m_y) - (m_{x23}, m_{y23})$ noktalarından geçen doğrunun eğimi m_2, $(x_2, y_2) - (x_3, y_3)$ noktalarından geçen doğrunun eğimi m_1 olsun.

$$m_1 = (y_2 - y_3) / (x_2 - x_3)$$

Bu iki doğru bir birine dik olduğu için :

$$m_2 = -1 / m_1$$

olarak hesaplanır. m_2 eğiminin ters tanjantı alındığında kutunun x ekseni ile yaptığı θ açısı bulunur. Şekil 3.16'da görüldüğü gibi α açısı θ açısına eşittir.

$$\alpha = \theta$$

Kutunun kenar uzunlukları bilindiğinden h değeri:

$$h = D_x / 2$$

olarak hesaplanır. α açısı ve h'dan faydalanılarak Δx ve Δy değerleri hesaplanır.

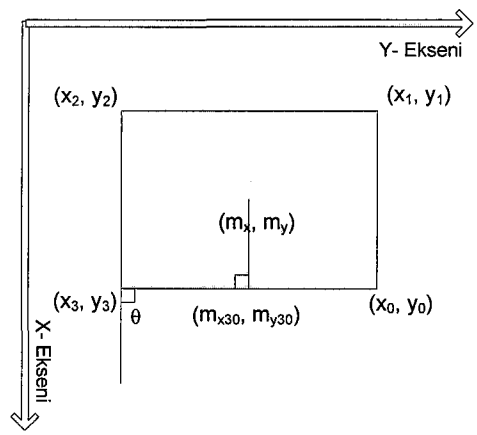
$$\Delta x = h \cdot \cos(\alpha)$$
$$\Delta y = h \cdot \sin(\alpha)$$

m_{x23}, m_{y23} koordinatlarından x ekseninde -Δx, y ekseninde -Δy kadar ilerlendiğinde kutunun ağırlık merkezi koordinatlarına ulaşılır.

$$m_x = m_{x23} - \Delta x$$
$$m_y = m_{y23} - \Delta y$$

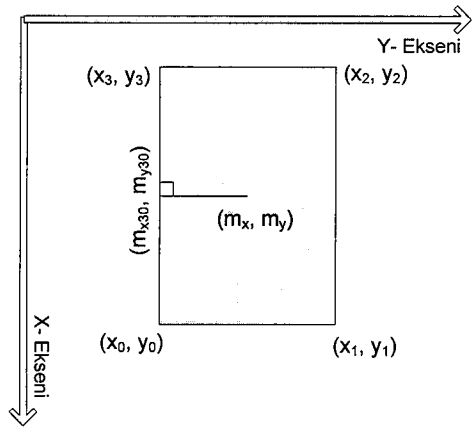
Bir kutunun 3 ve 0 indeksli köşelerin takip edildiği durumlarda, öncelikle, $|x_3 - x_0|$ ve $|y_3 - y_0|$ değerlerinin 10 mm. eşik değerinden küçük olup olunmadığına bakılır.

Eğer Şekil 3.17'de görüldüğü gibi $|x_3 - x_0|$ değeri 10 mm. eşik değerinden küçük ise, kutunun x ekseni ile yaptığı açı θ, π / 2 radyana, α açısı ise 0 radyana eşittir.



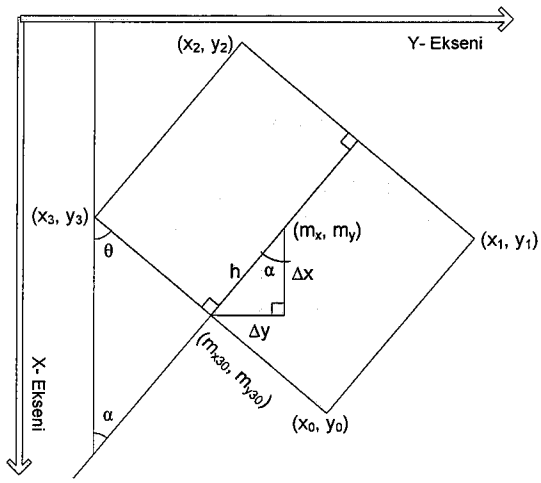
Şekil 3.17$|x_3-x_0|$ için 10 mm eşik değeri

Eğer Şekil 3.18'de görüldüğü gibi $|y_3 - y_0|$ değeri 10 mm. eşik değerinden küçük ise, kutunun x ekseni ile yaptığı açı θ, 0 radyana, α açısı ise π / 2 radyana eşittir.



Şekil 3.18 $|y_3-y_0|$ icin 10 mm eşik değeri

Eğer $|x_0 - x_3|$ ve $|y_3 - y_0|$ değerleri 10 mm. eşik değerinden küçük değilse, kutu Şekil 3.19'da görüldüğü gibi bir konumdadır. Bu durumda $(x_0, y_0) - (x_3, y_3)$ noktalarından geçen doğrunun eğimi hesaplanır.



Şekil 3.19 Bir kutunu 3 ve 0 indeksli köşelerinin izlenmesi durumu

$(m_x, m_y) - (m_{x30}, m_{y30})$ noktalarından geçen doğrunun eğimi m_2, $(x_0, y_0) - (x_3, y_3)$ noktalarından geçen doğrunun eğimi m_1 olsun.

$$m_1 = (y_3 - y_0) / (x_3 - x_0)$$

Bu iki doğru bir birine dik olduğu için :

$$m_2 = -1 / m_1$$

olarak hesaplanır. m_2 eğiminin ters tanjantı alındığında α açısı negatif yönlü olarak bulunur. Şekil 3.19'da görüldüğü gibi kutunun x ekseni ile yaptığı θ açısı

$$\theta = ((\pi / 2) - \alpha) \text{ radyan}$$

olarak bulunur. Kutunun kenar uzunlukları bilindiğinden h değeri:

$$h = D_y / 2$$

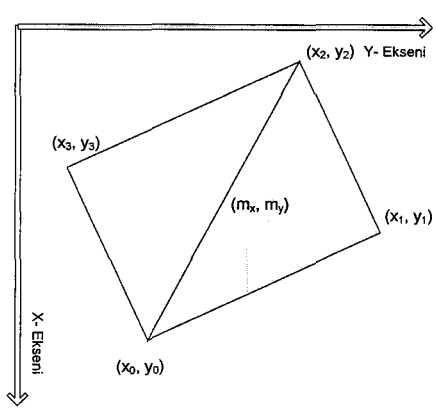
olarak hesaplanır. α açısı ve h'dan faydalanılarak Δx ve Δy değerleri hesaplanır.

$$\Delta x = h \,.\, \cos(\alpha)$$
$$\Delta y = h \,.\, \sin(\alpha)$$

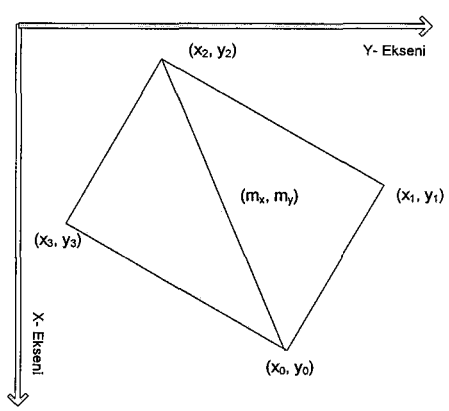
m_{x30}, m_{y30} koordinatlarından x ekseninde $-\Delta x$, y ekseninde $+\Delta y$ kadar ilerlendiğinde kutunun ağırlık merkezi koordinatlarına ulaşılır.

$$m_x = m_{x30} - \Delta x$$
$$m_y = m_{y30} + \Delta y$$

Herhangi bir kutunun 0 ve 2 indeksli köşelerinin takip edildiği durumlarda, kutunun iki farklı konumu göz önünde bulundurularak hesaplamalar yapılmalıdır. Bu konumlar Şekil 3.20 ve Şekil 3.21' de görüldüğü gibi 0 indeksli köşenin y eksenindeki koordinatının 2 indeksli köşeden büyük ve küçük olduğu konumlardır.



Şekil 3.20 Bir kutunun 0 ve 2 indeksli köşelerinin izlenmesi ($y_0 < y_2$)



Şekil 3.21 Bir kutunun 0 ve 2 indeksli köşelerinin izlenmesi ($y_0 > y_2$)

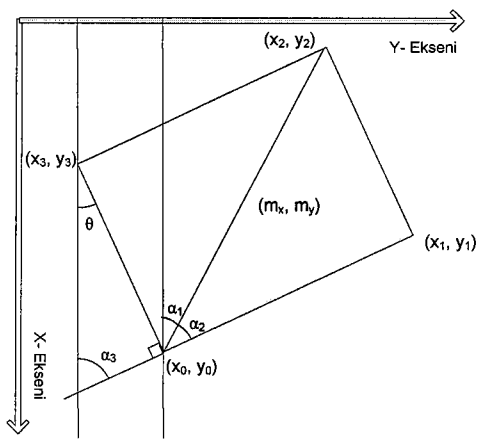
Kutunun her iki konumu içinde öncelikle kutunun ağırlık merkezi koordinatları bulunur. Şekil 3.20'de görülen konum için kutunun ağırlık merkezi:

$$m_x = x_2 + (x_0 - x_2) / 2$$
$$m_y = y_0 + (y_2 - y_0) / 2$$

Şekil 3.21'de görülen konum için kutunun ağırlık merkezi:

$$m_x = x_2 + (x_0 - x_2) / 2$$
$$m_y = y_2 + (y_0 - y_2) / 2$$

olarak hesaplanır.



Şekil 3.22 ($y_0 < y_2$) durumunda açı ve merkez hesaplanması

Kutunun Şekil 3.22'de ki konumu göz önüne alınırsa kutunun x ekseni ile yaptığı açının hesaplanması için öncelikle $(x_0,y_0) - (x_2, y_2)$ noktalarından geçen doğrunun eğimi hesaplanır.

$$m_1 = (y_2 - y_0) / (x_2 - x_0)$$

Bu eğimin ters tanjantı alındığında negatif yönlü α_1 açısı bulunur. Kutunun kenar uzunlukları da bilindiğinden α_2 açısı bulunabilir.

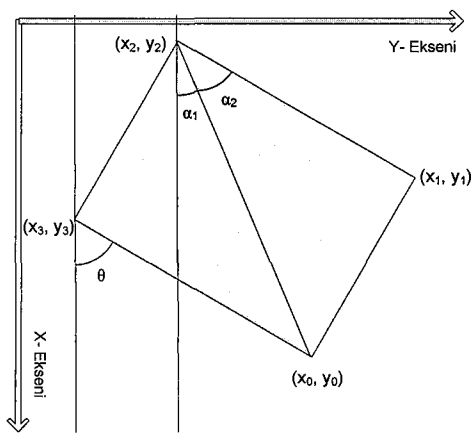
$$m_2 = D_x / D_y$$

$$\alpha_2 = \text{arctanjant}(m_2)$$

α_3 açısı α_1 ve α_2 açılarının toplamına eşittir. Kutunun x ekseni ile yaptığı θ açısı ise

$$\theta = ((\pi / 2) - \alpha_3) \text{ radyan}$$

olarak hesaplanır.



