모션

- 동적 비전
  - 움직이는 세상에서 획득한 여러 장의 영상을 대상으로 정보를 알아내는 기능
  - 응용 예 : 감시용 카메라, 스포츠 중계, 자율 주행 차량, 로봇 항해, 게임 등

움직이는 상황

- 동적 비전이 처리하는 연속 영상(동영상)
  - 시간 축 샘플링 비율
    - 보통 30 프레임 / 초 (30 fps)
    - 특수한 경우, 수천 프레임/초 또는 60 프레임/시

$$f(y,x,t), t = 1 \leq t \leq T \qquad (10.1)$$

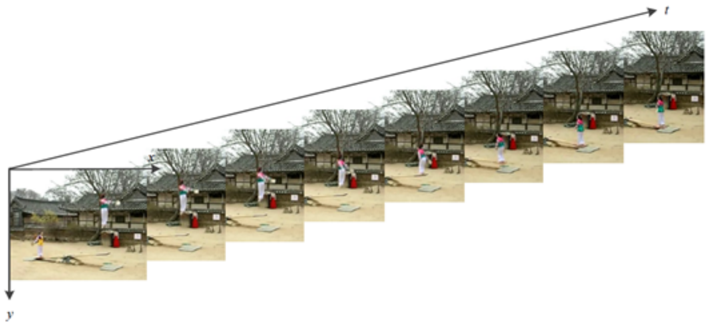


그림 10-3 시간축이 추가된 연속 영상

- 네 가지 상황
  - 정지 카메라와 정지 장면
    - 정지 영상 한 장이 주어진 상황으로서 앞 장에서 공부한 내용
  - 정지 카메라와 동적 장면
    - 동적 비전에서 다루는 문제 중 가장 단순한 경우로
    - 많은 연구를 수행하여 성공적인 알고리즘이 가장 많은 편
    - 과속 단속 또는 주차 관리용 카메라나 감시용 카메라 등은 대부분 고정되어 있으므로 이 상황에 해당
  - 동적 카메라와 정적 장면
    - 불법 주차되어 있는 차량을 찍고 다니는 단속용 차량이 이 경우에 해당
  - 동적 카메라와 동적 장면
    - 가장 복잡한 경우
    - 항해하는 로봇, 자율주행 자동차, 스포츠 중계 등
    - 스스로 방향과 줌을 조절할 수 있는 능동비전 기능을 가진 감시용 카메라 포함
- 알아내야 할 정보
  - 물체가 움직이는 방향과 속도
  - 물체에 대한 행위 인식

    (사람은 의도까지 추론, 언제쯤 컴퓨터 비전이 거기에 도달할 수 있을까?)
- 동적 비전의 기술 난이도

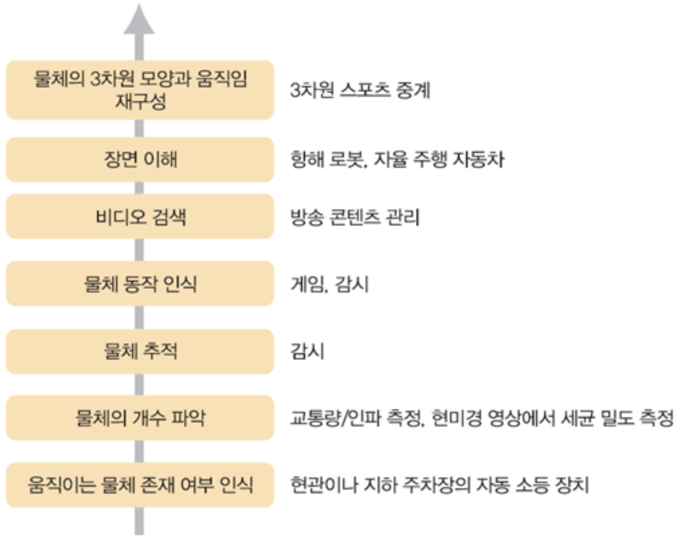


그림 10-5 **동적 비전의 기술 난이도**

- 연속 영상 처리 알고리즘
  - 알고리즘 10-1은 시간 일관성을 이용하지 않아 비효율적

알고리즘 10-1 연속 영상의 처리(시간 일관성을 사용하지 않는 순진한 버전)

