

**T.C.**  
**KOCAELİ ÜNİVERSİTESİ**  
**MÜHENDİSLİK FAKÜLTESİ**  
**ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ**



**BLACKFIN İŞLEMCİSİ İLE DSP UYGULAMALARI**

**BİTİRME TEZİ**

**Halim Cem KEFELİ**

**020207006**

**TEZ YÖNETİCİSİ : Doç. Dr. Sarp Ertürk**

**TEZ YÜRÜTÜCÜSÜ : Arş. Gör. Dr. Oğuzhan URHAN**

**KOCAELİ , HAZİRAN 2006**

## **ANAHTAR KELİMELER**

DSP, Sayısal İşaret İşleme, Blackfin, Görüntü İşleme, Video İşleme, İşaret İşleme, Görüntü Stabilizasyonu, Hareket Kestirimi, Mikroişlemci

**KEY WORDS**

DSP, Digital Signal Processing, Blackfin, Image Pprocessing, Video Processing, Signal Processing, Video Compansation, Motion Estimation, Microprocessor

## ÖZET

Bu tezde, Analog Devices™ firmasının üretmiş olduğu Blackfin BF533 işlemcisi ile görüntü stabilizasyonu yapılmıştır. Görüntü stabilizasyonu görüntülerdeki istenmeyen titreşim etkisinin bastırılmasıdır. Görüntü stabilizasyonu bütünsel hareket kestirimi ve hareket dengeleme aşamalarından oluşmaktadır. Bu tezde hareket kestirimi için bir-bir dönüşümü (one-bit transform) yaklaşımı kullanılmaktadır. Hareket dengeleme için ise basit bir alçak geçiren süzgeç kullanılmıştır.

Matlab™ ile yazılan uygulama yazılımının sonuçları VisualDSP++ ile yazılan DSP yazılımı ile karşılaştırılmış ve sonuçların aynı olduğu gözlemlenmiştir. Sistem şu an için çevrim-dışı çalışmakla birlikte çevrim-içi çalışması için çalışmalar devam etmektedir ve analog kameradan alınan görüntünün çevrim-içi stabilizasyonuna çalışılmaktadır.

## **ABSTRACT**

In this thesis, image sequence stabilization (ISS) is realized using Analog Devices™ Blackfin BF533 digital signal processor. The aim of the video stabilization is to remove unwanted camera motion while keeping intentional camera motion. Video stabilization methods consist of two stages: global motion estimation and motion correction. One-bit transform based fast motion estimation approach is utilized for motion estimation part of the ISS. A simple low-pass filter is for motion correction purpose.

Application software written in Matlab is compared to Analog Devices' VisualDSP++ integrated development environment. Experimental results show that the results of the Matlab simulation and DSP realization is similar. The system is working offline now, however we are working to make it online. We also try to cope with real time processing of image captured from an analog camera.

## TEŐEKKÜR

Bu tez yalnızca dört yılın birikimi değil çok uzun bir sürenin ve emeğin ürünü olarak öğrenim hayatımızın bir meyvesi olarak düşünölmelidir. Tüm yaşamım boyunca benden ilgilerini ve alakalarını bir saniye bile esirgemeyen anneme, babama, kardeşime, halama ve kısa bir süre önce aramızdan ayrılan çok sevdiğim babaanneme, dört yıl boyunca benim kendimi geliőtirmeme yardımcı olan değerli öğretmenlerim Doç. Dr. Sarp ERTÜRK, Dr. Oğuzhan URHAN ve Dr. M. Kemal GÜLLÜ' ye, Her zaman birlikte olduğumuz ve olacağımız sevgili arkadaşlarıma çok teşekkür ediyorum.

## İÇİNDEKİLER

ANAHTAR KELİMELER.....	1
KEY WORDS .....	2
ÖZET.....	3
ABSTRACT .....	4
TEŞEKKÜR .....	5
İÇİNDEKİLER.....	6
ŞEKİLLER DİZİNİ.....	8
TABLolar DİZİNİ .....	9
SİMGELER DİZİNİ ve KISALTMALAR .....	10
BÖLÜM 1. GİRİŞ .....	12
1.1. İşaret İşleme [1].....	12
1.2. Sayısal İşaret İşleme [1] .....	13
1.2.1. Sayısal İşaret İşleme Yöntemleri [1] .....	13
1.2.2. Sayısal İşaret İşlemenin Avantajları .....	14
1.2.3. Sayısal İşaret İşleyiciler (DSP).....	15
1.3. ADSP-BF533 EZ-LITE Kit.....	15
1.4. ADSP-BF533 EZ-KIT Lite Kit'in Kullanımı .....	17
1.4.1. Paket İçeriği.....	18
1.4.2. Varsayılan Yapılandırma.....	19
1.4.3. Yeni Oturum Açma .....	19
1.4.4. Hafıza Haritası.....	20
1.4.5. SDRAM Arayüzü .....	21
1.4.6. Flash Hafıza.....	21
1.4.7. Genel Amaçlı I/O Bağlantı Noktaları.....	22

1.4.8. Ledler ve Butonlar .....	24
1.4.9. Ses Arayüzü.....	24
1.4.10. Video Arayüzü .....	24
1.5. EZ-KIT Lite Evaluation Boardların Getirmiş Olduğu Avantajlar .....	25
1.6. Blackfin BF533 DSP işlemcisi.....	26
1.6.1 Blackfin BF533 ‘ün Özellikleri.....	26
1.7. VisualDSP++ Nedir?.....	27
1.7.1. VisualDSP++ 4.0’ın Desteklediği DSP’ler .....	28
<b>BÖLÜM 2. HAREKET KESTİRİM YÖNTEMLERİ.....</b>	<b>29</b>
2.1. Giriş.....	29
2.2. Hareket Kestirim Yöntemleri .....	30
2.2.1. Tam Arama.....	31
2.2.2. Üç Adımlı Arama .....	33
2.2.3. Sıra Düzensel Hareket Kestirimi .....	35
2.3. Bölge Eşleme-Uyumlama Ölçütleri .....	37
<b>BÖLÜM 3. GÖRÜNTÜ STABİLİZASYONU .....</b>	<b>40</b>
3.1. Giriş.....	40
3.2. Önerilen Görüntü Stabilizasyonu Yöntemi .....	41
3.2.1. Bir Bit Dönüşümü (One Bit Transform) .....	42
3.2.2. Blok Eşleme İşlemleri .....	44
3.2.3. Hareket Vektörlerinin Bulunması.....	45
3.2.4. Toplamsal Hareket Vektörlerinin Bulunması ve Süzgeçleme işlemleri.....	46
3.2.4. Farksal konum Vektörlerinin Bulunması ve Öteleme İşlemleri.....	47
<b>BÖLÜM 4. SONUÇLAR VE ÖNERİLER.....</b>	<b>50</b>
<b>EK-1 .....</b>	<b>51</b>
<b>KAYNAKLAR DİZİNİ .....</b>	<b>61</b>
<b>ÖZGEÇMİŞ .....</b>	<b>62</b>



## ŞEKİLLER DİZİNİ

Şekil 1.1	: EZ-LITE Kit için çevre birimler ve blok diyagram .....	16
Şekil 1.2	: EZ-LITE Kit için varsayılan yapılandırma .....	19
Şekil 1.3	: Blackfin BF531-BF532-BF533 işlemcileri için blok diagram .....	26
Şekil 2.1	: Hareket vektörünün gösterimi.....	29
Şekil 2.2	: Hareket gösterimi için çeşitli yöntemler. ....	30
Şekil 2.3	: Tam aramada işlemi için temsili gösterim .....	32
Şekil 2.4	: Üç adımlı arama yöntemi için temsili gösterim .....	34
Şekil 2.5	: Sıra-düzensel imge gösterimi.....	35
Şekil 2.6	: HME yönteminde toplam yer değiştirmenin bulunması.....	36
Şekil 2.7	: 2 seviyeli sıra-düzensel hareket kestirimi .....	37
Şekil 3.1	: Önerilen görüntü stabilizasyonu için blok işleyiş yapısı .....	42
Şekil 3.2	: Bir bit dönüşümü için kullanılan yapı elemanı matrisi .....	43
Şekil 3.3	: “Bike” görüntü dizisi için 40. imge çerçevesi .....	44
Şekil 3.4	: Blok eşleme işlemleri için sembolik işleyişi.....	45
Şekil 3.5	: “Bike” görüntü dizisinin 22. çerçevesi için elde edilen hareket vektörü .....	45
Şekil 3.6	: “Bike” görüntü dizisi için elde edilen hareket vektörleri.....	46
Şekil 3.7	: “Bike” görüntü dizisi için elde edilen toplamsal konum vektörleri ve filtrelenmiş toplamsal konum vektörleri.....	47
Şekil 3.8	: “Bike” görüntü dizisi için elde edilen hareket vektörleri.....	47
Şekil 3.9	: “Bike” görüntü dizisi için elde edilen stabilizasyon sonuçları .....	48

## TABLolar DİZİNİ

Tablo 1.1.	: ADSP-BF533 EZ-KIT Lite kit için harici hafıza haritası.....	20
Tablo 1.2.	: ADSP-BF533 EZ-KIT Lite kit için dahili hafıza haritası.....	20
Tablo 1.3.	: SDRAM için optimum yapılandırmalar .....	21
Tablo 1.4.	: Asenkron hafıza kontrol saklayıcılarının ayarlanması.....	21
Tablo 1.5.	: Flash hafıza haritası .....	22
Tablo 1.6.	: Flash A için yapılandırma saklayıcıları - PORT A,B .....	22
Tablo 1.7.	: Flash B için yapılandırma saklayıcıları - PORT A,B .....	23
Tablo 1.8.	: Flash A Port A Control .....	23
Tablo 1.9.	: Flash A Port B Control .....	23
Tablo 2.1.	: 5 seviyeli HEM için tipik parametreler.....	37

## SİMGELER DİZİNİ ve KISALTMALAR

i/o	Giriş/çıkış
DSP	Digital Signal Processing – Sayısal İşaret İşleme
DSP	Digital Signal Processor – Sayısal İşaret İşleyici
FFT	Fast Fourier Transform – Hızlı Fourier Dönüşümü
ASIC	Application Specific Integrated – Uygulamaya Özgü Bütünleşik Yapı
Osc.	Osilatör
RTC	Real Time Clock – Gerçek Zaman Saati
Pixel	Görüntü elemanı
FS	Full Search – Tam Arama
LS	Logarithmic Search – Logaritmik Arama
TSS	Three Step Search – Üç Adımlı Arama
DS	Diamond Search – Baklava Dilimli Biçimli Arama
CS	Cross Search – Çapraz Arama
HSM	Hierarchical Search Method – Sıra Düzensel Arama Yöntemi
BMC	Block Matching Criteria – Blok Eşleme Kriteri
CC	Cross Correlation – Çapraz Korelasyon
MSE	Mean Square Error – Ortalama Kareysel Hata
MAD	Mean Absolute Difference – Ortalama Mutlak Fark
GB	Global Based – Blok Temelli
BB	Block Based – Blok Temelli
RWA	Remote Window Area – Uzak Pencere Bölgesi
LWA	Local Window Area – Yerel Pencere Bölgesi
RGB	Red,Green,Blue – Kırmızı,Yeşil,Mavi
TDMA	Time Division Multiplexed Access – Zaman Bölmeli Çoklu Erişim
TWI	Two-Wire Interface - Çift Yollu Arayüz
PPI	Parallel Peripheral Interface – Paralel Birim Arayüzü
SCL	Serial CLock – Seri Saat Darbesi
SDAT	Serial DATA – Seri Veri
DMA	Direct Memory Access - Direk Hafıza Erişimi
SS	Step Size – Adım Boyutu

AGS	Alçak geçiren Süzgeç
YGS	Yüksek Geçiren Süzgeç
BGS	Bant Geçiren Süzgeç
DS	Down Sampling – Alt Örnekleme
US	Up Sampling – Yukarı Örnekleme
GDS	Görüntü Düzeltme Sistemi
LW	Local Window – Yerel Pencere
RW	Remote Window – Uzak Pencere
OBT	One Bit Transform – Bir Bit Dönüşümü
NNMP	Number of Non-Matching Pixels
EX-OR	Exclusive OR
$d_x$	X-eksenindeki hareket vektörü
$d_y$	Y-eksenindeki hareket vektörü
$d$	Hareket vektörü
$t$	Süre

## **BÖLÜM 1. GİRİŞ**

### **1.1. İşaret İşleme [1]**

İşaret ve sistemlerin teori ve uygulamaları, elektronik ve haberleşme teknolojileri alanında önemli bir yer tutmaktadır. İşaretler fiziksel bir durum hakkında bilgi taşıyan, bir veya birden fazla değişkene bağlı fonksiyonlar olarak tanımlanmaktadır. Sistemler ise istenilen niteliklerde işaret üretmek veya verilen giriş işaretine göre belirli çıkışlar üreten düzeneklerdir. Bir elektrik devresinde zamana bağlı olarak değişen akım ve gerilim değerleri işaretlere tipik bir örnek teşkil etmektedir. Elektrik devresinin kendisi ise bir sistem örneğidir. İşaret işleme temel olarak, sistem analiz ve sentezi için, sistemin işaretlerde yaptığı değişimlerin bulunması veya işaretle istenilen değişiklikleri yerine getirecek bir sistemin tasarlanması işlemidir.

Sistem analizi, var olan bir sistemin değişik girişlere ne şekilde yanıt verdiğinin bulunmasıdır. Bu sayede sistemin olası tüm girişler altında nasıl çalışacağı tespit edilebilmektedir. Örneğin bir haberleşme hattı ele alındığında, haberleşme hattından gerçekleştirilecek ilettime nasıl bir yanıt verdiğinin ve hattın iletilen işaretle bozulmalara neden olup olmadığının bulunması iletim kapasitesinin tespiti için önemlidir ve iletim kalitesinin artırılması için ek önlemlerin gerekliliği hakkında bilgi vermektedir.

Sistem tasarımı ise belirli bir işlevi yerine getirecek bir sistemin geliştirilmesidir. Örneğin videolarda, kameranın titreşim etkisinden kaynaklanan ve gözü rahatsız edici etkilerin giderilmesini sağlayacak ya da en aza indirebilecek bir sistem tasarlanabilmektedir. Sistem yapıları ve çalışma prensipleri doğal olarak yerine getirilmesi düşünülen işleme göre çeşitlilik göstermektedir. Örneğin, gelecek işaret değerlerinin tahmin edebilen bir sistem tasarımı (Örneğin bir ekonomik kestirim sistemi), herhangi bir şekilde bozulmuş olan bir işaretin yeniden onarılması (Örneğin arşiv filmlerindeki kir etkisinin giderilmesi), birden fazla işaret içerisinde belirli bir işaretin ayrıştırılması (Örneğin filtreleme) veya belirli girişlere göre belirli işlevlerin gerçekleştirilmesi (örneğin kontrol ve yapay sinir ağı uygulamaları) mümkündür.

## 1.2. Sayısal İşaret İşleme [1]

Sayısal işaret işleme işaretlerin analog biçimli olarak değil de sayısal şekle çevrildikten sonra sayısal şekliyle işlenmesidir. Sayısal işaret işleme için yaygın olarak DSP (Digital Signal Processing) kullanılmaktadır. Bazı durumlarda işaretlerin zaten sayısal formda (Bilgisayar verileri, CD, DVD) bulunmasının mümkün olmasıyla birlikte analog bir işaretin sayısal işaret işleme yöntemleri ile işlenebilmesi için ilk önce analog işaret bir analog/sayısal çevirici ile sayısal şekle çevrilmesi gerekmektedir. Sayısal olarak işlendikten sonra hedefte analog bir biçimde kullanılacak ise bir sayısal/analog çevirici yardımıyla analog şekle çevrilmelidir. Sayısal işaret işleme haberleşme, otomotiv, Endüstriyel, medikal, savunma sanayi vb. gibi birçok alanda aktif bir şekilde kullanılmaktadır.

Sayısal işaret işleme, temeli ve teorisi itibariyle yüksek oranda matematiksel altyapıya dayanmaktadır. Zaman uzayında ve frekans uzayında işaret ve sistemlerin analizi ve istenilen özelliklerin yerine getirilebilmesi için yapılması gereken işlemler çoğunlukla matematiksel tabanda geliştirilen yöntemler kullanılarak gerçekleştirilmektedir. Bu nedenle işaret işleme uygulamaları çoğunlukla yüksek bir hesap gücü gerektirmektedir.

Sayısal işaret işleme, Hızlı Fourier Dönüşümü (Fast Fourier Transform, FFT) gibi yüksek miktarda işlem gerektiren uygulama ve yöntemlerin bilgisayarlar ile gerçekleştirilmeye başladığı 1960'lı yıllarda ortaya çıkmıştır. 1970'li yılların sonu ve 1980'li yılların başında mikroşlemcilerin doğuşu ile birlikte DSP yöntemleri daha yaygın uygulama alanları bulmaya başlamıştır. Intel™ x86 ailesi gibi genel amaçlı mikroşlemcilerin yüksek miktarda hesap gücü gerektiren DSP uygulama ihtiyaçlarına cevap verememeye başlaması nedeniyle 1980'li yıllardan itibaren Texas Instruments™, Analog Devices™ ve Motorola™ gibi büyük üreticiler yapısı özel olarak sayısal işaret işleme uygulamalarının ihtiyaç duyduğu işlemleri yerine getirecek sayısal işaret işleyiciler (Digital Signal Processor, DSP) geliştirmişlerdir.

