

TEKNOFEST

HAVACILIK, UZAY VE TEKNOLOJİ FESTİVALİ

SÜRÜ İNSANSIZ HAVA ARACI (İHA) YARIŞMASI ÖN TASARIM FORMU

TAKIM ADI: ÇOBAN-Z

TAKIM ÜYELERİ: Serkan EREN, Emre BOY, Yüksel ZENGİÇ,
Okan TORUN, A. Samet KARAPINAR

İçindekiler

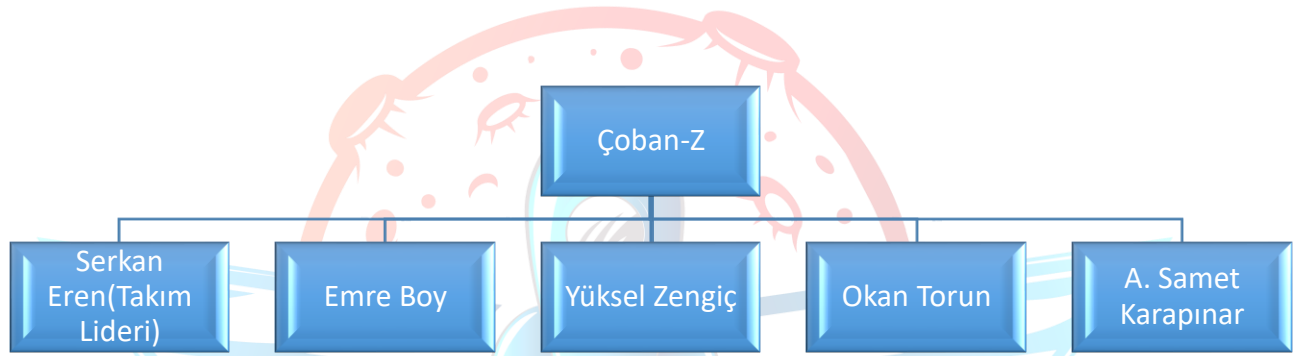
1-Yönetici Özeti.....	2
2-Proje Yönetimi	2
3-Proje Geresinimleri.....	3
3. 1. İHA bireysel kalkış, havada formasyon oluşturma görevi.....	5
3.2. Formasyon halinde kalkış görevi	5
3.3. Sürü halinde yükselme alçalma görevi	5
3.4. Sürü/sürüler halinde eş zamanlı / sıralı otomatik iniş görevi	5
3.5. Sürüden birey çıkarılması görevi	5
3.6. Sürüye yeni birey eklenmesi görevi	5
3.7. Sürü ayrılması görevi	6
3.8. Sürü birleştirme görevi	6
3.9. Formasyon koruma ile Sürü halinde yön değiştirme görevi	6
3.10. Formasyon değiştirme görevi	6
3.11. Sürü halinde navigasyon görevi	6
4-Tasarım Çözümü	6
4.a – İHA Kontrolü İçin Euler Açıları ve Quaternionlar(Dördeyler).....	7
4.b – Formasyon Kontrolü	9
4.b.1-Sanal Katı Yapı Temelli Yaklaşım	9
4.b.2-Lider-Takipçi Temelli Yaklaşım	10
4.b.3-Davranış Temelli Yaklaşım	11
4.c – Çarpışmadan Kaçınma.....	12
4.d – Optimizasyon için Macar Algoritması.....	13
4.{1, 2, 3, ..., 11} – Yarışma içindeki görevler için güncel konseptler ve çözüm önerileri.....	14
5-Temel Görev İsterlerinin Doğrulandığının Gösterilmesi	21
6-Kaynaklar	22

1-Yönetici Özeti:

Bu rapor Çoban-Z ekibi tarafından Teknofest-2021 Sürü İHA Yarışması için hazırlanmıştır. Rapor, sürü İHA yarışmasının şartnamelerinde verilen isterler doğrultusunda, gerekli olan proje planlaması, literatür taramaları, algoritma tasarımları ve yarışma ortamı tanıtımını içermektedir. Raporda özellikle Sürü İHA sistemleri için farklı konseptler tanıtılmıştır. Yarışma kapsamında, gerçek mini İHA'lar üzerinde uygulanacak olan bu algoritmalar, simülasyon programlarında denenmektedir [1].

Çoban-Z ekibi, Gebze Teknik Üniversitesi, Bilgisayar, Elektronik ve Makine Mühendisliği bölümlerinden 5 Lisans öğrencisinin bir araya gelmesiyle kurulmuştur. Takım olarak yarışmadaki amacımız; temel formasyon görevlerini yerine getirmek, daha sonrasında ise bunların optimizasyonlarını sağlamaktır.

2-Proje Yönetimi:



Serkan Eren, Elektronik Mühendisliği 2. Sınıf öğrencisidir

Emre Boy, Elektronik Mühendisliği 1.sınıf öğrencisidir.

Yüksel Zengiç, Makina Mühendisliği 2.sınıf öğrencisidir.

Okan Torun, Bilgisayar Mühendisliği 2.sınıf öğrencisidir.

A. Samet Karapınar, Elektronik Mühendisliği 3.sınıf öğrencisidir.

Ekibimiz farklı mühendislik disiplinlerinde öğrenim gören; robotik, yazılım ve havacılık gibi alanlara ilgi duyan öğrencilerden oluşmaktadır. Ekip olarak temel amacımız, insanlık yararına ülke sanayisinde verimli bir biçimde kullanılabilecek sistemler geliştirmektir. Ekibimiz bu yarışma vesilesiyle kurulmuş bir ekip olup birbirini yakından tanıyan üyelerden oluşmaktadır. Bundan sonraki arzumuz ekibimizi geliştirerek uzun yıllar boyunca yeni projelere imza atmaktır.

	Ocak	Şubat	Mart	Nisan	Mayıs	Haziran	Temmuz	Ağustos	Eylül
Takımın Kurulması									
Yarışma Başvurusu		28.02.2021							
Literatür Taraması									
Algoritma Geliştirme									
Ön Tasarım Raporu				4.04.2021					
Teknik Aşama-1					5.05.2021				
Teknik Aşama-2						21.06.2021			
Teknik Aşama-3							23.07.2021		
Proje Raporu								28.08.2021	
Teknofest									21-26

Tablo 1(Planlanan Proje Takvimi)

3-Proje Gereksinimleri

Yarışmada kullanılacak olan İHA modeli Crazyflie'dır. Bu İHA'lar dizayn olarak oldukça hafif ve küçük boyutludur. Bu yüzden üzerlerinde kamera, lidar, gps gibi gelişmiş donanımlar yerine en jiroskop, ivmeölçer, radyo gibi temel donanımları bulundurmaktadır.

Yarışma ortalama bir ev odası boyutlarında, kapalı bir alanda yapılacaktır. İHA'lar kendi aralarından doğrudan haberleşemeyeceklerdir. Haberleşme sistemi yarışma alanına önceden yerleştirilecek olan “**Konumlandırma Sistemleri**” ve “**hareket yakalayıcı sensörler**” [2] yardımıyla gerçekleştirilecektir. Bu sistemler anlık olarak tüm İHA'ların konumlarını eşzamanlı olarak merkez bilgisayara aktaracaklardır. Algoritmalar ve formasyonlar bu koşullar göz önüne alınarak geliştirilmelidir. Haberleşme için Şekil 1'den görüldüğü gibi radyo dalgaları kullanılmaktadır. Ancak unutulmaması gereken noktalardan biri de sürünün büyümesi durumunda haberleşme sisteminin performansdır. Crazyflie ekibi bu gibi durumlar için farklı önerileri hazır olarak sunmaktadır ancak yarışma gerekliliklerine göre maksimum performans için farklı tasarımlar yapılabilir.[3]



Şekil 1 (Crazyflie ekosistemi)

Yarışmada 11 temel görev vardır. Amaç bu görevleri istenilen biçimde, maksimum hızda gerçekleştirmektir. Bu görevler:

1. İHA Bireysel kalkış, havada formasyon oluşturma
2. Formasyon halinde kalkış
3. Sürü halinde yükselme alçalma
4. Sürü/sürüler halinde eş zamanlı / sıralı otomatik iniş
5. Sürüden birey çıkarılması
6. Sürüye yeni birey eklenmesi
7. Sürü ayrılması
8. Sürü birleştirme
9. Formasyon koruma ile Sürü halinde yön değiştirme
10. Formasyon değiştirme
11. Sürü halinde navigasyon

olarak sıralanabilir. Bu görevlerin gereksinimlerini incelemeye başlamadan önce birkaç temel hususa açıklık getirmek yerinde olacaktır.

Bahsi geçen görevler gerçekleştirilirken İHA'lar birbirleriyle çarpışmamalıdır. Bu durumdan kaçınmak için kod kısmında sisteme uygun bir algoritma geliştirilmelidir çünkü birçok gelişmiş sürü sisteminde çarpışmalar, algoritmanın yanı sıra İHA'lar üzerindeki algılayıcılar yardımıyla anlık olarak önlenmektedir.

Bunun yanında sürü algoritmalarımızı yönetebilmek için bir kontrol arayüzü (Bir nevi kumanda) gerekmektedir. Örneğin “1” tuşuna basıldığında sürü istenilen derecede rotasyon yapabilmeli ya da “2” tuşuna basıldığında sürü formasyonunu değiştirebilmelidir.

```

***** KONTROL KUMANDASI *****

---Hareket Tuşları---
İleri/geri/sağa/sola: w/s/d/a
Kalkış/İniş: q/e
Bireysel sıralı kalkış: t
Döndür saat yönü/saat yönü tersi: c/z
Birey ekleme/çıkarma: n/m
Bireyler arası mesafe arttır/azalt: +/-
Sürü ayrılması/birleşmesi: l/k
Navigasyon: f

---Formasyon değiştirme Tuşları---
Üçgen: 1
Kare: 2
Beşgen: 3
Altıgen: 4
Yıldız: 5

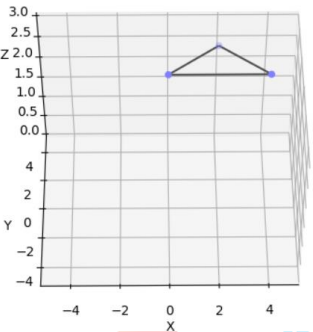
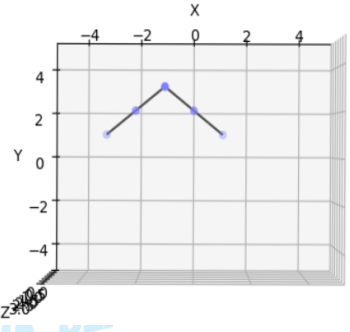
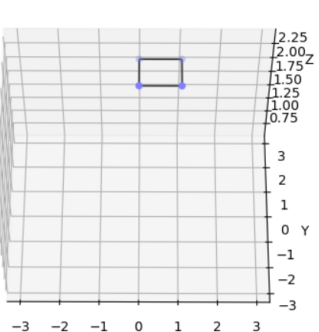
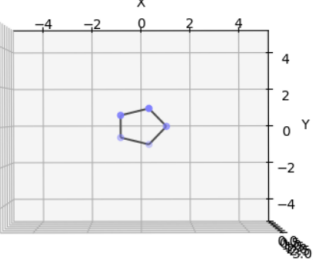
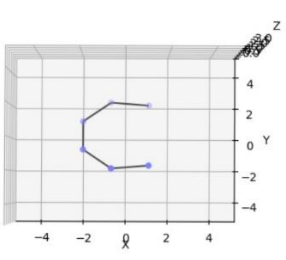
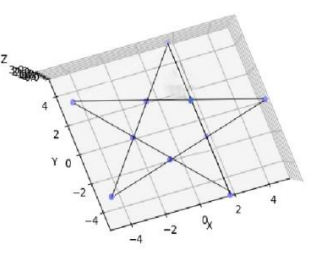
---Durum Göstergesi---
Konum:
Formasyon:
Birey Sayısı:

Komut girin: █

