일반논문 (Regular Paper)
방송공학회논문지 제28권 제4호, 2023년 7월 (JBE Vol.28, No.4, July 2023)
https://doi.org/10.5909/JBE.2023.28.4.457
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

촬영 환경 변화에 강인한 NetVLAD 제작 기술

최 건 찬[a)], 한 종 기[a)‡]

Robust NetVLAD Construction Technique for Circumstance Variation

Geonchan Choi[a)] and Jongki Han[a)‡]

요 약

VPR(Visual Place Recognition)은 로봇, 드론, 자율 주행, VR 등의 기술 발전과 밀접한 관련이 있는 작업이며, 따라서 최근에 중요성이 증가하고 있다. 본 논문에서는 VPR 작업 중 대표적인 연구인 NetVLAD에 대하여 소개하고, 더 나아가 이를 개선할 방법을 제안한다. NetVLAD는 각 이미지마다 그 이미지의 특성 정보를 담고 있는 global descriptor를 생성하고, 이 global descriptor들 사이의 matching을 실시하여 VPR을 수행한다. 본 논문에서는 matching되는 이미지 쌍이어도 각 이미지의 촬영 환경에 따라 global descriptor가 다르게 계산될 수 있다는 점을 고려하여 matching 과정에서의 weight를 부여한다. 이러한 과정을 통해 촬영 환경 변화에 보다 강인한 VPR을 수행할 수 있다. 본 논문에 마지막에는 기존 NetVLAD와의 성능 비교를 통해 제안하는 방법이 효과가 있음을 입증한다.

Abstract

VPR(Visual Place Recognition) is a task closely related to the development of technologies such as robots, drones, autonomous driving, and VR and has recently become more important. In this paper, we introduce NetVLAD, a representative study in VPR tasks, and propose ways to improve it. NetVLAD generates a global descriptor containing characteristic information of each image and performs VPR by matching between these global descriptors. In this paper, we give weight to the matching process by considering that global descriptors can be calculated differently depending on the circumstance of each image even if they are image pairs that match. Through this process, we can perform a more robust VPR against changes in circumstances. Finally, we prove that the proposed method is effective through performance comparison with existing NetVLAD.

Keyword : Visual Place Recognition, Visual Localization, Image Retrieval

a) 세종대학교(Sejong University)
‡ Corresponding Author : 한종기(Jongki Han)
E-mail: hjk@sejong.edu
Tel: +82-2-3408-3739
ORCID: https://orcid.org/0000-0002-5036-7199

※ This work was supported by the National Research Foundation of Korea (NRF) under Grant 2022R1F1A1071513 funded by the Korea government through the Ministry of Science and ICT (MSIT).
· Manuscript April 27, 2023; Revised June 21, 2023; Accepted June 21, 2023.

Ⅰ. 서 론

VPR(Visual Place Recognition)은 입력 영상(query image)이 database에 저장된 많은 영상들 중에 어떤 영상과 matching되는 지를 판단하여, query 영상의 위치를 추정하는 기술이다[1]. VPR은 Visual Localization 기술과 3D reconstruction, SLAM 등의 작업에도 활용될 수 있다[2]. 이러한 기술들은 최근에 자율주행, 로봇 공학, 드론, VR 및 AR 등의 분야에 관한 관심이 커짐에 따라 활발하게 연구되고 있다.

VPR 작업을 수행하기 위한 기존의 연구들에는 BoW (Bag of Words)[3], VLAD[4][5], NetVLAD[6], Patch-NetVLAD[2] 등이 있다. 이들에 대해 간단하게 설명하면 다음과 같다. 연구 논문 [3]에서는 BoW(Bag of Words)를 이미지에 적용하는 방법에 대하여 설명한다. 비교하려는 이미지들로부터 모든 SIFT[7] descriptor들을 추출하고, 이 SIFT descriptor들에 대하여 clustering을 수행한다. 각각의 이미지마다 SIFT descriptor가 어느 cluster에 얼마나 분포해 있는지를 histogram으로 나타내어, 이 histogram을 이용하여 이미지들을 matching시키는 방법이다. 이는 이미지의 모든 SIFT descriptor들을 histogram으로 함축하여 이미지의 특성을 나타낸다는 점에서 의의가 있으나 함축하는 과정에서 너무 많은 정보를 잃어버리기 때문에 정교한 matching작업을 수행하기 어렵다는 단점이 있다.

VLAD[4]는 cluster 당 SIFT descriptor들의 분포만 고려했던 BoW[3]보다 더 많은 정보를 지니는 방법이다. histogram을 구했던 BoW[3]과는 달리, VLAD에서는 VLAD descriptor라는 벡터를 구하고 이 VLAD descriptor를 이용하여 이미지 사이의 matching을 수행한다. BoW[3]과 마찬가지로 비교하고자 하는 모든 이미지에서 128차원의 SIFT descriptor를 추출하고, 이 모든 SIFT descriptor들에 대해 K-means clustering을 적용하여 K개의 cluster로 분류한다. 설명의 편의를 위해서, VLAD descriptor를 구하고자 하는 하나의 이미지만을 생각해보자. 이 이미지의 SIFT descriptor들 또한 K개의 cluster로 분류되어 있다. 각각의 cluster에서 '이 이미지의 각 SIFT descriptor'와 '그 SIFT descriptor가 속해있는 cluster의 centroid 벡터'간의 잔차 벡터들을 구한다. 그리고 각각의 cluster에서 해당 cluster에서 구한 잔차 벡터들을 모두 더한다. 이러면 각 cluster마다 하나의 벡터가 나오게 되고, 이 벡터들의 차원은 SIFT descriptor와 같이 128차원이 된다. 즉, 128차원의 벡터를 K개 얻을 수 있다. 이 벡터들을 일렬로 나열하여 하나로 묶은 $128\times K$ 차원의 벡터를 VLAD descriptor라고 한다. VLAD는 단순히 cluster마다 SIFT descriptor의 개수만을 고려했던 BoW[3]과는 달리 SIFT descriptor 벡터들이 가지는 값을 반영하여 descriptor를 만들었기 때문에 그 이미지에 대한 더 많은 정보가 담겨있다는 장점이 있다. 하지만 이 역시도 하나의 descriptor로 통합하는 과정에서 제법 많은 정보를 잃어버린다. 또한 BoW[3]과 마찬가지로 SIFT descriptor를 기반으로 하기 때문에 SIFT descriptor의 한계점을 같이 가지고 간다는 문제점이 있다.

NetVLAD[6]는 SIFT descriptor의 한계를 Neural Net을 통해 극복하고자 제안한 방법이다. 먼저, 각 이미지를 VGG-16[8] 등의 CNN architecture에 넣어서 $H\times W\times D$ 크기의 feature map을 만들고, SIFT descriptor 대신 이 feature map의 각 D차원 벡터들을 사용한다.(이 벡터들을 local descriptor라고 한다.) 이 local descriptor를 이용하여 VLAD descriptor를 만드는데, NetVLAD[6]에서는 clustering 및 VLAD descriptor 생성 과정 또한 Neural Net으로 구현하는 방안을 제안한다. VLAD descriptor를 계산하는 공식을 학습 가능하도록 변형하고, 이를 계산하는 Neral Net layer를 'NetVLAD layer'라고 칭한다. 따라서 앞의 CNN architecture와 NetVLAD layer를 통합하면 end-to-end 방식으로 NetVLAD descriptor를 생성할 수 있다. NetVLAD는 local descriptor를 만드는 과정과, clustering 과정을 학습할 수 있으므로 보다 다양한 상황에서 유연하게 작동할 수 있다. 하지만 기본적으로 VLAD descriptor를 생성하는 과정과 유사하기 때문에 이로부터 발생하는 정보 손실은 여전히 해결해야 할 문제이다.

