

# 왜 RAFT 대신 Farneback을 사용하였는가?

## 1. 실시간 추론 성능 (Real-time Latency)

- **RAFT의 문제:** RAFT는 최신 딥러닝 기반 Flow 알고리즘으로 정확도는 매우 높으나, 수많은 Iterative Update를 거쳐야 하므로 연산량이 매우 큼. NVIDIA RTX 3090 급에서도 10~20 FPS 내외의 속도를 보여, **\*\*YOLO의 30 FPS 이상의 실시간 탐지 성능을 저해하는 병목(Bottleneck)\*\***이 됩니다.
- **Farneback의 선택:** Polynomial Expansion(다항식 전개)을 기반으로 한 알고리즘으로, CPU/GPU 모두에서 고도로 최적화되어 있습니다. 전처리에 소요되는 시간을 최소화하여 **전체 시스템이 30 FPS 이상의 실시간성을 유지할 수 있게** 하였습니다.

## 2. 하드웨어 독립성 및 오버헤드 (Resource Overhead)

- **VRAM 관리:** RAFT를 사용하면 YOLO 모델 외에 또 다른 거대한 딥러닝 모델을 VRAM에 상주시켜야 합니다. 이는 선박 내 탑재되는 임베디드 장비(Jetson 등)에서 메모리 부족(OOM) 문제를 일으킬 수 있습니다.
- **Farneback의 장점:** 별도의 모델 가중치 로딩 없이 OpenCV 라이브러리 레벨에서 구동되므로, 시스템 자원을 YOLO 백본 연산에 더 집중할 수 있는 환경을 제공합니다.

## 3. 노이즈 강건성 (Noise Robustness)

- **도메인 적응성:** RAFT와 같은 학습 기반 모델은 특정 데이터셋(예: FlyingChairs)에 최적화되어 있어, 우리가 시뮬레이션한 특수한 '**폭풍우 노이즈(Synthetic Storm Noise)**' 환경에서 예측하지 못한 왜곡을 보일 수 있습니다.
- **Farneback의 물리적 계산:** 고전적인 픽셀 매칭 방식은 노이즈가 심할 때 해상도를 낮추는 '**Pyramid downsampling**' 기법을 통해, 지역적인 노이즈를 무시하고 객체의 거대한 전역적 움직임을 포착하는 데 더 안정적인(Stable) 결과를 보여주었습니다.

## 4. 아키텍처 단순화 (Architectural Simplicity)

- 본 프로젝트의 목적은 **\*\*"Optical Flow 정보가 YOLO의 인식률을 높여주는가"\*\***를 증명하는 것입니다. 따라서 Flow 추출 알고리즘은 최대한 가볍고 투명한(Deterministic) 방식을 사용하여, 모델 성능 향상의 원인이 Flow 정보 그 자체에 있음을 명확히 하였습니다.