입력 : 연속 영상 $f(\mathbf{x},t)$, $1 \le t \le T$

출력 : f에서 추출한 움직임 정보

```
for(t=1 to T) {
   f(x,t)를 처리하여 특징 집합 m_t를 추출한다.
}
T개의 특징 집합에서 움직임 정보를 생성한다.
```

- 영상 **일관성**(중요)
  - 공간 일관성 : 이웃 화소, $f(y,x,t)$와 $f(y+\triangle y, x+\triangle x, t)$는 비슷할 가능성이 높음
  - 시간 일관성 : 이웃 프레임, $f(y,x,t)$와 $f(y,x,t+\triangle t)$는 비슷할 가능성 높음

차 영상

- 차 영상 : 인접한 영상의 차
  - t는 현재 프레임이고 r은 기준 프레임

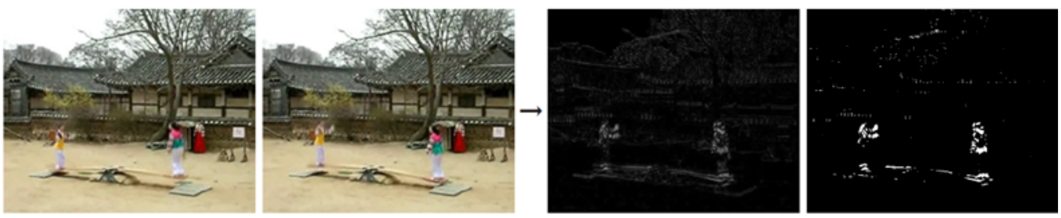
    (r은 $t-1$ 또는 물체가 나타나기 이전의 초기 배경 영상)

$$d_{tr}(y,x) = \begin{cases} 1, & |f(y,x,t) - f(y,x,r)| > \tau \\ 0, & \text{그렇지 않으면} \end{cases} \tag{10.2}$$

또는

$$d_{tr}(y,x) = \begin{cases} 1, & \frac{1}{\sigma_t \times \sigma_r}\left(\frac{\sigma_t + \sigma_r}{2} + \left(\frac{\mu_t - \mu_r}{2}\right)^2\right)^2 > \tau \\ 0, & \text{그렇지 않으면} \end{cases} \tag{10.3}$$

  - 신경 안써도 되는 수식임



> f_{11} > f_{12} > 차 영상   > 차 영상의 이진화

그림 10-6 **차 영상**

- 오차가 있을 수 있다
  - 바람
  - 조명
  - 센서 내부의 노이즈
- 차 영상을 이용한 움직임 추출 알고리즘
  - 배경과 물체의 색상에 큰 변화가 없는 상황에서만 동작

    예) 공장의 벨트 컨베이어, 조명이 고정된 실내 등

알고리즘 10-2 차 영상에서 움직임 추출

입력 : 두 장의 영상 $f(\mathbf{x},r)$, $f(\mathbf{x},t)$ // r은 기준 프레임, t는 현재 프레임

출력 : 움직임 정보

식 (10.2) 또는 식 (10.3)을 이용하여 차 영상 d를 구한다.
d의 연결요소를 구한다.
크기가 작은 연결요소를 제거한다. // 잡음으로 간주
적절한 모폴로지 연산을 수행한다(선택적). // 예를 들어 열기 연산(식 (2.24))
연결요소를 해석하여 움직임 정보를 추출한다.

