DeepSORVF Sistem Dokümantasyonu

# DeepSORVF Kod İşleyiş Dokümantasyonu

Bu doküman sadece kodun çalışma akışını ve fonksiyonların NEDEN var olduklarını açıklar. Kurulum, parametre listesi gibi bölümler (8. madde ve sonrası) çıkarılmıştır. Amaç: Kodu ilk gören birinin dosyalar arasında kaybolmadan mantığı takip edebilmesi.

## 1. Amaç

Deniz üzerindeki gemileri \*\*görsel tespit (YOLOX + DeepSORT)\*\* ve \*\*AIS (Automatic Identification System)\*\* verilerini zamansal ve mekânsal olarak hizalayıp eşleştirerek tekil gerçek dünya kimlikleriyle izlemek. Fusion süreci; iki farklı kaynaktan gelen (kamera + AIS) trajeleri aynı fiziksel gemiye ait olup olmadıklarına göre birleştirir.

## 2. En Üst Düzey Döngü Mantığı

`main.py` içindeki ana while döngüsü her video karesinde şu sırayı takip eder:

1. Frame oku

2. Mantıksal zaman & timestamp güncelle (`update\_time`)

3. Yeni saniyeye girildiyse AIS işleme (`AIS.process`)

4. Görsel pipeline (tespit + takip) (`VIS.feedCap`)

5. Yeni saniyeye girildiyse fusion (`FUS.fusion`)

6. Yeni saniyede sonuç CSV yazımı (`gen\_result`)

7. Çizim & overlay (`DRAW.draw\_traj`)

8. Video kaydı / gösterim

Temel prensip: AIS ve fusion yalnızca saniye sınırında (timestamp milisaniyesi < t eşiği) çalıştırılır, görsel tespit ise her karede yapılabilir.

## 3. Fonksiyonel Ayrıntılar (NEDEN / NASIL)

### 3.1 Giriş & Zaman Yönetimi (`utils/file\_read.py`)

- `read\_all(data\_path, result\_path)`: Video dosyasını, AIS klasörünü, çıktı dizinlerini ve kamera parametre dosyasını bulur. Neden: Tüm pipeline tek merkezli yapılandırma bilgisine ihtiyaç duyar.

- `ais\_initial(ais\_path)`: AIS CSV dosyalarının listesini ve ilk zaman etiketini çıkarır. Neden: Saniye bazlı erişimde dizin taramasını her döngüde tekrar etmemek.

- `update\_time(Time, t)`: Frame sayacından milisaniye bazında ilerleyerek yeni bir tam saniyeye geçtiğimiz anı belirler. Neden: AIS ve fusion işlemlerinin yalnızca saniye başında tetiklenebilmesi.

- `time2stamp(hhmmss)`: İnsan okunur zaman → milisaniye epoch. Neden: Farklı kaynaklardan gelen zaman formatlarını normalize etmek.

### 3.2 AIS İşleme Zinciri (`utils/AIS\_utils.py`)

Ana giriş noktası: `AISPRO.process(camera\_para, timestamp, Time\_name)`

Bu fonksiyon yalnızca yeni saniye tetik koşulunda çalışır (fazladan hesap yükünü engeller).

Adımlar:

1. `initialization()`: Geçici listeleri temizler. Neden: Saniyelik state kalıntılarının yeni okunan veriyle karışmasını önlemek.

2. `ais\_pro(...)`: İç içe iş akışı.

- `read\_ais(Time\_name)`: O saniyeye ait CSV var mı? Varsa satırları okuyup parse eder. Neden: Gerçek zamanlı akışın dosya tabanlı simülasyonu.

- `data\_coarse\_process(AIS\_read, AIS\_las, camera\_para, max\_dis)`: Hatalı / aşırı uzak / hız mantıksız kayıtlar elenir. Neden: Fusion maliyet matrisinin kirlenmesini ve yanlış pozitifleri azaltmak.

- `data\_pred(AIS\_cur, AIS\_read, AIS\_las, timestamp)`: (a) TIME\_OFFSET düzeltmesi (örn. +5h fark) (b) Eksik saniyeleri son iki geçerli noktanın hızı ve kursuyla ileri projekte eder (`data\_pre`). Neden: Kamera ve AIS zaman drift’i & paket kayıplarına karşı sürekli trajekte tutarlılık.

- `data\_tran(...)` → `transform()` + `visual\_transform()`: Coğrafi (lat/lon) → düzlemsel piksel koordinatı. Bearing + mesafe + kamera parametresi kullanır. Neden: Görsel uzay ile sayısal navigasyon verisini aynı koordinat sistemine getirmek.

Çıktılar: `AIS\_cur` (ham senkronize edilmiş), `AIS\_vis` (piksel lokasyonlu).

Kritik Nokta: Hız 0 iken yön bilgisi (course) gürültülü olabilir; filtre bunu dikkate alır veya prediction kısmı durur.

### 3.3 Görsel Pipeline (`utils/VIS\_utils.py`)

Giriş noktası: `VISPRO.feedCap(frame, timestamp, AIS\_vis, bin\_inf)`

Adımlar:

1. `detection()`: YOLOX modeli çalışır, gemi aday kutuları çıkar. Neden: Ham görsel akıştan potansiyel hedefleri izole etmek.

2. `anti\_occ(...)`: Örtüşen kutuları ve olası kaybolan hedefleri analiz eder (tamamlanmamış logic). Neden: Geçici kayıplarda kimlik sürekliliğini koruma hedefi.

3. `track(...)`: DeepSORT; her tespit için appearance + motion (Kalman) bileşimi ile ID ataması yapar. Neden: Kutu dizisini zamansal kimlik dizisine dönüştürmek.