Şekil 3.23 ($y_0 > y_2$) durumunda açı ve merkez hesaplanması

Kutunun Şekil 3.23' de ki konumu göz önündeki alındığında, kutunun x ekseni ile yaptığı açının hesaplanması için α_1 ve α_2 açıları hesaplanmalıdır. α_1 açısı (x_0,y_0) – (x_2, y_2) noktalarından geçen doğrunun eğimi kullanılarak hesaplanır.

$$m_1 = (y_0 - y_2) / (x_0 - x_2)$$

$$\alpha_1 = \text{arctanjant}(m_1)$$

α_2 açısı ise kutunun kenar uzunluklarından yararlanılarak bulunur.

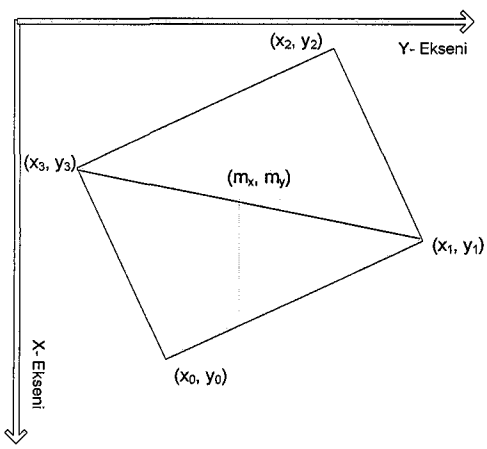
$$m_2 = D_x / D_y$$

$$\alpha_2 = \text{arctanjant}(m_2)$$

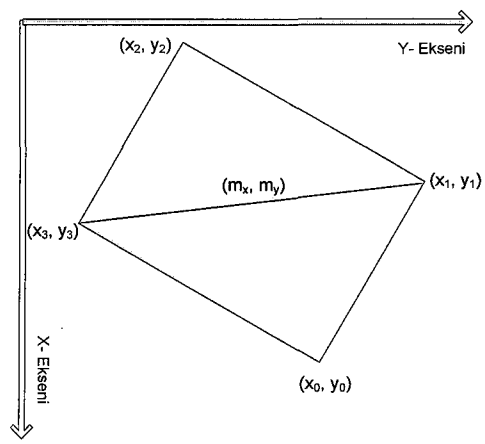
Şekil 3.23'de görüldüğü gibi kutunun x ekseni ile yaptığı θ açısı α_1 ve α_2 açılarının toplamına eşittir.

$$\theta = \alpha_1 + \alpha_2$$

Kutuların 1 ve 3 indeksli köşelerinin takip edildiği durumlarda 0 – 2 indeksli köşelerde olduğu gibi yine kutunun iki farklı konumu göz önünde bulundurularak hesaplamalar yapılmalıdır. Bu konumlar Şekil 3.24 ve Şekil 3.25'de görüldüğü gibi 1 indeksli köşenin x eksenindeki koordinatının 3 indeksli köşeden büyük ve küçük olduğu konumlardır.



Şekil 3.24 Bir kutunun 1 ve 3 indeksli köşelerinin izlenmesi ($x_1 > x_3$)



Şekil 3.25 Bir kutunun 1 ve 3 indeksli köşelerinin izlenmesi ($x_1 < x_3$)

Kutunun her iki konumu içinde öncelikle kutunun ağırlık merkezi koordinatları bulunur. Şekil 3.24' de görülen konum için kutunun ağırlık merkezi:

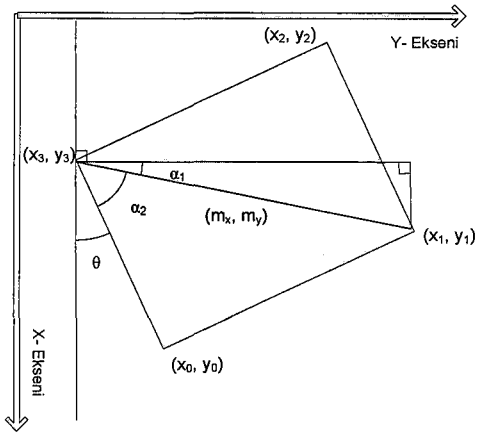
$$m_x = x_3 + (x_1 - x_3) / 2$$
$$m_y = y_3 + (y_1 - y_3) / 2$$

Şekil 3.25' de görülen konum için kutunun ağırlık merkezi:

$$m_x = x_1 + (x_3 - x_1) / 2$$

$$m_y = y_3 + (y_1 - y_3) / 2$$

olarak hesaplanır.



Şekil 3.26 ($x_1 > x_3$) durumunda açı ve merkez hesaplanması

Kutunun Şekil 3.26'da görüldüğü gibi x_1 koordinatının x_3 koordinatından daha büyük olduğu konumu göz önüne alınırsa, α_1 açısı, (x_1, y_1) ve (x_3, y_3) noktaları kullanılarak bulunabilir.

$$\text{arctanjant}(\alpha_1) = (x_1 - x_3) / (y_1 - y_3)$$

Kutunun kenar uzunlukları bilindiği için,

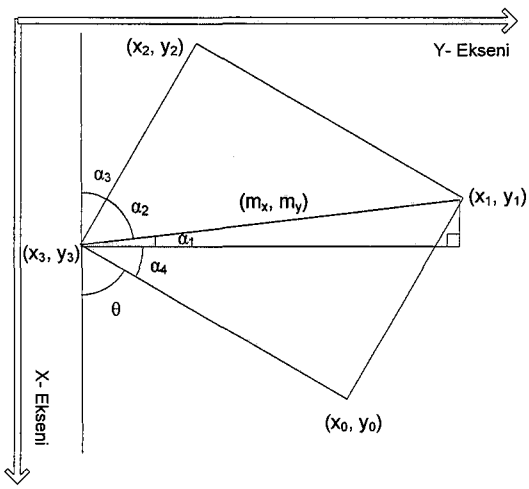
$$\text{arctanjant}(\alpha_2) = D_y / D_x$$

Bu durumda kutunun x ekseni ile yaptığı açı

$$\theta = (\pi / 2) - (\alpha_1 + \alpha_2) \text{ radyan}$$

olarak hesaplanır.

Kutunun Şekil 3.27'de görüldüğü gibi x_3 koordinatının x_1 koordinatından daha büyük olduğu konumu göz önüne alınırsa, α_1 açısı, (x_1, y_1) ve (x_3, y_3) noktaları kullanılarak bulunabilir.



Şekil 3.27 ($x_1 < x_3$) durumunda açı ve merkez hesaplanması

$$\text{arctanjant}(\alpha_1) = (x_3 - x_1) / (y_1 - y_3)$$

Kutunun kenar uzunlukları bilindiği için,

$$\text{arctanjant}(\alpha_2) = D_x / D_y$$

$\alpha_3 = \alpha_4 = (\pi / 2) - (\alpha_1 + \alpha_2)$ olduğuna göre

$$\theta = (\alpha_1 + \alpha_2) \text{ radyan}$$

olarak hesaplanır.

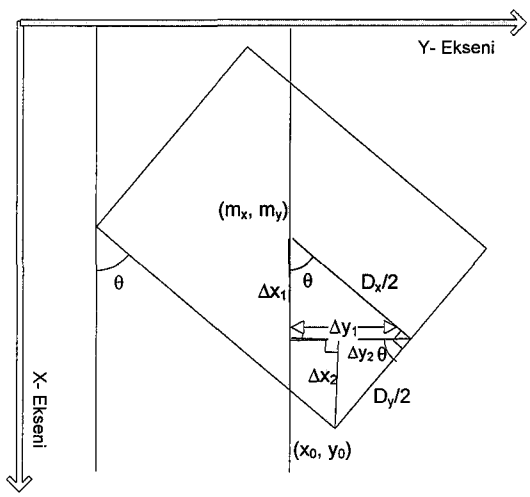
Sonuç olarak izlenebilen her köşe çifti için kutunun üst yüzey yapısı tekrar oluşturulur. k kutunun köşe sayısı(4), t izlenen köşe sayısı ise

$$n = \binom{k}{2} - \binom{k}{t} + 1$$

tane merkez koordinatı ve açı hesaplanır. Bu değerlerin ortalaması alındıktan sonra kutunun tahmin edilen merkez koordinatı ve açısı ile kutu tekrar oluşturularak dört köşe koordinatı tekrar hesaplanır.

$$\theta = \frac{\sum \theta}{n} \qquad m_x = \frac{\sum m_x}{n} \qquad m_y = \frac{\sum m_y}{n}$$

Bir kutunun x ekseni ile yaptığı açı, kutunun kenar uzunlukları ve ağırlık merkezi koordinatları biliniyorsa o kutunun köşe noktalarının koordinatları hesaplanabilir.

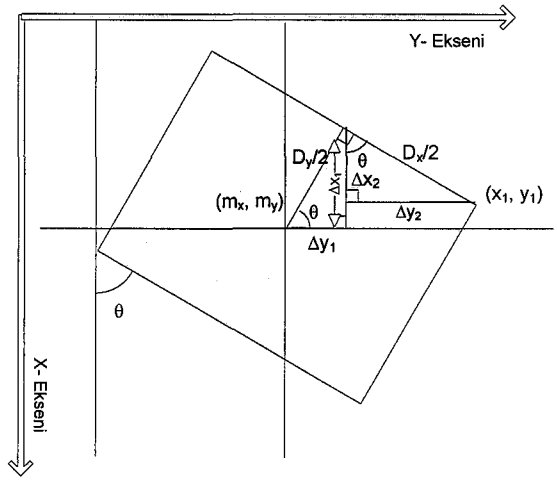


Şekil 3.28 Bir kutunun 0 indeksli köşesinin hesaplanması

Kutunun (x_0,y_0) köşe koordinatları Şekil 3.28'de görüldüğü gibi hesaplanır.

$$\Delta x_1 = (D_x/2) \,.\, \cos(\theta)$$
$$\Delta x_2 = (D_y/2) \,.\, \sin(\theta)$$
$$\Delta y_1 = (D_x/2) \,.\, \sin(\theta)$$
$$\Delta y_2 = (D_y/2) \,.\, \cos(\theta)$$
$$x_0 = m_x + \Delta x_1 + \Delta x_2$$
$$y_0 = m_y + \Delta y_1 - \Delta y_2$$

Kutunun (x_1,y_1) köşe koordinatları Şekil 3.29'da görüldüğü gibi hesaplanır.



Şekil 3.29 Bir kutunun 1 indeksli köşesinin hesaplanması

$$\Delta x_1 = (D_y/2) \,.\, \sin(\theta)$$

$$\Delta x_2 = (D_x/2) \,.\, \cos(\theta)$$

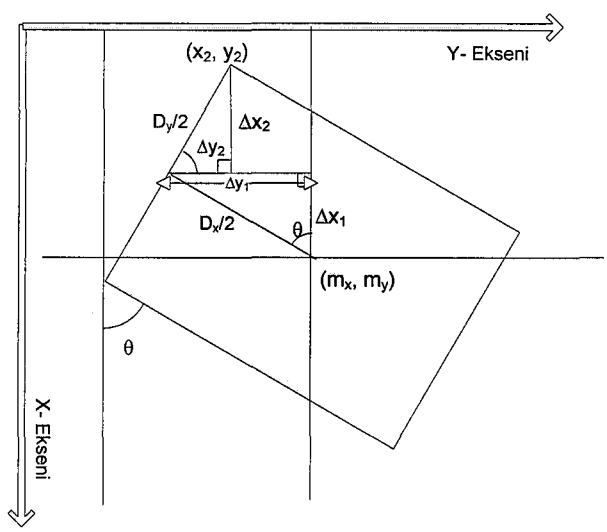
$$\Delta y_1 = (D_y/2) \,.\, \cos(\theta)$$

$$\Delta y_2 = (D_x/2) \,.\, \sin(\theta)$$

$$x_1 = m_x + \Delta x_2 - \Delta x_1$$

$$y_1 = m_y + \Delta y_1 + \Delta y_2$$

Kutunun (x_2,y_2) köşe koordinatları Şekil 3.30'da görüldüğü gibi hesaplanır.