```

Şekil 2(Örnek kontrol arayüzü)

Bir diğer önemli husus ise formasyon kontrolüdür. Yarışmada oluşturulması beklenen formasyonlar:

Üçgen	V Formasyonu	Kare
		
Beşgen	Hilal	Yıldız
		

Tablo 2 (Crazyswarm ortamında hazırlanan formasyonlar)

Oluşturmamız gereken formasyonlar Tablo 2'deki gibidir. Formasyonların tamamının iki boyutlu olması gözden kaçırılmamalıdır. Bu formasyonlara ek olarak şartnamede "rastgele formasyon" adı altında bir formasyon daha görülmektedir. Bunun anlamı, algoritma tasarlanırken, sürü herhangi bir anda arzulanan başka bir formasyona geçebilmelidir ayrıca buna ek olarak eğer İHA'lar yerdeyken rastgele konumlara bırakılırsa başlangıç noktalarını kaydedip bu noktaları formasyon olarak kendi envanterine ekleyebilmelidir. Bu önemli hatırlatmalardan sonra temel 11 görevi inceleyecek olursak:

3.1- İHA Bireysel kalkış, havada formasyon oluşturma Görevi

Burada istenilen ilk görev rastgele konumlanmış İHA'ların bireysel olarak kalkış yapmalarını sağlamaktır. Yani aslında İHA'lar Z eksenı boyunca belli bir mesafe katetmelidirler. Sonrasında ise arzulanan formasyonu oluşturmalarıdır. Burada önemli olan nokta, tüm bu işlemleri mümkün olan minimum konum hataları ve maksimum hız ile gerçekleştirmektir. Bunun için optimizasyon yöntemleri etkin bir biçimde kullanılmalıdır. Sonrasında ise sürü formasyonunu ve irtifasını belli bir süre korumalıdır. Bunun için de hata önleyici kontrol algoritmalar geliştirilmelidir.

3.2- Formasyon halinde kalkış Görevi

Bu kısımdaki görev ilk göreve oldukça benzemektedir. Ancak temel fark, formasyon ilk andan itibaren şeklini koruyarak yükselmelidir. Burada formasyon kontrol algoritmaları geliştirilerek işlemlerin eş zamanlı ve uyumlu olması sağlanmalıdır.

3.3-Sürü Halinde Yükselme-Alçalma Görevi

Bu kısımdaki görevin ilk kısmı 3.2 ile aynıdır. Ancak İHA'lar belirli bir irtifada, belirli bir süre formasyonlarını koruduktan sonra bir hamle daha yapıp bir miktar daha yükselmeli ve belli bir süre bekleyip tekrar belli bir miktar alçalmalıdır. Burada bir nevi, formasyonun manevra kabiliyeti için ilk sınamaya yapılmaktadır.

3.4-Sürü Halinde Eş Zamanlı / Sıralı Otomatik İniş Görevi

Bu kısımdaki görevimiz ilk görevlere ek olarak formasyon halinde sıralı iniş içermektedir. Bu kısımda da ilk görevlerdeki hususlara dikkat edilecektir. Aynı zamanda buradaki en önemli problemlerden biri İHA'ların sırasıyla iniş sırasında ortaya çıkmaktadır. İHA'lar sırasıyla iniş yapacaklar, sürüdeki diğer İHA'lar mevcut İHA'nın inmesini beklerken formasyonlarını korumalıdır.

3.5-Sürüden Birey Çıkarma Görevi

Bu kısımdaki görevimizde yerde rastgele dizili İHA'lar belirli bir irtifaya dikey kalkış gerçekleştirecekler. Sonrasında kontrol arayüzünden arzulanan formasyon komutu gönderilecek ve İHA'lar istenilen formasyona gireceklerdir. Bu formasyon belli bir süre korunduktan sonra görevli bir kişi uçuş alanına girip rastgele bir sürü üyesini sürüden çıkaracaktır. Algoritmanın bu işleme tepki vermesi beklenmektedir. Sürü sistemi çıkan İHA'yı algılayıp pozisyon ilişkilerini koruyarak düzenini devam ettirmelidir.

3.6-Sürüye Birey Ekleme Görevi

Bu görevin ilk kısmında yerde rastgele dizili İHA'lar kalkış yaparlar ve çember formasyonuna girerler. Ancak burada dikkat edilmesi gereken husus İHA'lar arasındaki mesafelerdir. Örneğin 4 İHA ile çember formasyonuna girmeyi denediğimizde gördüğümüz formasyon bir kare olacaktır. Sonrasında ise farklı bir konumda bekleyen yeni İHA sürüye katılmayı deneyecektir. Burada jenerik algoritmalar kullanılarak formasyonun kendini koruması ve yeni İHA'yı diğer İHA'ların arasında alması beklenmektedir. Yeni formasyon 5 İHA dan oluşacaktır (sonrasında 6 İHA...) ancak formasyonun yapısal özellikleri korunacaktır. Yeni İHA geldikten sonra diğer İHA'lar pozisyonlarını güncelleyecekler ve düzenli bir yapı görünümünü bozmayacaklardır. Bu kısımda özellikle çember şeklinin geometrik özellikleri ve koordinat dönüşümleri efektif biçimde kullanılmalıdır.

3.7. Sürü Ayrılma Görevi

Bu görevde sürü, iki farklı sürüye ayrılmalıdır. Bu ayrılma olayı sırasında hesaplanması gereken birçok parametre vardır. Öncelikle ilk sürü sistemi, iki farklı formasyona bölünebilecek kadar büyük olmalıdır ki bu sayı da minimum 6 İHA etmektedir. Sürü bölünmeden önce yeni oluşturulacak formasyonlar için hesaplamalar yapacak ve İHA'lar minimum sürede yeni sürüleri oluşturacaklardır. Burada maksimum hızda bölünme için pozisyon optimizasyonlarına önem verilmesi gerekmektedir.

3.8. Sürü Birleşmesi Görevi

Bu görev 3.7. görevin aksine iki farklı sürü sisteminin birleşerek tek bir formasyona girmesini gerektirmektedir. Bu görevde öngördüğümüz en büyük problem sürü birleşmesi sırasında oluşabilecek çarpışma tehlikesidir. Algoritma üzerinde sürekli çalışacak olan çarpışma önleyici fonksiyon, kısa sürede tek bir formasyona girmeye çalışan İHA kitlesi üzerinde başarılı bir performans göstermelidir. Diğer görevlerdeki genel gereklilikler burada da geçerlidir.

3.9. Formasyon Koruma ile Sürü Halinde Yön Değiştirme Görevi

Bu görevde sürü sistemi kalkışını yapacak, belli bir süre formasyonunu koruyacak ve daha sonrasında ise saat yönünde ya da tersinde arzulan açı kadar rotasyon yapacaktır. Örneğin kare formasyonundaki sürü 45 derece döndüğünde baklava dilimine benzer bir yapı almalıdır. Ya da "V" yapısındaki formasyon 90 derece döndüğünde "<" ya da ">" şeklinde görünmelidir. Burada İHA'ların pozisyonları raporun ileriki kısımlarında anlatılan çeşitli matematik dönüşümler ile ayarlanmalıdır.

3.10. Formasyon Değiştirme Görevi

Bu görevde sürü belli bir formasyonda kalkış yapacaktır. Sonrasında ise sürünün istenilen başka bir formasyona geçmesi beklenmektedir. Buradaki problem aslında rastgele dizilmiş İHA'ların havada formasyon oluşturması probleminde çok farklı değildir. Çünkü yeni bir formasyon oluşturmak isteyen İHA'lar her ne kadar belli bir formasyon içerisinde olsalar da aslında yeni formasyon için rastgele konumlarda bulunmaktadır. Burada formasyon değişiminin minimum sürede gerçekleştirilmesi için geometrik optimizasyon yöntemleri kullanılmalıdır (örneğin formasyonların ağırlık merkezleri kullanılarak sağlanabilir)

3.11. Sürü Halinde Navigasyon Görevi

Bu görev aslında sürünün birçok kabiliyetini sınanan bir görevdir. Sürü formasyon halinde kalkışını gerçekleştirdikten sonra jüri tarafından belirlenen nirengi noktalarına ulaşmaya çalışacaktır. Hareket halindeki sürü formasyonunu koruyarak yol almalıdır aynı zamanda gerekli noktalarda en kısa yolu hesaplayabilmeli ve düzeni bozmayacak maksimum hızda hedefe yönelebilmelidir. Bu yönelim esnasında hedefin konumunu göre rotasyon da gerekli olacaktır.

4-Tasarım Çözümü

3. kısımda açıkladığımız proje gerekliliklerinin çözüm önerileri ve problemlerin çözümleri için gerekli konseptler bu kısımda hazırlanacaktır.

Genellikle sürü sistemlerinin görevlerini icra ederken bazı özelliklere sahip olması beklenmektedir. Bu özellikler şu şekilde toparlanabilir [4]:

- 1-Sağlamlık: Sürü bireysel robotların kaybına veya başarısızlığa karşı dayanıklıdır.
- 2-Esneklik: Sürü, farklı görevlerin üstesinden gelmek için tekrar yapılandırılabilir.
- 3-Ölçeklenebilirlik: Sürü, küresel görevin ihtiyaçlarına bağlı olarak büyüyebilir ve küçülebilir.

Bu kısımda oluşturmaya çalıştığımız algoritmalar bu temellere dayanarak geliştirilmektedir.

Görevleri teker teker incelemeye başlamadan önce, görevlerin tamamında kullanılacak bazı ortak noktalara açıklık getirmekte fayda vardır.