### 1.2.1. Sayısal İşaret İşleme Yöntemleri [1]

Sayısal işaret işleme yöntemlerini kullanmak isteyen bir uygulama geliştirici için sembolik DSP analizi, gerçek zamanlı DSP ve gerçek zamanlı olmayan DSP olmak üzere üç tane seçenek bulunmaktadır.

Sembolik DSP: Sembolik DSP analizi, kâğıt ve kalem ile denklemlerin kurulup, işaretlerin tanımlanması ile birlikte bir sistemin çalışmasının kâğıt üzerinde incelenmesi ya da bir sistemin kâğıt üzerinde tasarlanmasıdır. Sembolik DSP'nin avantajı olarak yalnızca kâğıt ve kalem gibi bir teçhizata gerek duyması gösterilmekle birlikte gerçek dünya işaretlerinin analizini gerçekleştirilememesi yöntemin önemli bir eksiği olarak göze çarpmaktadır.

Gerçek Zamanlı DSP : Gerçek zamanlı DSP çoğunlukla donanım kullanılarak gerçekleştirilmektedir ve Uygulamaya özgü entegre tasarımı(ASIC) ile programlanabilir DSP kullanımı olmak üzere yöntem tarafından iki farklı yol uygulama geliştiricinin hizmetine sunulmaktadır. ASIC yapılar, programlanabilir DSP'lere göre çok daha hızlı işlem gerçekleştirilmesini sağlamakla birlikte tasarımlarının zaman alması ve üretim maliyetlerinin çok yüksek olması nedeniyle çok sık tercih edilmemektedirler. Bununla birlikte programlanabilir DSP'ler ASIC yapılara göre daha az çevre birime sahiptirler ve bazı işlevler donanım yetersizliği nedeniyle gerçekleştirilemeyebilmektedir.

Gerçek Zamanlı Olmayan DSP : Gerçek zamanlı olmayan sayısal işaret işleme, sayısal veriler bir sayısal bilgi depolayıcıya saklanmakta ve bilgisayar programı yardımıyla veriler okunarak işlendikten sonra tekrar aynı ortama geri saklanabilmektedir. Bu yöntemde geliştirilen uygulamaların derlenmesi ve çalıştırılması çok kolay olmakla birlikte işlemler için harcanılan süre oldukça fazladır.

### 1.2.2. Sayısal İşaret İşlemenin Avantajları

- ❖ Sayısal işaretler bozulmalardan daha az etkilenirler : Hata algılama ve hata düzeltme kodlamaları ile işaretlerin bozulması en asgari seviyeye çekilmektedir.
- ❖ Sayısal yöntemlerin kesinliği daha yüksektir : Analog sistemlerde kullanılan devre elemanlarının (Direnç, kapasitör vb.) belirli bir tolerans değeri vardır ve ortama göre de değişebilmektedirler oysa ki sayısal işaret işleme sistemleri ortamın durumundan bağımsızdır.
- ❖ Sayısal işaretlerin saklanması daha kolaydır : Analog veriye erişim sayısal veriye erişimden daha zordur.
- ❖ Sayısal yöntemler daha esnektir : Programlanabilir DSP'lerin kullanımı sayesinde geliştirilen sayısal işaret işleyen bir sistemin güncellenmesi programının

değiştirilmesinden ibarettir. Analog işaret işleyiciler ise devre elemanlarından oluştuğu için güncellenmesi oldukça güçtür.

- ❖ Sayısal sistemler daha karardır : Sayısal işaret işleme yöntemlerinin karallığı ortamdan bağımsızdır.
- ❖ Bazı işlevler yalnızca sayısal yöntemlerle sağlanabilmektedir : Şifreleme ve sıkıştırma işlevleri yalnızca sayısal işaret işleme yöntemleri için geçerlidir.

### 1.2.3. Sayısal İşaret İşleyiciler (DSP)

DSP kısaltması yerine göre hem işaretlerin sayısal yöntemlerle işlendiğini belirten sayısal işaret işleme hem de işaretleri sayısal yöntemlerle işlemek için özel olarak tasarlanan mikroişlemciler verilen ad olan sayısal işaret işleyiciler kavramları yerine kullanılabilir. Genel amaçlı mikroişlemcilerde olduğu gibi DSP'lerin de kendine özgü bir kod tablosu mevcuttur ve DSP'ler istenen işlevi yerine getirmek üzere programlanabilir. Sayısal işaret işleyiciler matematiksel işlemleri çok hızlı bir biçimde gerçekleştirmek üzere üretilmiştir. Günümüz teknolojisi, saniyede milyonlarca işlemi gerçekleştiren mikroişlemcileri kullanımımız için hizmetimize sunmuştur ve daha hızlı çalışan işlemcilerin geliştirilmesi için çalışmalar sürmektedir.

### 1.3. ADSP-BF533 EZ-LITE Kit

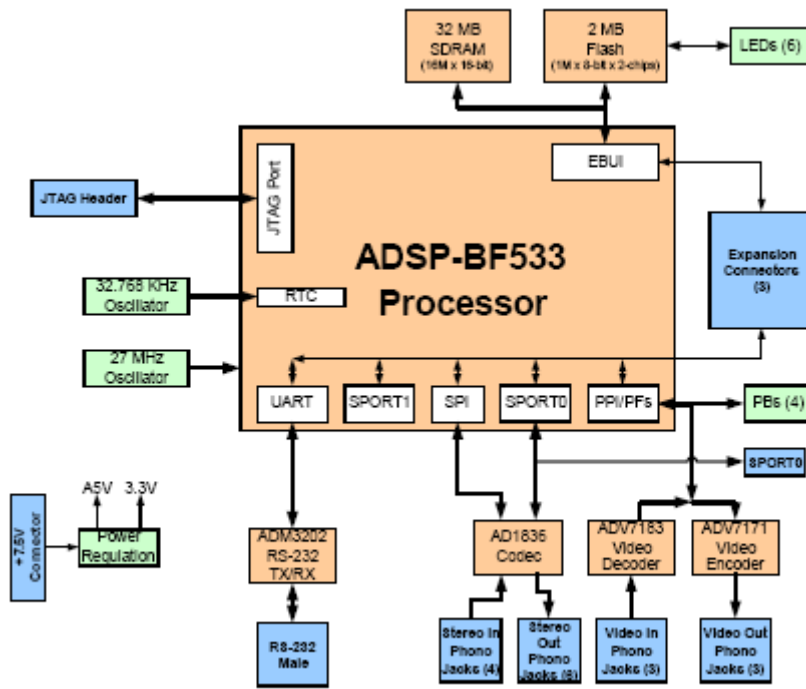
EZ-LITE Kit uygulama geliştirme platformu, mimarisi sayısal işaret işleme için özel olarak yapılandırılan ADSP-BF533 blackfin işlemcisinin yeteneklerinin ve verimliliğinin kullanıcı tarafından kolayca test edilebilmesi için tasarlanmıştır. VisualDSP++<sup>®</sup> yazılım geliştirme ortamı ile kolayca programlanabilen bu platform hem gelişmiş bazı sayısal işaret işleme uygulamalarına olanak tanırken diğer yandan da kullanıcıya DSP işlemcili temel bir yapının nasıl oluşturulacağı konusunda bilgi vermektedir. VisualDSP++<sup>®</sup> yazılım geliştirme ortamı beraberinde getirdiği bazı özellikler sayesinde kullanıcıya bazı olanaklar sunmaktadır,

- ❖ Yükleme, çalıştırma, adım modu, reset, ve duraklatma noktaları,
- ❖ Veri ve program bellek için okuma, yazma,
- ❖ Çekirdek ve çevresel saklayıcılar için okuma ve yazma,
- ❖ Hafıza gösterimi



- ❖ C++, c ve ADSP-BF533 assembler dilleri ile yazılan derleme, oluşturma ve makine kodu uygulamaları.

ASDP-BF533 EZ-LITE kit e erişebilmek için kart üzerinde bulunan doğrudan PC bağlantılı bir USB arayüzü kullanılmaktadır. USB arayüzü ADSP-BF533 işlemcisine ve uygulama geliştirme platformu çevre birimlerine sınırsız bir erişim imkanı sunmaktadır. Bununla birlikte JTAG terminali hedef donanım ve PC arasında daha hızlı bir haberleşme bağlantısına olanak sağlamaktadır. ASDP-BF533 EZ-LITE kit için çevre birimler ve kit için blok diyagram Şekil 1.1’de verilmektedir.



Şekil 1.1 : EZ-LITE Kit için çevre birimler ve blok diyagram

ASDP-BF533 AZ-LITE kit uygulama geliştirme platformunun özelliklerini şu şekilde sıralayabiliriz.

- ❖ Analog Devices ADSP-BF533 işlemci
  - 756 Mhz çalışma performansı
  - 160 pinli Mini-BGA kılıf
  - 27 Mhz dahili osilatör

- ❖ Senkron dinamik rastgele erişim belleği – Synchronous Dynamic Random Access Memory (SDRAM)
  - MT48LC32M16 – 64MB (32M x 16 bits)
- ❖ Flash bellek
  - 2 MB (512K x 16 x 2 mikroçip)
- ❖ Analog ses arayüzü
  - AD1836 – Analog Devices 96 kHz Ses kodek
  - 4 giriş RCA phono jak (2 kanal)
  - 6 çıkış RCA phono jak (3 kanal)
- ❖ Analog video arayüzü
  - ADV7183 video dekode w/ 3 giriş RCA phono jak
  - ADV7171 video dekode w/ 3 çıkış RCA phono jak
- ❖ Led
  - 10 LED : 1 güç(yeşil), 1 kart reset(kırmızı), 1 USB(kırmızı), 6genel amaçlı(değişken), 1 USB denetleyici(değişken)
- ❖ Buton
  - Lojik amaçlı 5 bas-çek türü buton : 1 reset
  - 4 programlanabilir
- ❖ Genişleme arayüzü
  - PPI, SPI, EBIU, Zamanlayıcı 2-0, UART, plogramlanabilir bayrak, SPORT0, SPORT1
- ❖ Diğer özellikler
  - JTAG ICE 14-pin

#### **1.4. ADSP-BF533 EZ-KIT Lite Kit'in Kullanımı**

Bu bölümde ADSP-BF533 EZ-KIT Lite ile program geliştirme üzerine belirli bilgiler verilecektir.

- ❖ 'Paket İçeriği' başlığı altında ADSP-BF533 EZ-KIT Lite kit ile birlikte program geliştiricilerin hizmetine sunulan donanımsal yapıların niteliği verilmektedir.
- ❖ 'Varsayılan Yapılandırma' başlığı altında ADSP-BF533 EZ-KIT Lite kit'in varsayılan yapılandırma(jumper konumları, vb.) verilmektedir.

- ❖ ‘Yeni Oturum Açma’ başlığı altında ADSP-BF533 EZ-KIT Lite kit ile nasıl yeni bir oturum açılacağı ve nasıl programlama yapılacağı anlatılmaktadır.
- ❖ ‘Hafıza Haritası’ başlıklı bölümde ADSP-BF533 EZ-KIT Lite in hafıza alanlarının değerleri ve durumu hakkında nasıl bilgi sahibi olunacağı anlatılmaktadır.
- ❖ ‘SDRAM Arayüzü’ bölümünde kit üzerindeki dahili SDRAM in nasıl yapılandırılacağı konusunda bilgilendirme yapılacaktır.
- ❖ ‘Flash Hafıza’ bölümünde ADSP-BF533 EZ-KIT Lite kit üzerinde bulunan dahili flashların nasıl yapılandırılacağı anlatılmaktadır.
- ❖ ‘Genel Amaçlı I/O Bağlantı Noktaları’ EZ-LITE kit üzerindeki flash bağlantı noktaları konusunda bilgilendirme yapılacaktır.
- ❖ ‘Ledler ve Butonlar’ bölümünde kit üzerindeki butonlar, ledler ve genel amaçlı giriş/çıkış (i/o) bağlantı noktaları hakkında bilgi verilecektir.
- ❖ ‘Ses Arayüzü’ bölümünde ses portları hakkında bilgi verilecektir
- ❖ ‘Video Arayüzü’ bölümünde video portları hakkında bilgi verilecektir

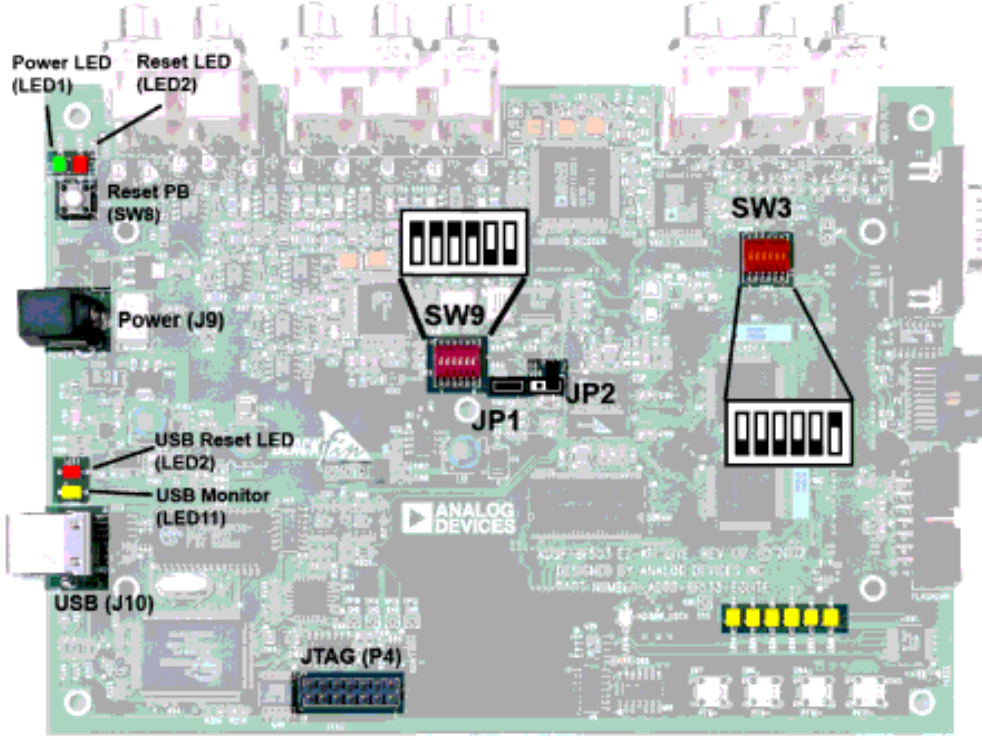
#### 1.4.1. Paket İçeriği

ADSP-BF533 EZ-KIT Lite kit ile birlikte program geliştiricilere;

- ❖ ADSP-BF533 EZ-KIT Lite kit Baskı devre kartı
- ❖ CD:
  - VisualDSP++ yazılımı
  - ADSP-BF533 EZ-KIT Lite kit ile koordine çalışacak yazılım
  - USB sürücüler
  - Örnek programlar
  - ADSP-BF533 EZ-KIT Lite kit el kitabı
- ❖ 7.5V DC gerilim kaynağı
- ❖ USB 2.0 türünde ara kablo

Donanımları sunulmaktadır.

### 1.4.2. Varsayılan Yapılandırma



Şekil 1.2 : EZ-LITE Kit için varsayılan yapılandırma

ADSP-BF533 EZ-KIT Lite kit bilgisayar kasasında herhangi bir yuvaya takmaya gerek kalmadan bir USB kablosu yardımıyla çalışabilecek şekilde tasarlanmıştır. Şekil 1.2 yardımıyla kit için varsayılan yapılandırma kontrol edilebilmektedir.

### 1.4.3. Yeni Oturum Açma

VisualDSP++ yazılım ile iki farklı şekilde oturum açılması mümkündür. Çevrimdışı bir şekilde oturum açılması durumunda ADSP-BF533 EZ-KIT Lite kit'in sisteme bağlı olmasına gerek kalmadan geliştirilen programlar test edilebilmektedir. Online oturum açılması durumunda ADSP-BF533 EZ-KIT Lite kit'in sisteme bağlı olması gerekmektedir. Aksi durumda VisualDSP++ yazılımı çalışmayacak ve hata verecektir. Kit USB bağlantısı yapıldıktan ve güç girişi takıldıktan sonra kısa bir süreliğine bekleme konumuna geçecek sistem ile bağlantı sağlayacak ve daha sonra 11 numaralı led aktif konumda kalacaktır. 11 numaralı ledin yanması sistemde bir sorun olmadığını ve haberleşmenin başarılı bir şekilde sağlanabileceği anlamına gelmektedir.

