T.C.
İSKENDERUN TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK ÇEKİRDEKLİ İŞLEMCİLERDE YENİ BİR PARALEL GÖRÜNTÜ İŞLEME YÖNTEMİ VE YÖNTEMİN YÜZ TANIMA PROBLEMİNE UYGULANMASI

Hüseyin ATASOY

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

HATAY
HAZİRAN-2016

T.C.

İSKENDERUN TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK ÇEKİRDEKLİ İŞLEMCİLERDE YENİ BİR PARALEL GÖRÜNTÜ İŞLEME YÖNTEMİ VE YÖNTEMİN YÜZ TANIMA PROBLEMİNE UYGULANMASI

Hüseyin ATASOY

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

HATAY
HAZİRAN-2016

T.C.
İSKENDERUN TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK ÇEKİRDEKLİ İŞLEMCİLERDE YENİ BİR PARALEL GÖRÜNTÜ İŞLEME YÖNTEMİ VE YÖNTEMİN YÜZ TANIMA PROBLEMİNE UYGULANMASI

Hüseyin ATASOY

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Yrd. Doç. Dr. Esen YILDIRIM danışmanlığında hazırlanan bu tez **10/06/2016** tarihinde aşağıdaki jüri üyeleri tarafından **OYBİRLİĞİ** ile kabul edilmiştir.

Yrd. Doç. Dr. Esen YILDIRIM
Başkan

Yrd. Doç. Dr. Yakup KUTLU
Üye

Doç. Dr. Mustafa ORAL
Üye

Kod No: 13

Doç. Dr. Mustafa DEMİRCİ
Enstitü Müdür V.

10/06/2016

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını ve tez üzerinde Yükseköğretim Kurulu tarafından hiçbir değişiklik yapılamayacağı için tezin bilgisayar ekranında görüntülendiğinde asıl nüsha ile aynı olması sorumluluğunun tarafıma ait olduğunu beyan ederim.

Hüseyin ATASOY

ÖZET

ÇOK ÇEKİRDEKLİ İŞLEMCİLERDE YENİ BİR PARALEL GÖRÜNTÜ İŞLEME YÖNTEMİ VE YÖNTEMİN YÜZ TANIMA PROBLEMİNE UYGULANMASI

Artan işlemci hızları ve çekirdek sayıları, kısa süre içerisinde çok fazla işlem yapılmasını gerektiren programların gerçek zamanlı olarak çalıştırılabilmesinin önünü açmıştır. Ancak çekirdek sayılarındaki artış, performans artışından ziyade çok görevliliği hedeflemektedir. Bu tez çalışmasında, gerçek zamanlı görüntü işleme uygulamaları için paralel görüntü işleme ve çerçeve akış hızı sabitleme yaklaşımı sunulmuştur. Bu yaklaşımın temel amacı, görüntü işleme yöntemlerinin sıradan bilgisayarlar üzerinde ve ek donanım desteği gerektirmeden çok çekirdekli işlemci mimarisinden yararlanılarak gerçek zamanlı olarak yürütülebilmesini sağlamaktır. Önerilen yaklaşımın test edilmesi için popüler yüz bulma ve tanıma algoritmaları da çalışılmış ve bu algoritmalar sunulan yaklaşım kullanılarak paralel işleyecek şekilde gerçeklenmiştir. Yaklaşım, Microsoft .NET çatısı altında Microsoft Windows 7 işletim sistemi üzerinde bir Intel i7-3770 3,40 GHz işlemcisi ile test edilmiştir. Sonuçlar görüntü işleme işlemlerinin 4 fiziksel çekirdek üzerine başarıyla dağıtıldığını ve 5,25 kata kadar performans artışı sağlandığını göstermiştir. Önerilen yöntem yüz tanıma problemi üzerinde de test edilmiştir. 25 ve 60 fps hıza sahip videolar için yapılan testler sonucunda, iş yüklerinin çekirdeklere dağıtılabildiği görülmüş ve istenen akış hızları başarıyla sağlanabilmiştir.

2016, 47 sayfa

Anahtar Kelimler: gerçek zamanlı görüntü işleme, paralelleştirme, yüz tanıma

ABSTRACT

A NOVEL PARALLEL IMAGE PROCESSING METHOD ON MULTI-CORE PROCESSORS AND ITS APPLICATION ON FACE RECOGNITION PROBLEM

Increasing processing speeds and number of processors' cores open a new door to run programs that require much processing in real-time. However, the increase in core numbers aims multitasking rather than performance improvement. In this thesis, a novel parallel image processing and frame rate stabilization approach for processing images in real-time is proposed. The main goal of this approach is to run real time image processing tasks on a regular PC with a multi-core CPU without requiring explicit hardware support. Various face detection and recognition algorithms are studied and implemented to run in parallel, using the presented approach. The approach is implemented under Microsoft .NET Framework and tested on Microsoft Windows 7 operating system with an Intel i7-3770 3,40 GHz CPU. Results show that the image processing tasks were successfully distributed to the cores and the performance is increased up to 5,25 times on 4 physical cores. The algorithm is also tested on face recognition task. It was shown that, workloads were successfully distributed to the cores and desired output frame rates were obtained for 25 and 60 fps videos.

2016, 47 pages

Keywords: real-time image processing, parallelization, face recognition

TEŞEKKÜR

Tez çalışmamın her aşamasında yardımlarını, desteklerini ve katkılarını esirgemeyen değerli tez danışmanım Yrd. Doç. Dr. Esen Yıldırım'a ve Doç. Dr. Serdar Yıldırım'a teşekkür ederim…

İÇİNDEKİLER

ÇİZELGELER DİZİNİ

ŞEKİLLER DİZİNİ

SİMGELER ve KISALTMALAR DİZİNİ

SİMGELER

A : Üstel dalgalandırıcıda kullanılan taban değeri

H : Okuma ve yazma işaretçilerinin işaret ettikleri numaralar arasındaki mesafe ile en güveli mesafe arasındaki fark.

M : İşlemciye verilen iş yükü birimi

T_0 : Görüntü okuma periyodu

λ_d : Doğrusal dalgalandırıcı

λ_u : Üstel dalgalandırıcı

KISALTMALAR

BBA : Bağımsız Bileşenler Analizi

CUDA : Compute Unified Device Architecture (Birleşik Hesaplama Aygıtı Mimarisi)

FDAA : Fisher Doğrusal Ayrışım Analizi

fps : Frames per Seconds (saniye başına kare)

GPU : Graphics Processing Unit (Grafik İşleme Ünitesi)

KLT : Kanade-Lucas-Tomasi Yöntemi

LBP : Local Binary Patterns (Yerel İkili Örüntüler)

ms : milisaniye

NDA : Numaralı Dairesel Arabellek

SIFT : Scale Invariant Feature Transform (Ölçekten Bağımsız Öznitelik Dönüşümü)

TBA : Temel Bileşenler Analizi

1. GİRİŞ

Son çeyrek asırda bilgisayarların işlem kapasitelerinin büyük bir hızla artması ile birlikte, görüntülerin gerçek zamanlı olarak işlenmesi gibi uzun süren işlemlerin sıradan bilgisayarlarda yürütülebilmesinin önü açılmıştır. Bilgisayarların işlem güçlerinin artmasına paralel olarak, grafik ve görüntülerin boyutları da artmış, görüntü işleme alanındaki yöntemlerde daha çok veri ile baş edilmesi gerekli ve mümkün hale gelmiştir. Bilgisayarların işlem hızlarının artmasının yanında, son 10 yıl içerisinde işlemcilerin çekirdek sayıları da katlanarak artmış ve temel amacı çok görevliliği sağlamak olan çok çekirdekli işlemciler barındıran bilgisayarlar sıradanlaşmıştır.

İşlemci hızının artması işlem yükü fazla olan bir program için avantaj olsa da programın yapısına müdahale edilmediği sürece çok çekirdekliliğin programın performansına katkısı yoktur. Programların çok çekirdekli işlemcilerde çok çekirdekliliğin avantajından faydalanarak çalışabilmesi için işlemlerin paralel çalışabilecek adımlara bölünmesi gerekir.

Karmaşık ortamlarda gerçek zamanlı yüz tanıma problemi, kısa süre içerisinde çok fazla işlem yürütülmesini gerektiren problemlere örnek olarak gösterilebilir. Bu problem kendi içerisinde, yüz görüntüsünün konumunun tespiti, tespit edilen yüz görüntüsünün daha az boyutlu veri kümeleri ile temsil edilmesi veya öznitelik çıkarımı, sınıflandırma ve karar verme gibi alt problemlere ayrılabilir. Tez kapsamında, yüz tanıma probleminin çözümünde kullanılan yöntemler gibi az zamanda çok fazla işlem yapılmasını gerektiren görüntü işleme yöntemlerinin gerçek zamanlı olarak yürütülebilmesini sağlayan bir paralelleştirme ve çerçeve akış hızı dengeleme yaklaşımı önerilmiştir. Önerilen yaklaşımın gerçek bir problem üzerinde test edilmesi amacıyla yüz bulma ve tanıma problemleri için literatürde en çok kullanılan çözümler incelenmiş, bu çözümler önerilen yaklaşım ile birlikte gerçeklenerek gerçek zamanlı olarak çalışacak şekilde paralelleştirilmiştir. Paralelleştirme için önerilen yaklaşım ile ilgili temel problemler aşağıda detaylandırılmıştır.

1.1. Zamanlayıcı Çözünürlüğü Problemi

Önerilen yöntemin üzerinde gerçekleneceği sistem olan Microsoft Windows, gerçek zamanlı bir işletim sistemi değildir. Yürütülen iş parçacıklarının işlemciyi ne

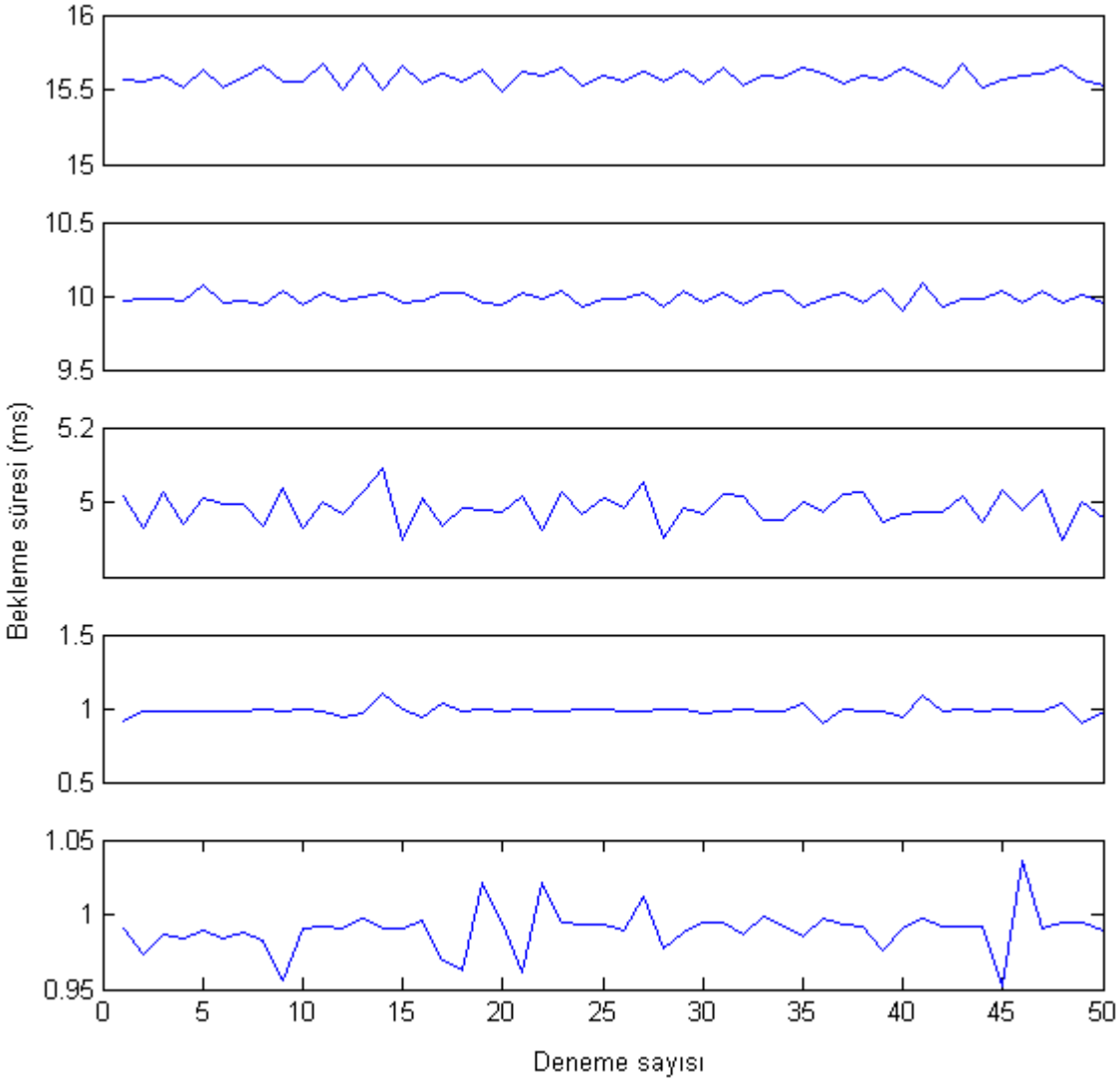
kadar süre boyunca kullanacağı veya tam olarak çalışmaya ne zaman başlayacağı sistem tarafından belirlenmektedir. Sistemin belli bir süreyi tamamlanmış kabul etmesi, zamanlayıcılarının ne kadar zamanlık sürelerle tetiklendiğine bağlı olarak değişmekte ve bu da periyodik işlemler yapan bir iş parçacığının 1 tam periyot süresi dolduğunda işine devam etmesini geciktirebilmektedir. Bu durum, gerçek zamanlı bir görüntü işleme sistemi için ciddi bir sorun teşkil etmekte ve özellikle çerçeve akış hızının sabit tutulmasını zorlaştırmaktadır.

Zamanlayıcı çözünürlüğü, bir zamanlayıcının tetiklenmesi için geçebilecek en küçük süreye verilen isimdir. Bu süre ne kadar düşük olursa zamanlayıcı çözünürlüğü o kadar yüksektir ve yüksek çözünürlüklü zamanlayıcılar yürütmenin gerçek zamanlılığa yaklaşmasını sağlar. Ancak yüksek çözünürlük, zamanlayıcıların tuttukları sürelerin daha sık güncellenmesini sağlayacağından, işlemcinin daha çok güç tüketmesine sebep olur. Bu nedenle bu değerin, yüksek çözünürlüğün gerekli olmadığı durumlarda yüksek seçilmesi yani çözünürlüğün düşük tutulması tercih edilir. Windows'ta zamanlayıcı çözünürlüğünün varsayılan değeri, desteklenen en düşük değer olan 15,6 milisaniyedir.

Düşük zamanlayıcı çözünürlüğünün gerçek zamanlı bir görüntü işleme uygulamasını ne denli olumsuz etkileyebileceği bir örnek senaryo ile açıklanabilir. 25 kare/saniye (frames per second, fps) akış hızı ile aldığı görüntülerin her birini yaklaşık 35 ms kadar bir sürede işleyip hazır hale getiren bir uygulama, bir görüntüyü işledikten sonra 1 tam periyodu (1000/25=40 ms) tamamlamak için 5 ms beklemelidir. Bu uygulamanın Windows işletim sistemi üzerinde koştuğu düşünülürse uygulama, iş parçacığını yaklaşık 5 ms uyutmak istediğinde, işletim sisteminin sürenin geçtiğini tespit edip sırayı tekrar aynı iş parçacığına vermesi yaklaşık olarak zamanlayıcı çözünürlüğü değeri kadar süre geçmesini gerektirecektir. Bu süre Windows'taki varsayılan değer olan 15,6 ms kadar olduğundan, iş parçacığı her iki görüntü arasında yaklaşık olarak 35+15,6=50,6 ms süre geçirecektir. Bu da 25 fps hızında çıkış vermesi beklenen uygulamanın çıkış hızının, işlemci kaynağı yeterli olduğu halde yaklaşık olarak 1000/50=20 fps hızına düşmesine neden olacaktır.

Windows NT işletim sistemi ailesi, ntdll.dll isimli kütüphanesinde, sistemin zamanlayıcı çözünürlüklerinin öğrenilmesini ve değiştirilebilmesini sağlayan NtQueryTimerResolution ve NtSetTimerResolution isimli iki fonksiyon sunmaktadır. İkinci fonksiyon kullanılarak zamanlayıcı çözünürlüğü, sırasıyla, varsayılan ve desteklenen en düşük çözünürlük değeri olan 15,6 ms, 10 ms, 5 ms, 1 ms ve desteklenen

en yüksek çözünürlük değeri olan 0,5 ms değerlerine ayarlanmıştır. Sistem her bir çözünürlük değerinde iken, çalıştığında 50 defa 1 ms boyunca uyutulma isteği gönderen bir iş parçacığı yürütülmüştür. İş parçacığının bu isteği yolladığı an ile işletim sisteminin bekletmeyi sonlandırıp işlemci kullanım hakkını tekrar aynı iş parçacığına verdiği an arasında geçen süreler ölçülüp kaydedilmiştir. Sonuçlar Şekil 1.1'de verilmiştir.



Şekil 1.1. 50 defa 1 ms süre boyunca bekletilme (uyutulma) isteği yollayan bir iş parçacığının, sırasıyla 15,6 ms, 10 ms, 5 ms, 1 ms ve 0,5 ms çözünürlüklerindeki bekletilme süreleri

Şekil 1.1.'de verilen sonuçlar, iş parçacığı her bir denemede yalnızca 1 ms uyutulma isteği gönderdiği halde elde edilmiştir. En düşük çözünürlükte sapmanın yaklaşık 14,5 ms olduğu görülebilmektedir. Bu değer gerçek zamanlıya yakın çalışması

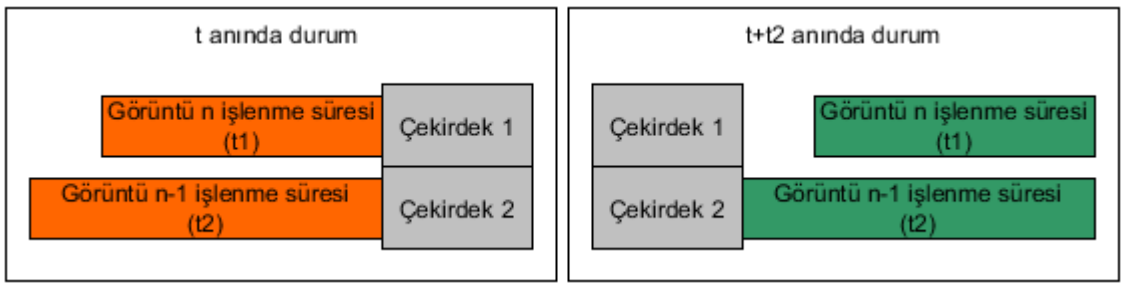
istenen bir uygulama için oldukça yüksektir. Ancak en yüksek çözünürlüklerde (1 ms ve 0,5 ms) sapmalar, sırasıyla 0,1 ve 0,05 ms değerlerini geçmemiştir.

Çözünürlük arttırıldıkça, istenen bekletilme süresi ile geçen süre arasındaki fark azalmaktadır. Çalışmada, yöntemler .NET çatısı altında gerçeklendiğinden ve iş parçacığı uyutma fonksiyonuna girilebilecek en düşük değer 1 ms olduğundan, zamanlayıcı çözünürlükleri, yazılan uygulamada 1 ms değerine ayarlanmıştır.

1.2. Sıraya Bağımlılık Problemi

Birbirinden tamamen bağımsız olan işlemlerin, bir işlemcinin farklı çekirdeklerinde yürütülmesi zaten mümkündür. Yürütülebilmek için başka bir işlemin tamamlanmasını bekleyen işlemlerin ise tamamlanması gereken işlem ile paralel olarak aynı anda yürütülmesi mümkün değildir.