Patch-NetVLAD[2]는 이미지의 NetVLAD descriptor뿐만 아니라 CNN architecture에서 얻은 feature map을 patch 단위로 쪼개어 얻은 Patch-NetVLAD descriptor를 함께 사용한다. 먼저, database의 모든 이미지에 대하여 NetVLAD descriptor를 생성하고, query 이미지의 NetVLAD descriptor도 생성한다. 유클리디안 거리를 기준으로 database의 NetVLAD descriptor들 중 query 이미지의 NetVLAD de-

scriptor와 가까운 이미지들을 일부 추려낸다. 이렇게 얻어낸 이미지들은 query 이미지와 matching될 가능성이 높은 후보 이미지들이다. 이제 후보 이미지들과 query 이미지에 대하여 feature map의 각 patch마다 Patch-NetVLAD descriptor를 생성하여 정밀하게 최종 matching 작업을 수행한다. 이 방법은 높은 정확도를 달성했지만, patch마다 생성한 descriptor를 저장해야 하고, matching 시간이 오래 걸린다는 문제점이 있다.

VPR에서는 환경 변화에 강인하도록 하는 것이 중요하다. 촬영 장소가 같더라도 촬영 시점, 촬영 시간 등은 다르기 때문에 조명 변화, 움직이는 object의 존재, 시점 변화 등의 환경이 변하더라도 같은 장소임을 식별할 수 있어야 한다. Neural Net의 발전으로 이러한 환경 변화에 더욱 유연해지고 있으나 아직 완전하다고는 볼 수 없다. 본 논문에서는 VPR 작업에서 효과적인 성능을 보였던 NetVLAD를 기반으로 이러한 환경 변화에 더 강인하게 하는 방법을 제안한다. NetVLAD descriptor에서 환경 변화에 취약한 원소들을 판별하고 이에 따라 matching과정에서 가중치를 달리함으로써 환경 변화에 취약한 원소들이 matching에 미치는 영향을 줄이는 방법을 제안한다.

본 논문의 구성을 다음과 같다. Ⅱ절에서는 VPR의 개념과 과정에 대하여 자세히 설명한다. Ⅲ절에서는 본 논문에서 제안하는 방법의 뼈대가 되는 NetVLAD descriptor 추출 과정에 대하여 상세히 다룬다. Ⅳ절에서는 제안하는 방법에 대하여 설명하고, Ⅴ절에서 실험을 통해 기존 NetVLAD에 비해 제안하는 방법의 정확도가 높음을 확인한다. 마지막으로 Ⅵ절에서는 본 논문에서 제안하는 방법의 의의를 되짚어보고, 앞으로의 연구 방향을 서술한다.

Ⅱ. Visual Place Recognition

VPR은 그림 1과 같이 특정 query 사진을 입력으로 넣었을 때, query 사진과 같은 장소에서 찍은 사진을 matching 시켜 찾아주는 작업을 의미한다.

그림 1을 보면, query 이미지가 있고 query와 matching되는 이미지 후보들이 있는 database가 있다. VPR을 수행하기 위해서는 먼저 query 이미지와 database의 각 모든 이미지들에 대하여 global descriptor를 구해야 한다. global descriptor는 각 이미지의 특성을 반영하고 있는 벡터이다. 이제 database 이미지들의 각 global descriptor들 중 어떤 global descriptor가 query 이미지의 global descriptor와 matching되는지를 확인한다. 주로 유클리디안 거리가 가장 짧은 global descriptor와 matching한다. 그림 1에서는 query 이미지의 global descriptor와 Image2의 global descriptor가 matching되었다. 이러한 작업을 통해 query 이미

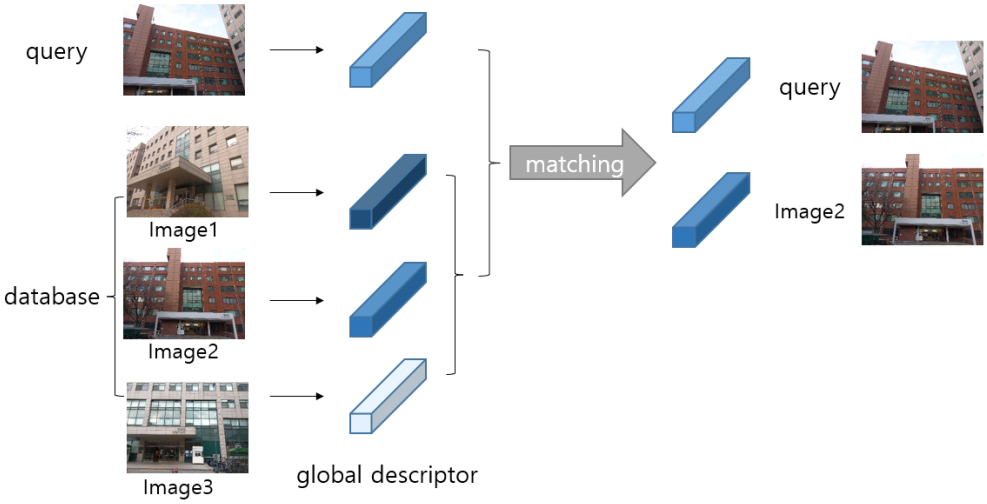

그림 1. VPR의 개념
Fig. 1. Concept of VPR

지와 같은 장소에서 촬영한 이미지를 찾을 수 있으므로 query 이미지가 어디서 촬영되었는지 알 수 있다. VLAD나 NetVLAD 같은 방법들은 VPR 작업을 잘 수행할 수 있는 global descriptor를 구하는 기술 중 하나이다.

Ⅲ. 기존 NetVLAD

이 장에서는 본 논문에서 제안하는 방법의 기초가 되는 기존의 NetVLAD[6]에 대하여 설명한다.

1. Local Descriptor 추출

그림 2의 'CNN architecture' 부분을 이용하여 local descriptor를 추출하는 과정을 설명한다. NetVLAD descriptor의 기반이 되는 local descriptor를 얻기 위해 NetVLAD descriptor를 구하고자 하는 Image를 CNN architecture에 넣어서 $H\times W\times D$ 크기의 feature map을 얻는다. CNN architecture로는 주로 AlexNet[9], VGG-16[8], ResNet[10] 등을 사용한다. 이렇게 얻은 feature map의 각 D차원 벡터들을 local descriptor라고 칭한다. 이미지 1장당 $H\times W=N$개의 local descriptor를 얻게 되며, i번째 local descriptor를 $\boldsymbol{x}_i$라고 한다.

2. VLAD Descriptor 추출 방법

NetVLAD의 과정 및 원리를 파악하기 위해서는 먼저 VLAD에 대해 알아야 한다. 원래 VLAD descriptor는 SIFT descriptor를 사용했지만, NetVLAD에서는 SIFT descriptor 대신 1절에서 설명한 local descriptor $\boldsymbol{x}_i$를 사용한다. 따라서, 이 절에서는 local descriptor $\boldsymbol{x}_i$를 이용한 VLAD descriptor 계산 방법을 설명한다.