#### 1.4.4. Hafıza Haritası

Tablo 1.1. : ADSP-BF533 EZ-KIT Lite kit için harici hafıza haritası

Başlangıç Adresi	Bitiş Adresi	İçerik
0x0000 0000	0x07FF FFFF	SDRAM Bank 0 (SDRAM).
0x2000 0000	0x200F FFFF	ASYNC hafıza Bank 0 (Birincil Flash A).
0x2010 0000	0x201F FFFF	ASYNC hafıza Bank 1 (Birincil Flash B).
0x2020 0000	0x202F FFFF	ASYNC hafıza Bank 2 (Flash A ve B, ikincil hafıza, SRAM ve dahili saklayıcılar).
Tüm diğer bölgeler		Kullanılmamakta

ADSP-BF533 EZ-KIT Lite kit veri saklama için SDRAM ve Flash hafıza türünde iki tane harici blok bulundurmaktadır. Ayrıca ADSP-BF533 Blackfin işlemcisi dahili olarak bir SRAM bulundurmaktadır. SDRAM 64 MB'lık(32M x 16-bit) bir hafızaya sahiptir. Flash bellek iki ise iki farklı bloktan oluşmaktadır ve flashA ve flashB olarak isimlendirilmektedir. Şekilde ADSP-BF533 EZ-KIT Lite kit için hafıza haritası gösterilmektedir.

Tablo 1.2. : ADSP-BF533 EZ-KIT Lite kit için dahili hafıza haritası

Başlangıç Adresi	Bitiş Adresi	İçerik
0xFF80 0000	0xFF80 3FFF	Data Bank A SRAM 16 KB
0xFF80 4000	0xFF80 7FFF	Data Bank A SRAM/CACHE 16 KB
0xFF90 0000	0xFF90 3FFF	Data Bank B SRAM 16 KB
0xFF90 4000	0xFF90 7FFF	Data Bank B SRAM/CACHE 16 KB
0xFFA0 0000	0xFFA0 FFFF	Komut SRAM 64 KB
0xFFA1 0000	0xFFA1 3FFF	Komut SRAM /CACHE 16 KB
0xFFB0 0000	0xFFB0 0FFF	Kullanıcı erişimli SRAM 4 KB
0xFFC0 0000	0xFFDF FFFF	Sistem MMRs 2 MB
0xFFE0 0000	0xFFFF FFFF	Çekirdek MMRs 2 MB
Tüm diğer bölgeler		Kullanılmamakta

### 1.4.5. SDRAM Arayüzü

Tablo 1.3. : SDRAM için optimum yapılandırmalar

Saklayıcı	SCLK = 133 MHz (işlemci MAX)	SCLK = 126 MHz (CCLK = 756 MHz)	SCLK = 118.8 MHz (CCLK = 594 MHz)
EBIU_SDGCTL	0x0091 998D	0x0091 998D	0x0091 998D
EBIU_SDBCTL	0x0000 0025	0x0000 0025	0x0000 0025
EBIU_SDRRC	0x0000 0406	0x0000 03CF	0x0000 0397

SDRAM(MT48LC32M19) 64 MB'lık veri saklama kapasitesine sahiptir. 16 bitten oluşan her bir adres 32 MB'lık bir blok boyunca yayılmıştır. Dolayısı ile toplam veri kapasitesi 64MB'tır. Her bir uygulama başlangıcında başlangıç adres değerleri başlık dosyaları tarafından yenilenmektedir ve hafıza haritası işlemciye ulaşmaktadır.

### 1.4.6. Flash Hafıza

ADSP-BF533 EZ-LITE kit STMicroelectronics firmasının üretmiş oldu iki tane PSD4256G6V modelinde birim barındırmaktadır. Bu birimler yalnızca Flash bellek olarak kullanılmamakta olup ayrıca genel amaçlı i/o portlarının da barındırılmasını sağlamaktadır.

Her bir birim aşağıda listelenmekte olan hafıza segmentlerini barındırmaktadır:

- ❖ Birincil hafızanın 1 MB'lık bir bölümünü,
- ❖ İkincil hafızanın 64 KB'lık bir bölümünü,
- ❖ Dahili SRAM'ın 64 KB'lık bir bölümünü,
- ❖ Yapılandırma saklayıcılarının (I/O kontrol) 256 byte'lık bir bölümünü.

Tablo 1.4. : Asenkron hafıza kontrol saklayıcılarının ayarlanması

Saklayıcı	Değer	Fonksiyon
EBIU_AMBCTL0	0x7BB0 7BB0	Bank0 ve Bank1 için zamanlama kontrolü
EBIU_AMBCTL1 bits 15-0	0x7BB0	Bank2 için zamanlama kontrolü Bank3 kullanılmamaktadır.
EBIU_AMBCTL bits 3-0	0xF	Tüm bank'ları aktifleştirmektedir.

Tablo 1.5. : Flash hafıza haritası

Başlangıç Adresi	Bitiş Adresi	İçerik
0x2000 0000	0x200F FFFF	Birincil Flash A (1 MB)
0x2010 0000	0x201F FFFF	Birincil Flash B (1 MB)
0x2020 0000	0x2020 FFFF	İkincil Flash A (64 Kb)
0x2024 0000	0x2024 7FFF	Flash A SRAM (32 KB)
0x2027 0000	0x2027 00FF	Flash A Saklayıcıları (256 KB)
0x2028 0000	0x2028 FFFF	İkincil Flash B (64 KB)
0x202C 0000	0x202C 7FFF	Flash B SRAM (32 KB)
0x202E 0000	0x202E 00FF	Flash B Saklayıcıları (256 KB)
Tüm diğer bölgeler		Reserved

#### 1.4.7. Genel Amaçlı I/O Bağlantı Noktaları

Flash A ve Flash B'nin uygun saklayıcılarının ayarlanmasıyla kontrol edilen genel amaçlı bu bağlantı noktaları için adres alanları yukarıdaki tabloda verilmektedir. Flash birimin I/O her bir bağlantı portu 8 bitlik olup A dan G'ye kadar isimlendirilmektedir ve her bir port için 8 bitlik saklayıcılar bağlantı noktası yönü, veri giriş ve veri çıkış bilgilerini taşımaktadır. Donanımın resetlenmesi ve ani gerilim değişimleri sonucu oluşan durumlarda port yön ve çıkış saklayıcıları temizlenmektedir. Direction saklayıcısının ilgili bitinin 0 olması onun giriş olarak, 1 olması ise ilgili pinin çıkış olarak konfigüre edildiğini belirtmektedir. Data In saklayıcıları yazma korumalı saklayıcılar olup portun o anki lojik durumunun okunmasını sağlamaktadır. Data Out saklayıcısı hem yazma hem de okuma işlemine olanak sağlamaktadır.

Tablo 1.6 : Flash A için yapılandırma saklayıcıları - PORT A,B

Saklayıcı Adı	Port A adres	Port B adres
Data In (yalnızca okuma)	0x2027 0000	0x2027 0001
Data Out (okuma-yazma)	0x2027 0004	0x2027 0005
Direction (okuma-yazma)	0x2027 0006	0x2027 0007

Tablo 1.7. : Flash B için yapılandırma saklayıcıları - PORT A,B

Saklayıcı Adı	Port A adres	Port B adres
Data In (yalnızca okuma)	0x202E 0000	0x202E 0001
Data Out (okuma-yazma)	0x202E 0000	0x202E 0005
Direction (okuma-yazma)	0x202E 0006	0x202E 0007

Tablo 1.8. : Flash A Port A Control

Bit #	I/O	Bit değeri
7	Belirtilmemiş	Any
6	Belirtilmemiş	Any
5	PPI Clock Select bit 1	00 = Local OSC (27 MHz)
4	PPI Clock Select bit 0	01 = Video Decoder Pixel Clock
3	Video Decoder Reset	0 = RESET ON; 1= RESET OFF
2	Video Encoder Reset	0 = RESET ON; 1= RESET OFF
1	Reserved	Any
0	Codec Reset	0 = RESET ON; 1= RESET OFF

Tablo 1.9. : Flash A Port B Control

Bit #	I/O	Bit değeri
7	Kullanılmamakta	Any
6	Kullanılmamakta	Any
5	LED9	0 = LED OFF; 1= RESET ON
4	LED8	0 = LED OFF; 1= RESET ON
3	LED7	0 = LED OFF; 1= RESET ON
2	LED6	0 = LED OFF; 1= RESET ON
1	LED5	0 = LED OFF; 1= RESET ON
0	LED4	0 = LED OFF; 1= RESET ON



#### **1.4.8. Ledler ve Butonlar**

EZ-LITE kite üzerinde 4 tane buton ve genel amaçlı kullanım için 6 tane lede bulunmaktadır. Butonlar SW4-SW7 arasında isimler almaktadır. Ve programlanabilir bayraklardan değerleri okunabilmektedir. İlgili bayrak için okunana değerin '1' olması ilgili butonun basıldığına karşılık gelmektedir. Ledler ise flash hafıza birimine bağlı genel amaçlı I/O bağlantı noktaları tarafından kontrol edilmektedir.

#### **1.4.9. Ses Arayüzü**

AD1836 ses kodek birimi 3 kanallı stereo çıkış ve 96KHz de giriş alan 2 tane giriş bağlantı noktasına gerekli servisi verebilmektedir. İşlemcinin SPORT0 birimi AD1836 ses kodek entegresine bağlanmıştır ve stereo ses giriş ve çıkışlarını bu port yardımıyla değerlendirmektedir. BF533 işlemcisi ses kodeği ile hem zaman bölüşümlü çoklu erişim (Time-Division Multiplexed Access - TDMA) hem de çift yönlü arayüz (Two-Wire Interface - TWI) modlarıyla haberleşme kurabilmektedir. TWI modu kodek için 96 KHz de örnekleme hızı sunabilmektedir ama bu durumda yalnızca çıkışların yalnızca iki tanesinin kullanımı mümkün olmaktadır. Simülasyon amaçlı ve optimum olarak değerlendirilen 48 KHz de yapılan örneklemelemlerde tüm giriş ve çıkış portlarının kullanımı mümkün olabilmektedir.

AD1836 ses kodeği üzerindeki dahili yapılandırma saklayıcılarına işlemcinin SPI portu kullanılarak erişilebilmektedir. İşlemcinin PF4 programlanabilir bayrağı cihaz seçimi için kullanılmaktadır. Bu bayrak uygun bir şekilde ayarlanarak istenilen cihaza seri arayüz ile erişilmesi mümkün olmaktadır. AD1836 için reset konumuna alınma bilgisi Flash A üzerinde bulunan PA0 bağlantı noktasından gönderilmektedir.

#### **1.4.10. Video Arayüzü**

BF533 EZ-LITE kit video giriş ve çıkışı kullanan çoklu ortam uygulamalarına yeterli desteği verebilmektedir. ADV7171 video kodlayıcı üç farklı analog video çıkışına olanak sağlarken ADV7183 video çözücü üç analog video girişini sayısal forma çevirerek işlenmesi destekleyebilmektedir. Video kodlayıcı ve video çözücü tümleşik elemanların her biri Blackfin BF533 işlemcisine paralel birim arayüzü (Parallel Peripheral Interface - PPI) ile

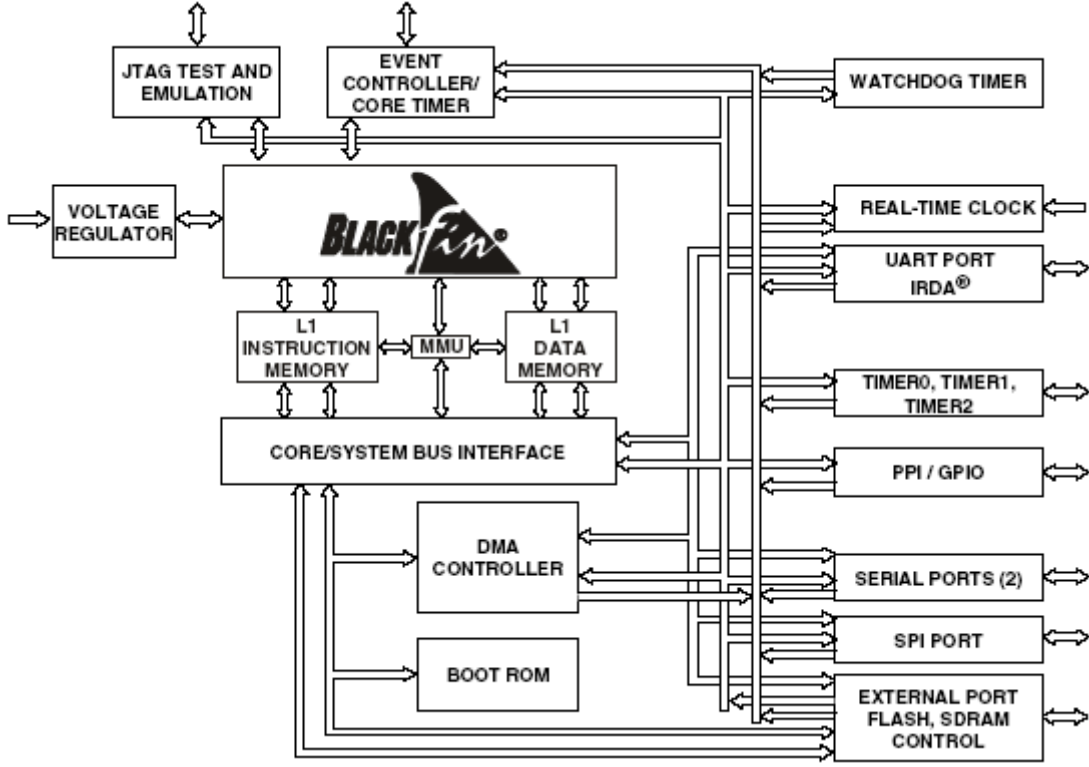
bağlanmaktadır. EZ-LITE kit ile video işleme yapılabilmesi için aşağıda listelenen aşamaların gerçekleştirilmesi gerekmektedir.

- ❖ BF533 EZ-LITE Kit üzerindeki SW3 yapılacak video uygulamasına göre ayarlanmalıdır.
- ❖ Eğer video kodlayıcı kullanılıyorsa;
  - PF2 bayrak saklayıcısı '0' yapılmalı
  - PPI saat frekansı seçilmeli
- ❖ Video birimlerinin dahili saklayıcılarına video işleme operasyonları süresince erişilmesi gerekmektedir. Her bir video birimine işlemcinin seri haberleşme ünitesi yardımıyla erişilebilmektedir. Bu haberleşmede PF0 (SCL) seri saat darbesini PF1 ise seri veriyi (SDAT) saklamaktadır.
- ❖ Blackfin BF533 işlemcisinin PPI arayüzü programlanmalıdır. (DMA, yapılandırma saklayıcıları. vb.)

### **1.5. EZ-KIT Lite Evaluation Boardların Getirmiş Olduğu Avantajlar**

- ❖ EZ-KIT Lite™ , ADI DSP'leri ile proje geliştirmek isteyen Ar-Ge mühendisleri için kullanımı kolay ve maliyeti düşük yöntem sunar.
- ❖ DSP mimarisini öğrenmeyi ve proje geliştirmeyi kolaylaştırmak için PC tabanlı tool'ları ile bir DSP evaluation board ve temel debugging software'den oluşur.
- ❖ Ar-Ge mühendisleri ADI DSP'leri hakkında hardware ve software olarak daha detaylı bilgi sahibi olabilir, uygulamalarını geliştirmeye başlayabilirler.
- ❖ VisualDSP++™ IDE ile birlikte C/C++ compiler, assembler ve linker beraberinde gelir.

## 1.6. Blackfin BF533 DSP işlemcisi



Şekil 1.3 : Blackfin BF531-BF532-BF533 işlemcileri için blok diyagram

BF533 işlemcisi BF532 ve BF531’inde aralarında bulunduğu Analog Devices™ firmasının tasarlamış olduğu blackfin işlemci ailesinin bir üyesi olarak uygulama geliştiricilerin hizmetine sunulmaktadır. Blackfin ailesi özellikle mobil uygulamalar için üretilmiş olup çok az bir besleme gerilim seviyesine ihtiyaç duyan ve düşük güç tüketen bir serinin ilk parçası olarak göze çarpmaktadır. Gerilimdeki dalgalanmalar ve frekans farklılıkları gibi sorunları aşması işlemciye yüksek bir dayanıklılık yeteneği ve uzun ömürlü bir mikroişlemci performansı katmaktadır. İşlemci ailesi sayısal haberleşme sistemleri ve çoklu ortam uygulamaları için oldukça verimli bir hız kapasitesine sahip bulunmaktadır. Şekil 1.3’de işlemciye ait blok diyagram verilmektedir.

