Neural Network 기반 Image Feature Extraction 기법

□ 최건찬, 한종기 / 세종대학교

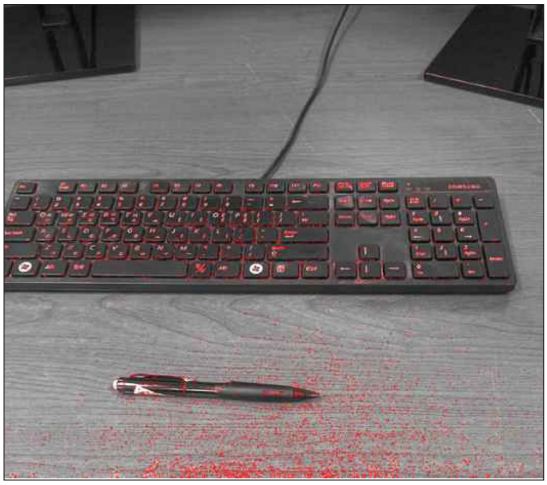
요 약

Feature extraction은 간단한 컴퓨터 비전 기술부터 VR, AR, 로봇공학 등 최첨단 기술에 이르기까지 다양한 기술에 활용될 수 있으며 따라서 이에 대한 연구가 활발히 이루어지고 있다. 기존 Feature extraction 방법에 해당하는 SIFT, SURF, ORB 등의 기술은 각자 나름대로 촬영 환경 변화에 강인한 feature를 얻기 위해 다양한 방법을 사용하지만, 모든 상황에서 유연한 feature를 얻기에는 한계가 존재한다. 이를 해결하기 위해 최근 다양한 Neural Network를 기반으로 하는 feature extraction 및 feature matching 방법이 연구되고 있다. 본 글에서는 Feature extraction에 대해 설명하고, 최근에 많이 활용되고 있는 Neural Network 기반의 feature extraction 및 feature matching 기술들을 소개한다.

Ⅰ. 서론

4차 산업혁명으로 인하여 다양한 최첨단 기술들에 대한 연구가 활발히 이뤄지고 있다. 이에 다양한 영상 처리 및 컴퓨터 비전 기술이 사용되는데, 가장 기초가 되는 기술 중 하나로 feature extraction 및 feature matching이 있다. Feature extraction은 이미지에서 사물 또는 local 한 영역의 고유 정보를 담고 있는 feature를 추출하는 과정으로, 기존 알고리즘들은 주로 feature의 위치를 탐색하고 각 feature마다 descriptor를 할당하는 과정을 거친다. 이러한 방식에서 feature는 주변의 정보를 담고 있으므로 중요한 정보가 많은 지점을 feature의 위치로 정하는 것이 중요한데, 따라서 corner point처럼 주변에서 값이 크게 변하는 위치를 사용하는 경우가 많다.

Feature extraction과 feature matching은 다양한 분야에서 활용될 수 있는 중요한 기술 중 하나이다. Feature 간의 matching을 통해 이미지간의 matching 작업을 수행할 수 있는데, 이를 통해 Image Retrieval 작업이나 VPR(Visual Place Recognition) 작업을 수행할 수 있다. 특히 VPR은 이미지간의 matching을 이용하여 camera

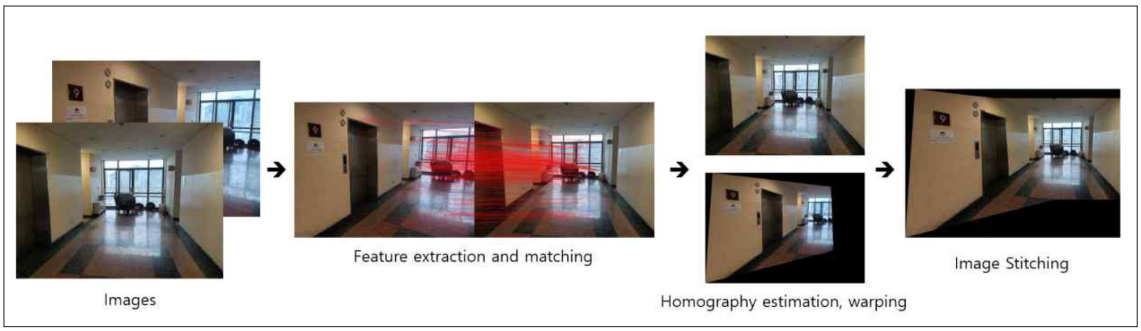


<그림 1> Feature의 개념

pose를 추정하는 Visual Localization에도 활용될 수 있다. 또한, 두 개의 이미지간의 matching을 통해 두 이미지간의 관계를 나타내는 homography 행렬 또는 essential 행렬 등을 추정할 수 있는데, 이러한 관계 행렬을 구함으로써 이미지 stitching이나 SFM(Structure from Motion) 또는 3D Reconstruction과 같이 VR에 활용될 수 있는 작업을 수행할 수도 있다. 이외에도 자율주행 기술에 쓰이는 SLAM(Simultaneous Localization and Mapping)과 같은 기술 등에도 활용될 수 있다. 이렇듯 feature extraction 및 feature matching은 기본적인 이미지 처리 작업뿐만 아니라 최첨단 기술에도 활용될 수 있는 중요한 작업이다.