먼저, 비교하고자 하는 모든 이미지들에 대하여 1절에서 설명한 방법을 사용하여 local descriptor들을 구한다. 이렇게 얻은 모든 local descriptor에 대해 k-means clustering을 실행하여 K개의 cluster로 분류한다. 이제 VLAD descriptor를 얻고자 하는 하나의 이미지를 생각한다. 이 이미지 역시 $H\times W=N$개의 D차원 local descriptor $\boldsymbol{x}_i$를 가지고 있으며, 각 $\boldsymbol{x}_i$들은 K개의 cluster에 각각 분류되어 있다. 각 k번째 cluster를 C_k라고 하고, k번째 cluster의 centroid를 $\boldsymbol{c}_k$라고 한다. 이제 k번째 cluster C_k에서 '모든 각 C_k에 속하는 $\boldsymbol{x}_i$'와 $\boldsymbol{c}_k$간의 차이 벡터들을 구한다. 그리고 이 차이 벡터들을 모두 더하면 벡터 하나를 얻을 수 있고, $\boldsymbol{x}_i$가 D차원이었으므로 이렇게 구한 벡터 역시 D차원의 크기를 가진다. 이를 모든 cluster에 대하여 반복하면 D차원의 벡터를 K개 얻을 수 있다. 이 K개의 벡터들을 일렬로 나열하여 만든 $K\times D$ 크기의 벡터를 VLAD descriptor라고 한다.

편의상 앞서 설명한 VLAD descriptor $\boldsymbol{V}_{\boldsymbol{VLAD}}$를 계산하는 과정을 하나의 식으로 나타내고자 한다. 이를 위해 a_k라는 변수를 도입한다. $a_k(\boldsymbol{x}_i)$는 $\boldsymbol{x}_i$가 k번째 cluster에 속했는지에 대한 여부를 반환하는 값으로, $\boldsymbol{x}_i$가 k번째 cluster에 속해있으면 $a_k(\boldsymbol{x}_i)=1$이고 $\boldsymbol{x}_i$가 k번째 cluster에 속해있지 않으면 $a_k(\boldsymbol{x}_i)=0$이다. $a_k(\boldsymbol{x}_i)$를 이용하여 $\boldsymbol{V}_{\boldsymbol{VLAD}}$를 구하는 공식을 나타내면 다음과 같다.

$$V_{VLAD}(j,k)=\sum_{i=1}^{N}a_k(\boldsymbol{x}_i)(x_i(j)-c_k(j)) \tag{1}$$

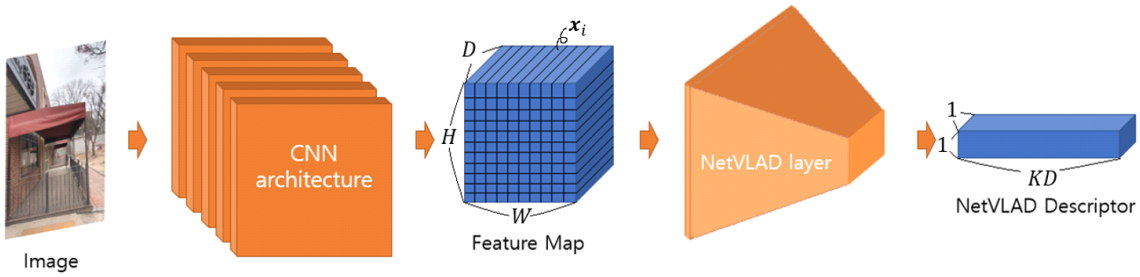

그림 2. NetVLAD 개략도
Fig. 2. NetVLAD Schematic

이때, $x_i(j)$는 $\boldsymbol{x_i}$의 j번째 원소값이며, $c_k(j)$는 $\boldsymbol{c_k}$의 j번째 원소값을 의미한다.

3. NetVLAD Descriptor 추출

앞 절에서 설명한 VLAD descriptor의 추출 과정을 기반으로, 본 절에서는 NetVLAD descriptor 계산 방법을 설명한다. 그림 2의 'NetVLAD layer'에서 이 계산을 수행한다.

NetVLAD descriptor는 기존 VLAD descriptor 계산 방식을 Neural Net으로 구현하여 얻은 global descriptor이다. 따라서, 기본적으로 식 (1)을 따라서 계산해야 한다. 즉, VLAD descriptor 계산 과정을 Neural Net으로 구현하기 위해서는 식 (1)을 학습가능한 형태로 만들어야 한다. 특히, back propagation을 위해서는 식 (1)이 미분 가능해야 할 것이다. 하지만 식 (1)의 $a_k(\boldsymbol{x_i})$는 $\boldsymbol{x_i}$가 k번째 cluster에 속해있는지에 대한 여부를 알려주는 변수로, 0 또는 1의 값을 가지는 불연속변수이므로 미분 불가능하다. 따라서 NetVLAD에서는 이러한 문제점을 해결하기 위해 $a_k(\boldsymbol{x_i})$를 연속으로 만들어 미분 가능한 형태로 변환하는 방법을 제안한다. 구별을 위해 이렇게 변환한 $a_k(\boldsymbol{x_i})$를 $\overline{a_k}(\boldsymbol{x_i})$라고 한다. 0 또는 1 두 가지 값만 가지는 $a_k(\boldsymbol{x_i})$와는 달리, $\overline{a_k}(\boldsymbol{x_i})$는 0~1 사이의 연속된 값을 가지도록 한다. 기존에 $\boldsymbol{x_i}$가 k번째 cluster에 속해있으면 $a_k(\boldsymbol{x_i})$가 1값을 가졌으므로, $\boldsymbol{x_i}$가 k번째 cluster에 속해있을 가능성이 클수록 $\overline{a_k}(\boldsymbol{x_i})$가 1에 가까운 값을 가지도록 설계한다. 반대로, $\boldsymbol{x_i}$가 k번째 cluster에 속해있을 가능성이 작으면 $\overline{a_k}(\boldsymbol{x_i})$가 0에 가까운 값을 가지도록 한다. $\boldsymbol{x_i}$가 'k번째 cluster를 제외한 나머지 cluster들의 centroid들'보다 'k번째 cluster의 centroid'와 가까우면 $\boldsymbol{x_i}$가 k번째 cluster에 속해있을 확률이 높다고 취급한다. 따라서, $\boldsymbol{x_i}$와 각 cluster의 centroid들과의 거리를 기준으로 하는 softmax 함수를 사용한다.

$$\overline{a_k}(\boldsymbol{x_i}) = \frac{e^{-\alpha \| \boldsymbol{x_i}-\boldsymbol{c_k} \|^2}}{\sum_{l=1}^{K} e^{-\alpha \| \boldsymbol{x_i}-\boldsymbol{c_l} \|^2}} \tag{2}$$

$\boldsymbol{x_i}$가 다른 centroid들에 비해 $\boldsymbol{c_k}$와 가까우면, $\overline{a_k}(\boldsymbol{x_i})$가 1에 가까워지므로 $\overline{a_k}(\boldsymbol{x_i})$는 $\boldsymbol{x_i}$가 k번째 cluster에 속할 확률을 의미한다. 따라서, hard assignment를 의미하던 $a_k(\boldsymbol{x_i})$를 soft assignment 형태로 변환한 것이다. α는 hard한 정도를 나타내는 값으로, α가 커질수록 확률값이 더 급격하게 변한다.