4. `update\_tra(...)`: Yeni saniyede gerçekleşir; aynı ID'nin o saniyedeki çoklu örnekleri ortalanır; `motion\_features\_extraction` ile hız / yön çıkarılır. Neden: Fusion için daha kararlı saniyelik temsil üretmek.

Çıktılar: `Vis\_cur` (o karedeki ID'ler), `Vis\_tra` (son ~2 dakikalık geçmiş penceresi). Bu geçmiş pencere DTW hesaplarında dokusuz (sparse) yerine yoğun (dense) örnek sunar.

### 3.4 Fusion (`utils/FUS\_utils.py`)

Giriş: `FUSPRO.fusion(AIS\_vis, AIS\_cur, Vis\_tra, Vis\_cur, timestamp)`

Alt Akış:

1. `traj\_group(... 'AIS')` ve `traj\_group(... 'VIS')`: Her kimlik için zaman sıralı (x,y) noktalar listesini derler. Neden: DTW gibi sekans tabanlı karşılaştırmalar için ham nokta akışını gruplamak.

2. `traj\_match(...)`: Orkestrasyon.

- `initialization()`: Önceki saniyeden gelen eşleşme geçmişini (`mat\_las`) alır; uzun vadeli istikrar için gerekli.

- `cal\_similarity()`: Her VIS–AIS çifti için:

\* Son konum mesafesi (gating) – çok uzaksa yüksek maliyet.

\* `angle()` ile yön farkı – aşırı sapmalar cezalandırılır.

\* `DTW\_fast()`: Sekansları önce downsample eder (`\_\_reduce\_by\_half`), ardından fastdtw ile yol maliyeti bulur, açı penalti uygulayarak nihai skor üretir.

\* Önceden bağlanmış çiftlere negatif offset (süreklilik ödülü).

- Hungarian (`linear\_assignment`) ile toplam maliyet minimizasyonu.

- `data\_filter(...)`: İkincil sert eşikler (mesafe < max\_dis, açı < limit) – yanlış pozitif süzgeci.

- `save\_data(...)`: Eşleşme sayaçlarını (match count) günceller, geçici kayıplara tolerans tanır (fog tolerance). Neden: Kısa süreli occlusion sırasında ilişkiyi koparmamak.

Çıktılar: `mat\_list` (saniyelik fusion kayıtları), `bin\_cur` (aktif bağlar tablosu).

### 3.5 Çizim (`utils/draw.py`)

- `draw\_traj(...)`: Her görsel ID'yi işler, eğer `bin\_cur` ile bir AIS eşleşmesi varsa kutuyu ve metni sarı renkte; yoksa kırmızı renkte çizer. Neden: Operatöre durumu hızlı görsel geri bildirimle iletmek.

- Yardımcılar `draw\_box`, `draw\_line`: Görsel bütünlük ve okunabilirlik için basit overlay araçları.

### 3.6 Sonuç Kaydı (`utils/gen\_result.py`)

- `gen\_result(...)`: Üç CSV setini saniye bazında append eder: detection (ham kare tespitleri), tracking (ID'li kutular), fusion (AIS ile zenginleştirilmiş ID'ler). Neden: Post-analiz, metrik hesaplama ve hata ayıklama.

## 4. Önemli Veri Yapıları

- AIS satırı: `[mmsi, lon, lat, speed, course, heading, type, timestamp, x, y]`

- Visual track satırı: `[ID, x1, y1, x2, y2, x, y, speed, timestamp]`

- Fusion çıktı satırı: `[ID, mmsi, lon, lat, speed, course, heading, type, x1, y1, w, h, timestamp]`

- Binding (`bin\_cur`): `[ID, mmsi, timestamp, match]`

## 5. Fusion Karar Kuralları (Kısa)

1. Geometrik yakınlık ve yön uyumu ilk kapı.

2. Zaman serisi benzerliği DTW ile ölçülür (ölçek & açı cezalı).

3. Hungarian ile global optimum eşleşme.

4. Son sert kapı + istikrar puanı (match count) → nihai bağ.

5. Kısa kayıplar toleransla tutulur (fog tolerance).

## 6. Ayrıntılı Akış Diyagramı

Metin tabanlı mermaid tanımı yanında, bu deponun ilerleyen aşamasında oluşturulacak `flowchart.png` dosyası aynı adımları görsel olarak gösterecektir.

Mermaid Kaynağı:

flowchart TD

A[Start] --> B[read\_all & ais\_initial]

B --> C[Open Video]

C --> D{Read Frame}

D -->|None| Z[End]

D --> E[update\_time]

E --> F{New Second?}

F -->|Yes| G[AIS.process]

F -->|No| H[VIS.feedCap]

G --> H[VIS.feedCap]

H --> I{New Second?}

I -->|Yes| J[FUS.fusion]

I -->|No| K[draw\_traj]

J --> L[gen\_result]

L --> K[draw\_traj]

K --> M[Write/Show]

M --> D

ASCII Özet:

Frame -> update\_time -> (sec?) AIS.process -> VIS.feedCap -> (sec?) FUS.fusion -> (sec?) gen\_result -> draw\_traj -> output -> next frame

## 7. Neden DTW + Hungarian?

- DTW: Farklı hız / ufak zaman kaymaları olsa bile yol şeklinin (trajectory pattern) benzerliğini ölçer.

- Hungarian: Tüm VIS ↔ AIS aday çiftleri aynı anda optimize ederek lokal hatalı seçimi engeller.

- Birlikte: Gürültülü tek frame uzaklıklarına dayanmak yerine sekans bağlamı + global optimizasyon.