VIT-LSTM : CLASSICAL VO INTRO

최병찬 (석사 4기 / 지능통신시스템 연구실)

목 차

Reference Materials

- Visual Odometry Tutorial – Davide Scaramuzza, Friedrich Fraundorfer

  (IEEE Robotics & Automation Magazine / https://ieeexplore.ieee.org/document/6096039)

- Pose from Epioplar Geometry – Thomas Opsahl (University of Oslo)

  (https://www.uio.no/studier/emner/matnat/its/nedlagte-emner/UNIK4690/v16/forelesninger/lecture_7_3-pose-from-epipolar-geometry.pdf)

- Introduction to Machine Vision (ENB339) – Peter Corke (Queensland University of Technology)

  (https://www.youtube.com/watch?v=N_a6IP6KUSE&list=PL-cZT00o1h6JGiQcrlNFGi2aVdWzH6ios)

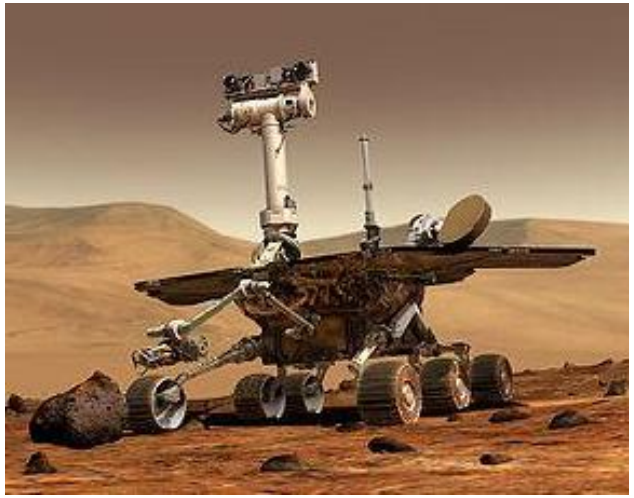
- QUT Robot Academy

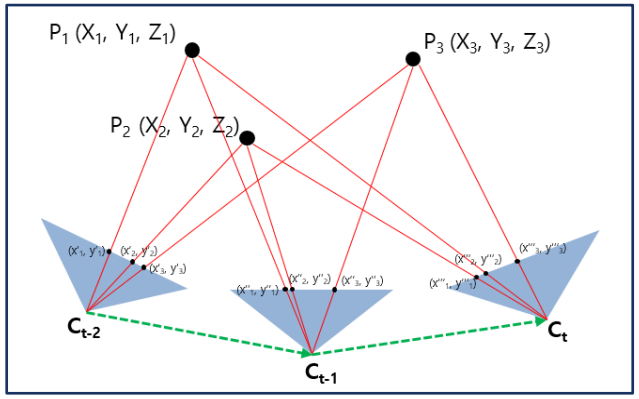
  (https://robotacademy.net.au/)

- 컴퓨터 비전 특강 – 2020 컴퓨터공학과 대학원 수업 (문영식 교수님)
- Optical Flow in OpenCV (C++/Python) (https://learnopencv.com/optical-flow-in-opencv/)
- Depth Estimation : Basics and Intuition (https://towardsdatascience.com/depth-estimation-1-basics-and-intuition-86f2c9538cd1)
- Monocular Visual Odometry using OpenCV – Avi Singh (https://avisingh599.github.io/vision/monocular-vo/)

CLASSICAL MONOCULAR VISUAL ODOMETRY

CLASSICAL MONOCULAR VO





[Visual Odometry (VO)란?]

- Camera를 이용하여 차량 또는 로봇의 이동경로 및 자세를 추정하는 방식
  - **Monocular VO : Camera 1개만 사용한 VO**
  - **Stereo VO : Stereo Vision을 이용한 VO**
  - **Multi-Camera VO : 다중 Camera의 Panorama 이미지를 이용한 VO**

[VO 등장배경]

- 화상 탐사 로봇의 위치 및 자세 추정을 위해 카메라만을 사용한 Odometry 기법 등장
- **우주 공간에서 중력에 의존적인 IMU 및 Wheel Encoder를 사용할 수 없기에** 로봇의 자세 추정을 위해 Computer Vision을 사용함

[VO 작동 원리]

- World Frame에서 존재하는 Landmark / Feature를 중심으로 카메라가 어떻게 이동했는지 파악함.
- 이 정보를 역으로 사용하여 이전 위치로부터 현재 위치에 도달하기 위해 어떻게 이동했는지 산출함.

CLASSICAL MONOCULAR VO

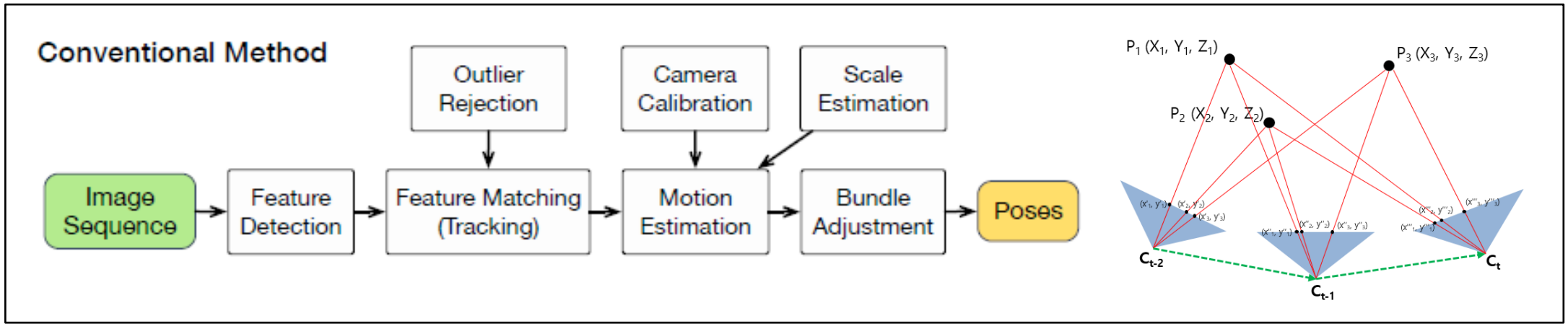
[Classical Monocular VO Pipeline]

① Feature Extraction / Detection
- 연속된 이미지에서 이미지를 표현할 수 있는 핵심 정보인 Feature를 추출함 / ORB, SURF, SIFT, Harris 등 다양한 Feature Extraction 사용 가능 / ORB는 OpenCV에서 기본적으로 사용 가능

② Feature Tracking
- 연속된 이미지에서 지속적으로 나타나는 Feature를 찾고, 추적함 / Brute Force Matching으로 모든 Feature를 비교하여 추적하거나 Optical Flow를 사용하여 손쉽게 Feature 추적할 수 있음

③ Motion Estimation
- 동일 Feature Keypoint (좌표)에 대한 Epipolar Geometry (Multi-view Geometry) 기법을 적용하여 카메라의 Rotation과 Translation을 산출함.

④ Global Optimization
- 연속된 카메라 이미지에서 동일 Feature Keypoint에 대한 Rotation과 Translation에 대한 Error를 최소화하기 위해 Reprojection Error을 최소화하게 만드는 Rotation과 Translation을 Bundle Adjustment와 같은 비선형 최적화 기법을 사용함.

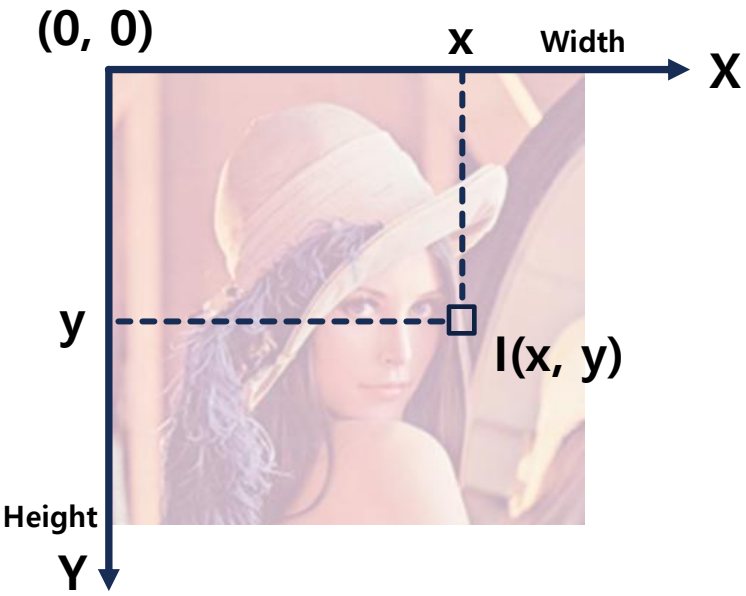


MATHEMATICAL EXPRESSION OF IMAGE

MATHEMATICAL EXPRESSION OF IMAGE

[Mathematical Expression of Image]

- **이미지를 어떻게 수학 공식으로 표현하는가?**
  - **X : 이미지의 가로 좌표**
  - **Y : 이미지의 세로 좌표**
  - **I(x, y) : x, y 좌표값에서의 이미지의 Pixel Intensity 값 • Grayscale 값 • 명암값**
  - **I_R(x, y) : RGB 이미지에서 R Channel에서의 x, y 좌표의 Pixel 값**
  - **I_G(x, y) : RGB 이미지에서 G Channel에서의 x, y 좌표의 Pixel 값**
  - **I_B(x, y) : RGB 이미지에서 B Channel에서의 x, y 좌표의 Pixel 값**
- **이미지는 각 좌표별 Pixel값의 모음 / 행렬 / 함수로 표현될 수 있음**
- **Channel, Width, Height로 이미지 행렬의 크기 • Shape이 결정됨**
  - **라이브러리에 따라 Channel First 또는 Channel Last로 이미지 행렬을 표현함**
  - ex : RGB 3 Channel 가로 224, 세로 224 Channel First 이미지 Shape : (3, 224, 224)
  - ex : Grayscale 가로 120, 세로 130 Channel Last 이미지 Shape : (120, 130, 1)



$$I = \begin{bmatrix} I(0,0) & \cdots & I(Width-1,0) \\ \vdots & \ddots & \vdots \\ I(0,Height-1) & \cdots & I(Width-1,Height-1) \end{bmatrix}$$

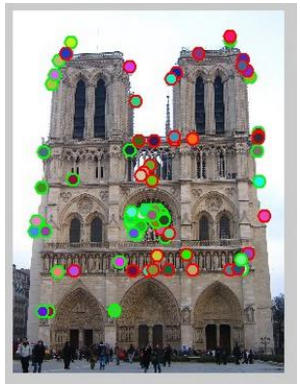
$$= \sum_{y=0}^{y=Height-1} \sum_{x=0}^{x=Width-1} I(x,y)$$