모션 필드

- 3차원 모션 벡터 v_3를 복원할 수 있을까?
  - 불가능 <- 수없이 많은 3차원 벡터가 2차원의 동일한 벡터로 투영되는데, 주어진 정보는 2차원
  - 3차원 복원을 위해선 시점이 다른 **여러 대의 카메라**를 사용해야 함

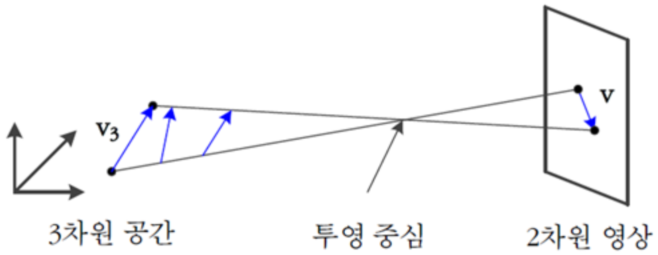
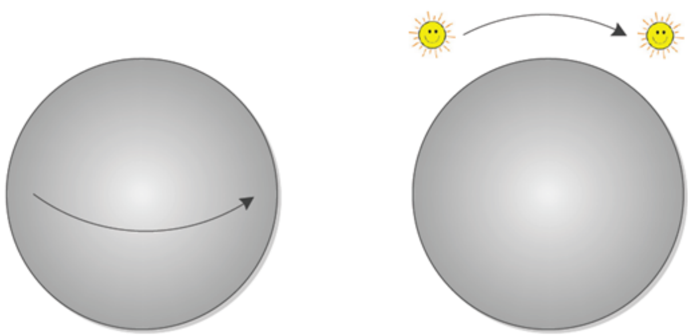


그림 10-7 3차원 모션의 2차원 투영

- 2차원 모션 벡터 추정
  - 대부분 연구는 두 장의 이웃 영상에서 2차원 모션 벡터를 추정하는 일로 국한
- 모션 필드 추정이 근본적으로 어려운 상황



(a) 구체 회전 (b) 광원 이동

그림 10-9 모션 필드 추정이 근본적으로 어려운 상황

광류

- 광류는 모션 필드의 근사 추정치

- 광류 알고리즘은 모든 화소의 **모션 벡터** $\mathbf{v} = (v, u)$를 추정해야 함
- 숫자로 구성된 영상에서 어떻게 모션 벡터를 추정하나?

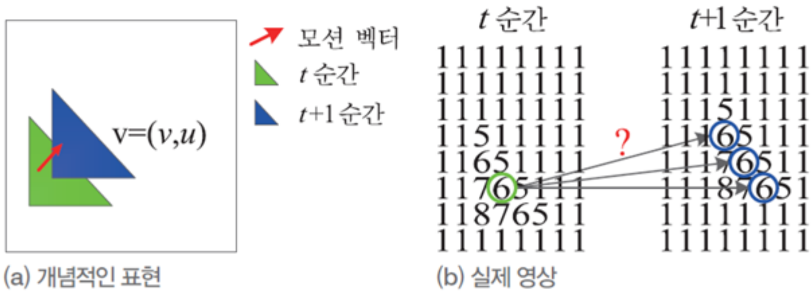


그림 10-10 모션 벡터(광류 알고리즘이 추정해야 할 정보)

  - 주변에 있는 값들도 같이 움직였다는 가정?

광류 추정의 원리

- 어려움
  - 수백 * 수백 이상 크기의 256 명암 영상
  - 여러 기하 변환과 광도 변환, 잡음 발생
  - 움직이는 물체와 정지한 물체가 혼재하고 움직임 도중에 가림도 발생
- 현실을 적절히 표현하는 모델 필요
  - 실제 세계를 훼손하지 않는 범위 내에서 적절한 가정 필요
  - **밝기 항상성** : 물체의 같은 점은 다음 영상에서 같은 (유사한) 명암 값을 가져야 함
    - 현실에 정확히 들어맞지 않지만,
      실험 결과에 따르면 받아들일 수 있을 정도의 오차 범위 이내
- 테일러 급수에 따르면

$$f(y+dy, x+dx, t+dt) = f(y,x,t) + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial t}dt + 2\text{차 이상의 항} \quad (10.4)$$

  - 두 영상 사이의 시간 차이 dt가 작다고 가정하고 2차 이상의 항은 무시
  - 밝기 항상성 가정에 따라 $f(y + \triangle y, x + \triangle x, t + \triangle t)$를 대입하면

$$\frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial t} = 0 \quad (10.5)$$

