컴퓨터비전(AI응용)

추적 알고리즘

00

수업목표

01. MeanShift & CamShift

MeanShift와 CamShift의 기본 개념과 작동 원리를 학습하고, 히스토그램 기반 추적 방식을 이해합니다.

02. Lucas-Kanade Optical Flow

Lucas-Kanade 알고리즘의 수학적 가정과 특징점 추적 원리를 이해하고 실제 코드 활용법을 익힙니다.

03. 기법 비교 및 시나리오

각 추적 기법들의 장단점을 비교 분석하여, 상황별 최적의 알고리즘을 선택하고 활용 시나리오를 설계합니다.

04. 특징자 기반 객체 추적

특징자 검출기를 활용해 의미 있는 특징점을 추출하고, 이를 Optical Flow와 결합하여 강건한 객체 추적을 구현합니다.

01

템플릿 매칭

이미지 내 특정 패턴 탐색 기법

기본 개념

이미지에서 특정 패턴이나 객체를 찾기 위해 사용하는 기법입니다.
주어진 작은 '템플릿 이미지'를 원본 이미지 전체 영역에서
슬라이딩하며 픽셀 값을 비교하여 가장 일치하는 부분을 탐색합니다.

핵심 함수: cv2.matchTemplate

원본 이미지와 템플릿 이미지 간의 매칭 정도를 분석하여 유사도 맵(결과 행렬)을 반환합니다.

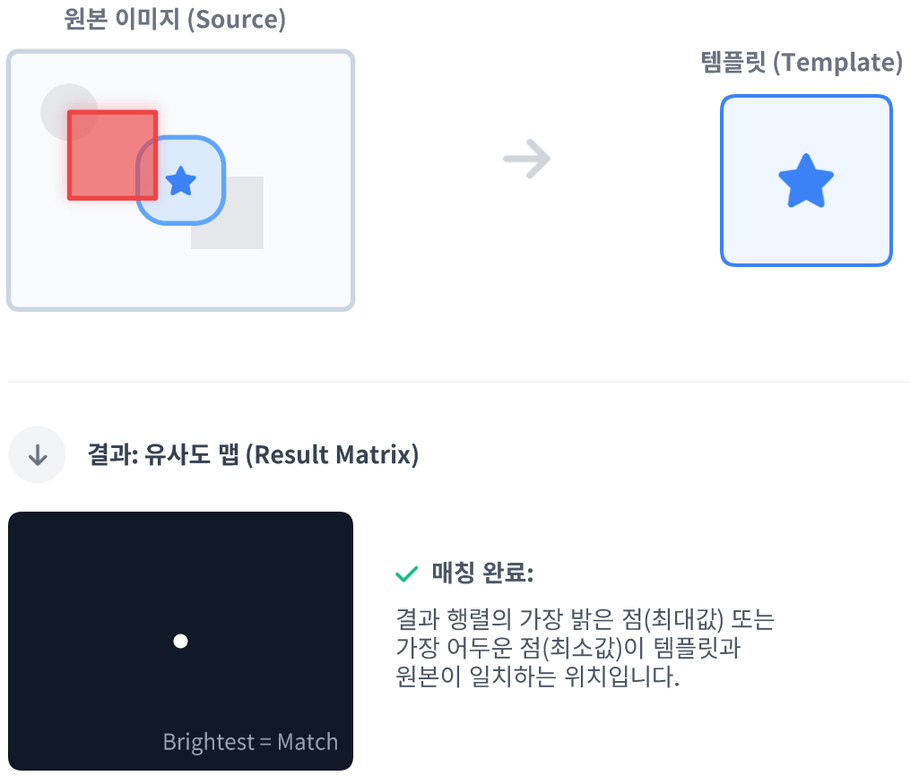
```
res = cv2.matchTemplate(image, template, method)
```

결과 분석: cv2.minMaxLoc

결과 행렬에서 최소값과 최대값, 그리고 해당 위치(좌표)를 찾아 매칭 지점을 결정합니다.

```
minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(res)
```

Visual Concept



01 템플릿 매칭

템플릿 매칭

- 이미지에서 특정 패턴이나 객체를 찾기 위해 사용하는 기법
- 주어진 템플릿 이미지를 원본 이미지에서 비교하며, 템플릿과 일치하는 부분을 탐색
- **cv2.matchTemplate**
  - 원본 이미지와 템플릿 이미지 간의 매칭 정도를 분석하여 반환(행렬)
  - 매칭 방법
    - cv2.minMaxLoc 함수로 결과 행렬에서 최소값과 최대값, 해당 좌표를 찾을 수 있음