FEATURE EXTRACTION & TRACKING

FEATURE EXTRACTION & TRACKING

[Feature Extraction]

- Feature (특징점)이란 무엇인가?
  - Feature = 이미지에서 공통적으로 나타나는 흥미로운 지점 / 이미지에서 고유하게 존재할 수 있는 요소들
  - Feature Keypoint = Feature가 존재하는 이미지상의 좌표 (x, y)
  - Feature Descriptor = Feature를 표현하는 1 x N 구조의 벡터



건물의 경우
주로 Corner에
Feature가 형성됨

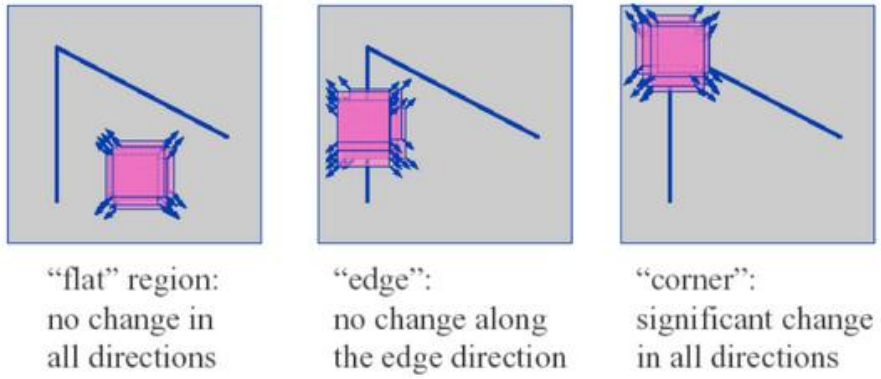
- 대표적인 Feature의 종류 : Corner, Edge, etc
  - 이미지에서 명암이 극적으로 변하는 부분 / 물체마다 공통적으로 나타나지만 형태나 개수가 고유함
  - Textureless / Plain (밋밋한) 이미지에서는 Feature를 추출하기 매우 어려움
  - 물체의 형태를 표현하는 경계선 지점에 주로 Feature로 사용될 수 있는 정보들이 많이 있음

사람의 경우
이목구비 중심으로
Feature가 형성됨

FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- **Harris Corner Detector – Corner의 명암 특징을 이용한 최초의 Feature Extractor**



Step 1. Compute the Corner Response at each pixel position:

(1) At every pixel position, compute $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}$ & $I_x^2,\ I_y^2,\ I_x I_y$

(2) At every pixel position, compute $H = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$ (Σ is within window W)

$$\Rightarrow \text{trace}(H) = \sum I_x^2 + \sum I_y^2, \quad \det(H) = [\sum I_x^2][\sum I_y^2] - [\sum I_x I_y]^2$$

$$\Rightarrow \text{코너 반응 값 } R = \det(H) - k[\text{trace}(H)]^2, \quad (k : 0.04 \sim 0.06)$$

Step 2. Thresholding on Corner Response R.

Step 3. The location with local maximum is a corner point

- 매우 작은 N x N Window (= Kernel, Filter)를 사용하여 그 주변을 움직이면서 위치 이동에 따른 이전 위치의 명암과 현재 위치의 명암 사이의 변화를 기록함

※ 위치 이동(Δx, Δy)에 따른 명암 변화 (ΔI(x, y))는 일종의 Gradient로 표현할 수 있음

- 영역의 종류별로 명암 변화가 다르게 나타나는 것을 응용함
  1. 밑밑한 영역(Flat Region)에서는 명암 변화가 매우 적음
  2. 일직선 형태의 영역 (Edge)에서는 특정 방향으로 명암 변화가 매우 적음
  3. 모서리 형태의 영역 (Corner)에서는 전 방향으로 명암 변화가 발생함
- 위와 같은 특성을 기반으로 조건식(Corner Response R)을 세우고 조건식 결과값 R이 특정 Threshold 이상인 위치를 Feature 후보군으로 모두 찾아냄
- Feature 후보군 중 주변 R값 대비 제일 높은 지점 (Local Maximum)을 Feature Keypoint로 최종 선정함

FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- Harris Corner Detector의 한계
  - Scale Change에 매우 취약함
    - Feature는 이미지마다 고유하기 때문에 이미지가 커지는 작아지든 동일 지점이 Feature로 선정될 수 있어야함
    - Harris Corner Detector와 같은 초기 Feature Extraction 알고리즘은 단순 명암 변화만을 사용하여 Feature를 찾기 때문에 이미지 크기가 변하면서 이미지의 Pixel 분포가 달라지면 Feature를 찾아내는 지점이 달라지는 문제점이 발생함

- Robust Feature Extraction의 요구 조건
  - Scale Invariance : 이미지가 커지는 작아지든 동일 지점을 Feature로 잡을 수 있어야함
  - Rotation Invariance : 이미지가 회전하더라도 동일 지점을 Feature로 잡을 수 있어야함
  - Illumination Invariance : 전반적인 명암이 변하더라도 동일 지점을 Feature로 잡을 수 있어야함

FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- SIFT – Scale / Rotation / Illumination Invariance를 갖춘 최초의 Feature Extractor이자 현대 Feature Extractor의 모태
  - SIFT = Scale Invariant Feature Transform
    - Use of DoG (Difference of Gaussian) : 다양한 Scale에서의 Corner Feature를 감지함 → Scale Invariance
    - Dominant Gradient Direction : 이미지에서 명암 변화가 제일 크게 발생하는 방향으로 모든 Feature를 통일하여 표현함

      : 이미지가 회전하더라도 명암 변화가 제일 크게 발생하는 방향도 같이 회전함

      : 회전에 맞춰서 Feature를 동일하게 표현할 수 있음 → Rotation Invariance
    - Use of Gradient : Window 움직임에 따른 명암 변화를 Gradient로 표현함 → Illumination Invariance
    - Robustness to Noise : Local Extrema 제거를 통한 이상치 제거

FEATURE EXTRACTION & TRACKING

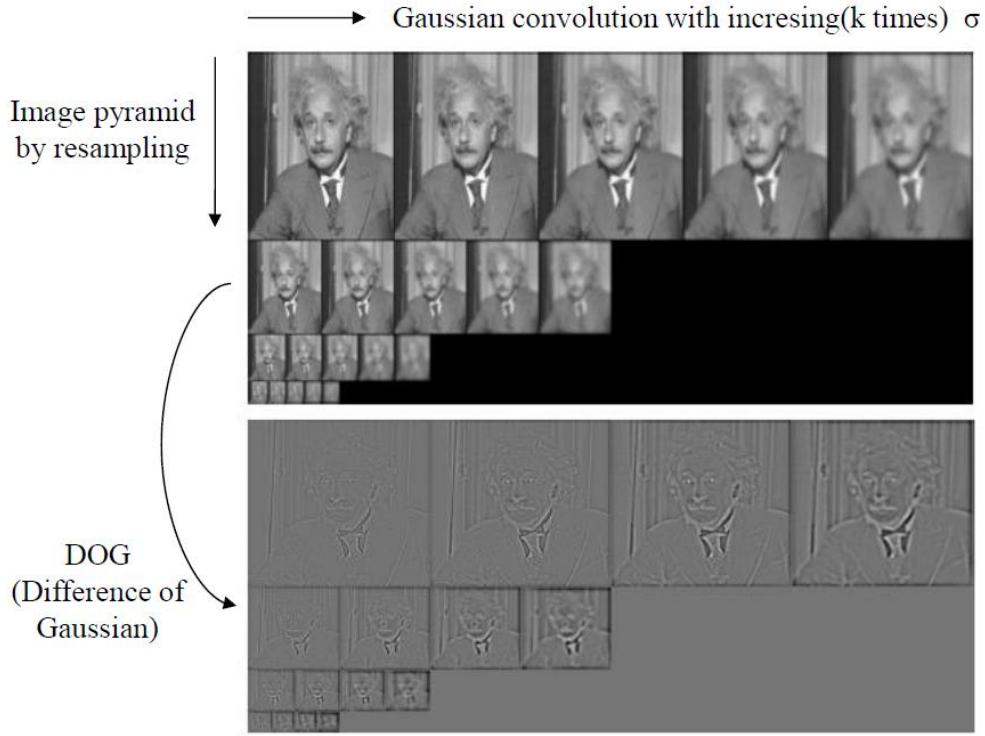
[Feature Extraction의 주요 원리]

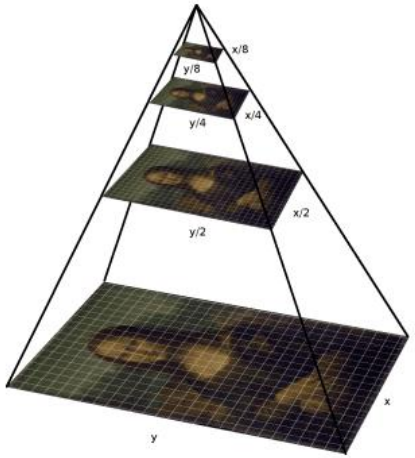
- SIFT 알고리즘 구성
  - Keypoint Detection
    - Gaussian Blur
    - Difference of Gaussian (DoG)
    - Peak Detection in Scale Space
    - Removal of Noisy Extrema
  - Keypoint Description
    - Determination of the Orientation
    - Compute the Histogram of Gradients (HoG)

FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- **SIFT – Keypoint Detection : Image Pyramid을 응용한 Scale Invariance 확보**
  - **① Gaussian Blur → ② Difference of Gaussian (DoG) → ③ Peak Detection in Scale Space → ④ Removal of Noisy Extrema**





[Image Pyramid]

- 이미지 크기를 특정 비율 단위로 줄여가면서 동일 영상처리 알고리즘을 적용하는 방법
- 전통적인 영상처리 기법에서는 이미지 크기 변화에 상관없이 성능을 유지하는 Scale Invariance를 확보하기 위해 Image Pyramid 기법을 도입함
- 딥러닝에서는 Layer가 깊어지면서 이미지 사이즈가 줄어들면서, Kernel의 상대적인 크기 비율이 커지게 만드는 것이 Image Pyramid와 유사한 방법임

FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

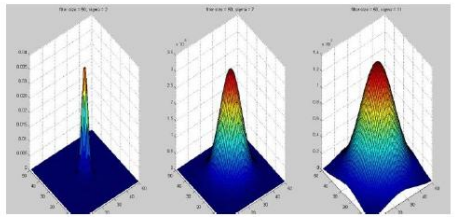
- **SIFT – Keypoint Detection**

동일 Scale에 대해 Gaussian Blur의 Variance를 바꿔서 적용한 여러 복제본을 만듬

[① Gaussian Blur]