Eğer bir işlem, yürütülmek için başka bir işlemin sonucunu beklemeyecekse, ancak bu iki işlemin sonunda elde edilecek sonuçların sırası önemli ise, bu iki işlem paralel olarak yürütülebilir. Fakat bu durumda elde edilen sonuçların elde edilme sırası ile çıkışa verilmesi halinde sıra bağımlılığı ihlal edilebilir. Çıkışa verilmesi için kendisinden önceki görüntünün çıkışa verilmesini ve sıranın kendisine gelmesini bekleyen bir görüntü, bahsedilen probleme örnek gösterilebilir. n. sırada ekrana yansıtılması gereken görüntü eğer çekirdeklerden birinde (n-1). sırada yer alması gereken görüntüden daha önce işlenip hazır hale gelirse, bu görüntü ekrana yansıtılmadan önce başka bir çekirdekte işlenen diğer görüntünün hazır hale gelmesi için bekletilmelidir. Hangi çekirdeğin görüntüleri hangi sırada alacağı veya hazır hale getireceği önceden kestirilemeyeceğinden, görüntülerin ekrana yansıtılabilmesi için en uzun süren işlemin süresi kadar beklenmek zorunda kalınacaktır.



Şekil 1.2. Sıraya bağımlı iki iş farklı çekirdeklerde paralel yürütüldüğünde geçen süre

Durum, Şekil 1.2.'de 2 çekirdek ve 2 iş ile basitçe örneklenmiştir. Görüntülerin sıralı olarak ekrana yansıtılabilmesi için en uzun süren işin süresi kadar (t2) beklenmelidir. Çekirdek sayısı arttıkça görüntülerin sıralaması daha da karışacaktır.

1.3. Çerçeve Akış Hızı Problemi

Tez kapsamında yazılacak uygulamada, paralel işlenen görüntülerin ekrana hemen yansıtılması ve gerçek zamanlıya yakın bir uygulama yazılması hedeflenmektedir. Dolayısıyla, yukarıda açıklanan sıra problemi çözülse bile, görüntülerin periyodik olarak ve doğru sıra ile ekrana yansıtılması gerekmektedir. 1 tam okuma periyodu dolduğunda, sıradaki görüntü mutlaka hazırda bekliyor olmalıdır. Aksi halde görüntüler istenen akış hızında ekrana yansıtılamayacak ve akıcı bir görüntü elde edilemeyecektir.

Bu nedenlerle tez çalışması kapsamında hazırlanan yüz tanıma sisteminde, görüntülerin birden fazla çekirdek üzerinde paralel yürütülerek çok çekirdekliliğin avantajından faydalanmak için bir yönteme gereksinim duyulmuş ve bahsedilen problemler için aşamalı olarak yeni çözüm yöntemleri önerilmiştir.

2. ÖNCEKİ ÇALIŞMALAR

Paralelleştirme ve yüz tanıma ile ilgili ayrı ayrı yapılmış çok sayıda çalışma bulunmaktadır. Daha önce yüz tanıma üzerine yapılmış çalışmaların bir kısmı yüz görüntüsünü bir bütün olarak ele alırken (Turk ve Pentland, 1991; Belhumeur ve ark., 1997; Bartlett ve Sejnowski, 1997; Lai ve ark., 2001; Ahonen ve ark., 2006), bir kısmı da görüntünün parçalar halinde işlenmesini önermektedir (Wright ve ark., 2009; Liao ve ark., 2013).

Turk ve Pentland (1991) tarafından önerilen öz yüzler yöntemi ile Belhumeur ve ark. (1997) tarafından önerilen Fisher yüzleri yöntemi, başarıları ile öne çıkan ilk yöntemlerdendir. Öz yüz yöntemi, yüz görüntüleri arasındaki doğrusal ilişkileri azaltarak, görüntüleri daha düşük boyutlu yeni uzaylara düşürme işlemine dayanır. Bunun için temel bileşenler analizi (TBA) adı verilen yöntem kullanılır. TBA, kovaryansı minimize etmeyi amaçlarken hem sınıf-içi varysansı, hem sınıflar arası varyansı maksimize eder. Fisher yüzleri yönteminde ise sınıflar arası varyansın sınıf-içi varyansa oranının maksimize edilmesi amaçlanır. Bunun için Fisher doğrusal ayrışım analizi (FDAA) kullanılır. FDAA, TBA'nın aksine sınıf-içi varyansı da düşürmeye yönelik olarak işlediği için sınıflandırma amacıyla kullanıldığında TBA'dan daha iyi sonuçlar vermektedir (Belhumeur ve ark.,1997).

Barlet ve Sejnowski (1996), öz yüzleri bağımsız bileşenler analizine (BBA) tabi tutmuştur. BBA, bağımsız belli sayıda kaynaktan gelen verilerin karışımı olduğu varsayılan işaretleri kaynaklarına ayırmak için kullanılır. Çalışmada, TBA ile elde edilen öz yüzler, bilinmeyen kaynakların doğrusal kombinasyonları olarak kabul edilmiş ve BBA kullanılarak öz yüz sayısı kadar kaynak ve katsayı hesaplanmıştır. Hesaplanan kaynakların seçimi için, FDAA'da olduğu gibi, kaynakların sınıflar arası varyanslarının sınıf içi-varyanslarına oranı hesaplanmış ve en yüksek değerler en iyi olarak kabul edilmiştir. Yöntem TBA'ya göre daha iyi sonuçlar vermiştir.

Başka bir çalışmada Lai ve ark. (2001), spektroyüz adını verdikleri yöntemleri ile yüz görüntülerini dalgacık ve Fourier dönüşümlerinden geçirmiştir. Çalışmalarında, dalgacık dönüşümü sonunda elde edilen görüntülerden düşük frekanslı olanlarının, yüz ifadelerindeki farklılıklara daha az duyarlı olduğunu tespit etmişlerdir.

Bir diğer çalışmada Ahonen ve ark. (2006), yüz görüntülerini parçalara ayırmış ve her parçadan yerel ikili örüntüler (Local Binary Patterns, LBP) çıkarmıştır. LBP'ler,

nxn boyutlarındaki bir matrisin bir görüntü parçasında üzerine geldiği piksellerin, matrisin merkezindeki piksel ile eşiklenmesi sonucunda elde edilir. Çalışmada, farklı boyutlardaki matrisler kullanılarak parçalardan çıkarılan LBP'ler ile sınıflandırma yapılmış, sonuçlar TBA ile karşılaştırılmış ve daha iyi sonuçlar alındığı belirtilmiştir.

Wright ve ark. (2009) çalışmalarında seyrek betimleme yöntemini yüz tanıma problemine uyarlamıştır. Buna göre bir yüz görüntüsünün eğitim setindeki görüntülerin direkt doğrusal kombinasyonu olduğunu varsaymak yerine, bir seyrek doğrusal kombinasyonu olduğu varsayılır ve en iyi seyrek doğrusal kombinasyon, l1-minimizasyon yöntemi ile bulunur.

Liao ve ark. (2013), yüzün hizalanmasını ve tamamının görünmesini gerektirmeden tanıma yapılabilmesini sağlayan bir yöntem geliştirmişlerdir. Yüzü betimlemek için üçlü Gabor desenlerini kullandıkları çalışmalarında, yüzün bir bölümünün eksik olduğu görüntülerde dahi iyi sonuçlar elde etmişlerdir.

Paralel işleme ile ilgili çalışmalara bakıldığında ise donanım tabanlı problem ve çözümlerin daha çok ele alındığı görülmektedir. İşlemlerin grafik işleme ünitelerine yaptırılabilmesini sağlayan Birleşik Hesaplama Aygıtı Mimarisi (Compute Unified Device Architecture, CUDA) gibi teknolojiler, literatürde paralelleştirme problemlerinin bu birimler üzerinde çözülmesine yönelinmesine neden olmuştur (Sinha ve ark., 2006; Chen ve ark., 2007; Castaño-Díez ve ark., 2008; Pham ve ark., 2010; Akil ve ark., 2012; Karas ve ark., 2012; Acharya ve ark., 2014).

Sinha ve ark. (2006) yaptıkları çalışmada Kanade-Lucas-Tomasi (KLT) ve ölçekten bağımsız öznitelik dönüşümü (scale invariant feature transformation, SIFT) yöntemlerini grafik işleme üniteleri (Graphics Processing Unit, GPU) üzerinde paralel çalışacak şekilde gerçeklemiştir. Yöntemlerin performansı, GPU üzerinde yaklaşık 20 kat arttırılmıştır.

Chen ve ark. (2007), çift yönlü süzgeçler gibi kenarları koruyarak çalışan yöntemlerin GPU üzerinde paralel olarak yürütülebilmesini sağlayan çift yönlü bir ızgara yapısı önermiştir.

Bir başka çalışmada Castaño-Díez ve ark. (2008), GPU ve CPU üzerinde çok sayıda görüntü işleme algoritmasını deneyerek performans karşılaştırması yapmıştır. Performans iyileştirmeleri yöntemden yönteme farklılık gösterse de, GPU üzerindeki gerçeklemeler yaklaşık olarak 10-20 kat daha hızlı çalışmıştır.

Genişletilmiş Gauss karışım modeli ile görüntülerde arkaplan ayırma yönteminin GPU üzerinde gerçek zamanlı olarak çalışabilecek hızda gerçeklendiği çalışmada, yöntemin performansı GPU üzerinde en az 10 kat arttırılmıştır (Pham ve ark., 2010).

Akil ve ark. (2012), yazdıkları araç simülasyonunun gerçek zamanlı olarak çalışabilmesi için görüntülerde ton eşleme işlecini GPU'da gerçekleyerek, yöntemin GPU üzerinde yaklaşık 15 kat daha hızlı çalışmasını sağlamıştır.

İstenen açılarla morfolojik açma işleminin GPU üzerinde gerçeklendiği Karas ve ark. (2012) çalışmasında, farklı boyutlardaki görüntülerde GPU ve CPU üzerindeki gerçeklemelerin performanslarını değerlendirmiş ve GPU üzerinde 50 kata kadar performans artışı sağlamıştır.

Ölçekten bağımsız öznitelik dönüşümü (scale invariant feature transformation, SIFT) yönteminin GPU'ya uyarlanarak yazıldığı bir başka çalışmada, yöntemin paralel işlemeye uygun hale getirilmesi için birleştirilmiş çekirdek yöntemini önerilmiştir (Acharya ve ark., 2014). Bu sayede yöntemin performansı %12,2 kadar arttırılmıştır.

Literatürde paralelleştirme ile ilgili yöntemlerin çoğunda, CUDA destekli grafik işleme üniteleri üzerinde çalışıldığı görülmektedir. Yöntemlerin büyük bir kısmında paralelleştirme yönteme özgü olarak yapılmıştır. Dolayısıyla performans artışları da oldukça farklı olmuştur. Çok çekirdekli işlemcilerin kullanıldığı çalışmalarda da yine yöntemlere özgü paralelleştirmeler önerilmiştir (Lu ve ark., 2009; Butari ve ark., 2009; Laurent ve ark., 2010; Hong ve ark., 2011; Sujatha ve ark., 2015). Lu ve ark (2009), en küçük akış maliyeti algoritmasını, Butari ve ark. (2009), ise matrislerde çarpanlara ayırma yöntemlerinden olan Cholesky, LU ve QR yöntemlerini matrisleri veri bloklarına ayırıp paralel işleyerek paralelleştirmiştir. Laurent ve ark. (2010) bir kural madenciliği yöntemi olan GRITE (Gradual Itemset Extraction) yöntemini paralelleştimek için bir yaklaşım sunmuştur. Hong ve ark. (2011) önerdikleri yöntemle, graflarda genişlik öncelikli arama algoritmasını CPU ve GPU üzerinde paralelleştirmiştir. Sujatha ve ark. (2015), bazı popüler sıralama ve arama algoritmalarını paralelleştirip büyük veri setleri üzerinde test ederek sonuçları değerlendirmiştir.

Bu çalışmada, gerçek zamanlı görüntü işleme uygulamalarının çok çekirdekli işlemciler üzerinde ve ek bir donanıma gereksinim duymadan, sağladıkları anlık görüntü akış hızı kararlığını koruyarak paralel çalışmasını sağlayan bir yöntem önerilmiştir.

3. MATERYAL VE YÖNTEM

Tez kapsamında bir paralel görüntü işleme yöntemi önerilmiştir. Bu yöntem gerçek zamanlı yüz tanıma problemine uygulanmıştır. Yüz tanıma probleminin çözümü 3 temel aşamadan oluşmaktadır. İlk aşama görüntülerdeki yüzlerin tespit edilmesi aşamasıdır ve bu aşamada kademeli Haar sınıflandırıcıları kullanılmıştır (Viola ve Jones, 2001; Lienhart ve Maydt, 2002). İkinci aşamada, tespit edilen yüz bölgeleri önce temel bileşenler uzayına düşürülmüştür. Böylece yüzü betimleyen pikseller arasındaki doğrusal ilişkiler elenmiş ve öznitelik matrisinin boyutu azaltılmıştır. Daha sonra temel bileşenler uzayındaki yüz görüntüleri doğrusal ayrışım analizi ile FDAA uzayına düşürülmüştür. Bu sayede sınıflandırma yapılmadan önce yeni uzayda aynı sınıflardaki yüz görüntülerinin birbirlerine mümkün olduğu kadar yakın, farklı sınıflardaki yüz görüntülerinin birbirlerinden uzak olması sağlanmaya çalışılmıştır. Üçüncü aşama, FDAA uzayındaki yüz görüntülerinin eşleştirilme aşamasıdır. Bu aşamalar önerilen yöntem ile birlikte Microsoft .NET ortamında gerçeklenmiştir.

3.1. Önerilen Paralelleştirme Yöntemi

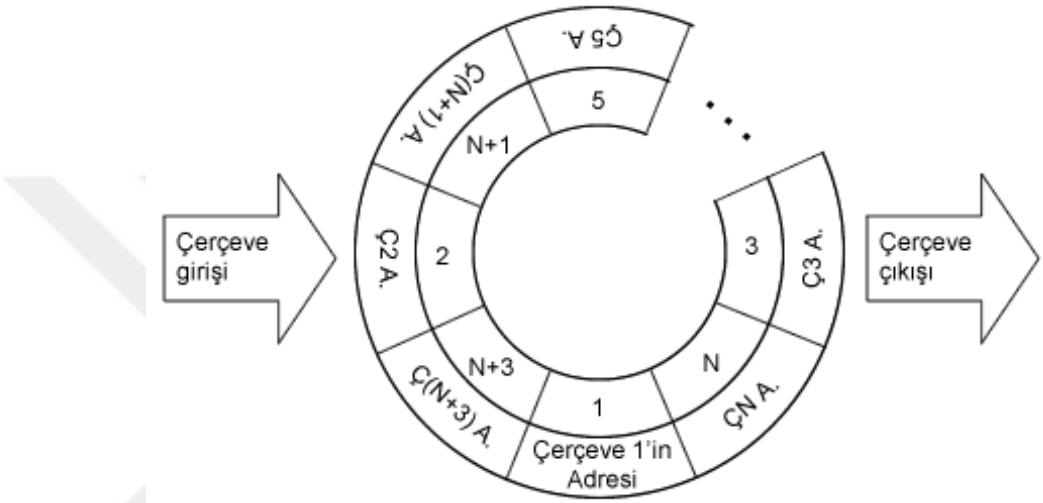
Bu bölümde, gerçek zamanlı görüntü işleme problemi gibi sıraya bağımlı, uzun ve periyodik işlemlerden oluşan problemleri, problemin her bir adımını parçalara ayırmadan ve işleme esnasında adım sırasını önemsemeden işlemci çekirdeklerine dağıtan, ardından işlenen parçaları olmaları gereken sıra ve hızda çıkışa veren bir paralelleştirme yaklaşımı önerilmiştir. Önerilen yaklaşımın temel amacı, görüntü işleme yöntemlerinin sıradan bilgisayarlar üzerinde ek donanım desteği gerektirmeden çok çekirdekliliğin avantajından faydalanılarak gerçek zamanlı olarak çalışabilmesini sağlamaktır.

3.1.1. Sıra Numaralı Dairesel Arabellek

Dairesel arabellek, sabit alanlı bir bellek bölgesinde, bitişin başlangıç tarafından takip ediliyormuşçasına kullanıldığı bir veri yapısıdır. Önerilen yöntemde işlemci çekirdekleri tarafından işlenecek görüntülerin okunma hızında meydana gelebilecek küçük değişimlere karşı bir tampon oluşturmak ve sürekli okuma-yazma işlemleri

yapılacak olan bellek üzerinde işgal edilecek alanın sabit uzunluklu olmasını sağlamak amacıyla dairesel arabellek yapısı kullanılmıştır.

Gerçek zamanlı bir görüntü işleme uygulamasında işlenmiş çerçevelerin ekrana sıralı olarak verilmesi ve dolayısıyla işlenecek çerçevelerin işlendikten sonra tekrar sıraya konması gerekir. Bu nedenle dairesel arabelleğe yazılan çerçevelerin sıra numaraları da bellekte tutulabilmelidir. Bunu sağlamak için iç içe 2 dairesel bellek yapısı kullanılarak sıra numaralı bir dairesel arabellek (NDA) yapısı türetilmiştir.



Şekil 3.1. N hücreli, sıra numaralı dairesel çerçeve arabelleği

Arabellekteki hücrelere yazılan verilerin sıra numaraları ardışık olmak zorunda değildir ve sıra numarasının bir üst sınırı yoktur. Arabelleğe giriş ve çıkışlar okuma işaretçisi ve yazma işaretçisi olmak üzere farklı iki işaretçi yardımıyla yapılır. Şekil 3.1.'de N hücreli olarak gösterilen dairesel arabellekte, bilgisayarın rastgele erişimli belleğine (RAM) atılan her bir çerçevenin bellekteki konumu, dışardaki dairede yer alan ve konumu, yazma işaretçisi yardımıyla tutulan hücreye yazılır. Ardından çerçevenin numarası içteki dairede dıştaki hücreye karşılık gelen iç hücreye yazılır.

Arabellekten okuma yapılırken, iç taraftaki hücrelerde en son okunan veri numarasının bir fazlası aranır. Böylece yazma işlemi esnasında verilerin belleğe sıralı olarak yazılması gerekmeyecek ve ilerleyen kısımlarda detaylı olarak açıklanacak olan yöntem gereğince, arabellekten okunan her çerçeve, aslında sıradaki çerçeve olacaktır.

Eğer herhangi bir nedenden dolayı arabelleğe yazma hızı, arabellekten okuma hızından fazla olursa, bellek dolduğunda okunmamış bir verinin üzerine başka bir veri

yazılmasının önlenmesi için yazma fonksiyonu bellek üzerinde bir değişiklik yapmadan başarısızlıkla sonuçlanmalıdır. Benzer şekilde, okuma hızı yazma hızından fazla olursa, bir süre sonra bellekteki tüm veriler zaten daha önce okunmuş olacağından, okuma fonksiyonu başarısızlıkla sonuçlandırılmalıdır. Bu kontroller için gerekli işaret, okunmuş veya boş olan hücrelerin numaralarına -1 değeri verilerek sağlanabilir. Böylece -1 değerine sahip hücrelerde sadece yazma işlemi, -1'den büyük değerlere sahip hücrelerde ise sadece okuma işlemi yapılır. Çalışmada bu yapı, görüntülerin akışı için kullanılacağından, okuma/yazma hızlarındaki anlık değişimlerden kaynaklanabilecek görüntü kayıpları böylelikle önlenmiş olacaktır.

Açıklanan dairesel arabellek yapısı için bellekte ayrılan alanlara aynı anda birden fazla iş parçacığının müdahale etmesini önlemek amacıyla yazma ve okuma fonksiyonlarının başlangıç kısımları bir karşılıklı dışlayıcı (mutex) ile kilitlenmeli ve bu kilit, fonksiyonlar sonlanırken açılmalıdır.