Bunları karmaşıla yaratmaması ağısından listelersek:

4.a – İHA Kontrolü İin Euler Aıları ve Quaternionlar (Dördeyler)

4.b – Formasyon Kontrol

4.c – arpışmadan Kaınma

4.d – Optimizasyon iin Macar Algoritması

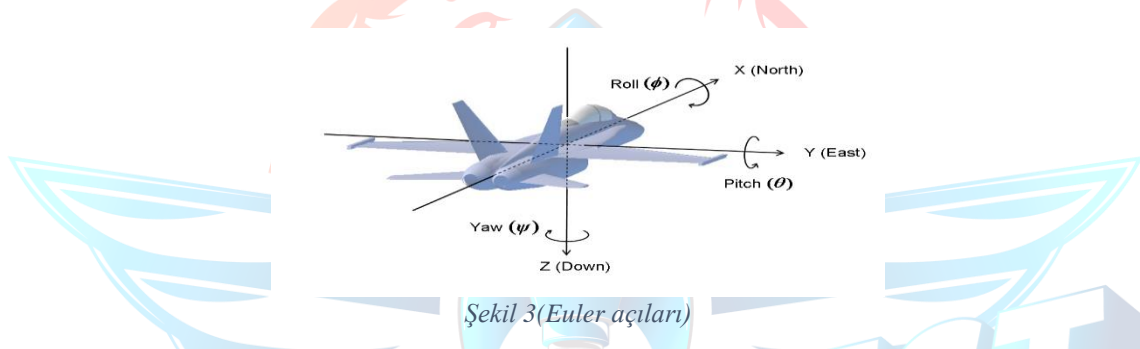
4.{1, 2, 3, ..., 11} – Yarışma iindeki görevler iin güncel konseptler ve çözüm önerileri

4.a – İHA Kontrolü İin Euler Aıları ve Quaternionlar(Dördeyler)[5]

İHA'ların tutum ve yön sensörleri, hem Euler Aılarını hem de kuaterniyonları kullanarak oryantasyon bilgisi sağlayabilir. Kuaterniyonlarla karşılaştırıldığında, Euler Aıları basit ve sezgiseldir yani basit analiz ve kontrole çok uygundur.

Kuaterniyon

Kuaterniyon ,3D koordinat sistemindeki herhangi bir dönüşü kodlamak iin kullanılabilen dört elemanlı bir vektördür. Teknik olarak bir kuaterniyon, bir gerek eleman ve üç karmaşık elemandan oluşur ve rotasyonlardan çok daha fazlası iin kullanılabilir.



Vektör q_i^b , eylemsizlik erevesinden (Inertial Frame) sensörün gövde erevesine (body frame) dönüşünü kodlayan birim vektör kuaterniyonu olarak tanımlansın:

$$q_i^b = (a \ b \ c \ d)^T$$

Burada T, vektörün transpose operatörüdür. B, c ve d öğeleri, kuaterniyonun "vektör kısmı" dır ve döndürmenin gerekleştirilmesi gereken bir vektör olarak düşünülebilir. "a" öğesi, vektör bölümü etrafında gerekleştirilmesi gereken dönüş miktarını belirten "skaler bölümdür". Spesifik olarak, θ dönme aısı ise ve vektör $(V_x \ V_y \ V_z)^T$ dönme eksenini temsil eden bir birim vektörse, kuaterniyon elemanları şu şekilde tanımlanır:

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \cos(0.5\theta) \\ V_x \sin(0.5\theta) \\ V_y \sin(0.5\theta) \\ V_z \sin(0.5\theta) \end{pmatrix}$$

Pratikte, bu tanımın açıka kullanılmasına gerek yoktur, ancak burada, kuaterniyonun neyi temsil ettiğinin sezgisel bir tanımını sağladığı iin dahil edilmiştir.

Kuaterniyonları Kullanarak Vektörleri Döndürme;

Davranış kuaterniyonu q_i^b aşağıda tanımlanmış işlem ile rastgele bir 3 elemanlı vektörü eylemsiz ereveden (inertial frame) gövde erevesine (body frame) döndürmek iin kullanılabilir.

Kuaterniyon q_1 ve q_2 'yi tanımlarsak;

$$q_1 = (a_1 \ b_1 \ c_1 \ d_1)^T \quad q_2 = (a_2 \ b_2 \ c_2 \ d_2)^T$$

Daha sonra $q_1 q_2$ kuaterniyon çarpımı şu şekilde verilir,

$$q_1 q_2 = \begin{pmatrix} a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 \\ a_1 b_2 - b_1 a_2 - c_1 d_2 - d_1 c_2 \\ a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2 \\ a_1 d_2 - b_1 c_2 - c_1 b_2 - d_1 a_2 \end{pmatrix}$$

Bir vektörü gövde çerçevesinden (body frame) eylemsizlik çerçevesine (inertial frame) döndürmek için, yukarıda tanımlandığı gibi iki kuaterniyon çarpımı gereklidir. Alternatif olarak, davranış kuaterniyonu, dönüşü tek bir matris çarpma işleminde gerçekleştirmek için bir 3x3 rotasyon matrisi oluşturmak için kullanılabilir. Kuaterniyon elemanlarını kullanan eylemsiz çerçeveden (inertial frame) gövde çerçevesine (body frame) dönme matrisi şu şekilde tanımlanır:

$$R_i^b(q_i^b) = \begin{pmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2bd - 2ac \\ 2bd - 2ad & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 - b^2 - c^2 + d^2 \end{pmatrix}$$

Daha sonra eylemsiz çerçeveden (inertial frame) gövde çerçevesine (body frame) dönüş, matris çarpımı kullanılarak gerçekleştirilebilir.

$$V_B = R_i^b(q_i^b) V_I$$

Dönüşü gerçekleştirmek için kuaterniyon çarpımının mı yoksa matris çarpımının mı kullanıldığına bakılmaksızın, dönüşü gerçekleştirmeden önce basitçe davranış kuaterniyonunu ters çevirerek dönüş tersine çevrilebilir.

Kuaterniyonları Euler Açılara Dönüştürme

İHA sensörleri, kuaterniyon tahmin modunda bile otomatik olarak kuaterniyon tutum tahminini Euler Açılara dönüştürür. Bu, daha sağlam kuaterniyon tahmini kullanıldığında bile Euler Açısı tahmininin rahatlığının sağlandığı anlamına gelir.

Kullanıcı sensörün hem Euler Açısı hem de Kuaterniyon verilerini iletmesini istemezse (örneğin, iletişim bant genişliği gereksinimlerini azaltmak için), bu durumda kuaterniyon verileri alıcı uçta Euler Açılara dönüştürülebilir.

Kuaterniyonları Euler Açılara dönüştürmek için tam denklemler dönme sırasına bağlıdır. İHA sensörleri, önce Roll, sonra Pitch ve son olarak Yaw kullanarak eylemsiz çerçeveden gövde çerçevesine hareket eder. Bu, aşağıdaki dönüşüm denklemleriyle sonuçlanır:

$$\begin{aligned} \text{Roll}(\phi) &= \arctan\left(\frac{2(ab+cd)}{a^2+b^2-c^2-d^2}\right) \\ \text{Pitch}(\theta) &= -\arcsin(2(bd-ac)) \\ \text{Yaw}(\varphi) &= \arctan\left(\frac{2(ad+bc)}{a^2+b^2-c^2-d^2}\right) \end{aligned}$$

Kuaterniyonları Euler Açılara dönüştürülürken, gimbal lock probleminin hala kendini gösterdiğine dikkat edin.

Euler yöntemi kuaterniyona göre daha fazla trigonometrik ve matris hesap gerektiren 3 boyutlu dönmelerde büyük bir sadelik ve avantaj sağlar.

4.b- Formasyon Kontrolü

Aslına bakacak olursak formasyon kontrolü için algoritmalar, üç genel yaklaşım halinde kategorize edilebilir [6];

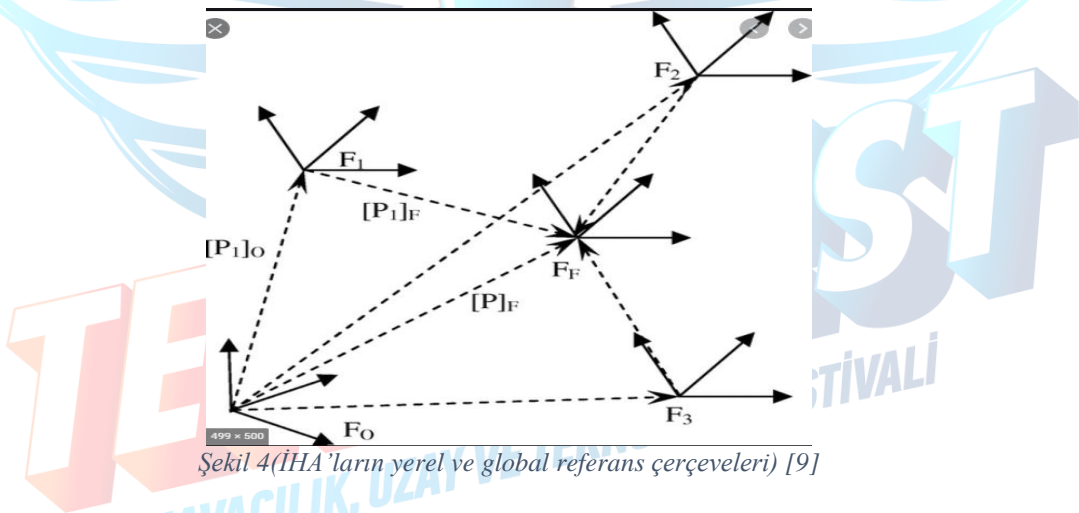
- 1-Sanal Katı Yapı Temelli Yaklaşım
- 2-Lider-Takipçi Temelli Yaklaşım
- 3-Davranış Temelli Yaklaşım

4.b.1- Sanal Katı Yapı Temelli Yaklaşım [7][8]

“Virtual Rigid Body (VRB)” yani “Sanal Katı Yapı” yaklaşımında İHA’lar belirli konumlara gömülüymüş gibi hareket ederler. Özellikle belirli bir formasyonu korumak için çok elverişli bir yöntemdir. Örneğin agresif manevralar yapan, dar bir kanyonda görev icra eden ya da kapalı bir alanda gösteri yapacak olan sürü sistemleri için oldukça kullanışlı ve performanslıdır. Ayrıca, sürü sisteminin katı bir yapı gibi davranması, sürünün bir kumanda veya kontrolör ile kontrol edilmesini mümkün kılar.

Bu yaklaşımda her bir İHA yapının üzerinde bir nokta üzerinde durur. Bu İHA’ların konumları çeşitli matematik operasyonlardan geçirilerek formasyon kontrolü gerçekleştirilmiş olur. Bu konseptin matematiksel temellerine değinecek olursak.