식 (2)의 분자, 분모를 $e^{-\alpha \| \boldsymbol{x_i} \|^2}$로 나누고 $2\alpha \boldsymbol{c_k^T} = \boldsymbol{w_k^T}$로, $-\alpha \boldsymbol{c_k^T c_k} = b_k$로 치환하여 정리하면, 다음과 같이 정리할 수 있다.

$$\overline{a_k}(\boldsymbol{x_i}) = \frac{e^{\boldsymbol{w_k^T x_i}+b_k}}{\sum_{l=1}^{K} e^{\boldsymbol{w_l^T x_i}+b_l}} \tag{3}$$

이때, $\boldsymbol{w_k}$와 b_k를 convolution layer의 학습 가능한 파라미터로 설정함으로써, $\overline{a_k}(\boldsymbol{x_i})$를 학습할 수 있다.

이제 식 (1)의 $a_k(\boldsymbol{x_i})$를 미분 가능하고 학습 가능한 형태인 식 (3)의 $\overline{a_k}(\boldsymbol{x_i})$로 대체하면, VLAD descriptor 계산식을 Neural Net으로 구현한 NetVLAD descriptor 계산식을 얻을 수 있다.

$$V_{NetVLAD}(j,k) = \sum_{i=1}^{N} \overline{a_k}(\boldsymbol{x_i})(x_i(j)-c_k(j)) \tag{4}$$

1절에서 설명한 CNN architecture로부터 얻은 각 local descriptor $\boldsymbol{x_i}$들을 NetVLAD layer에 넣으면, NetVLAD layer에서 식 (4)를 계산하여 $\boldsymbol{V_{NetVLAD}}$를 얻을 수 있다. $\boldsymbol{V_{NetVLAD}}$는 $\boldsymbol{V_{VLAD}}$와 마찬가지로 $K \times D$ 크기의 벡터이다.

4. Loss 함수

앞서 각 이미지마다 NetVLAD descriptor를 계산하는 방법을 설명하였다. 이제, 이렇게 만든 NetVLAD descriptor가 올바르게 만들어졌는지 평가함으로써 학습을 진행해야 한다. 학습을 진행하기 위해서는 ground truth를 제공하는 dataset이 필요하다. 모든 query와 database의 NetVLAD descriptor들을 생성하고 이를 이용하여 이미지들을 matching하는데, loss 함수를 설계하려면 matching 결과가 잘 이뤄졌는지를 판단할 수 있어야 한다. matching되는 이미지

쌍을 전부 표시하면 좋겠지만 수많은 이미지에 대하여 matching 정답을 전부 labeling 하기가 불가능하므로 다른 방법을 사용한다. NetVLAD에서는 이러한 문제를 해결하기 위해 GPS 정보를 사용한다. 학습을 위한 dataset의 모든 이미지들은 GPS 정보를 가지고 있는데, 만약 두 이미지의 GPS 거리가 멀리 떨어져 있으면 그 두 개의 이미지는 실제로 matching되는 이미지라고 보기 어려울 것이다. 반대로, GPS 거리가 가까우면 두 이미지는 실제로 matching되는 이미지일 가능성이 존재한다. 즉, query 이미지를 q라고 할 때, q와 GPS 거리가 많이 먼 database 이미지들은 절대로 q와 matching되지 않으므로 이 이미지들을 'definite negative' 집단이라고 분류한다. 반대로, q와 GPS 거리가 가까운 database 이미지들은 matching될 가능성이 제법 있으므로 'potential positive' 집단이라고 분류한다.

그림 3처럼 query 이미지 q로부터 10m 이내에서 찍힌 이미지들을 'potential positive' 집단으로, q로부터 25m 바깥에서 찍힌 이미지들을 'definite negative' 집단으로 분류한다. 편의를 위해 q의 각 i번째 potential positive 이미지들을 $\{p_i^q\}$로, q의 각 j번째 definite negative 이미지들을 $\{n_j^q\}$로 칭한다.

이제, q와 $\{p_i^q\}$들, $\{n_j^q\}$들의 NetVLAD descriptor를 이용하여 loss 함수를 정의한다. 먼저, 편의상 두 NetVLAD descriptor간의 유클리디안 거리를 반환하는 함수를 정의한다. 이미지 I_1의 NetVLAD descriptor와 이미지 I_2의 NetVLAD descriptor 사이의 유클리디안 거리를 $d(I_1, I_2)$라고 한다. matching 여부를 판단하는 것은 NetVLAD descriptor 사이의 유클리디안 거리이므로, $\min_i d(q, p_i^q)$가 각 $d(q, n_j^q)$보다 작아지도록 유도하는 loss 함수를 설계한다. 즉, $\min_i d(q, p_i^q)$가 $d(q, n_j^q)$보다 충분히 작으면 loss = 0이 되도록 하고 $\min_i d(q, p_i^q)$가 $d(q, n_j^q)$보다 충분히 작지 않으면 loss가 증가하도록 한다. 따라서 loss 함수 L은 다음과 같다.

$$L = \sum_j l(\min_i d^2(q, p_i^q) + margin - d^2(q, n_j^q)) \quad (5)$$

위 식 (5)에서 $l(x)$는

$$l(x) = \max(x, 0) \quad (6)$$

이다. 위 수식에서 $\min_i d^2(q, p_i^q)$이 각 $d^2(q, n_j^q)$보다 충분히($margin$만큼) 작으면 L에 0이 더해지고, 그렇지 않으면 $\min_i d^2(q, p_i^q) + margin - d^2(q, n_j^q)$만큼 L이 증가한다. 이 loss 함수를 사용함으로써 NetVLAD descriptor끼리 matching시에 definite negative 이미지들보다 potential positive 이미지들과 matching될 가능성을 높일 수 있다.

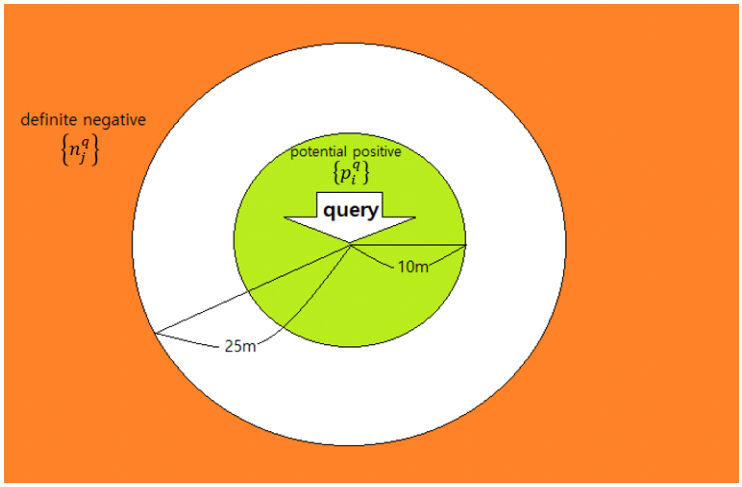

그림 3. Database 이미지 분류
Fig. 3. Classification of Database Images

Ⅳ. Proposed Algorithm

1. 제안 기술의 요약

기존 NetVLAD에서는 loss 함수(식 (5))를 구하는 과정에서 일반 유클리디안 거리를 계산하는 $d(I_1, I_2)$ 함수를 사용하였다. 하지만 본 논문에서는 일반 유클리디안 거리 대신, VPR의 환경 변화를 고려한 새로운 거리 함수를 사용할 것을 제안한다.