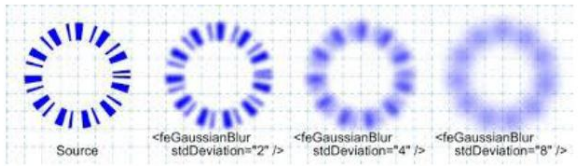
- **Convolution operation similar to averaging**
- **Gaussian convolution with scale σ => *L(x,y,σ)***

$$L(x,y,\sigma) = G(x,y,\sigma) * f(x,y) \; where \; G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$



$\frac{1}{16}$

1	2	1
2	4	2
1	2	1



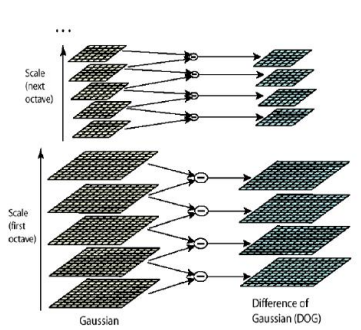
$\frac{1}{273}$

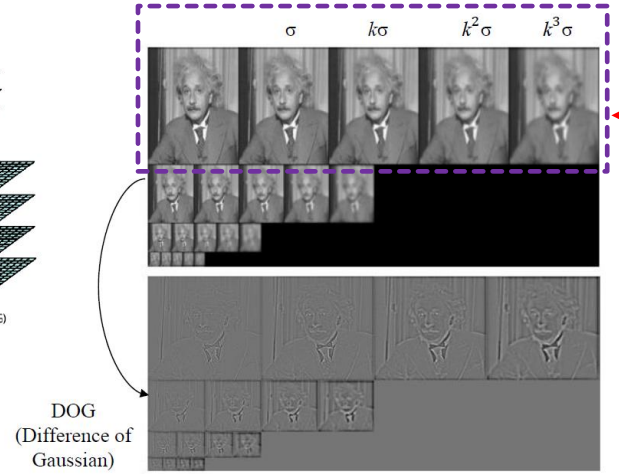
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

[② DoG (Difference of Gaussian)]

- **Compute the difference image between neighboring blur images**

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * f(x,y) = L(x,y,k\sigma) - L(x,y,\sigma)$$





- 동일 Scale 내에서 서로 이웃한 Gaussian Blur 복제본 사이 Pixel-by-Pixel 차이값 구성된 이미지 (DoG)를 생성함
- Blurring 결과물 사이의 차이값을 통해 이미지 내에서 Edge 또는 Detail한 부분에 대한 Visibility를 증가시킨 결과물을 만들어낼 수 있음

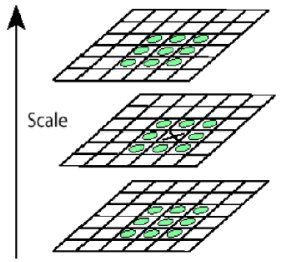
FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- **SIFT – Keypoint Detection**

[③ Peak Detection]

- **For all points in DOG images, find extrema (maxima or minima) in neighboring pixels considering 3 scales (previous, current, next)**
  => Candidate keypoints

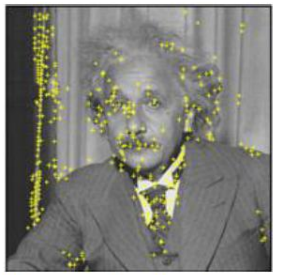


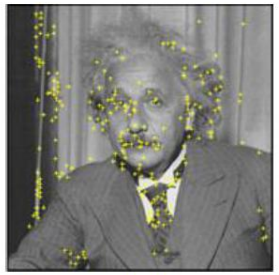
- 현재 스케일에서 특정 위치 X를 중심으로 이웃한 8개의 Pixel, 한층 높은 Scale에서 해당 위치의 9개 Pixel, 한층 낮은 Scale에서 해당 위치의 9개 Pixel 중에서 최대값과 최소값에 해당되는 지점을 찾음
- Scale이 달라지더라도 이미지에서 Edge나 Detail이 국소적으로 (Locally) 뚜렷하게 나타나는 지점들을 찾아냄으로서 Scale Invariance를 확보할 수 있음

[④ Removal of Noisy Extrema]

- **Remove extrema with low contrast**
- **Remove extrema on edges**


Candidate extrema


Removal of extrema with low-contrast

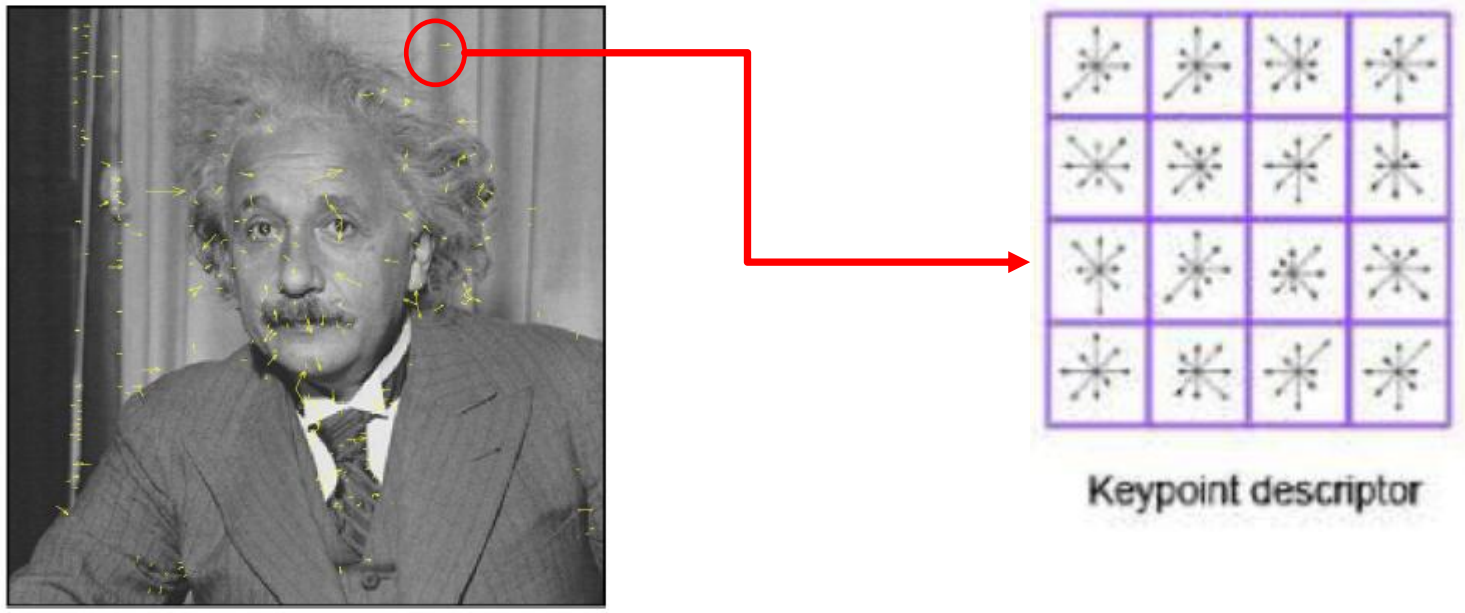

Removal of extrema on edges

- **Parameters of SIFT: No. of scale levels (4), No. of blurs(5) for each scale, values of Gaussian σ(1.6) and $k(\sqrt{2})$, contrast threshold (0.03), edge threshold(10)**
  (** values in () are recommended values)

FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- **SIFT – Keypoint Description : 이미지의 주된 빛의 방향으로 통일된 Gradient 표현을 통한 Rotation Invariance 확보**
  - **① Determine Gradient Direction → ② Compute Feature Vector using Gradient Histogram**



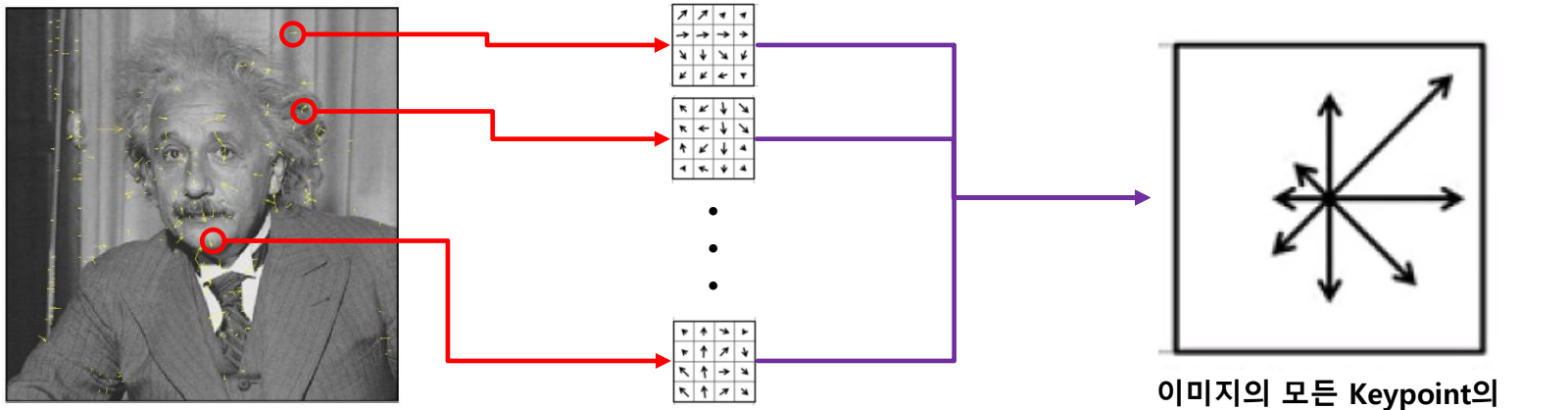
FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- SIFT – Keypoint Description

[① Determine Dominant Gradient Direction → Rotation Invariance]

- 각각의 Keypoint를 중심으로 Window 내에서 주변의 Gradient 방향값을 계산함 (Gradient를 구한 후 Magnitude만큼 나눔)
- 모든 Keypoint가 가진 Gradient 방향에 대해 Histogram을 만듬 (10^{o} 단위로 각도를 나눠서 Histogram을 생성함)
- Gradient Direction Histogram에서 제일 큰 값을 가진 Gradient Direction을 Dominant Gradient Direction라 선정함
- Dominant Gradient Direction은 이미지의 주요 광원의 방향으로 이용될 수 있으며, 이 방향은 이미지가 회전하면 같이 회전됨



각 Keypoint의 Local Gradient Direction 모음

이미지의 모든 Keypoint의
Gradient Direction에 대한 Histogram

FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- SIFT – Keypoint Description

[② Compute Feature Vector using Gradient Histogram]