$\{1, 2, 3, \dots, N\}$ ile etiketlenmiş olan N adet robot düşünelim. \mathcal{F}_i İHA’nın yerel referans çerçevesini ifade etsin ve $\mathbf{p}_i(t) \in \mathbb{R}^3$ de i . İHA’nın pozisyonunu ifade etsin. $\mathbf{R}_i(t)$ de İHA’nın \mathcal{F}_w global referans çerçevesindeki oryantasyonunu ifade etsin.



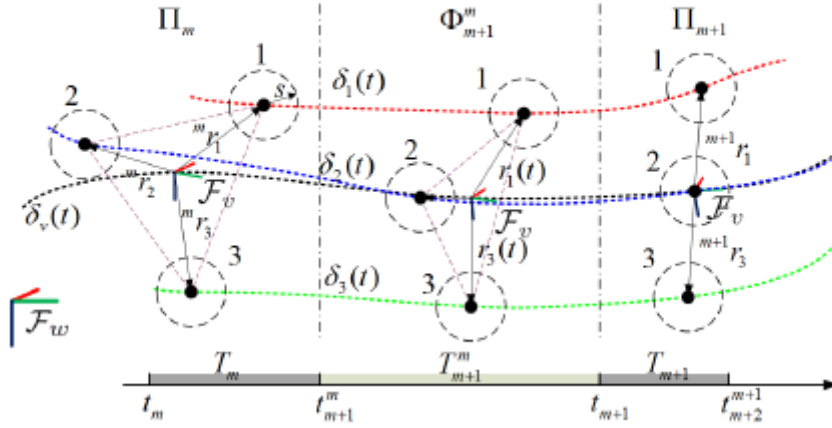
Şekil 4(İHA'ların yerel ve global referans çerçeveleri) [9]

Bu tanımlara göre Sanal Katı Yapı aslında N adetten oluşan bir grup İHA ve yerel bir referans çerçevesi olan \mathcal{F}_v den oluşmaktadır. \mathcal{F}_v burada İHA'ların yerel pozisyonlarını içeren, zamana bağlı bir vektördür $\{\mathbf{r}_1(t), \mathbf{r}_2(t), \dots, \mathbf{r}_n(t)\}$.

Yapıyı matematiksel olarak ifade ettikten sonra şimdi de formasyon şeklini ifade edelim: Π bir formasyonu ifade etsin. Bu formasyon aslında N robottan oluşan, \mathcal{F}_v çerçevesindeki sabit pozisyon vektöründen ibarettir $\{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n\}$.

Bir formasyonun içindeki İHA'ların yerel pozisyonları sabittir ancak yerel oryantasyon sabit olmak zorunda değildir. Aşağıdaki resim tüm bu durumları daha net açıklamaktadır.

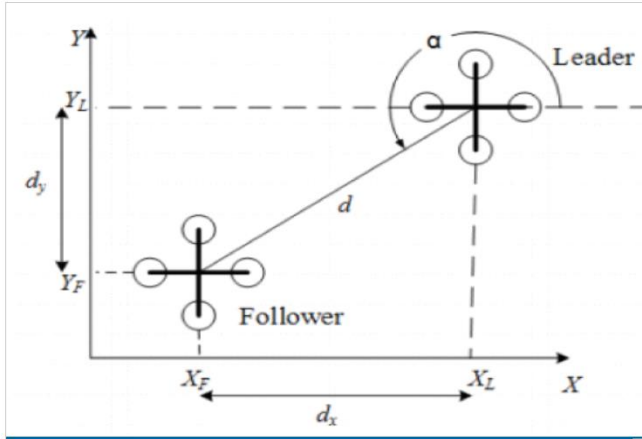
Şekil 5'teki “ Φ ” sembolü formasyon değişimini ifade etmektedir. “ δ ” sembolü ise her bir İHA’nın takip etmesi gereken yolu bildiren bir matristir. Bu yaklaşımda sürünün takip edeceği gezeceği önceden bir algoritma ile hesaplanmalıdır burada bu kısma değinmeyeceğiz ancak bu durum bazı senaryolarda bir eksiklik olarak hissedilebilir lakin bizim yarışmamız için bu bir eksiklik değildir.



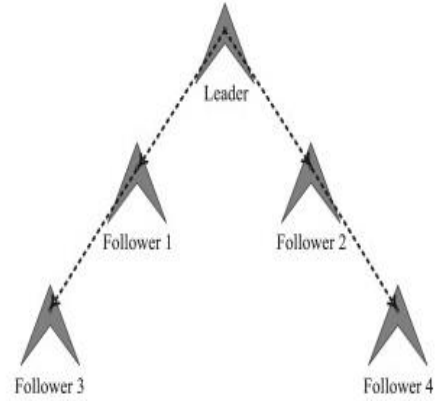
Şekil 5(Sanal Katı Yapı Konseptinin örnek uygulaması)

4.b.2- Lider-Takipçi Temelli Yaklaşım [10]

Lider – Takipçi temelli yaklaşımda, sürü elemanlarından biri lider olarak seçilmektedir. Geri kalan İHA'lar ise en basit anlatımla lideri takip ederler. Bu yaklaşımın diğer yaklaşımlara göre bazı dezavantajları bulunmaktadır. Bunlardan ilki, lider İHA'nın düşme durumu ya da herhangi bir sebepten dolayı yörüngesini kaybetmesi olabilir. Böyle bir durumda sürüdeki tüm İHA'lar lidersiz kaldıklarından dolayı görevlerini başarılı bir şekilde tamamlayamayacaklardır. Bir diğer önemli dezavantaj da sürünün formasyon değiştirme ya da rotasyon gibi görevlerde diğer yaklaşımlara göre daha düşük bir performans göstermesidir.



Şekil 6(Lider-Takipçi Konsepti)



Şekil 7(Lideri takip eden sürü)

2 boyutlu düzlem için, formasyon kontrolünde amaç X-Y düzleminde çeşitli konfigürasyonları oluşturmaktır. Bunun için hareket sırasındaki koordinat değişimlerini incelemek gereklidir.

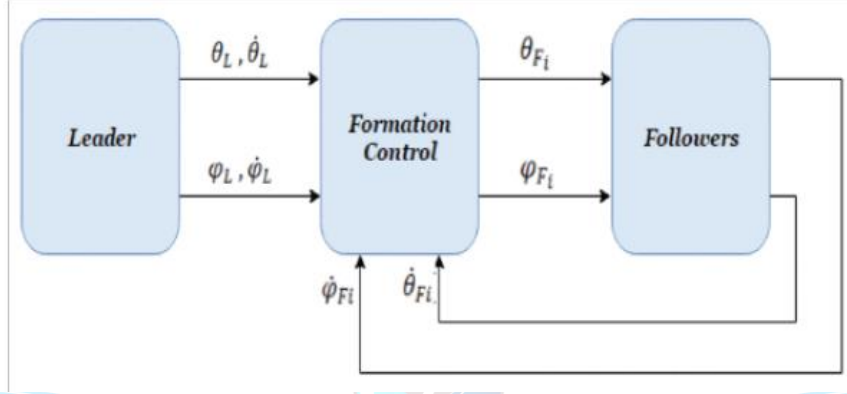
$$\begin{aligned} d_x &= -(X_L - X_F) \cos(\psi_L) - (Y_L - Y_F) \sin(\psi_L) \\ d_y &= (X_L - X_F) \sin(\psi_L) - (Y_L - Y_F) \cos(\psi_L) \end{aligned}$$

Burada sembollerin geometrik anlamları Şekil 6'ya bakılarak kolaylıkla anlaşılmaktadır. Burada İHA'ların lideri takip edebilmeleri ve aralarındaki arzulanan açı ve uzaklığı sabit tutabilmeleri için bir kontrol algoritması gereklidir o da aşağıdaki gibi formülize edilebilir:

$$\lim_{t \rightarrow \infty} \|e_x\| = \|d_x^d - d_x\| = 0$$

$$\lim_{t \rightarrow \infty} \|e_y\| = \|d_y^d - d_y\| = 0$$

Burada e_x ve e_y hataları temsil etmektedir. Ve hata, yani arzulanan ile mevcut durum arasındaki fark 0'a gitmelidir. “d” harfinin üzerindeki d_x^d de “desired” yani arzulanan uzaklığı belirtmektedir.



Şekil 8(Lider-Takipçi konsepti için kontrolör yapısı)

Lider- Takipçi konseptinde lider seçimi de oldukça önem arz etmektedir. Başlangıç durumu için lider rastgele seçilebilir ancak eğer ki formasyon bir rotasyona ya da navigasyona geçiyorsa burada lider güncellenmelidir. Örneğin navigasyon için en sık kullanılan yöntemlerden biri, ulaşılması arzulanan nirengi noktasına en yakın duran İHA'yı lider olarak seçmektir. Lider İHA değiştiğinden sonra geri kalan İHA'lar tekrardan lider İHA ya göre konumlanmalıdırlar.

4.b.3-Davranış Temelli Yaklaşım [11]

Sürü sistemlerini kontrol etmek için merkezi ve dağıtık kontrol sistemleri mevcuttur. Merkezi sistemler hata toleransının, esnekliğin ve uygunluğun düşük olduğu sistemlerdir dolayısıyla daha az sayıda birey bulunduran sürülerde daha kullanışlıdır. Dağıtık sistemlerde ise hata toleransı, esneklik ve uygunluk yüksektir bu durumda dağıtık sistemler çok fazla sayıda birey içeren sürülerde kullanılır ve davranışa dayalı kontrol yaklaşımlarını ele alır. Davranış temelli oluşum kontrol yöntemi, hedefe gitme, formasyonu koruma ve engellerden kaçınma gibi önceden tanımlanmış bir dizi temel davranışı kullanır.

Bu konseptin en güzel örneklerinden biri doğadaki arı sürüleridir. Arı sürüleri doğada uçarken birçok probleme çözüm getirerek uçmaktadırlar. Avcıdan kaçmak, sürü halinde uçarken veya kocalarda dar bir alanda hareket ederken çarpışmamak, çiçekten çiçeğe gezinirken en kısa yolları bulmak gibi problemlere belli bir sistematikte çözüm üretirler. Arılar gibi doğada bulunan sürüleri inceleyen birçok çalışma mevcuttur.[12]

Aynı arı sürülerinde olduğu gibi İHA'lar da belirli görevleri gerçekleştirmek için çeşitli problemlere, sistematik çözümler getirmelidirler aslında bu sistematik, davranış olarak isimlendirilmiştir.

Biz, davranış temelli formasyon kontrolünü incelemekteyiz. Burada robotların hareket ederken aralarındaki açı ve uzaklığı koruma davranışını göstermesi ve istenilen formasyonu sağlama gerekmektedir.