기존에 많이 활용되던 feature extraction 기법에는 SIFT(Scale Invariant Feature Transform)[1][2], SURF(Speeded Up Robust Features)[3], ORB(Oriented FAST and Rotated BRIEF)[4] 등이 있다. SIFT[1][2]는 scale에 강인한 feature를 얻기 위해 Scale Space를 생성하여 feature의 위치를 찾는다. DoG 연산을 기반으로 feature의 위치를 찾고, 회전에 강인한 feature matching을 위해 feature descriptor 생성 과정에서 feature 주위 픽셀들의 gradient를 고려한다. SURF[3]는 SIFT의 원리를 바탕으로 하면서 더 빠르게 feature를 추출하고자 하는 목적으로 연구된 방법이다. SIFT처럼 Scale Space를 만들지만 DoG 필터를 단순화시킨 필터를 사용하고 적분 영상을 이용함으로써 계산 속도를 높이고자 하였다. ORB[4]는 FAST[5] 알고리즘을 개선시킨 방법으로 feature의 위치를 찾고, BRIEF[6] 알고리즘을 개선시킨 방법으로 각 feature descriptor를 생성한다. ORB는 BRIEF를 기반으로 descriptor를 생성하므로 이진 descriptor 형태이기 때문에 계산량이 적다는 장점이 있다. Feature는 사물 또는 이미지의 특성을 대표하므로 scale 변화, 회전, 조명 변화, 밝기 변화 등과 무관하게 matching될 수 있어야 한다. 이를 위해 기존 feature extraction 알고리즘들이 다양한 방법을 사용하고 있지만, 모든 상황에 유연한 알고리즘을 만드는 것은 어렵다.

최근 딥러닝 분야가 급속도로 성장함으로써, 영상 처리 및 컴퓨터 비전에도 딥러닝을 접목하는 연구가 활발히 진



<그림 2> Feature Extraction, Matching을 이용한 Image Stitching

행되고 있다. Feature extraction이나 feature matching 분야에서도 신경망 구조를 이용하여 성능을 높이고자 하는 연구들이 계속해서 진행되고 있다. 기존 방법으로는 무수히 많은 예외적인 상황을 모두 고려하기 어려우므로, 이를 학습을 통해 극복하려는 것이다. 그중 많이 사용되고 있는 기술로 SuperPoint[7], LoFTR(Local Feature TRansformer)[8] 등이 있다. SuperPoint에서는 CNN을 이용하여 기존 feature extraction처럼 feature의 위치를 탐지하고 feature의 descriptor를 생성한다. 반면에 LoFTR은 따로 feature의 위치를 탐지하는 과정 없이 CNN에서 나온 feature map간의 matching을 수행한다. II장과 III장에서, 이 두 가지 방법에 대하여 자세히 알아본다.

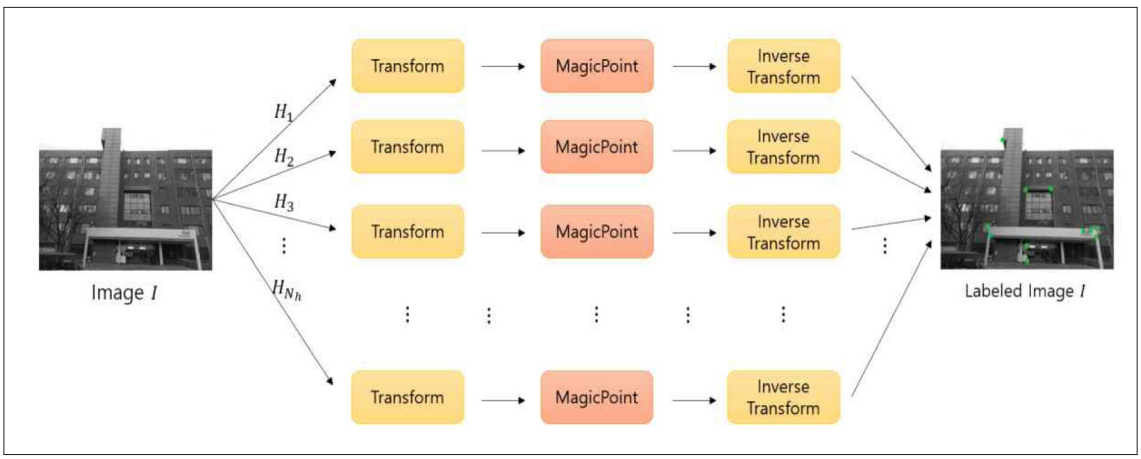
II. SuperPoint[7]

SuperPoint는 크게 3개의 모듈로 나눌 수 있다. 먼저, 학습을 위해 training dataset에 labeling을 수행하는 모듈이 있다. 그리고 이렇게 생성한 labeling된 data를 이용하여 feature의 위치를 찾는 모듈과, feature의 descriptor를 생성하는 모듈이 있다.