- 각각의 Keypoint를 중심으로 4 x 4 Pixel Patch를 생성함
- Pixel Patch의 각각의 Pixel에 대해 Gradient의 Direction 45º 단위로 Magnitude를 계산하여 1x8 벡터로 생성함

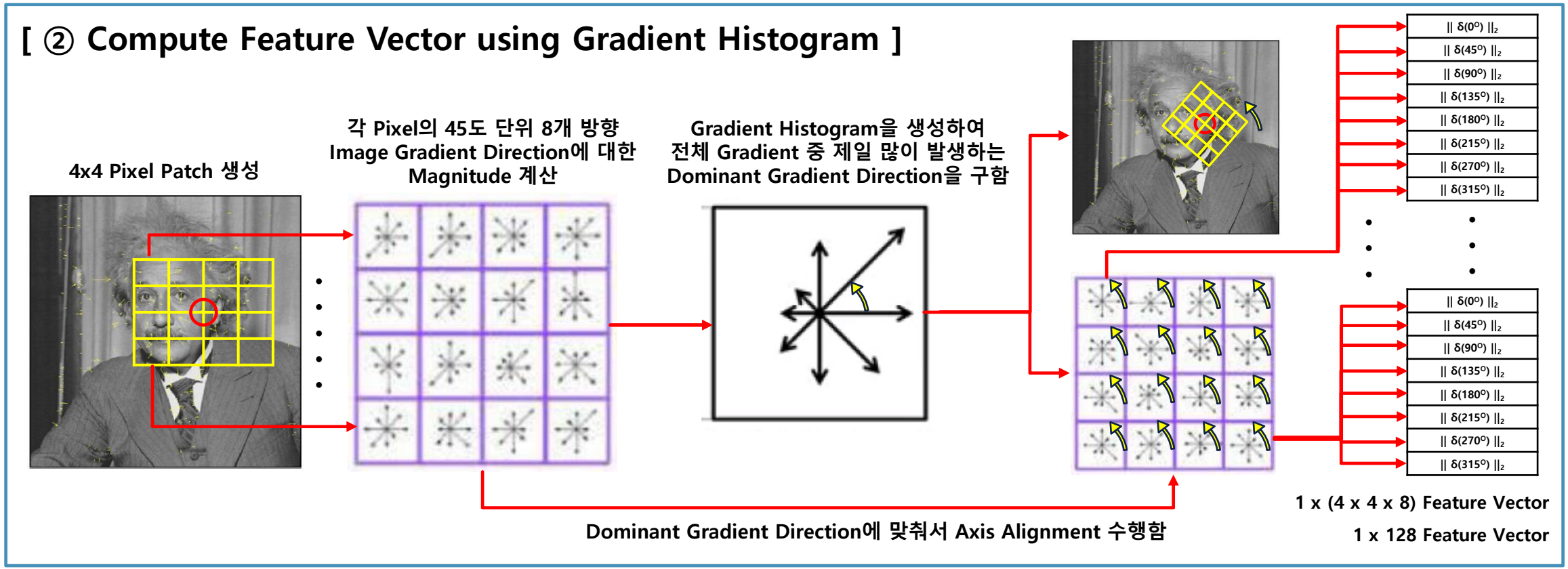
  (※ 0º, 45º, 90º, 135º, 180º, 225º, 270º, 315º 총 8가지 Gradient Direction에 대한 Magnitude를 계산할 수 있음)
- 각 Pixel은 0º를 기준으로 Gradient Direction과 Magnitude를 구했기 때문에

  이전 단계에서 구한 Dominant Gradient Direction에 0º가 맞춤 (Axis Alignment) / 이에 따라 Magnitude의 Index를 Shift함
- Keypoint의 4 x 4 Pixel Patch에서 만들어진 1x8 Gradient Magnitude 벡터를 하나로 모아서 1x128 Feature Vector 생성함

FEATURE EXTRACTION & TRACKING

[Feature Extraction의 주요 원리]

- SIFT – Keypoint Description



FEATURE EXTRACTION & TRACKING

[OpenCV의 Feature Extraction]

- Harris Corner Detection : 최초의 Corner Detection 알고리즘 **(단점 : Scale Invariance를 확보하지 못함)**
- Shi-Tomasi Corner Detection : Harris Corner Detection을 개선한 알고리즘
- SIFT (Scale-Invariant Feature Transform)
  - 최초의 Scale/Translation/Illumination/Rotation Invariant Feature Extraction 알고리즘
  - 최초로 Image Pyramid라는 개념을 도입함 (장점 : 매우 Robust함 / 단점 : 연산 요구량이 매우 높음)
  - SIFT 이후의 Feature Extraction 알고리즘은 SIFT의 파생작 (Feature Vector 형식 변화, 추출 방법 변화 etc)
  - **OpenCV 3.0 이후 Patent로 인해 Non-Free로 분류되어 별도의 방식으로 OpenCV Source Build를 해야함**
- SURF (Speeded-Up Robust Features)
  - **OpenCV 3.0 이후 Patent로 인해 Non-Free로 분류되어 별도의 방식으로 OpenCV Source Build를 해야함**

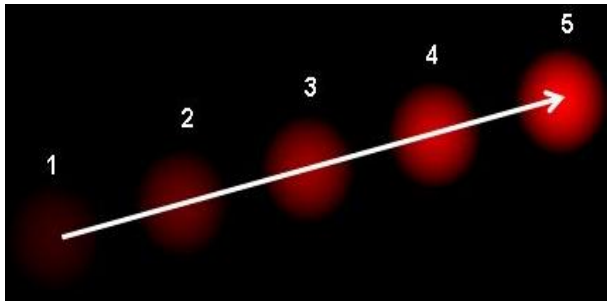
FEATURE EXTRACTION & TRACKING

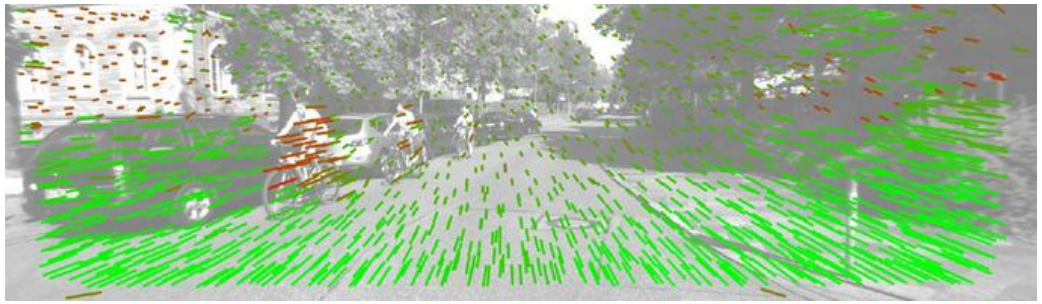
[OpenCV의 Feature Extraction]

- **FAST (Features from Accelerated Segment Test)**
  - 고속 연산이 요구되는 상황에 특화된 Feature Extraction 알고리즘 (Visual Odometry, Visual SLAM, Detection, etc)
- **BRIEF (Binary Robust Independent Elementary Test)**
  - Feature 벡터를 실수 벡터가 아닌 Binary 벡터로 산출하는 방식
  - Feature간 유사성은 Euclidean Distance가 아닌 Hamming Distance를 사용함
- **ORB (Oriented FAST and Rotated BRIEF)**
  - FAST 알고리즘과 BRIEF를 섞어서 사용하는 알고리즘 (Orientation은 FAST / Rotation을 BRIEF 산출)
  - 대표적인 무료 Feature Extraction 알고리즘
  - SIFT/SURF의 대체품으로 자주 사용되며, SIFT/SURF 대비 연산 속도와 Robustness가 거의 비슷함
  - cv.ORB를 통해 ORB Feature Extractor 객체를 생성하고, detect함수와 compute함수를 통해 Feature Descriptor와 Feature Keypoint를 산출함
  - 생성자의 nfeatures를 변경함으로서 Feature의 최대 추출 개수를 정할 수 있음

FEATURE EXTRACTION & TRACKING

[Feature Tracking – Optical Flow]





❖ 연속된 이미지 프레임 사이에서 Pixel의 밝기값(Intensity) 기반 Pixel/Keypoint 추적을 통한 물체 움직임 추적 알고리즘

❖ 사용 이유 : Pose Estimation을 위해 연속된 이미지에서 동일한 Feature Keypoint를 추적하기 위해 사용함
: 동일한 Feature Keypoint를 계속 추적해야 Pose Estimation Error를 최소화할 수 있음

❖ 추적 대상 : 2D 이미지 공간(Image Plane)의 특정 Pixel 또는 Keypoint의 이동 방향 $(\frac{du}{dt}, \frac{dv}{dt})$ (※ 3D World 좌표계상의 이동 방향이 아님)

❖ Optical Flow의 가정

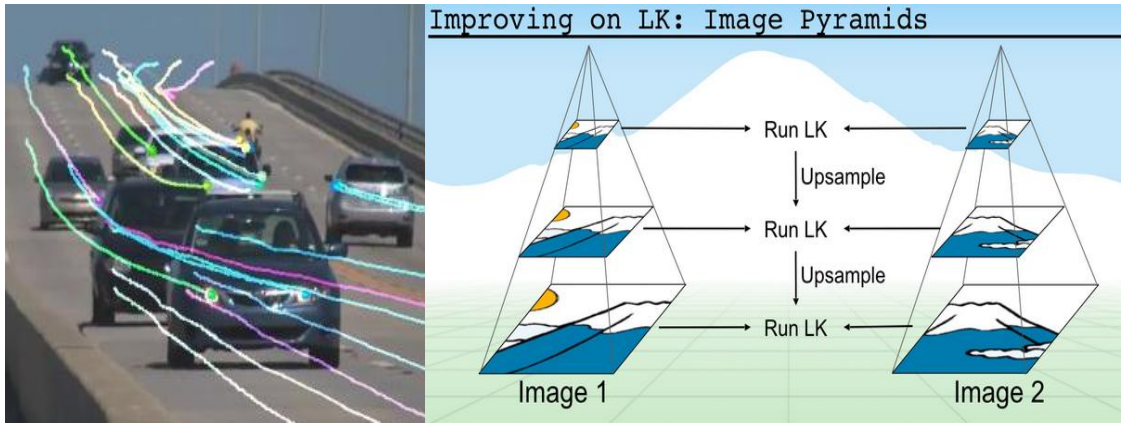
① Input 이미지는 연속적인 이미지임

② 연속되는 이미지 사이에서 관심 Pixel의 밝기값은 변함이 없음

③ 관심 Pixel의 이동과 그 인근 Pixel의 이동은 유사함 (카메라는 단시간에 크게 움직이지 않음)