3.1.2. İşlemci Çekirdekleri ve İş Bölümü

Yöntem, N adet çekirdeği olan bir işlemcinin çekirdeklerini iki gruba ayırır. İlk çekirdek, kontrol çekirdeğidir ve bu çekirdek üzerinde 4 ana iş parçacığı çalışır:

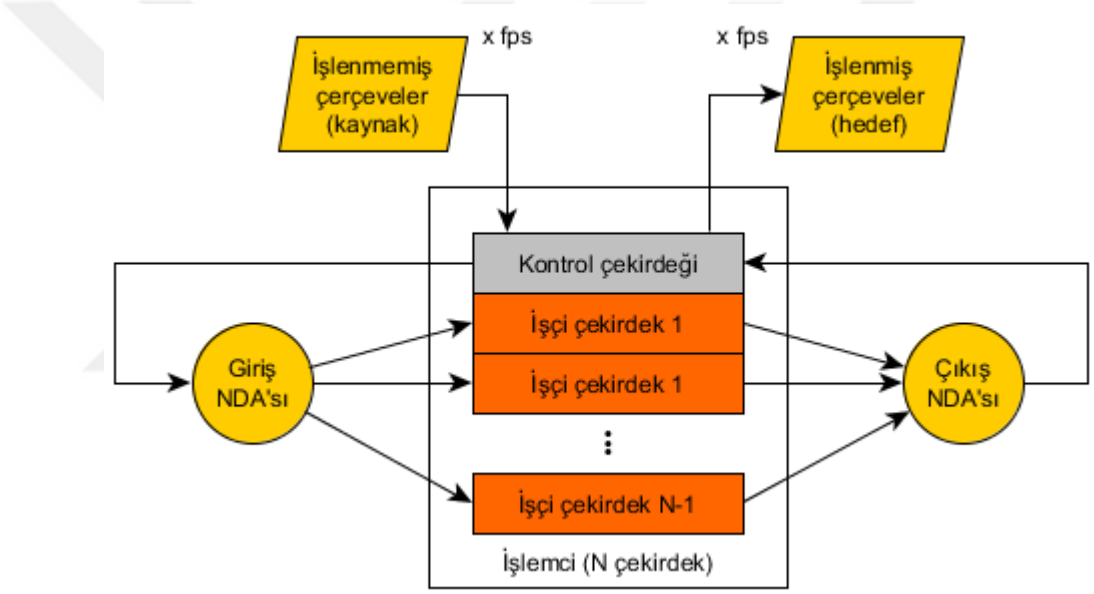
- Girişe yazıcı iş parçacığı: Görüntüleri, görüntü kaynağından x fps hızında okuma ve giriş NDA'sına yazma işlemlerini periyodik (periyot (*T*): 1000/x ms) olarak tekrar eder. Yazma işlemi yukarıda açıklanan nedenden dolayı başarısızlıkla sonuçlanırsa, okuma/yazma periyodu kadar beklendikten sonra yeniden denenir. Bu denemeler yazma işlemi başarılı olana kadar, yani aslında NDA'da bir boşluk açılana kadar tekrar edilir.
- Çıkıştan okuyucu iş parçacığı: İşlenmiş görüntüleri ve/veya bu görüntülerden elde edilen verileri, tutuldukları NDA üzerinden x fps hızı ile okuyup hedefe (örneğin ekrana) gönderir.
- Arayüzden sorumlu iş parçacığı: Uygulamanın ana iş parçacığıdır ve kullanıcı arayüzünün (GUI) yanıt vermesinin kesintisiz olarak sürdürülmesinden sorumludur.
- Değerlendirici iş parçacığı: Gerekiyorsa, yapılan işlemlerle ilgili diğer çekirdeklerin yaptığı ölçümleri değerlendirir. (örneğin çekirdek kullanımlarının

veya bellek kullanımlarının kullanıcı arayüzüne yansıtılması veya kayıt altına alınması)

Diğer çekirdeklerse işçi çekirdekleridir ve bu çekirdekler üzerinde aşağıdaki işlemlerden sorumlu birer iş parçacığı yürütülür:

- Sıradaki görüntünün sıra numarasıyla birlikte giriş NDA'sı üzerinden okunması
- Okunan görüntünün işlenmesi ve işlenmiş görüntü dışındaki verilerin bellekten temizlenmesi
- Okunan görüntünün sıra numarası ile birlikte çıkış NDA'sına yazılması: Burada da kontrol çekirdeğinde yapıldığı gibi, yazma işlemi başarı ile sonuçlanana kadar, okuma/yazma periyoduyla tekrar tekrar denenir.

Çerçevelerin akış şeması Şekil 3.2.'de verilmiştir.

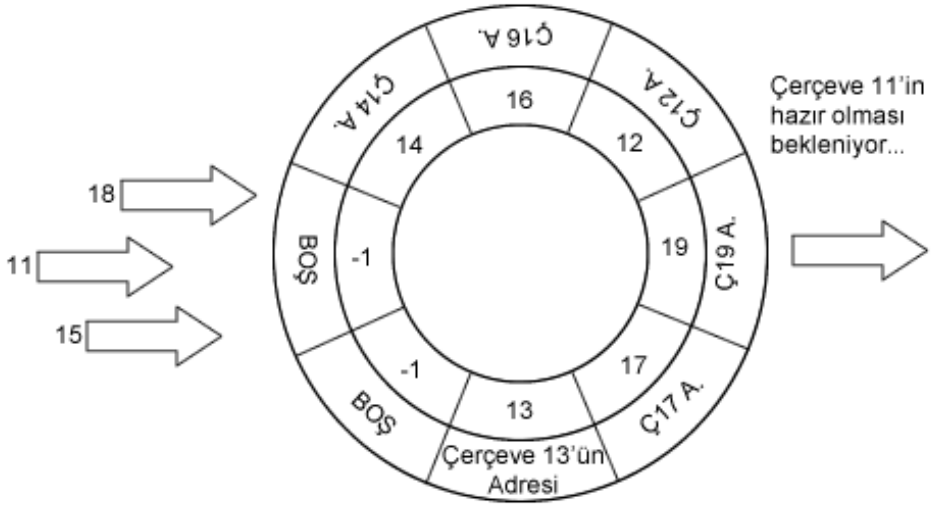


Şekil 3.2. Çerçevelerin arabellekler ve işlemci üzerindeki akış şeması

3.1.3. Çerçeve Akış Hızının Sabitlenmesi

İşçi çekirdeklerinin her biri üzerinde çalışan her iş parçacığı, işlediği görüntüyü bitirir bitirmez çıkış NDA'sına atar ve giriş NDA'sındaki sıradaki görüntüyü okuyarak onu işlemeye başlar. Dolayısıyla giriş NDA'sından okunan çerçevelerin hangisinin, işlemci çekirdeklerinin hangisi tarafından, hangi sıra ile işlenip tamamlanacağı önceden kestirilemez. Bu da periyodik olarak kendisinden veri alınan çıkış NDA'sının işlemci çekirdekleri tarafından sabit periyodlarla beslenemeyeceği anlamına gelir.

Arabellekleme, okumanın gecikmesi halinde çıkıştan okuyucu iş parçacığına bir miktar hazır veri sunulabilmesini sağlasa da, çıkışta her zaman hazırda veri bulunacağı ya da arabelleğin dolmayacağı garanti edilmez. Arabelleğe birden fazla işçi tarafından veri atıldığından ve veriler işlenirken sıra numaraları önemsenmediğinden, bu durum Şekil 3.3.'te gösterilen kilitlenme probleminin ortaya çıkmasına neden olabilir.

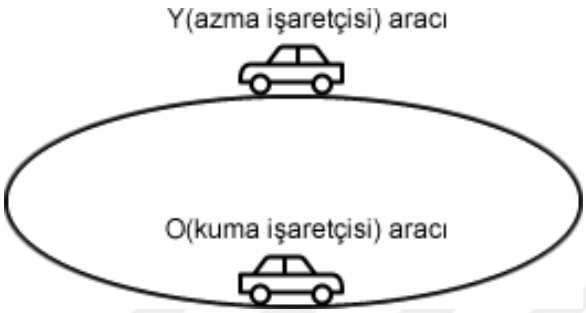


Şekil 3.3. Numaralı dairesel arabelleklere dengesiz veri giriş-çıkışı sebebiyle oluşabilecek bir kilitlenme problemi örneği

Şekil 3.3.'te okuyucu iş parçacığı 11 numaralı çerçeveyi beklemektedir. Yazıcı iş parçacıkları, 11, 15 ve 18. çerçeveleri soldaki okların sırası ile hazır hale getirdiğinde, bellekte sadece 2 boş yer bulunduğundan boş yerlere 18. ve 15. çerçeveler yerleşecektir. 11. çerçeveyi hazır hale getiren iş parçacığı boş yer bulamayacağından sürekli olarak bekleyecek ve 11. çerçevenin hazır hale gelmesi için beklemekte olan okuyucu iş parçacığının işine devam etmesini engelleyecektir. Bu kilitlenme, diğer işçi çekirdeklerde yürüyen iş parçacıklarının da boş yer beklemesine neden olacak ve çerçeve akışı tümüyle kesilecektir.

Yukarıda açıklanan nedenlerden dolayı, okuma ve yazma işaretçilerinin işaret ettiği numaralar arasındaki farkın belli bir aralıkta tutulması ve bu esnada çerçeve akış hızının da korunması gerekmektedir. Problemin ve probleme önerilen çözümümün ifadesi için bir metafor kullanılacaktır. Dairesel bir pistte hareket halinde olan iki araç (Şekil 3.4.) sabit hızlarla ilerlerse araçların çarpışma ihtimali yoktur. Ancak araçlardan biri (O aracı) sabit hızla ilerlerken diğerinin (Y aracı) hızı değişkense araçların çarpışma ihtimalinin olduğu söylenebilir. Bu ihtimalin düşük tutulabilmesi için iki araç arasında

korunabilecek en güvenli mesafe, dairesel pistin uzunluğunun yarısı kadardır. Hızı sabit olan araçtaki şoförün, hızı değişken olan araç üzerinde bir hâkimiyeti yoktur. Ancak güvenli mesafenin korunması için kendi aracını yavaşlatıp hızlandırabilir. Bunu yaparken aracının sabit bir hızla gittiği algısını korumak isterse, çok düşük bir ivme ile hızlanıp yavaşlamalı ve hızını bir aralık ile sınırlamalıdır.



Şekil 3.4. Dairesel pist ve araçlar

Yazma işaretçisi, çekirdeklerin herhangi biri işlediği görüntüyü bitirip arabelleğe attığında yer değiştirdiğinden, yazma işaretçisinin hızı çekirdeklerin hızlarına bağlıdır. Yazma işaretçisinin hızına ancak işçi çekirdeklerinin yazma işlemini geciktirmeleri sağlanarak müdahale edilebilir. Bu da performans kaybı anlamına gelebilir. Ancak çıkıştan okuma işlemi kontrol çekirdeğinin denetimi altında olduğundan, görüntülerin işlenme hızı düşürülmeden, okuma işaretçisinin hızına müdahale etmek mümkündür. Okuma işaretçisi periyodik olarak yer değiştirdiğinden işaretçinin hızı, periyoda müdahale edilerek değiştirilmelidir (3.1).

$$t_b = max\left(\left(T_0 - t_g\right) - T_0\lambda, 0\right) \tag{3.1}$$

Burada t_b yeni bir okuma işlemine başlanmadan önce beklenmesi gereken süreyi, T_0 okuma periyodunu, t_g son yapılan okuma eylemi esnasında geçirilen süreyi temsil etmektedir. λ, periyodu arttırıp azaltarak dalgalandırmak için kullanılacak ve dalgalandırıcı olarak anılacaktır.

3.1.3.1.Doğrusal Dalgalandırıcı

Dalgalandırıcının hedefi H (3.3), okuma periyodunu dalgalandırarak okuma ve yazma işaretçilerinin işaret ettiği alanların numaraları (sırasıyla X_o ve X_y) arasındaki

mesafenin (3.2), en güvenli değeri olan NDA'nın kapasitesinin yarısı ($N/2$) kadar tutulmasıdır. O halde *H*'nin en uygun değeri sıfırdır. *H* pozitif veya negatif hesaplandığında, periyoda yapılan müdahale de doğrusal dalgalandırıcı (3.4) ile orantılı olarak sırasıyla negatif veya pozitif olacak ve böylece okuma işlemi hızlandırılıp yavaşlatılarak *H* sıfırda tutulmaya çalışılacaktır.

$$X = X_y - X_o \tag{3.2}$$

$$H = X - \frac{N}{2} \tag{3.3}$$

$$\lambda_d = \frac{H}{\frac{N}{2}} = \frac{2(X_y - X_o)}{N} - 1 \tag{3.4}$$

Daha önce verilen dairesel pist metaforunda açıklandığı gibi, hızdaki değişimlerin fark edilmeyecek kadar az olması ve hızın belli sınırlarda tutulması gerektiğinden, periyoda yapılacak müdahalelerin, hızı belli sınırlarda tutacağından emin olunmalıdır. *X*'in en büyük değeri *N*, en küçük değeri 0'dır. Böylece *H*'nin en büyük değeri *N/2*, en küçük değeri *–N/2* olur. *H*, *N/2* değerine bölünerek [-1,1] aralığına çekilir ve ardından dalgalandırıcı olarak kullanılır. Bekleme süresindeki değişim (t_b) en çok T_0 kadar olacağından, hızdaki değişim de sınırlandırılmış olacaktır. Dalgalandırıcı, kullanılmadan önce başka bir sayıya bölünerek daha dar bir aralığa çekilip hızdaki değişim daha dar bir sınırda tutulabilir.

3.1.3.2.Üstel Dalgalandırıcı

Çalışmanın devamında, doğrusal dalgalandırıcının periyoda sert müdahalelerde bulunabildiği gözlenmiştir. Bunun çıkış hızına etkisi, 4. bölümde daha detaylı olarak tartışılacaktır. İşlemci çekirdekleri işledikleri görüntüleri yaklaşık olarak aynı anda NDA'ya yazdıklarında, okuma ve yazma işaretçilerinin işaret ettiği alanların numaraları arasındaki fark çekirdek sayısının 1 eksiği kadar artmaktadır. Bu durum, farkın yüksek olduğu bir ana denk gelirse periyot çok fazla dalgalanacağından, çıkış hızına anlık yükselişler olarak yansıyacaktır.

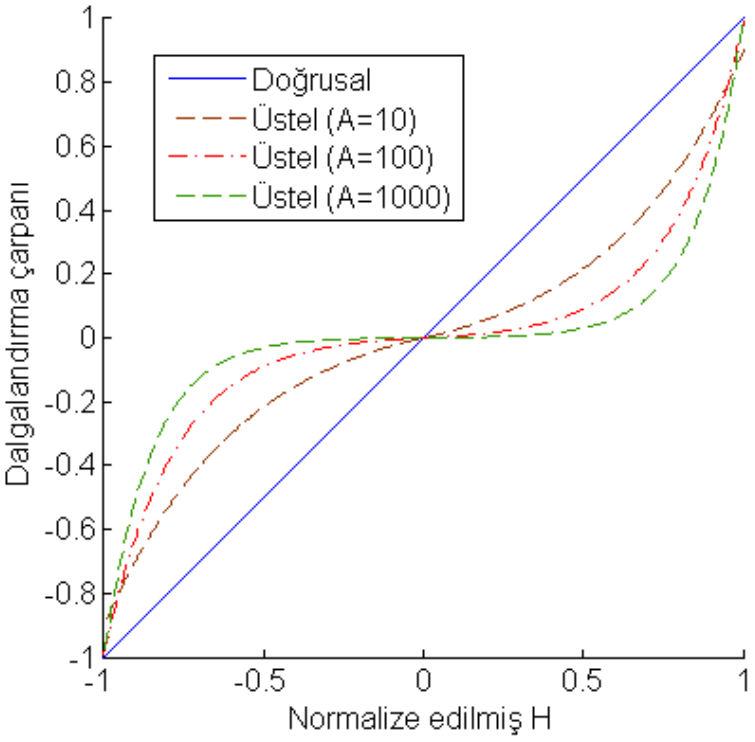
Çekirdeklerin neredeyse aynı anda NDA'ya veri yazma ihtimali gerçekleştiğinde, işaretçilerin işaret ettiği numaralar arasındaki farkta meydana gelen

dalgalanmaya anında sert tepkiler verilmemesi için daha yumuşak bir dalgalandırıcı daha önerilmiştir:

$$\lambda_u = \left(A^{|\lambda_d|-1} - A^{-1}\right) \operatorname{sgn}(\lambda_d) \tag{3.5}$$

Doğrusal dalgalandırıcının değeri (λ_d) [-1,1] aralığında olduğundan, mutlak değeri bir A sabit tabanının ($A > 1, A \in \mathbb{Z}^+$) üzerine alındığında [1,A] aralığında doğrusal olmayan değerler döndürecektir. Bu aralık önce üsten 1 çıkarılarak [A^{-1},1] aralığına ve daha sonra sonuçtan A^{-1} değeri çıkarılarak [$0, 1 - A^{-1}$] aralığına çekilir. Dönen sonuç doğrusal dalgalandırıcının işareti ile çarpılır ve böylece periyodun ne yönde (negatif veya pozitif) değiştirileceğinin bilgisi çarpana geri kazandırılır. Elde edilen sonuç üstel dalgalandırıcı (λ_u) olarak kullanılır.

A sabiti, doğrusal dalgalandırıcı doğrusunu bükerek, işaretçilerin işaret ettiği numaralar arasındaki mesafe ile en uygun mesafe arasındaki fark düşükken, üstel dalgalandırıcının düşük bir değerde olmasını sağlayıp periyoda sert müdahalelerde bulunulmasını önleyecektir. Ancak fark arttıkça üstel dalgalandırıcının değeri artarak artacak ve çok kritik durumlarda (arabellek dolmak veya boşalmak üzereyken) sert müdahalelerden kaçınılmamasını sağlayacaktır.

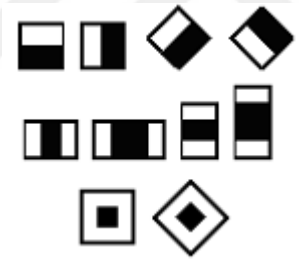


Şekil 3.5. Normalize edilmiş H (işaretçilerin işaret ettiği alanların numaraları arasındaki mesafe ile en güvenli mesafe arasındaki fark) değerleri ve dalgalandırıcı eğrileri

A değeri daha yüksek seçilerek, eğrinin daha da bükülmesi mümkündür. Ancak çok yüksek *A* değerleri, işaretçilerin işaret ettiği numaralar arasındaki fark kritik değerlere yaklaştığında okuma periyoduna yapılacak yumuşak müdahalenin yetersiz kalmasına neden olup arabelleğin tamamen boşalmasına veya dolmasına neden olabilir. Normalize edilmiş *H* değerleri ve dalgalandırıcı eğrilerinin grafiği Şekil 3.5.'te verilmiştir.

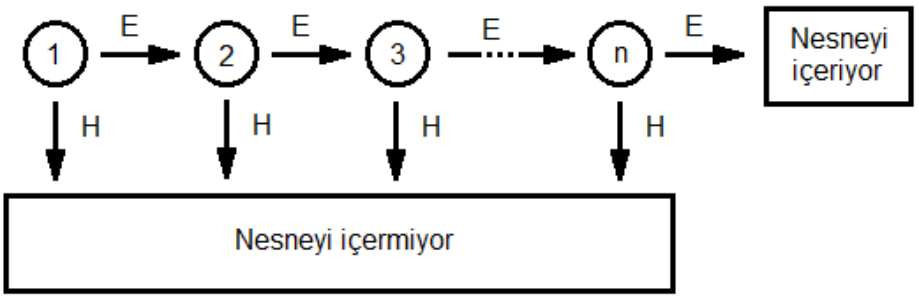
3.2. Yüz Konumunun Tespit Edilmesi

Yüz konumlarının tespiti için Viola ve Jones tarafından önerilen ve Lienhart tarafından geliştirilen kademeli Haar sınıflandırıcıları yöntemi kullanılmıştır (Viola ve Jones, 2001; Lienhart ve Maydt, 2002). Kademeli Haar sınıflandırıcıları, görüntüleri Haar benzeri dikdörtgensel öznitelikler (Şekil 3.6.) kullanarak basit sınıflandırıcılardan meydana gelmiş güçlü bir sınıflandırıcı zinciri ile sınıflandırır.



Şekil 3.6. Dikdörtgensel öznitelik örnekleri

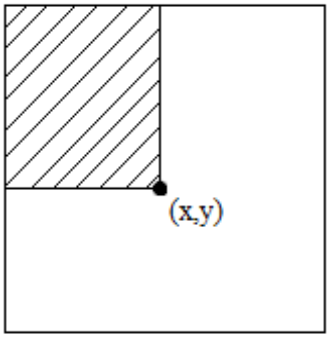
Dikdörtgensel öznitelikler, görüntü üzerinde farklı ölçeklerle gezdirilerek sınıflandırıcının karar vermesini sağlayacak değerler hesaplar. Bu değerler, dikdörtgensel özniteliklerin siyah kısımları ile beyaz kısımları altında kalan piksellerin ortalamasının, bu kısımların ağırlıkları ile çarpılıp toplanması ile elde edilir. Hesaplanan öznitelikler, eğitim aşamasında belirlenen bir eşik değer ile karşılaştırılarak incelenen bölgenin sınıflandırıcıyı geçip geçmediğine karar verilir. Öznitelikler, ağırlıklar ve eşik değerleri eğitim aşamasında belirlenir. Zayıf sınıflandırıcıların birinden geçemeyen bir bölge, geri kalan sınıflandırıcılar tarafından test edilmeden doğrudan elenir (Şekil 3.7.).