1. Training dataset에 labeling

Supervised 학습을 위해서는 labeling된 training dataset이 필요하다. 하지만 feature의 위치를 사람이 직접 일일이 labeling하기에는 그 수가 너무 많고, feature의 위치를 정의하는 것 또한 모호함이 있다. 따라서 [7]에서는 labeling을 위해 MagicPoint라는 Neural Network 구조를 제안하였다. MagicPoint는 convolution layer를 기반으로 이루어져 있으며, 흑백 이미지를 입력 받아서, '각 픽셀마다 feature가 위치할 확률값'을 가지는 확률 map을 출력한다. 처음에는 실제로 사용할 training dataset에 label이 없으므로, 임의로 간단한 도형이 그려져 있는 dataset을 생성한다. 이 도형들의 꼭짓점 위치를 알 수 있으므로, 이 꼭짓점 위치를 feature의 위치로 labeling할 수 있다. 일단 이 labeling된 도형 dataset으로 MagicPoint를 학습시킨다. 이렇게 학습을 수행한 MagicPoint는 단순한 이미지에 대해 feature의 위치를 제법 잘 찾을 수 있지만, 실제 복잡한 dataset에 대한 feature 위치 탐지 정확도는 떨어진다. [7]에서는 MagicPoint에 'homography adaptation'이라는 방법을 적용하여 실제 label이 없는 이미지에 labeling을 부여하는 방법을 제안하였다.

〈그림 3〉은 homography adaptation 과정을 나타낸



<그림 3> Homography Adaptation

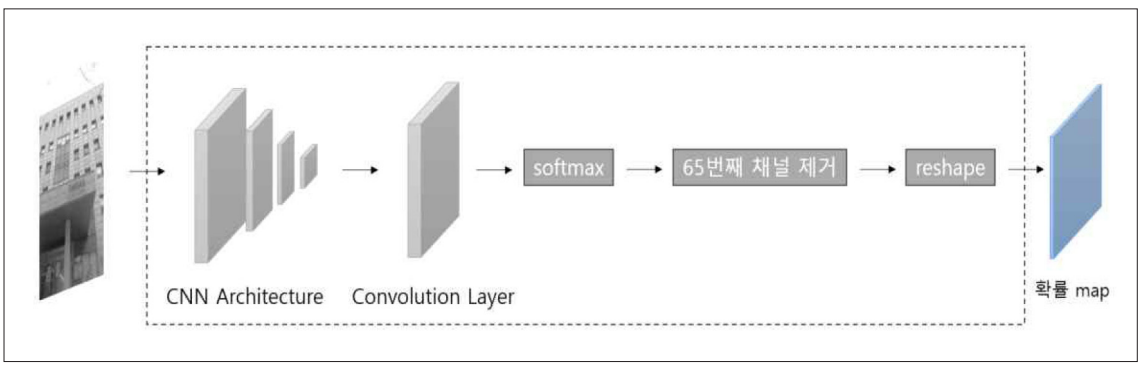
것이다. 먼저, 임의로 homography 행렬들 $H_1, H_2, ..., H_{N_h}$을 생성하고, label이 없는 이미지에 적용하여 여러 개의 homography 변환 이미지를 생성한다. 이 homography 변환 이미지들을 각자 MagicPoint에 넣어서 나름대로 feature의 위치를 추정한다. 앞서 적용했던 homography 행렬을 이용하여 역변환하고, 역변환된 이미지들에서 검출된 feature 위치를 통합하여 labeling을 수행한다.