FEATURE EXTRACTION & TRACKING

[Feature Tracking – 기초 Optical Flow 기법]

Lucas-Kanade Method	Gunnar-Farnebeck Method
	
❖ Sparse / Indirect / Feature-based Method ❖ 연속된 이미지 전체 Pixel 중 Feature Keypoint에 대해서만 Search Window Size 내에서 이동 변화(Flow) 추적하는 방식 ❖ Scale Invariance와 Robustness를 위해 Image Pyramid를 사용함 ❖ Grayscale 이미지에서의 Feature Keypoint를 요구함 ❖ 이미지 전체 Pixel 중 일부만 사용하기에 비교적 빠름 ❖ OpenCV cv.calcOpticalFlowPyrLK API 사용	❖ Dense / Direct Method ❖ 연속된 이미지 전체 Pixel에 대해 Pixel의 이동 변화(Flow)를 추적함 ❖ Grayscale 이미지에서의 Feature Keypoint를 요구함 ❖ 이미지 전체 Pixel에 대한 이동 변화를 추적하기에 비교적 느림 ❖ OpenCV cv.calcOpticalFlowFarneback API 사용

FEATURE EXTRACTION & TRACKING

[Feature Tracking – Lucas-Kanade Optical Flow 사용법]

- Feature-based Tracking을 위해 Grayscale 이미지에 대한 Feature Extraction과 연동하여 사용함
- Input
  - ① prevImg : 연속되는 이미지 중 이전 이미지 (I_{t-1})
  - ② nextImg : 연속되는 이미지 중 현재 이미지 (I_t)
  - ③ prevPts : 이전 이미지에서의 Feature Keypoint
  - ④ Optical Flow Parameter
    - Size : Search Window Size (N x N)
    - maxLevel : Image Pyramid 층 개수
    - TermCriteria : Match Point 탐색 종료 조건

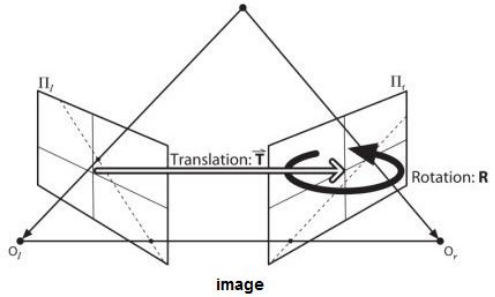
prevPts	(u_1, v_1)	(u_1, v_1)	• • •	(u_{n-1}, v_{n-1})	(u_n, v_n)
nextPts	(u'_1, v'_1)	(u'_1, v'_1)	• • •	(u'_{n-1}, v'_{n-1})	(u'_n, v'_n)
status	0	1	• • •	1	0
	미사용	사용	• • •	사용	미사용

- Output
  - ① nextPts : 현재 이미지 (I_t)에서 이전 이미지의 Feature Keypoint가 있을 예상 위치 ((u, v)로 구성된 좌표 리스트)
  - ② status : 연속된 이미지의 Feature Keypoint의 Feature Match • Correspondence 여부 (일치 여부 (0, 1)로 구성된 리스트)
- 사용방법
  - ① 연속된 이미지의 Feature Keypoint를 cv.calcOpticalFlowPyrLK에 전달하여 nextPts를 얻어냄
  - ② Output 'status'에서 1인 요소의 Index가 prevPts와 nextPts 사이의 Feature Correspondence임
  - ③ prevPts와 nextPts 사이에서 'status'가 1인 요소의 Index에 해당되는 Keypoint (Optical Flow를 통해 Match된 요소)를 사용함

EPIPOLAR GEOMETRY

EPIPOLAR GEOMETRY

[Epipolar Geometry]



image

- 2개의 이미지가 3D World 좌표계에서 동일한 점을 바라보면 해당 3D 지점과 2개 이미지상 투영된 Pixel, 총 3개의 점이 동일한 평면에 존재할 수 있다면, 2개 이미지상 투영된 Pixel 사이의 Rotation과 Translation을 구하여 카메라의 Transformation을 (Rotation + Translation) 구할 수 있다.

[Pose Estimation through Epipolar Geometry – OpenCV에서의 구현]

① 2개의 이미지 간 Feature Extraction을 통해 Feature를 추출함

➔ Feature Extraction : SIFT, SURF, ORB, etc

② Feature Tracking 또는 Matching을 통해 각 이미지에 동시에 존재하는 Feature Keypoint를 구함

➔ Feature Tracking : Lucas-Kanade, Gunnar-Farneback, PCAFlow, SimpleFlow, etc

➔ Feature Matching : Brute Force Matcher, FLANN Matcher, etc

③ 동일한 3D 지점에 대한 각 이미지의 Pixel 위치(Matching Feature Keypoint)를 기반으로 Essential Matrix를 유도함

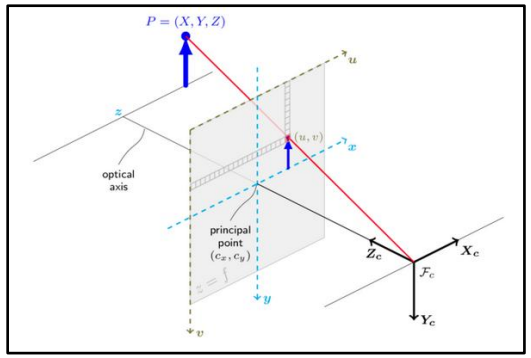
➔ OpenCV cv.findEssentialMatrix API 사용

④ Essential Matrix를 Decomposition하여 Rotation Matrix와 Translation Matrix을 유도함

➔ OpenCV cv.recoverPose API 사용

EPIPOLAR GEOMETRY

[Intrinsic Parameter for Epipolar Geometry]



3D-2D Projection Relationship

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Projected Image Point | Intrinsic Camera Matrix | Extrinsic Camera Matrix

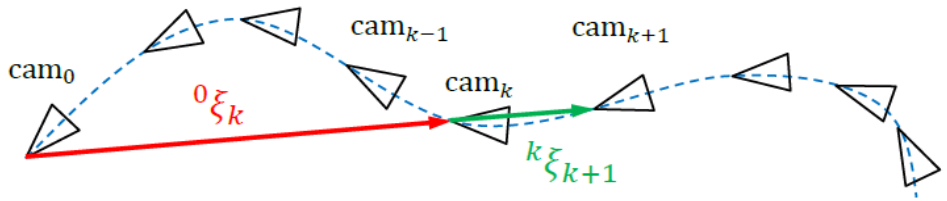
- **Intrinsic Parameter : 3D World 상의 좌표가 2D 이미지에 Projection 물리적인 특성 정의**
  - **fx, fy (Focal Length) :** 렌즈의 중심과 이미지센서와의 거리 / **OpenCV API 상 단위 = Pixel ≠ mm (거리 개념이나 거리 단위가 아닌 Pixel로 표현함)**
  - **cx, cy (Optical Center / Principal Point) :** 이미지의 중심점 ((cx = 이미지 가로 길이 * 0.5 / cy = 이미지 세로 길이 * 0.5))
- **Extrinsic Parameter : 외부 변화에 의해 3D – 2D Projection 변화되는 특성 정의 / Transformation Matrix**
  - **Rotation Matrix (r_{11} ~ r_{33}) :** 카메라의 회전을 표현함 / **Translation Matrix (t_1 ~ t_3) :** 카메라의 수평이동을 표현함

※ Camera Calibration을 통한 확실한 Camera Matrix가 확보되지 못하거나 API에 제대로 Camera Matrix를 전달하지 못하면 Essential Matrix Decomposition을 통한 Rotation/Translation Estimation이 틀리게 되면서 Odometry Error가 기하급수적으로 높아짐

(∵ 카메라의 3D-2D Projection의 물리적 특성에 대해 Error가 발생하기 때문임)

EPIPOLAR GEOMETRY

[지금까지의 과정]



Visual odometry from 2D-correspondences

1. Capture new frame img_{k+1}
2. Extract and match features between img_{k+1} and img_k
3. Estimate the essential matrix $E_{k,k+1}$
4. Decompose the $E_{k,k+1}$ into ${}^kR_{k+1}$ and ${}^k\boldsymbol{t}_{k+1}$ to get the relative pose
$$ {}^k\xi_{k+1} = [{}^kR_{k+1} \quad {}^k\boldsymbol{t}_{k+1}] $$
5. Calculate the pose of camera $k+1$ relative to the first camera
$$ {}^0\xi_{k+1} = {}^0\xi_k \, {}^k\xi_{k+1} $$

→ **Feature Extraction**

→ **Feature Tracking with Optical Flow**

→ **Essential Matrix Decomposition using Epipolar Geometry**

과연 이것만으로 모든 Visual Odometry 경우에 충분히 적용할 수 있는가?

➔ **Stereo Vision에서는 사용 가능**
(∵ 서로 고정된 2개의 카메라 간 관계를 통해 Depth를 구할 수 있음)

➔ **Monocular Vision에서는 사용 불가능**
(∵ Monocular Vision의 한계점)

SCALE ESTIMATION

SCALE ESTIMATION

[Monocular VO의 한계점 – 거리감 소실에 의한 거리 단위 소실]

- Epipolar Geometry의 Essential Matrix를 통해서 얻어낸 Translation Matrix와 Rotation Matrix는 Unit Matrix임.

- 이 뜻은 카메라의 이동 방향과 회전 방향만 알 수 있을 뿐 얼마만큼 이동과 회전을 했는지, 어떤 단위 (m or cm / Degree or Radian) 표현 해야하는지 모른다는 것임.

- Scale에 대한 정의 및 추측 (Scale Estimation)이 필요함.

- 이와 같은 현상의 이유는 Pinhole 카메라 모델에서 3D → 2D Projection의 왜곡에 의해 거리 정보가 소실되기 때문임.

SCALE ESTIMATION

[Monocular VO의 한계점 – 거리감 소실에 의한 거리 단위 소실 예시]

과연 좌측의 2D 이미지는 3D World 좌표계에서 사람의 배치를 정상적으로 보여주는가?

➔ 그렇지 않음.