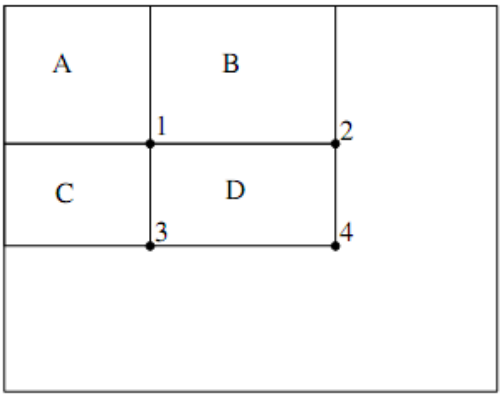
Şekil 3.7. Zayıf sınıflandırıcılarla elde edilen sınıflandırıcı zinciri

Dikdörtgensel öznitelikler, toplam görüntü (3.6) (Şekil 3.8.) yöntemi kullanılarak çok hızlı bir şekilde hesaplanır. Toplam görüntü matrisi görüntüdeki piksellerin artımlı olarak toplanması ile elde edilir.

$$tg(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} g(x',y') \tag{3.6}$$



Şekil 3.8. Bir (x,y) noktasındaki toplam görüntü



Şekil 3.9. Belli bölgelerdeki piksel toplamlarının toplam görüntü yöntemi ile hesaplanması

Şekil 3.9.'da, 1 noktasındaki toplam görüntü değeri, A bölgesindeki piksellerin toplamına eşittir. Benzer şekilde 2 noktasında A+B, 3 noktasında A+C ve 4 noktasında A+B+C+D olarak hesaplanır. Sadece D bölgesindeki piksellerin toplamı gerektiğinde, noktalarda hesaplanan toplam görüntü değerleri yardımıyla bu toplam, 4+1-(2+3) şeklinde çok hızlı bir biçimde hesaplanabilir (Viola ve Jones, 2001). Bunun için gerekli olan toplam görüntü matrisi aşağıdaki ifadelerle hesaplanır:

$$bt(x,y) = \begin{cases} 0, & y = 0 \\ bt(x,y-1) + g(x,y), & y > 0 \end{cases} \tag{3.7}$$

$$tg(x,y) = \begin{cases} 0, & x = 0 \\ tg(x-1,y) + bt(x,y), & x > 0 \end{cases} \tag{3.8}$$

Bu ifadelerde *bt*, birikimli satır toplamını, *g* görüntüyü, *tg* toplam görüntüyü temsil etmektedir.

Kademeli Haar sınıflandırıcıları eğitilirken AdaBoost (Freund ve Schapire, 1997) yöntemi kullanılır. Yöntem şu şekilde çalışır:

1. a ve b, sırasıyla pozitif ve negatif örneklerin sayısı, y_i pozitif örnekler için 1, negatif örnekler için 0 ve eğitim kümesi $Z = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$ olsun.
2. Ağırlıkların başlangıç değerleri $w_i^{(0)}$, pozitif örnekler için $1/2a$, negatif örnekler için $1/2b$ olarak alınır.
3. Her bir sınıflandırıcı eklenirken:
   a. $m = m + 1$
   b. Ağırlıklar normalize edilir:

   $$w_i^{(m)} = \frac{w_i^{(m)}}{\sum_{j=1}^{n} w_j^{(m)}} \tag{3.9}$$

   c. Her bir öznitelik için bir h_j sınıflandırıcısı eğitilir. Ağırlık değerlerine göre sınıflandırıcının hata oranı şöyle hesaplanır:

   $$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right| \tag{3.10}$$

   d. En düşük hataya sahip sınıflandırıcı h_m seçilir. Eğer x_i doğru sınıflandırılmışsa $e_i = 0$, yanlış sınıflandırılmışsa $e_i = 1$ ve $\beta_m = \frac{\varepsilon_m}{1-\varepsilon_m}$ olmak üzere ağırlıklar şu ifade ile güncellenir:

$$w_i^{(m+1)} = w_i^{(m)} \beta_m^{1-e_i} \tag{3.11}$$

4. Sınıflandırıcılar eğitildikten sonra, $\alpha_m = \log \frac{1}{\beta_m}$ olmak üzere şöyle bir sınıflandırıcı dizisi elde edilir:

$$h(x) = \begin{cases} 1 & \sum_m \alpha_m h_m(x) \geq \frac{1}{2} \sum_m \alpha_m \\ 0 & diğer\ durumlar \end{cases} \tag{3.12}$$

3.3. Özniteliklerin Elde Edilmesi

Yüz konumları, kademeli Haar sınıflandırıcıları ile tespit edildikten sonra, yüz görüntülerinin etiketlenebilmesi için var olan görüntülerle eşleştirilmesi gerekmektedir. Bu aşamada temel bileşenler analizi (Turk ve Pentland, 1991) ve Fisher doğrusal ayrışım analizi (Belhumeur ve ark., 1997) yöntemleri kullanılmıştır. Bu iki yöntem, piksel değerleri ile temsil edilen yüz görüntülerini başka uzaylara taşıyan doğrusal dönüşümlerin bulunmasında kullanılır.

3.3.1. Temel Bileşenler Analizi

Temel bileşenler analizi (TBA), bir veri kümesinin daha az sayıda veri ile temsil edilebilmesini amaçlayan bir yöntemdir. TBA ile verilerin varyanslarının en yüksek olduğu doğrultuda uzanan yeni eksenler hesaplanır. Veriler bu eksenlere düşen koordinatlarla temsil edilir. Böylece verileri temsil eden değerler orijinal uzaylarındayken aralarındaki doğrusal ilişkiler en aza indirilmeye çalışılır (Fukunaga, 2013).

Temel bileşenler analizine, veriler bulundukları uzayda ortalanarak başlanır. Verilerin ortalanması veya sıfır ortalamalı hale getirilmesi, belirlenecek eksenlerin verilerin merkezinden geçmesi açısından önemlidir. Verilerin ortalanması için örneklerin ortalaması her bir örnekten çıkarılır.

X, örneklerin sütunlar halinde yer aldığı veri seti matrisi, *n* veri setindeki örnek sayısı olsun:

$$X = [x_1, x_2, \dots, x_n] \tag{3.13}$$

Veri setinin bulundukları uzayda ortalanması için her bir örnekten bütün örneklerin ortalaması çıkarılır (3.14) ve sıfır ortalamalı örnekler matrisi (3.15) elde edilir:

$$\bar{x}_i = x_i - \frac{1}{n}\sum_{j=1}^{n} x_j \tag{3.14}$$

$$\bar{X} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n] \tag{3.15}$$

Elde edilen sıfır ortalamalı örnekler matrisi, kendi transpozu ile çarpılır ve böylece köşegenlerinde varyansları ve diğer elemanlarında kovaryansları taşıyan kovaryans matrisi elde edilir:

$$C = \bar{X}\bar{X}^T \tag{3.16}$$

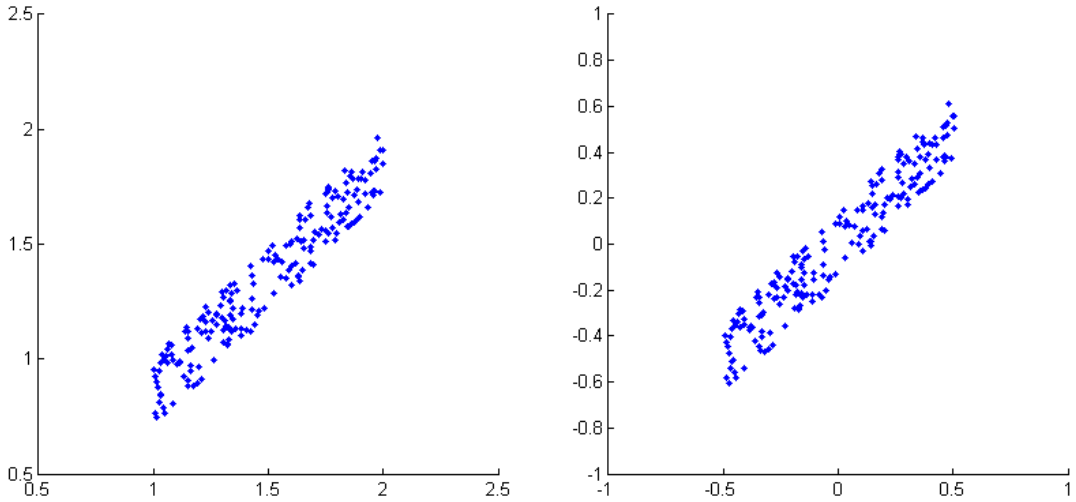
Kovaryans matrisinin öz değer ve öz vektörleri hesaplanarak, varyansların minimum olmasını sağlayacak eksenlere ulaşılır:

$$CV_i = \lambda_i V_i \tag{3.17}$$

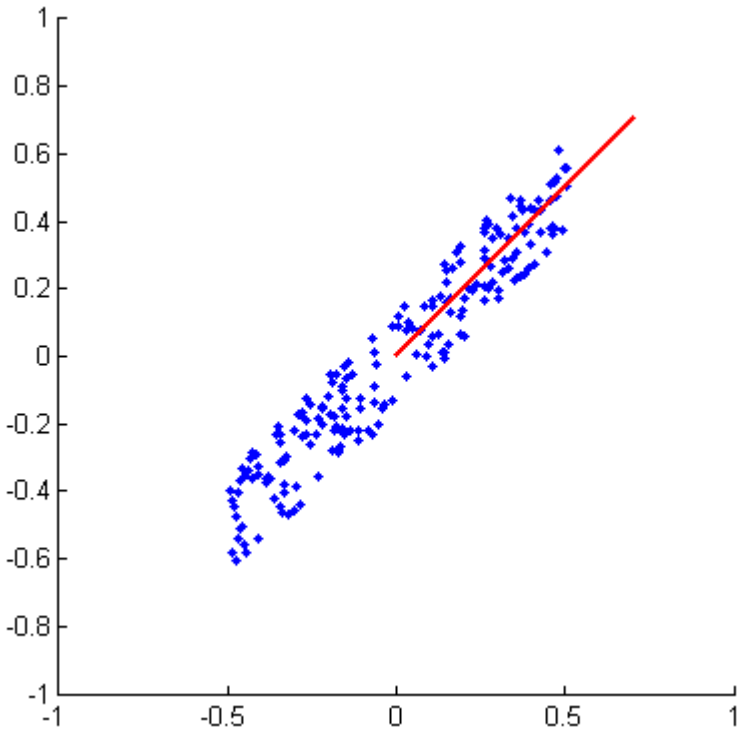
Burada V_i temel bileşenler olan öz vektörleri, λ_i karşılık gelen öz değerleri temsil etmektedir. V_i vektörlerinin sütunlar halinde yazılmasıyla elde edilen W matrisi, örnekleri yeni uzaya düşüren (3.18) dönüşüm matrisidir.

$$x_i' = W^T \bar{x}_i \tag{3.18}$$

Küçük öz değerler düşük varyanslı eksenlere karşılık geldiğinden, öz değerler ve karşılık gelen öz vektörler büyükten küçüğe doğru sıralanır. En yüksek varyans, veriler ilk öz vektör üzerine düşürüldüğünde elde edilir. Bu nedenle sıralamadan sonra ilk öz vektör 1. temel bileşendir ve dağılımı tek başına en iyi temsil edebilecek bileşen kendisidir. Diğer öz vektörler, veri setinin temsilinde önemi giderek azalan diğer temel bileşenlerdir. Sıralamanın sonlarına doğru temsil edilen varyans düştüğü için düşük varyanslı öz vektörler atılarak veri setinin yeni uzaydaki boyutu büyük oranda düşürülebilmektedir.



Şekil 3.10. Bulunduğu uzayda ortalanmadan önceki ve sonraki örnek veri kümesi



Şekil 3.11. Örnek veri kümesi üzerinde temel bileşenler analizi ile bulunan ilk temel bileşenin doğrultusu

Bir veri seti örneği oluşturularak bu veri setinin bulunduğu uzayda ortalanmadan önceki ve ortalandıktan sonraki görüntüsü Şekil 3.10.'da verilmiştir. Bu veri seti üzerinde temel bileşenler analizi uygulanarak, varyansın en yüksek olacağı yeni eksenin, yani 1. temel bileşenin doğrultusu hesaplanmıştır. Şekil 3.11.'de ilk temel bileşenin, verilerin en dağınık olduğu doğrultuda uzandığı görülmektedir.

3.3.2. Fisher Doğrusal Ayrışım Analizi

Temel bileşenler analizi, bir veri kümesinin daha az sayıda veri ile temsil edilebilmesini amaçlayan bir yöntemdir. Ancak bu yöntemin verilerin ayırt edilebilirliği üzerine bir etkisi bulunmamaktadır. Temel bileşenler analizi ile varyansın yüksek olduğu doğrultular tespit edilirken, Fisher doğrusal ayrışım analizi (FDAA) ile sınıflar arası varyansların yüksek, sınıf içi varyansların düşük olduğu doğrultuların tespit edilmesi hedeflenir. Böylece veriler daha kolay ayırt edilebilecekleri yeni uzaylara düşürülür (Welling, 2005). Bu amaçla sınıflar arası dağılımların toplamının (3.19) sınıf içi dağılımların toplamına (3.20) oranının en büyük değerinde olmasını sağlayacak W matrisi hesaplanır (3.21).

$$D_a = \sum_{k=1}^{s}(\mu_k - \mu)(\mu_k - \mu)^T \tag{3.19}$$

$$D_i = \sum_{k=1}^{s}\sum_{j=1}^{N_k}(x_{kj} - \mu_k)(x_{kj} - \mu_k)^T \tag{3.20}$$

$$\underset{W}{\operatorname{argmax}} \frac{W^T D_a W}{W^T D_i W} \tag{3.21}$$

Bu oran, W matrisi , $D_i^{-1}D_a$ ifadesinin öz vektörlerinden meydana geliyorken en büyük değerindedir. Dolayısıyla bu ifade genelleştirilmiş öz değer denklemi çözülerek öz değer ve öz vektörlerine ayrıştırılır ve W matrisi elde edilir (3.22) (3.23).

$$D_a W = D_i W \Lambda \tag{3.22}$$

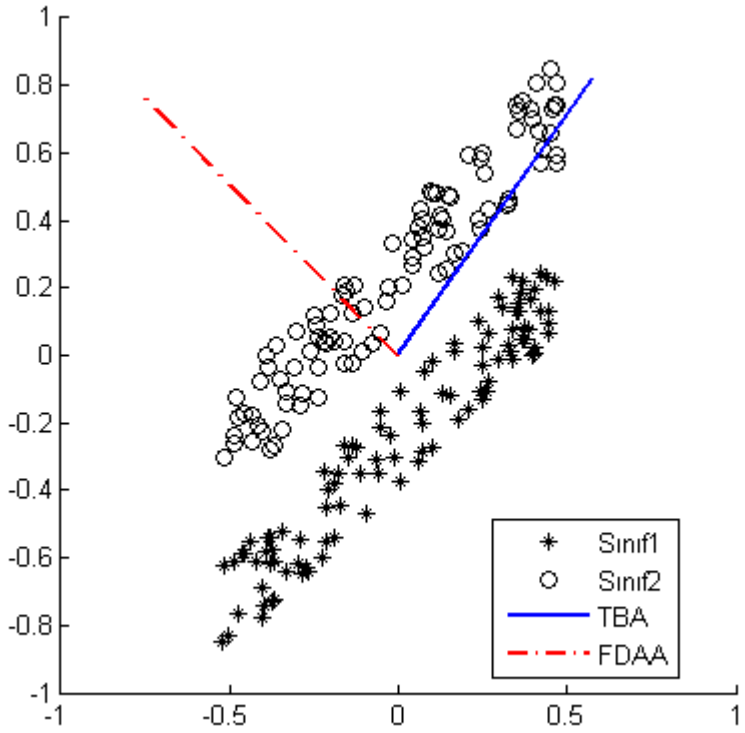
$$D_i^{-1} D_a W = W \Lambda \tag{3.23}$$

İfadedeki Λ, W öz vektörlerine karşılık gelen öz değerleri barındıran köşegen matristir:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \lambda_m \end{bmatrix} \tag{3.24}$$

İki sınıfa ait farklı iki veri kümesi oluşturularak üzerlerinde TBA ve FDAA uygulanmıştır. TBA ile elde edilen ilk eksen, TBA'nın amacına uygun olarak verilerin

en dağınık olduğu doğrultuda yer almaktadır (Şekil 3.12.). FDAA ile bulunan eksen ise, veriler bu eksen düştüğünde sınıfların ortalamalarının birbirine mümkün olduğu kadar uzak kalmasını sağlamaktadır.



Şekil 3.12. İki sınıftan oluşan bir örnek veri kümesi üzerinde temel bileşenler analizi ve Fisher doğrusal ayrışım analizi ile bulunan yeni eksenlerin doğrultuları

3.4. Yüzlerin Eşleştirilmesi

Yüz görüntülerini temsil eden pikseller, TBA ve/veya FDAA yöntemleri ile çok daha az boyutu olan yeni uzaylara düşürüldükten sonra, her bir yüz görüntüsü bu uzaylarda bir noktadır. Bu noktaların birbirlerine olan uzaklıkları benzerlik ölçüsü olarak ele alınabilir.

Veritabanı, eğitim verisi olarak kullanılacak yüz görüntüleri üzerinde TBA veya FDAA uygulanarak elde edilen öznitelikleri içerir. Yeni bir yüz görüntüsünün veritabanındaki görüntülerden biri ile eşleştirilebilmesi için yeni görüntü aynı yöntemlerle veritabanındaki örneklerin düşürüldüğü uzaya düşürülür. Ardından aynı uzayda birer nokta ile temsil edilen yeni yüz görüntüsü ile veritabanındaki yüz görüntüleri arasındaki Öklid uzaklıklarına bakılarak yeni yüz görüntüsünün sınıfına karar verilir.

Yeni bir yüz görüntüsü kendisine en yakın yüz görüntüsü ile eşleşmiş kabul edilirse, bütün yeni yüzler için mutlaka bir eşleşme bulunur. Oysa veritabanında yer almayan bir kişinin yüz görüntüsünün hiçbir görüntü ile eşleşmemesi gerekir. Bu nedenle yüz görüntülerinin veritabanındaki yüzlerden biri ile eşleşmiş kabul edilmesi için 2 yüz görüntüsü arasındaki Öklid uzaklığının belli bir eşik değeri aşıp aşmadığı kontrol edilmelidir. Ancak eşik değerin sabit alınması, veritabanındaki yüz görüntülerine yeni yüzler eklendikçe eşleştirme duyarlılığının değişmesine neden olacaktır. Bu sebeple tez çalışmasında veritabanındaki örneklerin ortalaması ve standart sapması dikkate alınarak eşik değerin değişken olması sağlanmıştır. Buna göre öncelikle veritabanında yer alan her bir sınıfın örneklerinin birbirlerine uzaklıklarının ortalaması (μ_i) ve standart sapması (σ_i) ayrı ayrı hesaplanır:

$$\mu_i = \frac{1}{\mathrm{C}(N_i,2)} \sum_{j=1}^{N_i} \sum_{k=j+1}^{N_i} \left\| x'_k - x'_j \right\| \tag{3.25}$$

$$\mathrm{C}(N_i, 2) = \frac{N_i!}{2(N_i-2)!} \tag{3.26}$$

$$\sigma_i = \sqrt{\frac{1}{N_i} \sum_{j=1}^{N_i} \left(x'_j - \mu_i\right)^2} \tag{3.27}$$

Yeni bir yüz görüntüsü veritabanındaki yüz görüntülerinden biri ile eşleştirilmek istendiğinde, yeni görüntü (*y*) yeni uzaya düşürüldükten sonra (*y'*), yeni görüntünün diğer her bir sınıfın (S_i) elemanlarına uzaklıklarının ortalamaları (d_i) ayrı ayrı hesaplanır:

$$d_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \left\| y' - x'_j \right\| \quad , \quad x_j \in S_i \tag{3.28}$$

Yeni görüntüye en yakın görüntünün sınıfı, (3.29)'daki ifadeyi de sağlaması koşulu ile yeni görüntünün sınıfı olarak kabul edilir. Eğer aşağıdaki koşul sağlanmazsa, yeni yüz görüntüsünün hiçbir görüntü ile eşleşmediği varsayılır.