Query 및 database의 이미지들은 촬영 시점 변화, 조명, 계절, 사람이나 자동차와 같이 움직이는 object 등의 환경 변화에 영향을 받는다. matching 되어야 하는 이미지들도 서로 촬영된 환경이 다르므로 NetVLAD descriptor가 완전히 같은 값을 가질 수는 없다. 특히, local descriptor $\boldsymbol{x_i}$ 중 환경 변화로부터 값이 크게 달라지는 local descriptor들이 존재하기 때문에 local descriptor $\boldsymbol{x_i}$를 이용해 생성한 NetVLAD descriptor도 환경에 따라 값이 크게 달라질 수 있다. 본 논문에서는 환경 변화로 인하여 NetVLAD descriptor간의 matching이 잘못되는 경우를 간접적으로 줄이는 방법을 제안한다. 앞서 말한 'local descriptor $\boldsymbol{x_i}$들 중 환경 변화로부터 값이 크게 달라지는 local descriptor들'을 편의상 outlier라고 칭한다. outlier들을 직접 추적하여 제거하면 가장 이상적이겠지만 실제로 이들을 추적하는 것은 상당히 어렵다. 따라서 outlier를 직접 추적하는 방법이 아니라 NetVLAD descriptor의 원소들 중에서 'outlier의 영향을 크게 받을 가능성이 큰 원소'들을 찾고, 이 원소들이 NetVLAD descriptor간의 matching에 영향을 적게 미치게 하여 간접적으로 outlier의 영향을 줄이는 방안을 제안한다. NetVLAD descriptor의 각 원소들은 'local descriptor $\boldsymbol{x_i}$들을 clustering하여 얻은 각 cluster'들로부터 도출되므로, 'outlier에 취약하다고 판단되는 cluster'로부터 도출된 원소에 해당하면 NetVLAD descriptor끼리 matching하는 과정에서 적은 영향을 주도록 한다. 이를 위해서 먼저 어떤 cluster가 outlier에 취약한지를 파악하고, outlier에 취약한 정도에 따라 matching 과정에서 해당 cluster에 가중치를 부여한다.

제안하는 방법을 간단히 나타내면 그림 4와 같다.

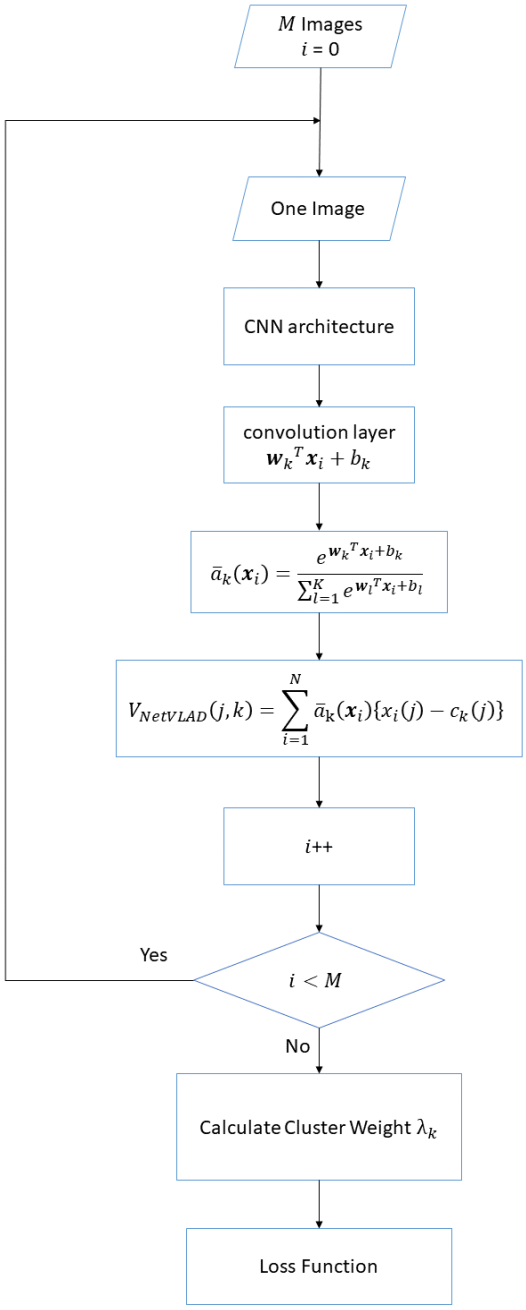

그림 4. 제안하는 방법의 Flow Chart
Fig. 4. Flow Chart of Proposed Algorithm

모든 Query 이미지들과 모든 Database 이미지들의 총 개수를 M개라고 하자. 먼저, 모든 각 이미지들에 대하여 기존

방법처럼 NetVLAD descriptor를 만들어야 한다. 각 이미지마다 CNN architecture를 통과시켜 local descriptor $\boldsymbol{x_i}$ 들을 생성한다. 그리고 이 모든 $\boldsymbol{x_i}$ 들을 convolution layer에 넣어서 $\boldsymbol{w_k^T x_i}+b_k$ 를 계산한다. 이때, convolution layer는 1×1×D차원의 kernel을 K개 가진다. 이렇게 구한 모든 $\boldsymbol{w_k^T x_i}+b_k$ 를 식 (3)에 넣어서 모든 $\overline{a_k}(\boldsymbol{x_i})$ 를 계산한다. 이제 모든 $\overline{a_k}(\boldsymbol{x_i})$ 를 식 (4)에 넣어서 해당 이미지의 NetVLAD descriptor $V_{NetVLAD}(j,k)$를 구한다. 앞서 설명했듯이, 이는 clustering과 VLAD descriptor 생성과정을 식 (3)과 식 (4)로 근사하여 Neural Net으로 구현한 과정이다. 이 과정을 모든 이미지에 대해 반복하면, 모든 각 이미지에 대한 NetVLAD descriptor를 얻게 된다. 기존 방법에서는 이 NetVLAD descriptor들로 식 (5)를 계산하여 Loss를 구하지만, 본 논문에서는 그 전에 cluster weight를 계산하고, 이를 이용한 새로운 Loss 함수를 계산함으로써 학습을 진행한다.

2. Cluster Weight 계산

본 논문에서는 'cluster의 원소 개수'를 기반으로 outlier에 취약한 정도를 추정한다. 원소의 개수가 적은 cluster로부터 계산된 잔차 벡터 합은 소수 outlier에 의해 비교적 영향을 크게 받기 때문이다. 그림 5를 보면, cluster (a)는 6개의 local descriptor $\boldsymbol{x_i}$ 를 가지고 있고 cluster (b)는 2개의 local descriptor $\boldsymbol{x_i}$ 를 가지고 있다. 식 (1)에 따르면, cluster (a)는 6개의 잔차 벡터들을 더하므로 outlier 잔차 벡터의 영향이 상대적으로 적을 것이다. 반대로, 2개의 잔차 벡터를 더해야 하는 cluster (b)는 outlier 잔차 벡터의 영향이 cluster (a)에 비하여 상당히 클 것이다.