Şekil 9'da R_i ve R_j iki İHA'yı temsil etmektedir. α_{i-j} oluşum doğrultusu ile iki İHA'nın doğrultusu arasındaki açıdır. d_{j-i} ise iki İHA arasındaki mesafedir. Formasyondaki R_j 'nin ideal konumu ve formasyonun hareket yönü olan R_i 'nin konumu, α_{j-i} ve d_{j-i} tarafından belirlenir. Özetle burada bir İHA'nın



Çarpışmadan kaçınma sürü algoritmaları için en önemli hususlardan biridir ve 2 ye ayrılır;

1- Sürü çarpışması kaçınması

Sürüş çarpışma kaçınmasında 2 ayrı sürünün birbirleriye çarpışmasını önlemek için öncelikle kaçınma rlığı şekil 1 deki gibi hesaplanır hesaplanılır:

Burada w_{AV_i} kaçınma ağırlığı, P_i İHA'nın konumu $C_{p,i}$ ise bir sürünün dış sınır çizgisidir, bu çizgiler şekil 2'de örneklendirilmiştir.



Burada f_1 diğer sürünün x, y ve z koordinatlarına göre şekil 4 teki gibi hesaplanılır

$$f_1 = \begin{cases} 1 & \text{if } (\pi/4 < \psi_i \leq 3\pi/4) \text{ and if } x_i \geq X_{ca} \\ -1 & \text{if } (\pi/4 \leq \psi_i < 3\pi/4) \text{ and if } x_i < X_{ca} \\ -1 & \text{if } (5\pi/4 < \psi_i \leq 7\pi/4) \text{ and if } x_i \geq X_{ca} \\ 1 & \text{if } (5\pi/4 \leq \psi_i < 7\pi/4) \text{ and if } x_i < X_{ca} \\ 1 & \text{if } (3\pi/4 < \psi_i \leq 5\pi/4) \text{ and if } y_i \geq Y_{ca} \\ -1 & \text{if } (3\pi/4 \leq \psi_i < 5\pi/4) \text{ and if } y_i < Y_{ca} \\ -1 & \text{if } (7\pi/4 < \psi_i \leq \pi/4) \text{ and if } y_i \geq Y_{ca} \\ 1 & \text{if } (7\pi/4 \leq \psi_i < \pi/4) \text{ and if } y_i < Y_{ca} \end{cases}$$

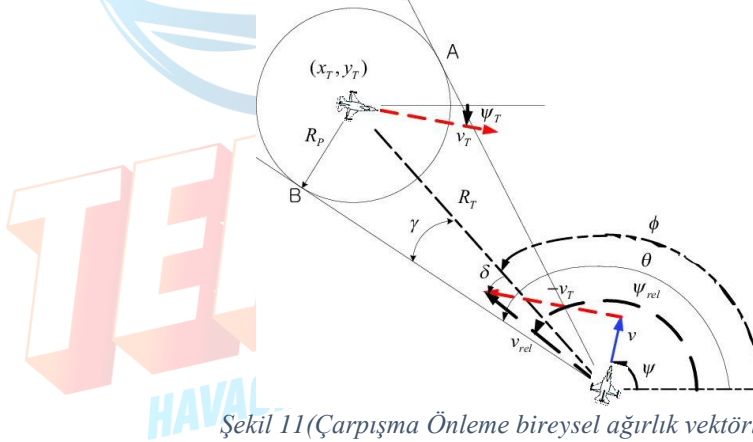
Gerekli açı bulunduktan sonra komutlar sürü bireylerine iletilir ve sürü çarpışmasından kaçınılmış olur.

2-Bireysel çarpışma kaçınması

Bireysel çarpışma kaçınması sürü kaçınmasıyla oldukça büyük bir benzerlik gösterir. Aradaki tek fark kaçınma ağırlığının farklı olmasıdır. Bireysel kaçınmanın ağırlığının hesaplanması ise şekil 5'te gösterilmiştir:

$$d_{\min} = \begin{cases} d_1 & \text{if } (P_i - C_{\rho,i}) \leq \rho_{\exp} \\ d_2 & \text{if } (P_i - C_{\rho,i}) > \rho_{\exp} \end{cases}$$

Bireysel kaçınma mesafesi yönetici tarafından verilir sonra ise şekil 11'de gösterildiği gibi İHA etrafında sanal bir alan oluşturulur eğer 2 uçağın alanları kesişir ise kaçınma açıları şekil 2 'deki gibi hesaplanır ve gerekli manevralar yapılır.



Şekil 11 (Çarpışma Önleme bireysel ağırlık vektörleri)

Formasyon kontrolünden sonra incelenmesi gereken bir diğer konu İHA'ların konum değişiklikleri sırasında ortaya çıkan optimizasyon sorunudur. N adet İHA N adet konuma gitmelidir. Bu durumda birçok kombinasyonlar vardır ancak acaba bunlardan hangisi en verimlidir? Bu soruya cevap veren algoritma literatürde çok sık kullanılan "Macar Algoritmasıdır".

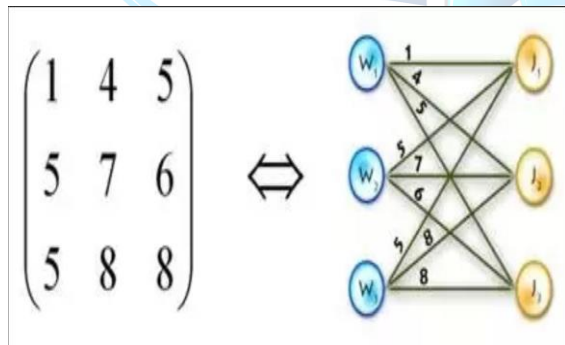
4.d – Macar Algoritması [15][16][17][18]

1- Her bir İHA'nın her bir noktaya olan uzaklığı hesaplanır ve bir matris oluşturulur.

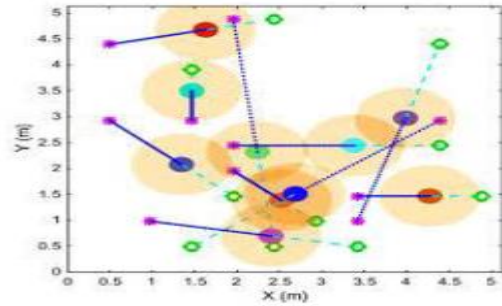
$$D = \sum_{i=1}^N \sum_{j=1}^N C_{ij} X_{ij} = \sqrt{(C_i - X_i)^2 + (C_j - X_j)^2 + X_z^2}$$

	Konum-1	Konum-2	Konum-3	Konum-N
İHA-1	C_{11}	C_{12}	C_{13}	C_{1N}
İHA-2	C_{21}	C_{22}	C_{23}	C_{2N}
İHA-3	C_{31}	C_{32}	C_{33}	C_{3N}
İHA-N	C_{N1}	C_{N2}	C_{N3}	C_{NN}

- 2- Her bir satırda bulunan minimum değer bulunur ve bulunduğu satırın bütün elemanlarından çıkarılır.
- 3- Her bir sütundaki bulunan minimum değer bulunur ve bulunduğu sütunun bütün elemanlarından çıkarılır.
- 4- Olabilecek en az sayıda çizgiyle 0 değerlerinin bulunduğu satırın veya sütunun üstü çizilir. Eğer çizilen çizgi sayısı satır veya sütun sayısına eşitse adım 6 ya geçilir değil ise 5. adımdan devam edilir.
- 5- Üstü çizili olmayan en küçük değer güncelleme değeri olarak seçilir ve üstü çizili olmayan bütün elemanlardan çıkarılır üstünden 2 tane çizgi geçen eleman ile toplanır. Üstünden çizgi geçmeyenler ise aynı şekilde kalırlar. Daha sonra 4. adıma tekrar dönülür.
- 6- Matrisimizin son halinde bulunan 0'lara atama yaparak minimum değeri bulabiliriz. Burada dikkat edilmesi gereken nokta her bir satır veya sütuna yalnızca 1 adet atama yapılabilir.



Şekil 12(Macar algoritması,kombinasyonlar)



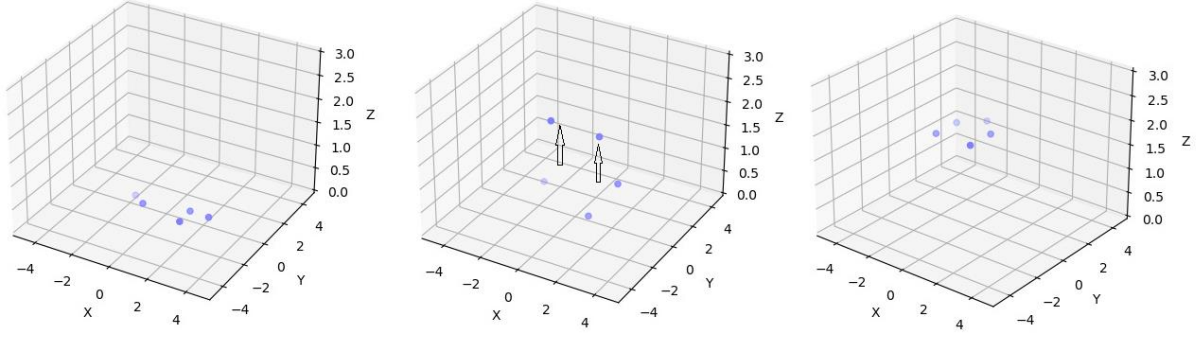
Şekil 13(Macar algoritması örnek uygulama)

Genel konseptleri anlattıktan sonra şimdi teker teker görevlerimizi inceleyecek olursak:

4.1 İHA Bireysel Kalkış ve Havada Formasyon Oluşturma

Bu görevde temel amaç rastgele veya düzenli bir şekilde zeminde konuşlandırılan İHA'ları sırasıyla kaldırarak istenilen irtifada istenilen formasyon durumuna getirmektir. Bu görevi yaparken karşılaşılan en büyük problem işgücü problemidir yani İHA'ların her birinin konumlarına gitmesi için gerekli olan enerjiyi minimum değerde tutmaktır. Bu problemin çözümü için 4.d'de açıkladığımız MACAR algoritmasını kullanmak görevi tamamlamak için bir çözüm oluşturmaktadır.

Şekil 14'te İHA'lar rasgele konumlardan kalkış yapıp istenen formasyona girmesi temsili olarak görselleştirilmiştir.