	매칭 옵션	해석
1	cv2.TM_CCOEFF	상관계수
2	cv2.TM_CCOEFF_NORMED (추천)	정규화 된 상관계수
3	cv2.TM_CCORR	상관관계
4	cv2.TM_CCORR_NORMED	정규화 된 상관관계
5	cv2.TM_SQDIFF	제곱 차이의 합
6	cv2.TM_SQDIFF_NORMED	정규화 된 제곱 차이의 합

01

템플릿 매칭

매칭 방법 비교 Matching Methods

Tip: 보통 TM_CCOEFF_NORMED가 가장 좋은 결과를 보여주며 널리 사용됩니다.

매칭 방법에 따른 해석의 차이

매칭 방법에 따라 결과값(유사도)의 해석이 달라집니다. **상관관계 기반**은 값이 클수록, **오차 기반**은 값이 작을수록 유사함을 의미합니다.

No.	Method Flag	설명 (Description)	특징 및 추천
1	TM_CCOEFF	**상관계수 (Correlation Coefficient)** 평균을 뺀 영상에서의 상관계수를 계산	밝기 변화에 민감함
2	TM_CCOEFF_NORMED 추천	**정규화된 상관계수** 0~1 사이 값으로 정규화하여 비교 용이	**가장 정확함** 빛의 변화에 강함
3	TM_CCORR	**상관관계 (Cross Correlation)** 단순 픽셀 곱의 합	빠르지만 정확도 낮음
4	TM_CCORR_NORMED	**정규화된 상관관계** 상관관계를 정규화하여 안정성 확보	CCORR보다 안정적
5	TM_SQDIFF	**제곱 차이의 합 (Squared Difference)** 픽셀 값 차이의 제곱 합	완전 일치 시 0
6	TM_SQDIFF_NORMED추천	**정규화된 제곱 차이** 제곱 차이를 0~1 사이로 정규화	SQDIFF보다 비교 용이

02

특징자 추출

특징자란?

이미지 내에서 강하게 구분되는 독특한 부분입니다.
주로 **모서리(Corner)**, **점(Blob)**, **특정 패턴** 등
주변과 뚜렷하게 구별되는 지점을 의미합니다.

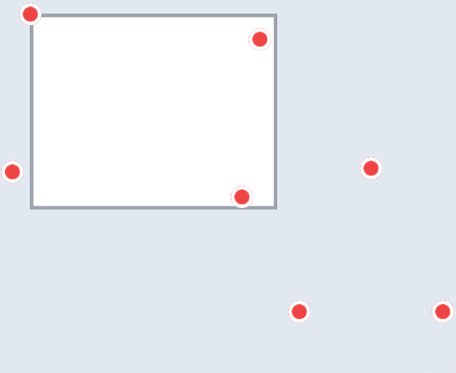
특징자 검출 및 목적

- **검출 (Feature Detection):**
  영상 전체에서 의미 있는 특징점을 자동으로 찾아내는 알고리즘 과정입니다.
- **목적:** 조명 변화, 회전, 크기 변화에도 잃어버리지 않고 안정적으로 찾을 수 있는 **기준점(Keypoint)**을 만들기 위함입니다.