Şekil 3.30 Bir kutunun 2 indeksli köşesinin hesaplanması

$$\Delta x_1 = (D_x/2) \,.\, \cos(\theta)$$

$$\Delta x_2 = (D_y/2) \,.\, \sin(\theta)$$

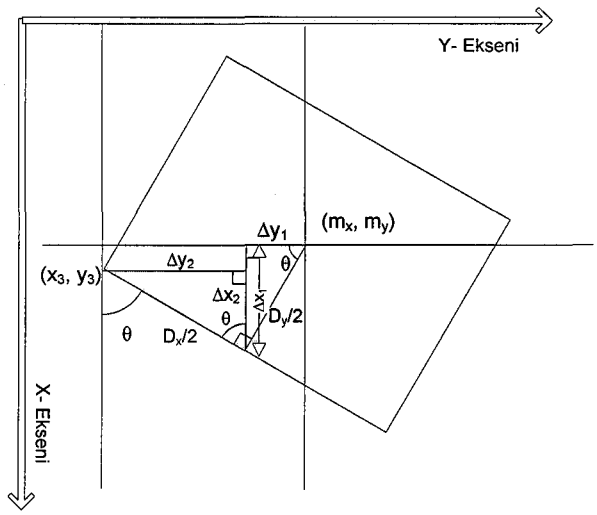
$$\Delta y_1 = (D_x/2) \,.\, \sin(\theta)$$

$$\Delta y_2 = (D_y/2) \cdot \cos(\theta)$$

$$x_2 = m_x - (\Delta x_1 - \Delta x_2)$$

$$y_2 = m_y + \Delta y_2 - \Delta y_1$$

Kutunun (x_3,y_3) köşe koordinatları Şekil 3.31'de görüldüğü gibi hesaplanır.



Şekil 3.31 Bir kutunun 3 indeksli köşesinin hesaplanması

$$\Delta x_1 = (D_y/2) \cdot \sin(\theta)$$

$$\Delta x_2 = (D_x/2) \cdot \cos(\theta)$$

$$\Delta y_1 = (D_y/2) \cdot \cos(\theta)$$

$$\Delta y_2 = (D_x/2) \cdot \sin(\theta)$$

$$x_1 = m_x + \Delta x_1 - \Delta x_2$$

$$y_1 = m_y - (\Delta y_1 + \Delta y_2)$$

3.1.3 Değerlendirme

LKT izleme algoritması ve kutuların köşelerinin üç boyuttaki koordinatlarının takip edilebilen en az iki köşe koordinatları sayesinde tekrar oluşturulmasıyla izleme işlemi başarısız olmuştur. Ek – 1, Şekil 1 – 2'de izleme algoritmasının başarısız olduğunu gösteren bazı ekran görüntüleri görülmektedir. Bu görünütlerde bazı kutuların kenarları yeşil, bazılarının ise mavi görülmektedir. Kutular bandın izleme yapılan kısıma gelmeden önce yüksekliklerinin ve izleme bandına girer girmezki köşe noktlarının hesaplanma işlemleri yapılırken kutuların kenarları yeşil ile çizilmektedir ve bu konu bu tezin kapsamı içerisinde değildir. Kutular bant üzerinde izleme yapılan alana gelir gelmez izleme algoritması tarafından hesaplanan köşelerden faydalanarak kenarlar mavi renkte çizilmiştir.

Algoritmanın başarısız olması ve köşe noktalarının doğru takip edilememesi sonunca Ek-1 Şekil 4.1 – 4.2'de görüldüğü gibi kutuların üzerindeki mavi kenar çizgileri olmaları gereken yerde değildir.

LKT izleme algoritması doğrudan kullanılması yeterli olmamıştır. Bu başarısızlığın en önemli sebebi görüntü kalitesi, ışıklandırma ve üretim bandının yapısından kaynaklanmaktadır. Görüntülerin alındığı üretim bandının mevcut ışıklandırma koşulları görüntü işleme için elverişli değildir. Her kamera arasında büyük ışıklandırma farklılıkları yer almaktadır. Bir kutu tek bir kamera görüntü içerisinde ilerlerken dahi kameranın görüntüleme sınırlarına girdiği konumu ile çıktığı konumu arasında görüntülerde yüksek seviyede parlaklık ve renk yoğunluğu farkları yer almaktadır.

Üretim bandının yapı itibariyle iki kenarı açık renkte ve kutuların ilerlediği bantlar koyu renktedir. Kutuların yüksekliği belli bir değerden büyük olduğunda mercek bozulmasının etkisiyle görüntülerde köşeler üretim bandının kenarları ile üst üste gelmektedir. Bu durumda, kenarlıkla üst üste gelen köşelerin çevresi ile arasındaki yoğunluk farkı azalmaktadır.

Bütün bu sorunlar bir araya geldiğinde, LKT algoritması doğrudan kullanıldığında ancak bir kaç görüntülük izleme yapılabilmekte ve daha sonra uygulama başarısız olmaktadır.

İzlemede başarılı olunmasına rağmen kutuların üst yüzey yapılarının korunması ve izlenemeyen köşelerin izlenebilen köşeler yardımıyla bulunması sağlanmıştır. İzlenemeyen köşelerin hesaplanması için diğer köşelerin doğru izlenildiği varsayılmıştır.

Bu durumda, tekrar görüntü düzeyine inilerek izleme algoritmasından elde edilen her bir yeni özelliğin kontrol edilerek doğruluğunun onaylanması gerekliliği ortaya çıkmıştır. Eğer algoritmanın döndürdüğü değer doğru değilse görüntü üzerinde bu nokta doğru koordinatlara ötelenmelidir.

3.2 Görüntüler Üzerinde Özellik Algılama

Bölüm 3.1.3' de belirtildiği gibi, LKT algoritmasından dönen sonuçların doğrudan kullanılması ile elde edilen köşe koordinatlarına dayalı olarak yapılan izleme işlemi başarısız olmaktadır. Bu yüzden LKT algoritmasından elde edilen sonuçların düzeltilmesi gerekmektedir. Bu düzeltme işlemi için Bölüm 2.3.4'de

anlatılan özellik algılama algoritması kullanılmıştır. LKT algoritmasından dönen noktaların etrafında belli bir çerçevedeki pikseller incelenmiş ve bu noktalar ilgili çerçevede bulunan en iyi özelliğe ötelenmiştir. Daha sonra Bölüm 3.1.2'de anlatılan kutuların üç boyuttaki koordinatlarının hesaplanması algoritması kullanılmıştır. Bu durumda yeni algoritma aşağıdaki gibidir:

1. Kutunun köşelerinin üç boyuttaki koordinatlarının kalibrasyondan yararlanılarak iki boyuttaki görüntü piksel değerlerine çevrilmesi

2. Bu köşe noktalarının LKT algoritmasına bir sonraki görüntüde izlenmek üzere girdi olarak kullanılması

3. LKT algoritması ile piksellerin izlenmesi

4. LKT algoritmasından dönen noktaların etrafında özellik algılama algoritmasını kullanarak bu noktaların daha iyi noktalara ötelenmesi

5. İyileştirilmiş yeni köşe koordinatları ile kutuların üç boyuttaki koordinatlarının hesaplanması

Kutular, üretim bandı üzerinde izleme yapılacak olan alana girdiklerinde izleme algoritmasının özellik algılaması yapmasına izin vermeksizin izlemesi gereken noktalar özellik listesinde algoritmaya doğrudan verilerek izlenmesi beklenmiştir.

İzleme algoritmasından elde edilen sonuçlar bir sonraki görüntüde özellik listesi biçiminde girdi olarak kullanılmıştır. Bu durumda izleme algoritmasının yaptığı hatalar bir sonraki görüntüye aktarılmış ve görüntüden görüntüye aktarılan bu hatalar bir kaç görüntü sonrasında izleme algoritmasının başarısız olmasına sebep olmuştur.

LKT izleme algoritması özellik izlemenin yanı sıra özellik algılama işleminde de başarılı sonuçlar vermektedir. Bu sebeple t zamanındaki bir görüntüdeki görüntü listesi $t + 1$ zamanındaki görüntüde izlendikten sonra $t + 1$ zamanındaki görüntüde özellik algılaması yapılarak izlendiği varsayılan ama hatalı izlenen noktalar doğru yerlerine ötelenebilir.

Kısaca, her izleme işlemi sonucunda izlenen köşe noktaları çevresinde bir özellik algılama yapılarak köşe noktaları yeni bulunan özellik koordinatlarına

ötelenmiştir. Bir görüntü üzerindeki özelliklerin algılanması için her bir piksel tek tek ele alınır ve 2 x 2 gradyan matrisinin minimum özdeğeri bu özelliğin izlenebilir bir özellik olup olmadığına karar vermek için kullanılır. Her piksel için özdeğerleri hesaplandıktan sonra bu pikseller özdeğerlerine göre azalan sırada dizilirler ve en üstten başlayarak izleme içeriğinde belirtilen minimum özdeğerden büyük olanlar yine izleme içeriğinde belirtilen özellikler arasındaki minimum uzaklık göz önünde bulundurularak seçilir ve özellik algılama işlemi tamamlanır.

İzleme ve kutuların tekrar oluşturulması algoritmaları özellik algılama ile desteklendiğinde izleme algoritmasının çevresi ile belirgin renk yoğunluğu taşıyan kutular için iyi sonuçlar vermiştir ancak denetleyici daha iyi sonuçlara ihtiyaç duymaktadır. Çevresi ile yeterli renk yoğunluğunu oluşturamayan kutular için özellik algoritması doğası gereği iyi sonuçlar vermemiştir. Ek – 2, Şekil 5.1 – 5.4'de izlemenin başarısızlığı görülmektedir. Görünütlerde kenarları yeşil renkte çizilmiş olan kutular için henüz izleme işlemleri başlamamıştır. Ek – 2, Şekil 5.1'de son kamerada görüldüğü gibi üretim bandına ilk giren iki kutu izlenebilmiştir fakat köşe noktalarının daha az kusurlu olması gerektiği açıktır. Ek – 2 Şekil 5.2' de ise üretim bandının kenarına yakın olan kutunun bir köşesinin bandın kenarlığı ile üst üste gelmesiyle yoğunluk farkının korunamadığı izleme işlemi yanlış sonuçlar vermiştir. Benzer şekilde Ek – 2, Şekil 5.3 – 5.4'de de izleme işlemindeki başarısızlıklar açıkça görülmektedir.

Sonuç olarak, özellik algılama ile izleme sürecinde büyük bir başarı sağlanmış olmasına rağmen bazı kutuların izlenmesi sağlanamamış ve izlenebilenlerin köşe noktalarının daha az kusurlu koordinatlarına ihtiyaç duyulmuştur. Bu durumda özellik algılama ile yapılan düzeltme işlemi yeterli olmamıştır.