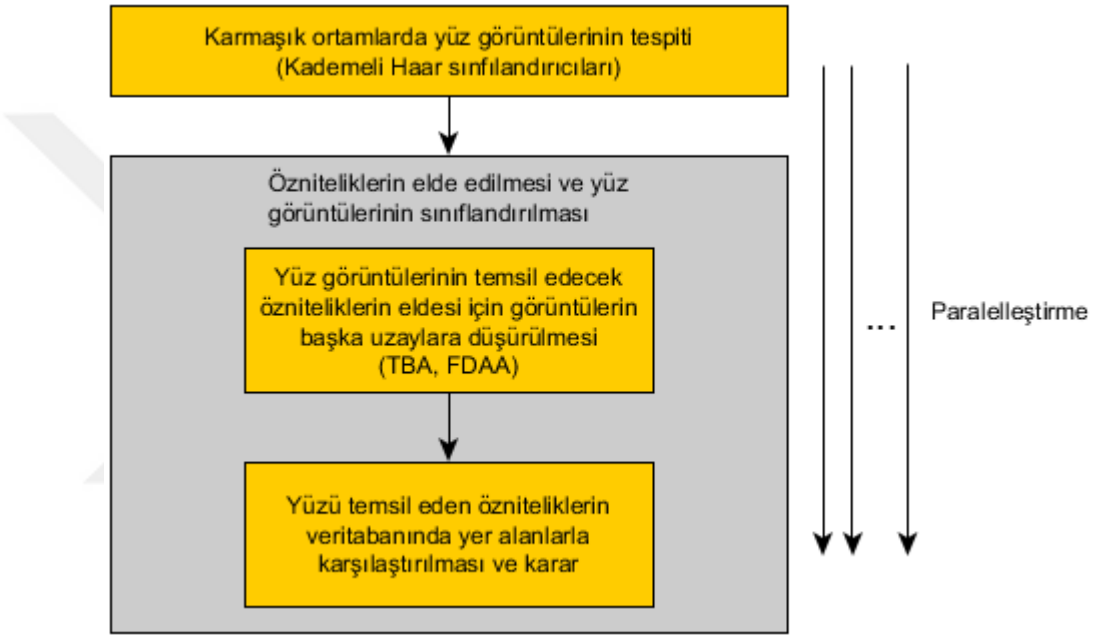
$$d_i \leq \mu_i + \alpha\sigma_i \tag{3.29}$$

α çarpanı z skoru olarak bilinir ve burada yeni görüntünün veritabanındaki görüntülere uzaklıklarının ortalamasının, veritabanındaki görüntülerin birbirlerine uzaklıklarının ortalamasından en çok ne kadar sapabileceğini ayarlar. Dolayısıyla bu çarpan eşleştirme duyarlılığını belirler. Yazılan uygulamada α çarpanına çok düşük,

düşük, orta, yüksek ve çok yüksek duyarlılıklı eşleştirme yapılması için sırasıyla 1, 0.5, 0, -0.5 ve -1 değerleri verilebilmektedir.

3.5. Yöntemlerin Microsoft .NET Ortamında Gerçeklenmesi

Önceki bölümlerde açıklanan yüz bulma, eşleştirme yöntemleri ve önerilen paralelleştirme yöntemi Microsoft Visual Studio kullanılarak Visual Basic.NET ve C# dilleri ile gerçeklenmiştir. Programın temel akışı şeması Şekil 3.13.'te verilmiştir.



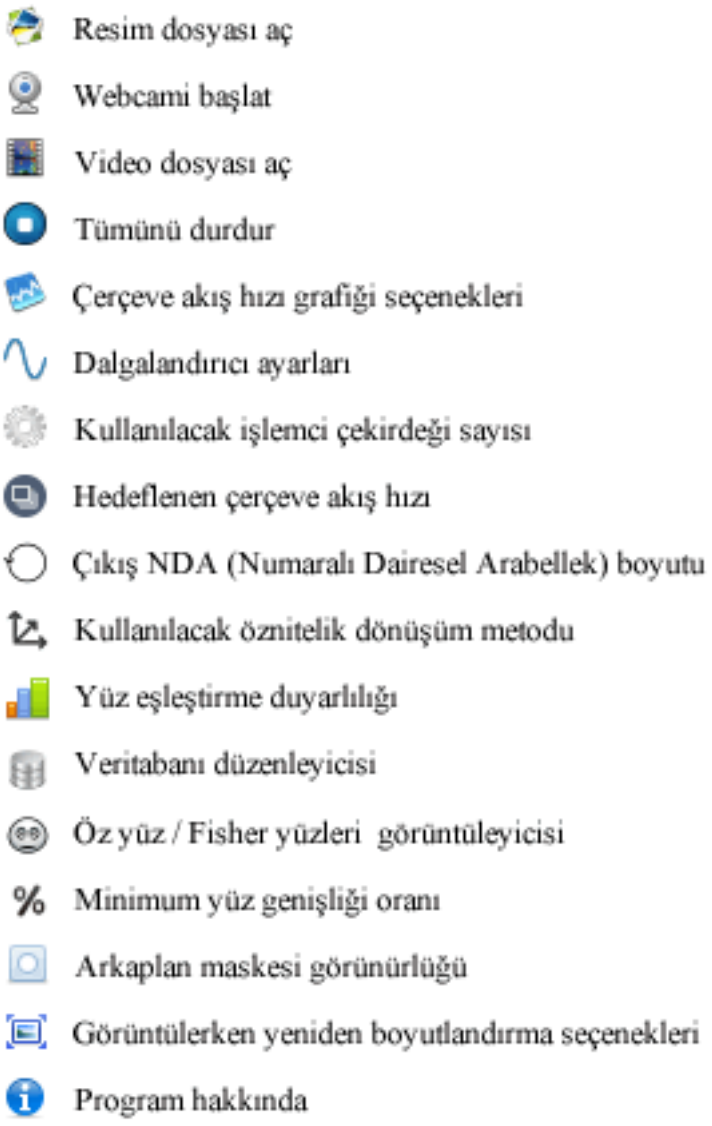
Şekil 3.13. Yazılan programın temel akış şeması

Yüz tanıma programı, Visual Basic.NET dili ile yazılmıştır. Kademeli Haar sınıflandırıcıları için açık kaynak kodlu OpenCV ve OpenCVSharp kütüphaneleri kullanılmıştır. TBA ve FDAA yöntemleri gerçeklenirken gerekli olan matris işlemleri, aynı kütüphane kullanılarak yapılmıştır. Yazılan programda, Haar sınıflandırıcıları ile yüzlerin konumlarının tespitinde kullanılan dikdörtgensel özniteliklerin boyutları 24x24'tür ve tespit edilen yüz görüntüleri TBA ve FDAA yöntemleri ile yeni uzaylara düşürülmeden önce 100x100 boyutlarına ölçeklenmektedir.

Önerilen paralelleştirme yöntemi başka uygulamalarda da kolayca kullanılabilecek şekilde bir .NET kütüphanesi olarak C# dili ile yazılmıştır. Uygulama yazıldıktan sonra, kütüphane kullanılarak paralelleştirme yapılmıştır.

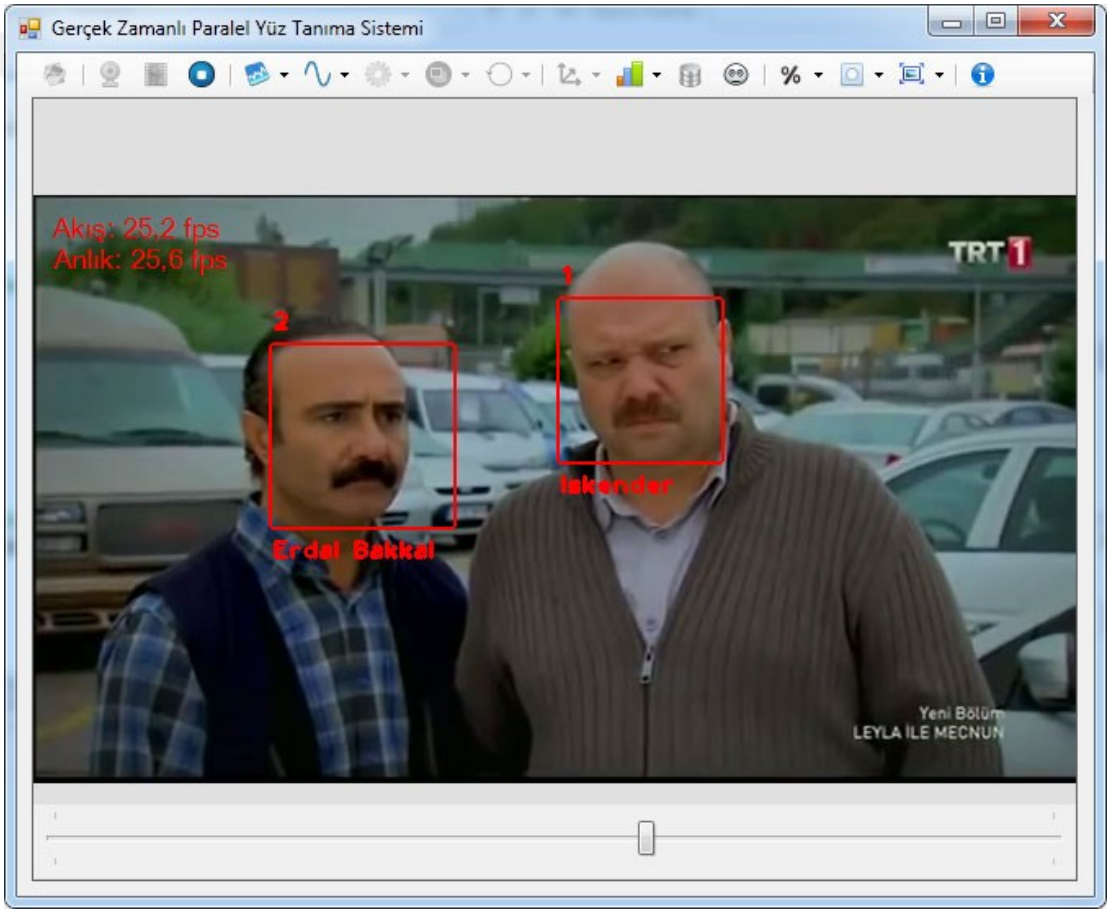
Görüntü kaynağı olarak bir video dosyası veya bir web kamerası kullanabilen programın arayüzü ile sunulan menüler (Şekil 3.14.) yardımıyla, kullanılmak istenen yüz tanıma yönteminin ve paralelleştirme için kullanılacak dalgalandırıcının seçilebilmesi, yöntemlerin işleyişini etkileyen değişkenlerin çoğuna dışarıdan müdahale edilebilmesi sağlanmıştır. Ayrıca paralelleştirme esnasında anlık çerçeve akış hızlarının ve çerçeve akış hızı grafiğinin çizilebilmesi için de bir seçenek yerleştirilmiştir.

Programın bir örnek veritabanı ile hazırladığı öz yüzlerin ve Fisher yüzlerinin gösterildiği arayüz Şekil 3.15.'te, program çalışıyorken alınan ekran görüntüsü Şekil 3.16.'da ve anlık akış hızı grafiği çiziliyorken alınan ekran görüntüsü Şekil 3.17.'de verilmektedir.

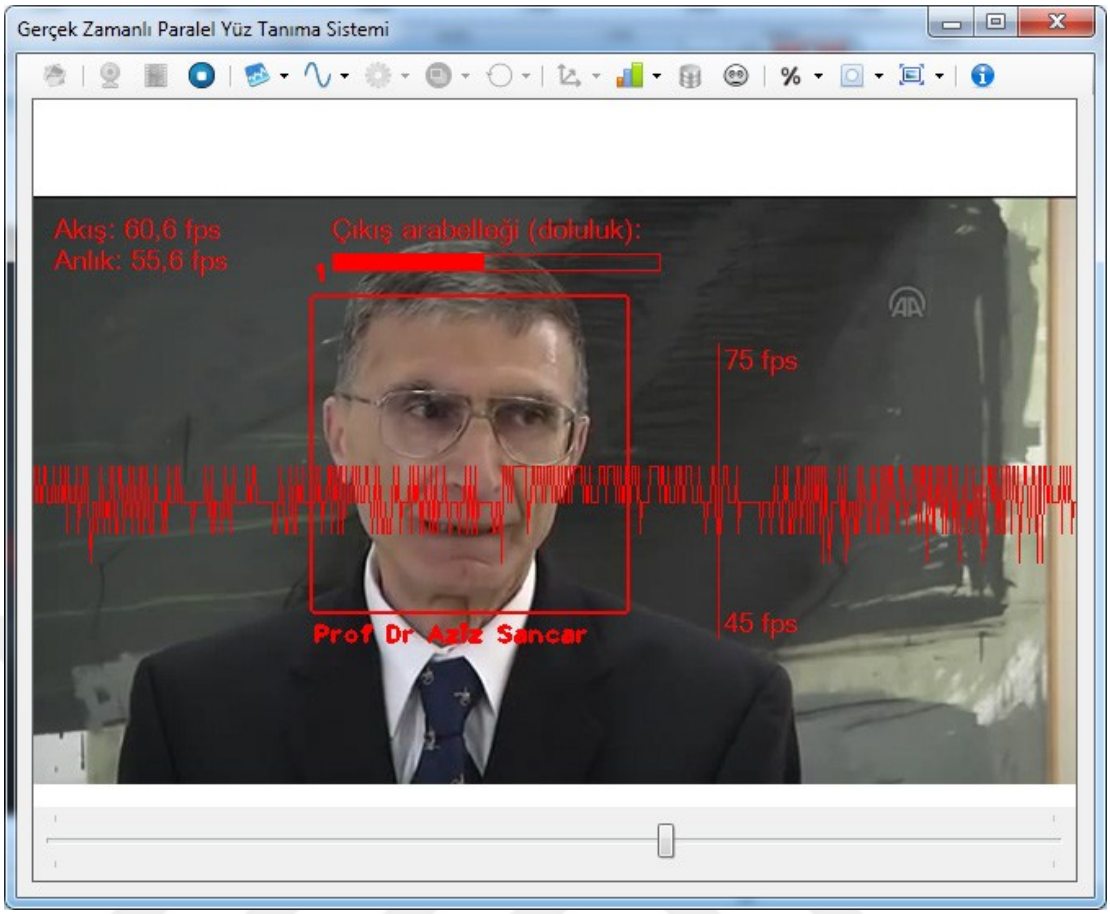


Şekil 3.14. Yazılan arayüz ile sunulan menüler ve işlevleri

Şekil 3.15. Yazılan programın bir örnek veritabanındaki verileri kullanarak hesapladığı öz yüzlerin ve Fisher yüzlerinin gösterildiği arayüz



Şekil 3.16. Yazılan program 25 fps akış hızı ile yürütülmesi gereken bir videoyu işlerken ve doğrusal dalgalandırıcı seçili iken alınmış ekran görüntüsü



Şekil 3.17. Yazılan program bir videoyu 60 fps hızı ile işlerken ve doğrusal dalgalandırıcı seçeneği ile anlık akış grafiği çizim seçeneği seçili iken alınan ekran görüntüsü

4. ARAŞTIRMA BULGULARI VE TARTIŞMA

Önerilen paralelleştirme yaklaşımı, yüz tanıma problemine uygulanmadan önce çeşitli senaryolarla test edilmiştir. Testler, 3,40 GHz hızında çalışan 4 fiziksel (8 mantıksal) çekirdek içeren bir Intel i7-3770 işlemcisi üzerinde yapılmıştır. Testlerde, 25 fps hızında gösterilmesi gereken 1280x720 piksel çözünürlüğünde 750 çerçeve içeren bir video, gerçek zamanlı bir görüntü kaynağı gibi kullanılmıştır. Test için yazılan programa, işlenecek görüntüler üzerinde uygulaması için bir görüntü işleme yöntemini temsilen aşağıdaki adımlardan oluşan bir görev verilmiştir.

- Görüntüyü gri seviyeye indirge ve gri görüntünün iki kopyasını oluştur
- İlk kopya üzerinde histogram dengeleme işlemini uygula
- İlk kopya ile 2x2'lik ortalama filtresini konvolüsyona tabi tut ve bu adımı *M* defa tekrarla
- İkinci kopyayı ilk kopyanın üzerine kenarlardan 20 piksel içeriye çizdir

Bu adımlar, açık kaynak kodlu görüntü işleme kütüphanesi OpenCV kullanılarak yürütülmüştür. 3. adımdaki *M* değeri, iş yükü birimi olarak kabul edilmiştir. Testlerdeki *M* değerleri, işlemci çekirdeklerinin ortalama kullanılma oranlarının %100'den düşük ve %100'e en yakın değerlerde olmasını sağlayacak şekilde seçilmiştir. Örneğin görev tek bir çekirdek üzerinde yürütülürken *M*=8 alındığında ortalama çekirdek kullanımı %85,45 ve *M*=9 alındığında %100 olmaktadır. Bu nedenle bu senaryo için en uygun *M* değeri 8'dir.

Tüm denemelerde giriş ve çıkış NDA'larının kapasitesi sırasıyla 5 ve 25 olarak ayarlanmıştır. Böylece çıkış NDA'sında işaretçilerin işaret ettiği numaralar arasındaki en uygun mesafe 12,5 birim ve giriş ile çıkış arasındaki gecikme 0,5 saniye olmaktadır.

Yöntem, önce doğrusal dalgalandırıcı kullanılarak 4 farklı senaryo ile test edilmiştir. Yapılan ilk denemede (D1) görevler tek bir işlemci üzerinde yürütülmüş ve paralelleştirme kullanılmamıştır. İkinci denemede (D2) 4 mantıksal (2 fiziksel) çekirdek kullanılmıştır. Üçüncü denemede (D3) yine 4 çekirdek kullanılmış, ancak bu sefer iş parçacıkları 4 ayrı fiziksel çekirdek üzerinde yürütülmüştür. Son denemede (D4) 4 fiziksel çekirdek, toplam 8 mantıksal çekirdek ile birlikte kullanılmıştır. İlk 4 senaryo için elde edilen sonuçlar Çizelge 4.1.'de verilmiştir.

Çizelge 4.1. İlk 4 denemede iş yükleri ve çekirdek kullanımları. Ç1, Ç2, …, Ç8 mantıksal çekirdekleri ve her bir ikili (Ç1-Ç2, Ç3-Ç4, Ç5-Ç6, Ç7-Ç8), fiziksel çekirdekleri temsil etmektedir. (M: Mantıksal, F: Fiziksel)

No	Kullanılan Çekirdek Sayısı	İş Yükü	Çekirdek Kullanımı (Ortalama %)							
			Ç1	Ç2	Ç3	Ç4	Ç5	Ç6	Ç7	Ç8
D1	1M (1F)	8	85,45	-	-	-	-	-	-	-
D2	4M (2F)	20	20,18	88,09	88,64	87,82	-	-	-	-
D3	4M (4F)	30	17,61	5,46	95,88	0,36	94,76	0,28	95,01	0,21
D4	8M (4F)	42	21,53	86,63	90,30	92,52	90,33	91,46	89,37	90,32

Yapılan ilk denemede (D1), tüm okuma, işleme, yazma işlemleri ve arayüz ile ilgili işlemler tek bir çekirdek üzerinde yürütülmüştür. Tek bir çekirdek M=8 birime kadar iş yükünü problemsiz bir şekilde yürütebildiğinden diğer testlerde elde edilen performans artışları değerlendirilirken bu değer baz alınmıştır.