이때, 고려해야 할 것은 식 (1)을 식 (4)로 바꾸는 과정에서 hard assignment를 soft assignment 형태로 바꿨다는 점이다. 즉, cluster간의 경계가 명확한 hard clustering의 형태가 아니기 때문에 직접 원소의 개수를 세기는 어렵다. 따라서 원소의 개수를 직접세는 대신에 확률을 나타내는 식 (3)의 $\overline{a_k}(\boldsymbol{x_i})$ 를 이용한다. 앞서 언급했듯이 $\overline{a_k}(\boldsymbol{x_i})$ 는 $\boldsymbol{x_i}$ 가 k번째 cluster에 속할 확률을 의미하므로, 각각의 cluster마다 $\overline{a_k}(\boldsymbol{x_i})$ 의 누적합을 구하면 해당 cluster의 대략적인 원소의 개수를 파악할 수 있을 것이다. k번째 cluster 원소의 개수를 n_k라고 하면,

$$n_k \simeq \sum_{i=1}^{M}\overline{a_k}(\boldsymbol{x_i}) = \sum_{i=1}^{M}\frac{e^{\boldsymbol{w_k^T x_i}+b_k}}{\sum_{l=1}^{K}e^{\boldsymbol{w_l^T x_i}+b_l}} \tag{7}$$

M은 모든 database와 query 이미지로부터 얻은 local descriptor의 총개수를 의미한다. 학습 과정에서 모든 database와 query 이미지의 NetVLAD descriptor를 구하는 과정에서 $\overline{a_k}(\boldsymbol{x_i})$ 를 계산하게 되므로 모든 database와 query 이미지의 local descriptor를 이용하여 각 n_k를 구할 수 있다.

이렇게 구한 n_k를 이용하여 matching에 사용할 각 clus-

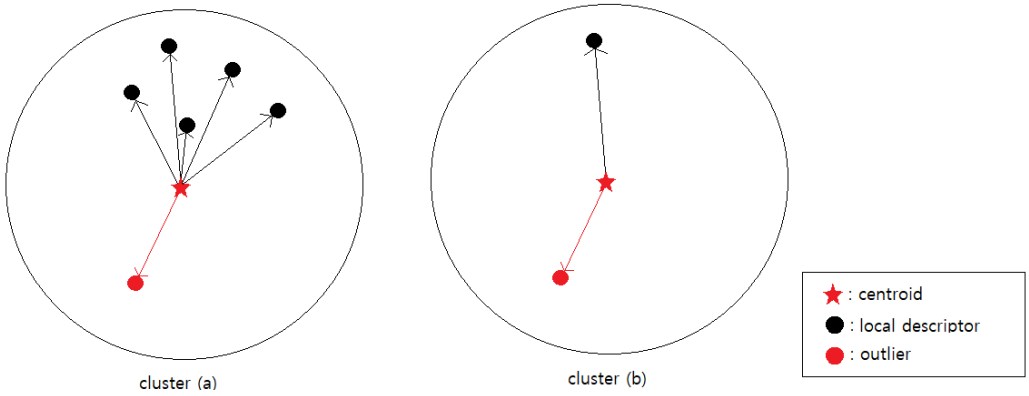

그림 5. $\boldsymbol{x_i}$ 들로 이루어진 cluster 원소의 개수에 따른 outlier의 영향
Fig. 5. Effect of Outlier on the Number of Cluster Elements Composed of $\boldsymbol{x_i}$

ter마다의 가중치 λ_k를 계산한다.

$$\lambda_k = 1 - e^{-\frac{1}{\beta} n_k} \tag{8}$$

cluster weight λ_k를 사용하는 목적은 n_k가 많이 적은 cluster의 영향력을 줄이기 위한 것이므로, n_k가 많이 적지 않은 cluster에 대해서는 거의 1에 수렴하도록 지수함수를 사용하였다. β는 database 및 query 이미지의 개수에 따라 결정하는 파라미터이다.

3. Cluster Weight를 이용한 Matching

기존에는 NetVLAD descriptor끼리의 거리를 계산할 때 유클리디안 거리를 사용하였는데, 본 논문에서는 유클리디안 거리를 구할 때 앞서 계산한 cluster weight λ_k를 사용하는 방법을 제안한다. matching 여부를 알고자 하는 두 이미지를 I_1, I_2라고 하자. I_1와 I_2의 NetVLAD descriptor를 각각 $V_{NetVLAD1}$, $V_{NetVLAD2}$ 라고 하면 이 두 벡터는 식 (9), (10)과 같다.

따라서, 일반 유클리디안 거리 $d(I_1, I_2)$는 식 (11)과 같이 구할 수 있다.

하지만 식 (11)은 outlier에 취약한 cluster를 고려하지 않았으므로 이에 cluster weight λ_k를 적용한다. 거리를 계산하는 과정에서 각 k번째 cluster에 대한 비중을 조절하기 위해, k번째 cluster로부터 얻은 원소들끼리의 계산값에 λ_k를 곱해준다. 본 논문에서 제안하는 거리 함수를 $d_{weight}(I_1, I_2)$라고 하면, $d_{weight}(I_1, I_2)$는 식 (12)와 같다.

따라서, 식 (5)에서 $d(I_1, I_2)$ (식 (11)) 대신 $d_{weight}(I_1, I_2)$ (식 (12))를 사용한다.

V. Experiment Result

본 장에서는 본 논문에서 제안한 알고리즘의 성능을 평가하기 위해서 VPR 분야에서 우수한 성능을 가지고 있는 기존 NetVLAD와의 성능을 비교한다. 공정한 비교를 위해 동일한 training set으로 처음부터 훈련을 실행하였다. 먼저, 정량적 실험 결과로 Recall 값을 제시하여 제안한 방법이 수치상으로 높은 정답률을 가진다는 것을 제시한다. 그리고, 정성적 실험 결과로 실제 두 방법을 이용하여 matching된 이미지들을 제시하고 이를 분석한다.

본 성능 평가를 위해서, 학습에 사용한 training dataset은 'Pittsburgh30k-train'[11] (query 7416장, database 10000장)이다. test dataset으로는 'Pittsburgh30k-val'[11] (query 7608장, database 10000장), 'Pittsburgh30k-test'[11] (query 6816장, database 10000장), 'Pittsburgh250k-test'[11] (query 8280장, database 83952장)을 사용하였다. learning rate = 0.0001, batch size = 4로 설정하였으며 기존 NetVLAD와 제안한 방법 모두 $margin$ = 0.1로 하였다. 제안한 방법의 β = 200000으로 설정하였다.

1. 정량적 성능 평가

표1, 표2, 표3은 각 testset에 대하여 기존 NetVLAD와 제안한 방법의 Recall을 정리한 것이다. 세 dataset 모두에서 Proposed Algorithm이 기존 NetVLAD보다 높은 Recall을 얻은 것을 확인할 수 있다.

$$V_{NetVLAD1} = (x_{11}, x_{12}, ..., x_{1D}, x_{21}, x_{22}, ..., x_{2D}, ..., x_{K1}, x_{K2}, ..., x_{KD}) \tag{9}$$

$$V_{NetVLAD2} = (y_{11}, y_{12}, ..., y_{1D}, y_{21}, y_{22}, ..., y_{2D}, ..., y_{K1}, y_{K2}, ..., y_{KD}) \tag{10}$$

$$d(I_1, I_2) = \sqrt{\{(x_{11}-y_{11})^2 + \cdots + (x_{1D}-y_{1D})^2\} + \cdots + \{(x_{K1}-y_{K1})^2 + \cdots + (x_{KD}-y_{KD})^2\}} \tag{11}$$

$$d_{weight}(I_1, I_2) = \sqrt{\lambda_1\{(x_{11}-y_{11})^2 + \cdots + (x_{1D}-y_{1D})^2\} + \cdots + \lambda_K\{(x_{K1}-y_{K1})^2 + \cdots + (x_{KD}-y_{KD})^2\}} \tag{12}$$

$$L = \sum_j l(\min_i d^2_{weight}(q, p_i^q) + margin - d^2_{weight}(q, n_j^q)) \tag{13}$$

여기서 Recall@A는 query 이미지와 matching될 후보 이미지 A개를 찾았을 때, 올바르게 matching 시켰는지에 대한 정답률을 의미한다. 예를 들어, Q개의 query 이미지에 대하여 Recall@5를 구한다고 가정한다. 하나의 query 이미지와 matching될 후보 이미지 5개 중에서 올바르게 matching되는 이미지가 있으면 정답, 5개 모두 잘못된 matching이면 오답으로 간주한다. 이렇게 Q개의 query 이미지에 대하여 정답률을 계산한 값이 Recall@5가 된다. 따라서 Recall@A는 0~1 사이의 값을 가지며, 높을수록 정답률이 높음을 의미한다.