3.3 Doğru Uyarlama Algoritması

Bölüm 3.2'de açıklanan algoritma iyi sonuçlar vermiştir fakat bazı kutular izlenememiş ve izlenebilenler için daha az kusurlu koordinatlarla ihtiyaç duyulmuştur. Bu sebeple özellik algılama ve kutuların üç boyuttaki koordinatlarının hesaplanma işlemlerinden sonra tekrar görüntü seviyesine inilerek kutuların kenarları üzerinde son bir düzeltme algoritması kullanılmasına karar verilmiştir. Kullanılan son algoritma aşağıdaki gibidir:

1. Kutunun köşelerinin üç boyuttaki koordinatlarının kalibrasyondan yararlanılarak iki boyuttaki görüntü piksel değerlerine çevrilmesi

2. Bu köşe noktalarının LKT algoritmasına bir sonraki görüntüde izlenmek üzere girdi olarak kullanılması

3. LKT algoritması ile piksellerin izlenmesi

4. LKT algoritmasından dönen noktaların etrafında özellik algılama algoritmasını kullanarak bu noktaların daha iyi noktalara ötelenmesi

5. İyileştirilmiş yeni köşe koordinatları ile kutuların üç boyuttaki koordinatlarının hesaplanması

6. Görüntülerde kutuların kenar bilgisini kullanarak daha önceki adımlarda elde edilen koordinatların düzeltilerek güncellenmesi.

LKT izleme algoritması ve bunu takip eden kutuların üst yüzeylerinin tekrar oluşturulması ile özellik algılama algoritmalarından sonra istenilen sonuçlara ulaşabilmek için tekrar görüntü seviyesine inilerek hesaplanan kutu kenarlarının doğruluğu test edilmiştir.

Her bir görüntüde LKT izleme algoritması sonucunda kutuların bazı köşeleri izlenmiştir. Daha sonra izlenemeyen köşeler, izlenebilenler kullanılarak hesaplanmıştır ve özellik algılama algoritması ile yanlış hesaplanan köşeler doğru yerlerine ötelenmeye çalışılmıştır. Köşeler üzerinde yapılan bu çalışmalardan sonra gene hesaplanan köşelerden yararlanılarak kutuların kenarlarının doğruluğu test edilmiştir.

Kutuların bilinen köşelerinden yola çıkılarak öncelikle Bölüm 2.4'te anlatılan doğru uyarlama algoritması ile köşeleri birleştiren kenar doğruları hesaplanmıştır. Oluşturulan bu kenar doğrularının ideal yerlerinin hesaplanabilmesi için bir arama uzayı oluşturulmuştur. Arama uzayı; kutunun x ekseni ile yaptığı θ açısının sırasıyla -15, 0, 15 derece değişimi, kutunun merkez koordinatlarının x ekseni -20, 0, 20, 40 mm ve y ekseni ile -20, 0, 20 mm ötelenmeleri hesaplanarak oluşturulmuştur. Bu ötelemeler sayesinde köşe koordinatlarından hesaplanan kenar doğrusunun gerçek yeri 36 farklı konumda aranmıştır.

Bu arama için her konumda hesaplanan yeni kenar doğrularının doğruluğu test edilmiştir. Bu test sırasında öncelikle Bölüm 2.4'te açıklanan doğru uyarlama algoritması ile ilgili doğruyu oluşturan pikseller bulunmuştur. Daha sonra bu piksellerin 3'er piksel aralığında alt ve üst pikseller ile arasındaki renk yoğunluk farkları hesaplanmıştır. İdeal kenar doğrularının en yüksek yoğunluk farkını vereceği için bu farkı veren kenar doğruları ideal kenarlar olarak seçilmiş ve daha önceki algoritmaların yaptığı hatalar düzeltilmeye çalışılmıştır.

Doğru uyarlama algoritmasının da kullanılması ile Ek – 3, Şekil 6.1 – 6.4'de bazı örneklerde görüldüğü gibi bütün görüntülerde bütün kutular başarılı bir şekilde izlenmiştir. Bu algoritmanın çalışma zamanı değerlendirmesi Bölüm 3.5' de yapılmıştır.

3.4 Sadece Doğru Uyarlama Algoritmasının Kullanılması

Bölüm 3.3' de anlatılan uygulamanın başarılı sonuçlar vermesi üzerine kutuların izlenmesi için sadece kenarların üzerinde uygulanan algoritmanın yeterli olabileceği düşünülmüştür. Bu hipotezin test edilmesi için yazılım tekrar düzenlenmiştir.

Bu algoritma, kutuların bir önceki görüntüden elde edilen köşe noktaları koordinatlarından faydalanılarak mevcut görüntüdeki konumunun aranmasından ibarettir. Algoritma Bölüm 2.4 ve Bölüm 3.3'de açıklanmıştır. Ancak burada doğru uyarlama algoritmasının öncesinde LKT algoritması kullanılmadığı için arama uzayı daha geniş tutulmuştur.

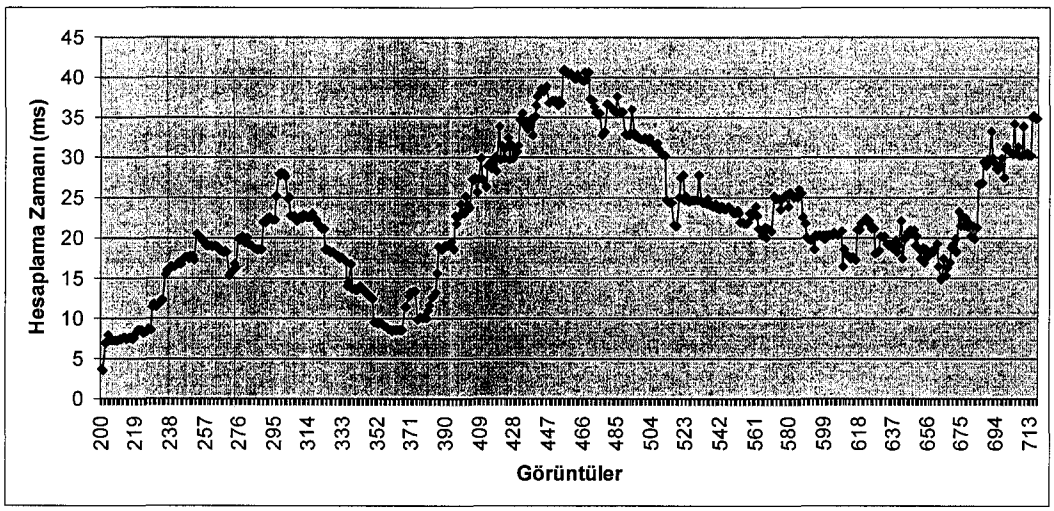
Arama uzayı; kutuların x ekseni ile yaptığı θ açısına üçer derecelik açı değişim ile -15 derece ile +15 derece arasında, kutu merkezinin x koordinatına 20 mm'lik değişim ile 0 ve 120 arasında, y koordinatına ise yine 20 mm değişim ile -60 ile 60 arasında değerler verilerek oluşturulmuştur. Bu durumda bir kutu 11 x 7 x 7 = 539 farklı konumda aranmıştır. LKT algoritmasını takriben kullanılan arama uzayında ise bir kutu sadece 36 farklı konumda aranmıştı.

Sonuç olarak Ek – 4, Şekil 7.1 – 7.4'de bazı örneklerde görüldüğü gibi bütün görüntüler boyunca bütün kutuların başarılı bir şekilde izlendiği görülmüştür. Bu algoritmanın çalışma zamanı değerlendirmesi Bölüm 3.5'de yapılmıştır.

3.5 Zaman Değerlendirmeleri

Algoritmaların zaman değerlendirmelerinin yapılmasında, Intel Pentium IV, 1.90 GHz işlemcili ve 1 Gb Ram'li bir bilgisayar kullanılmıştır.

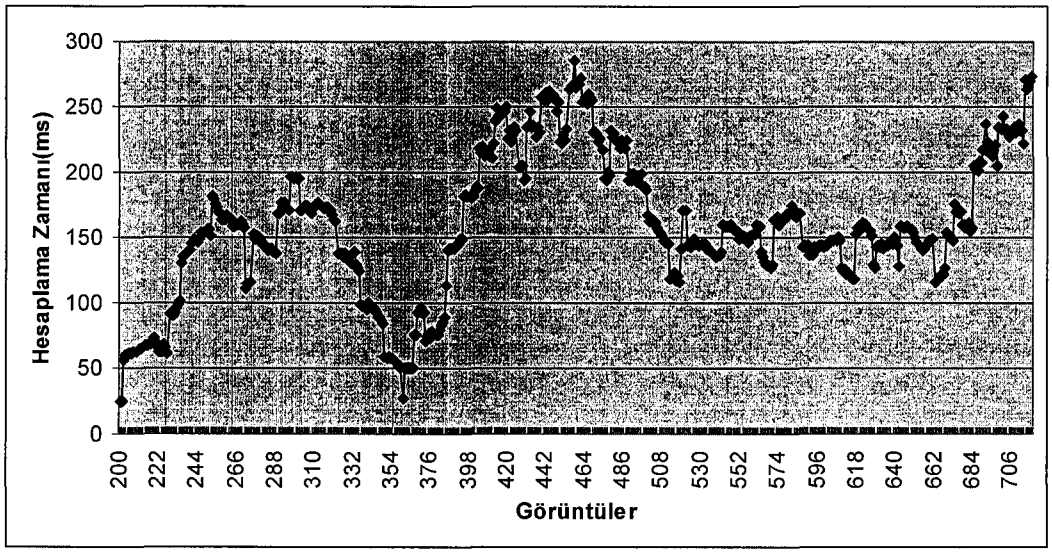
İlk olarak LKT algoritmasının diğer algoritmalarla desteklenerek izleme işleminin yapıldığı metotta, her görüntüde izleme işleminin ne kadar zaman aldığı bir zamanlayıcı fonksiyonu sayesinde hesaplanmıştır. Bu durumda ortaya çıkan grafik Şekil 3.32'de görüldüğü gibidir.



Şekil 3.32 LKT algoritması, özellik algılama ve doğru uyarlama algoritmalarının zaman değerleri

Grafikte görüldüğü gibi kutuların çok yoğun olarak geldiği yaklaşık 430-480'inci görüntüler arasında 35 ms.'dan daha fazla zaman alan hesaplamalar yapılmıştır. Diğer görüntülerde izleme algoritması 35 ms'ın altında bir süre içerisinde hesaplandığı görülmektedir. Bu algoritmada bütün görüntülerde izleme algoritmasının tuttuğu toplam süre, kutu sayısına bölündüğünde sonuç 3.7 ms. bulunmuştur. Yani bu algoritma ile bir kutunun bir görüntüde izlenmesi için 3.7 ms. geçmektedir.

İkinci olarak doğrudan büyük arama uzaylı bir doğru uyarlama algoritmasının kenar bilgileri üzerinde kullanılması ile gerçekleştirilen metodun zaman değerleri alınmıştır. Bu durumda ortaya çıkan grafik Şekil 3.33'de görüldüğü gibidir.



Şekil 3.33 Sadece doğru uyarlama algoritmasının kullanılması ile elde edilen zaman değerleri

Grafikte görüldüğü gibi izleme algoritmasının çalışma süresi 300 ms'a kadar ulaşmış ve genel olarak hesaplamalar 50 ms'nin üzerinde zaman almıştır. Bu algoritmada bütün görüntülerde izleme algoritmasının tuttuğu toplam süre, kutu sayısına bölündüğünde sonuç 34 ms bulunmuştur. Yani bu algoritma ile bir kutunun bir görüntüde izlenmesi için 34 ms geçmektedir.