İkinci denemede (D2) kullanılan çekirdek sayısı 4 katına çıkarılmıştır. Ancak sorunsuz olarak yürütülebilen iş yükü M=20 olarak ölçülmüştür ve bu değer, tek çekirdek kullanılırken yürütülebilen iş yükünün 4 katından daha düşüktür. Bunun nedeni, önerilen yöntemin ilk çekirdeği kontrol çekirdeği olarak kullanması ve dolayısıyla bu çekirdek üzerinde direkt olarak görüntülerin işlenmesi ile ilgili herhangi bir işlem yürütülmemesidir. Buna rağmen yürütülebilen iş yükü değeri, ilk denemedekinin 3 katının da altında kalmıştır. Bu denemede 4 mantıksal çekirdek kullanılmışsa da işlemler aslında 2 fiziksel çekirdek üzerinde yürütülmüştür. Her bir mantıksal çekirdek çifti (Ç1-Ç2, Ç3-Ç4), mümkün olduğunda aynı anda 2 iş parçacığı yürütebilen bir fiziksel çekirdeği temsil ettiğinden, aynı anda iki mantıksal çekirdek kullanıldığında elde edilen işlem gücü, bir tek mantıksal çekirdek kullanıldığında elde edilen işlem gücünün iki katından daha düşük olmaktadır.

4 mantıksal çekirdeğin 4 fiziksel çekirdek üzerinde kullanıldığı üçüncü denemede (D3) program, M=30 birime kadar iş yükünü yürütebilmiştir. 4 fiziksel çekirdekten üçünün işçi çekirdek olarak kullanılmasına rağmen yürütülebilen iş yükü, D1'e göre 3,75 kat daha fazladır. Tek bir çekirdek kullanıldığında çekirdek yalnızca görüntülerin işlenmesinden değil, okunması ve görüntülenmesi gibi diğer işlemlerden de sorumludur. Ancak önerilen yaklaşım ile görüntülerin işlenmesi dışındaki işlemler kontrol çekirdeğinde yürütüldüğünden, işçi çekirdekler yalnızca görüntüleri işlemekle

yükümlüdür. Bu sayede 3 işçi çekirdek kullanıldığında yürütülebilen iş yükü, tek çekirdek kullanıldığından yürütülebilen iş yükünün 3 katından daha fazla olmaktadır.

Dördüncü denemede (D4) tüm mantıksal çekirdekler kullanılarak M=42 birimlik iş yükü problemsiz olarak yürütülebilmiştir. Bu değer, yürütülebilen en yüksek iş yükü değeridir ve yukarıda anlatılan nedenlerden dolayı tek çekirdek kullanılarak elde edilen değerin yaklaşık olarak 8 katı değildir. 5,25 katlık performans artışı, işlemcinin 8 değil, 4 fiziksel çekirdeğe sahip olduğu dikkate alınarak değerlendirildiğinde performans artışı oldukça yüksektir.

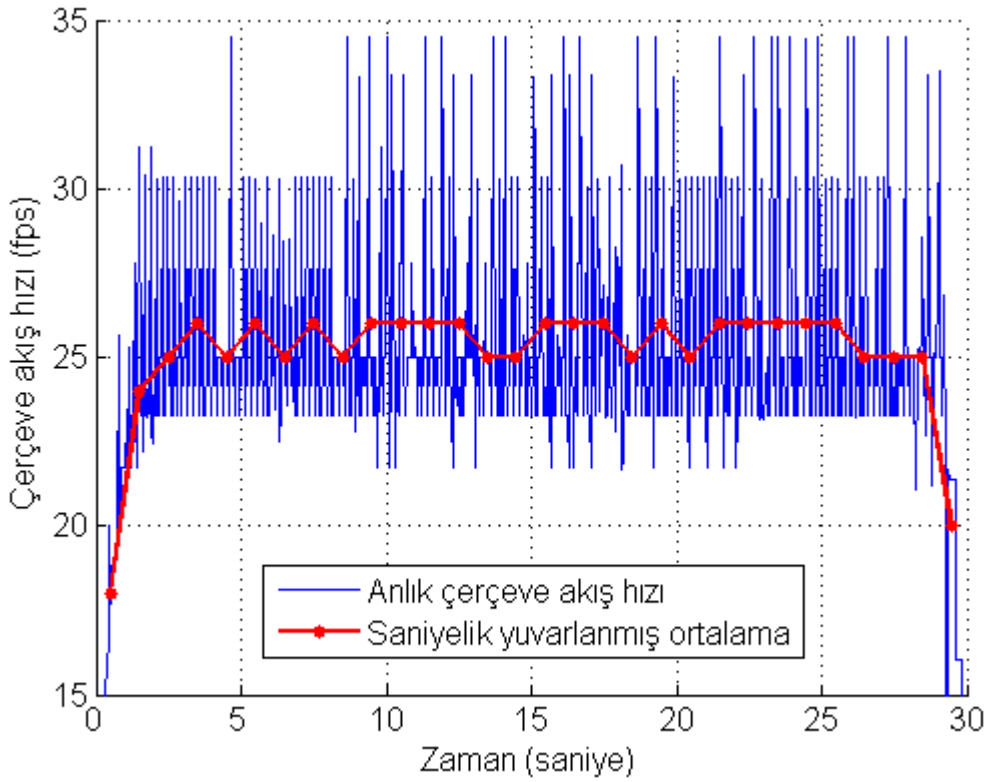
Kullanılan dalgalandırıcı, çekirdek performansını etkilemeyecek kadar az bir hesaplama kapasitesi gerektirdiğinden doğrusal dalgalandırıcı yerine üstel dalgalandırıcı kullanılması, sağlanması beklenen performans artışını etkilemeyecektir. Dolayısıyla ilk 3 denemede yalnızca doğrusal dalgalandırıcı kullanılmıştır. Üstel dalgalandırıcının amacı çerçeve akış hızının kararlı tutulması olduğundan 4. deneme, doğrusal ve üstel dalgalandırıcı kullanılarak farklı A değerleri ile tekrar edilmiş ve çerçeve akış hızları incelenmiştir. Sonuçlar Çizelge 4.2.'de verilmiştir.

Çizelge 4.2. Denemeler esnasında elde edilen anlık çerçeve akış hızlarının en yüksek, en düşük değerleri, standart sapmaları ve ortalamaları

Dalgalandırıcı	**Anlık Çerçeve Akış Hızı (fps)**			
	En Yüksek	**En Düşük**	**Ortalama**	**Standart Sapma**
Doğrusal	34,52	21,68	25,57	2,99
Üstel (A=10)	28,58	22,46	25,28	0,86
Üstel (A=100)	26,33	22,45	25,13	0,48
Üstel (A=1000)	26,32	22,62	25,08	0,44

Okuma ve yazma işaretçilerinin işaret ettiği alanların numaraları arasındaki fark başlangıçta sıfırdır ve farkın en güvenli değerine ulaşması daha önce açıklandığı gibi süre almaktadır. Ayrıca kaynaktan okunan görüntüler bittiğinde ve çıkış NDA'sı da boşalmaya yaklaştığında algoritma, periyodu genişleterek numaralar arasındaki farkı tekrar en güvenli değerine yaklaştırmaya çalışacaktır. Bu nedenlerle Çizelge 4.2.'deki değerler hesaplanırken ilk 4 ve son 4 saniye hesaba katılmamıştır.

Doğrusal dalgalandırıcı kullanılırken elde edilen anlık çerçeve akış hızlarının ve saniyelik ortalamaların grafiği Şekil 4.1.'de verilmiştir.

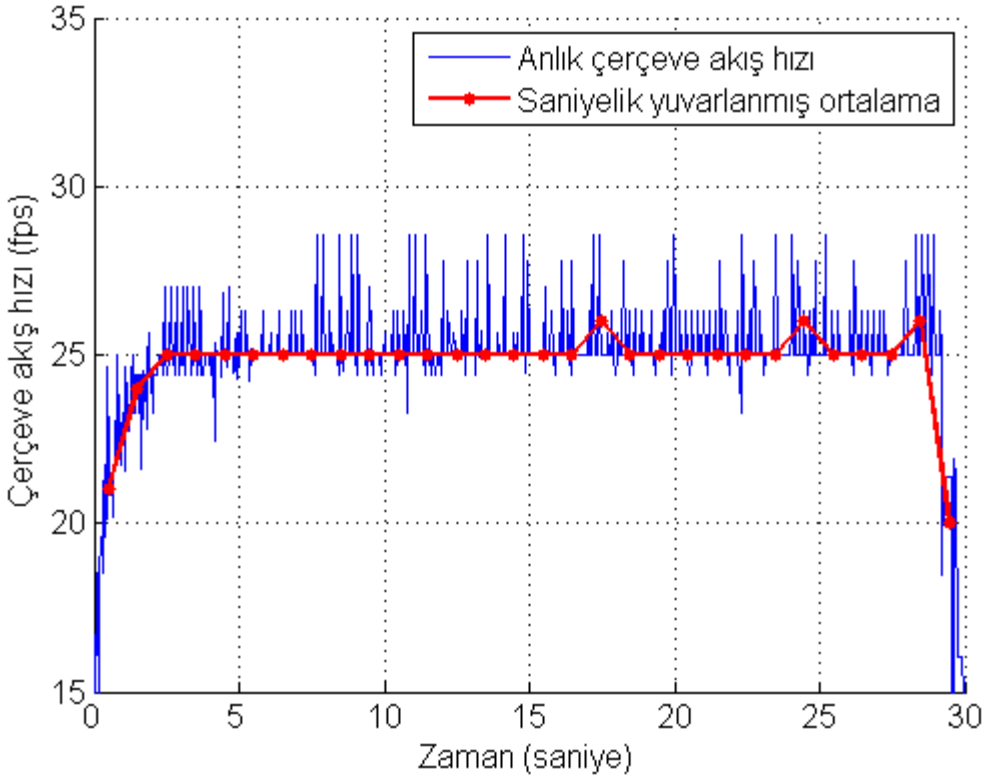


Şekil 4.1. Doğrusal dalgalandırıcı kullanıldığında elde edilen çerçeve akış hızları

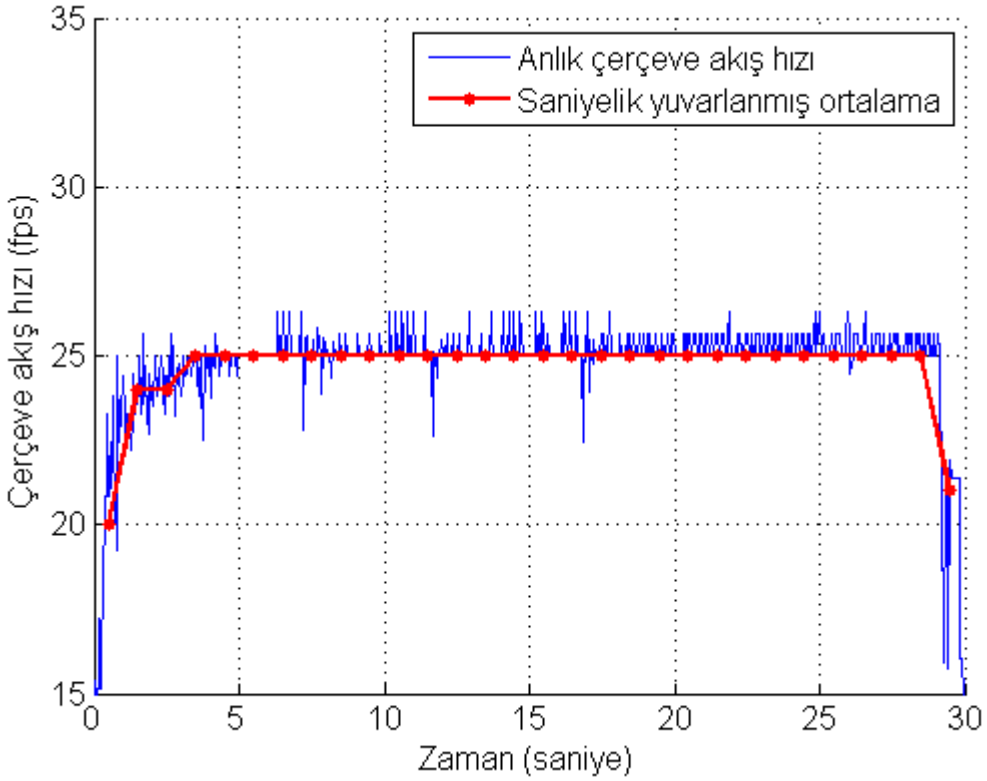
Çizelge 4.2. ve Şekil 4.1.'e bakıldığında, doğrusal dalgalandırıcı kullanılırken anlık çerçeve akış hızları kararsız görünse de saniyelik ortalamalar çok fazla değişmemiştir. Çerçeve akış hızındaki anlık değişimler işlenen görüntüyü izleyen bir izleyici için gözle kolayca algılanabilecek düzeyde rahatsız edici değildir. Ancak doğrusal dalgalandırıcı kullanıldığında elde edilen anlık çerçeve akış hızlarının standart sapma değeri yüksektir (2,99 fps).

Doğrusal dalgalandırıcı yerine üstel dalgalandırıcı A=10 değeri ile kullanıldığında elde edilen anlık çerçeve akış hızlarının daha kararlı olduğu Şekil 4.2.'de görülebilmektedir. A=10 alındığından anlık akış hızlarının standart sapması 1 fps altındadır.

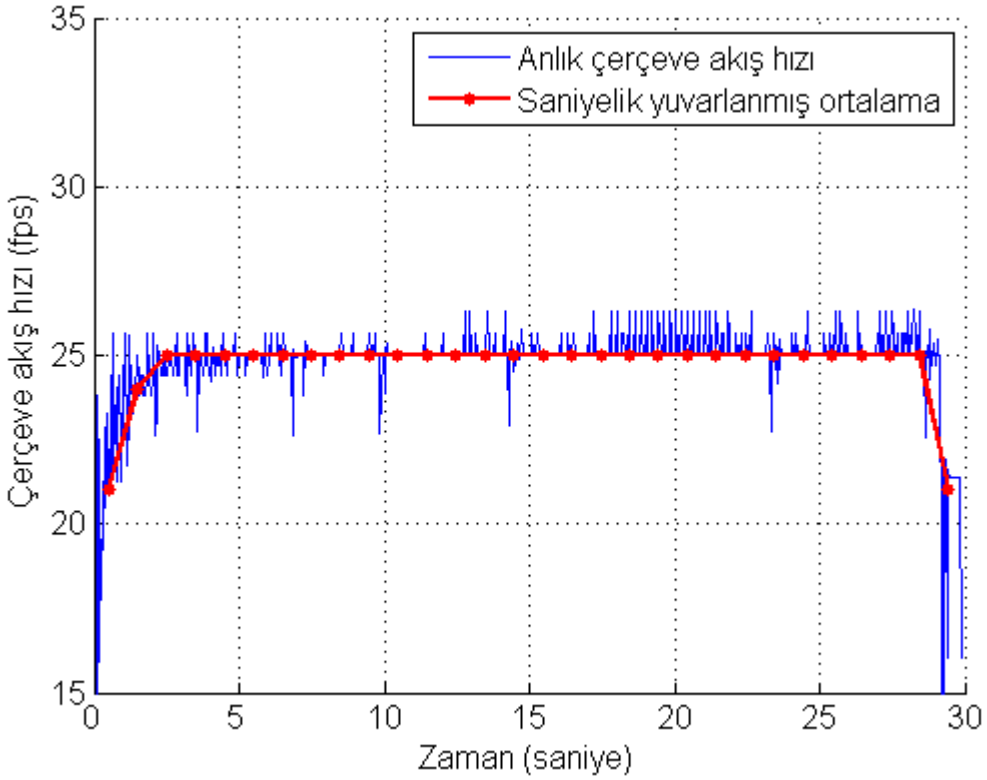
Üstel dalgalandırıcı A=100 ve A=1000 alınarak kullanıldığında ise çerçeve akış hızlarının daha da kararlı hale geldiği Şekil 4.3. ve Şekil 4.4.'te görülebilmektedir. Anlık çerçeve akış hızlarının standart sapması her iki durumda da 0,5 fps değerinin altındadır. En son ve çerçeve akış hızının sabit tutulmaya çalışılması bakımından en başarılı deneme yapılırken ölçülen çekirdek kullanımları Şekil 4.5.'te verilmiştir.



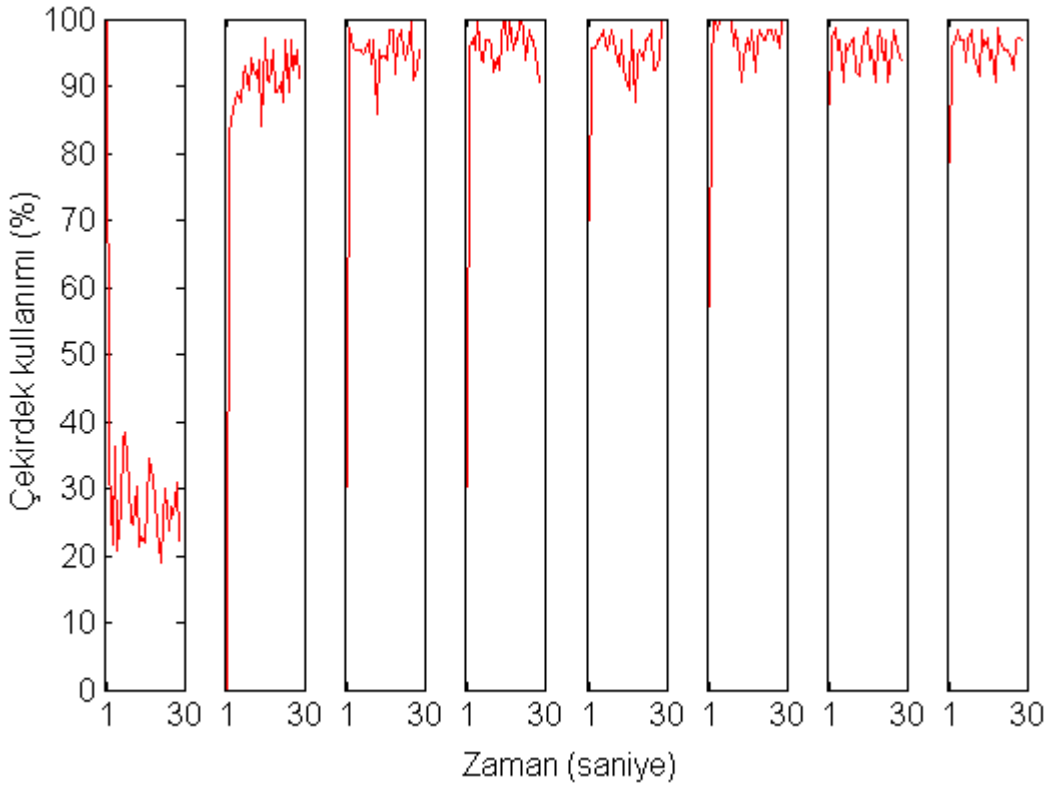
Şekil 4.2. Üstel dalgalandırıcı A=10 değeri ile kullanıldığında elde edilen çerçeve akış hızları



Şekil 4.3. Üstel dalgalandırıcı A=100 değeri ile kullanıldığında elde edilen çerçeve akış hızları

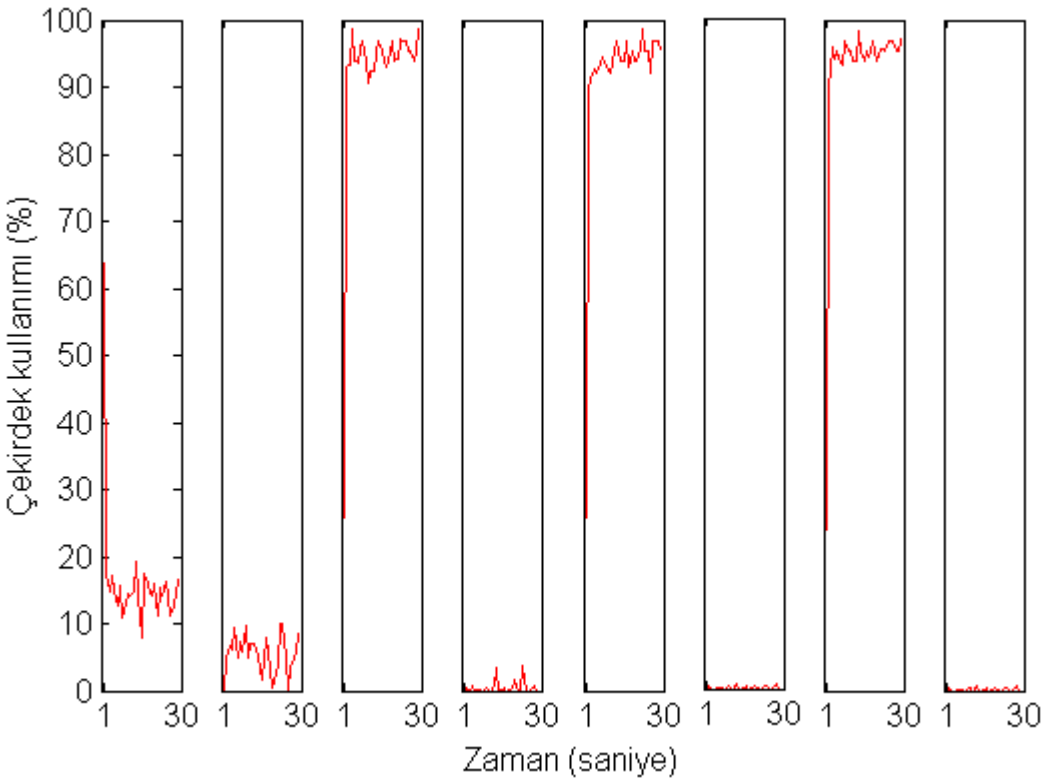


Şekil 4.4. Üstel dalgalandırıcı A=1000 değeri ile kullanıldığında elde edilen çerçeve akış hızları



Şekil 4.5. Üstel dalgalandırıcı A=1000 değeri ile kullanıldığında ve 8 mantıksal çekirdek kullanılarak M=42 birimlik iş yükü yürütüldüğünde ölçülen çekirdek kullanımları

Kontrol çekirdeği olarak kullanılan ilk çekirdek, kaynaktan görüntü okumak, görüntüleri ekrana yansıtmak, anlık akış hızları ve çekirdek kullanımları ile ilgili veri toplamak ve kullanıcı arayüzünün kesintisiz olarak kullanıcıya yanıt vermesini sağlamak ile meşguldür. Şekil 4.5.'te kontrol çekirdeğinin yoğun olarak kullanılmadığı görülebilmektedir. Bu sayede kullanıcı herhangi bir kesinti veya takılma olmadan uygulamanın arayüzü ile etkileşimini sürdürebilmektedir. İşçi çekirdeklerse işlem kapasitelerini giriş NDA'sından aldıkları görüntüleri işlemek için kullanmaktadır.