$\frac{\partial f}{\partial y}$와 $\frac{\partial f}{\partial x}$는 에지 검출에 사용한 식 (3.5)에 해당

$\frac{dy}{dt}$와 $\frac{dx}{dt}$는 dt 동안 y와 x 방향으로 이동량이므로 모션 벡터에 해당, 즉 $\frac{dy}{dt}=v$이고 $\frac{dx}{dt}=u$

- 정리해보면,

$$\frac{\partial f}{\partial y}v + \frac{\partial f}{\partial x}u + \frac{\partial f}{\partial t} = 0 \quad (10.6)$$

  - 이 식을 **광류 조건식**(그레이디언트 조건식) 이라 부름
  - 미분을 이용한 광류 추정 알고리즘 (Lucas-Kanade 알고리즘, Horn-Schunck 알고리즘 등)은 대부분 이 식을 이용
  - 그레이디언트를 구성하는 세 항의 값을 알아도 v와 u를 유일한 값으로 결정 불가능
    - (하나의 방정식에 두 개의 미지수) -> 추가적인 가정 필요

예제 10-1 광류 조건식

이 예제는 [그림 10-10]의 영상을 활용한다. 편도함수 값은 다음과 같이 이웃한 점과의 차이로 구한다고 하자.

$$\frac{\partial f}{\partial y} = f(y+1,x,t) - f(y,x,t),\ \ \frac{\partial f}{\partial x} = f(y,x+1,t) - f(y,x,t),\ \ \frac{\partial f}{\partial t} = f(y,x,t+1) - f(y,x,t)$$

그림에서 동그라미로 표시한 화소 $(5,3,t)$에 대해 편도함수 값을 계산해 보면 다음과 같다.

$$\frac{\partial f}{\partial y} = f(6,3,t) - f(5,3,t) = 7-6 = 1,\ \frac{\partial f}{\partial x} = f(5,4,t) - f(5,3,t) = 5-6 = -1,\ \frac{\partial f}{\partial t} = f(5,3,t+1) - f(5,3,t) = 8-6 = 2$$

계산 값을 식 (10.6)에 대입하면 다음과 같은 방정식을 얻는다.

$$v - u + 2 = 0$$

그래프에 표시하면 [그림 10-11]과 같다. 이 식을 해석해 보자. 구하려는 모션 벡터 (v,u)는 이 직선 상에 놓여야 하는데 유일한 한 점으로 결정할 수는 없다. 이어서 광류 추정 알고리즘에서는 밝기 항상성 이외에 또 다른 가정을 추가하여 유일한 해를 찾는 방법을 다룬다.

TIP [그림 10-10]의 상황에서는 (-1,1)이 해이다.

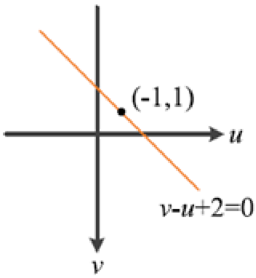


그림 10-11 모션 벡터를 위한 광류 조건식

광류 추정 알고리즘

- Lucas-Kanade 알고리즘 [Lucas84]
  - **화소 (y,x)를 중심으로 하는 윈도우의 영역 N(y,x)의 광류는 같다**라는 가정

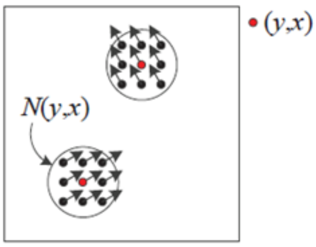


그림 10-12 Lucas-Kanade의 가정(이웃 화소는 같은 모션 벡터를 가짐)

  - 식으로 쓰면,

$$\frac{\partial f(y_i,x_i)}{\partial y}v + \frac{\partial f(y_i,x_i)}{\partial x}u + \frac{\partial f(y_i,x_i)}{\partial t} = 0,\ \ (y_i,x_i) \in N(y,x) \qquad (10.7)$$