4. SONUÇLAR

Bu tezde kutuların üretim bandı boyunca izlenmesi için iki farklı metot denenmiş ve ikisinde de izleme işlemeleri açısından başarıya ulaşılmıştır. Bu metotlardan birincisi, temel izleme algoritması olarak LKT algoritmasının kullanılması ve bu algoritmanın özellik algılama ile küçük arama uzaylı bir doğru uyarlama algoritmasının kenar bilgileri üzerinde kullanılması ile desteklenmesidir. İkincisi ise doğrudan büyük arama uzaylı bir doğru uyarlama algoritmasının kenar bilgileri üzerinde kullanılmasıdır.

Bu tezde yapılan çalışma için izleme işleminin başarılı olması yeterli değildir. Yapılan çalışma ile bir gerçek zamanlı bir sistem tasarlanmaya çalışılmıştır. Kameralardan bir saniye içerisinde 30 görüntü geldiği için kullanılan yazılımın gerçek zamanda çalışabilmesi için her görüntüde 1 / 30 sn = 33 ms. civarında izleme işlemi gerçekleştirmelidir.

Bu tezde kullanılan ilk metot olan LKT ve onu destekleyen algoritmalar gerçek zamanlı bir sistemde kullanılabilecek seviyede başarılı sonuçlar vermiştir. Üretim bandının çok yoğun olduğu bir süre içerisinde hesaplamalar 35 ms'ı geçmiştir. Lakin, test işlemlerinde 1.9 GHz hızında tek işlemcili bir bilgisayar kullanılmıştır. Daha hızlı ya da iki işlemcili bir bilgisayar kullanılarak bu zamanlama değerleri çok daha aşağıya çekilebilir. Bununla beraber bir kutu için harcanan zaman için 3.7 ms gibi oldukça başarılı bir değer hesaplanmıştır. Bu değer algoritmanın ortalama olarak bir görüntüde on kutuya kadar izleme işlemini gerçek zamanlı sistemlerde yapabileceğini göstermektedir. Tabiki daha önce bahsedildiği gibi daha hızlı bir işlemci ya da iki işlemci kullanarak bu değer çok daha aşağı seviyelere çekilebilir.

Bu durumda LKT ve onu destekleyen diğer algoritmalar ile yapılan izlemenin gerek işlev gerek zaman değerlendirmesi açısından tam olarak başarılı olduğu gözlemlenmiştir.

Bu tezde kullanılan ikinci metot ise Bölüm 3.3' de anlatılan doğru uyarlama algoritmasının büyük bir arama uzayı içerecek şekilde değiştirilmesiyle oluşturulan bir metottur. Bir önceki metot gibi bu metotta izleme sürecinde tam olarak başarılı olmuştur. Bu yöntem her ne kadar kutuların hepsinin başarılı bir şekilde izlenmesini sağlasa da izleme işleminin tuttuğu zaman bir gerçek zamanlı

sistemde kullanılabilecek olan zaman aralığından çok uzaktadır. Algoritmanın başarılı olabilmesi için arama uzayı büyük tutulmuştur. Bölüm 3.5' de belirtildiği gibi üretim bandında çok az kutu varken dahi izleme işlemi için 50 ms harcanmıştır. Bandın yoğun olduğu görüntülerde ise 250 ms'nin üzerinde hesaplama zamanları harcanmıştır. Bununla beraber bütün kutular üretim bandı üzerinden geçtikten sonra, toplam izleme işleminin süresi, kutu sayısına bölünerek, bir kutu için harcanan ortalama zaman 34 ms. olarak hesaplanmıştır. Bu hesaplama ile elde edilen sonuç bu metotla gerçek zamanlı olarak sadece bir kutunun izlenebileceğidir.

Sonuç olarak her iki metotla da kutuların izlenmesinde başarıya ulaşılmıştır. Lakin, sadece LKT izleme algoritması ve onu destekleyen algoritmaların kullanıldığı metot gerçek zamanlı bir sistemde kullanılabilecek zamanlama değerleri vermiştir.

KAYNAKLAR

[1] LUCAS, B.D. ve KANADE, T., *An Iterative Image Registration Technique with an Application to Stereo Vision,* International Joint Conference on Artificial Intelligence, 674-679 (1981).

[2] TOMASI, C. ve KANADE, T., *Detection and Tracking of Point Features,* Carnegie Mellon University Technical Report CMU-CS-91-132, (1991).

[3] SHI, J. ve TOMASI, C., *Good Features to Track,* IEEE Conference on Computer Vision and Pattern Recognition, 593-600 (1994).

[4] WONG, K.Y. ve SPETSAKIS, M.E., *Tracking, Segmentation and Optical Flow*, Proceedings of 16th International Conference on Vision Interface, 57-64, S1.5, Halifax, Canada (2003).

[5] BAKER, S., GROSS, R. ve MATTHEWS, I., *Lucas-Kanade 20 years on: A unifying framework* Part 4, Technical Report CMU-RI-TR-04-14, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, (2004).

[6] BERGEN, J.R., ANANDAN, P., HANNA, K.J. ve HINGORANI, R., *Hierarchical model-based motion estimation,* Proceedings of the European Conference on Computer Vision, (1992).

[7] ZIVKOVIC, Z. ve HEIJDEN, F., *Better Features to Track by Estimating the Tracking Convergence Region,*International Conference Pattern Recognition, Canada, (2002).

[8] KUCUK, Haluk., ZENGIN, R. ve KUCUK, Habib., *Görüntü Destekli Obje Tanımlama ve Endüstriyel Bir Uygulama*, Elektrik-Elektronik-Bilgisayar Mühendisliği 10. Ulusal Kongresi ve Fuarı, Istanbul (2003).

[9] ANDREWS, R.J. ve LOVELL, B. C., *Color optical flow*, Proceedings Workshop on Digital Image Computing, 135-139, Brisbane (2003)

[10] RUSSELL, S. ve NORVIG, P., *AI: A Modern Approach*, Prentice Hall, Figure **24.8**, 736, (1981).

[11] HORN, B. ve SHUNCK, B., *Determining optical flow*, Artificial Intelligence, **17**, 185–203, (1981).

[12] FENNEMA, C. ve THOMPSON, W., *Velocity determination in scenes containing several moving objects*, Computer Graphics and Image Processing **9**, 301–315, (1979).

[13] BARRON, J. L., FLEET, D.J. ve BEAUCHEMIN, S.S., *Performance of Optical Flow Techniques*, Int J Comp Vis, **12, 1**, 43-77, (1994).

[14] URAS., GIROSI, F., VERRI, A. ve TORRI, V., *A computational approach to motion perception"*, Biol. Cybern. **60**, 79-97, (1988)

[15] NAGEL, H., *On the Estimation of Optical Flow: Relations Between Different Approaches And Some New Results,* Artificial Intelligence, **33**, 299–324, (1987).

[16] ANANDAN, P., *Measuring Visual Motion from Image Sequences,* PhD dissertation, COINS TR 87-21, Univ of Massachusetts, Amherest, MA, (1987).

[17] ANANDAN, P., *A computational framework and an algoritm for the measurement of visual motion,* Int. J. Comp. Vision **2**, 283-310, (1989).

[18] SINGH, A., *An estimation-theoretic framework for image-flow computation,* Proc. IEEE ICCV, Osaka, 168-177, (1990).

[19] SINGH, A., *Optic flow computation: A unifed perspective,* IEEE Computer Society press, (1992).
[20] HEEGER, D.J., *Model for the extraction of image flow,* J. Opt. Soc. Am. A4, 1455-1471, (1987).
[21] WAXMAN, A.M., WU, J. ve BERGHOLM, F., *Convected Activation profiles and receptive fields for real time measurement of short range visual motion,* Proc. IEEE CVPR, Ann Arbor, 717-723, (1988).
[22] FLEET, D.J. ve JEPSON, A.D., *Computation of component image velocity from local phase information,* Int. J. Comp. Vision **5**, 77-104, (1990).
[23] FLEET, D.J., *Measurement of Image Velocity,* Kluwer Academic Publishers, Norwell, (1992).
[24] KEARNEY, J.K., THOMPSON, W.B. ve BOLEY D.L., *Optical flow estimation: An error analysis of gradient-based methods with local optimization,* IEEE Trans. on PAMI Proc. IEEE ICCV, Tampa, 454-459, (1987).
[25] WAXMAN, A.M. ve WOHN, K., *Contour evolution, neighbourhood deformation and global image flow: Planar surfaces in motion,* Int. J. Rob. Res. **4**, 95-108, (1985).
[26] GLAZER, F., REYNOLDS, G. ve ANANDAN, P., *Scene matching through hierarchical correlation,* Proc. IEEE CVPR, Washington, 432-441, (1983).
[27] BURT, P.J., YEN, C. ve XU, X., *Multiresolution flow-through motion analysis,* Proc. IEEE CVPR, Washington, 246-252, (1983).
[28] LITTLE, J.J. ve VERRI, A., *Analysis of differential and matching methods for optical flow,* IEEE Workshop on Visual Motion, Irvine CA, 173-180, (1989)
[29] ADELSON, E.H. ve BERGEN, J.R., *The extraction of spatiotemporal Energy in human and machine vision,* Proc IEEE Workshop on Visual Motion, Charleston, 151-156, (1986).
[30] BARMAN, H., HAGLUND, L., KNUTSSON, H., ve GRANLUND, G., *Estimation of velocity, acceleration and disparity in time sequences,* Proc. IEEE Workshop on Visual Workshop, Princeton, pp. 44-51, (1991).
[31] HAGLUND, L., *Adaptive Multidimensional Filtering,* PhD Dissertation, Dept. Electrical Engineering, Univ. of Linkoping, ISSN 0345-7524, (1992).
[32] ADELSON, E.H. ve BERGEN, J.R., *Spatiotemporal energy models for the perception of motion,* J. Opt. Soc. Am. A2, 284-299, (1985).
[33] FLEET, D.J., *Measurement of Image Velocity,* Kluwer Academic Publishers, Norwell, (1992).
[34] SIMONCELLI, E.P., *Distributed Reprentation and Analysis of Visual Motion,* PhD Dissertation, Dept. of Electrical Engineering and Computer Science, MIT, (1993).
[35] BUXTON, B., ve BUXTON, H., *Computation of optical flow from the motion of edge features in image sequences,* Image and Vision Computing **2**, 59-74, (1984).
[36] DUNCAN, J. H. ve CHOU, T. C., *Temporal Edges: The detection of motion and the computation of optical flow,* Proc. IEEE CVPR, San Diego, 374- 382, (1988).
[37] HILDRETH, E.C., *The computation of the velocity field,* Proc. Roy. Soc. London B221, 189-220, (1984).

[38] ELTOUKHY, H. ve SALAMA K., *Multiple Camera Tracking*, Project Report, (2002).

[39] STEFANO, L.D. ve VIARANI E., *Vehicle Detection and Tracking Using The Block Matching Algorithm*, Proc. of 3rd IMACS/IEEE Int'l Multiconference on Circuits, Systems, Communications and Computer" Athens, Greece, **1**, 4491-4496, (1999).

[40] LIU, H., HONG, T.H., HERMAN, M. ve CHELLAPPA, R., *Accuracy vs. Efficiency Trade-offs in Optical Flow Algorithms,* Computer Vision and Image Understanding, **72** , 271 – 286, (1998).