### 1.6.1 Blackfin BF533 ‘ün Özellikleri

- ❖ Blackfin çekirdek
  - 756 MHz’e kadar saat frekansı
- ❖ Hafıza

- 80KB'a kadar L1 Instr. (16KB Cache/SRAM)
- 64KB L1 Data (32KB Cache/SRAM)
- ❖ 16-Bit external bus interface
- ❖ 16-Bit PPI/Video I/O portları
- ❖ I<sup>2</sup>S destekleyen 2 adet seri port(SPORTs)
- ❖ SPI
- ❖ UART, IrDA
- ❖ 4 adet genel amaçlı timer
- ❖ 16 adet bidirectional GPIO pini
- ❖ Watchdog timer
- ❖ Real time clock (RTC)
- ❖ Dahili gerilim regülatörü
- ❖ 0.8 V to 1.2 V V<sub>DD</sub> çekirdek gerilimi
- ❖ 2 x 16-bit MACs
- ❖ 2 x 40-bit ALUs
- ❖ 4 x 8-bit video ALUs
- ❖ 40-bit shifter
- ❖ 12 kanal DMA

### 1.7. VisualDSP++ Nedir?

- ❖ VisualDSP++ gelişmiş bir Project magener'ı olan bir integrated development ve debugging environment(IDDE)'dir. Bu sayede kullanıcılar editing, building ve debugging arasında kolay ve hızlı geçiş yapabilirler.
  - VisualDSP++ Kernel (VDK)
  - C/C++ compiler
  - Gelişmiş “plotting tools” sayesinde kullanıcılar yazdıkları software'in performansını görsel olarak ölçebilirler
  - İstatistiksel profili
- ❖ Güçlü ve esnek programming tool
- ❖ Basılı ve .pdf olarak full manual set

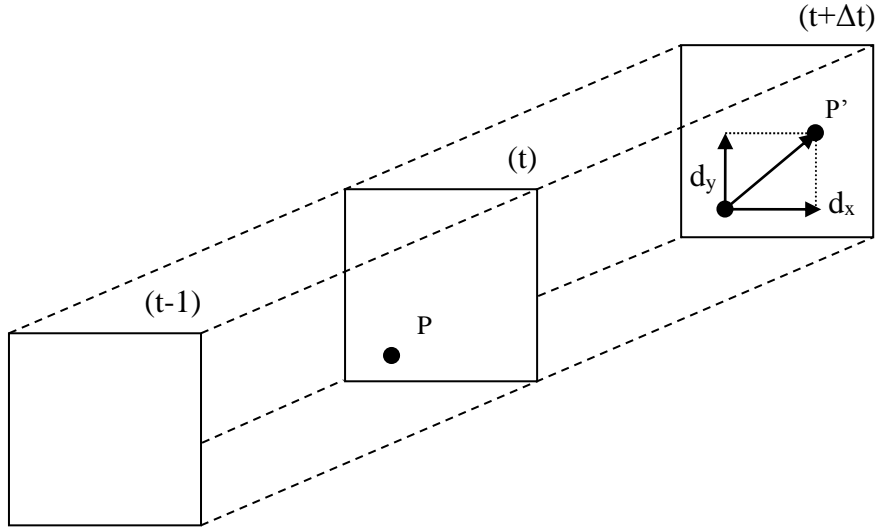
### 1.7.1. VisualDSP++ 4.0'ın Desteklediği DSP'ler

- ❖ VisualDSP++ 4.0 bir “all-inclusive” dir
  - Blackfin, Sharc ve TigerSHARC aileleri için tek kurulum
  - Ayrıca ICE,EZ-KIT Lite veya Windows driver'larını kurmaya gerek yoktur.
  - gerekli driverlar VDSP++ ile birlikte otomatik olarak yüklenir.
  - Güncellemeler ortalama iki ayda bir [www.analog.com](http://www.analog.com)'dan yapılabilmektedir.
- ❖ Desteklenen DSP'ler (4.0 için yeniler eflatun renkli)
  - Blackfin : BF531, BF532, BF533, BF534, BF535, BF536, BF537, BF561, AD6532
  - SHARC : 21020, 21060, 21061, 21062, 21065L, 21160, 21161, 21261, 21262, 21266, 21267, 21363, 21364, 21365, 21365, 21366, 21367, 21368, 21369
  - TigerSHARC : TS101, TS201, TS202, TS203

## BÖLÜM 2. HAREKET KESTİRİM YÖNTEMLERİ

### 2.1. Giriş

Bilindiği gibi video görüntüleri çerçevelerden oluşmaktadır. Ardışık çerçeveler arası ise global olarak bir yer değişim söz konusudur. Bu yer değişim hareket vektörünü oluşturmaktadır. Hareket vektörü başka bir ifadeyle  $(t)$  anındaki çerçevenin  $(t+1)$  anında genel olarak ne kadar yer değiştirdiğidir. Hareket vektörü kestirimi MPEG-1, MPEG-2, MPEG-4, H.261, H.262 gibi video kodlama sistemlerinde yaygın olarak kullanılmaktadır. Şekil 2.1’de bir hareket vektörü gösterilmektedir.  $(t)$  anındaki  $p$  noktası  $(t+\Delta t)$  anında  $p'$  noktasına taşınmıştır. Dolayısıyla  $p'$  noktası ile  $p$  noktası arasında bir hareket vektörü oluşmaktadır. Bu hareket vektörünün x-eksenindeki bileşeni  $d_x$  vektörü, y-eksenindeki bileşeni ise  $d_y$  vektörüdür.  $d_x$  ve  $d_y$  bileşenleri ise  $d$  vektörünü yani hareket vektörünü oluşturmaktadır.

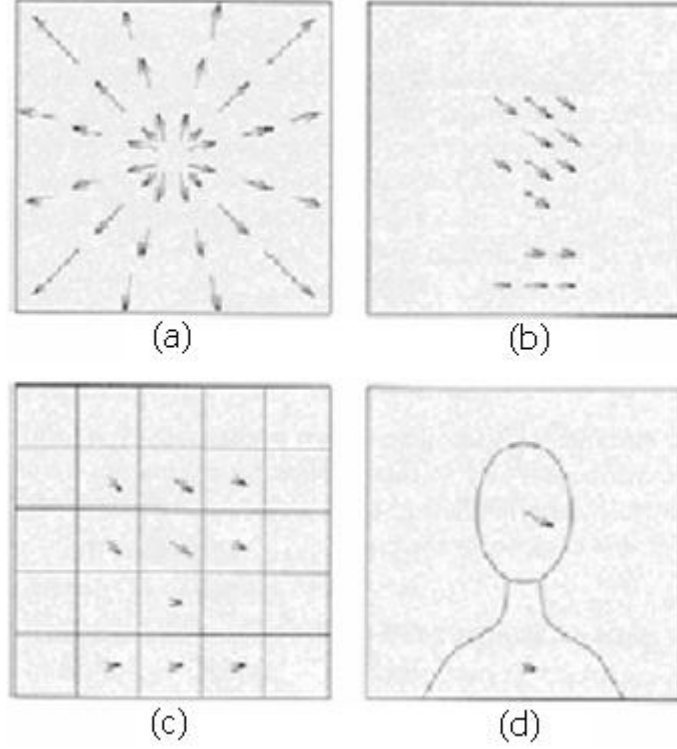


Şekil 2.1 : Hareket vektörünün gösterimi

Hareket gösterimi için çeşitli yöntemler bulunmaktadır. Hareket gösterimleri blokların hareketini, piksellerin hareketlerini, bölgelerin hareketini ve genel hareket olmak üzere 4 ana başlık altında toplanabilmektedir. Şekil 2.2’de bu gösterimler ve hareket vektörlerinin durumu verilmektedir.

- Genel bazlı ( global based )

- Piksel bazlı ( pixel based )
- Blok bazlı ( block based )
- Bölge bazlı ( region based )



Şekil 2.2 : Hareket gösterimi için çeşitli yöntemler. (a) genel hareket (b) piksellerin hareketi (c) blokların hareketi (d) bölgelerin hareketi

Görüntü stabilizasyonunda çerçevelerin genel hareketi önemli olduğu için bu çalışmada genel yani global bazlı hareket kestirimi kullanılacaktır. Global temelli hareket kestiriminde blok temelli hareket kestiriminde olduğu gibi çerçeveler bloklara ayrılmaz. Tek bir blok kullanılır. Blok büyüklüğü hareket tahmini için önemlidir.

## 2.2. Hareket Kestirim Yöntemleri

Hareket kestirimi için literatürde çeşitli yöntemler önerilmiştir. Bu yöntemlerin her birinin kendine özgü bazı avantajları olmasıyla beraber bazı önemli dezavantajları da beraberinde getirmektedirler. Aşağıda 6 ana gruba ayrılır incelenen hareket kestirim yöntemlerinin listesi verilmektedir. Bunlardan tam arama, üç adımlı arama ve sıra-düzensel arama yöntemleri detaylı olarak incelenmiş olup ilerleyen bölümlerde verilmiştir.

- ❖ Tam arama ( Full search )
- ❖ Logaritmik arama ( 2-D logarithmic search )
- ❖ Üç adımlı arama ( three-step search )
- ❖ Baklava biçimli arama ( diamond search )
- ❖ Çapraz arama ( cross search )
- ❖ Sıra-düzensel arama (hierarchical search method)

Bu arama yöntemleri, arama prosedürleri ( search procedures ) veya arama stratejileri ( search strategy ) olarak da bilinmektedir. Hareket kestirimi yönteminde en iyi sonucu veren tam aramadır (FS). Fakat bu arama yönteminin hesap yükünün çok fazla olması beraberinde yeni algoritmaların bulunmasını sağlamıştır. Tam arama yöntemi dışındaki diğer arama yöntemleri hızlı algoritmalarlardır. Hesapsal yükleri azdır. Bu yöntemlerden bulunan hareket vektörleri tam arama yönteminde bulunan hareket vektörlerine yakın değerlerdir. Üç adımlı arama yöntemi diğer hızlı algoritmaların temelini oluşturmaktadır.

Arama stratejilerinden herhangi birini kullanırken hareket vektörlerini bulmada bölge eşleme-uyumlama kriteri ( block matching criteria ) kullanılmaktadır. Bu ölçüt yardımıyla imge çerçevelerinin içerisinde seçilen belirli büyüklükteki blokların, takip eden çerçevelerdeki davranışları incelenerek çerçeveler arası hareket konusunda bilgi edinilebilmektedir.

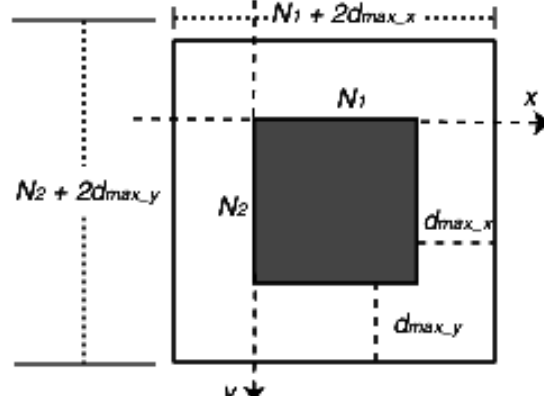
### **2.2.1. Tam Arama**

Hareket vektörü kestiriminde en iyi sonucu ‘tam arama’ algoritması vermektedir. Fakat bu algoritmanın çok geniş kapsamlı ve ayrıntılı bir arama yöntemi olmasından dolayı aşırı yüksek bir hesap yükü gerekmektedir. Tam arama genel-bazlı (global-based) veya blok-bazlı (block-based) olarak gerçekleştirilebilmektedir.

Global-temelli aramada ardışık iki görüntü çerçevesi karşılaştırılmakta ve (k) numaralı çerçeveden seçilecek büyük bir blok (k+1) numaralı çerçevede aranmaktadır. Arama işlemleri sırasında uygun bir blok eşleme ölçütü seçilerek, genellikle MAD seçilmektedir, en düşük hatayı veren bloğun tespit edilmesine çalışılmaktadır. Bu iki blok arasındaki yer değişim vektörü ise ilgili imge çerçevesi için hareket vektörünü vermektedir. Şekil 2.3’te



arama işlemi için temsili gösterim verilmektedir. Boyutları  $N_1, N_2$  ile verilen küçük blok genellikle yerel blok olarak, boyutları  $(N_1 + d_{\max_x})$  ve  $(N_2 + d_{\max_y})$  ile verilen büyük blok ise uzak blok olarak adlandırılmaktadır. Yerel blok, uzak blok içerisinde olası her konum için aranmakta ve her bir konum için bir hata değeri çıkarılmaktadır.



Şekil 2.3 : Tam aramada işlemi için temsili gösterim

Literatürde verilen çoğu uygulamada uzak blok alanı ( remote window area ) önceden belirlenmiş bir tamsayı ile sınırlandırılmaktadır. Bu sınırlama işlemi (2.1) ve (2.2) eşitlikleri ile verilmektedir.

$$-M_1 \leq d_{\max_x} \leq M_1 \quad (2.1)$$

$$-M_2 \leq d_{\max_y} \leq M_2 \quad (2.2)$$

Tam arama yönteminin hesapsal yükünün ne kadar ağır olduğunun bir göstergesi olarak eşitlik (2.3) ile verilen yapılan karşılaştırma sayısı verilebilmektedir. Bu eşitlik ne kadar karşılaştırma yapılacağıнын sayısını veren matematiksel bir ifadeden oluşmaktadır.

$$(2M_1 + 1) * (2M_2 + 1) \quad (2.3)$$

$M_1=16$  ve  $M_2=16$  alınacak olursa (2.3) ile verilen eşitliğin sonucu 1089 olarak bulunmaktadır. Buradan 1089 farklı nokta için blok karşılaştırılması yapılacağı sonucuna

ulaşmaktadır. Görüldüğü gibi karşılaştırma değerinin çok yüksek çıkmasıyla beraber bir imge çerçevesinin RGB uzayında ( Red, Green, Blue ) ifade edilmesiyle birlikte işlem yükünün daha da artacağı eşitlik (2.4) ile verilmektedir.

$$3N_1N_2F(2M_1 + 1)^2 \quad (2.4)$$

Eşitlik (2.4)'te kullanılan F sembolü saniyede gösterilen çerçeve sayısını belirtmektedir.

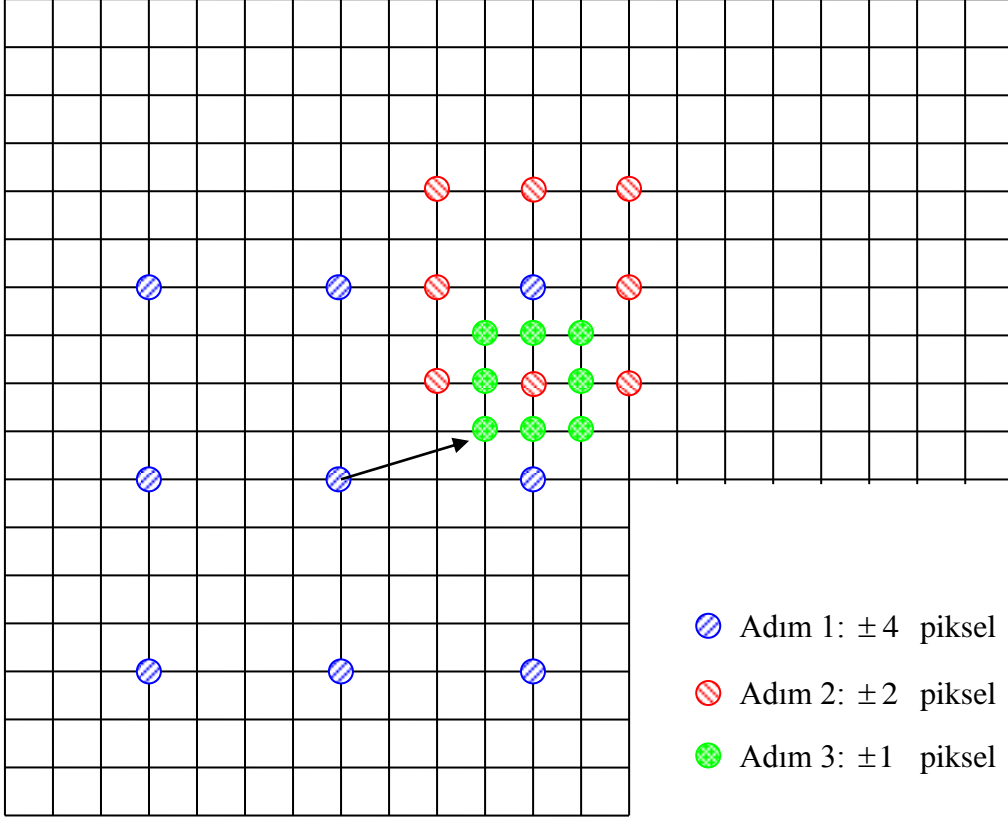
### 2.2.2. Üç Adımlı Arama

Hızlı arama algoritmalarından bir tanesi olan üç adımlı arama (Three Step Search - TSS) 1981 yılında Koga ve arkadaşları tarafından önerilmiştir [10]. Önerilen yöntemin temel amacı, işlem yükünü azaltabilmek için arama noktalarının sayısını asgari seviyeye çekebilmektir. Bu bağlamda blok eşleme-uyumlama işlemleri üç adımda gerçekleştirilmesi uygun görülmektedir. Üç adımlı arama yönteminde, ilk adım büyüklüğü (step size) maksimum arama aralığının yarısına eşit ya da yarısından biraz daha büyük olarak seçilmektedir. Yapılan tez çalışması boyunca kullanılan arama aralıkları eşitlik (2.5) ve (2.6)'da verilmektedir.

$$-8 \leq d_{\max\_x} \leq 8 \quad (2.5)$$

$$-8 \leq d_{\max\_y} \leq 8 \quad (2.6)$$

Tss yönteminin temsili arama işlemleri şekil 2.4'te verilmektedir. İlk aramada 9 arama noktası karşılaştırılmaktadır. Adım boyutu ise (step size)  $d = \pm 4$  tür (  $\pm 8/2$  ). İkinci aramada adım boyutu  $d = \pm 2$  olan 8 arama noktası kullanılmaktadır. Son aramada ise yine 8 arama noktası kullanılmasıyla beraber adım boyutu bu sefer  $d = \pm 1$  olarak seçilmektedir. Bu ifadelerden anlaşılacağı gibi her adımda, adım boyutu yarıya inmekte fakat arama noktası sabit kalabilmektedir.