주요 활용 분야

추적	정합	인식
Tracking	Matching	Recognition

특징점 검출 예시



특징자 처리 프로세스

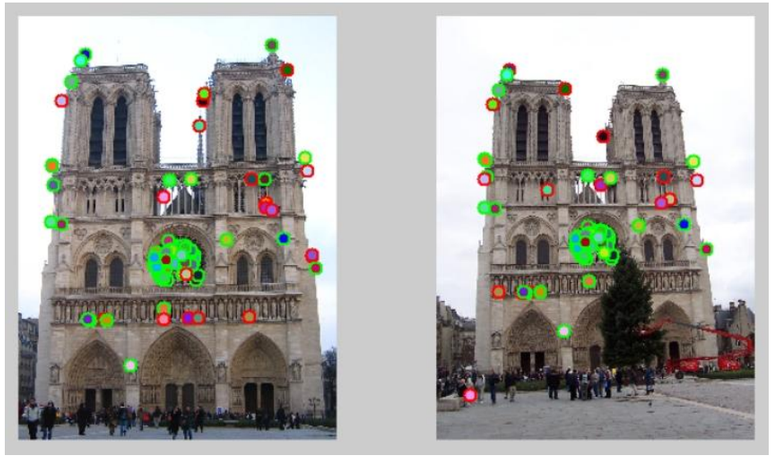
입력 영상 → 특징점 검출 → 기술자 생성

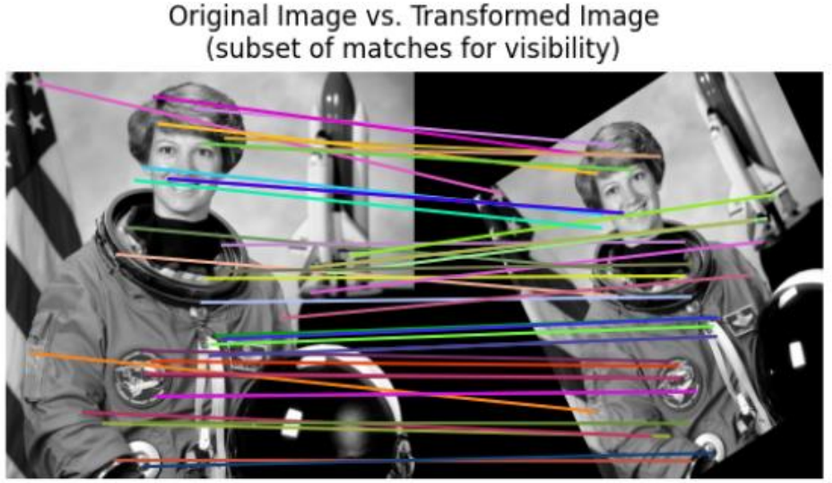
02

특징자 추출

특징자(Descriptor)

- 이미지 내에서 강하게 구분되는 부분(예: 모서리, 점, 특정 패턴)
- 특징자 검출(Feature Detection) : 특징자를 자동으로 찾아내는 알고리즘
- 목적 : 영상 내에서 안정적으로 찾을 수 있는 기준점을 만들기 위함
- 활용 : 추적(Tracking), 정합(Matching), 인식(Recognition) 등에 활용





02

특징자 매칭

특징자 매칭 vs 템플릿 매칭

템플릿 매칭

기존 방식

접근 방식

픽셀 기반 (Pixel-based)

이미지의 픽셀 값을 그대로 비교하여 유사도 측정

주요 한계점

환경 변화에 민감
빛의 밝기가 변하면 픽셀 값이 달라져 매칭 실패

기하학적 변형에 취약
회전(Rotation), 크기(Scale) 변화 시 템플릿 불일치

회전된 객체는 사각형 템플릿과 맞지 않음

특징자 매칭

권장 방식

접근 방식

특징점 + 디스크립터 (Keypoint & Descriptor)

이미지의 지역적(Local) 특징 정보를 벡터화하여 비교

처리 절차 (Workflow)

특징점 추출
Keypoint

벡터화
Descriptor

매칭
Matching

강인함 (Robustness)
회전, 크기 변화, 밝기 변화가 있어도 **고유한 특징 벡터**는 유지되므로 매칭 성공률이 높음

02

특징자 매칭

- **템플릿 매칭**의 경우, '**픽셀 기반**' 매칭이기 때문에 빛의 밝기, 회전, 색상 등에 민감
- **특징자(Descriptor) 매칭**은 이러한 약점을 보완하기 위해 **이미지의 특징점을 추출**
  - 특징점의 디스크립터를 생성하여 지역적인 특징을 비교하려 함
  - 의미 있는 특징자를 추출하여, 패치를 추출하고 벡터화 하는 단계를 거침

	매칭 옵션	년도	특징
1	SIFT	1999	특징점 주변의 그래디언트 분포를 히스토그램으로 표현
2	HOG	2005	지역적 그래디언트의 방향 정보를 히스토그램으로 표현
3	SURF	2006	SIFT 방법의 계산 효율 증가 + Haar 웨이블릿 필터와 적분 이미지 사용
4	DAISY	2009	그래디언트의 방향과 세기 계산 + 방사형 패턴의 히스토그램 표현
5	BRIEF	2010	특징점 주변의 픽셀 값을 비교
6	ORB	2011	BRIEF의 단점 개선(회전, 스케일 변화에 강건)
7	FREAK	2012	망막 구조를 모방한 특징점 표현
8	AKAZE	2013	비선형 스케일 공간으로 특징점을 검출
9	LATCH	2015	픽셀 패치의 패턴 학습
10	딥러닝 기반 디스크립터		LOFTR(transformer)