2. Feature 위치 검출

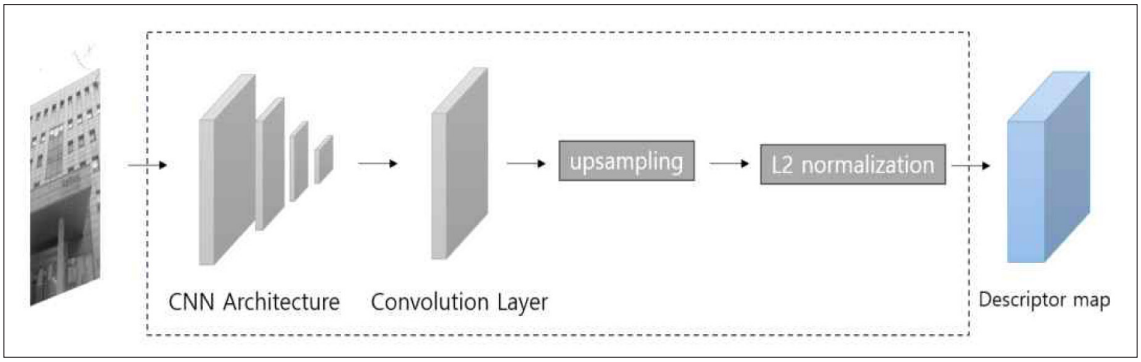
MagicPoint로 구한 labeling된 training dataset을 이용하여 본격적으로 feature의 위치를 찾는 모듈을 학습한다. 〈그림 4〉는 이 모듈의 pipeline을 간단히 나타낸 것이다. 먼저, $H \times W \times 1$ 크기의 이미지를 VGG[9]와 같은 CNN architecture에 입력한다. 이때, CNN architecture에서 max pooling을 3번 수행하여 출력 feature map의 가로·세로 크기가 각각 원본 입력 이미지의 $\frac{1}{8}$이 되도록 한다. 즉, 입력 이미지의 $8 \times 8 = 64$개의 픽셀 정보를 하나의 벡터로 표현한 꼴이 된다. 이후 feature map을 convolution layer에 넣어서 channel 수를 65가 되게 한다. 이를 수행하면 64개의 픽셀정보를 $1 \times 1 \times 65$ 크기의 벡터로 나타내는 의미가 된다. 그리고 이 벡터 단위로 softmax 연산을 취한다. 각 픽셀마다 '해당 픽셀이 feature의 위치일 확률'을 학습할 것이므로, 65개의 channel 값은 각각 '각 픽셀이 feature의 위치일 확률'과 '64개의 픽셀 중에 feature가 존재하지 않을 확률'을 의미한다. 이제 '64개의 픽셀 중에 feature가 존재하지 않을 확률'을 의미하는 65번째 채널값을 버리면 feature map의 channel 수가 64로 줄어들고, 이를 원본 이미지 크기대로 reshape를 수행한다. 최종적으로 원본 입력 이미지와 크기가 같고, 각 픽셀이 feature의 위치일 확률값을 가지는 확률 map을 얻을 수 있다.

3. Feature descriptor 생성

Feature간의 matching을 수행하기 위해서는 feature의 위치뿐만 아니라, feature마다의 descriptor를 생성해야 한다. 이번에는 feature descriptor를 생성하는 모듈에 대해서 설명한다. 〈그림 5〉에서 pipeline을 나타내고 있다. 앞에서 VGG와 같은 CNN architecture로부터 얻었던 feature map을 그대로 사용한다. 즉, 가로·세로 크기가 원본의 $\frac{1}{8}$인 feature map을 사용한다. 이 feature map을 convolution layer에 넣어주는데, 이번에는 channel 수가 D차원이 되도록 한다. 여기서 D는 생성하고자 하는 descriptor의 차원수를 의미한다. Channel 수를 그대로 유지하고, 가로·세로 크기가 원본 이미지와 같아지도록 interpolation을 적용한 upsampling을 수행한다. 이렇게 얻은 feature map은 $H \times W \times D$ 크기를 가지게 되고 각 D



<그림 4> SuperPoint의 Feature 위치 탐지 Pipeline



<그림 5> SuperPoint의 Feature Descriptor 생성 Pipeline

차원 벡터에 대해 L2 normalization을 수행하여 각 픽셀에 대응되는 descriptor를 얻을 수 있다.

4. Loss함수

학습 시에는 이미지 pair를 입력한다. Training dataset의 한 이미지에 대해 임의의 homography 변환을 취하여 다른 이미지를 생성하고 두 이미지를 하나의 pair로 묶는다. Pair가 된 두 이미지를 각각 SuperPoint에 넣어서 feature의 위치와 descriptor를 생성한다. SuperPoint는 올바른 feature의 위치를 찾고 올바른 descriptor를 생성하기 위한 모델이므로 loss함수 또한 크게 두 가지 항으로 나뉜다. 첫 번째 항은 feature 위치에 대한 loss항이고, 두 번째 항은 feature descriptor에 대한 loss항이다. 먼저, pair를 이루는 두 이미지 모두 feature 위치에 대한 ground truth를 가지고 있으므로 이를 이용하여 cross entropy loss함수를 적용하여 feature 위치에 대한 loss를 계산할 수 있다.

두 이미지간의 homography 행렬을 알고 있으므로 올바른 matching을 수행하였을 때는 descriptor의 내적값이 클수록 loss값을 작게, 잘못된 matching을 수행하였을 때는 descriptor의 내적값이 작을수록 loss값을 작게 설정함으로써 feature descriptor를 학습할 수 있다.