Şekil 2.4 : Üç adımlı arama yöntemi için temsili gösterim

Tss yönteminde her aşamada uygun bir blok eşleme ölçütü kullanılarak en iyi hareket vektörü bulunmaktadır. En iyi hareket vektörü 1. aşama için 9 noktadan biri olarak seçilmekte ve bir sonraki aşamada en iyi hareket vektörünün bulunduğu nokta etrafında yeni bir arama başlatılmaktadır. Bu aşamadaki adım boyutu bir önceki aşamadaki adım boyutunun yarısı olarak seçilmektedir. Yine ilk aşamanın sonunda olduğu gibi en iyi hareket vektörü 2. aşama için de 8 nokta arasından seçilmektedir. Son aşamada ise bu hareket vektörünün bulunduğu nokta etrafında yeni bir arama başlatılmaktadır. Adım boyutu değeri bir önceki adım boyutunun yarısı olarak seçilmektedir. Nihayetinde son aşamada bulunan nokta ile ilk aşamada belirlenen merkez nokta arasındaki koordinat farkı, hareket vektörünü göstermektedir.

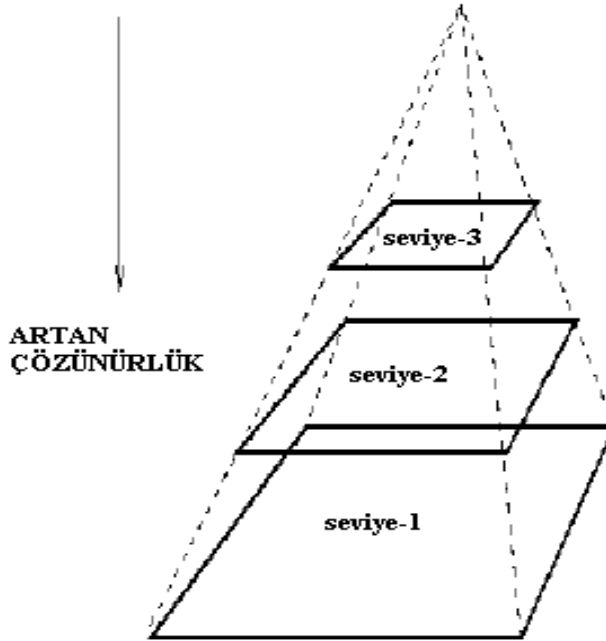
Bir diğer yaklaşım olarak üç adımlı arama N-adımlı arama olarak geliştirilebilmektedir. N-adımlı arama, hareket vektörünün N-adımda tespit edilmesi anlamına gelmektedir. Örneğin; maksimum arama aralığı  $\pm 32$  olan bir çerçevenin adım sayısı 6 olarak bulunmaktadır ve altı adımlı arama ( $\pm 16, \pm 8, \pm 4, \pm 2, \pm 1, \pm 0.5$ ) olarak adlandırılmaktadır. Hareket vektörü doğruluğu ise bu yöntem için  $\pm 0.5$  olarak bulunmaktadır..

(2.7) ile verilen ve  $d_{\max}$  'ın adım sayısını gösterdiği eşitlik yardımıyla  $d_{\max}$  adımlı bir arama işlemi için yapılacak karşılaştırma sayısı bulunabilmektedir.

$$\lceil 1 + 8 * \log_2(d_{\max}) \rceil \quad (2.7)$$

### 2.2.3. Sıra Düzensel Hareket Kestirimi

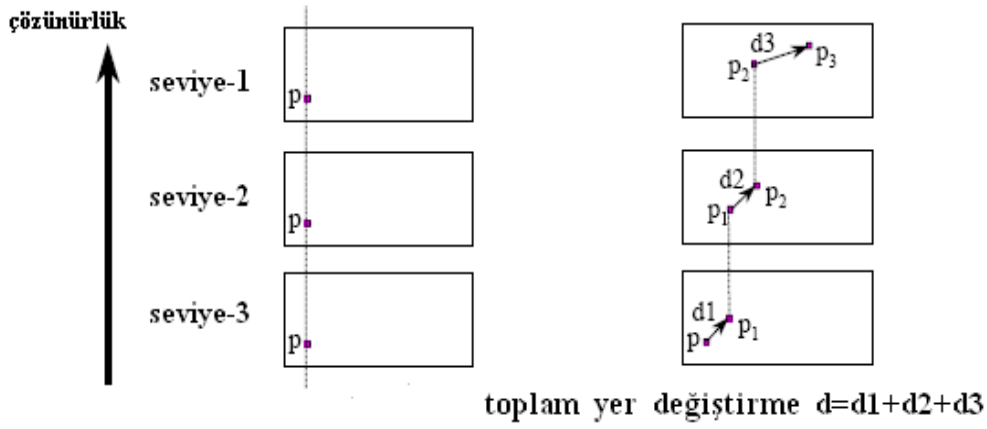
Daha iyi hareket vektörü kestirimi için, blok eşleme metodu veya faz-korelasyonu uygulamalarında sıra-düzensel hareket kestirimi (Hierarchical Motion Estimation - HME) kullanılmaktadır. Tek bir çerçevenin piramit gösterimi Şekil 2.5'te gösterilmektedir. Verilen şekilde seviye-1 en yüksek çözünürlük bölgesini belirtmektedir. Seviye-2'nin çözünürlüğü ise seviye-1'e göre daha az olmaktadır. Seviye-3 ise çözünürlüğü en az olan bölge olarak verilmektedir. Düşük çözünürlüklü imgeler uygun bir alçak geçiren filtre ve alt-örnekleme (subsampling) ile elde edilebilmektedir.



Şekil 2.5 : Sıra-düzensel imge gösterimi

Sıra düzensel hareket kestiriminin (HME) temel amacı, en düşük çözünürlükten başlayarak her seviyede hareket kestirimini bulmak ve bu hareket vektörlerini skaler olarak toplamaktır. Şekil 2.6'te HME için toplam hareket vektörü bulma yöntemi gösterilmektedir.

Önceki arama yöntemlerinde olduğu gibi HME global-temelli veya blok-bazlı olarak gerçekleştirilebilmektedir. HME yönteminde en düşük çözünürlük seviyesinde ön bilgi verebilecek seviyede detay içeren bir yer değiştirme vektörü hesaplanmaktadır. Daha yüksek çözünürlüklü seviyede ise önceki seviyede bulunan nokta (minimum MAD veya MSE) başlangıç noktası olarak kabul edilmektedir ve arama bu noktadan yapılmaktadır. Daha yüksek çözünürlüklü seviyeler daha iyi hareket vektörünün bulunmasını sağlamaktadır. Seviye sayısının artması hareket vektörünü bulmada daha iyi performans sunmaktadır ve vektör ile ilgili daha sağlıklı ve detaylı bilgilerin elde edilmesini sağlamaktadır.



Şekil 2.6 : HME yönteminde toplam yer değiştirmenin bulunması

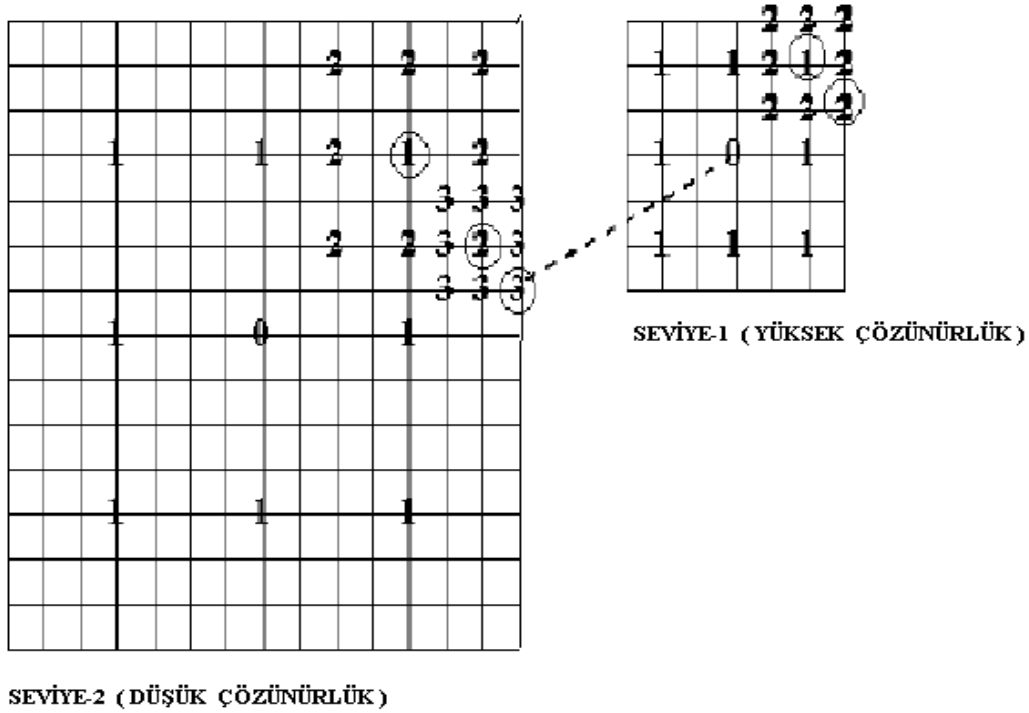
Blok-temelli hareket kestirimi uygulamalarında blok büyüklüğü, maksimum yer değiştirme değeri ve alçak-geçiren filtrenin büyüklüğü önem taşımaktadır. Tablo 2.1’de 5 seviyeli sıra-düzensel hareket kestirimi algoritması için bu niceliklerin tipik değerleri verilmektedir. Alçak-geçiren filtre olarak konvolüsyon kernel filtresi kullanılabilir. Dikkat edilmesi gereken bir nokta ise kernel filtresinin büyüklüğünün daha düşük çözünürlüklü seviyelerde daha büyük olmasıdır.

Şekil 2.7’de iki seviyeli sıra-düzensel hareket kestirimi için çalışma yapısı gösterilmektedir. HME ile hareket kestirimi bulunurken hızlı bir hareket kestirimi algoritması kullanmak gerekmektedir. Yine Şekil 2.6’da üç-adımlı arama ile sıra-düzensel hareket kestirimi uygulaması gösterilmektedir. Üç adımlı aramanın seçilme nedeni daha az arama noktası içermesi dolayısıyla işlem yükünün az olması ve hareket vektörünü bulmada iyi bir

performans sergilemesidir. Bu yöntemde bir üst seviyeye çıkıldığında toplam adım sayısı bir azaltılmaktadır.

Tablo 2.1. : 5 seviyeli HEM için tipik parametreler

Seviye	FİLTRE BÜYÜKLÜĞÜ	MAKSİMUM YER DEĞİŞTİRME	BLOK BÜYÜKLÜĞÜ
1	3	$\pm 1$	4
2	5	$\pm 3$	8
3	5	$\pm 7$	16
4	10	$\pm 15$	32
2	10	$\pm 31$	64



Şekil 2.7 : 2 seviyeli sıra-düzensel hareket kestirimi

### 2.3. Bölge Eşleme-Uyumlama Ölçütleri

Hareket vektörlerini bulunurken bölge eşleme-uyumlama ölçütleri kullanılmaktadır. Farklı arama stratejileri olduğu gibi farklı eşleme-uyumlama ölçütleri de literatürde önerilmiştir. Kullanılan ölçütler maksimum çapraz-ilişim ( cross-correlation ) ve minimum hatadır (

minimum error ). Minimum hata ölçütünün hesaplanabilmesi için dört değişik temel yöntem literatürde önerilmektedir. Bu yöntemler için ölçüt çıkarma denklemleri (2.8, 2.9, 2.10, 2.11, 2.12) ve yöntemlerin işleyişi aşağıda verilmektedir.

- ❖ Ortalama Karesel Hata ( mean square error ) : Bu ölçüt hesaplanırken yerel ve uzak blokların farkı alınmakta, fark değerlerinin karelerinin toplamı eleman sayısına bölünmektedir.

$$MSE(d_1, d_2) = \frac{1}{N_1 * N_2} \sum_{n_1} \sum_{n_2} [s(n_1, n_2, k) - s(n_1 + d_1, n_2 + d_2, k + 1)]^2 \quad (2.8)$$

- ❖ Ortalama Mutlak Değer (mean absolute difference ) : Bir başka uyumlama-eşleme ölçütü olan ortalama mutlak değer hesabı yerel bloğun ve uzak bloğun mutlak farklarının aritmetik ortalaması alınarak yapılmaktadır.

$$MAD(d_1, d_2) = \frac{1}{N_1 * N_2} \sum_{n_1} \sum_{n_2} |s(n_1, n_2, k) - s(n_1 + d_1, n_2 + d_2, k + 1)| \quad (2.9)$$

- ❖ Eşik farklılığı sayısı ( number of thresholded differences ) :

$$NTD(d_1, d_2) = \frac{1}{N_1 * N_2} \sum_{n_1} \sum_{n_2} N(s(n_1, n_2, k), s(n_1 + d_1, n_2 + d_2, k + 1)) \quad (2.10)$$

$$N(\alpha, \beta) = \begin{cases} 1 & , \quad |\alpha - \beta| > T_0 \\ 0 & , \quad |\alpha - \beta| \leq T_0 \end{cases} , \quad T_0 = esik \quad (2.11)$$

Bu bağıntılarda;

$(d_1, d_2)$ , Hareket vektörünü

K, imgenin çerçeve numarasını

$(n_1, n_2)$ , imgenin k nolu çerçevenin koordinatlarını

$N_1, N_2$  çerçeveden alınacak bloğun büyüklüğünü

Göstermektedir.

- ❖ Number of Non-Matching Pixel ( Eşleşmeyen piksel sayısı ) : Bu yöntem ikili veri türü içeren imgeler için eşleme ölçütü olarak kullanılmaktadır. MSE ve MAD blok eşleme ölçütüne göre daha az işlem yükü gerektirmektedir. İşlemin donanımsal olarak yalnızca EX-OR kapılarıyla gerçekleştirilmesi sisteme işlem yükü açısından çok büyük avantajlar sunmaktadır. Hesaplanan *ERR* değerinin düşük olması o blok için daha iyi bir eşleme yapıldığı anlamına gelmektedir. Eşleme işlemleri için kullanılan matematiksel yapı (2.13)'de verilmektedir.

$$I_2(i, j) = \begin{cases} 1 & , I^l(i, j) \neq I^r(x+i, y+j) \\ 0 & , I^l(i, j) = I^r(x+i, y+j) \end{cases} \quad , \quad i, j, x, y \in N \quad (2.12)$$

$$ERR(x, y) = \sum_{i=1}^{satir-32} \sum_{j=1}^{sutun-32} I_2(i, j)$$

Blok eşleme ölçütlerinde en iyi sonucu maksimum çapraz-İlgileşim vermektedir. Bununla beraber hesap yükünün diğer yöntemlere göre çok daha fazla olması nedeniyle pratikte gerçekleştirilmesi oldukça zor olmaktadır. Video kodlama sistemlerinde ve stabilizasyon ( compensation ) işlemlerinde genellikle, diğer blok eşleme kriterlerine göre daha az hesapsal yük içeren minimum ortalama mutlak değer (MAD) kullanılmaktadır.



## BÖLÜM 3. GÖRÜNTÜ STABİLİZASYONU

### 3.1. Giriş

Görüntü stabilizasyonunun, amatör bir kameraman tarafından yapılan veya hareket halindeki araçlardan yapılan çekimlerde, görüntülü cep telefonları ve robot-kamera uygulamaları gibi istenmeyen kamera hareketinin sebep olduğu düzensiz titreşimlerin gözleendiği alanlarda uygulamaları mevcuttur. Kaymanın sebep olduğu titreşimlerin arındırılması “iki-boyutlu görüntü dizini stabilizasyonu”, kaymanın yanında rotasyon ve zoomun sebep olduğu düzensizliklerin arındırılması “üç-boyutlu görüntü dizini stabilizasyonu” olarak adlandırılmaktadır.

Video görüntülerinde, kamera hareketinden dolayı oluşan ve çerçevenin yer deęişimi şeklinde gözlenen “global hareket” ve çerçeve içindeki bir objenin hareketi olarak beliren “yerel hareket” olmak üzere iki çeşit hareket şekli gözlenmektedir. Görüntü stabilizasyonu açısından global hareketin, kameranın yer deęişiminden dolayı gerçekleşen "istenen" ve titreşimlerden dolayı gözlenen "istenmeyen" hareket bileşenleri mevcuttur. Görüntü çerçevesini kaybetmemek için uzun süreli kamera hareketi korunarak, istenmeyen ani deęişimli hareketler stabilizasyon işlemi ile bastırılmaktadır.