  - 행렬 형태로 바꾸어 쓰면

$$\mathbf{A}\mathbf{v}^{\mathrm{T}} = \mathbf{b}$$

$$\text{이때 } \mathbf{A} = \begin{pmatrix} \frac{\partial f(y_1,x_1)}{\partial y} & \frac{\partial f(y_1,x_1)}{\partial x} \\ \vdots & \vdots \\ \frac{\partial f(y_n,x_n)}{\partial y} & \frac{\partial f(y_n,x_n)}{\partial x} \end{pmatrix},\ \ \mathbf{v} = (v \quad u),\ \ \mathbf{b} = \begin{pmatrix} -\frac{\partial f(y_1,x_1)}{\partial t} \\ \vdots \\ -\frac{\partial f(y_n,x_n)}{\partial t} \end{pmatrix} \qquad (10.8)$$

    - $\mathbf{v}$로 정리하면,

$$\mathbf{v}^{\mathrm{T}} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b} \qquad (10.9)$$

- 행렬의 원소가 나타나도록 쓰면,

$$\mathbf{v}^{\mathrm{T}} = \begin{pmatrix} v \\ u \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} \left(\frac{\partial f(y_i,x_i)}{\partial y} \right)^2 & \sum_{i=1}^{n} \frac{\partial f(y_i,x_i)}{\partial y} \frac{\partial f(y_i,x_i)}{\partial x} \\ \sum_{i=1}^{n} \frac{\partial f(y_i,x_i)}{\partial y} \frac{\partial f(y_i,x_i)}{\partial x} & \sum_{i=1}^{n} \left(\frac{\partial f(y_i,x_i)}{\partial x} \right)^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum_{i=1}^{n} \frac{\partial f(y_i,x_i)}{\partial y} \frac{\partial f(y_i,x_i)}{\partial t} \\ -\sum_{i=1}^{n} \frac{\partial f(y_i,x_i)}{\partial x} \frac{\partial f(y_i,x_i)}{\partial t} \end{pmatrix}$$

- 가우시안 스무딩 항을 추가하면

$$\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{A}\mathbf{v}^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{b}, \text{ 이 식을 풀면 } \mathbf{v}^{\mathrm{T}} = (\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{b} \qquad (10.10)$$

  - 행렬의 원소가 나타나도록 풀어 쓰면

$$\begin{pmatrix} v \\ u \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} w_i \left(\frac{\partial f(y_i,x_i)}{\partial y} \right)^2 & \sum_{i=1}^{n} w_i \frac{\partial f(y_i,x_i)}{\partial y} \frac{\partial f(y_i,x_i)}{\partial x} \\ \sum_{i=1}^{n} w_i \frac{\partial f(y_i,x_i)}{\partial y} \frac{\partial f(y_i,x_i)}{\partial x} & \sum_{i=1}^{n} w_i \left(\frac{\partial f(y_i,x_i)}{\partial x} \right)^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum_{i=1}^{n} w_i \frac{\partial f(y_i,x_i)}{\partial y} \frac{\partial f(y_i,x_i)}{\partial t} \\ -\sum_{i=1}^{n} w_i \frac{\partial f(y_i,x_i)}{\partial x} \frac{\partial f(y_i,x_i)}{\partial t} \end{pmatrix}$$

알고리즘 10-3 Lucas-Kanade 광류 알고리즘

입력 : 인접한 두 장의 영상 $f(y,x,t)$와 $f(y,x,t+1)$, $0 \le y \le M-1$, $0 \le x \le N-1$, 임계값 ε

출력 : 광류 맵 $\mathbf{v}(y,x)$, $0 \le y \le M-1$, $0 \le x \le N-1$

```
for(y=0 to M-1)
 for(x=0 to N-1)
   v(y, x)=velocity_vector(y, x, f_t, f_t+1);  // f_t와 f_t+1은 두 장의 입력 영상
function velocity_vector(y, x, f_t, f_t+1) {
 if((y, x)에 씌운 윈도우가 영상을 벗어나지 않으면) {  // 영상의 경계 부근은 제외
   cy=y; cx=x;
   repeat {
     식 (10.10)을 이용하여 화소 (cy, cx)의 모션 벡터 (v,u)를 계산한다.
     cy=cy+v; cx=cx+u;
   } until(‖(v,u)‖ < ε);
   return((cy-y, cx-x));  // (cy-y, cx-x)는 추정된 모션 벡터
 }
 else return(Nil);  // (y,x)는 광류 계산 불가
}
```