III. LoFTR[8]

이번 장에서는 LoFTR에 대해 설명한다. LoFTR은 feature의 위치를 따로 찾는 SuperPoint 및 기존 알고리즘들과 달리, CNN architecture의 출력으로 나온 feature map간의 matching을 수행한다. 직접 탐색한 feature 위치에서만 matching을 수행하면 sparse한 matching 결과를 얻게 되는데, LoFTR과 같이 feature map 단위로 matching을 수행하면 dense한 matching을 수행할 수 있다. LoFTR은 크게 4가지 모듈로 나눠서 설명한다. 첫 번째 모듈은 CNN architecture로부터 feature map을 얻는 과정이다. 두 번째 모듈에서 이 feature map에 self attention과 cross attention을 적용한다. 그리고 세 번째 모듈에서 coarse한 matching을 수행하고 이 matching 정보를 기반으로 네 번째 모듈에서 더 정밀한 matching을 수행한다.

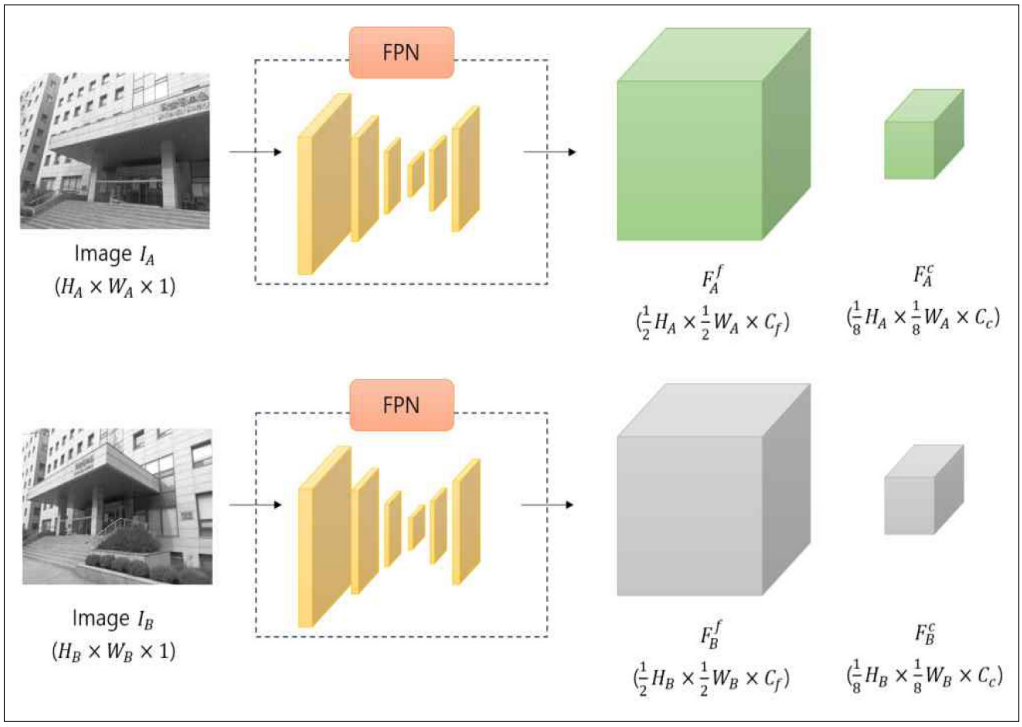
1. Feature map 얻기

LoFTR도 SuperPoint처럼 이미지 pair를 입력하여 학습을 진행한다. 먼저, 두 이미지 I_A와 I_B를 FPN(Feature Pyramid Network)[10]에 넣는다. FPN에서는 입력 이미지의 $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, ... 등의 크기를 가지는 feature map을 얻을 수 있는데, 〈그림 6〉처럼 그중에서 입력 이미지의 $\frac{1}{8}$ 크기를 가지는 feature map F_A^C, F_B^C와 $\frac{1}{2}$ 크기를