Görüntü dizini stabilizasyon (GDS) sistemini iki parçaya ayırmak mümkündür;

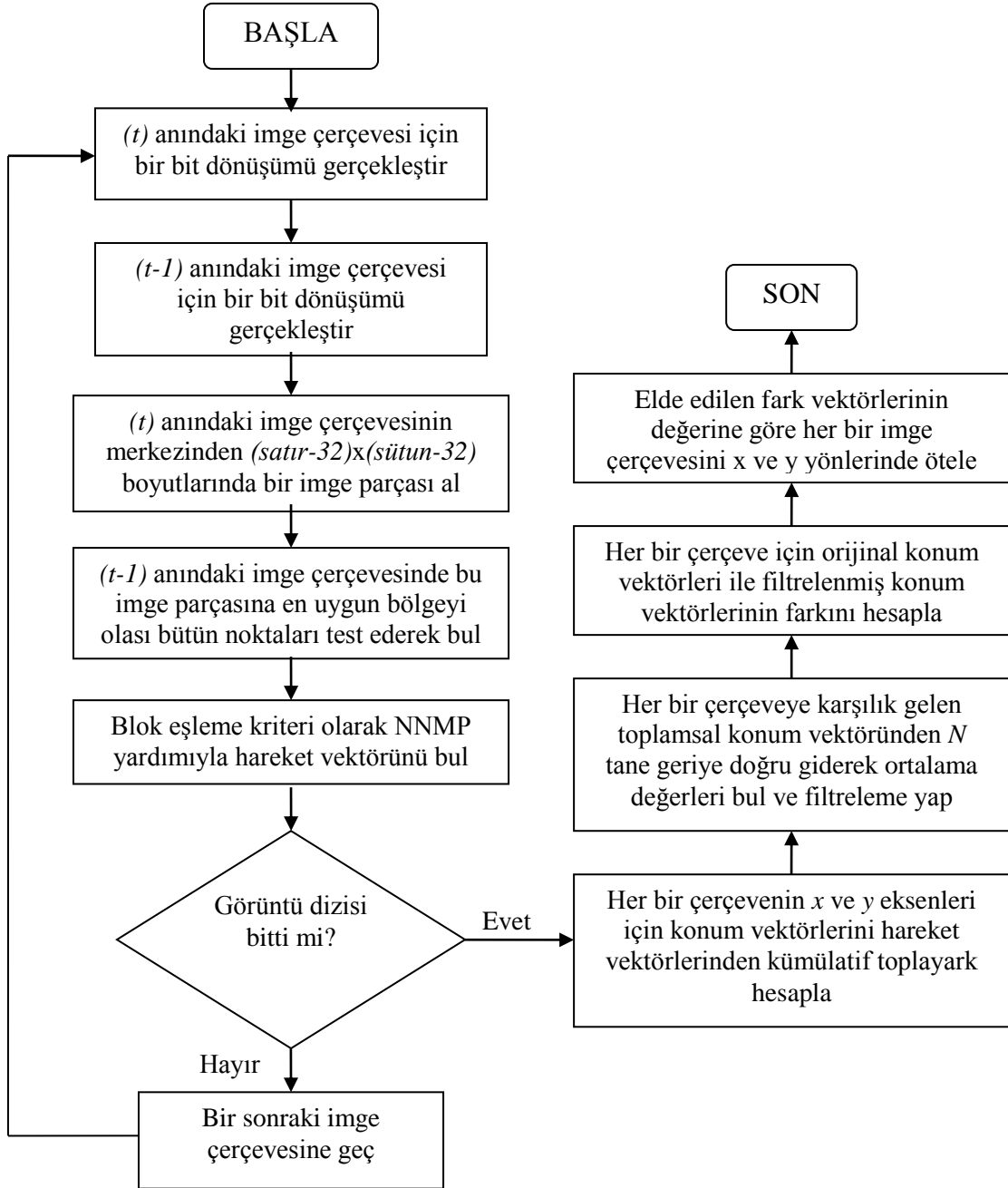
- ❖ Hareket tahmin sistemi
- ❖ Hareket düzeltme sistemi

Hareket tahmin sistemi GDS sistemi için önemlidir. Çünkü bu aşamada yapılabilecek bir hata aynı zamanda doğrudan hareket düzeltme sistemini de etkileyecektir. Hareket tahmin işlemi için genel olarak blok karşılaştırması, FFT tabanlı faz korelasyonu ve özellik karşılaştırması kullanılmaktadır. Bu çalışma boyunca hareket tahmini için bir blok karşılaştırma yöntemi olan bir bit dönüşümü temelli blok uyumlama ölçütü kullanılmaktadır. Hareket tahmin sisteminde, görüntü dizini için, global hareket vektörleri bir önceki çerçeveye göre hesaplanmaktadır.

Herhangi bir çerçevenin birinci çerçeveye göre gösterdiği toplam yer değişimi, o görüntü ve ondan önceki tüm görüntülerin global hareket vektörleri toplamına eşittir.

### 3.2. Önerilen Görüntü Stabilizasyonu Yöntemi

Bu çalışmada blok eşleme kriteri olarak bir bit dönüşüm yapısı kullanan ve bu kriter yardımıyla her bir imge çerçevesi için bir önceki çerçeveye göre hareket vektörlerini hesaplayan bir görüntü stabilizasyonu yöntemi önerilmektedir.



Şekil 3.1 : Önerilen görüntü stabilizasyonu için blok işleyiş yapısı

Kameranın genel hareketini izleyebilmek için her bir eksendeki hareket vektörleri toplamsal olarak elde edilmiştir. Stabilizasyon işlemleri aşamasında, oluşan titreşim etkilerinin giderilmesi için toplamsal hareket vektörlerindeki değişim takip edilerek  $N$  gecikmeli ortalama alan bir filtre yapısı kullanılmaktadır. Böylece toplamsal vektörlerdeki titreşimi sembolize eden yüksek frekans bileşenleri sistemden arındırılmakta ve stabilize olmuş yeni toplamsal hareket vektörleri elde edilmektedir. Gecikme parametresi olarak kullanılan  $N$  değerinin artışı alçak geçiren bu filtrenin kesim frekansı daha düşük değerlere doğru çekmektedir. İmge çerçevelerinin yeni konumlarının bulunması aşamasında her bir çerçevenin iki ekseni için orginal toplamsal konum vektörü ile filtrelenmiş toplamsal konum vektörlerinin fark değerinden yararlanılmaktadır. Nihayetinde stabilize olmuş imge çerçevelerinin elde edilebilmesi için fark vektörlerinin değerine göre imge çerçeveleri fark vektörlerinin tersine etki yapacak şekilde  $x$  ve  $y$  eksenleri boyunca ötelenmektedir. Şekil 3.1’de önerilen yöntemin işleyişi için blok yapı gösterilmektedir.