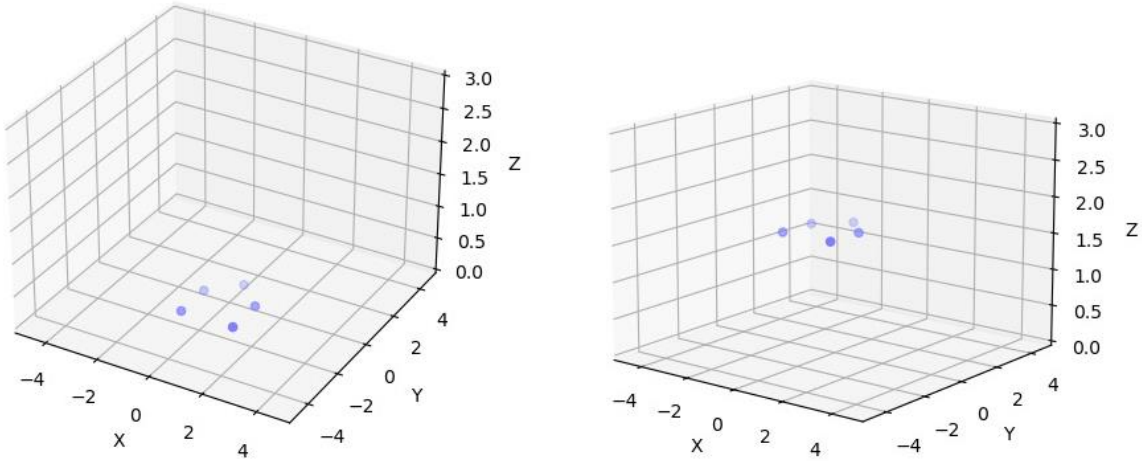


Şekil 14

4.2 Formasyon Halinde Kalkış

İHA'ların formasyon halinde kalkışa geçebilmesi için ilk komutla beraber bir formasyon kontrol algoritması ile hareket etmeleri gerekmektedir. Bu görev için 4.b-1' de de açıkladığımız sanal yapı yaklaşımını kullanmak görevi gerçekleştirmek adına gayet uygun olacaktır. Bu yaklaşımla her İHA belirlenen konumlara bir bütün gibi hareket edecek ve formasyon halinde kalkış gerçekleştirilmiş olacaktır. Sonrasında sanal bir yapı gibi davranan İHA'lar bu yapıyı koruyarak başlangıç konumlarına inmiş olacaktırlar.

Şekil 15'te İHA'ların formasyonunu koruyarak kalkış yapması gösterilmiştir.

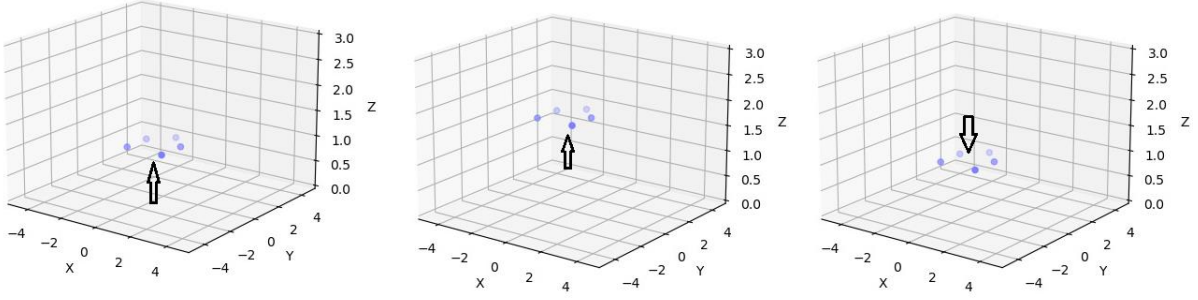


Şekil 15

4.3 Sürü Halinde Yükselme-Alçalma

Burada İHA'lar kalkış ve formasyon oluşturmaya yaptıktan sonra ek olarak formasyonu koruyarak yükselme-alçalma hareketi yapmalıdır. Bunu yaparken formasyon korunmalıdır. Bu durumda İHA'lar sanal yapı halini sürdürerek bu görevi gerçekleştirmelidirler. İHA'ların birbirleriyle uyumlu hareket etmesi gerektiği için tüm İHA'ların hızları aynı oranda değişmelidir. Yani tüm İHA'lar aynı ivmeye sahip olmalıdır.

Şekil 16'da belli bir irtifada bulunan İHA'ların formasyonu koruyarak yükselme-alçalma hareketi yapması gösterilmiştir.

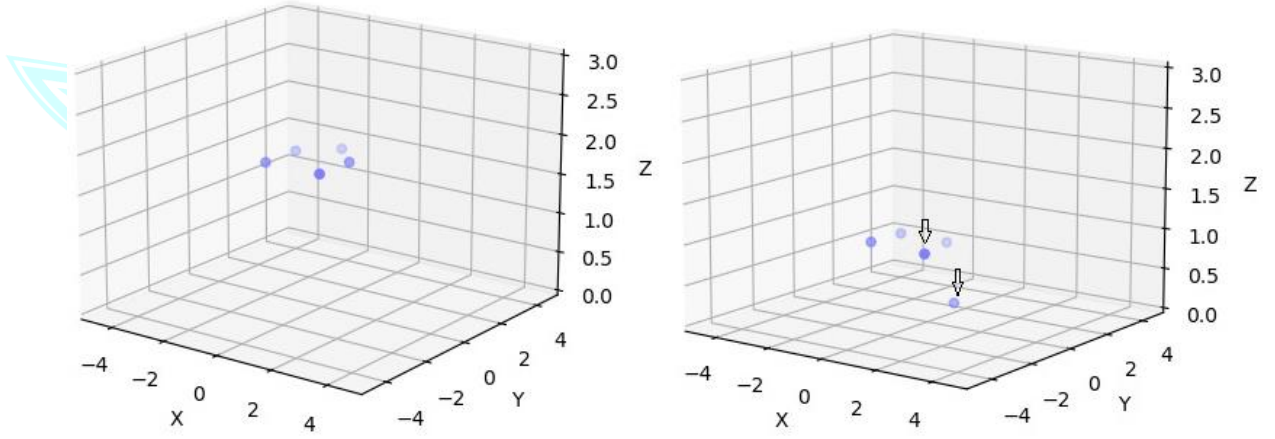


Şekil 16

4.4 Sürü Halinde Eş Zamanlı/Sıralı Otomatik İniş

Eş zamanlı/sıralı iniş için belli bir irtifada bulunan İHA'ların art arda eşit zaman aralıklarıyla iniş yapması gerekmektedir. İHA'lar sırayla iniş yaparken henüz iniş yapmamış olan İHA'lar formasyonunu korumaya devam etmelidir.

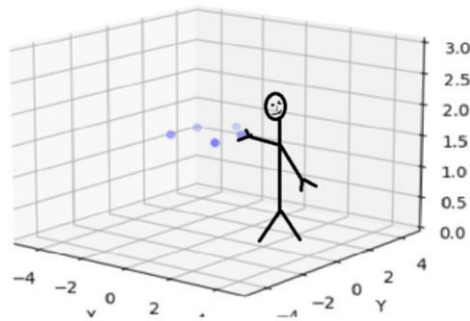
Şekil17'de İHA'ların eş zaman aralıkları ile sırayla iniş yapmaları gösterilmektedir.



Şekil 17

4.5 Sürüden Birey Çıkarma

Bu görevde formasyon halinde, belli bir irtifada uçuş yapan İHA'ların içinden birisi seçilip, görevli bir kişi tarafından el ile sürüden çıkartılacaktır. Burada uygulanacak olan yöntem şudur. Formasyon kontrol algoritması sürüden çıkarılan elemanı tespit edecek, bu yokluğa karşı diğer İHA'lar arasındaki geometrik bağlantıları bozmadan formasyon uçuşuna devam edecektir. Sürüden çıkarılan İHA'nın belirlenmesi ise kod üzerindeki birtakım kontrol mekanizmaları sayesinde sağlanacaktır.

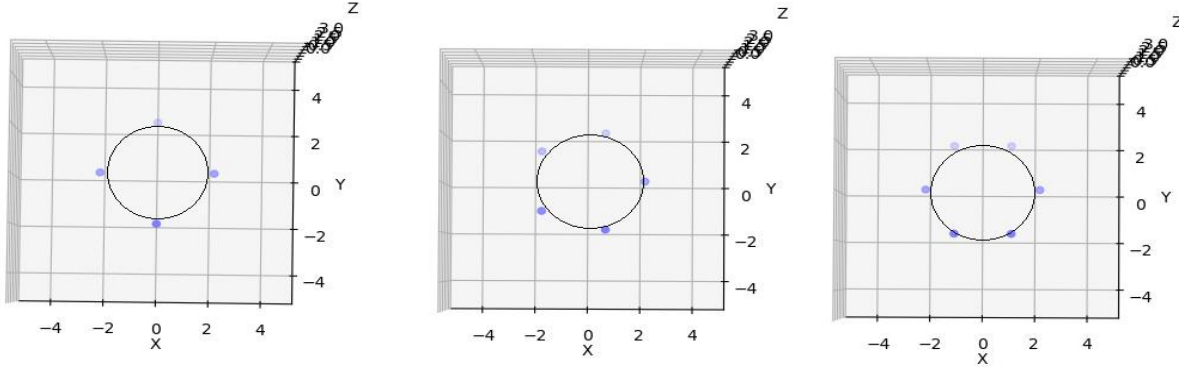


Şekil 18

4.6 Sürüye Yeni Birey Ekleme

Rasgele konumdaki İHA'lar kalkış yapıp çember formasyonuna girdikten sonra farklı bir yerde konumlanmış olan bir İHA sürüye dahil olur. İlk durumda İHA'lar belli bir açıyla çember formasyonunda bulunmaktaydı. Yeni İHA'nın katılmasıyla birlikte güncel İHA sayısına göre çember formasyonunu koruyan İHA'lar çembere orantılı bir şekilde dağılmak için aralarındaki açığı günceller. Böylece formasyon korunur ve düzgünlüğü sürdürür. Daha sonra yeni bir birey eklenmesi için de aynı adımlar tekrarlanır. İHA'lar arasındaki yeni mesafeler $2\pi / (\text{İHA sayısı})$ olacaktır. Bu işlemlerden sonra polar koordinatlar tekrardan x, y, z, koordinatlarına döndürülerek referans çerçevesi korunmuş olur.

Şekil 19'da ilk başta dört elemanla çember formasyonu oluşturulmuş ve birey eklenerek devam edilmiştir ve yine çember formasyonunun korunduğu gösterilmiştir.