[41] BARALDI, P., SARTI, A., LAMBERTI, C., PRANDINI, A.ve SGALLARI F., *Evaluation of differential optical flow techniques on synthesized echo images*, IEEE Transactions on Biomedical Engineering, **43**, 259-272, (1996).

[42] BOUGUET, J.Y., *Pyramidal Implementationof the Lucas Kanade Feature Tracker*, OpenCV Documentation, Microprocessor Re-search Labs, Intel Corp., (2000).

[43] BRESENHAM, J.E., *Algorithm for Computer Control of a Digital Plotter*, IBM Systems Journal, **4**, **1**, 25-30, (1965).

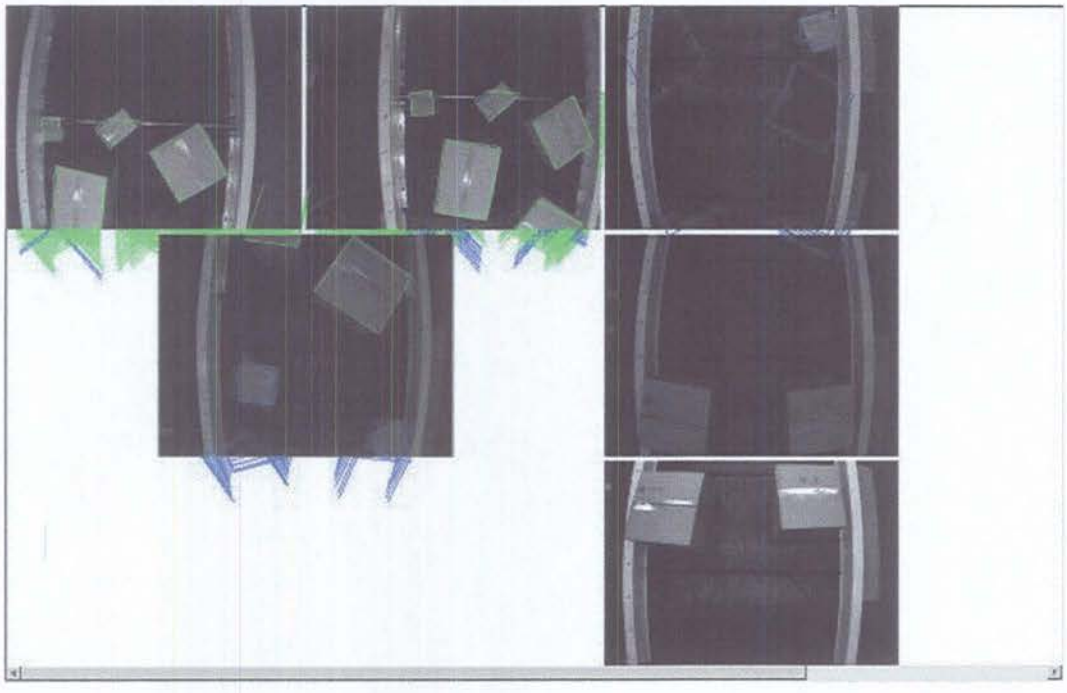
[44] PITTEWAY, M.L.V., *Algorithm for Drawing Ellipses or Hyperbolae with a Digital Plotter*, Computer J., **10**, **3**, 282-289, (1967).

[45] VAN AKEN, J.R., ve M.Novak, *Curve-Drawing Algorithms for Raster Displays*, ACM TOG, **4**, **2**, 147-169, (1985).

[46] BRESENHAM, J.E., *A linear Algorithm for Incremental Digital Display of Circular Arcs*, Communications of the ACM, **20**, **2**, 100-106, (1977).

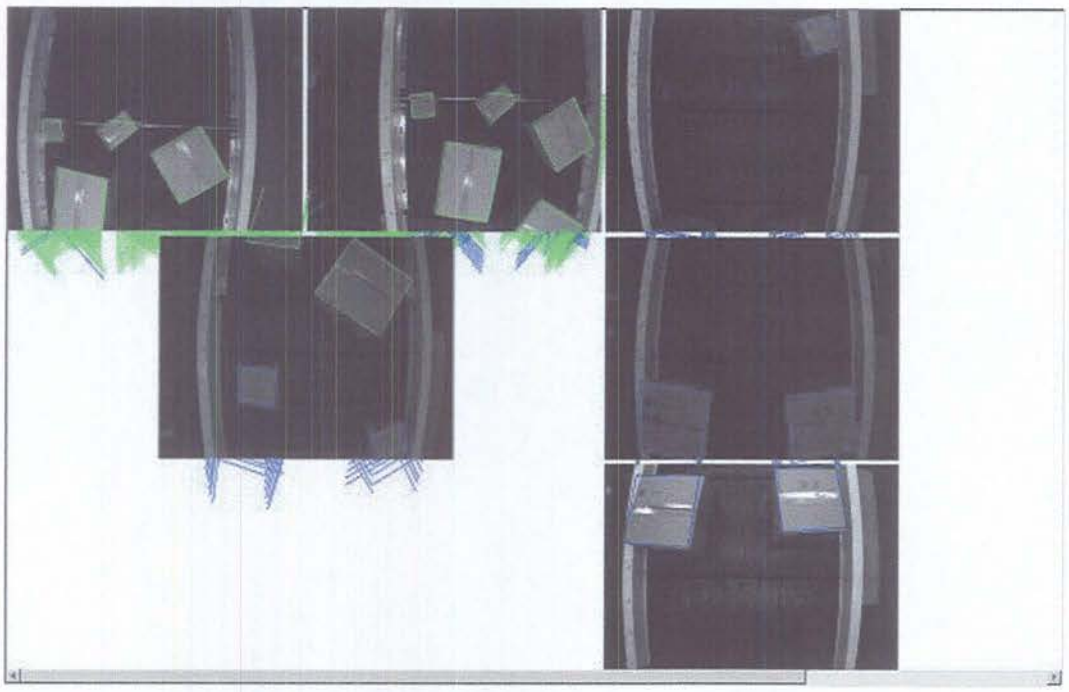
EK – 1 LKT Algoritması Ekran Çıktıları

Şekil 4.1 LKT Algoritması 220. ekran görüntüsü

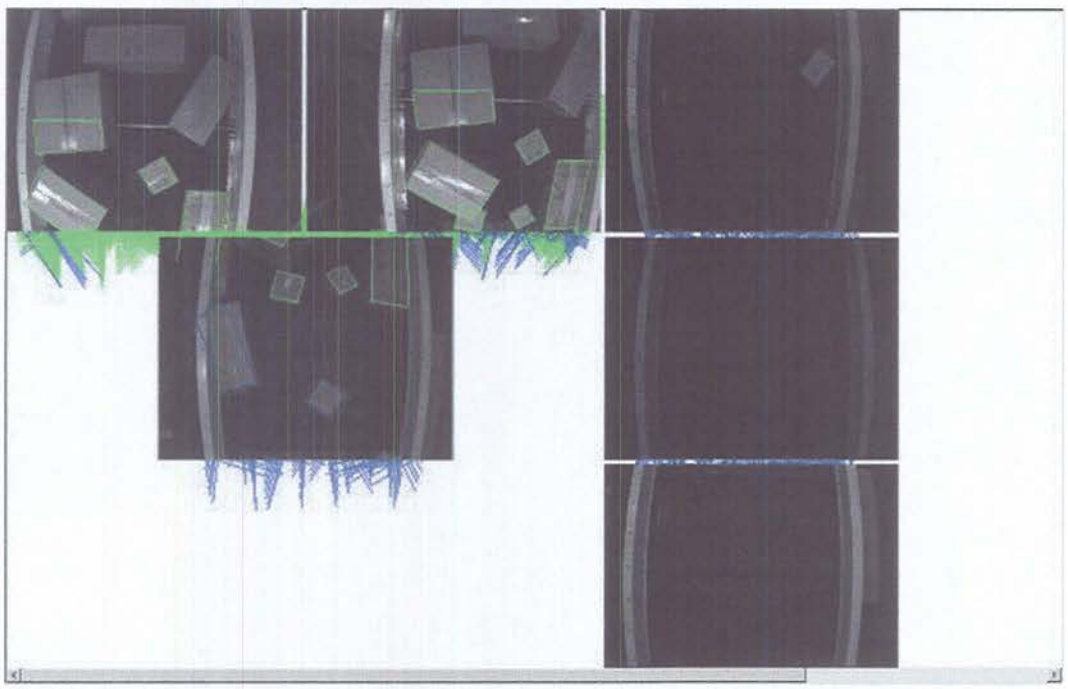


Şekil 4.2 LKT Algoritması 250. ekran görüntüsü

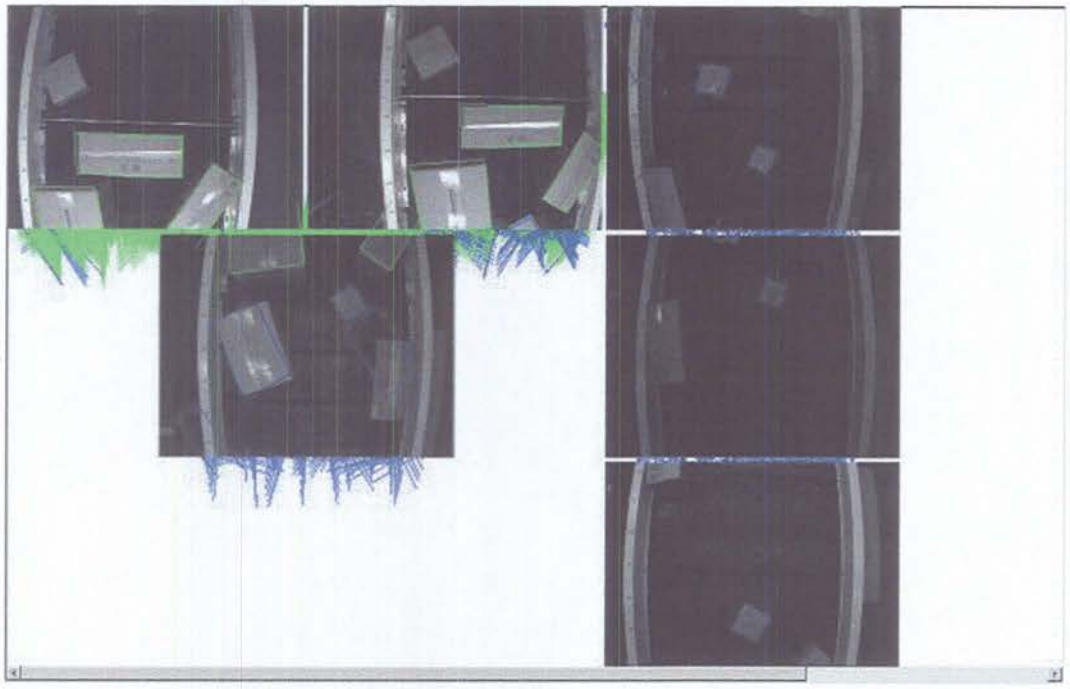
EK – 2 LKT Algoritması ve Özellik Algılama Ekran Çıktıları