### 3.2.1. Bir Bit Dönüşümü (One Bit Transform)

Bir bit dönüşümü imgeleri çok bit ile ifade edilen renk uzayından tek bit ile ifade edilebilen renk uzayına indirmek için kullanılmaktadır ve bu çalışma boyunca  $19 \times 19$  boyutlarında bit veri türünde değerler içeren yapı elemanı matrisi kullanan bir bit dönüşümü yapısı

```
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

Şekil 3.2 : Bir bit dönüşümü için kullanılan yapı elemanı matrisi

kullanılmıştır. Kullanılan yapı elemanı matrisi Şekil 3.2’de verilmektedir. Bir bit dönüşümü yapılmış imge çerçevesi işlemi Şekil 3.2’de verilen yapı elemanı ile orijinal imge çerçevesinin konvolüsyonu sonucunda elde edilmektedir. Konvolüsyon işlemleri için kullanılmakta olan matematiksel ifadeler (3.1)’de verilmektedir. Bu işlem sonucunda elde edilen imge çerçevesinin her bir pikselinin ışıklılık değeri, konvolüsyon işlemlerinde gördüğü anlamlı bit sayısına bölünerek dönüşüm yapılmış ışıklılık değerleri elde edilmektedir.

$$C(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2) \quad (3.1)$$

(3.2)’de verilen bir bit dönüşümü ilkesi bu çalışmada hareket vektörlerinin hesaplanabilmesi için kullanılmaktadır. Bir bit dönüşümü için konvolüsyon işlemleri tamamlandıktan sonra eşitlik 3.2 ile verilen kontrol kullanılarak her bir piksel için ışıklılık seviyeleri tek bit ile ifade edilebilmektedir. Burada  $I_1$  orijinal imge çerçevesini,  $I_1'$   $I_1$  imgesinin Şekil 2.8’de verilen yapı elemanı ile konvolüsyon işleminin sonucunu,  $I_2$  ise bir bit dönüşümü alınmış imge çerçevesini,  $x$  yatay uzamsal konumu,  $y$  ise dikey uzamsal konumu göstermektedir.

$$I_2(x, y) = \begin{cases} 1 & , I_1(x, y) \geq I_1'(x, y) \\ 0 & , I_1(x, y) < I_1'(x, y) \end{cases} , x, y \in N \quad (3.2)$$

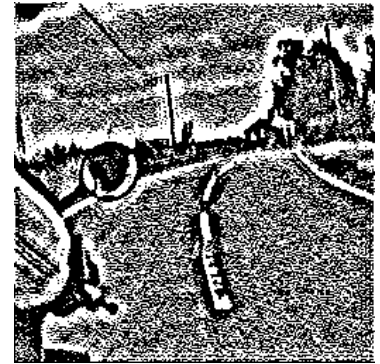
Şekil 3.3’de “Bike” görüntü dizisinin 40. ve “silent” görüntü dizisinin 76. kareleri için orijinal, konvolüsyon işlemi sonucu ve bir bit dönüşümü alınan imge çerçeveleri verilmektedir.



(a)



(b)



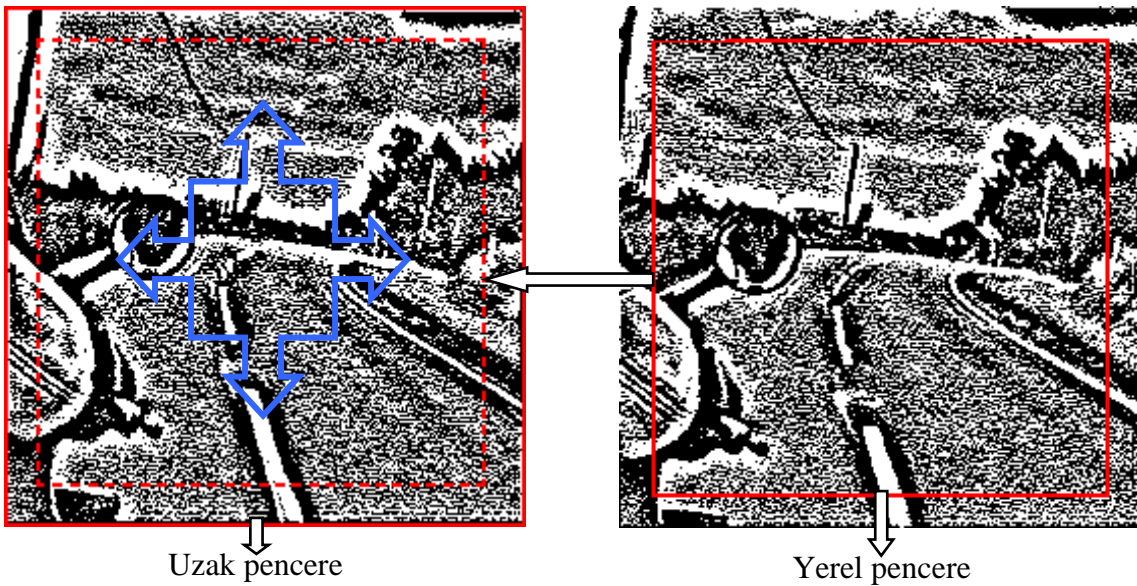
(c)

Şekil 3.3 : “Bike” görüntü dizisi için 40. imge çerçevesi (a) Orijinal imge çerçevesi (b) Konvolüsyon işleminin sonucunda oluşan imge çerçevesi (c) Bir bit dönüşümü alınmış imge çerçevesi

### 3.2.2. Blok Eşleme İşlemleri

Bu çalışmada blok eşleme ölçütü olarak kullanılan bir bit dönüşüm temelli ex-or işlemi kullanılmıştır. Hareket kestirimi için ( $t$ ) anındaki bir bit dönüşümü yapılmış imge çerçevesinden ( $\text{satir sayı}-32$ ) $\times$ ( $\text{sutun sayı}-32$ ) boyutlarında bir imge parçası alınarak ( $t-1$ ) anındaki bir bit dönüşümü yapılmış imge çerçevesi içerisinde olası tüm uzamsal konumlarda aranmaktadır. ( $t$ ) anındaki imge çevresinden alınan imge parçası bundan sonraki aşamalarda yerel pencere (Local Window - LW), ( $t-1$ ) anındaki imge çerçevesi ise uzak pencere (Remote Window - RW) olarak isimlendirilecektir. Arama işlemleri sonucunda minimum hatayı verecek piksel konumu belirlenmektedir. Eşitlik (2.12)’de bir blok için bölge uyumlama kriterinin çıkarılma eşitliği verilmektedir.

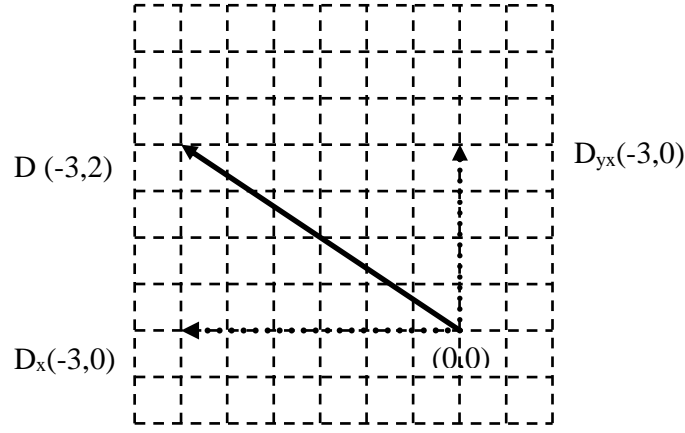
Burada  $I_2$  bit veri türünde değer bulunduran ve karşılaştırma sonucu elde edilen hataları barındıran ikili matrisi;  $I^l$  yerel pencereyi;  $I^r$  uzak pencereyi;  $ERR$  toplam uyumlama kriteri için kullanılan hata miktarını;  $i,j$  yerel pencere için yatay ve dikey uzamsal konumları;  $x,y$  ise uzak pencere için yatay ve dikey uzamsal düzlemde başlangıç konumlarını, satir,sutun imge çerçevesi için en,boy bilgilerini göstermektedir.  $ERR$ ’nin düşük çıkması o blok için daha iyi bir eşleşme yapıldığı anlamına gelmektedir. Şekil 3.4’de blok eşleme işlemleri için sembolik işleyiş verilmektedir. Bu şekilde ‘uzak pencere’, “Bike” görüntü dizisininin 21. çerçevesinden, ‘yerel pencere’ ise 22. çerçevesinden seçilmiştir.



Şekil 3.4 : Blok eşleme işlemleri için sembolik işleyişi

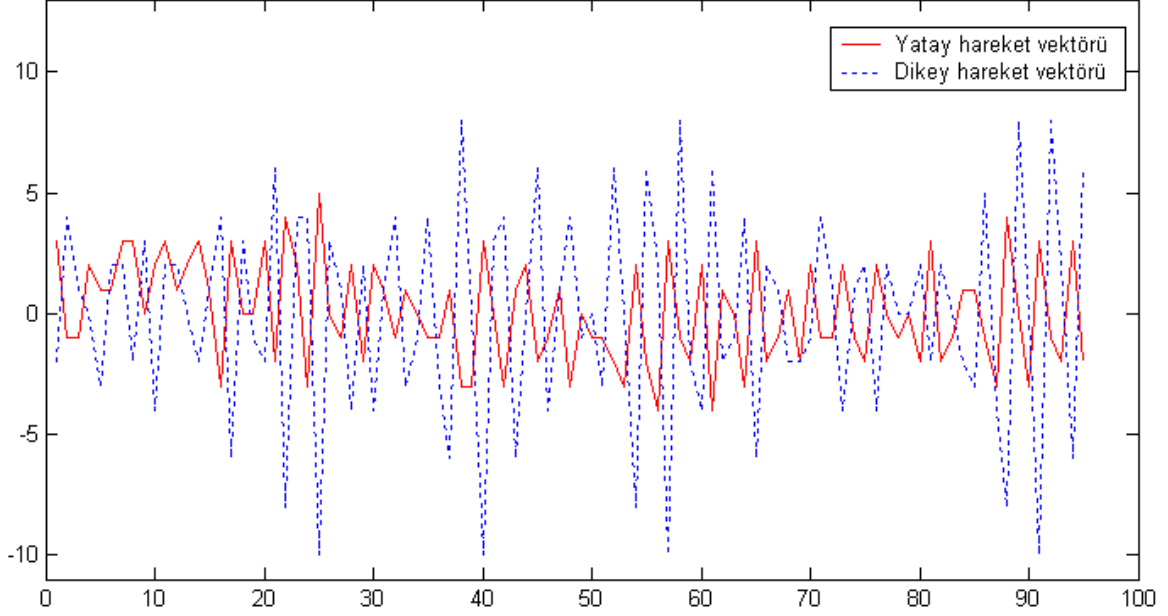
### 3.2.3. Hareket Vektörlerinin Bulunması

Blok uyumlama işlemlerinin ardından en düşük *ERR* değerini veren konuma göre hareket vektörü hesaplanmaktadır. Hareket vektörleri ( $t$ ) anındaki imge çerçevesinin ( $t-1$ ) anındaki imge çerçevesine göre yatay ve düşey konumda ne miktarda yer değiştirdiğini göstermektedir. Şekil 3.5’de verilen imge çerçeveleri için çıkarılan hareket vektörü Şekil 3.4’de verilmektedir.



Şekil 3.5 : “Bike” görüntü dizisinin 22. çerçevesi için elde edilen hareket vektörü

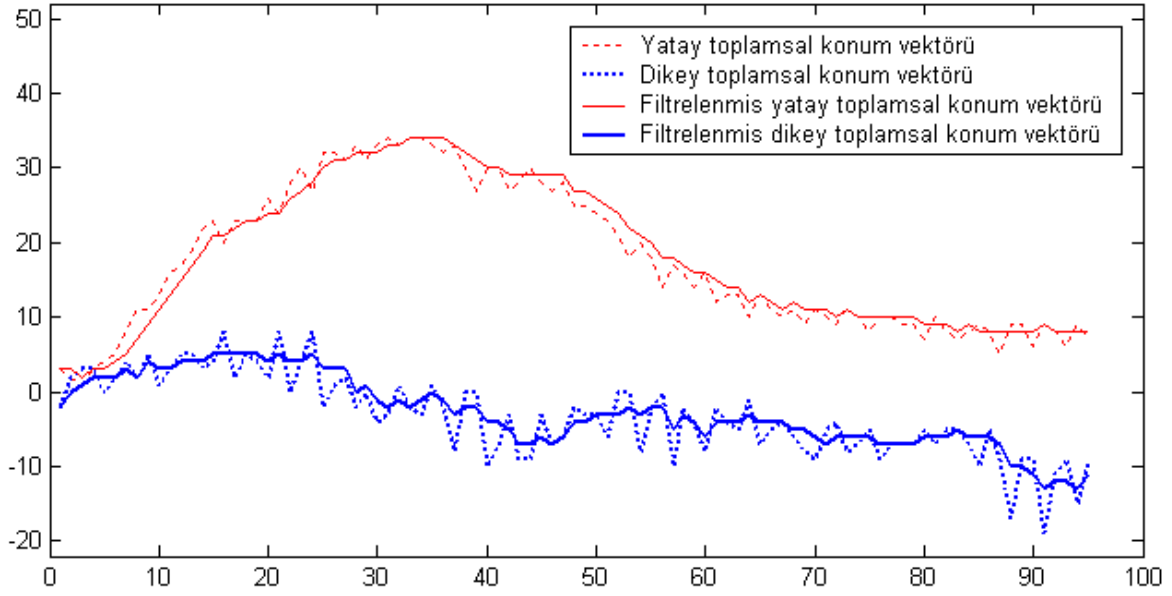
Şekilden de anlaşılacağı gibi 22. imge çerçevesi 21. imge çerçevesine göre düşey konumda 2 piksel aşağıya ve yatay konumda da 3 piksel sağa kaymıştır. Tüm çerçeveler için bu işleyişe göre yatay ve düşey konum vektörleri elde edilmektedir. Şekil 3.6’de “Bike” görüntü dizisinin 96 çerçevesi için elde edilen konum vektörleri verilmektedir. Bu grafikten faydalanılarak düşey konumdaki titreşimin yatay konumdaki titreşime oranla daha sık ve daha fazla olduğu yorumu yapılabilmektedir.



Şekil 3.6 : "Bike" görüntü dizisi için elde edilen hareket vektörleri

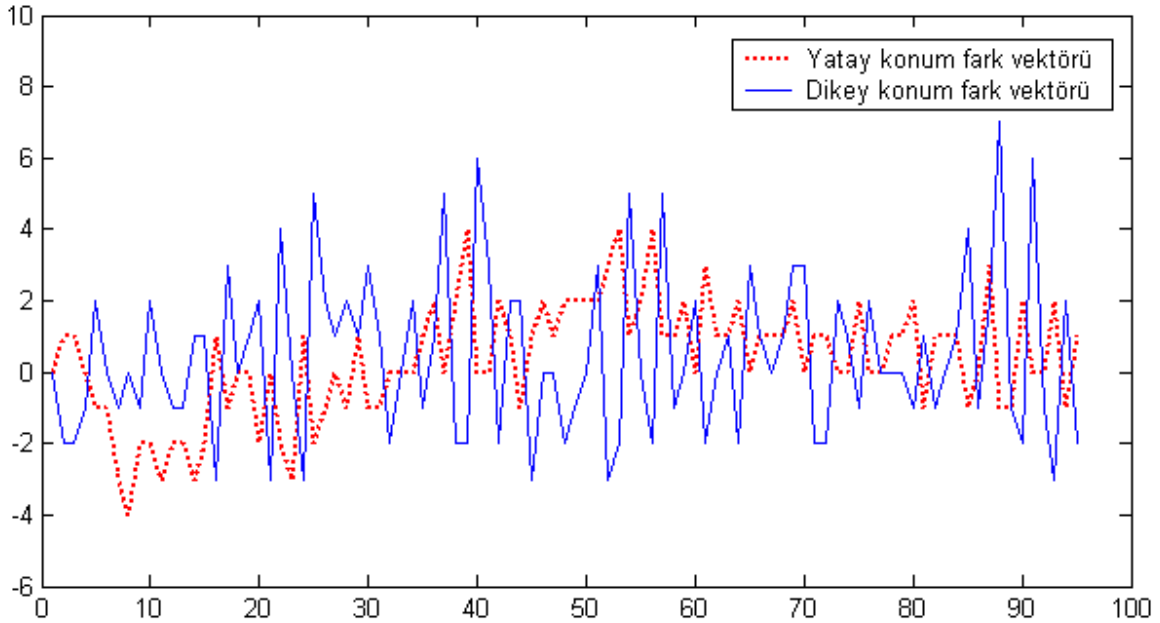
### 3.2.4. Toplamsal Hareket Vektörlerinin Bulunması ve Süzgeçleme işlemleri

Görüntü stabilizasyonu aşamasında toplamsal konum vektörleri elde edilmekte ve bu değerler alçak geçiren bir filtreden geçirilerek filtrelenmiş konum vektörü değerleri bulunmaktadır. Toplamsal konum vektörü o ana kadar olan tüm vektör değerlerinin toplanması sonucunda elde edilmektedir ve  $x,y$  uzamsal konumları için ayrı ayrı hesaplanmaktadır. Filtreleme aşamasında  $(t)$  nolu vektörden geriye doğru  $(t-N)$  nolu vektöre doğru  $N$  kadar gidilerek bu değerler toplanmakta ve sonrasında aritmetik ortalamaları alınmaktadır. Böylece alçak geçiren süzgeç işlevini yerine getiren bir filtre yardımı ile titreşimlere neden olan yüksek frekans bileşenleri bastırılmaktadır. Şekil 3.7'da hesaplanan toplamsal konum vektörleri ve  $N=3$  değeri ile geri ortalaması hesaplanan filtrelenmiş konum vektörleri verilmektedir.



Şekil 3.7 : “Bike” görüntü dizisi için elde edilen toplamsal konum vektörleri ve filtrelenmiş toplamsal konum vektörleri

### 3.2.4. Farksal konum Vektörlerinin Bulunması ve Öteleme İşlemleri



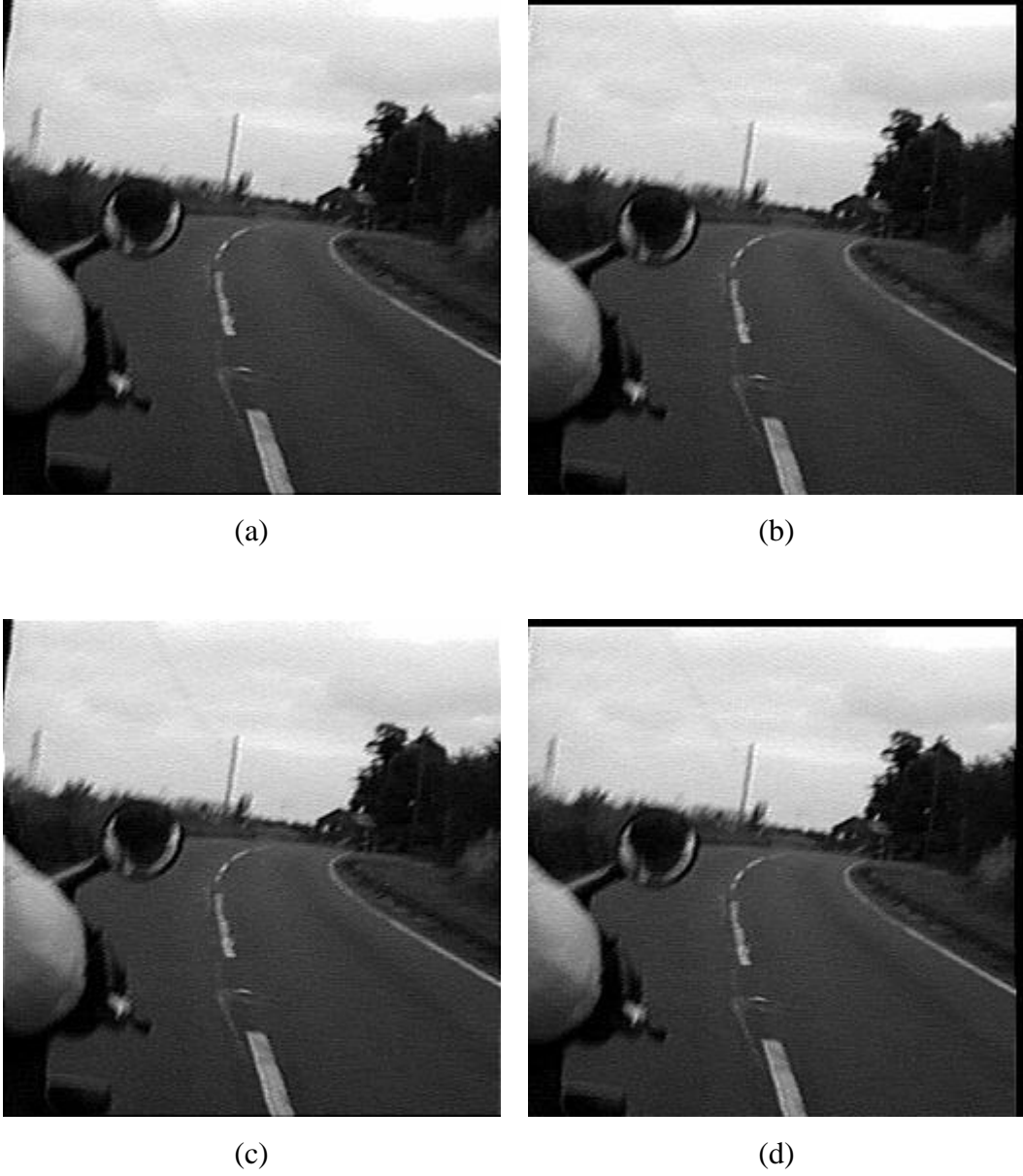
Şekil 3.8 : “Bike” görüntü dizisi için elde edilen hareket vektörleri

Filtrelenmiş toplamsal konum vektörlerinin orijinal toplamsal konum vektörleri ile farkı bulunarak yumuşak geçişler ve titreşim etkisi olmayan sanal görüntü dizisi için çerçeve



hareketi bilgisi çıkarılmaktadır. İlgili çerçeve, elde edilen değerlere göre fark vektörünün tersine etki yapacak şekilde yatay ve dikey konumda ötelenmektedir. Şekil 3.8’de “Bike” görüntü dizisinin 96 çerçevesi için elde edilen fark vektörü değerlerinin grafiği verilmektedir.

Görüntü stabilizasyonu işlemlerinin son basamağını ilgili çerçevenin hareket dengelemesi yapmak için eksenlerde yapacağı kayma işlemleri oluşturmaktadır. Şekil 3.9’de “Bike”



Şekil 3.9 : “Bike” görüntü dizisi için elde edilen stabilizasyon sonuçları

görüntü dizisine ait 23. ve 92. çerçevelerine ait orijinal imgeler ve eksenlerde kayma ile elde edilen imgeler verilmektedir. Grafikten de anlaşılacağı gibi 23. çerçeve için yatay değişim

(-2), dikey deęişim (4) ; 92. çeręeve için ise yatay deęişim (0), dikey deęişim ie 6 olarak bulunmuştur.

## **BÖLÜM 4. SONUÇLAR VE ÖNERİLER**

Bu tez kapsamında Analog Devices™ firmasının Blackfin DSP işlemcisi kullanılarak görüntü stabilizasyonu yapılmıştır. Eşleşmeyen piksel sayısı blok eşleme ölçütü donanımsal olarak kolay gerçekleştirilebildiği ve yalnızca EX-OR kapılarından oluştuğu için kolayca gerçekleştirilmektedir. Bu ölçütün kullanılması gerçek zamanlı uygulamalarda büyük bir hız avantajı sağlamaktadır. Bu sebeplerden dolayı bu tez kapsamında bir-bit dönüşümü temelli hareket kestirimi yapısı kullanılmıştır.

DSP’de elde edilen sonuçlar MATLAB™’da elde edilen sonuçlarla karşılaştırılmış ve çıkışta elde edilen sonuçların aynı olduğu gözlemlenmiştir. Tez boyunca elde edilen tüm grafikler ve çıktılar verilerek sistemin işleyişi incelenmiştir.

VisualDSP++ programı kullanılarak yazılan program kodları EK-1’de verilmektedir.

**EK-1**

Bu bölümde DSP’de Görüntü Stabilizasyonu için yazılmış olan kodlara yer verilmiştir. DSP için yazılım geliştirme aşamasında ara birim olarak VisualDSP++ kullanılmıştır. Konu ile ilgili her türlü kaynak ve program kodu için özgeçmiş bölümünde verilen adreslerden bağlantı kurulabilmektedir.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define VIDEO_DIZISI_BASLANGIC_DEGERI          1
#define VIDEO_DIZISI_BITIS_DEGERI            95
#define VIDEO_DIZISI_SATIR_SAYISI            256
#define VIDEO_DIZISI_SUTUN_SAYISI            256
#define BASLANGIC_ADRESI                      0x00
#define KERNEL_SATIR                          19
#define KERNEL_SUTUN                          19
#define N                                      7

void kaydir( unsigned char *gelen , unsigned char *giden , int cerceve_satir , int
cerceve_sutun , int y , int x )
{
    int baslangic_cikis_satir=0;
    int baslangic_giris_satir=0;
    int baslangic_cikis_sutun=0;
    int baslangic_giris_sutun=0;
    int satir_,sutun_;
    if( x<0 )
    {
        baslangic_cikis_satir=0;
        baslangic_giris_satir=(-1)*x;
    }
    if( y<0 )
    {
        baslangic_cikis_sutun=0;
        baslangic_giris_sutun=(-1)*y;
    }
    if( x>0 )
    {
        baslangic_cikis_satir=x;
        baslangic_giris_satir=0;
    }
    if( y>0 )
    {
```

```

        baslangic_cikis_sutun=y;
        baslangic_giris_sutun=0;
    }
    satir_=baslangic_giris_satir;
    for( int i=0 ; i<cerceve_satir ; i++ )
    {
        for( int j=0 ; j<cerceve_sutun ; j++ )
        {
            *(giden+i*cerceve_satir+j) = 0;
        }
    }
    for( int i=baslangic_cikis_satir ; i<cerceve_satir ; i++ )
    {
        sutun_=baslangic_giris_sutun;
        for( int j=baslangic_cikis_sutun ; j<cerceve_sutun ; j++ )
        {
            *(giden+cerceve_satir*i+j))=*(gelen+cerceve_satir*satir_+sutun_);
        }
        sutun_=sutun_+1;
        if( sutun_>=cerceve_sutun )
            break;
    }
    satir_=satir_+1;
    if( satir_>=cerceve_satir )
        break;
    }
}

```

```

void konvole_et( unsigned char *gelen , unsigned char *giden , unsigned char *kernel
, int cerceve_satir , int cerceve_sutun , int kernel_satir , int kernel_sutun )

```

```

{
    int orgin_left;
    int orgin_top;
    int xx;
    int yy;
    int toplam;
    int count;
    int deger;

    orgin_left = (kernel_satir-1)/2;
    orgin_top = (kernel_sutun-1)/2;

    for( int i=0 ; i<cerceve_satir ; i++ )
    {
        for( int j=0 ; j<cerceve_sutun ; j++ )
        {
            xx=0;
            toplam=0;
            count=0;
            for( int ii=(i-organ_left) ; ii<=(i+organ_left) ; ii++ )

```

```

    {
    yy=0;
    for( int jj=(j-organ_top) ; jj<=(j+organ_top) ; jj++ )
    {
    if( ii<0 || ii>=cerceve_satir || jj<0 || jj>=cerceve_sutun )
    {
    continue;
    }
    else
    {
    if( *(kernel+kernel_satir*xx+yy)!=0 )
    {
    count++;
    deger=*(gelen+cerceve_satir*ii+jj);
    toplam=toplam+deger;
    }
    }
    yy++;
    }
    xx++;
    }
    if( count!=0 )
    {
    deger=toplam/count;
    *(giden+cerceve_satir*i+j)=deger;
    }
    else
    *(giden+cerceve_satir*i+j)=*(gelen+cerceve_satir*i+j);
    }
    }
}

```

```

void bir_bit_donusumu( unsigned char *orj_frame , unsigned char *konvoled_frame ,
unsigned char *giden , int cerceve_satir , int cerceve_sutun )