식 (10.10)을 반복 적용

- 특성
  - 이웃 영역만 보는 지역적 알고리즘
    - 윈도우의 크기 중요
      클수록 큰 움직임을 알아낼 수 있지만 스무딩 효과로 모션 벡터의 정확성 낮아짐
    - 해결책으로 피라미드 활용하는 기법 [Bouguet2000]
  - 명암 변화가 적은 물체 내부에 0인 벡터 발생

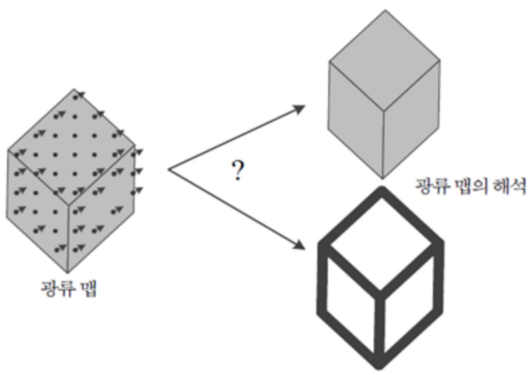
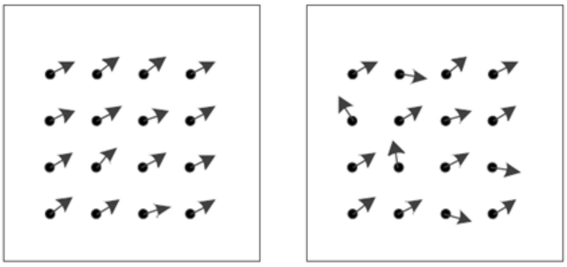


그림 10-13 Lucas-Kanade로 구한 광류 맵을 해석할 때 애매함이 생기는 경우

        - 명압값이 적으면 알아내기 힘들다(Optical Flow가 잘 작동 안함)
- Horn-Schunck 알고리즘 [Horn81]
    - **광류는 부드러워야 한다**는 가정
    - 그림의 상황에서 HS 알고리즘은 왼쪽 선호



(a) 광류가 균일한 경우 (b) 광류가 균일하지 않는 경우

그림 10-14 Horn-Schunck 가정

    - 부드러운 정도를 식으로 쓰면,
        - 값이 작을수록 부드러움

$$\|\nabla v\|^2 + \|\nabla u\|^2 = \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial x}\right)^2 \qquad (10.11)$$

    - 두 가지 목적을 동시에 만족
        - 식 (10.6)을 0에 가깝게 하면서 식 (10.11)을 될 수 있는 한 작게 함
            - 두 번째 항은 **정규화 항**
            - 정규화 기법 : 정규화 항을 이용하여 부드러운 해를 구하는 방법
        - α는 어느 것에 비중을 둘 지 결정하는 매개변수 (α가 클수록 부드러운 광류 맵)

$$E = \iint\left(\left(\frac{\partial f}{\partial y}v + \frac{\partial f}{\partial x}u + \frac{\partial f}{\partial t}\right)^2 + \alpha^2\left(\|\nabla v\|^2 + \|\nabla u\|^2\right)\right)dydx \qquad (10.12)$$