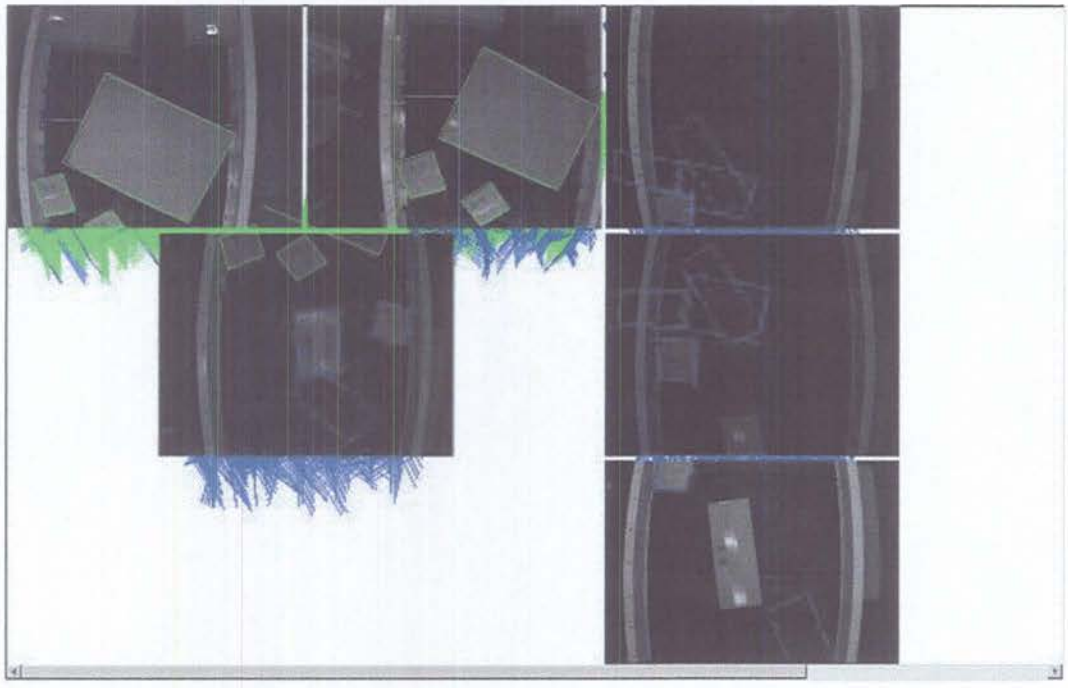
Şekil 5.1 LKT Algoritması ve Özellik Algılama 250. ekran görüntüsü



Şekil 5.2 LKT Algoritması ve Özellik Algılama 380. ekran görüntüsü

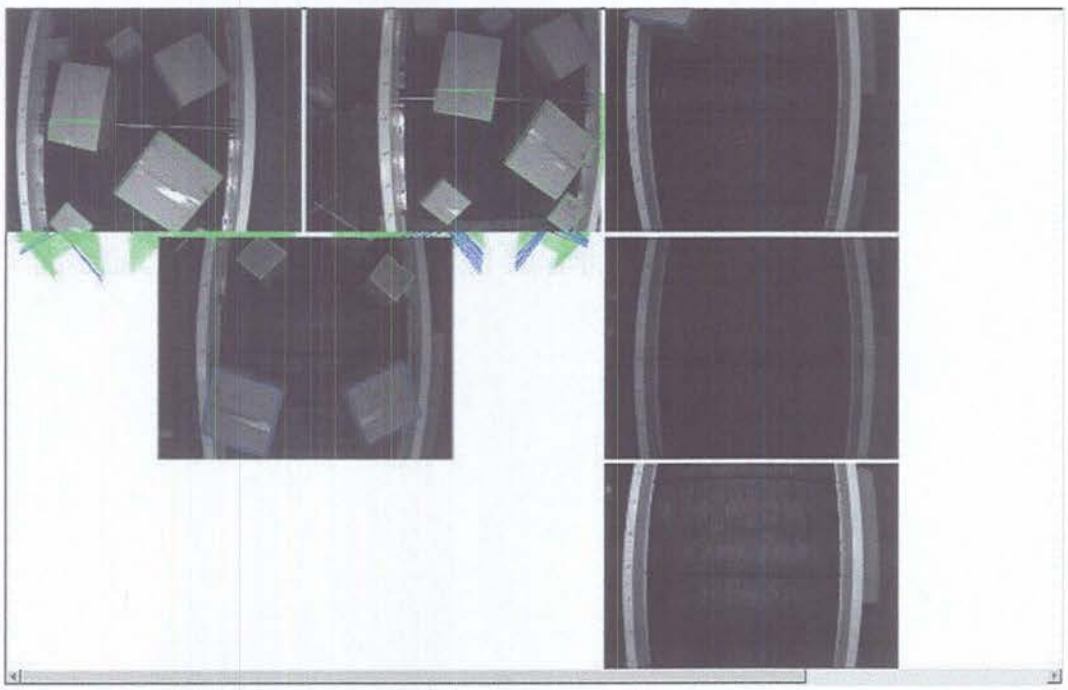


Şekil 5.3 LKT Algoritması ve Özellik Algılama 415. ekran görüntüsü

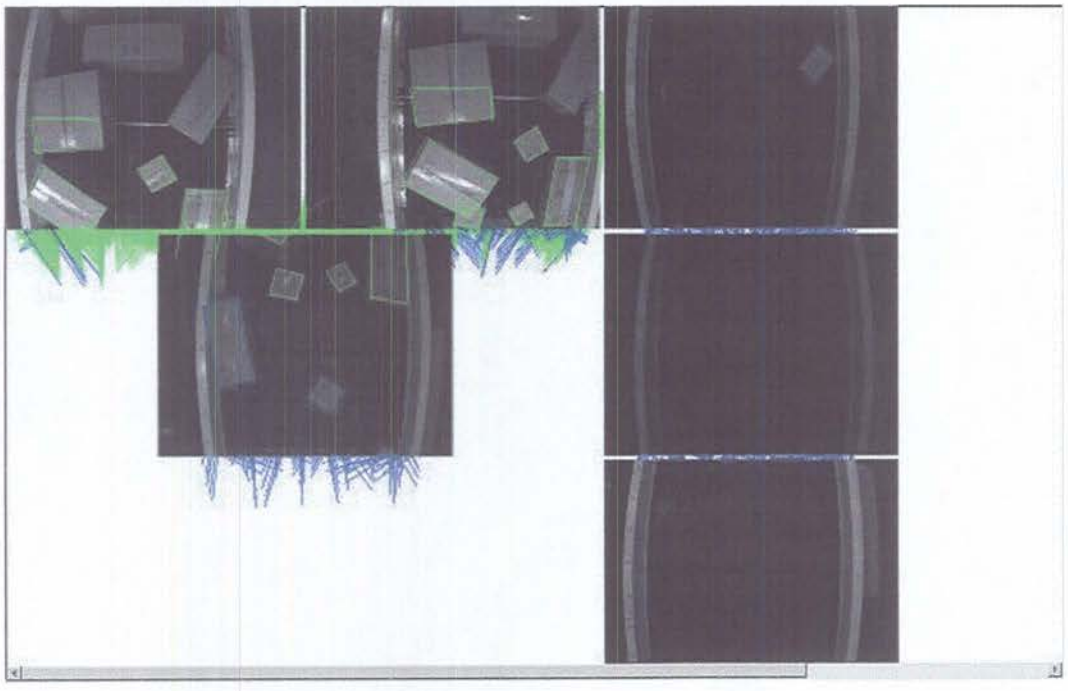


Şekil 5.4 LKT Algoritması ve Özellik Algılama 530 ekran görüntüsü

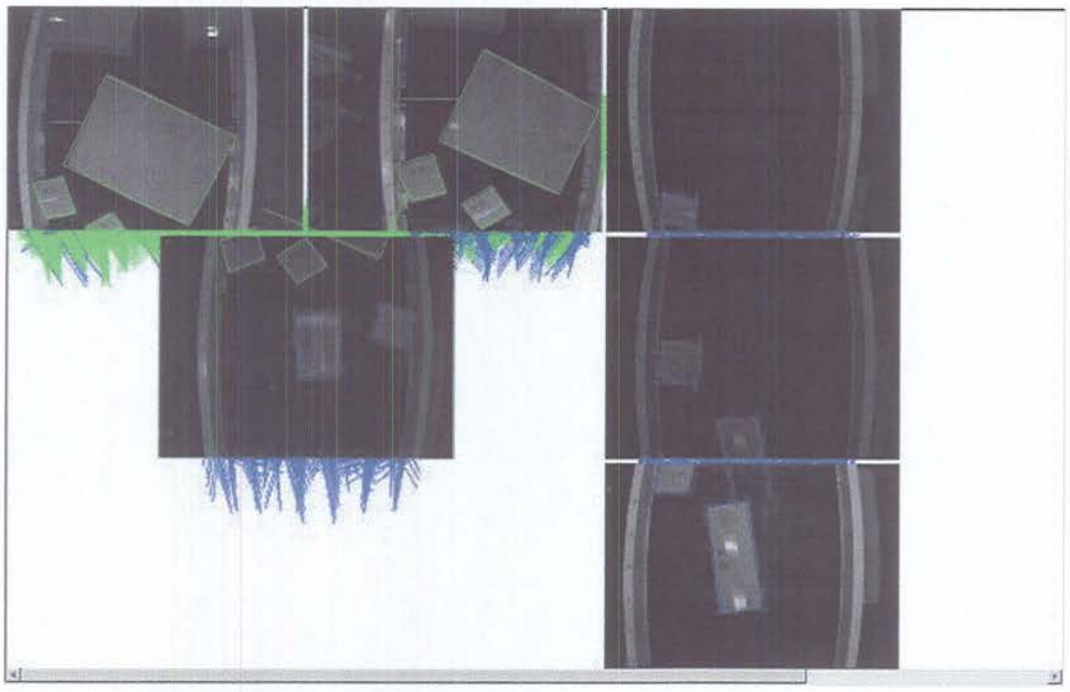
EK – 3 LKT Algoritması, Özellik Algılama ile Doğru Uyarlama Ekran Çıktıları



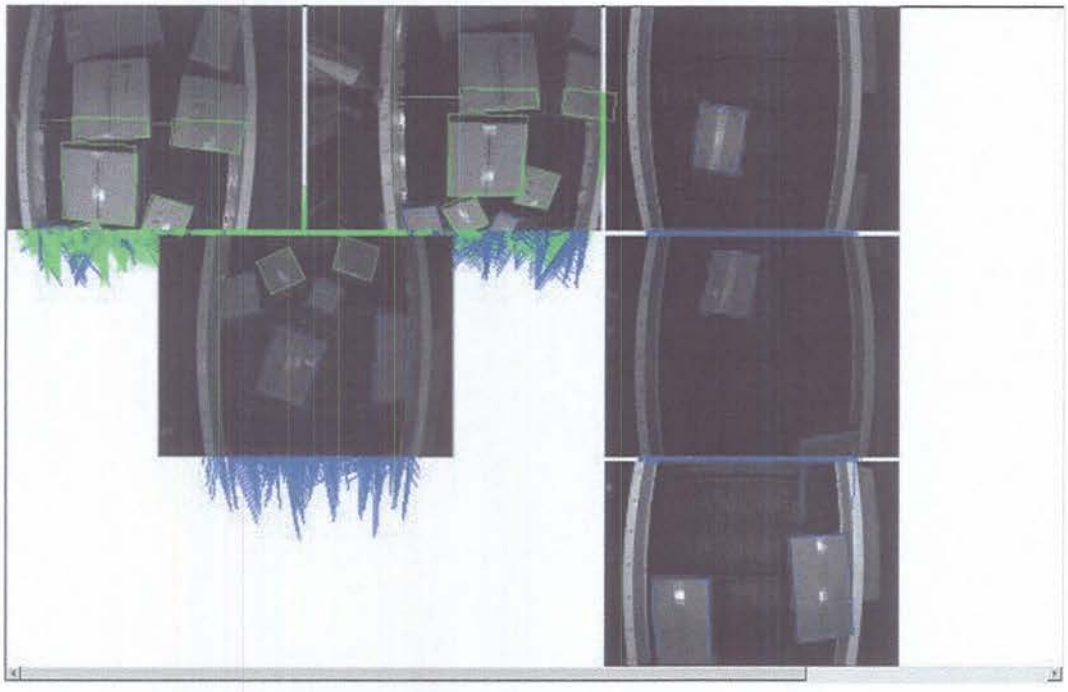
Şekil 6.1 LKT Algoritması, özellik algılama ve doğru uyarlama 220. ekran görüntüsü