```

```

{
for( int i=0 ; i<cerceve_satir ; i++ )
{
for( int j=0 ; j<cerceve_sutun ; j++ )
{

if(*(orj_frame+i*cerceve_satir+j)>*(konvoled_frame+i*cerceve_satir+j))
*(giden+i*cerceve_satir+j) = 1;
else
*(giden+i*cerceve_satir+j) = 0;
}
}
}
}

```

```
int main (void)
{
FILE *stream;
char katar[256];
int min_satir_koor;
int min_sutun_koor;
int min;
int count;
int deger_x;
int deger_y;

unsigned char *BUFFER;
unsigned char *PREVIOUS_FRAME;
unsigned char *CURRENT_FRAME;
unsigned char *CONVOLED_PREVIOUS_FRAME;
unsigned char *CONVOLED_CURRENT_FRAME;
unsigned char *ONEBIT_PREVIOUS_FRAME;
unsigned char *ONEBIT_CURRENT_FRAME;
unsigned char *CROPPED_FRAME;
unsigned char *GECICI;
unsigned char *KATAR;
unsigned char *KERNEL;
char *VEKTOR_X;
char *VEKTOR_Y;
char *TOPLAMSAL_VEKTOR_X;
char *TOPLAMSAL_VEKTOR_Y;
char *YENI_TOPLAMSAL_VEKTOR_X;
char *YENI_TOPLAMSAL_VEKTOR_Y;
char *FARK_VEKTOR_X;
char *FARK_VEKTOR_Y;
BUFFER = (unsigned char *) (BASLANGIC_ADRESI);
PREVIOUS_FRAME = (unsigned char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*1 + 100);
CURRENT_FRAME = (unsigned char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*2 + 100);
CONVOLED_PREVIOUS_FRAME = (unsigned char *)
(BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*3 + 100);
CONVOLED_CURRENT_FRAME = (unsigned char *)
(BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*4 + 100);
ONEBIT_PREVIOUS_FRAME = (unsigned char *)
(BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*5 + 100);
ONEBIT_CURRENT_FRAME = (unsigned char *)
(BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*6 + 100);
CROPPED_FRAME = (unsigned char *) (BASLANGIC_ADRESI +
```

```

VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*7 + 100);
    GECICI      = (unsigned char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*8 + 100);
    KERNEL = (unsigned char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*9 + 100);
    VEKTOR_X = (char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*10 + 100);
    VEKTOR_Y = (char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*10 + 200);
    TOPLAMSAL_VEKTOR_X = (char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*10 + 300);
    TOPLAMSAL_VEKTOR_Y = (char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*10 + 400);
    YENI_TOPLAMSAL_VEKTOR_X = (char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*10 + 500);
    YENI_TOPLAMSAL_VEKTOR_Y = (char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*10 + 600);

    FARK_VEKTOR_X = (char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*10 + 700);
    FARK_VEKTOR_Y = (char *) (BASLANGIC_ADRESI +
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI*10 + 800);

// İLK ÖNCE KERNEL OKUNUYOR
sprintf(katar,"kernel.ucdat");
stream = fopen(katar, "rb" );
fread( BUFFER , sizeof( unsigned char ),
KERNEL_SATIR*KERNEL_SUTUN , stream );
fclose( stream );
// KERNELİ OKUDUK STREAM İ DE KAPATTIK
// BUNDAN SONRA BUFFER İ KERNEL E ATACAZ
for( int i=0 ; i<KERNEL_SATIR ; i++ )
    {
        for( int j=0 ; j<KERNEL_SUTUN ; j++ )
            {
                *(KERNEL+i*KERNEL_SATIR+j) =
*(BUFFER+i*KERNEL_SATIR+j);
            }
    }

    sprintf(katar,"D:/MATLAB6p5/work/ONE BIT
TRANSFORM/KAYNAK/test_dizileri/bike_RAW/bike.%03d.raw",VIDEO_DIZISI
_BASLANGIC_DEGERI);
    stream = fopen(katar, "rb" );
    fread( BUFFER , sizeof( unsigned char ),
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI , stream );
    fclose( stream );
    for( int i=0 ; i<VIDEO_DIZISI_SATIR_SAYISI ; i++ )
        {

```



```

        for( int j=0 ; j<VIDEO_DIZISI_SUTUN_SAYISI ; j++ )
        {
            *(CURRENT_FRAME+i*VIDEO_DIZISI_SATIR_SAYISI+j) =
*(BUFFER+i*VIDEO_DIZISI_SATIR_SAYISI+j);
        }
    }
    for( int cerceve=(VIDEO_DIZISI_BASLANGIC_DEGERI+1) ;
cerceve<=VIDEO_DIZISI_BITIS_DEGERI ; cerceve++ )
    {
        for( int i=0 ; i<VIDEO_DIZISI_SATIR_SAYISI ; i++ )
        {
            for( int j=0 ; j<VIDEO_DIZISI_SUTUN_SAYISI ; j++ )
            {
                *(PREVIOUS_FRAME+i*VIDEO_DIZISI_SATIR_SAYISI+j) =
*(CURRENT_FRAME+i*VIDEO_DIZISI_SATIR_SAYISI+j);
            }
        }

        sprintf(katar,"D:/MATLAB6p5/work/ONE BIT
TRANSFORM/KAYNAK/test_dizileri/bike_RAW/bike.%03d.raw",cerceve);
        stream = fopen(katar, "rb" );
        fread( BUFFER , sizeof( unsigned char
),VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI , stream );
        fclose( stream );
        // SIRADAKİ ÇERÇEVEYİ OKUDUK STREAM İ DE KAPATTIK
        for( int i=0 ; i<VIDEO_DIZISI_SATIR_SAYISI ; i++ )
        {
            for( int j=0 ; j<VIDEO_DIZISI_SUTUN_SAYISI ; j++ )
            {
                *(CURRENT_FRAME+i*VIDEO_DIZISI_SATIR_SAYISI+j) =
*(BUFFER+i*VIDEO_DIZISI_SATIR_SAYISI+j);
            }
        }

        konvole_et( PREVIOUS_FRAME ,
CONVOLED_PREVIOUS_FRAME , KERNEL , VIDEO_DIZISI_SATIR_SAYISI ,
VIDEO_DIZISI_SUTUN_SAYISI , KERNEL_SATIR , KERNEL_SUTUN );

        konvole_et( CURRENT_FRAME ,
CONVOLED_CURRENT_FRAME , KERNEL , VIDEO_DIZISI_SATIR_SAYISI ,
VIDEO_DIZISI_SUTUN_SAYISI , KERNEL_SATIR , KERNEL_SUTUN );

        bir_bit_donusumu( PREVIOUS_FRAME ,
CONVOLED_PREVIOUS_FRAME , ONEBIT_PREVIOUS_FRAME ,
VIDEO_DIZISI_SATIR_SAYISI , VIDEO_DIZISI_SUTUN_SAYISI );
        bir_bit_donusumu( CURRENT_FRAME ,
CONVOLED_CURRENT_FRAME , ONEBIT_CURRENT_FRAME ,
VIDEO_DIZISI_SATIR_SAYISI , VIDEO_DIZISI_SUTUN_SAYISI );

        // ŞİMDİ ONEBIT_CURRENT_FRAME İN SAĞINDAN,

```

```

SOLUNDAN, ALTINDAN, ÜSTÜNDEN KROPLAYIP 244x244 bir imge alcaz
for( int i=15 ; i<(VIDEO_DIZISI_SATIR_SAYISI-16) ; i++ )
{
for( int j=15 ; j<(VIDEO_DIZISI_SUTUN_SAYISI-16) ; j++ )
{
*(CROPPED_FRAME+(i-15)*(VIDEO_DIZISI_SATIR_SAYISI-
31)+(j-15)) =
*(ONEBIT_CURRENT_FRAME+i*VIDEO_DIZISI_SATIR_SAYISI+j);
}
}

min_satir_koor=0;
min_satir_koor=0;
min=1000000;
for( int i=0 ; i<32 ; i++ )
{
for( int j=0 ; j<32 ; j++ )
{
count=0;
for( int ii=0 ; ii<(VIDEO_DIZISI_SATIR_SAYISI-31) ; ii++ )
{
for( int jj=0 ; jj<(VIDEO_DIZISI_SUTUN_SAYISI-31) ; jj++ )
{
if( *(CROPPED_FRAME+ii*(VIDEO_DIZISI_SATIR_SAYISI-
31)+jj) !=
*(ONEBIT_PREVIOUS_FRAME+(ii+i)*VIDEO_DIZISI_SATIR_SAYISI+(jj+j)) )
count++;
}
}
if( count<min )
{
min=count;
min_satir_koor = i;
min_sutun_koor = j;
//min_satir_koor --> dikey vektör
//min_sutun_koor --> yatay vektör
}
}
}

// ŞİMDİ İLGİLİ KARE İÇİN MİN KOORDİNAT NOKTASINI
KAYDA ALIYORUZZ
// AMA ZERO POSITION OLARAK (15,15) NOKTASINI KABUL
EDİYORUZ
*(VEKTOR_X+cerceve-VIDEO_DIZISI_BASLANGIC_DEGERI-
1)=min_sutun_koor-15;
*(VEKTOR_Y+cerceve-VIDEO_DIZISI_BASLANGIC_DEGERI-
1)=min_satir_koor-15;
//deger_x=*(VEKTOR_X+cerceve-
VIDEO_DIZISI_BASLANGIC_DEGERI-1);

```

```

        //deger_y=*(VEKTOR_Y+cerceve-
VIDEO_DIZISI_BASLANGIC_DEGERI-1);
        //printf("\nvektor_x : %d   vektor_y : %d",min_sutun_koor-
15,min_satir_koor-15);
        //printf("\nvektor_x : %d   vektor_y : %d",deger_x,deger_y);

    }

    // ŞİMDİ TOPLAMSAL KONUM VEKTÖRLERİ BULUNUYOR
    for( int i=0 ; i<(VIDEO_DIZISI_BITIS_DEGERI-
VIDEO_DIZISI_BASLANGIC_DEGERI) ; i++ )
    {
        *(TOPLAMSAL_VEKTOR_X+i)=0;
        *(TOPLAMSAL_VEKTOR_Y+i)=0;
        deger_x=0;
        deger_y=0;
        for( int j=0 ; j<=i ; j++ )
        {
            deger_x=deger_x+*(VEKTOR_X+j);
            deger_y=deger_y+*(VEKTOR_Y+j);
        }
        *(TOPLAMSAL_VEKTOR_X+i)=deger_x;
        *(TOPLAMSAL_VEKTOR_Y+i)=deger_y;
        deger_x=*(TOPLAMSAL_VEKTOR_X+i);
        deger_y=*(TOPLAMSAL_VEKTOR_Y+i);
        printf("\ntoplamsal_vektor_x : %d   toplamsal_vektor_y :
%d",deger_x,deger_y);
    }

    // ŞİMDİ SÜZGEÇŞELNMIŞ TOPLAMSAL KONUM VEKTÖRLERİ
    BULUNUYOR
    for( int i=0 ; i<(VIDEO_DIZISI_BITIS_DEGERI-
VIDEO_DIZISI_BASLANGIC_DEGERI) ; i++ )
    {
        deger_x=0;
        deger_y=0;
        count=0;
        for( int j=0 ; j<=N ; j++ )
        {
            if( (i-j)>=0 )
            {
                deger_x=deger_x+*(TOPLAMSAL_VEKTOR_X+i-j);
                deger_y=deger_y+*(TOPLAMSAL_VEKTOR_Y+i-j);
                count++;
            }
        }
        if( count!=0 )
        {
            deger_x=deger_x/count;

```

```

deger_y=deger_y/count;
*(YENI_TOPLAMSAL_VEKTOR_X+i)=deger_x;
*(YENI_TOPLAMSAL_VEKTOR_Y+i)=deger_y;

}
//printf("\nyeni_toplamsal_vektor_x : %d yeni_toplamsal_vektor_y :
%d",deger_x,deger_y);
}

//ŞİMDİ İSE FARK VEKTORLERİ BULUNUYOR
for( int i=0 ; i<(VIDEO_DIZISI_BITIS_DEGERI-
VIDEO_DIZISI_BASLANGIC_DEGERI) ; i++ )
{
*(FARK_VEKTOR_X+i)=*(YENI_TOPLAMSAL_VEKTOR_X+i)-
*(TOPLAMSAL_VEKTOR_X+i);
*(FARK_VEKTOR_Y+i)=*(YENI_TOPLAMSAL_VEKTOR_Y+i)-
*(TOPLAMSAL_VEKTOR_Y+i);
printf("\nfark_vektor_x : %d fark_vektor_y :
%d",*(FARK_VEKTOR_X+i),*(FARK_VEKTOR_Y+i));
}

//ŞİMDİ İSE YENİ İMGELER OLUŞTURULUP DOSYAYA YAZILIYOR
for( int cerceve=(VIDEO_DIZISI_BASLANGIC_DEGERI+1) ;
cerceve<=VIDEO_DIZISI_BITIS_DEGERI ; cerceve++ )
{
// İLK ÇERÇEVEYİ OKUCAZ
sprintf(katar,"D:/MATLAB6p5/work/ONE BIT
TRANSFORM/KAYNAK/test_dizileri/bike_RAW/bike.%03d.raw",cerceve);
stream = fopen(katar, "rb" );
fread( BUFFER , sizeof( unsigned char ),
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI , stream );
fclose( stream );
// İLK ÇERÇEVEYİ OKUDUK STREAM İ DE KAPATTIK
// BUNDAN SONRA BUFFER I CURRENT_FRAME E ATACAZ

for( int i=0 ; i<VIDEO_DIZISI_SATIR_SAYISI ; i++ )
{
for( int j=0 ; j<VIDEO_DIZISI_SUTUN_SAYISI ; j++ )
{
*(CURRENT_FRAME+i*VIDEO_DIZISI_SATIR_SAYISI+j) =
*(BUFFER+i*VIDEO_DIZISI_SATIR_SAYISI+j);
}
}
deger_x=*(FARK_VEKTOR_X+cerceve-2);
deger_y=*(FARK_VEKTOR_Y+cerceve-2);

kaydir(CURRENT_FRAME,GECICI,VIDEO_DIZISI_SATIR_SAYISI,VIDE
O_DIZISI_SUTUN_SAYISI,(-1)*deger_x,(-1)*deger_y);
sprintf(katar,"cikis/bikebike.%03d.raw",cerceve);

```

```
        stream = fopen( katar, "wb" );
        fwrite( GECICI , sizeof( unsigned char ),
VIDEO_DIZISI_SATIR_SAYISI*VIDEO_DIZISI_SUTUN_SAYISI, stream );
        fclose( stream );
        printf("\nSIRA : %d ",cerceve);
    }
}
```

## KAYNAKLAR DİZİNİ

1. ERTURK, S., 2001. Sayısal İşaret İşleme, Birsen Yayınevi
2. ADSP-BF533 EZ-KIT LITE Evalation System Manual, Analog Devices™
3. E. Yaman, S. Ertürk, "Görüntü Stabilizasyonu için Paralel İşlev Gören İki Kalman Filtresiyle İşlem Gürültü Varyansının Adaptifleştirilmesi", 10. IEEE Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SIU'2002) Bildirileri Kitabı, pp. 506-511, Haziran 2002
4. <http://www.analogdevices.com> - Analog Devices™ web site
5. M. K. Güllü, S. Ertürk, "Bulanık Süzgeç ile Görüntü Stabilizasyonu" 11.Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SIU'2003) Bildirileri Kitabı, pp. 156-159.

## ÖZGEÇMİŞ



Halim Cem Kefeli Zonguldak'ta 1983 yılında dünyaya geldi. İlk ve orta öğrenimini Zonguldak Bahçelievler İlköğretim Okulu'nda tamamladıktan sonra lise öğrenimini de yine Zonguldak'ta bulunan Zonguldak Mehmet Çelikel Anadolu Lisesi'nde tamamladı. 2002 yılının Öğrenci Seçme ve Yerleştirme sınavına girerek Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümünde okumaya hak kazandı. Dinamik tabanlı web sayfası tasarımı, mikro işlemciler ve mikrodenetleyicili sistemler, sayısal işaret işleme programlama konularına özellikle ilgi duymaktadır. Üniversite eğitimi boyunca "Lazerle Malzeme İşleme" ve "X-Işını İle Kızılötesi Bölgesi Arasında Yayılan Elektromanyetik Spektrumların Kaydedilmesine Yönelik Spektrometrelerin Tasarımı ve Gerçeklenmesi." gibi birçok projede görev alarak bu konuda çalışmalarda bulundu. 2004 yılının sonlarına doğru Kocaeli Üniversitesi Lazer Teknolojileri Araştırma ve Uygulama Merkezi bünyesinde yazılım geliştirici olarak disiplinler arası işbirliği kapsamında farklı branşlardaki akademisyenlerle çalışmalar yaptı. Halen bu konularda bazı araştırmalar ve çalışmalar yapmakta olup bölümünden 2006 yılının bahar yarıyılında elektronik ve haberleşme mühendisi olarak mezun olmaya hak kazanmıştır.

<http://kulis.kou.edu.tr/cemkefeli/>

[cemkefeli@gmail.com](mailto:cemkefeli@gmail.com)