표 1. Pittsburgh30k-val dataset에 대한 NetVLAD와 Proposed Algorithm의 Recall
Table 1. Recall of the NetVLAD and the Proposed Algorithm on Pittsburgh30k-val dataset

	Recall@1	Recall@5	Recall@10
NetVLAD	0.8475	0.9472	0.9708
Proposed Algorithm	0.8552	0.9503	0.9723

표 2. Pittsburgh30k-test dataset에 대한 NetVLAD와 Proposed Algorithm의 Recall
Table 2. Recall of the NetVLAD and the Proposed Algorithm on Pittsburgh30k-test dataset

	Recall@1	Recall@5	Recall@10
NetVLAD	0.8093	0.9074	0.9354
Proposed Algorithm	0.8206	0.9123	0.9376

표 3. Pittsburgh250k-test dataset에 대한 NetVLAD와 Proposed Algorithm의 Recall
Table 3. Recall of the NetVLAD and the Proposed Algorithm on Pittsburgh250k-test dataset

	Recall@1	Recall@5	Recall@10
NetVLAD	0.8054	0.9036	0.9268
Proposed Algorithm	0.8095	0.9062	0.9285

2. 정성적 성능 평가

다음 그림 6은 Pittsburgh30k-val dataset에서 두 방법으로 수행된 VPR 결과 중 몇 가지를 나타낸 것이다.

그림 6. Pittsburgh30k-val dataset에 대한 정성적 결과
Fig. 6. Qualitative Results on Pittsburgh30k-val dataset

8개의 query 이미지에 대하여 기존 NetVLAD와 제안하는 방법으로 matching 시킨 이미지들을 각각 제시하였다. 초록색 테두리로 표시한 것은 올바르게 matching을 수행한 경우이며, 반대로 빨간색 테두리로 표시한 것은 잘못된

matching을 수행한 경우이다. 먼저, (a)를 보면, 두 방법 모두 query 이미지와 올바르게 matching되는 이미지를 찾았다. 반면 (b) ~ (g)를 보면 기존 NetVLAD는 잘못된 이미지를 찾았고, 제안하는 방법은 올바른 이미지를 찾은 것을 확인할 수 있다. (b)부터 자세히 살펴보면, query 이미지에 강한 햇빛이 있는 것을 확인할 수 있다. 이 때문에 기존 NetVLAD에서는 마찬가지로 강한 햇빛이 있는 사진을 찾았다. 실제 건물과 간판이 일치하지 않지만 강한 햇빛의 영향으로 인하여 잘못된 이미지를 matching시켰다. 제안하는 방법으로 찾은 이미지를 보면, 햇빛은 없지만 건물이 일치하는 사진을 찾았다. 즉, 탐색 과정에서 햇빛의 비중을 적게 두고 건물과 간판 등에 비중을 크게 두었기 때문에 올바른 matching을 수행하였다는 것을 알 수 있다. (c)에서 query 이미지를 보면 하늘이 맑고, 이에 따라 사진에 찍힌 건물들도 밝은색을 보인다. 기존 NetVLAD에서는 건물의 구도와 색상이 매우 유사한 사진을 찾아냈으나, query 이미지의 건물이 아닌 다른 건물을 찾았다. 반면에 제안하는 방법에서는 날씨가 흐리고, 건물색이 비교적 탁하게 나타났으나 query 이미지와 같은 장소를 올바르게 찾아내었다. 장소 및 건물은 같으나 햇빛이나 날씨 등의 환경 변화로 인해 잘못된 이미지를 찾은 기존 NetVLAD와는 달리, 제안하는 방법에서는 이러한 환경 변화에 더 강인하여 올바른 이미지를 찾은 것을 확인할 수 있다. (d)에서도 기존 NetVLAD는 잘못된 matching을 하였는데, 제안한 방법으로 찾은 올바르게 matching된 이미지와 query를 비교해보면 건물의 간판이 사라졌고 사람들이 지나가고 있는 것을 확인할 수 있다. (e)의 query에서 제법 큰 차가 건물을 가리고 있는데, 이 경우에도 제안하는 방법에서는 올바른 matching을 수행하였다. (f)나 (g) 역시 날씨가 조금 다르고 차와 사람들이 많이 촬영되었음에도 불구하고 제안한 방법에서 올바르게 matching한 것을 확인할 수 있다. 하지만 (h)에서는 기존 NetVLAD와 제안하는 방법에서 모두 잘못된 이미지를 찾았다. query 이미지와 두 방법에서 찾은 이미지들을 보면 모두 비슷한 패턴의 건물, 신호등이 있고 날씨 또한 비슷해 보인다. 제안하는 방법에서 환경 변화에는 더 강인해졌지만, 환경마저도 비슷한 상황에서 세밀하게 올바른 이미지를 찾는 능력은 아직 부족하다는 것을 확인할 수 있다.

그림 7의 그래프는 그림 6에서 (b)의 'query 이미지의 NetVLAD descriptor'와, 'Proposed Algorithm이 찾은 정답 이미지의 NetVLAD descriptor' 사이의 거리를 분석한 것이다. 이 실험에서는 $K = 64$로 설정하였고, 1번 cluster로부터 얻은 원소부터 64번 cluster로부터 얻은 원소에서의 거리를 막대 그래프로 표시하였다. 파란색과 주황색을 더한 거리값은 cluster weight를 적용하지 않고 기존의 거리함수

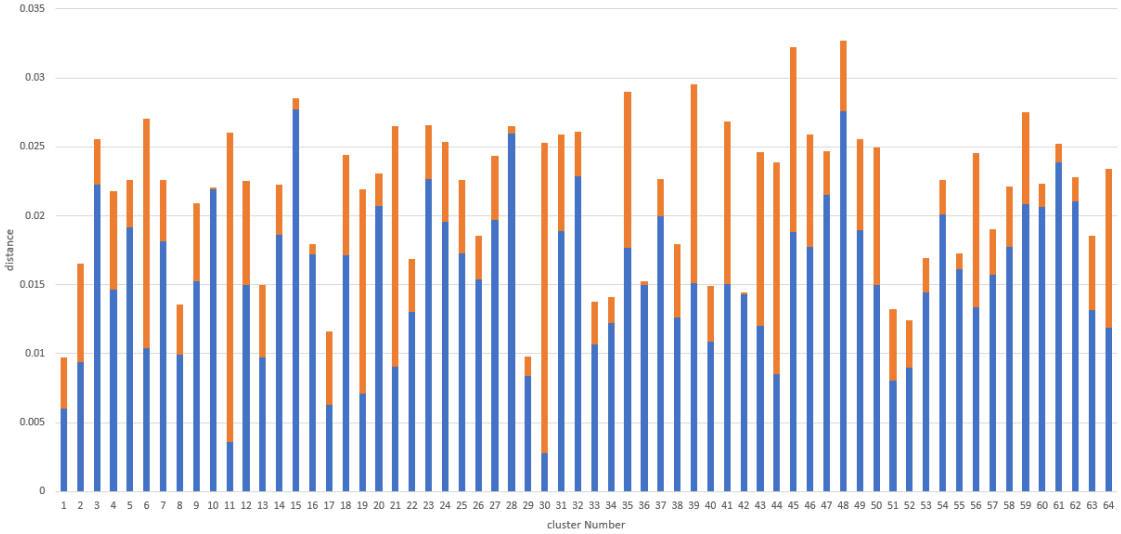

그림 7. 정성적 결과의 (b)에 대한 추가 분석
Fig. 7. Additional Analysis of (b) of Qualitative Results