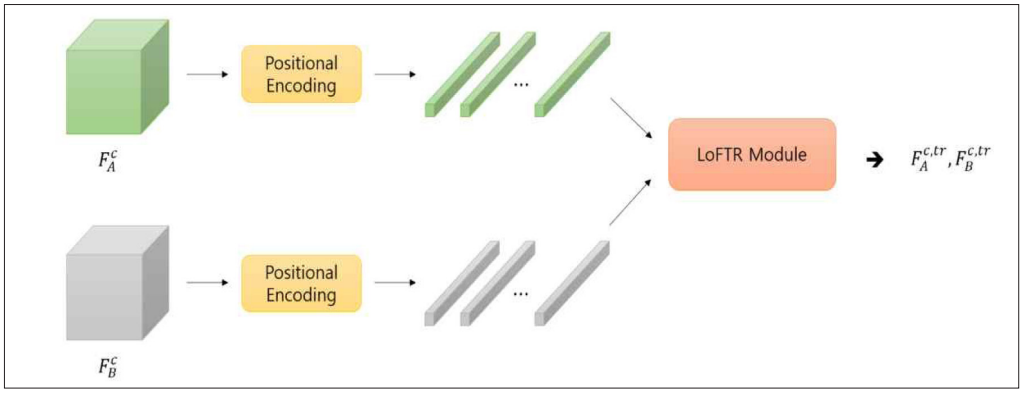
가지는 feature map인 F_A^f, F_B^f를 사용할 것이다. 이때 두 이미지 I_A와 I_B의 크기를 각각 $H_A \times W_A \times 1$, $H_B \times W_B \times 1$라고 하고, channel 크기를 C_C라고 하면 F_A^c와 F_B^c의 크기는 각각 $\frac{1}{8}H_A \times \frac{1}{8}W_A \times C_C$, $\frac{1}{8}H_B \times \frac{1}{8}W_B \times C_C$이 된다. 마찬가지로 F_A^f와 F_B^f의 channel 크기를 C_f라고 하면, F_A^f와 F_B^f의 크기는 각각 $\frac{1}{2}H_A \times \frac{1}{2}W_A \times C_f$, $\frac{1}{2}H_B \times \frac{1}{2}W_B \times C_f$이다.

2. LoFTR 모듈

먼저 앞서 구한 입력 이미지의 $\frac{1}{8}$크기를 가지는 feature map F_A^c, F_B^c를 사용한다. 이 두 feature map에 positional encoding을 수행하고 각 C_C차원 벡터를 LoFTR 모듈에 입력으로 넣는다. LoFTR 모듈은 transformer의 원리를



<그림 6> LoFTR에서 Feature Map 얻기



<그림 7> LoFTR 모듈

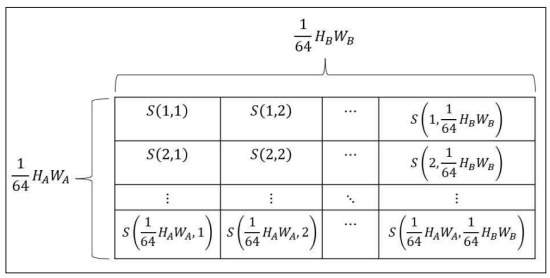
기반으로 하며, self attention layer와 cross attention layer를 교대로 N_C번 수행한다. Attention layer에서 Query와 Key, Value를 모두 같은 feature map으로부터 얻었으면 self attention이고 다른 feature map으로부터 얻었으면 cross attention이다. LoFTR 모듈의 출력 또한 각각 $\frac{1}{8}H_A \times \frac{1}{8}W_A \times C_C$, $\frac{1}{8}H_B \times \frac{1}{8}W_B \times C_C$의 크기를 가지는데, 이를 $F_A^{c,tr}$, $F_B^{c,tr}$이라고 둔다.

3. Coarse level matching

LoFTR 모듈에서 출력으로 나온 $F_A^{c,tr}$, $F_B^{c,tr}$의 각 벡터들을 matching하는 모듈이다. 이미지 I_A의 $F_A^{c,tr}$로부터 $\frac{1}{64}H_AW_A$개의 C_C차원 벡터를 얻을 수 있고, 이미지 I_B의 $F_B^{c,tr}$로부터 $\frac{1}{64}H_BW_B$개의 C_C차원 벡터를 얻을 수 있으므로 이 벡터간의 matching을 수행한다. 먼저 $F_A^{c,tr}$의 i번째 벡터를 $F_A^{c,tr}(i)$라고 하고 $F_B^{c,tr}$의 j번째 벡터를 $F_B^{c,tr}(j)$라고 할 때, 각 $F_A^{c,tr}(i)$와 $F_B^{c,tr}(j)$간의 내적 연산을 통하여 score map $S(i,j)$를 계산한다. 이를 식으로 나타내면 식 (1)과 같고, $S(i,j)$의 구조는 〈그림 8〉과 같다.

$$S(i,j) = \frac{1}{\tau}(F_A^{c,tr}(i) \cdot F_B^{c,tr}(j)) \quad (1)$$

즉, $S(i,j)$는 $F_A^{c,tr}(i)$와 $F_B^{c,tr}(j)$ 간의 유사도를 의미한다. $S(i,j)$의 각 i번째 행마다 softmax 연산을 취한 값을 $softmax_i(i,j)$라고 하고, $S(i,j)$의 각 j번째 열마다 softmax 연산을 취한 값을 $softmax_j(i,j)$라고 하자. 이 식 (2)와 같이 두 값을 곱하면 $F_A^{c,tr}(i)$와 $F_A^{c,tr}(j)$가 matching될 확률을 의미하는 $P_c(i,j)$를 구할 수 있다.