Şekil 6.2 LKT Algoritması, özellik algılama ve doğru uyarlama 380. ekran görüntüsü

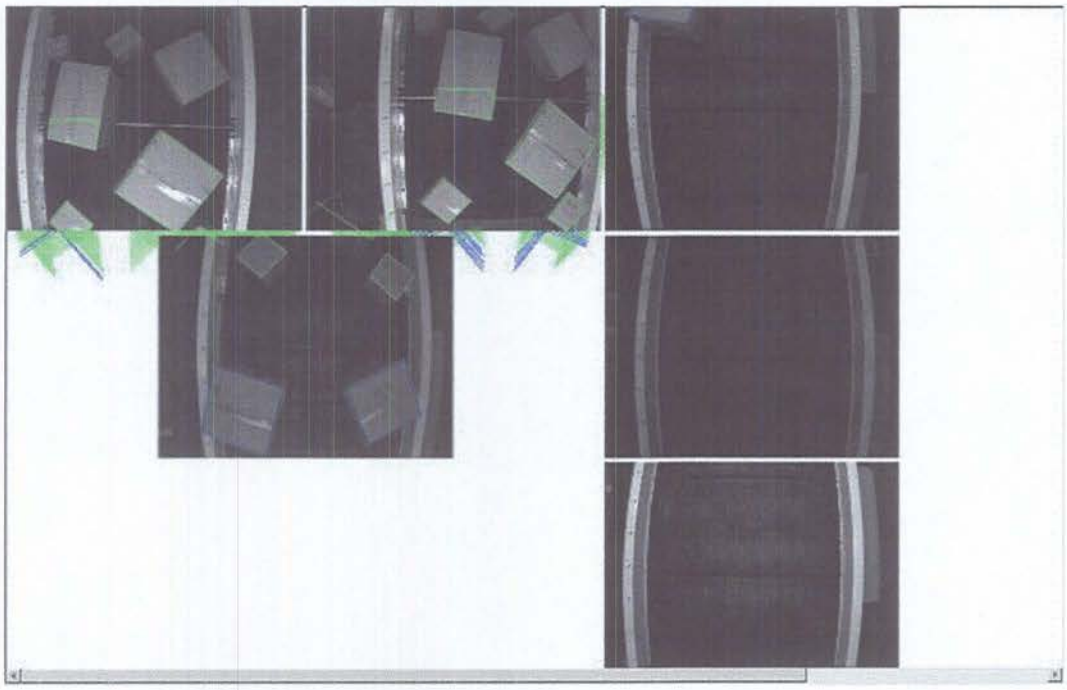


Şekil 6.3 LKT Algoritması, özellik algılama ve doğru uyarlama 530. ekran görüntüsü

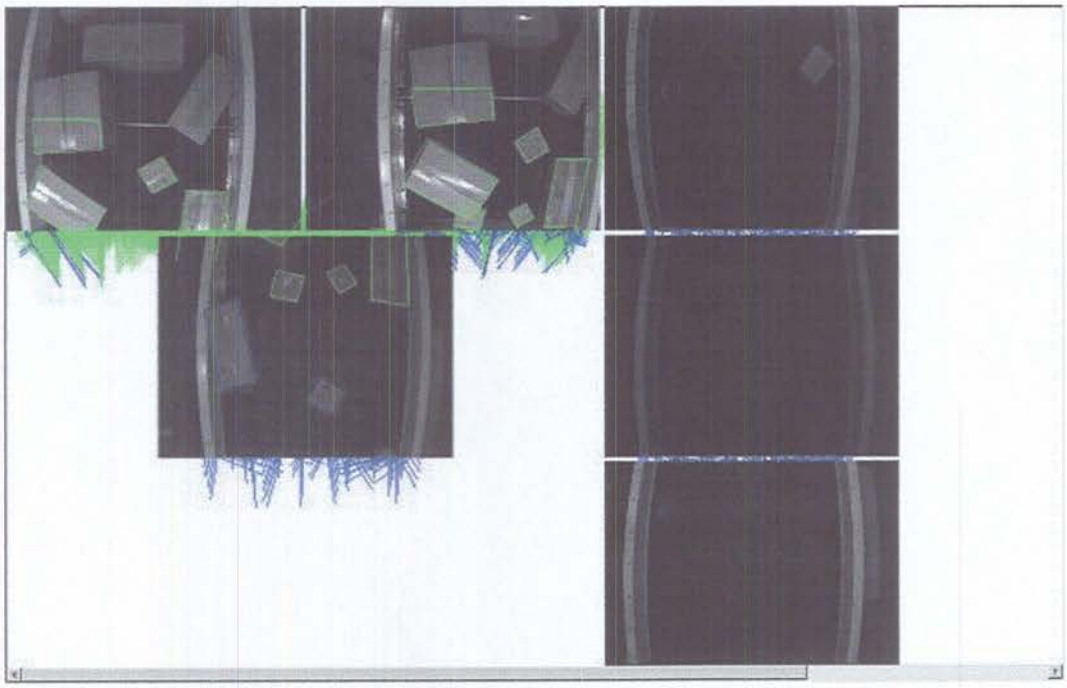


Şekil 6.4 LKT Algoritması, özellik algılama ve doğru uyarlama 683. ekran görüntüsü

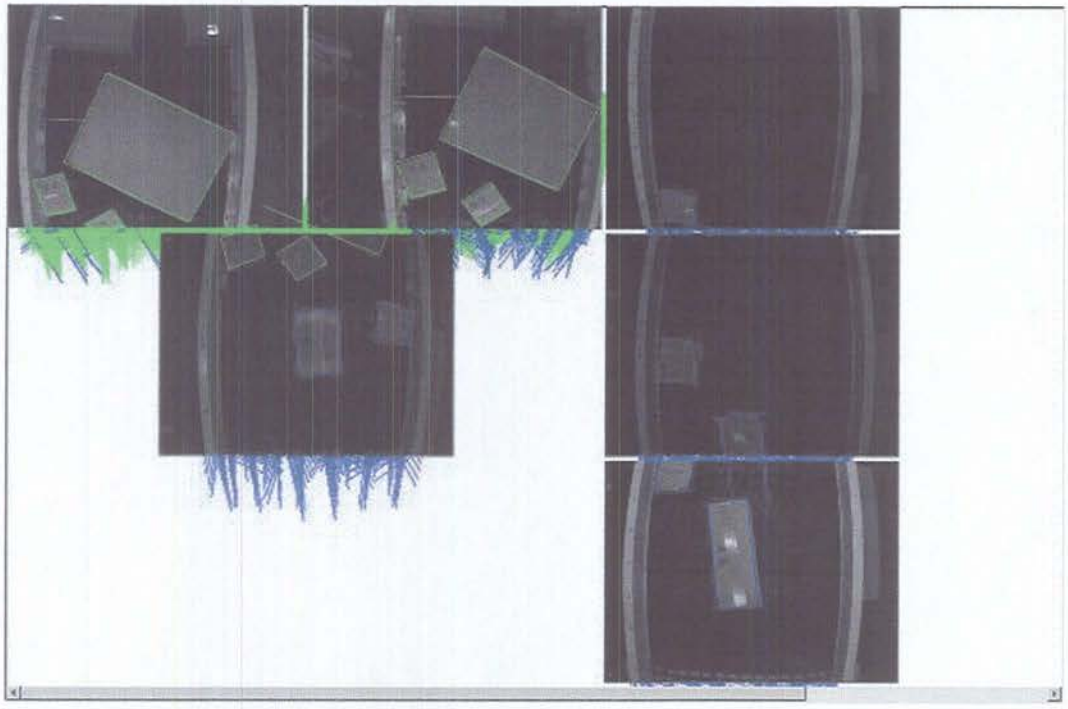
EK – 4 Sadece Doğru Uyarlama Algoritması Ekran Çıktıları



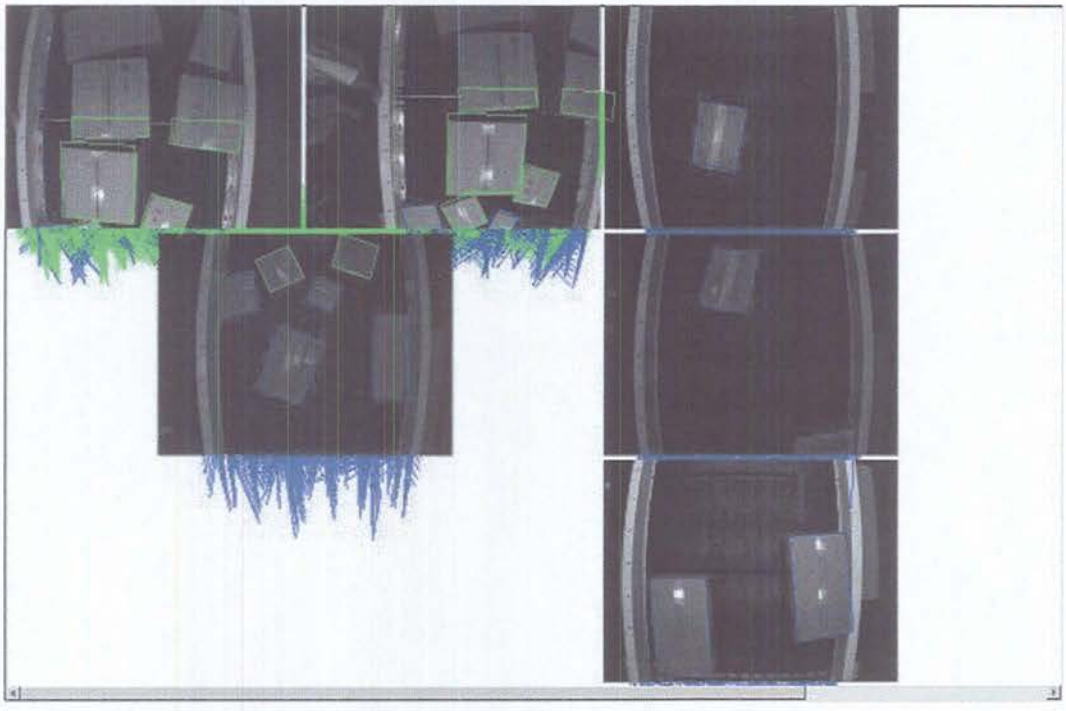
Şekil 7.1 Sadece doğru uyarlama algoritması 220. ekran görüntüsü



Şekil 7.2 Sadece doğru uyarlama algoritması 380. ekran görüntüsü



Şekil 7.3 Sadece doğru uyarlama algoritması 530. ekran görüntüsü



Şekil 7.4 Sadece doğru uyarlama algoritması 683. ekran görüntüsü