➔ 사람의 크기는 거의 비슷한데 단순히 멀리 있다고 크기가 작아지고 가까이 있다고 크게 보이는 것이 왜곡임

➔ 2D 이미지만 가지고 특정 지점이 카메라로 부터 얼마나 어떤 단위로 떨어져 배치되어있는지 알 수 없음

- 카메라 1개만 가지고 특정 물체에 대한 거리감 (Depth)를 알 수 없음 (∵ 3D → 2D Projection에 의한 왜곡이 발생하기 때문임)
- 거리감을 알 수 없기 때문에 실제 움직인 거리가 m단위인지 cm단위인지 알 수 없음
- 먼 위치에서 1m 움직인 것과 가까운 위치에서 1cm 움직인 것이 단순히 Pixel 단위로는 동일하게 보임

➔ Scale Variance가 발생함

➔ Translation에 대한 Rescaling이 필요함 (Monocular Visual Odometry의 한계점)

➔ OpenCV cv.recoverPose가 제공하는 Translation Matrix는 Rescaling이 반영이 안 된 Magnitude 1의 Unit Matrix 형태임

SCALE ESTIMATION

[Stereo Vision VO에서 다른 점]

- **Stereo Vision의 경우 2개의 카메라 사이 떨어진 거리가 고정된 크기와 단위를 가지기 때문에 이를 이용하여 2개의 카메라가 동일하게 바라보는 Feature에 대해 (Feature Correspondence) 서로 간의 거리 차이값 (Disparity)으로 구성된 Depth 이미지를 만들어낼 수 있음.**
- **Disparity를 계산할 때 2개의 카메라 사이 떨어진 거리 정보와 삼각함수 관계를 이용하기 때문에 Depth 이미지의 모든 값은 특정 단위를 가질 수 있게됨. Stereo Vision은 Scale에 대한 단위를 부여하여 거리감 소실에 대한 문제가 해소될 수 있음.**

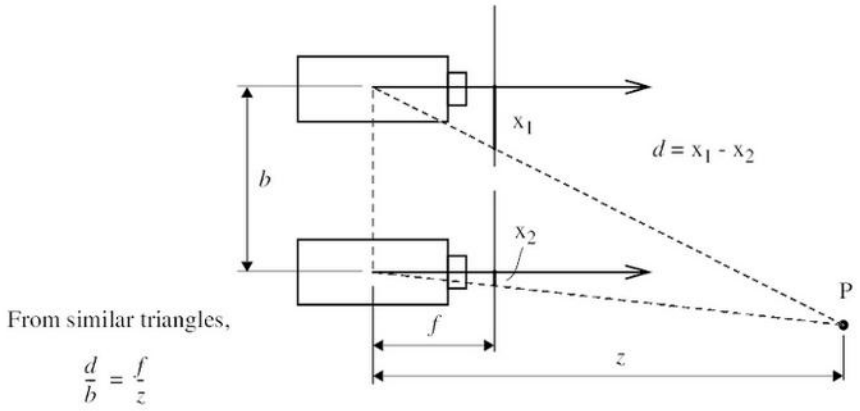
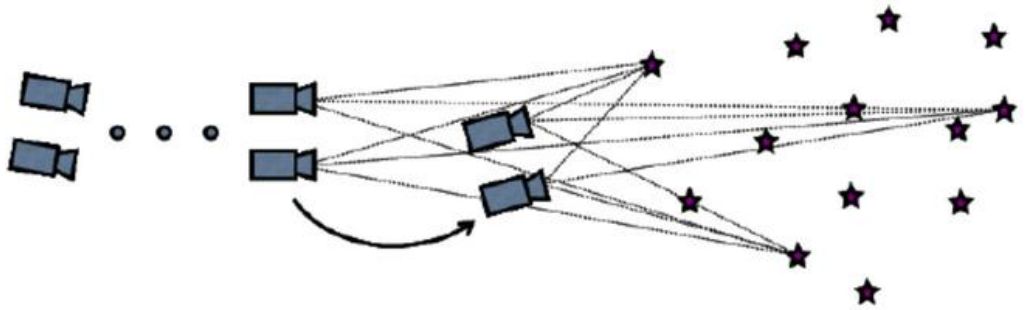


Fig 8. Stereo geometry



SCALE ESTIMATION

[Monocular VO의 한계점 극복 방법]

① 외부 센서와의 Sensor Fusion을 이용한 World 좌표계 내에서의 이동 단위 Absolute Scale 산출

: IMU 센서 or Wheel Encoder의 Ground Truth로 이용하여 이를 실제 이동 단위를 산출함

- 장점 : 현실적으로 구현하기 쉬우며, 많은 예제 Resource가 존재함

  : Sensor Fusion을 통해 센서 간 단점을 서로 보완하는 현실적인 접근 방법

  (ex : VIO = Visual Inertial Odometry (카메라 + IMU))

  : 최근 로봇 및 자율주행에서 다중 센서 기반 시스템을 구성하기에 Sensor Fusion 기반 기법들이 많이 사용됨

- 단점 : Sensor Fusion을 사용하기 때문에 센서 동기화 메커니즘이 요구됨

  : 외부 센서의 정확도에 의존하게 되며, 센서 동기화 시 제일 느린 센서 기준으로 연산 속도가 결정됨

SCALE ESTIMATION

[Monocular VO의 한계점 극복 방법]

② **거리감 확보 : 2D → 3D Triangulation을 이용한 Depth (Point Cloud) 산출 후 Relative Scale 산출**

: Triangulation은 영상처리에서 다수의 점을 이용해서 특정 지점의 Depth (거리감)을 구하는 방법

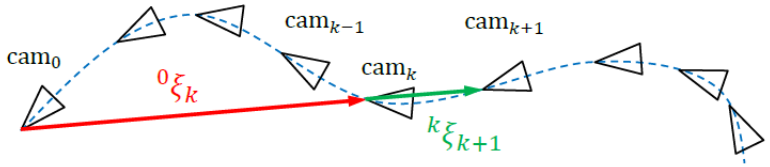
: OpenCV cv.triangulatePoints API는 2개의 이미지 사이의 Epipolar Geometry를 통해 구하는 Rotation Matrix와 Translation Matrix를 기반으로 2개 이미지가 바라보는 동일 지점에 대한 Depth를 3D Point Cloud 형태로 산출함 (2D Pixel 좌표 → 3D World 좌표 전환)

- 장점 : 외부 센서의 의존없이 카메라만 이용하여 Odometry 구현 가능
- 단점 : t-1, t-2 시간대 2개의 이미지 사이의 Depth 뿐만 아니라 t, t-1 시간대 2개의 이미지 사이의 Depth까지 필요하기 때문에

3개의 이미지에 걸쳐서 동일한 Feature Keypoint가 필요하며, 총 3개의 이미지를 기반으로 Relative Scale을 계산함

SCALE ESTIMATION

[Scale Estimation for Monocular VO]



Visual odometry from 2D-correspondences

1. Capture new frame img_{k+1}
2. Extract and match features between img_{k+1} and img_k
3. Estimate the essential matrix $E_{k,k+1}$
4. Decompose the $E_{k,k+1}$ into ${}^kR_{k+1}$ and ${}^k\boldsymbol{t}_{k+1}$ to get the relative pose
$$ {}^k\xi_{k+1} = [{}^kR_{k+1} \quad {}^k\boldsymbol{t}_{k+1}] $$
5. Compute $\|{}^k\boldsymbol{t}_{k+1}\|$ from $\|{}^{k-1}\boldsymbol{t}_k\|$ and rescale ${}^k\boldsymbol{t}_{k+1}$ accordingly
6. Calculate the pose of camera $k+1$ relative to the first camera
$$ {}^0\xi_{k+1} = {}^0\xi_k \, {}^k\xi_{k+1} $$

Monocular Visual Odometry의 경우에는...

거리감에 대한 문제 / Scaling 문제가 발생하기 때문에

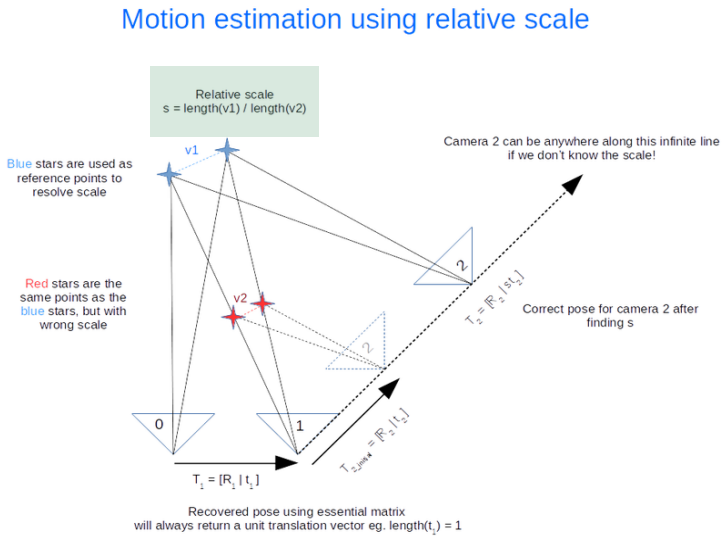
각 이미지 프레임 사이의 Relative Scale 또는

World 좌표계 내에서의 이동 단위인 Absolute Scale를 구하고

이를 거리 이동과 관련된 Translation에 반영하여 Pose를 구해야함

SCALE ESTIMATION

[Relative Scale Estimation for Monocular VO]



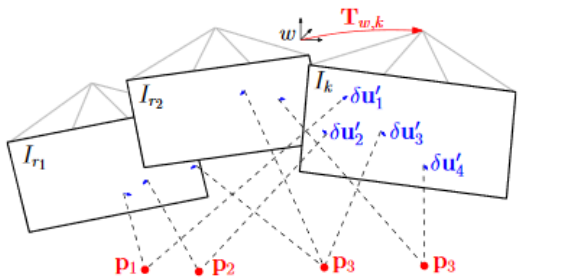


Fig. 4: In the last motion estimation step, the camera pose and the structure (3D points) are optimized to minimize the reprojection error that has been established during the previous feature-alignment step.