03

MeanShift 알고리즘

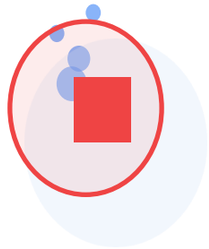
확률 분포 기반의 객체 추적 기법

핵심 아이디어

데이터 집합의 **밀도 분포(Density Distribution)**를 이용하여 가장 조밀한 부분(Mode)을 찾아가는 알고리. 영상 처리에서는 주로 **색상 히스토그램**을 기반으로 객체의 중심을 반복적으로 추적

#밀도추정 #히스토그램 #반복적수렴

Concept Visual



무게 중심으로 이동

실행 순서 (4단계)

1

색상 모델 생성

추적할 객체의 영역(ROI)을 설정하고
색상 히스토그램을 계산하여
색상 분포 모델을 만듭니다.

2

유사 영역 탐색

전체 영상에서 모델과
유사한 색상 분포를 가진 영역을
역투영(Back Projection)하여
확률 맵을 생성합니다.

3

중심 이동 (Shift)

현재 윈도우 내의 픽셀값들의
무게 중심(Mass Center)을 계산하고,
윈도우의 중심을 그곳으로 이동시킵니다.

4

반복 수행

윈도우의 중심 이동 거리가
매우 작아지거나(수렴),
지정된 횟수에 도달할 때까지
3번 과정을 반복합니다.

03

MeanShift 알고리즘

- **실행 순서**

1. 객체의 색상 분포를 모델로 만든다
2. 타겟 중 가장 비슷한 영역을 찾아 윈도우를 이동함
3. 윈도우 이동 시, 픽셀들의 중심(Mean)을 계산하여 업데이트
4. 딱 맞는 영역을 찾아낼 때 까지 이를 반복함

04

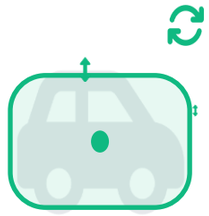
CamShift 알고리즘

크기와 회전에 적응하는 연속적 MeanShift

MeanShift의 개선 및 보완

- **문제점:** MeanShift는 윈도우(탐색 영역)의 크기와 방향이 고정되어 있어, 객체가 카메라에서 멀어지거나(크기 축소), 회전할 경우 추적에 실패할 수 있습니다.

- **해결책 (CamShift):** Continuously Adaptive Mean Shift의 약자
  매 프레임마다 윈도우의 **크기(Scale)**와 **방향(Orientation)**을 스스로 갱신하여 변화하는 객체에 적응



크기 & 각도 자동 조절

실행 순서 (5단계)

1

색상 모델 생성

HSV 색공간에서
Hue(색조) 히스토그램을 사용하여
조명 변화에 강한 모델을 생성합니다.

2

확률 맵 생성

백프로젝션을 수행하여
영상의 각 픽셀이 객체 모델과
얼마나 유사한지 확률값으로 변환

3

MeanShift 수행

확률 맵 위에서 MeanShift 알고리즘
을 적용하여 밀도가 가장 높은
최적의 중심 위치로 이동합니다.

4

윈도우 업데이트

영역 내 픽셀 분포(2차 모멘트)를
계산하여 윈도우의 **크기(너비,높이)**와
회전 각도를 갱신합니다.

5

반복 수행

다음 프레임에서 갱신된
윈도우 정보를 초기값으로 사용하여
추적을 계속 반복합니다.

04

CamShift 알고리즘

- 실행 순서

1. 객체의 색상 분포를 모델로 생성

- 보통 HSV 색공간에서 Hue 히스토그램 사용

2. **백프로젝션(Backprojection)으로 확률 맵 생성**

- 각 픽셀이 객체와 얼마나 비슷한지 표시

3. **MeanShift 적용**

- 현재 윈도우 위치에서 최적의 중심으로 이동

4.. **윈도우 크기와 방향 업데이트**

- 추적 영역의 분포(2차 모멘트)를 계산하여 사각형의 크기·각도를 동적으로 조정

5. 다음 프레임에서도 **반복 수행**

- 객체가 이동·확대·축소·회전해도 추적 가능

05

루카스-카나데 알고리즘

희소(Sparse) 특징점 추적을 위한 핵심 원리

Optical Flow 추적

영상 내에서 물체의 움직임 패턴을 조밀한 벡터 필드가 아닌,
주요 특징점(Corner 등) 위주로 계산하여 추적하는 알고리즘입니다.