Şekil 4.6. Doğrusal dalgalandırıcı kullanıldığında ve 4 fiziksel çekirdek üzerinde 4 mantıksal çekirdek kullanılarak *M*=30 birimlik iş yükü yürütüldüğünde ölçülen çekirdek kullanımları

Şekil 4.6.'da, iş yükü 4 fiziksel çekirdek kullanılarak paralelleştirildiğinde elde edilen çekirdek kullanımları gösterilmektedir. Şekilde iş yükünün başarılı bir biçimde işçi çekirdeklere dağıtılabildiği görülmektedir. Kontrol çekirdeği olarak seçilen ilk mantıksal çekirdeğin işlemlerinin mümkün olduğunda işletim sistemi tarafından ikinci mantıksal çekirdek üzerinde yürütülebildiği de gözlemlenebilmektedir.

Yapay olan görüntü işleme problemi üzerinde yapılan testler tamamlandıktan sonra, yöntem gerçeklenerek yazılan .NET kütüphanesi, yazılan yüz tanıma programında kullanılarak program üzerinde de denemeler yapılmıştır. Önceki

denemelerin tamamı sabit iş yükleri ile yapılmıştır. Ancak yüz tanıma aşamasından önceki yüz bulma işlemlerinin süresi, görüntülerdeki alt bölgeler kademeli Haar sınıflandırıcıları ile sınanırken birinden geçemeyen bölgenin diğerleri ile sınanması gerekmediği için değişkendir. Ayrıca görüntüde yüz bulunamaması halinde, yüz tanıma ile ilgili hiçbir işlem yapılmayacağı için yüz içeren ve içermeyen görüntülerin de işlem yükleri farklı olacaktır. Bu nedenlerle yüz tanıma programı üzerinde yapılan testler yöntemin değişken iş yükleri ile kullanıldığında nasıl sonuçlar vereceğinin incelenmesi açısından önemlidir.

Program üzerinde yapılan denemelerde 25 fps ve 60 fps hızlarında yürütülmesi gereken 640x360 piksel çözünürlüğünde iki video dosyası kullanılmıştır. Program bu video dosyalarını işlerken her bir denemede ilk 20 saniye boyunca anlık akış hızları ve işlemci çekirdeklerinin kullanımları kayıt altına alınmıştır. İlk dört denemede 25 fps hızında yürütülmesi gereken video işlenirken doğrusal ve üstel dalgalandırıcılar kullanılmıştır. 60 fps hızında yürütülmesi gereken diğer video üzerinde yapılan denemelerde ise üstel dalgalandırıcı ile yapılan önceki üç denemede hedeflenen akış hızına en yakın ortalamayı veren A değeri kullanılmıştır. Sonuçlar Çizelge 4.3.'te verilmiştir.

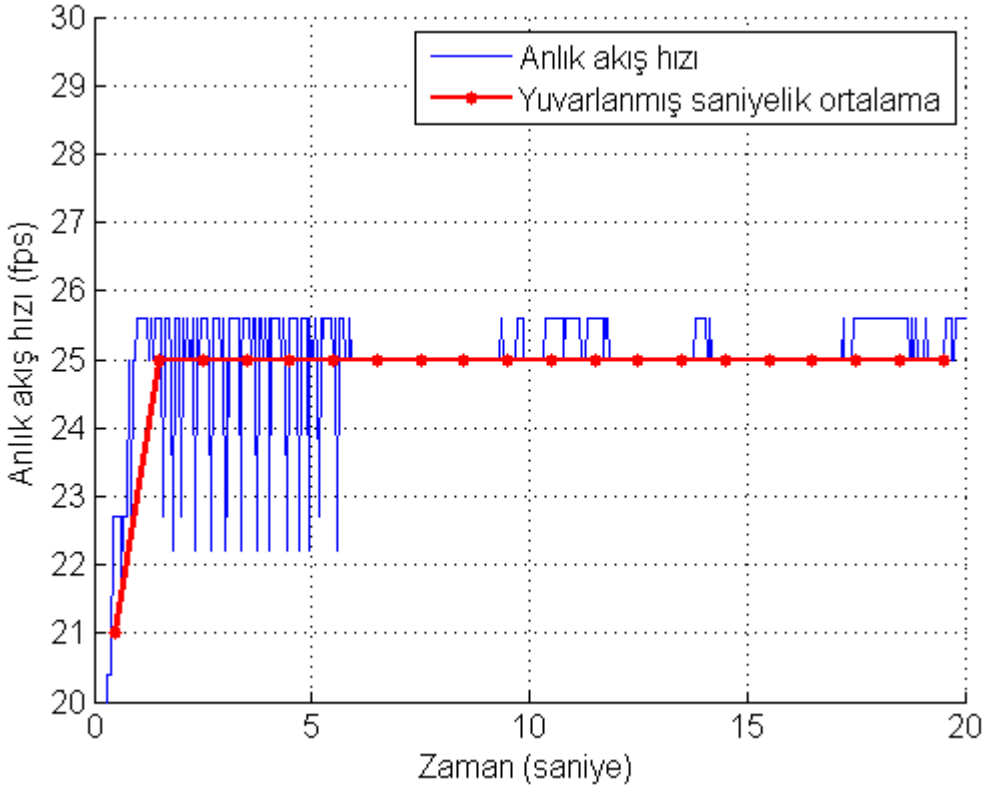
Çizelge 4.3. Yüz tanıma programı ile yapılan denemeler ve bu denemeler sonucunda elde edilen anlık çerçeve akış hızları (M: Mantıksal, F: Fiziksel)

Hedef Akış Hızı (fps)	**Dalgalandırıcı**	**Kullanılan Çekirdek Sayısı**	**Anlık Çerçeve Akış Hızı (fps)**			
			En Yüksek	**En Düşük**	**Ortalama**	**Standart Sapma**
25	Doğrusal	8M (4F)	25,60	22,20	25,14	0,45
25	Üstel (A=10)	8M (4F)	25,00	24,40	24,80	0,28
25	Üstel (A=100)	8M (4F)	25,60	24,40	24,95	0,36
25	Üstel (A=1000)	8M (4F)	25,00	23,80	24,63	0,30
60	Üstel (A=100)	8M (4F)	66,70	52,60	59,56	2,40
60	Üstel (A=100)	4M (2F)	66,70	47,60	58,37	3,55

İlk 5 denemede yüz bulma ve tanıma işlemlerinin yarattığı iş yükü, kullanılan çekirdeklerin rahatlıkla yürütebileceği seviyelerdedir. Çizelge 4.3.'ün son satırında sonuçları verilen denemede diğerlerinden farklı olarak çekirdeklerin yetiştiremeyeceği bir iş yükü yaratmak amacıyla 2 fiziksel çekirdek kullanılmış ve görüntülerde aranacak minimum yüz oranı %10'a ayarlanmıştır. Diğer denemelerde bu oran %20'dir ve bu değer düşürüldükçe içerisinde yüz görüntüsü aranacak olan alt bölge sayısı katlanarak

artmaktadır. Dolayısıyla %10 seçeneği seçili iken yapılması gereken işlem sayısı, %20 seçili iken yapılması gerekenin iki katından çok daha fazladır. Böylece son denemede kullanılan işlemci çekirdekleri istenen akış hızını yakalamaya yetmediğinden, yöntemin böyle durumlarda arabellekteki veri miktarındaki anlık yükselişler veya düşüşlere nasıl tepkiler vereceği de incelenmiştir.

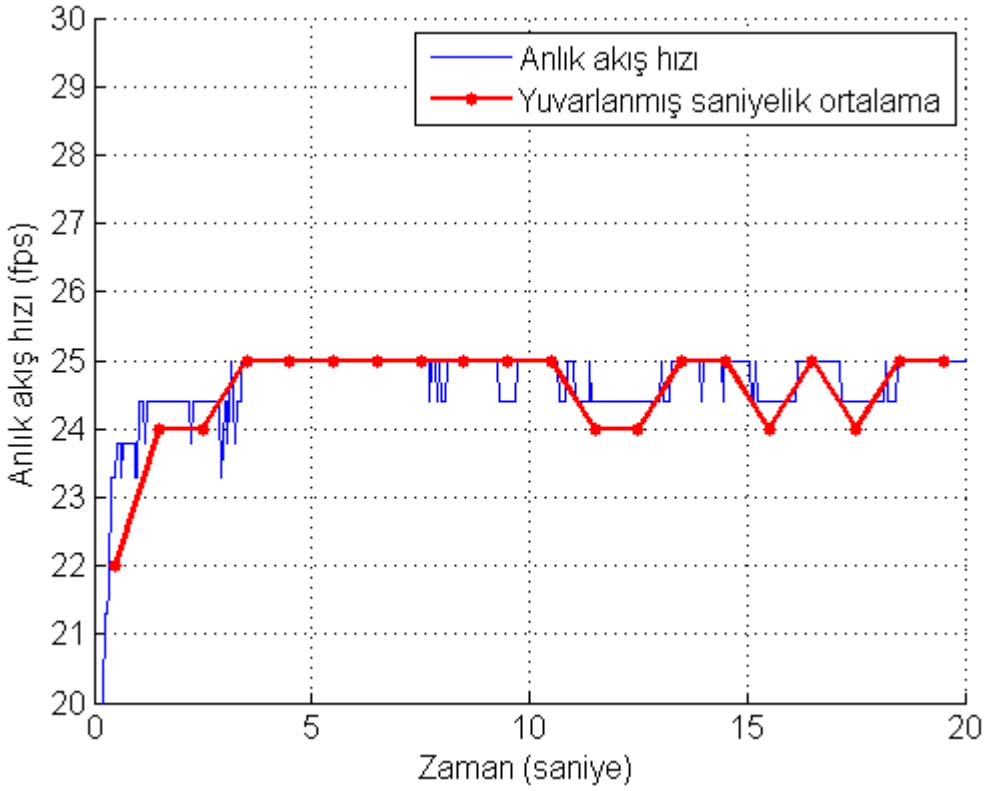
Çizelge 4.3.'teki denemelerde elde edilen anlık akış hızlarının grafikleri sırasıyla Şekil 4.7., Şekil 4.8., Şekil 4.9., Şekil 4.10., Şekil 4.11. ve Şekil 4.12.'de gösterilmektedir.



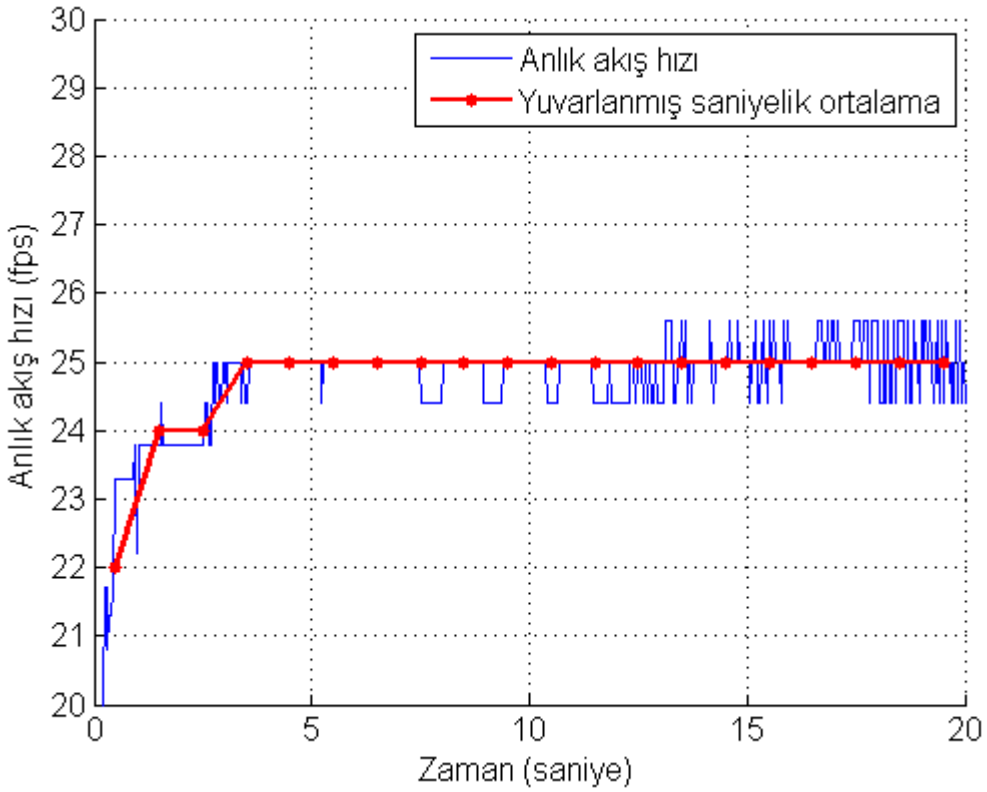
Şekil 4.7. Yüz tanıma programında 25 fps akış hızı hedeflenerek doğrusal dalgalandırıcı kullanıldığında elde edilen çerçeve akış hızları

Şekil 4.7.'de başlarda periyodun genişletilmesi için müdahaleler yapıldığı görülebilmektedir. Yaklaşık olarak 9. saniyeden sonra periyot daraltılarak NDA'nın dolmaya yaklaşması önlenmektedir.

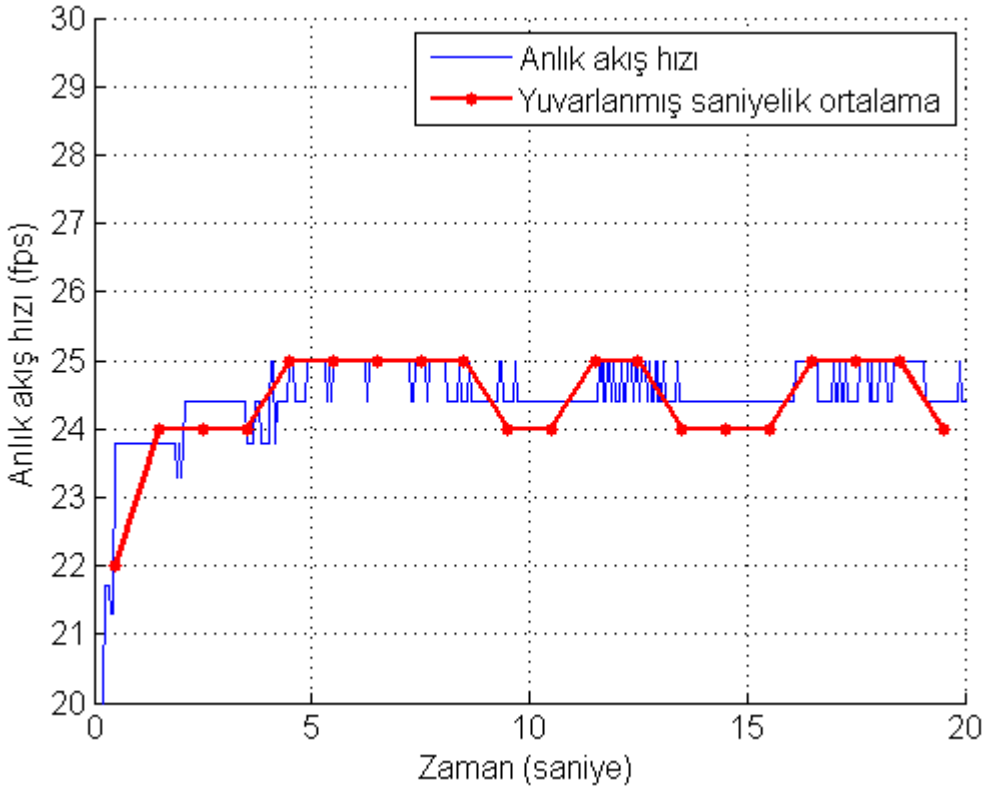
Şekil 4.8., Şekil 4.9. ve Şekil 4.10. incelendiğinde, üstel dalgalandırıcı için kullanılan taban değeri (A) büyütüldükçe, NDA'nın işaretçilerinin işaret ettikleri numaralar arasındaki mesafeyi güvenli olan değerine yaklaştırmak için periyoda yapılan müdahale zayıfladığından hedeflenen akış hızına daha geç yaklaşıldığı görülmektedir.