[Relative Scale 실제 구현 방향]

- 3개의 이미지에 걸쳐서 동일한 Feature Keypoint가 필요하기 때문에 **Optical Flow와 같은 Feature Tracking 알고리즘을 사용**

  ex : RTAB의 경우 : Feature Extraction → Feature Matching • Tracking (Optical Flow)

  → Optimization / Bad Pose Rejection

- 3개의 이미지에 걸쳐서 구한 동일 Feature Point에 대해 각각 Relative Scale를 구하고, **여러 Relative Scale 중 Median 값을 선정하여 Translation에 반영함**

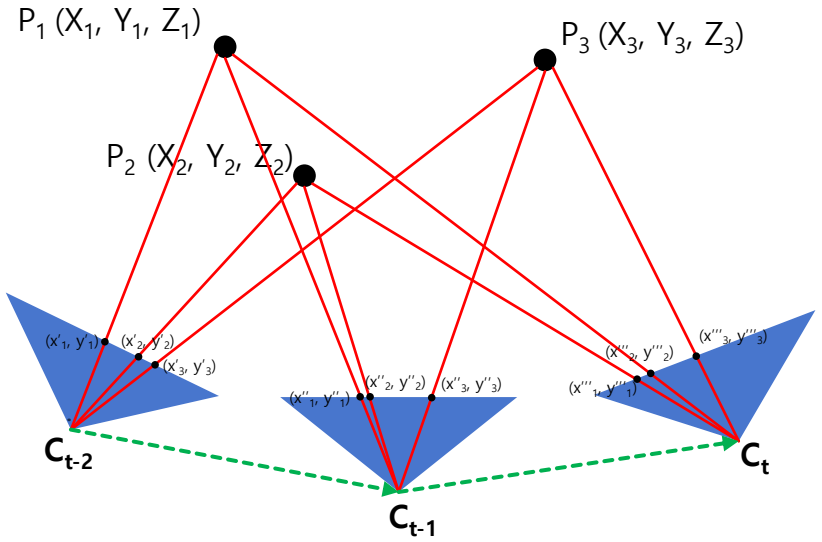
  (※ Median : 오름/내림차순으로 정렬된 배열에서 정중앙에 위치한 값)

  (※ Median을 사용하는 이유 : Depth 연산에 대한 Error를 최소화하기 위한 방법)

SCALE ESTIMATION

[Relative Scale Estimation for Monocular VO]

[Ideal Case]
카메라가 이동하더라도
2D 이미지에서 보는 동일한 지점/물체가
3D World에 동일한 지점/물체로 재투영됨

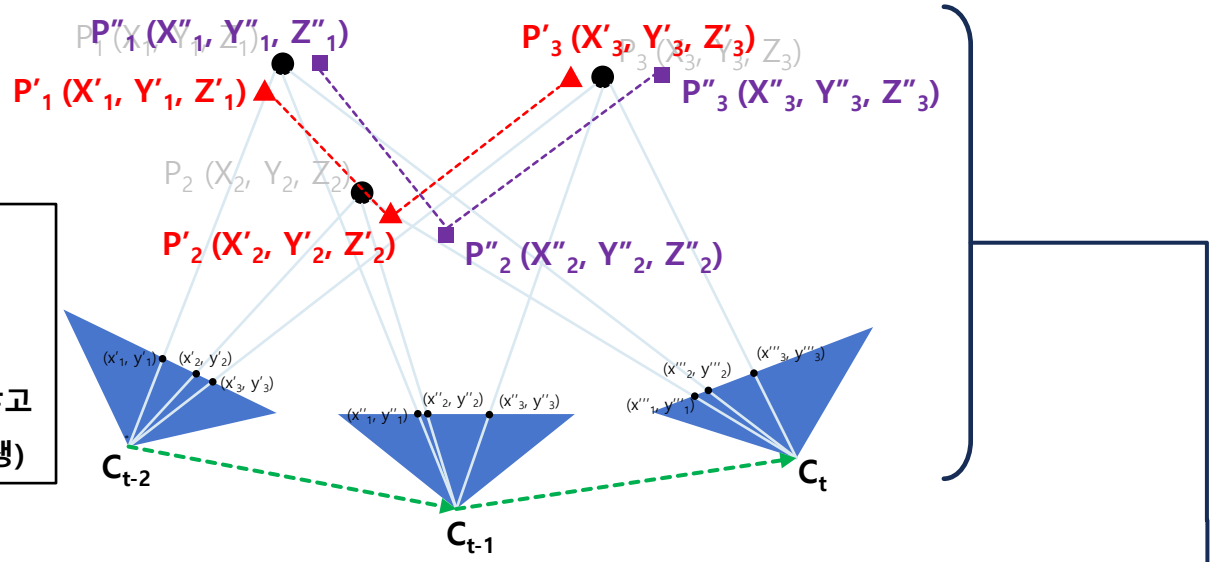


- 3개의 이미지에 걸쳐서 감지된 동일한 Feature는 각 시점에서 2D 이미지로 각각 다른 2D Pixel 좌표(Keypoint)로 Mapping됨.
- 연속된 2개의 이미지에 감지된 동일한 Feature에 대한 각 이미지의 Keypoint를 Triangulation (cv.triangulatePoints)을 통해 3D World에서의 Depth Z값을 구하여 각 Feature Keypoint를 3D Point Cloud로 산출할 수 있음.
- 연속된 3개의 이미지에서 2개 연속 이미지 1쌍씩 Feature Extraction & Matching을 통해 동일한 Feature Keypoint를 구하고 Triangulation을 통해 t-2 ~ t-1 사이의 3D Point Cloud와 t-1 ~ t 사이의 3D Point Cloud를 각각 구함

SCALE ESTIMATION

[Relative Scale Estimation for Monocular VO]

[Real Case]
카메라가 이동하면서
2D 이미지에서 보는 동일한 지점/물체가
3D World에 동일한 지점/물체로 재투영되지 않고
다른 지점에 재투영됨 (Reprojection Error 발생)

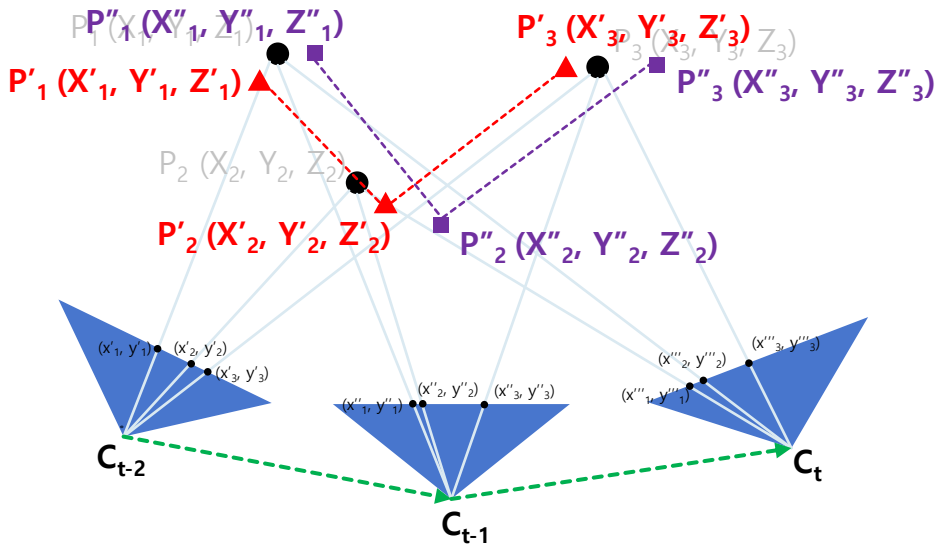


$$\therefore \text{Relative Scale} = \text{median}\left(r = \frac{||X_{k-1,i} - X_{k-1,j}||}{||X_{k,i} - X_{k,j}||} \cdot \right) = \text{median}\left(\left(\frac{||P'1 - P'2||}{||P''1 - P''2||}, \frac{||P'2 - P'3||}{||P''2 - P''3||}, \cdots \right) \right)$$

- 만약 Absolute Scale이 반영되었다면 연속된 3개의 이미지에서 2개 연속 이미지 1쌍에 대한 Triangulation을 통해 산출된 t-2 ~ t-1 사이의 3D Point Cloud (P′)와 t-1 ~ t 사이의 3D Point Cloud (P″)는 3D World 상에 있는 좌표 (P)와 거의 일치하는 좌표값을 가져야함
- 그러나 2개 연속 이미지 쌍이 각자 Triangulation을 통해 기준없이 Depth를 추정하기 때문에 각 이미지 쌍의 Point Cloud의 분포는 각자 다르게 배치됨.
  카메라 이동에 따른 각 이미지 쌍 사이의 3D Point Cloud 이동 비율을 알기 위해서 각 Point Cloud의 지점 간 거리 비교 비율을 Relative Scale로 사용함.

SCALE ESTIMATION

[Relative Scale Estimation for Monocular VO]



※ Relative Scale Calculation 구현 유의사항 (Many Good Features & Matches / 최소 1000개 단위 공통 Feature 필요)

① t-2, t-1, t 이미지가 바라보는 Feature가 모두 동일할 것이라는 보장이 없음

② t-2, t-1, t 이미지가 바라보는 Feature Keypoint가 최대한 같게 만들기 Translation • Rotation-Invariant한 고성능의 Feature Extraction과 Parameter Tuning 필요함

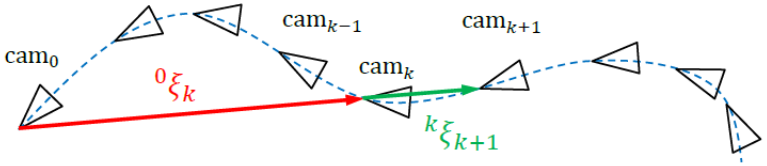
③ t-2, t-1, t 이미지의 동일한 Feature Keypoint를 최대한 많이 추적하기 위한 Feature Tracking (ex : Optical Flow)가 필요함

④ t-2, t-1, t 이미지 Feature Match의 괴리가 클수록 Relative Scale Error가 누적되어 Scale Drift가 발생함

CLASSICAL MONOCULAR VO PIPELINE

CLASSICAL MONOCULAR VO PIPELINE

[Overview of Monocular VO Algorithm]



Visual odometry from 2D-correspondences

1. Capture new frame img_{k+1}
2. Extract and match features between img_{k+1} and img_k
3. Estimate the essential matrix $E_{k,k+1}$
4. Decompose the $E_{k,k+1}$ into ${}^kR_{k+1}$ and ${}^k\boldsymbol{t}_{k+1}$ to get the relative pose
   $${}^k\xi_{k+1} = [{}^kR_{k+1} \quad {}^k\boldsymbol{t}_{k+1}]$$
5. Compute $\|{}^k\boldsymbol{t}_{k+1}\|$ from $\|{}^{k-1}\boldsymbol{t}_k\|$ and rescale ${}^k\boldsymbol{t}_{k+1}$ accordingly
6. Calculate the pose of camera $k+1$ relative to the first camera
   $${}^0\xi_{k+1} = {}^0\xi_k \, {}^k\xi_{k+1}$$