인 식 (11)로 계산한 것이고, 파란색은 제안하는 방법의 거리함수인 식 (12)를 사용하여 계산한 것이다. (따라서 주황색은 두 방법으로 구한 거리값의 차이를 의미한다.) 식 (12)는 식 (11)에서 각 원소별 거리값에 0~1사이의 값을 가지는 cluster weight λ_k를 곱한 것이므로 기본적으로 거리값들이 전부 작아진다.

두 이미지는 matching되어야 하는 이미지이므로 거리값이 매우 낮아야 하지만, 시점·햇빛·조명·사람 등의 환경 변화가 존재하기 때문에 두 이미지의 NetVLAD descriptor간에는 차이가 존재한다. 이 거리값을 0으로 만드는 것은 불가능하므로 본 논문에서는 cluster 별로 weight를 부여하여 차이가 특히 큰 원소(환경의 영향을 크게 받는 원소)의 비중을 줄이고자 하였다. 그래프를 보면, 16개의 cluster(6번, 11번, 12번, 19번, 21번, 30번, 35번, 39번, 41번, 43번, 44번, 45번, 46번, 50번, 56번, 64번)들은 기존 거리값이 높았지만 cluster weight를 적용한 후에 거리값이 크게 줄어들었다. 반면 6개의 cluster(10번, 15번, 28번, 60번, 61번, 62번)들은 기존 거리값이 높았음에도 거리값이 줄어들지 않았다. cluster weight를 계산하는 과정에서 outlier들을 직접 골라내지는 못했기 때문에 이처럼 환경 변화의 영향을 받는 모든 cluster를 골라내기는 어렵다는 한계점이 존재한다. 하지만 제법 많은 수의 '차이값이 컸던 원소들'의 비중이 줄어든 것을 확인할 수 있고, 정량적 실험 결과에서 알 수 있듯이 전체적인 정확도가 개선되는 결과로 이어졌다.

3. 실험 결과에 대한 분석

정량적 결과를 보면 세 dataset 모두, 모든 Recall에 대하여 제안하는 방법이 더 성능이 좋은 것을 알 수 있다. 즉, VPR 작업에서 '제안하는 방법으로 생성된 global descriptor를 사용한 matching'이 '기존 NetVLAD descriptor를 사용한 matching'보다 정확하다는 것을 의미한다. 즉, 제안하는 방법이 기존 방법에 비해 '환경 변화로 인한 잘못된 matching' 횟수가 감소하였음을 알 수 있다.

또한, 정성적 결과를 보면 그림 6의 (b) ~ (g)와 같이 기존 NetVLAD에서는 햇빛, 날씨, 사람 등의 환경 변화로 인하여 잘못된 이미지를 찾는 경우가 많은 것을 확인할 수 있다. 반면에 제안하는 방법에서는 이러한 환경 변화에 대하여 영향을 적게 받기 때문에 올바른 이미지들을 찾을 수 있었다. 하지만 간접적인 방법을 사용하기 때문에 환경의 영향을 제거하는 것에 한계가 있고, 그림 6의 (h)와 같이 환경마저도 비슷한 이미지들에 대하여 더욱 정밀한 matching을 수행하지는 못한다는 문제점이 있다.

Ⅵ. 결 론

기존의 NetVLAD는 Neural Net을 통해 local descriptor 생성 및 NetVLAD descriptor를 학습 가능한 형태로 구성하기에 성공했으나, 환경 변화에 따른 outlier들을 따로 고려해주지 않았다. 본 논문에서는 global descriptor의 matching 과정에서 outlier에 취약한 부분들을 고려함으로써 보다 환경 변화에 강인한 학습을 유도하였고, 기존 NetVLAD 보다 높은 정확도를 얻는 것을 정량적·정성적인 실험 결과를 통하여 확인하였다. 하지만, outlier를 직접 추적하지 않고 outlier에 취약한 cluster에 가중치를 주는 간접적인 방식을 사용하였기 때문에 정확도가 큰 폭으로 증가하지는 않았다. 또한, 촬영 환경마저도 비슷한 상황에서의 정밀한 matching과 관련된 개선점은 없었다. 따라서, 앞으로 outlier의 특성을 더 구체화하여 더욱 직접적인 방식으로 outlier의 영향을 제거하는 방법을 연구해야 할 필요성이 있고, 더욱 정밀하게 matching을 수행하는 방법을 고안할 것이다.

참 고 문 헌 (References)

[1] Y. Cai, J. Zhao, J. Cui, F. Zhang, T. Feng and C. Ye, "Patch-NetVLAD+: Learned patch descriptor and weighted matching strategy for place recognition," *2022 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Bedford, United Kingdom, pp. 1-8, 2022.
doi: https://doi.org/10.1109/MFI55806.2022.9913860

[2] S. Hausler, S. Garg, M. Xu, M. Milford and T. Fischer, "Patch-NetVLAD: Multi-Scale Fusion of Locally-Global Descriptors for Place Recognition," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, pp. 14136-14147, 2021,
doi: https://doi.org/10.1109/CVPR46437.2021.01392

[3] Csurka, Gabriella, et al. "Visual categorization with bags of keypoints." *Workshop on statistical learning in computer vision, ECCV*. Vol. 1.

No. 1-22. 2004.
doi: https://www.researchgate.net/publication/228602850

[4] R. Arandjelovic and A. Zisserman, "All About VLAD," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, pp. 1578-1585, 2013.
doi: https://doi.org/10.1109/CVPR.2013.207

[5] H. Jégou, M. Douze, C. Schmid and P. Pérez, "Aggregating local descriptors into a compact image representation," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, pp. 3304-3311, 2010.
doi: https://doi.org/10.1109/CVPR.2010.5540039

[6] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1437-1451, 1 June 2018.
doi: https://doi.org/10.1109/TPAMI.2017.2711011

[7] 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: https://doi.org/10.1109/ICCV.1999.790410

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

[9] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", *NIPS*, pp. 1106-1114, 2012,
doi: https://dl.acm.org/doi/10.5555/2999134.2999257

[10] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 770-778, 2016.
doi: https://doi.org/10.1109/CVPR.2016.90

[11] A. Torii, J. Sivic, T. Pajdla and M. Okutomi, "Visual Place Recognition with Repetitive Structures," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, pp. 883-890, 2013.
doi: https://doi.org/10.1109/CVPR.2013.119

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 : 비디오 코덱, 영상 신호처리, 정보 압축, 방송 시스템