    - 식 (10.12)의 최적해를 구하는 반복식

v^0와 u^0는 0으로 초기화

$\bar{v}^k$와 $\bar{u}^k$는 현재 화소를 중심으로 하는 윈도우의 평균

$$\begin{aligned} v^{k+1} &= \bar{v}^k - \frac{\frac{\partial f}{\partial y}\left(\frac{\partial f}{\partial y}\bar{v}^k + \frac{\partial f}{\partial x}\bar{u}^k + \frac{\partial f}{\partial t}\right)}{\alpha^2 + \left(\frac{\partial f}{\partial y}\right)^2 + \left(\frac{\partial f}{\partial x}\right)^2} \\ u^{k+1} &= \bar{u}^k - \frac{\frac{\partial f}{\partial x}\left(\frac{\partial f}{\partial y}\bar{v}^k + \frac{\partial f}{\partial x}\bar{u}^k + \frac{\partial f}{\partial t}\right)}{\alpha^2 + \left(\frac{\partial f}{\partial y}\right)^2 + \left(\frac{\partial f}{\partial x}\right)^2} \end{aligned} \qquad (10.13)$$

알고리즘 10-4 Horn-Schunck 광류 알고리즘

입력 : 인접한 두 장의 영상 $f(y,x,t)$와 $f(y,x,t+1)$, $0 \le y \le M-1$, $0 \le x \le N-1$, 임계값 ε

출력 : 광류 맵 $\mathbf{v}(y,x)$, $0 \le y \le M-1$, $0 \le x \le N-1$

```
for(모든 화소 (y,x)) v⁰(y,x)=0;  // v⁰=(v⁰,u⁰)를 0으로 초기화
k=0;
repeat {
  식 (10.13)을 이용하여 v^(k+1)를 구한다.  // 모든 화소에 대해 적용함
  식 (10.12)를 이용하여 오류 E를 계산한다.
  k++;
}until(E<ε);  // 오류가 충분히 작으면 수렴했다고 간주하고 멈춤
```

- 기타 광류 추정 알고리즘
    - Lucas-Kanade는 지역적, Horn-Schunck는 전역적 알고리즘
        - LK는 값이 정해지지 않은 곳이 군데군데 발생 (명암 변화가 적은 데에서 심함)
        - HS는 번복 과정에서 값이 파급되어 밀집된 광류 맵 생성
        - 정확도 면에서는 LK가 뛰어남
    - 두 알고리즘의 장점을 결합한 아이디어 [Bruhn2005]
    - 제곱 항을 절대값으로 대치하여 물체 경계선이 불분명한 단점을 극복 [Zach2007]
- 미분 이외의 방법
    - 다양한 방법 비교분석
    - MRF
    - 그래프 절단
    - 학습 기반 필터 설계

그림 10-15 광류 맵의 예

광류의 활용

- 광류는 중간 표현
    - 물체 추적이나 제스처 인식과 같은 고급 비전을 처리하려면, 광류에서 움직임 정보나 패턴을 추출해야 함
    - 광류 맵에 나타나는 여러가지 패턴

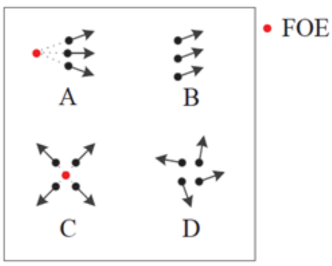


그림 10-16 광류로부터 모션 추정

- 시간이라는 축을 하나 둬서 연속적으로 들어오는 영상에 대해서의 차이를 잘 표현하는 방법 : Optical Flow Algorithm

물체 추적

KLT 추적 알고리즘

- 세부 문제들
    - 첫 프레임에서 관심 물체 검출
    - 이후 영상에서 궤적 추적
    - 가림 현상과 장면에서 사라졌다 다시 나타나는 상황 처리
    - 여러 물체의 동시 추적
- 카네기 멜론 대학의 연구
    - Lucas-Kanade 광류 알고리즘을 개조한 물체 추적 알고리즘
    - Kanade, Lucas, Tomasi의 앞 글자르 따 **KTL** 추적 알고리즘이라 부름
- 추적에 사용할 특징점 검출 방법
    - 식 (10.9)의 $\mathbf{A}^T\mathbf{A}$를 이용

$$\mathbf{H} = \mathbf{A}^{\mathrm{T}}\mathbf{A} = \begin{pmatrix} \sum_{i=1}^{n}\left(\frac{\partial f(y_i,x_i)}{\partial y}\right)^2 & \sum_{i=1}^{n}\frac{\partial f(y_i,x_i)}{\partial y}\frac{\partial f(y_i,x_i)}{\partial x} \\ \sum_{i=1}^{n}\frac{\partial f(y_i,x_i)}{\partial y}\frac{\partial f(y_i,x_i)}{\partial x} & \sum_{i=1}^{n}\left(\frac{\partial f(y_i,x_i)}{\partial x}\right)^2 \end{pmatrix} \tag{10.14}$$