Feature Extraction

Feature Tracking with Optical Flow

Essential Matrix Decomposition using Epipolar Geometry

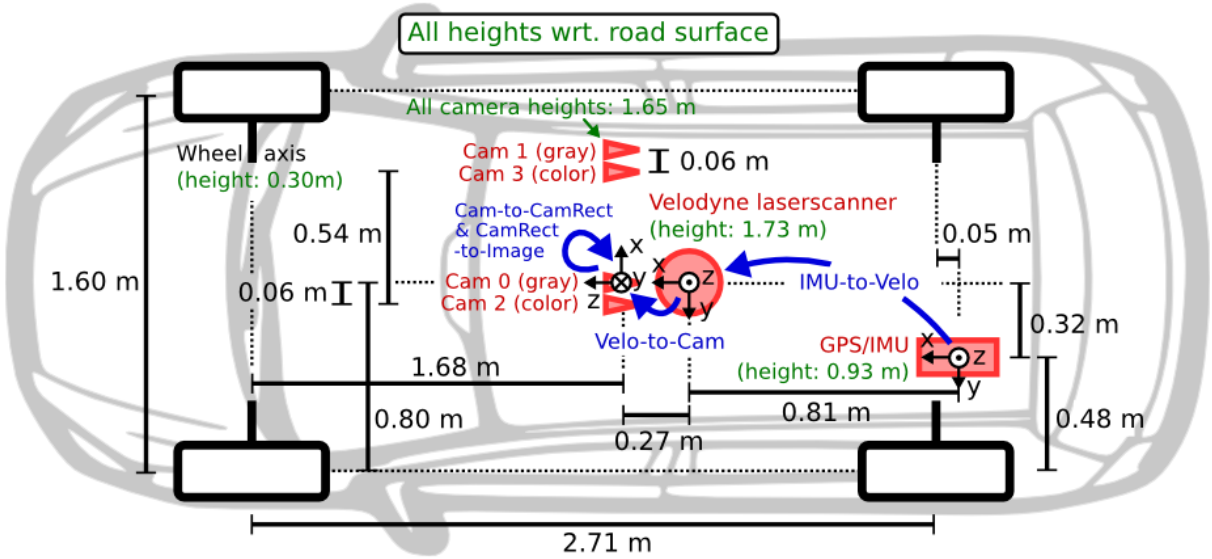
Common Feature Selection

Scale Estimation

Odometry Calculation

CLASSICAL MONOCULAR VO PIPELINE

[KITTI Odometry Dataset]



- 독일 Karlsruhe Institute of Technology에서 제작한 Odometry / SLAM 데이터셋
- VO / VSLAM 연구에서 매우 유명하고, 널리 사용되는 데이터셋
- 좌측 카메라 기준으로 Z축이 Forward Motion (앞뒤), X축이 Lateral Motion (좌우)으로 Groundtruth IMU 좌표계를 변환하여 정의함
- Groundtruth IMU의 Z축과 X축 값을 사용하여 Groundtruth 경로를 그릴 수 있음

- 1 Inertial Navigation System (GPS/IMU): OXTS RT 3003
- 1 Laserscanner: Velodyne HDL-64E
- 2 Grayscale cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3M-C)
- 2 Color cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3C-C)
- 4 Varifocal lenses, 4-8 mm: Edmund Optics NT59-917

CLASSICAL MONOCULAR VO PIPELINE

[Monocular VO Pipeline Implementation – Common Feature Selection & Relative Scale Estimation]

t=0

ORB Feature Extraction

t=1

ORB Feature Extraction

t-2 (pprev) 이미지와 t-1 (prev) 이미지 사이의 Match되는 Feature를 다음 단계에서 사용해야 3개의 이미지 모두에서 보는 동일한 지점을 찾을 수 있음

Feature Correspondence가 확보 안 되면 Scale Drift가 매우 크게 발산함

t=2

ORB Feature Extraction

prev_keypoints

pprev_keypoints

Lucas-Kande Optical Flow

prev_keypoints[st==1]

pprev_keypoints[st==1]

Lucas-Kande Optical Flow

pprev, prev에 Match되는 Feature 중 Current Feature와 Match 되는 것만 사용

current_keypoints

(prev_keypoints[st==1])[st==1]

(pprev_keypoints[st==1])[st==1]

current_keypoints[st==1]

prev_keypoints[st==1]

pprev_keypoints[st==1]

Pose Estimation Through Epipolar Geometry

Unit Rotation Matrix

Unit Translation Matrix

3D Point Cloud Generation Through Triangulation

Reprojected 3D Point Cloud

Pose Estimation Through Epipolar Geometry

Unit Rotation Matrix

Unit Translation Matrix

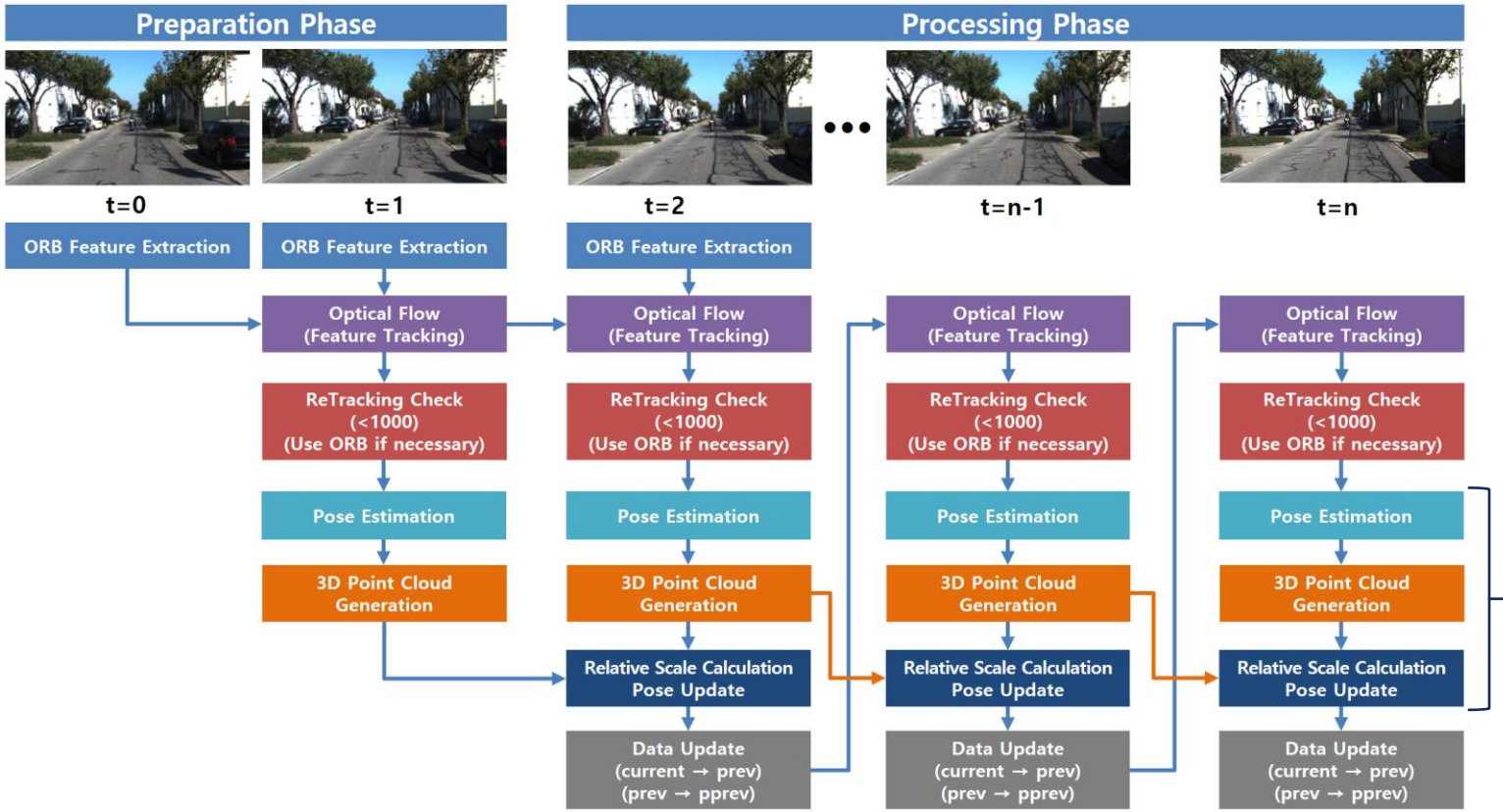
3D Point Cloud Generation Through Triangulation

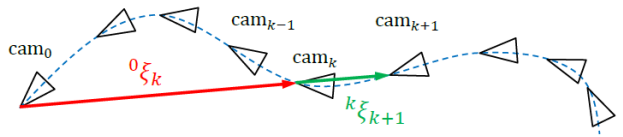
Reprojected 3D Point Cloud

Relative Scale Calculation

CLASSICAL MONOCULAR VO PIPELINE

[Monocular VO Pipeline Implementation – Overall Algorithm Process]





Visual odometry from 2D-correspondences

1. Capture new frame img_{k+1}
2. Extract and match features between img_{k+1} and img_k
3. Estimate the essential matrix $E_{k,k+1}$
4. Decompose the $E_{k,k+1}$ into ${}^{k}R_{k+1}$ and ${}^{k}\boldsymbol{t}_{k+1}$ to get the relative pose
   $${}^{k}\xi_{k+1} = [{}^{k}R_{k+1} \quad {}^{k}\boldsymbol{t}_{k+1}]$$
5. Compute $\|{}^{k}\boldsymbol{t}_{k+1}\|$ from $\|{}^{k-1}\boldsymbol{t}_{k}\|$ and rescale ${}^{k}\boldsymbol{t}_{k+1}$ accordingly
6. Calculate the pose of camera $k+1$ relative to the first camera
   $${}^{0}\xi_{k+1} = {}^{0}\xi_{k}\,{}^{k}\xi_{k+1}$$

$$R_{pos} = RR_{pos}$$

$$t_{pos} = t_{pos} + tR_{pos}$$

t_{pos}의 X축, Z축 값을 Trajectory로 그려내면 Odometry Trajectory가 됨

소스코드 : https://github.com/luwis93choi/Multi-View_Monocular_Visual_Odometry

CLASSICAL MONOCULAR VO PIPELINE

[Optimization & Bad Pose Rejection]

- Epipolar Geometry와 2D → 3D Triangulation을 통한 Pose Estimation이 항상 정확한 것은 아님

(!) 카메라가 정지하고 있으나 외부 영향으로 Feature Keypoint가 흔들리면 카메라가 움직였다고 오인식함.

➔ Forward Dominant Motion Model 적용 : 카메라가 탑재된 차량은 앞뒤 이동을 우선적으로 반영함

➔ Feature Keypoint 이동이 매우 미세한 경우 Frame Skip 기능 추가

(!) Triangulation 연산이 정확도가 높은 편은 아님. 이로 인해 Pose값이 Jump하는 현상이 발생함.

➔ Bundle Adjustment와 같은 Optimization 적용

(ex : RTAB의 경우 g2o Bundle Adjustment 라이브러리를 적용하여 Bad Pose Rejection 수행)

다음주 주제 / 이번주 과제

[다음주 주제]

- Classical Monocular VO Implementation using OpenCV

[이번주 과제]

① A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks 읽기

- https://arxiv.org/pdf/1909.05214.pdf

② 격주 발표 과제

- 12월 24일까지 진행한 강의내용 및 읽은 논문을 정리하여 발표 준비 (교수님 참관 예정)
- 발표일 : 1월 7일

감사합니다