이를 위해 3가지 강력한 가정을 전제로 합니다.

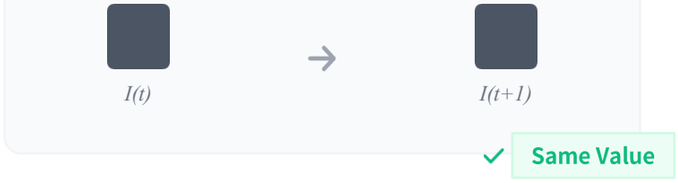
Optical Flow Equation

$$I(x, y, t) = I(x+dx, y+dy, t+dt)$$

시간 t와 t+dt 사이의 픽셀 밝기는 동일하다는 전제

1. 밝기 보존 가정

물체의 한 점은 시간이 지나 위치가 변하더라도
그 점이 가진 밝기 값(Intensity)은 변하지 않는다고 가정합니다.

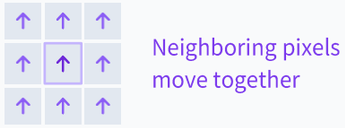


2. 작은 이동 가정

연속된 프레임 사이에서 물체의 이동량은 매우 작다
(Small Motion)고 가정합니다.
이를 통해 **테일러 급수 근사**가 가능해집니다.

3. 국소 영역 가정

어떤 픽셀의 이웃한 픽셀들은 서로 **비슷한 움직임(Motion)**을
가진다고 가정합니다. 이를 통해 연립방정식을 세워 해를 구함



05

루카스 카나데 알고리즘

- 두 프레임 사이에서 특징점의 움직임(optical Flow)을 추적하는 알고리즘
- **핵심 아이디어**

1. 밝기 보존 가정 : 물체의 한 점은 시간에 따라 밝기가 변하지 않는다.
2. 작은 이동 가정 : 한 프레임에서 다음 프레임으로 점이 조금만 이동한다.
3. 국소 영역 가정 : 인접한 픽셀은 비슷한 움직임을 가진다.

06

Dense Optical Flow

모든 픽셀의 움직임 추적

Dense Optical Flow 밀집 광류

모든 픽셀 추적

특징점(Corner)만 추적하는 Sparse Optical Flow와 달리,
영상 내 **모든 픽셀**에 대해 움직임 벡터를 계산합니다.

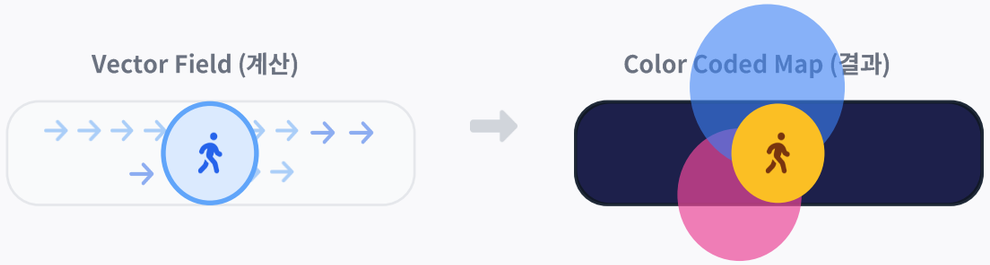
흐름 벡터(Flow Vector) 계산

각 픽셀 (x, y)마다 x축 속도(u)와 y축 속도(v)를 포함하는 2차원 벡터 필드를 생성하여 미세한 움직임까지 표현합니다.

장면 전체의 움직임 파악

배경과 객체를 포함한 전체적인 장면의 구조와 움직임을 이해할 수 있어,
동작 인식, 영상 분할(Segmentation), 자율주행 등에 활용됩니다.