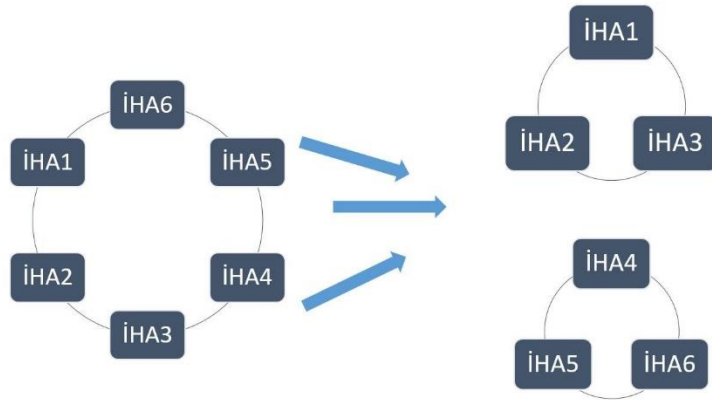


Şekil 19

4.7 Sürü Ayrılması

Sürü ayrılmasında önemli olan nokta grup oluşturacak olan İHA'ları seçmektir. Bu seçilimi yapmak için sürünün dışında bir eksen referans seçilir ve bu referansa olan uzaklığa göre seçimler yapılır. İşlemleri sırasıyla açıklayacak olursak;

1. Sürü dışında bir eksen referans olarak seçilir.
2. Sürünün kaç parçaya bölünmesi isteniyor ise sürüdeki birey sayısı yeni sürü sayısına bölünür.
2.1-Eğer bölüm işleminden kalan olur ise bu İHA'lar eksene olan uzaklıklarına göre sırasıyla gruplara eklenirler.
- 3- Seçilen referans eksenine olan uzaklıklarına göre yeni sürülerde kaçar İHA bulunacak ise o kadar İHA seçilir ve gruptan ayrılır. Eğer 2 İHA'nın uzaklığı aynı ise numarası küçük olan numarası küçük olan gruba eklenir.
4. Gruplara ayrılan İHA'lar birbirlerinden ayrılır.

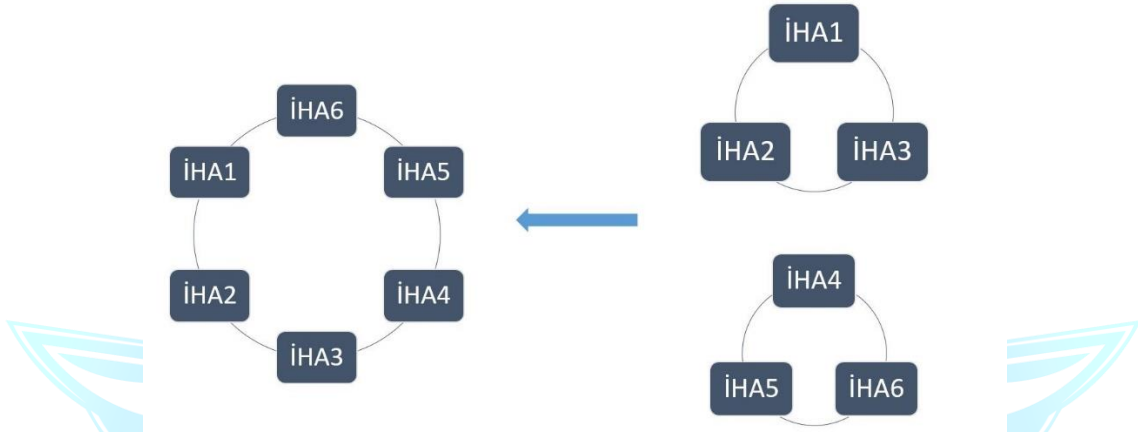


Şekil 20

4.8. Sürü Birleşmesi

Sürü birleşmesinde önemli olan nokta sürülerin birleşmek için eşleşecek olan İHA'larını seçmektir. Bu seçilimi yapmak için birbirlerine yakın olan sürüler bulunur ve bu sürülerin birbirlerine en yakın 4 İHA'sı seçilir. Daha sonra bu İHA'lar üstünden yeni bağlar kurularak birleşen sürüdeki İHA sayısına göre uygun olan formata geçilir. Adımları sırasıyla açıklayacak olursak;

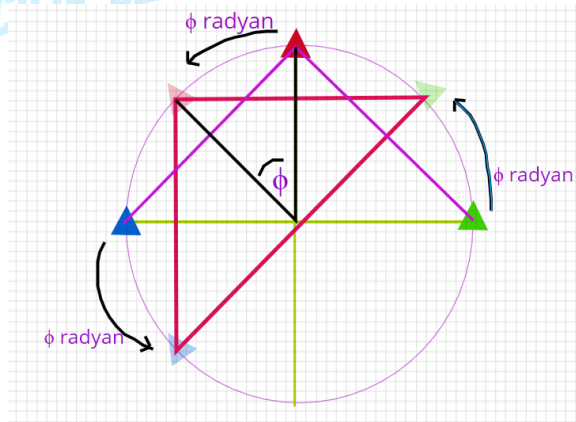
1. Sürülerin merkezleri hesaplanarak birbirlerine en yakın sürüler eşleştirilir.
2. Eşleşen sürülerden karşısındaki sürüye en yakın 2 tane yan yana olan İHA'yı bağlantı elemanı olarak seçer.
3. Sürü içerisindeki seçilen İHA'lar aralarındaki uzaklık bağıntısını koparır ve karşı sürüdeki seçilen İHA'lardan kendisine yakın olanı ile uzaklık bağıntısını kurar.
4. Son olarak yeni sürüdeki eleman sayısına göre yeni formasyon oluşturulur.



Şekil 21

4.9. Formasyon Koruma ile Sürü Halinde Yön Değiştirme Görevi

Sürü halinde yön değiştirme görevinde öncelikli hedefimiz İHA'ları bireysel olarak döndürmektir, bir diğer önemli nokta ise İHA'lar döndükten sonra rotasyonun bozulmamasıdır. Yani başka bir deyişle, sürünün rotasyonunu koruyarak dönmesi gerekir. Bunu tasarlamamızın bir yolu sanal yapıdır(virtual structure). Sanal yapı kullanmamızdaki amacımız, bütün sürüyü kapsayan bir yapıyı ele alıp formasyonunda buna bağlı olarak dönmesini sağlamaktır. Şekil 22'de görüldüğü gibi, sürüdeki İHA'ların üzerinden geçen sanal bir çember tanımladığımızı varsayalım. Sanal çember sayesinde formasyonumuzun temel yapısını biliyor olacağız ve aynı zamanda rotasyon görevini uygularken yapmak istediğimiz açısal işlemleri çok daha rahat kullanmış olacağız.



Şekil 22

Şekil 22'de görüldüğü gibi İHA'ların rotasyona uğraması isteniyor. Bunun için öncelikle gitmek istedikleri konum belirleniyor, bu belirlendikten sonra İHA'nın o anki konumu ve gitmek istediği konum arasındaki mesafe ile rotasyon açısını (φ) hesaplıyoruz. Elde ettiğimiz veri ile İHA'lar bireysel olarak φ radyanlık rotasyon yapacaklardır. Aynı zamanda sanal yapı olarak belirlediğimiz çember de φ radyanlık bir rotasyon yapıp formasyonumuzun korunmasını sağlayacaktır. Bu şekilde sürü halinde rotasyon görevi tamamlanacaktır.

4.10. Formasyon Değişimi [19]

Görev gereği sürü İHA sistemimizin istenilen formasyonlara sorunsuz bir şekilde geçebilmesi gerekmektedir. Bu görevi yerine getirebilmek için geliştirilmiş olan bir algoritmayı bu bölümde açıklayacağız. Açıkladığımız bu algoritmayı kaynak kodumuza uyarlayarak kullanacağız.

Bu algoritma ile en optimum şekilde formasyon değişikliği yapabilmemiz mümkün olacaktır. Optimizasyonu sağlamak için Macar algoritması kullanılmıştır. Daha önceki görevlerde de olduğu gibi Macar algoritması optimizasyon konusunda oldukça kullanışlıdır.

- Bu algoritma için bazı tanımlamalar:

$$\begin{aligned} p_i &= (p_{ix}, p_{iy}, p_{iz})^T, \quad i = 1, \dots, n \text{ (i'inci İHA'nın başlangıç konumu)} \\ s_i &= (s_{ix}, s_{iy}, s_{iz})^T, \quad i = 1, \dots, n \text{ (i'inci İHA'nın istenilen konumu)} \\ q_i &= (q_{ix}, q_{iy}, q_{iz})^T, \quad i = 1, \dots, n \text{ (i'inci İHA'nın hedeflenen konumu)} \\ d &= (d_x, d_y, d_z)^T, \quad (\text{öteleme vektörü}) \\ \alpha &: \text{İstenilen konumu en uygun orana getiren parametre} \end{aligned}$$

- Hedeflenen konuma gitmek için uygulanması gereken denklem:

$$q_i = \alpha s_i + d, \quad \alpha \in (0, \infty)$$

- Bu denklemdeki değişkenlerin optimizasyonu için oluşturulmuş maliyet denklemi:

$$C = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

- $\sum_{i=1}^n x_{ij} = 1$, ifadesi her İHA'nın bir konuma gidebileceğini belirtir. $\sum_{j=1}^n x_{ij} = 1$, ifadesi ise her konumda bir İHA'nın bulunabileceğini belirtir.

- Başlangıç noktasından hedeflenen noktaya gitmenin maliyet denklemi:

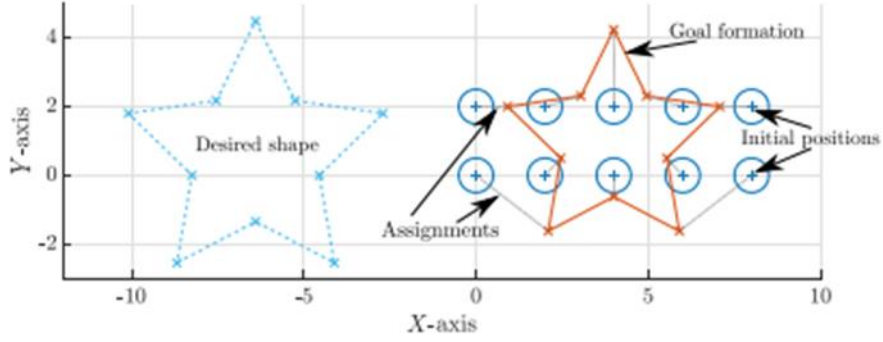
$$\begin{aligned} c_{ij}^{\alpha d} &= \|p_i - q_j\|_2^2 = (p_i - \alpha s_j - d)^T (p_i - \alpha s_j - d) \\ &= p_i^T p_i + \alpha^2 s_j^T s_j - 2\alpha p_i^T s_j + 2\alpha s_j^T d - 2p_i^T d + d^T d \end{aligned}$$

- En uygun maliyeti hesaplamak için kullanılacak olan genel denklem:

$$\begin{aligned} C_{ad}(\alpha, d, X) &= \sum_{i=1}^n \sum_{j=1}^n c_{ij}^{\alpha d} x_{ij} \\ &= d_p^2 + \alpha^2 d_s^2 + nd^T d + 2\alpha s^T d - 2p^T d + 2\alpha \sum_{i=1}^n \sum_{j=1}^n (-p_i^T s_j x_{ij}) \end{aligned}$$

- Bu formülde gerekli işlemler yapıldıktan sonra tüm parametrelerin türevi alınır ve sıfıra eşitlenir. Bu durumda $C_{ad}(\alpha^*, d^*, X^*)$ ifadesi en uygun maliyeti sağlar.