    - $\mathbf{H}$의 고유값을 계산하고 추적에 유리한 정도 측정

$$\min(\lambda_1, \lambda_2) > \lambda \tag{10.15}$$

- 알고리즘

알고리즘 10–5 KLT 추적 알고리즘

입력 : 연속 영상 $f_t(y,x)$, $0 \le y \le M-1$, $0 \le x \le N-1$, $1 \le t \le T$

출력 : 궤적 $\mathbf{p}_t(i)$, $1 \le i \le n, 1 \le t \le T$

1	f_1에서 식 (10.15)를 만족하는 점을 찾고, 그들 중 특징점을 골라내 $\mathbf{p}_1(i)$, $i=1,2,\cdots,n$에 저장한다.
2	for(t=1 to T−1)
3	for(i=1 to n) {
4	$\mathbf{p}_{t+1}(i)$=velocity_vector($\mathbf{p}_t(i).v$, $\mathbf{p}_t(i).u$, f_t, f_{t+1});
5	$\mathbf{p}_{t+1}(i)$의 실종 여부를 판정하고, 실종되었다면 $\mathbf{p}_{t+1}(i)$=Nil로 설정하여 추적을 포기한다.
6	}

    - 추적 대상이 되는 특징점 골라내는 일 (1행)
    - 물체의 실종 처리 (5행)

큰 이동 추적

- 큰 이동이 발생하는 상황에서의 어려움
    - 순간적으로 크게 이동한 손과 팔에서 모션 벡터 추정 실패

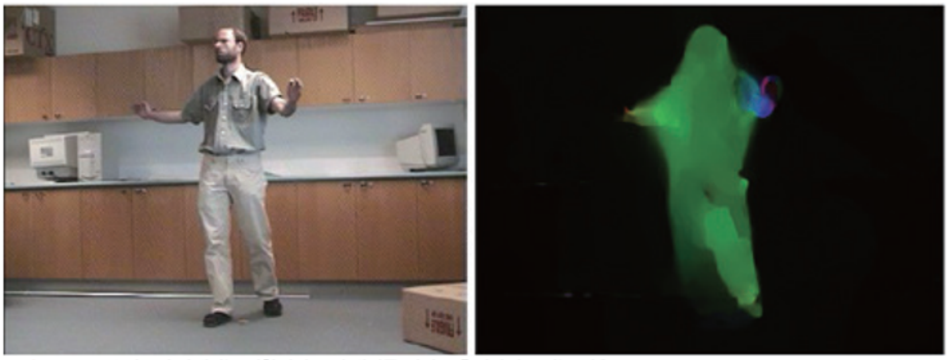


그림 10-17 큰 이동이 발생하는 상황에서 모션 벡터를 제대로 추정하지 못하는 경우

- 대응점 찾기 알고리즘으로 해결
    - [알고리즘 7-2]로 이웃한 두 영상에서 대응점을 찾은 후, 대응점을 잇는 벡터를 모션 벡터로 취함
        - 희소한 문제
        - 아웃라이어 문제
    - [Brox2011] : 대응점 찾기 적용 후, Horn-Schunck로 밀집된 맵을 구하는 2단계 처리
    - [Weinzaepfel2013] : 비슷한 접근을 사용하나, 유연한 SIFT 아이디어를 추가로 사용

(a) f_t (b) f_{t+1} : 경직된 SIFT (c) f_{t+1} : 유연한 SIFT

그림 10-18 기존 SIFT(b)와 깊은 신경망이 사용하는 SIFT(c)