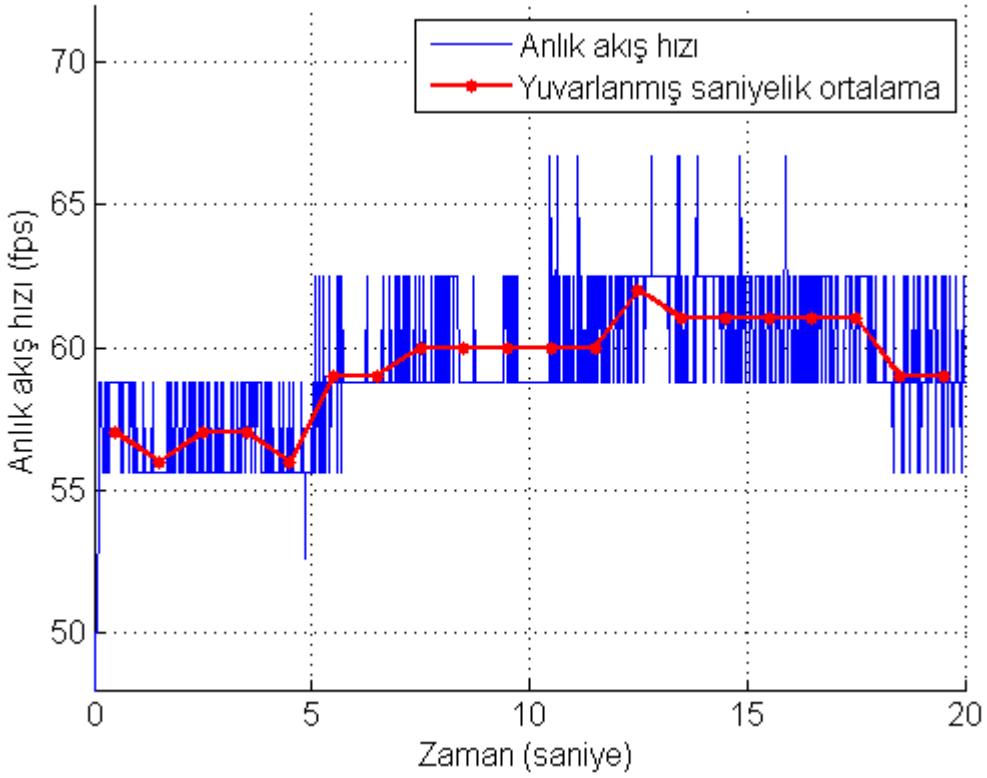
Şekil 4.8. Yüz tanıma programında 25 fps hızı hedeflenerek üstel dalgalandırıcı A=10 değeri ile kullanıldığında elde edilen çerçeve akış hızları



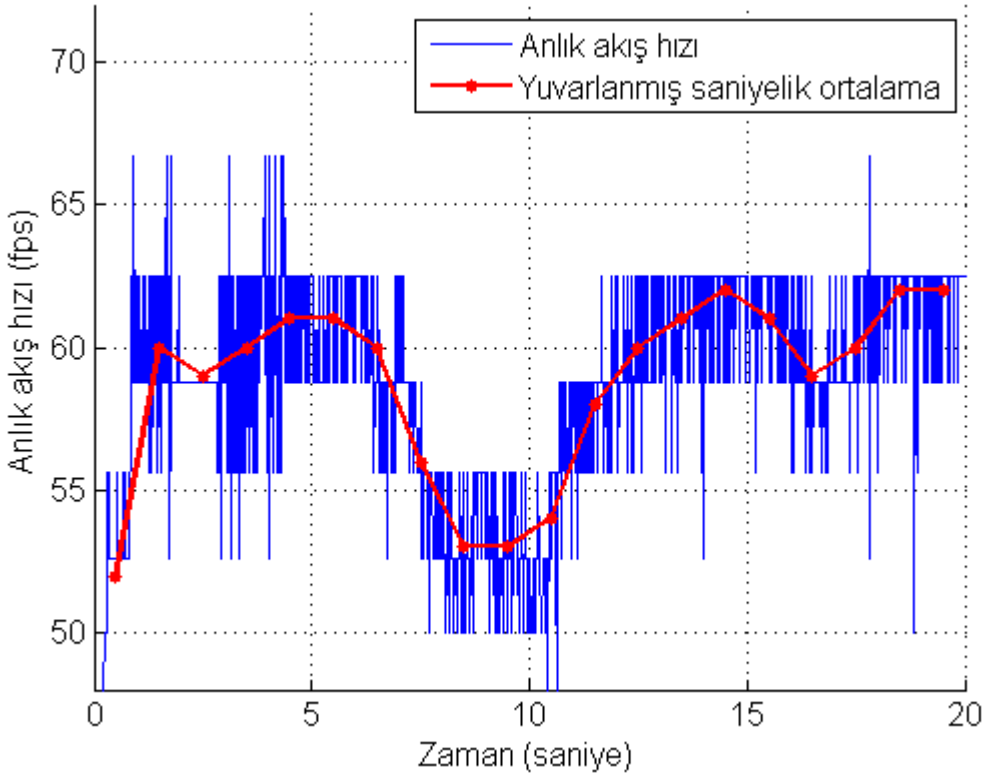
Şekil 4.9. Yüz tanıma programında 25 fps hızı hedeflenerek üstel dalgalandırıcı A=100 değeri ile kullanıldığında elde edilen çerçeve akış hızları



Şekil 4.10. Yüz tanıma programında 25 fps hızı hedeflenerek üstel dalgalandırıcı A=1000 değeri ile kullanıldığında elde edilen çerçeve akış hızları

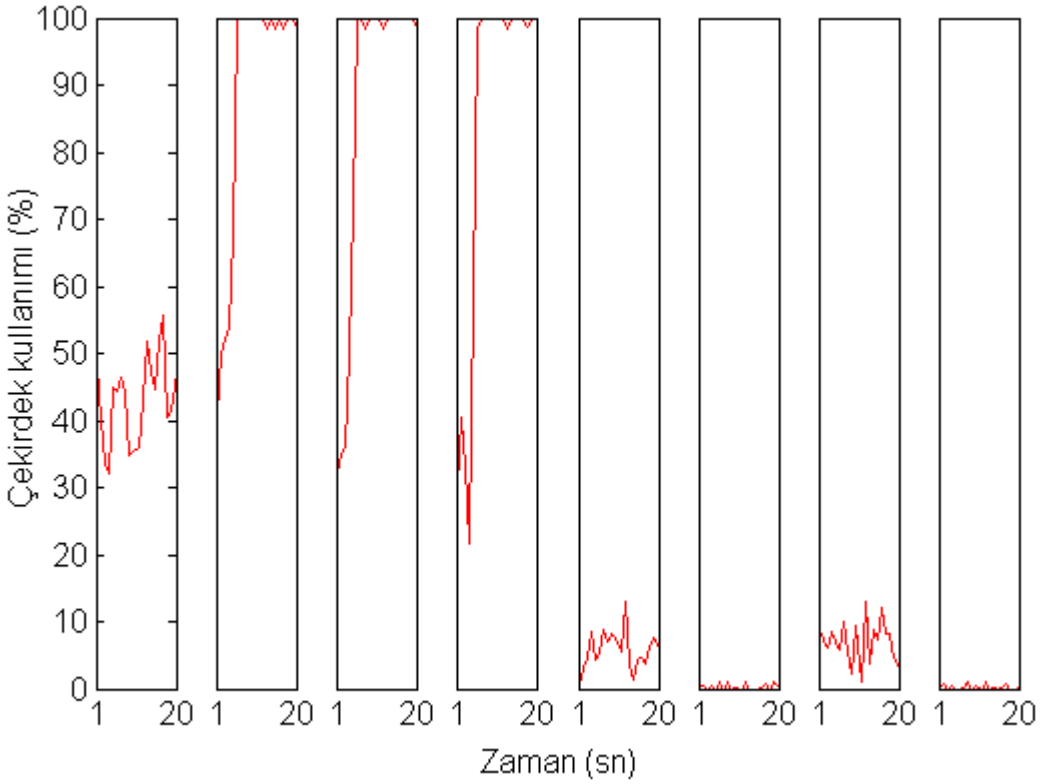


Şekil 4.11. Yüz tanıma programında 60 fps akış hızı hedeflenerek üstel dalgalandırıcı A=100 değeri ile kullanıldığında elde edilen çerçeve akış hızları

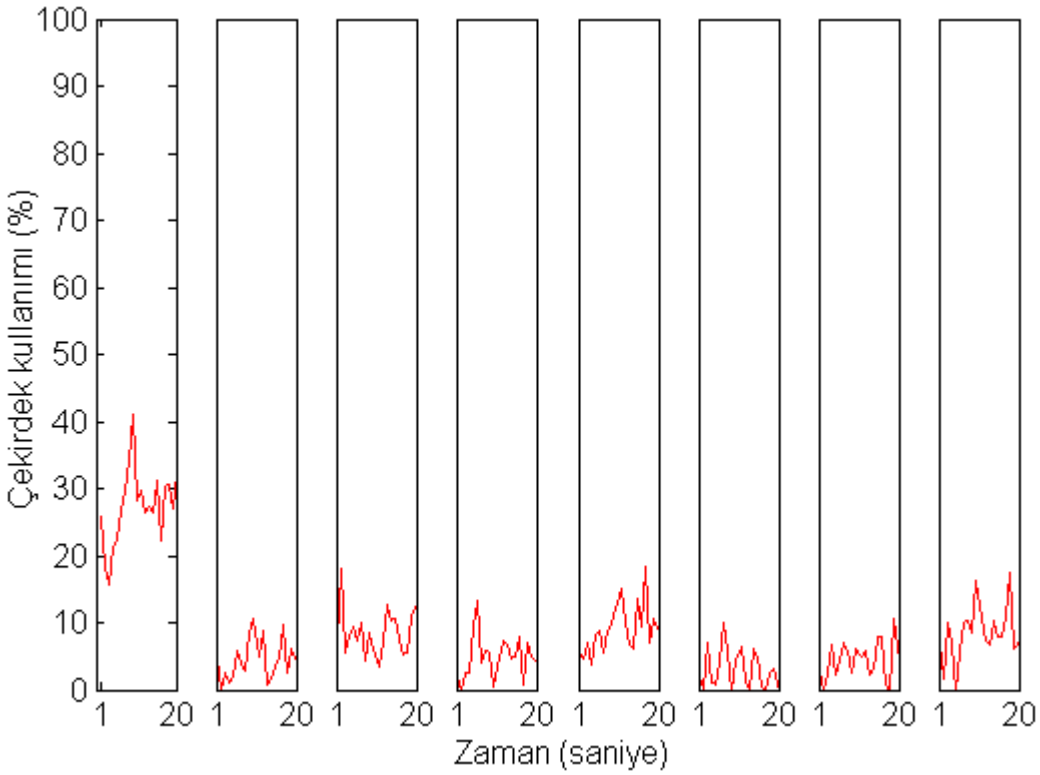


Şekil 4.12. Yüz tanıma programında 60 fps akış hızı hedeflenerek üstel dalgalandırıcı A=100 değeri ile kullanıldığında, 4 mantıksal çekirdek iş yükünü kaldırmakta yetersiz kalırken elde edilen çerçeve akış hızları

60 fps akış hızı hedeflenerek yapılan denemelerdeki sonuçlara ve akış hızı grafiklerine bakıldığında (Çizelge 4.3., Şekil 4.11., Şekil 4.12.), 60 fps hızındaki ilk denemede akış hızının standart sapması 2,40, ikinci denemede 3,55 olmuştur. Bu değerler 60 fps hedefi için yüksek değildir. 60 fps'nin hedeflendiği ikinci denemede elde edilen grafikte (Şekil 4.12.) çekirdeklerin 60 fps akış hızı elde edilebilmesi için yapması gereken işlemleri yetiştiremediği anlaşılmaktadır. İş yükü zaman ilerledikçe değiştiğinden, periyoda yapılan müdahaleler ortalamanın hedefe yaklaşmasını sağlayabilmiştir. Bu deneme boyunca ölçülen çekirdek kullanımları (Şekil 4.13) işçi çekirdeklerin sürekli olarak meşgul olduğunu ve işletim sisteminin program tarafından kullanıma ayrılmayan diğer fiziksel çekirdekleri de kullanabildiğini göstermektedir. İşlemci çekirdekleri işleri rahatlıkla yetiştirebiliyorken ölçülen çekirdek kullanımlarını örneklemek amacıyla 25 fps hedeflenerek yapılan denemelerden dördüncüsü (Çizelge 4.3, 4. satır) esnasında kaydedilen çekirdek kullanımlarının grafiğine de Şekil 4.14.'te yer verilmiştir. Bu şekilde de iş yükünün çekirdeklere başarıyla paylaştırabildiği görülmektedir.



Şekil 4.13. Yalnızca 4 mantıksal çekirdek kullanılıyorken ve hedeflenen 60 fps akış hızı sağlanamıyorken ölçülen çekirdek kullanımları



Şekil 4.14. Program ile yapılan 4. denemede, çekirdeklere az iş yükü düşerken ölçülen çekirdek kullanımları

5. SONUÇ VE ÖNERİLER

Tez çalışması kapsamında, görüntü işleme yöntemlerinin çok çekirdekli işlemcilere sahip sıradan bilgisayarlar üzerinde, ek bir donanım desteğine ihtiyaç duymadan işlemci çekirdekleri üzerinde paralelleştiren bir yöntem sunulmuştur. Literatürde yüz bulma ve yüz tanıma problemlerine çözüm olarak sunulan en popüler yöntemlerden olan kademeli Haar sınıflandırıcıları, temel bileşenler analizi ve Fisher doğrusal ayrışım analizi yöntemleri incelenmiş, bu yöntemler Microsoft .NET çatısı altında gerçeklenmiş ve sunulan yaklaşımla paralelleştirilmiştir.

Önerilen paralelleştirme yaklaşımı, çeşitli senaryolarla denenerek performans artışları ve çerçeve akış hızlarının kararlılığı incelenmiştir. Tek çekirdek kullanılarak yürütülebilen iş yükü miktarı 8 birim olurken, bu değer 2 fiziksel, 4 mantıksal çekirdek kullanıldığında 2,5 kat, 4 fiziksel, 4 mantıksal çekirdek kullanıldığında 3,75 kat, 4 fiziksel, 8 mantıksal çekirdek kullanıldığındaysa 5,25 kat arttırılmıştır.

Çerçeve akışının devamlılığının ve kararlılığının sağlanması için önerilen doğrusal dalgalandırıcının akış hızı kararlılığını üstel dalgalandırıcı kadar iyi sağlayamadığı görülmüştür. Üstel dalgalandırıcı, farklı taban değerleri ile çok daha kararlı çerçeve akış hızları elde edilmesini sağlamıştır. Doğrusal dalgalandırıcı kullanıldığında elde edilen anlık çerçeve akış hızlarının standart sapması 2,99 fps olurken, üstel dalgalandırıcı kullanıldığında A=10, A=100 ve A=1000 değerleri için sırasıyla 0,86, 0,48 ve 0,44 olarak ölçülmüştür. Bu sonuçlar çerçeve akışı devamlılığının ve akış hızı kararlılığının başarıyla sağlandığını göstermektedir.

Yürütülebilecek en yüksek iş yükü değerleri yürütülüyorken işlemci çekirdeklerinin kullanımları değerlendirildiğinde, iş yükünün başarılı bir şekilde çekirdeklere paylaştırılabildiği görülmektedir.

Önerilen yaklaşım, gerçek zamanlı yüz tanıma problemi üzerinde de test edilmiştir. Uygulamada, 25 ve 60 fps hıza sahip videolar üzerinde yapılan testler sonucunda, iş yükü çekirdeklere dağıtılmış ve istenen akış hızları başarıyla sağlanabilmiştir. 25 fps hızı hedeflenirken sağlanan anlık akış hızlarının standart sapması doğrusal dalgalandırıcı kullanılırken 0,45, üstel dalgalandırıcı A=10, A=100 ve A=1000 değerleri ile kullanılırken sırasıyla 0,28, 0,36 ve 0,30 olarak ölçülmüştür. 60 fps hızı hedeflendiğinde ise, üstel dalgalandırıcı, 25 fps hedeflenerek yapılan testlerde hedeflenen akış hızına en yakın akış hızını sağlayan A=100 değeri ile kullanılmış ve

elde edilen anlık akış hızlarının standart sapması 2,40 olmuştur. İşlemci çekirdeklerinin kaldıramayacağı bir iş yükü oluşturmak amacıyla tespit edilecek minimum yüz oranının %10'a düşürüldüğü ve 60 fps akış hızının hedeflendiği son testte uygulama, 2 fiziksel çekirdek üzerinde pararlelleştirme yapılarak çalıştırılmış ve elde edilen akış hızlarının standart sapması 3,55 olarak ölçülmüştür. Bu test ile görüntü işleme hızının çok fazla kararsızlaştığı durumda bile akış hızının tekrar kararlı hale getirilebildiği gösterilmiştir.

Yöntem sıra bağımlılığı olan veri kümelerini işlemek için kullanılan yöntemlerin gerçek zamanlı çalışacak şekilde paralelleştirilmesi için de kullanılabilir. Gerçek zamanlı olarak sesin işlenmesi, EEG verilerinin işlenmesi, video formatlarının gerçek zamanlı olarak çözümlenmesi / kodlanması gibi, verilerin bağımsız paketler halinde işlenebileceği ve veriler üzerinde yapılan işlemlerin periyodik parçalara bölünebileceği yöntemler, önerilen yaklaşımın uyarlanabileceği yöntemlere örnek olarak verilebilir.

KAYNAKLAR

Acharya, K. A., Babu, R. V. and Vadhiyar, S. S., 2014. A real-time implementation of SIFT using GPU. **Journal of Real-Time Image Processing**, p.1-11.

Ahonen, T., Hadid, A. and Pietikainen, M., 2006. Face description with local binary patterns: Application to face recognition. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 28(12), p.2037-2041.

Akil, M., Grandpierre, T. and Perroton, L., 2012. Real-time dynamic tone-mapping operator on GPU. **Journal of Real-Time Image Processing**, 7(3), p.165-172.

Bartlett, M. S. and Sejnowski, T. J., 1997. Independent components of face images: A representation for face recognition. **In Procs. of the Annual Joint Symposium on Neural Computation**, p.3-10.

Belhumeur, P. N., Hespanha, J. P. and Kriegman, D., 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 19(7), p.711-720.

Buttari, A., Langou, J., Kurzak, J. and Dongarra, J. (2009). A class of parallel tiled linear algebra algorithms for multicore architectures. **Parallel Computing**, 35(1), p.38-53.

Castaño-Díez, D., Moser, D., Schoenegger, A., Pruggnaller, S. and Frangakis, A. S., 2008. Performance evaluation of image processing algorithms on the GPU. **Journal of structural biology**, 164(1), p.153-160.

Chen, J., Paris, S. and Durand, F., 2007. Real-time edge-aware image processing with the bilateral grid. **In ACM Transactions on Graphics (TOG)**, 26(3), p. 103

Freund, Y. and Schapire, R. E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of computer and system sciences**, 55(1), p.119-139.

Fukunaga, K., 2013. **Introduction to statistical pattern recognition.** Academic press.

Hong, S., Oguntebi, T. and Olukotun, K., 2011. Efficient parallel graph exploration on multi-core CPU and GPU. **In Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on**, p.78-88

Karas, P., Morard, V., Bartovsky, J., Grandpierre, T., Dokládalová, E., Matula, P. And Dokládal, P., 2012. GPU implementation of linear morphological openings with arbitrary angle. **Journal of Real-Time Image Processing**, 10(1), p.27-41.

Lai, J. H., Yuen, P. C. and Feng, G. C., 2001. Face recognition using holistic Fourier invariant features. **Pattern Recognition**, 34(1), p.95-109.

Laurent, A., Negrevergne, B., Sicard, N. and Termier, A., 2010. Towards a multicore parallel approach for mining gradual patterns. **In Database Systems for Advanced Applications**. p.78-84.

Liao, S., Jain, A. K. and Li, S. Z. (2013). Partial face recognition: Alignment-free approach. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 35(5), p.1193-1205.

Lienhart, R. and Maydt, J., 2002. An extended set of haar-like features for rapid object detection. **In Image Processing. Proceedings. 2002 International Conference on**, 1, p.900-903.

Lu, Y., Zhou, H., Shang, L. and Zeng, X., 2009. Multicore parallel min-cost flow algorithm for CAD applications. **In Proceedings of the 46th Annual Design Automation Conference**. p.832-837

Pham, V., Vo, P., Hung, V. T., 2010. GPU implementation of extended gaussian mixture model for background subtraction. **In Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)**, p.1-4.

Sinha, S. N., Frahm, J. M., Pollefeys, M., Genc, Y., 2006. GPU-based video feature tracking and matching. **In EDGE, Workshop on Edge Computing Using New Commodity Architectures**, 278, p.695-699.

Sujatha, K., Nageswara Rao, P. V., Rao, A., Sastry, V. G., Praneeta, V. and Bharat, R. K., 2015. Multicore parallel processing concepts for effective sorting and searching. **In Signal Processing And Communication Engineering Systems (SPACES), 2015 International Conference on**, p.162-166.

Turk, M. A. and Pentland, A. P., 1991. Face recognition using eigenfaces. **In: Computer Vision and Pattern Recognition Proceedings CVPR'91**, p.586-591.

Viola, P., and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. **In Computer Vision and Pattern Recognition**, 1, p.511-518.

Welling, M., 2005. **Fisher linear discriminant analysis.** Department of Computer Science, University of Toronto.

Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S. and Ma, Y., 2009. Robust face recognition via sparse representation. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 31(2), p.210-227.

ÖZGEÇMİŞ

Hüseyin Atasoy 1991 yılında Hatay'da doğdu. Lise öğrenimini Selim Nevzat Şahin Anadolu Lisesi'nde tamamladıktan sonra 2012 yılında Mustafa Kemal Üniversitesi Bilgisayar Mühendisliği Bölümü'nden mezun oldu. 2014 yılında Mustafa Kemal Üniversitesi Enformatik programında başladığı yüksek lisans çalışmalarına İskenderun Teknik Üniversitesi Elektrik-Elektrik Elektronik Mühendisliği programında devam etmekte ve İskenderun Teknik Üniversitesi Bilgisayar Mühendisliği bölümünde araştırma görevlisi olarak görev yapmaktadır.