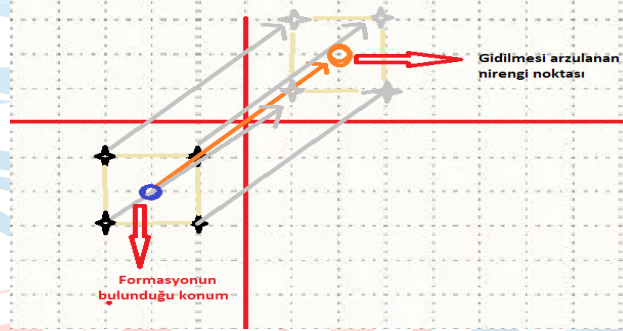
- Kullandığımız algoritma ile yapmayı hedeflediğimiz görev Şekil 23'de anlatılmıştır. Görselde görüldüğü üzere İHA'lar rasgele konumlarda daire şeklinde bulunmaktadır, mavi noktalar bizden istenen formasyonun noktalarıdır. Turuncu noktalar ise bizden istenen formasyona göre optimize edilmiş yeni formasyondur. Mavi noktalar ile turuncu noktalar arasındaki uzaklık da bize d öteleme vektörünü vermektedir. Formülde kullanılan α burada formasyon şeklini en uygun boyuta getirmek için oranlama yapmaktadır.



Şekil 23

4.11 Sürü Halinde Navigasyon

Sürü halinde navigasyon görevinde ilk olarak sürü sistemi formasyon halinde kalkış yapacaktır. Sürü bir süre formasyonunu korumalıdır, bu koruma işlemi 4.b de bahsettiğimiz formasyon kontrol yapıları yardımıyla sağlanacaktır. Daha sonrasında yarışma komitesi tarafından belirtilen ilk nirengi noktasına doğru navigasyon görevi başlayacaktır. Bu navigasyon hareketi için farklı yaklaşımlar mevcuttur. Bunlardan ilki formasyonun geometrik merkezini belirleyerek, bu geometrik merkezi yeni nirengi noktasına götürecek vektörü bulmaktır. Bu vektör bulunduğundan sonra tüm İHA'lar lokal koordinat çerçevelerini kullanarak bulunan vektör doğrultusunda ve büyüklüğünde öteleme yaparlar bu durumu gösteren örnek aşağıdaki gibidir:



Şekil 24

Şekil 24'te görüldüğü gibi, ilk olarak kare formasyonunun geometrik merkezi bulunmuştur. Daha sonrasında bu pozisyonlardan istenilen nirengi noktasına bir konum vektörü çekilmiştir. Sürü içindeki diğer tüm İHA'lar bu vektör yönünde ve büyüklüğünde, kendi konumlarını orijin kabul eden kendi koordinat sistemlerine göre hareket ederler. Şekil 24'ten görüldüğü gibi aslında tüm öteleme vektörleri eşit vektörler olacaktır.

Ancak bu yöntemin de bazı dezavantajları vardır örneğin bir diğer formasyon olan "V" formasyonu için sürü sistemi "V" 'nin sivri kısmı ile hareket vektörü aynı yöne bakacak şekilde hareket etmelidir. Diğer formasyonlar daha simetrik yapılara sahip olduğundan şimdilik bu problem sadece "V" formasyonu için ortaya çıkmaktadır lakin bunun çözümü İHA'ları kendi etraflarında rotasyon yaptırmaktır.

Bir diğer dezavantaj da yolların eğri olması durumunda ortaya çıkabilir bu durum için de çeşitli yaklaşımlar mevcuttur [20] ancak yarışmamız da böyle bir ihtiyaç ortaya çıkmamıştır.

5-Görev İsterlerinin Doğrulandığının Gösterilmesi

Görev No:	Görev İsterleri:	Görev Doğrulamaları:
1.	Sürü formasyonunun kontrolü görevlerin uyumlu bir şekilde yapılabilmesi için önemlidir. Bu yüzden formasyon kontrolü sağlanmalıdır.	Formasyon kontrolü 4.b kısmında detaylı bir şekilde anlatılmıştır. İlk planlamada Sanal yapı yaklaşımının kullanılması planlanmaktadır.
2.	Her İHA bireysel olarak 1 metre irtifaya yükselmeli ve daha sonra belirli bir formasyona girmelidir.	Burada İHA'ların konumları kullanılarak 1 metre irtifaya çıkmaları ve macar algoritması ile en uygun maliyetle formasyona girmeleri hedeflenmektedir.
3.	İHA'lar sürü halinde formasyonunu koruyarak kalkış ve iniş gerçekleştirmelidir.	Bölüm 4.2'de açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
4.	İHA'lar sürü halinde formasyonunu koruyarak 1 metre kalkış yaptıktan sonra yine formasyonunu koruyarak istenilen mesafe kadar daha yükselmeli ve sonra da alçalmalıdır.	Bölüm 4.3'da açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
5.	Sürü kalkışı yaptıktan sonra her bir İHA bireysel olarak eşit zaman aralıklarıyla ard arda iniş yapmalıdır.	Bölüm 4.4'de açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
6.	Görevli bir kişi tarafından herhangi bir İHA çıkarıldığında sürü formasyonunu korumalı ve belli bir süre sonra inişe geçmelidir.	Bölüm 4.5'da açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
7.	Sürü kalkış yapar ve çember formasyonuna girer daha sonra başka bir İHA gelir ve sürüye dahil olur bu durumda çemberin genişlemesi ve düzenin korunması beklenmektedir.	Bölüm 4.6'da açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
8.	Sürü formasyon halinde kalkış yaptıktan sonra iki farklı sürüye ayrılmalı ve iki sürü formasyonlarını koruyarak iniş yapmalıdır.	Bölüm 4.7'da açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
9.	İki sürü kalkış yapmalı ve formasyonlarını korumalıdır. Daha sonra sürüler birleşip tek bir sürü halinde formasyona girmelidir.	Bölüm 4.8'da açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
10.	Sürü istenilen formasyonu koruyarak saat yönünde veya saat yönünün tersi yönde istenilen açılarda belli bir açısal hızla dönüşler yapmalıdır.	Bölüm 4.9'da açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
11.	İstenilen bir formasyonda kalkış yapan sürü istenilen başka bir formasyona istenilen sürelerde geçiş yapmalıdır.	Bölüm 4.10'da açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.
12.	Sürü formasyonunu koruyarak istenilen noktalara navigasyon gerçekleştirmelidir.	Bölüm 4.11'da açıklanan algoritma ile bu görevin gerçekleştirilmesi hedeflenmektedir.

KAYNAKLAR:

Havelsan Eğitim videoları:

<https://www.youtube.com/playlist?list=PLvpJkeZbByrMBPQrU8qEcZBQgeNMyq0YA>

- 1- <https://crazyswarm.readthedocs.io/en/latest/>
- 2- <https://www.vive.com/us/accessory/base-station/>
- 3- <https://www.bitcraze.io/2016/08/crazyswarm/>
- 4- Coppola M, McGuire KN, De Wagter C, de Croon GCHE. A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints. *Front Robot AI*. 2020 Feb 25;7:18. doi: 10.3389/frobt.2020.00018. PMID: 33501187; PMCID: PMC7806031.
- 5- <http://www.chrobotics.com/library/understanding-quaternions>
- 6- Jawad Naveed Yasin, Mohammad-Hashem Haghbayan, Jukka Heikkone, Hannu Tenhunen, Juha Plosila, "Formation Maintenance and Collision Avoidance in a Swarm of Drones", September 25–27, 2019, Amsterdam, Netherlands
- 7- Dingjiang Zhou, Mac Schwager, "Virtual Rigid Bodies for coordinated agile maneuvering of teams of micro aerial vehicles", Article in *Proceedings - IEEE International Conference on Robotics and Automation* · June 2015
- 8- Zijian Wang, Dingjiang Zhou, Mac Schwager, "Agile Coordination and Assistive Collision Avoidance for Quadrotor Swarms Using Virtual Structures", Article in *IEEE Transactions on Robotics* · August 2018
- 9- A. Askari , M. Mortazavi , H. A. Talebi, "UAV Formation Control via the Virtual Structure Approach", *Journal of Aerospace Engineering / Volume 28 Issue 1 - January 2015*
- 10- K. CHOUTRI, M. LAGHA, L. DALA, M. LIPATOV, "Quadrotors UAVs Swarming Control Under Leader-Followers Formation", 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)
- 11- D. Xu, X. Zhang, Z. Zhu, C. Chen and P. Yang, "Behavior-Based Formation Control of Swarm Robots", *Mathematical Problems in Engineering*, 2014.
- 12- Sangita Roy, Samir Biswas, Sheli Sinha Chaudhuri, "Nature-Inspired Swarm Intelligence and Its Applications", *IJMECS*, vol.6, no.12, pp.55-65, 2014. DOI: 10.5815/ijmecs.2014.12.08
- 13- Han, SC., Bang, H. & Yoo, CS. Proportional navigation-based collision avoidance for UAVs. *Int. J. Control Autom. Syst.* **7**, 553–565 (2009). <https://doi.org/10.1007/s12555-009-0407-1>
- 14- Sharma, Rajnikant & Ghose, Debasish. (2009). Collision avoidance between UAV clusters using swarm intelligence. *International Journal of Systems Science*. 40. 521-538. 10.1080/00207720902750003.
- 15- <https://www.quora.com/What-is-the-Hungarian-Algorithm>
- 16- Amponsah, Samuel Kwame & Otoo, Dominic & Salhi, Said & Quayson, Ebenezer. (2016). Proposed Heuristic Method for Solving Assignment Problems. *American Journal of Operations Research*. 06. 436-441. 10.4236/ajor.2016.66040.
- 17- Peng, Hu & Rong, Li & Liang-lin, Cao & Li-xian, Li. (2011). Multiple Swarms Multi-Objective Particle Swarm Optimization Based on Decomposition. *Procedia Engineering*. 15. 3371-3375. 10.1016/j.proeng.2011.08.632.
- 18- <https://www.youtube.com/watch?v=LY9xO7PUJvY&t=560s>
- 19- S. Agarwal and S. Akella , "Simultaneous Optimization of Assignments and Goal Formations for Multiple Robots", 2018 IEEE International Conference on Robotics and Automation (ICRA), May 2018.
- 20- Saska, Martin & Hert, Daniel & Báča, Tomáš & Krátký, Vít & Nascimento, Tiago. (2020). Formation control of unmanned micro aerial vehicles for straitened environments. *Autonomous Robots*. 44. 10.1007/s10514-020-09913-0.