<그림 8> $S(i,j)$의 구조

$$P_c(i,j) = softmax_i(i,j)softmax_j(i,j) \quad (2)$$

모든 각 i에 대해서 $P_c(i,j)$값이 가장 높은 j를 찾으면 $F_A^{c,tr}(i)$와 $F_B^{c,tr}(j)$간의 matching쌍을 구할 수 있다. 단, 이 때 임계값 θ_C에 대하여 $P_C(i,j)<\theta_C$이면 matching 확률이 너무 낮다고 판단하여 matching쌍에 포함시키지 않는다. 이렇게 구한 matching쌍들을 집합 M_C라고 둔다.

4. Fine level matching

앞서 구한 matching쌍 M_C는 해상도가 낮았던 F_A^c, F_B^c에서의 matching 결과이므로 이번에는 보다 해상도가 높은 feature map에서 matching을 시도한다. 앞에서 구했던 입력 이미지의 $\frac{1}{2}$크기를 가지는 feature map F_A^f, F_B^f를 사용한다. 먼저 F_A^f와 F_B^f에서 coarse level matching으로 얻은 M_C의 각 (i,j)쌍 위치에 해당하는 $w \times w$크기의 윈도우를 추출한다. 이 두 윈도우를 $F_A^f(i)$, $F_B^f(j)$라고 하고 $F_A^f(i)$와 $F_B^f(j)$를 LoFTR 모듈에 넣어서 나온 값을 $F_A^{f,tr}(i)$와 $F_B^{f,tr}(j)$이라고 한다. $F_A^{f,tr}(i)$의 중심 벡터와 $F_B^{f,tr}(j)$의 각 모든 벡터간의 correlation을 구하고 softmax를 취하여 '$F_A^{f,tr}(i)$의 중심 벡터와 $F_B^{f,tr}(j)$의 각 벡터간의 matching 확률'을 의미하는 heatmap을 얻는다. 이 확률분포의 expectation을 구하여 $F_B^{f,tr}(j)$ 내부에서 $F_A^{f,tr}(i)$의 중심 벡터와 matching되는 위치 j'를 구할 수 있다. 이렇게 얻은 (i,j')쌍들의 집합을 M_f라고 한다.

5. Loss함수

LoFTR은 ScanNet[11], MegaDepth[12]처럼 camera pose와 depth map data가 주어져 있는 training set을 사용한다. 따라서 이미지 pair간의 warping 관계를 알 수 있고, 모든 픽셀에서 올바른 matching관계를 알 수 있

<표 1> Detector based model들간의 AUC 비교

Threshold \ Model	R2D2+NN	DISK+NN	SuperPoint+ SuperGlue
3pixel	50.6	52.3	53.9
5pixel	63.9	64.9	68.3
10pixel	76.8	78.9	81.7

<표 2> Detector free model들간의 AUC 비교

Threshold \ Model	Sparse-NCNet	DRC-Net	LoFTR
3pixel	48.9	50.6	65.9
5pixel	54.2	56.2	75.6
10pixel	67.1	68.3	84.6

다. 이 정보를 ground truth로 하여 loss함수를 설계한다. LoFTR의 loss함수도 크게 두 항으로 나뉜다. 첫 번째 항은 coarse level에서의 loss를 계산한다. Coarse level matching 수행 과정에서 matching 확률을 의미하는 $P_c(i,j)$를 계산하였었는데, ground truth matching에 해당하는 확률값이 높을수록 loss가 작아지게끔 학습한다. 나머지 항은 fine level에서의 loss로, M_f의 각 j'와 ground truth matching 픽셀 j'_{gt}간의 유클리디안 거리가 짧을수록 loss값이 작아지게 설계한다.

IV. 알고리즘 성능 비교

본 장에서는 SuperPoint 및 LoFTR과 기타 딥러닝 기반 feature extraction, feature matching 알고리즘에서의 성능을 비교한다. 해당 실험 결과는 [8]에서 실행한 실험 결과를 정리한 것이다.

Test dataset으로는 HPatches[13] dataset이 사용되었고, 평가 방식으로는 homography estimation을 사용하였다. Homography 변환 관계를 가지는 이미지 pair들에 대하여 각 feature matching 방법을 사용하여 RANSAC으로 homography 행렬을 추정한다. 이렇게 추정한 homography 행렬을 이용하여 이미지 pair 중 한 이미지를 다른 이미지로 homography 변환을 수행할 수 있는데, 이때 이미지 꼭짓점들의 reprojection 에러를 구한다. 에러값이 threshold보다 적으면 feature matching이 잘 수행되었다고 판단하고 정답으로 처리한다. 이를 반복하여 정답률을 구할 수 있다. 〈표 1〉과 〈표 2〉는 각 몇 가지 모델에 대하여 threshold를 각각 3pixel, 5pixel, 10pixel까지 설정하여 AUC를 정리한 것이다. 〈표 1〉은 앞서 설명했던 SuperPoint를 포함한 'feature의 위치를 별도로 탐지하는' model들의 성능을 비교한 것이다. R2D2[14]와 DISK[15]에서는 matching 방법으로 NN을 사용하였고, SuperPoint에서는 SuperGlue[16]라는 matching 방법을 사용하였다. 반면 〈표 2〉는 LoFTR처럼 'feature의 위치를 별도로 탐지하는 모듈이 없는' model들의 성능을 비교한 것이다. Sparse-NCNet[17], DRC-Net[18]이 이에 해당한다. 이 결과들을 보면 SuperPoint와 LoFTR이 다른 model들에 비해 좋은 성능을 가지는 것을 확인할 수 있다.