Visual Concept



시각화 원리 (HSV Color Representation)

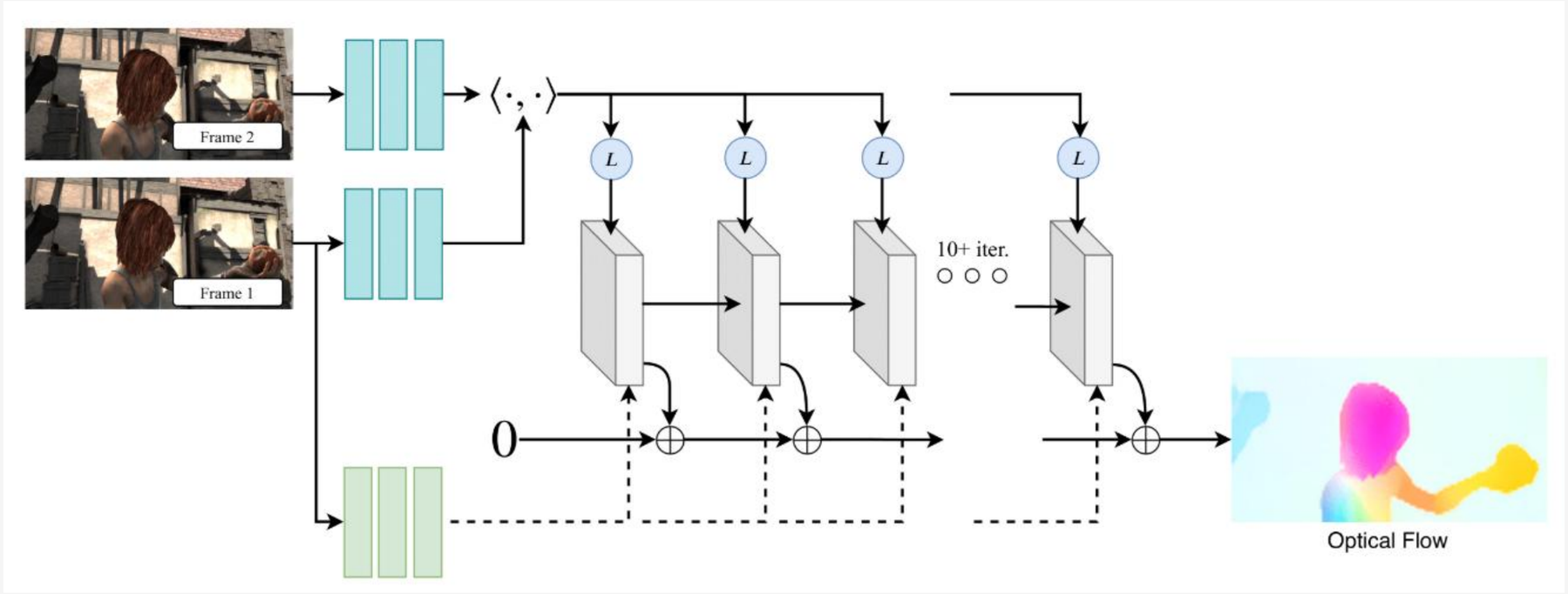
- 색상(Hue) = 움직임 방향
- 명도(Value) = 속도 크기

06

Dense Optical Flow

모든 픽셀의 움직임 추적

Dense Optical Flow 밀집 광류



07

RAFT Recurrent All-Pairs Field Transforms

딥러닝 기반 최신 기법

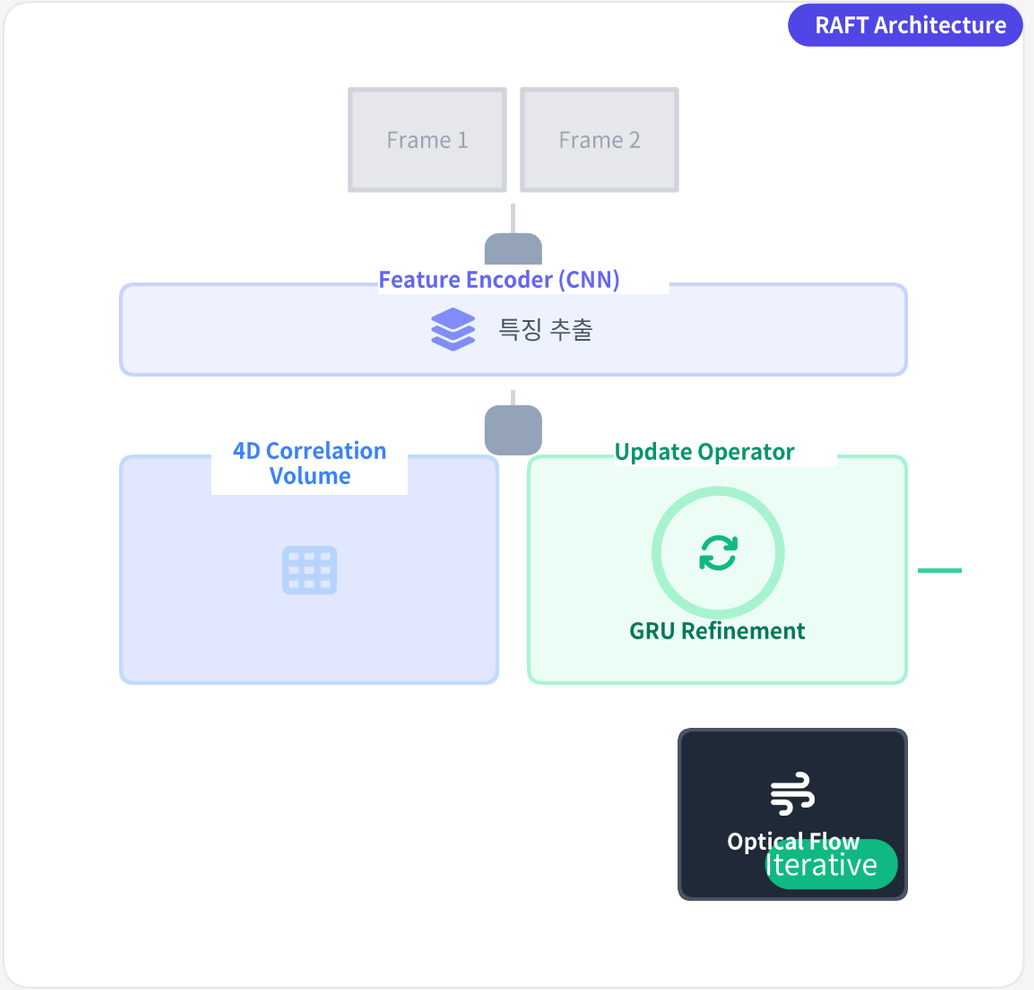
기존의 수학적 최적화 방식이 아닌, 대량의 데이터로 학습된 딥러닝 모델 (CNN + RNN)을 사용하여 Optical Flow를 추정합니다.
복잡한 움직임이나 가려짐(Occlusion) 상황에서도 강건한 성능을 보입니다.

All-Pairs Correlations (모든 쌍 상관관계)

두 이미지 프레임의 **모든 픽셀 쌍(Pixel Pairs)** 간의 유사도를 계산하여 4D Correlation Volume을 생성합니다. 이를 통해 국소적인 매칭뿐만 아니라 이미지 전체의 관계를 파악하여 큰 움직임도 놓치지 않습니다.

Recurrent Refinement (반복적 정제)

GRU(Gated Recurrent Unit) 기반의 모듈이 Flow 필드를 **반복적으로 업데이트**하며 점진적으로 정확도를 높입니다.
마치 사람이 그림을 스케치한 후 세부 묘사를 다듬어가는 과정과 유사합니다.



추적 기법 비교 및 요약 Summary

템플릿 매칭

기초

- 간단하고 직관적인 구현
- 크기, 회전, 밝기 변화에 매우 취약

활용: 고정된 아이콘/로고 검출

MeanShift

색상추적

- 빠른 속도, 색상 분포 기반 추적
- 객체의 크기 변화 및 회전에 적응 불가

활용: 단순 색상 객체 추적

CamShift

적응형

- 객체의 크기와 회전에 유연하게 적응
- 배경과 색상이 비슷하면 실패 (색상 의존)

활용: 얼굴 추적, 거리 변화 객체

Lucas-Kanade

Sparse

- 특징점 기반의 정밀한 움직임 추적
- 큰 움직임(Large Motion)이나 빠른 이동에 약함

활용: 영상 안정화, 특징점 추적

Dense Flow

전체흐름

- 배경을 포함한 장면 전체의 움직임 파악
- 모든 픽셀 연산으로 인해 계산 비용이 높음

활용: 동작 인식, 자율 주행

RAFT

Deep Learning

- 현존 최고 수준의 정확도와 강건함
- 높은 하드웨어 성능(GPU) 필수

활용: 고정밀 연구, AI 기반 비전 시스템