V. 결 론

본 글에서는 feature extraction 및 feature matching에 대하여 설명하였다. Feature extraction 및 feature matching은 여러 분야에서 기초적인 기술로 사용되고 있기 때문에 상당히 중요한 기술이다. 만약 제대로 된 feature matching이 수행되지 않으면 당연히 이를 활용한 기술들의 성능도 안 좋을 수밖에 없을 것이다. 따라서 기존 알고리즘의 한계를 뛰어넘고자 SuperPoint, LoFTR 등 Neural Network을 이용한 방법들이 꾸준히 연구되고 있다. 비록 feature extraction 및 feature matching에 대한 연구는 굉장히 오래되었지만, 그 중요성은 여전히 줄어들지 않고 있다.

참 고 문 헌

[1] D. G. Lowe, "Object recognition from local scale-invariant features," Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, pp. 1150-1157 vol.2, 1999, doi: 10.1109/ICCV.1999.790410.

[2] D. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis., vol. 60, no. 2, pp. 91-110, 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[3] H. Bay et al., "Speeded-up robust features (SURF)," Comput. Vision and Image Understanding, vol. 110, no. 3, pp. 346-359, 2008.

[4] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, Spain, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.

[5] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In European Conference on Computer Vision, volume 1, 2006.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In European Conference on Computer Vision, 2010.

[7] D. DeTone, T. Malisiewicz and A. Rabinovich, "SuperPoint: Self-Supervised Interest Point Detection and Description," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, USA, 2018, pp. 337-33712, doi: 10.1109/CVPRW.2018.00060.

[8] J. Sun, Z. Shen, Y. Wang, H. Bao and X. Zhou, "LoFTR: Detector-Free Local Feature Matching with Transformers," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021, pp. 8918-8927, doi: 10.1109/CVPR46437.2021.00881.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. ICLR*, 2015, doi: https://arxiv.org/abs/1409.1556.

[10] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan and Serge J. Belongie, "Feature Pyramid Networks for Object Detection," CVPR, 2017.

[11] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser and Matthias Nießner, "ScanNet: Richly-annotated 3d reconstructions of indoor scenes," CVPR, 2017.

[12] Zhengqi Li and Noah Snavely, "Megadepth: Learning single-view depth prediction from internet photos," CVPR, 2018.

[13] Vassileios Balntas, Karel Lenc, Andrea Vedaldi and Krystian Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors," CVPR, 2017.

[14] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, et al., "R2D2: repeatable and reliable detector and descriptor," NeurIPS, 2019.

[15] Michał Tyszkiewicz, Pascal Fua and Eduard Trulls, "DISK: Learning local features with policy gradient," NeurIPS, 2020.

[16] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz and Andrew Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," CVPR, 2020.

참 고 문 헌

[17] Ignacio Rocco, Relja Arandjelović and Josef Sivic, "Efficient neighbourhood consensus networks via submanifold sparse convolutions," ECCV, 2020.
[18] Xinghui Li, Kai Han, Shuda Li and Victor Prisacariu, "Dual-resolution correspondence networks," NeurIPS, 2020.
[19] J. L. Schönberger and J. -M. Frahm, "Structure-from-Motion Revisited," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 4104-4113, doi: 10.1109/CVPR.2016.445.

저 자 소 개

최 건 찬

- 2018년 ~ 현재 : 세종대학교 전자정보통신공학과 학사과정
- ORCID : https://orcid.org/0009-0004-9156-8853
- 주관심분야 : 영상 신호처리, VR

한 종 기

- 1992년 : KAIST 전기및전자공학과 공학사
- 1994년 : KAIST 전기및전자공학과 공학석사
- 1999년 : KAIST 전기및전자공학과 공학박사
- 1999년 3월 ~ 2001년 8월 : 삼성전자 DM연구소 책임연구원
- 2001년 9월 ~ 현재 : 세종대학교 전자정보통신공학과 교수
- 2008년 9월 ~ 2009년 8월 : University California San Diego (UCSD) Visiting Scholar
- ORCID : https://orcid.org/0000-0002-5036-7199
- 주관심분야 : VR, 3D 영상신호 구현, 비디오 코덱, 영상 신호처리, 정보 압축, 방송 시스템